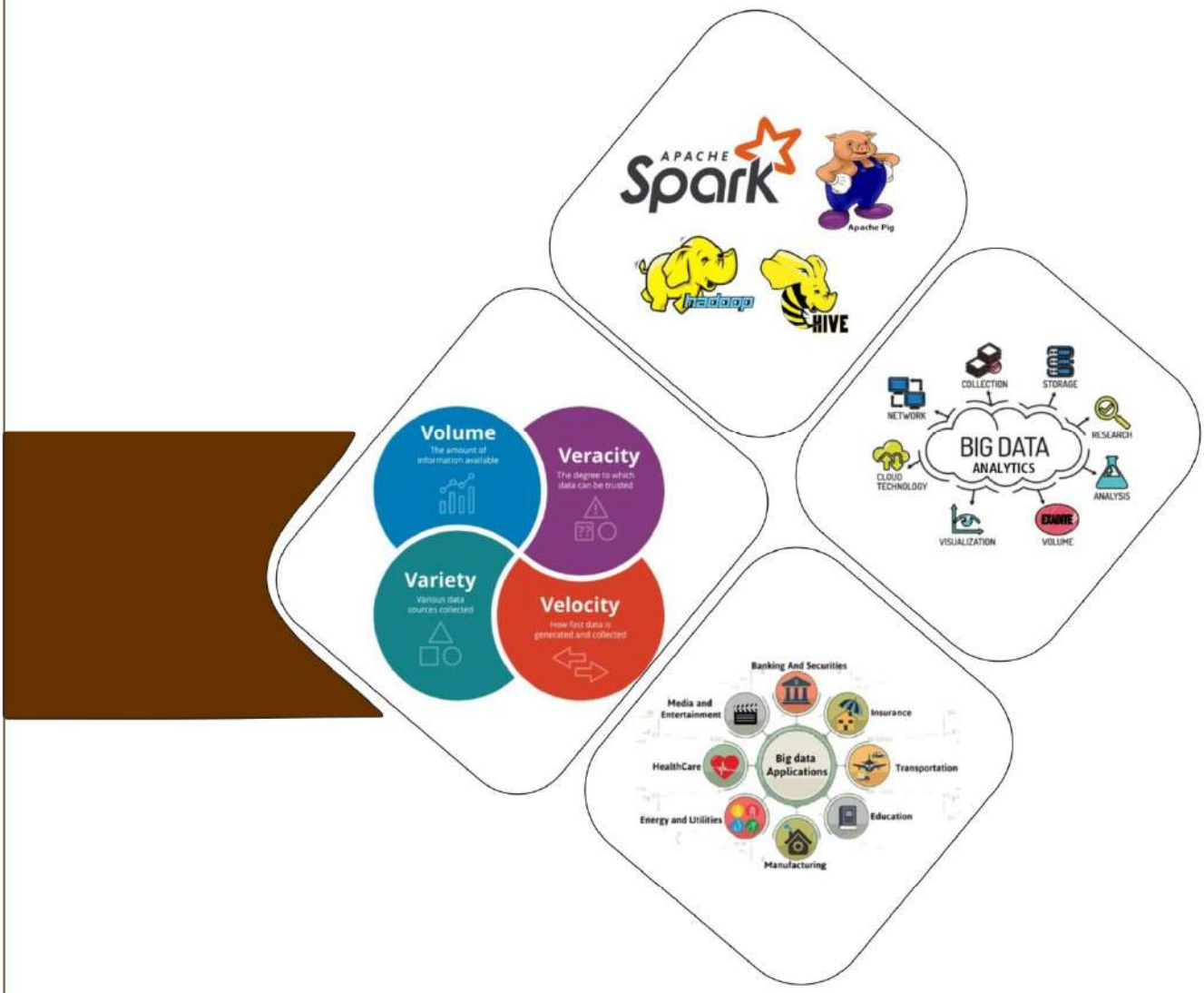
SCHEME :K

Name :_____
Roll No.:_____ Year : 20 ___ 20 __
Exam Seat No. :_____

LABORATORY MANUAL FOR

BIG DATA ANALYTICS (316318)



COMPUTER ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous)(ISO21001:2018)(ISO/IEC27001:2013)

Vision

To ensure that the Diploma level Technical Education constantly matches the latest requirements of Technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

Mission

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the challenging technological & environmental challenges.

Quality Policy

We, at MSBTE are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation,valuation and monitoring system along with adequate faculty development programmes.

Core Values

MSBTE believes in the following:

- Skill development in line with industry requirements
- Industry readiness and improved employability of Diploma holders
- Synergistic relationship with industry
- Collective and Cooperative development of all stake holders
- Technological interventions in societal development
- Access to uniform quality technical education

A Laboratory Manual

For

Big Data Analytics

(316318)

Semester-VI

(AI /AN /DS)

Maharashtra State Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018) (ISO/IEC 27001:2013)

'K' Scheme Curriculum

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

Address: 4th floor, Govt. Polytechnic Building, 49,

Kherwadi, Bandra (E), Mumbai- 400 051

Tel: 022 62542100

Email: secretary@msbte.com

Maharashtra State Board of Technical Education

CERTIFICATE

This is to certify that Mr./Ms. ……Roll No……. of the **Sixth Semester** of Diploma in …… Engineering/Technology (Program Code - .....6K) of the Institute …… (Inst. Code……) has completed the practical work satisfactorily for the course **Big Data Analytics (Course Code: 316318)** for the academic year 20…– 20…..... as prescribed in the curriculum.

Place ……. Enrollment No……

Date: …..... Exam Seat No. …….....

Course Teacher **Head of Department** **Principal**

Seal of the Institute

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much-needed industry relevant competencies and skills. Therefore, for the successful implementation of this curriculum, every practical has been designed to serve as a 'vehicle' to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the 'K' scheme laboratory manual development team designed the practicals to focus on outcomes, rather than the traditional age-old practice of conducting practical's to 'verify the theory' (which may become a by-product along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read the concerned practical procedure that they will do the next day and understand minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. Students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

The Big Data Analytics course provides an introduction to the fundamental concepts of handling large and diverse datasets, including data classification, architecture, and processing techniques.It covers core components and practical skills using basic Big Data tools like Hadoop, NoSQL, Hive, Pig, and Spark. The course aims to help learners manage Big Data efficiently and support data handling tasks in various contexts, preparing them to apply big data analytics to manage and analyse large datasets.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome.

Program Outcomes (POs) to be achieved through Course:

PO1	Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
PO2	Problem analysis: Identify and analyses well-defined engineering problems using codified standard methods.
PO3	Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
PO4	Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
PO5	Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.
PO6	Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
PO7	Life-long learning: Ability to analyses individual needs and engage in updating in the context of technological changes.

List of Relevant Skills

The following industry relevant skills of the competency “Apply Big Data Analytics Concepts” are expected to be developed in you by performing practicals of this laboratory manual and they will be able to:

1. Classify data and illustrate the characteristics big data.
2. Set up and configure the Hadoop environment and its core componentsfor big data processing.
3. Apply NoSQL database concepts and architecture patterns manage big data.
4. Use Hive for data definition, manipulation, and querying with HiveQL.
5. Use Pig for data transformation by writing and executing Pig scripts.
6. Use Spark to process and analyze big data in real-time or archives.
7. Apply machine learning algorithms.

Practical Course Outcome Matrix

Course Outcomes (COs)

CO1	Illustrate different phases of big data with respect to real world application.
CO2	Demonstrate the use of Hadoop core components for big data processing.
CO3	Apply NoSQL database concepts and architecture patterns to manage big data.
CO4	Use Hive and Pig for data processing and transformation within big data environments.
CO5	Use Spark to process and analyze big data in real-time or archives.

Sr. No.	Title of the Experiment	CO1	CO2	CO3	CO4	CO5
1	*Conduct a Case Study on Big Data and BigData Analysis	✓	--	--	--	--
2	*Install Hadoop Ecosystem on Local Cluster	--	✓	--	--	--
3	*Configure Hadoop Ecosystem on Local Cluster	--	✓	--	--	--
4	Install Hadoop on multiple nodes	--	✓	--	--	--
5	Configure Multi-Node Hadoop Cluster	--	✓	--	--	--
6	Use Monitoring tools to Observe ClusterResources	--	✓	--	--	--
7	*Perform Basic File Operation using HDFS	--	✓	--	--	--
8	Perform Backup and Restore of Datasets inHDFS	--	✓	--	--	--
9	*Execute WordCount MapReduce Program	--	✓	--	--	--
10	*Process Large CSV Dataset Using MapReduce	--	✓	--	--	--
11	*Install NoSQL Database (MongoDB) andCreate Collections	--	--	✓	--	--
12	*Create a schema for unstructured data inMongoDB	--	--	✓	--	--
13	*Run basic aggregation queries on Unstructureddataset in MongoDB	--	--	✓	--	--
14	Perform Basic MongoDB OperationsDemonstrating CAP Theorem Behavior	--	--	✓	--	--
15	*Install and Configure Hive	--	--	--	✓	--
16	*Execute Data Queries Using HiveQL	--	--	--	✓	--

17	Run Pig Scripts for data transformation	--	--	--	✓	--
18	*Execute User Defined Functions (UDF) in Pigfor Data Normalization	--	--	--	✓	--
19	*Install Spark on Cluster environment and verifyinstallation with any dataset	--	--	--	--	✓
20	*Perform Data Transformations with RDDs andapply map, filter, reduce operations	--	--	--	--	✓
21	*Perform ETL Process on Big Data Using Spark	--	--	--	--	✓
22	Load Structured Dataset and Create TemporaryViews in Spark SQL	--	--	--	--	✓
23	Perform Select, Join, and Group By Queries inSpark SQL	--	--	--	--	✓
24	Perform Real-Time Word Count Using SparkStreaming	--	--	--	--	✓
25	Store Real-Time Streamed Data from Sparkstreaming into HDFS	--	--	--	--	✓
26	*Apply Simple Regression on Large Dataset inSpark	--	--	--	--	✓
27	*Apply K-Means Clustering on any Datasetusing Spark MLlib	--	--	--	--	✓
28	*Visualize K-Means Clustering Results UsingSpark MLlib	--	--	--	--	✓
29	Apply Decision Tree Classification on anyDataset using Spark MLlib	--	--	--	--	✓
30	Display Classification Results from DecisionTree Model in Spark MLlib	--	--	--	--	✓

Guidelines to Teachers

1. Teachers should align the explanation of the topic to teaching learning outcome (TLOs).
2. Refer to laboratory learning outcome (LLOs) for the execution of the practical to focus on the defined objectives.
3. Promote life-long learning by training the students to equip themselves with essential knowledge, skills and attitudes.
4. If required, provide demonstration for the practical emphasizing on the skills that the student should achieve.
5. Teachers should give opportunity to the students for exhibiting their skills after the demonstration.
6. Provide feedback and/or suggestions and share insights to improve effectiveness.
7. Assess students' skill achievement related to COs of each unit.

Instructions for Students

1. 100% attendance is compulsory for all practical sessions.
2. Students must adhere to ethical practices.
3. All the students must follow the schedule of practical sessions, complete the assigned work/activity and submit the assignment in stipulated time as instructed by the course teacher.
4. Students shall listen carefully the lecture given by teacher about importance of subject, learning structure, course outcomes.
5. Students shall understand the purpose of experiment and its practical implementation.
6. Students shall write the answers of the questions during practical.
7. Student should feel free to discuss any difficulty faced during the conduct of practical.
8. Students shall develop web based and window-based applications an expected by the industries.
9. Student shall attempt to develop related hands-on skills and gain confidence.
10. Students should develop habit to submit the write-ups on the scheduled dates and time.

Content Page

List of Practical and Formative Assessment Sheet

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (25)	Teacher's Sign	Remark
1	*Conduct a Case Study on Big Data and Big Data Analysis					
2	*Install Hadoop Ecosystem on Local Cluster					
3	*Configure Hadoop Ecosystem on Local Cluster					
4	Install Hadoop on multiple nodes					
5	Configure Multi-Node Hadoop Cluster					
6	Use Monitoring tools to Observe Cluster Resources					
7	*Perform Basic File Operation using HDFS					
8	Perform Backup and Restore of Datasets inHDFS					
9	*Execute WordCount MapReduce Program					
10	*Process Large CSV Dataset Using MapReduce					
11	*Install NoSQL Database (MongoDB) andCreate Collections					
12	*Create a schema for unstructured data inMongoDB					
13	*Run basic aggregation queries on Unstructureddataset in MongoDB					
14	Perform Basic MongoDB OperationsDemonstrating CAP Theorem Behavior					
15	*Install and Configure Hive					
16	*Execute Data Queries Using HiveQL					
17	Run Pig Scripts for data transformation					
18	*Execute User Defined Functions (UDF) in Pigfor Data Normalization					
19	*Install Spark on Cluster environment and verifyinstallation with any dataset					

20	*Perform Data Transformations with RDDs andapply map, filter, reduce operations					
21	*Perform ETL Process on Big Data Using Spark					
22	Load Structured Dataset and Create TemporaryViews in Spark SQL					
23	Perform Select, Join, and Group By Queries inSpark SQL					
24	Perform Real-Time Word Count Using SparkStreaming					
25	Store Real-Time Streamed Data from Sparkstreaming into HDFS					
26	*Apply Simple Regression on Large Dataset inSpark					
27	*Apply K-Means Clustering on any Datasetusing Spark MLlib					
28	*Visualize K-Means Clustering Results UsingSpark MLlib					
29	Apply Decision Tree Classification on anyDataset using Spark MLlib					
30	Display Classification Results from DecisionTree Model in Spark MLlib					
				Total		

***Total marks to be transferred to proforma published by MSBTE**

Note:

'*' Marked Practical's (LLOs) Are mandatory.

- Minimum 80% of above list of lab experiment are to be performed.

Judicial mix of LLOs are to be performed to achieve desired outcomes.

Practical No. 1: *Conduct a Case Study on Big Data and Big Data Analysis

I Practical Significance

This practical helps student understand how Big Data is collected, stored, processed, and analyzed in real-world applications. It enables learners to apply analytical techniques on large datasets for meaningful insights.

II Industry / Employer Expected Outcome(s)

Employers expect students to understand Big Data workflows, tools, and technologies and be capable of analyzing large datasets for decision-making. Students should also be able to interpret results and prepare concise case study reports.

III Course Level Learning Outcomes(s)

CO1 - Illustrate different phases of big data with respect to real world application.

IV Laboratory Learning Outcome(s)

LLO1.1: Identify Big Data use cases and explain the analytics process applied.

V Relevant Affective Domain related Outcomes

Students develop curiosity and appreciation for data-driven decision-making in modern industries.

VI Relevant Theoretical Background

Introduction:

Big Data refers to extremely large datasets that cannot be processed using traditional data processing tools. Big Data analytics involves collecting, storing, cleaning, processing, and analyzing data to extract useful patterns and insights.

Characteristics of Big Data:

- **Volume –** massive datasets
- **Velocity –** high speed of data generation
- **Variety –** structured, unstructured & semi-structured data
- **Veracity –** data uncertainty

Big Data Analytics Process:

A typical Big Data case study involves the following steps:

Step-by-step Practical Explanation:

Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL).

Step 1: Identify a Real-World Problem (Case Study Selection)

Choose any industry-based scenario where Big Data is used, such as:

- E-commerce customer behavior analysis
- Social media sentiment analysis
- Banking fraud detection
- Healthcare patient data prediction
- Smart city traffic analysis

Step 2: Dataset Identification

Select a publicly available large dataset (CSV/JSON).

Examples:

- Kaggle datasets
- UCI Machine Learning Repository
- Government open data portals

Commands:

```
mkdir dataset
cd dataset
wget https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data -O dataset.csv
cd ..
sudo apt install python3.12-venv
python3 -m venv ml_env
source ml_env/bin/activate
```

Step 3: Data Import

Load the dataset using tools like Python, Jupyter Notebook, Hadoop, or Spark.

Example (Python):

Please install the necessary Python libraries using the pip package manager. You can do this with a single command:>pip install pandas matplotlib sklearn seaborn

```
import pandas as pd

df = pd.read_csv("dataset/dataset.csv")
print("data loaded successfully ")
print(df.head())
```

Step 4: Data Pre-processing

Perform cleaning and transformation:

- Remove missing values
- Normalize data
- Label encoding (if required)
- Remove duplicates

Example (Python):

```
import pandas as pd
df = pd.read_csv("dataset/dataset.csv")
df = df.dropna()
df = df.drop_duplicates()
print("After cleaning")
print(df.info())
```

Step 5: **Exploratory Data Analysis (EDA)**

Perform statistical and visual analysis:

- Mean, median, mode
- Correlation
- Graphs: bar chart, histogram, scatter plots

Example (Python):

```
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt

df = pd.read_csv("dataset/dataset.csv", header=None)
df.columns = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
numeric_df = df.select_dtypes(include=['float64', 'int64'])
print("Statistical Summary:")
print(numeric_df.describe())

print("\nCorrelation Matrix:")
print(numeric_df.corr())
sns.pairplot(df, hue="class")
plt.savefig("eda_pairplot.png")
print("\nEDA completed. Plot saved as eda_pairplot.png")
```

Step 6: **Apply Big Data Analytical Method**

Any one technique may be used:

- Clustering (K-means)
- Classification
- Prediction/Regression
- Sentiment Analysis (NLP)
- Association Rule Mining

Example (K-Means):

```
import pandas as pd
from sklearn.cluster import KMeans

df = pd.read_csv("dataset/dataset.csv")
df_numeric = df.select_dtypes(include=['float64', 'int64'])
kmeans = KMeans(n_clusters=3, n_init='auto')
kmeans.fit(df_numeric)

print("Cluster Centers:")
print(kmeans.cluster_centers_)
print("Labels:")
print(kmeans.labels_)
```

Step 7: Interpretation of Results

Explain the patterns or insights obtained, such as:

- Customer segments
- Fraud likelihood
- Sentiment score trends
- Prediction accuracy

Step 8: **Conclusion of the Case Study**

Summarize the findings, challenges, and improvements for future work.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity
1	Computer System	Minimum 8 GB RAM,250 GB Hard disk, Quad Core Processor with Stable internet connection	As per Batch Size
2	Python and Jupyter Notebook		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Select any dataset from Kaggle and perform EDA to identify at least five insights.
2. Apply any one machine learning model (clustering/classification/regression) and analyze the model performance.
3. Prepare a case study report describing the problem, dataset, tools used, analysis performed, and final outcomes
4. Explain the main characteristics of Big Data and their significance
5. Explain the steps involved in performing data pre-processing before Big Data analysis.
6. Describe the role of exploratory data analysis in extracting insights from large datasets
7. List the common challenges faced while working with Big Data.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

X References:

1. https://www.kaggle.com (Datasets & notebooks)
2. https://hadoop.apache.org (Hadoop framework)
3. https://hadoop.apache.org (Hadoop framework)
4. https://hadoop.apache.org (Hadoop framework)
5. Assignment 1 -15 demo :
https://computermsbte.blogspot.com/2025/12/big-data-analytics-practical-series.html

Scan QR code

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 2: *Install Hadoop Ecosystem on Local Cluster.

I Practical Significance

This practical enables student to understand the installation, configuration, and functioning of the Hadoop ecosystem in a local cluster environment. It provides hands-on experience in setting up distributed storage and processing frameworks.

II Industry / Employer Expected Outcome(s)

Employers expect students to install, configure, and manage Hadoop environments independently. Students should be able to deploy Big Data tools and troubleshoot cluster-related issues.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 2.1: Setup a Hadoop ecosystem on a local cluster.

V Relevant Affective Domain related Outcomes

Students develop responsibility and accuracy while handling system configurations. They demonstrate a positive attitude toward learning open-source tools and maintaining distributed systems.

VI Relevant Theoretical Background

Introduction to Hadoop Ecosystem

Hadoop is an open-source Big Data framework that provides distributed storage using HDFS (Hadoop Distributed File System) and distributed processing using MapReduce and YARN. A local cluster (pseudo-distributed mode) allows Hadoop services (NameNode, DataNode, ResourceManager, NodeManager) to run on a single machine for learning and testing.

Installation Procedure (Step-by-Step)
Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL).
Step 1: **Update System Packages**
sudo apt update
sudo apt upgrade -y

Step 2: **Install Java (Required by Hadoop)**

sudo apt install openjdk-8-jdk -y
java -version

Step 3: **Create a Dedicated Hadoop User**

sudo adduser hadoop
sudo usermod -aG sudo hadoop
su - hadoop

Step 4: **Configure SSH for Passwordless Login**

Generate SSH keys:

ssh-keygen -t rsa -P ""

Add public key:

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

Set Permissions: # suggested

chmod 600 ~/.ssh/authorized_keys

Start SSH service

sudo service ssh start

Test SSH:

ssh localhost

Extra Commands

sudo apt update
sudo apt install -y openssh-server

# enable and start the service now
sudo systemctl enable --now ssh

# quick status check
sudo systemctl status ssh --no-pager

Step 5: **Download and Install Hadoop**

wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
tar -xzf hadoop-3.3.6.tar.gz
sudo mv hadoop-3.3.6 /usr/local/hadoop
sudo chown -R hadoop:hadoop /usr/local/hadoop

Step 6: **Set Hadoop Environment Variables**

Edit .bashrc:

nano ~/.bashrc

Add:

export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bi

For WSL:
nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
Add:
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Reload:

source ~/.bashrc

Step 7: **Configure Hadoop Core Files**

Edit core-site.xml
File Path: $HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://machine_ip_address:9000</value>
  </property>
</configuration>
```

Edit hdfs-site.xml
File Path: $HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/data/nn</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/data/dn</value>
  </property>
</configuration>
```

Edit mapred-site.xml
File Path: $HADOOP_HOME/etc/hadoop/mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Edit yarn-site.xml
File Path: $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

Step 8: Format NameNode

hdfs namenode -format

Step 9: **Start Hadoop Services**

start-dfs.sh
start-yarn.sh

Step 10: **Verify Services**
jps
Expected services:

- NameNode
- DataNode
- ResourceManager
- NodeManager
- SecondaryNameNode

Step 11: **Access Web Interfaces**

Service	**URL**
NameNode	http://localhost:9870
ResourceManager	http://localhost:8088

Step 12: Test HDFS Commands

hdfs dfs -mkdir /demo
hdfs dfs -put sample.txt /demo
hdfs dfs -ls /demo

VII Recourses Required:

Sr. No.	**Name of Resource**	**Broad Specification**	**Quantity**	**Remarks**
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk, Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service		

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Explain the role of NameNode and DataNode in Hadoop architecture
2. Describe the need for passwordless SSH in a Hadoop cluster environment
3. Explain the purpose of core-site.xml and hdfs-site.xml configuration files.
4. Install Hadoop and after installing Hadoop, create a directory in HDFS, upload any file, and

list the directory contents.

5. Prepare a report explaining the Hadoop setup process with screenshots.

Space for Answer

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

X References:

1. https://hadoop.apache.org
2. https://data-flair.training/blogs/hadoop-tutorial
3. Install Hadoop https://www.youtube.com/watch?v=ncGRUOs7M0g
4. Install Hadoop using VirtualBox https://www.youtube.com/watch?v=R_kid1CHuOw

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 3: *Configure Hadoop Ecosystem on Local Cluster.

I Practical Significance

This practical helps students understand the configuration of Hadoop's core components and enables them to run a functional Hadoop environment on a local cluster. It provides hands-on experience with HDFS and essential Hadoop configuration files.

II Industry / Employer Expected Outcome(s)

Employers expect students to configure Hadoop clusters, manage Hadoop services, and use HDFS for distributed storage. Students should be able to set up, troubleshoot, and handle data operations in a Hadoop ecosystem.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 3.1: Configure Hadoop Ecosystem on Local Cluster.

V Relevant Affective Domain related Outcomes

Students develop confidence and responsibility while working with system-level configurations. They demonstrate patience, accuracy, and respect for proper procedures in configuring distributed systems.

VI Relevant Theoretical Background

Introduction

The Hadoop ecosystem uses several XML configuration files to define cluster behavior. In a local (pseudo-distributed) setup, Hadoop runs multiple daemons NameNode, DataNode, ResourceManager, and NodeManager on a single machine. Configuring these files correctly is essential for HDFS and MapReduce to work.

Purpose of Configuration Files

- **core-site.xml**: Defines the NameNode URI and Hadoop's default filesystem.
- **hdfs-site.xml**: Contains settings for HDFS storage directories, replication factor, etc.
- **mapred-site.xml**: Defines MapReduce framework settings (usually set to YARN).

Practical Steps

Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL).

Step 1: Configure core-site.xml

File Path: $HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://machine_ip_address:9000</value>
  </property>
</configuration>
```

This tells Hadoop the address of the NameNode and default filesystem protocol.

Step 2: Configure hdfs-site.xml
File Path:$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/data/nn</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/data/dn</value>
  </property>
</configuration>
```

This defines:

- One replica
- Local storage directory for NameNode & DataNode

Step 3: Configure mapred-site.xml

Copy template file if needed:
it can throw error, not available in some distributions

```
cp $HADOOP_HOME/etc/hadoop/mapred-site.xml.template \
   $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

File Path:$HADOOP_HOME/etc/hadoop/mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

This tells Hadoop to run MapReduce using YARN.

Step 4: Format NameNode (run once)

```
hdfs namenode -format
```

Step 5: Start HDFS daemons

Run Hadoop services:

```
start-dfs.sh
```

Verify using:

```
jps
```

Expected daemons:

- NameNode
- DataNode

- SecondaryNameNode

Step 6: Upload any dataset to HDFS

Create an HDFS directory:
You should have a sample dataset in same directory
hdfs dfs -mkdir /dataset

Upload file to HDFS:
hdfs dfs -put sample.csv /dataset

Verify:
hdfs dfs -ls /dataset
If the file lists correctly, the dataset is successfully uploaded.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Modify the replication factor from 1 to 3 and observe the behavior in a multi-node simulation.
2. Upload three different datasets to HDFS and prepare a report showing directory structure and file permissions.
3. Configure an additional Hadoop component (like YARN) and verify its daemon processes.
4. Explain the significance of core-site.xml in Hadoop configuration
5. Why is dfs.replication set to 1 in a single-node Hadoop cluster?

Space for Answer

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://hadoop.apache.org
2. https://www.digitalocean.com/community/tutorials/install-hadoop-ubuntu
3. https://data-flair.training/blogs/hadoop-tutorial
4. https://www.youtube.com/watch?v=YLtf2WlWcWI&t=646s

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 4: Install Hadoop on Multiple Nodes (Multi-Node Cluster)

I Practical Significance

This practical introduces students to the installation and setup of a distributed Hadoop cluster using multiple nodes. It demonstrates how real-world big data clusters operate in industry using master–slave architecture.

II Industry / Employer Expected Outcome(s)

Employers expect students to configure, deploy, and maintain multi-node Hadoop clusters for distributed storage and processing. Students should understand cluster roles, node communication, and basic troubleshooting.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 4.1: Setup multi-node hadoop cluster.

V Relevant Affective Domain related Outcomes

Students develop teamwork, accuracy, and responsibility while configuring multiple systems. They develop confidence in handling distributed environments and demonstrate patience when resolving configuration errors.

VI Relevant Theoretical Background

Introduction to Multi-Node Hadoop Cluster

A multi-node Hadoop cluster consists of:

- **Master Node (NameNode + ResourceManager)**
- **Slave Nodes (DataNode + NodeManager)**

The master manages metadata and job scheduling, while slaves store data blocks and execute processing tasks.

Cluster Architecture Overview

- **NameNode:** Controls filesystem namespace
- **DataNodes:** Store data blocks in HDFS
- **ResourceManager:** Manages cluster resources
- **NodeManagers:** Execute tasks on worker nodes

Practical Steps for Multi-Node Hadoop Installation

The procedure is broken down into the four parts: Installation, Shell Initiation, Data Loading, and Verification. Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL). Use two machine

Step 1: Prepare All Machines

Perform these on all nodes (master + workers).

Update system:
sudo apt update && sudo apt upgrade -y

Install Java:
sudo apt install openjdk-8-jdk -y

Create Hadoop user:
sudo adduser hadoop
sudo usermod -aG sudo hadoop

Switch to Hadoop user:
su –Hadoop

Step 2:Configure SSH for Passwordless Communication

Generate SSH key (on master node only):
ssh-keygen -t rsa -P ""

Enable Password Authentication in worker (if Applicable):
Login to your worker
sudo nano /etc/ssh/sshd_config
Find these:
PasswordAuthentication no
PubkeyAuthentication yes
Change to:
PasswordAuthentication yes
PubkeyAuthentication yes
Restart:
sudo systemctl restart ssh

Copy key to all nodes:
ssh-copy-id hadoop@master_ip_address
ssh-copy-id hadoop@worker1_ip_address
ssh-copy-id hadoop@worker2_ip_address

Test:
ssh hadoop@worker1_ip_address
ssh hadoop@worker2_ip_address

Step 3: Install Hadoop on All Nodes

Download and extract Hadoop:
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
tar -xzf hadoop-3.3.6.tar.gz
sudo mv hadoop-3.3.6 /usr/local/hadoop
sudo chown -R hadoop:hadoop /usr/local/hadoop

Step 4: Set Environment Variables

Add to ~/.bashrc (on all nodes):
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bi

Before (worker + master):
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh

Add:
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Reload:
source ~/.bashrc

Step 5: Configure Hadoop Files on Master Node

a) core-site.xml
File Path: $HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master_ip_address:9000</value>
  </property>
</configuration>
```

b) hdfs-site.xml
File Path: $HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/data/nn</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/data/dn</value>
  </property>
</configuration>
```

c) mapred-site.xml
File Path: $HADOOP_HOME/etc/hadoop/mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

d) yarn-site.xml
File Path: $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>
</configuration>
```

Step 6: **Define Worker Nodes**

Edit workers file on the master:
nano $HADOOP_HOME/etc/hadoop/workers
Add:
worker1
worker2
worker3

Step 7: Copy Configuration Files to All Worker Nodes

scp -r /usr/local/hadoop/etc/hadoop hadoop@worker1:/usr/local/hadoop/etc/
scp -r /usr/local/hadoop/etc/hadoop hadoop@worker2:/usr/local/hadoop/etc/

Step 8: Format the NameNode (Run ONCE on master)

hdfs namenode -format

Step 9: Start the Cluster

Start HDFS:
start-dfs.sh
Start YARN:
start-yarn.sh

Step 10: Verify Cluster

On master:
jps
Expected services:
- NameNode
- SecondaryNameNode
- ResourceManager

On worker nodes:
jps
Expected services:
- DataNode
- NodeManager
-

Step 11: Access Web UI

Component	**URL**
NameNode	http://master_ip_address:9870
ResourceManager	http://master_ip_address:8088

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System - Master Node(1) and Worker Nodes (min. 2)	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service		

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Add one more worker node to the cluster and verify its registration in the NameNode UI.
2. Upload a dataset to HDFS and test replication across nodes.
3. Differentiate between NameNode and DataNode in a multi-node Hadoop cluster
4. Explain the need of passwordless SSH essential for Hadoop multi-node setup

Space for Answer

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..
……
……
……..
……
……
……..
……
……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

X References:

1. https://data-flair.training/blogs/hadoop-cluster-setup/
2. https://www.guru99.com/hadoop-tutorial.html
3. https://www.digitalocean.com/community/tags/hadoop
4. Installation Video https://www.youtube.com/watch?v=_iP2Em-5Abw

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 5: Configure Multi-Node Hadoop Cluster.

I Practical Significance

This practical enables students to understand how Hadoop functions in a distributed environment using multiple nodes. It provides hands-on experience in configuring HDFS for replication and distributed data storage..

II Industry / Employer Expected Outcome(s)

This practical enables students to understand how Hadoop functions in a distributed environment using multiple nodes. It provides hands-on experience in configuring HDFS for replication and distributed data storage.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 5.1: Configure multi-node hadoop cluster.

V Relevant Affective Domain related Outcomes

Students develop responsibility, accuracy, and confidence while configuring distributed systems. They show patience and teamwork while managing multiple machines in a coordinated manner.

VI Relevant Theoretical Background

Introduction

A multi-node Hadoop cluster uses a master–slave architecture where:

- **Master Node** runs *NameNode* and manages file system metadata.
- **Worker/Slave** Nodes run *DataNodes* and store actual data blocks.

HDFS ensures fault tolerance by replicating blocks across multiple DataNodes.

Practical Steps to Configure Multi-Node HDFS

Firstly (master+worker):
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
Add:
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Step 1: Configure core-site.xml (Master Node)

File Path:$HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://machine_ip_address:9000</value>
  </property>
</configuration>
```

Purpose: Defines the NameNode URI for all cluster nodes.

Step 2: Configure hdfs-site.xml (Master + Worker Nodes)

File path:$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/data/nn</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/data/dn</value>
  </property>
</configuration>
```

Purpose:

- Sets replication factor = 3
- Specifies storage directories for NameNode & DataNode

Step 3: Add Worker Nodes to workers File

Edit on master only:
nano $HADOOP_HOME/etc/hadoop/workers
Add hostnames:
worker1
worker2
worker3

Step 4: Distribute Configuration to Workers

scp -r $HADOOP_HOME/etc/hadoop hadoop@worker1:/usr/local/hadoop/etc/
scp -r $HADOOP_HOME/etc/hadoop hadoop@worker2:/usr/local/hadoop/etc/

Step 5: Format NameNode (Master Only)

hdfs namenode -format

Step 6: Start NameNode and DataNode Daemons

Start HDFS daemons (on master):
start-dfs.sh

Verify processes:
On Master (NameNode):
jps
Expected:

- NameNode
- SecondaryNameNode

On Worker Nodes (DataNode):
jps
Expected:

- DataNode

Step 7: Upload Dataset to HDFS

Create directory:
hdfs dfs -mkdir /data

Upload any dataset (e.g., sample.csv):
hdfs dfs -put sample.csv /data/

List files:
hdfs dfs -ls /data/

Step 8: Verify Distributed Storage

Check block locations:
hdfs fsck /data/sample.csv -blocks -locations

Verification:

- Data blocks should appear distributed across **multiple DataNodes**.
- Replication factor should match **dfs.replication**

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System- Master Node(1) and Worker Nodes (min. 2)	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service		

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Add one additional worker node to the cluster and verify DataNode registration in the NameNode UI.
2. Upload a large dataset to HDFS and check how block distribution varies across nodes.
3. Stop one DataNode and analyze how Hadoop manages replicated blocks.
4. Describe the purpose of configuring dfs.replication in a multi-node Hadoop cluster
5. Why is the NameNode configured only on the master node while DataNodes run on workers?
6. Explain how HDFS ensures fault tolerance in multi-node storage.

Space for Answer

……..
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

X References:

1. https://data-flair.training/blogs/hadoop-cluster-setup/
2. https://www.digitalocean.com/community/tutorials/how-to-set-up-a-hadoop-cluster
3. https://www.guru99.com/hdfs-tutorial.html
4. https://www.youtube.com/watch?v=c2Lg5c8v4YQ

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 6: Use Monitoring Tools to Observe Cluster Resources

I Practical Significance

This practical helps students understand how to monitor Hadoop cluster performance and resource usage in real time. It enables learners to analyze system health and identify bottlenecks in distributed computing environments.

II Industry / Employer Expected Outcome(s)

Employers expect students to use Hadoop monitoring tools to ensure stable cluster performance. Students should be able to interpret resource metrics such as CPU, memory, disk usage, and detect performance issues

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 6.1: Use resource utilization and performance tools in Hadoop Cluster..

V Relevant Affective Domain related Outcomes

Students develop attentiveness and responsibility in system observation and performance analysis. They cultivate patience and analytical thinking while diagnosing resource utilization patterns.

VI Relevant Theoretical Background

Introduction

Monitoring is a critical part of Hadoop cluster administration. Hadoop provides multiple tools to observe cluster health, including:

- **Web UI dashboards (NameNode, ResourceManager)**
- **CLI commands (hdfs dfsadmin, yarn top, jps, du, df)**
- **System-level tools (top, free, iostat)**

These tools help administrators track CPU load, memory usage, disk space, active DataNodes, running jobs, and block distribution.

Practical Steps

Step 1: Access Hadoop Monitoring Interfaces (Web UI)

start-dfs.sh
start-yarn.sh

NameNode Web UI:
http://localhost:9870

Provides:

- Live DataNodes
- Disk usage
- HDFS capacity
- Block distribution

ResourceManager Web UI:
http://localhost:8088

Displays:

- Running applications
- Cluster resources (CPU, memory)
- Node usage

Step 2: Locate Resource Usage Indicators (CPU, Memory, Disk)

A) System-level monitoring commands

Run on any node:

CPU Usage

1. top
2. mpstat

Memory Usage

1. free -h

Disk Usage

1. df -h

Disk I/O

1. iostat

B) Hadoop-specific monitoring indicators

HDFS Storage Report

1. hdfs dfsadmin -report

Node Resource Status

1. yarn node -list

HDFS Storage Report

1. yarn top

Step 3: Run Basic Monitoring Commands

Check running Hadoop Daemons

1. jps

Check HDFS Block Status

1. hdfs fsck / -files -blocks -locations

Check DataNode storage usage:

1. hdfs dfs -du -h /

Check YARN Memory usage:

1. yarn top

View System Performance:

1. top

Step 4: Record Resource Values During Cluster Operation

Perform the following during file upload or job execution:
Example table for recording values:

Parameter	Master Node Value	Worker Node 1 Value	Worker Node 2 Value
CPU (%)	30%	40%	35%
Memory Usage	2.5 GB	1.8 GB	1.6 GB
Disk Usage	20 GB used	18 GB used	17 GB used
HDFS Load	120 blocks	98 blocks	110 blocks

Students should monitor values at:

- Idle state
- During dataset upload
- During job execution (e.g., wordcount)

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service		

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Upload a dataset to HDFS and record CPU, memory, and disk usage on all nodes during the operation.
2. Run a sample MapReduce or Spark job and record changes in cluster resource usage.
3. Interpret HDFS storage details using hdfs dfsadmin -report and prepare a summar
4. Explain the information provided by the NameNode Web UI about the HDFS cluster.
5. Explain how to check real-time YARN memory usage using the command-line interface (CLI).
6. Explain monitoring disk usage importance in a Hadoop cluster.

Space for Answer

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://data-flair.training/blogs/hadoop-administration-tutorial/
2. Interacting with HDFS using Command Line Interface and Ambari Web UI https://www.youtube.com/watch?v=LO8jTO5qsCE

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 7: *Perform Basic File Operations Using HDFS

I Practical Significance

This practical helps students understand how to work with the Hadoop Distributed File System (HDFS) using command-line operations. It provides hands-on experience in managing distributed files within a big data environment.

II Industry / Employer Expected Outcome(s)

Employers expect students to efficiently perform file operations in HDFS, manage permissions, and verify replication settings. Students should be able to interact with distributed storage systems confidently.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 7.1: Execute file operations using HDFS commands.

V Relevant Affective Domain related Outcomes

Students develop responsibility and accuracy while performing operations on distributed storage. They also gainconfidence and discipline in executing commands and managing file permissions.

VI Relevant Theoretical Background

Introduction

HDFS is a distributed storage system that stores files across multiple DataNodes with replication for fault tolerance. Hadoop provides command-line utilities to create directories, upload files, read data, modify permissions, and verify replication.

Practical Steps

Step 1: Create Directories and Files in HDFS

Create a directory in HDFS
hdfs dfs -mkdir /student

Create a nested directory
hdfs dfs -mkdir -p /student/records

Upload (create) a file in HDFS
hdfs dfs -put sample.txt /student/records

Create an empty file in HDFS
hdfs dfs -touchz /student/emptyfile.txt

Step 2: Perform Read, Write, Update, Delete Operations

Read a file from HDFS
hdfs dfs -cat /student/records/sample.txt

Copy file from HDFS to local (read/download)
hdfs dfs -get /student/records/sample.txt .

Write/Append data to an HDFS file
hdfs dfs -appendToFile localfile.txt /student/records/sample.txt

Update a file
HDFS does not support in-place editing. Update is done by:

1. Download → Edit locally → Re-upload (overwrite)

hdfs dfs -put -f updated.txt /student/records/sample.txt

Delete a file
hdfs dfs -rm /student/records/sample.txt

Delete a directory
hdfs dfs -rm -r /student

Step 3: Set Directory Permissions

Change permissions
hdfs dfs -chmod 755 /student

Change ownership
hdfs dfs -chown hadoop_user:hadoop /student

Change group
hdfs dfs -chgrp hadoop /student

Step 4: Verify File Replication

Check replication factor of a file
hdfs dfs -stat %r /student/records/sample.txt

Change replication factor
hdfs dfs -setrep 3 /student/records/sample.txt

Verify block locations
hdfs fsck /student/records/sample.txt -files -blocks -locations

This shows how the file is distributed across DataNodes.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 /OpenJDK 11 Hadoop 3.3.x(stable version) SSH Service		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Create a directory structure in HDFS for any department (e.g., /dept/cse/notes), upload three files, set permissions, and list directory contents.
2. Upload a dataset to HDFS, set its replication factor to 2 or 3, and verify block distribution using hdfs fsck.
3. Why does HDFS use replication for file storage? Explain its advantages.
4. Differentiate between chmod, chown, and chgrp in HDFS?

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

X References:

1. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsCommands.html
2. https://www.tutorialspoint.com/hadoop/hadoop_hdfs_commands.htm
3. https://data-flair.training/blogs/hdfs-commands-tutorial/
4. Basic HDFS Commands https://www.youtube.com/watch?v=7cbLOWAnBFc

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 8: Perform Backup and Restore of Datasets in HDFS.

I Practical Significance

This practical helps students understand how to securely back up and restore datasets stored in the Hadoop Distributed File System (HDFS). It provides hands-on knowledge for preventing data loss and maintaining data availability in distributed systems.

II Industry / Employer Expected Outcome(s)

Employers expect students to manage data safety in big data environments by performing backup, recovery, and data restoration operations. Students should be able to handle accidental deletions and ensure business continuity.

III Course Level Learning Outcomes(s)

CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 8.1: Execute basic backup and restore operations for Big Data stored in HDFS.

V Relevant Affective Domain related Outcomes

1. Students **explain** the importance of data backup and **justify** recovery strategies for fault tolerance.
2. Students **organize** datasets systematically and **value** data protection practices in real-world scenarios.

VI Relevant Theoretical Background

Introduction

Backups are essential in any big data environment to protect important datasets from accidental deletion, corruption, or node failures. HDFS allows users to copy datasets within the file system and restore them when required. Backup and restore operations help maintain data integrity and fault tolerance.

Practical Steps

Step 1: Load Any Dataset into HDFS

Upload dataset to main directory:
note: Make sure sample.csv exists on your local system (Linux home folder).
hdfs dfs -mkdir /data
hdfs dfs -put sample.csv /data/

Verify upload:

hdfs dfs -ls /data

Step 2: Copy Dataset to a Backup Directory

Create backup directory:
hdfs dfs -mkdir /backup

Copy dataset:
hdfs dfs -cp /data/sample.csv /backup/

Verify backup:
hdfs dfs -ls /backup

Step 3: Simulate Accidental Data Removal

Delete dataset from main directory:
hdfs dfs -rm /data/sample.csv

Check if deleted:
hdfs dfs -ls /data

Expected result: **file not found**

Step 4: Restore Dataset from Backup

Restore backup to original location:
hdfs dfs -cp /backup/sample.csv /data/

Verify restoration:
hdfs dfs -ls /data

Dataset should now appear in /data.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS	1	
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH Service	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Create a directory structure /project/data and /project/data_backup. Upload two datasets, delete one dataset, and restore it from backup.
2. Change the replication factor of the backup file to 2 or 3 and verify block distribution before and after restoration.
3. Explain importance to maintain backups in HDFS
4. Differentiate between hdfs dfs -cp and hdfs dfs -mv during backup operations?
5. Explain how accidental data deletion can be handled using HDFS commands.

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://hadoop.apache.org
2. https://www.digitalocean.com/community/tutorials/install-hadoop-ubuntu
3. https://data-flair.training/blogs/hadoop-tutorial

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 9: *Execute WordCount MapReduce Program

I Practical Significance

This practical teaches students how to implement and execute a basic MapReduce program for word counting, which helps them understand distributed data processing. It demonstrates how Hadoop splits tasks across multiple nodes for parallel processing.

II Industry / Employer Expected Outcome(s)

Employers expect students to write, compile, and execute basic MapReduce programs and understand distributed job execution in Hadoop. Students should be able to test and validate outputs from distributed processing jobs.

III Course Level Learning Outcomes(s)

CO1 - Illustrate different phases of big data with respect to real world application.
CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 9.1: Develop a Map Reduce program.

V Relevant Affective Domain related Outcomes

1. Students **demonstrate** the ability to design and execute a complete distributed application.
2. Students **explain** how MapReduce processes data in parallel and **justify** its importance in large-scale analytics

VI Relevant Theoretical Background

Introduction

MapReduce is a distributed programming model used to process large datasets in parallel. A WordCount program counts the occurrences of each word in a given text file. It consists of two main phases:

- **Mapper**: Processes input and emits key-value pairs (word → 1).
- **Reducer**: Aggregates count for each word.

Practical Steps

Step 1: Load Any Text File into HDFS

Create input directory:
hdfs dfs -mkdir /input

Upload a text file:
hdfs dfs -put sample.txt /input

Verify:
hdfs dfs -ls /input

Step 2: Develop Word Count Program in Java

nano WordCount.java

WordCount.java
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

   public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(Object key, Text value, Context context
      ) throws IOException, InterruptedException {
         String[] tokens = value.toString().split("\\s+");
         for (String token : tokens) {
            word.set(token);
            context.write(word, one);
         }
      }
   }

   public static class IntSumReducer
        extends Reducer<Text, IntWritable, Text, IntWritable> {

      public void reduce(Text key, Iterable<IntWritable> values,
                         Context context
      ) throws IOException, InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         context.write(key, new IntWritable(sum));
      }
   }

   public static void main(String[] args) throws Exception {

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");

      job.setJarByClass(WordCount.class);
      job.setMapperClass(TokenizerMapper.class);

      job.setCombinerClass(IntSumReducer.class);
      job.setReducerClass(IntSumReducer.class);

      job.setOutputKeyClass(Text.class);
```

```
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Save file as: **WordCount.java**

Step 3: Compile, Execute & Validate Output

Compile the Program

```
hadoop com.sun.tools.javac.Main WordCount.java #instead of this
javac -cp $(hadoop classpath) WordCount.java
```

Create JAR File

```
jar cf wc.jar WordCount*.class
```

Run the MapReduce Job

```
hadoop jar wc.jar WordCount /input /output
```

Check Output Files

```
hdfs dfs -ls /output
hdfs dfs -cat /output/part-r-00000
```

This file contains the list of words and their occurrence counts.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version) SSH ServiceHadoop MapReduce Libraries		

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Modify the WordCount program to ignore punctuation and convert all words to lowercase before counting.
2. Run the program on a large text dataset and compare execution time between local mode and cluster mode.
3. Explain roles of Mapper and Reducer play in a MapReduce program

4. Explain why data splitting and parallel processing are important in Big Data analytics.

Space for Answer

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://hadoop.apache.org
2. https://www.digitalocean.com/community/tutorials/install-hadoop-ubuntu
3. https://data-flair.training/blogs/hadoop-tutorial
4. WordCount Program in Java Hadoop MapReduce Model
   https://www.youtube.com/watch?v=uH5y6nTo_04

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 10: Process Large CSV Dataset Using MapReduce

I Practical Significance

This practical teaches students how to process large CSV datasets using MapReduce by extracting specific fields and performing aggregation. It provides real-world experience in handling structured data in big data environments.

II Industry / Employer Expected Outcome(s)

Employers expect students to develop MapReduce programs that process CSV data, filter required columns, and compute aggregated results. Students should be able to handle large-scale batch processing using Hadoop.

III Course Level Learning Outcomes(s)

CO1 - Illustrate different phases of big data with respect to real world application.
CO2 - Demonstrate the use of Hadoop core components for big data processing.

IV Laboratory Learning Outcome(s)

LLO 10.1: Execute a data processing workflow with MapReduce for CSVfiles.

V Relevant Affective Domain related Outcomes

1. Students **demonstrate** the ability to analyze and extract meaningful information from large datasets.
2. Students **explain** how MapReduce handles structured data and **justify** the use of parallel processing.

VI Relevant Theoretical Background

Introduction

CSV (Comma Separated Values) files are widely used in data analysis and storage. Processing large CSV files requires distributed computing for efficiency.
MapReduce enables:

- Field extraction in the **Mapper**
- Data aggregation in the **Reducer**

This practical uses MapReduce to extract specific CSV columns and compute aggregated results.

Practical Steps

Step 1: Load a CSV Dataset into HDFS

Create directory:
hdfs dfs -mkdir /csvdata

Upload dataset:
hdfs dfs -put large_dataset.csv /csvdata/

Verify:
hdfs dfs -ls /csvdata

Step 2: Run a MapReduce Job to Extract Specific Fields

Below is an example MapReduce program to extract:

- Field 1: Category

- Field 2: Sales Amount (for aggregation)

File Creation - nano CSVProcessing.java

CSVProcessing.java

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class CSVProcessing {

   public static class CSVMapper extends Mapper<Object, Text, Text, IntWritable> {

      private Text category = new Text();
      private IntWritable amount = new IntWritable();

      public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

         String[] fields = value.toString().split(",");

         try {
            category.set(fields[0]);                  // Extract Category
            int sales = Integer.parseInt(fields[2]);     // Extract Amount
            amount.set(sales);
            context.write(category, amount);
         } catch (Exception e) {
            // Skip invalid rows
         }
      }
   }

   public static class CSVReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

      public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

         int total = 0;
         for (IntWritable val : values) {
            total += val.get();        // Aggregate amounts
         }
         context.write(key, new IntWritable(total));
      }
   }

   public static void main(String[] args) throws Exception {
```

```
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "CSV Processing");

        job.setJarByClass(CSVProcessing.class);
        job.setMapperClass(CSVMapper.class);
        job.setReducerClass(CSVReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Step 3: Compile, Execute, and Validate Output

Compile
hadoop com.sun.tools.javac.Main CSVProcessing.java
javac -classpath "$(hadoop classpath)" -d . CSVProcessing.java

Create JAR
jar cf csvproc.jar CSVProcessing*.class

Execute
hadoop jar csvproc.jar CSVProcessing /csvdata /outputcsv

View Output
hdfs dfs -cat /outputcsv/part-r-00000
This displays each **Category** and its **Total Aggregated Sales**.

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11 Hadoop 3.3.x (Latest stable version)CSS dataset		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such

questionsto ensure the achievement of identified CO.

1. Modify the program to calculate the average sales amount per category instead of total.
2. Extract three fields from a CSV file (e.g., category, region, amount) and perform aggregation for each combination.
3. Explain challenges arise when processing large CSV datasets in a single machine, and how does MapReduce help
4. Discuss the roles of the Mapper and Reducer in performing field extraction and aggregation.
5. Explain what happens if the CSV contains invalid rows or missing values

Space for Answer

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

……

……

……

……

……

……

……

X References:

1. https://data-flair.training/blogs/mapreduce-example-wordcount/
2. https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
3. https://github.com/apache/hadoop
4. Performing Sum operation on a dataset in Hadoop
   https://www.youtube.com/watch?v=K0aDh_sfVrc

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 11: Install NoSQL Database (MongoDB) and Create Collections.

I Practical Significance

This practical helps students understand the installation and setup of MongoDB, a popular NoSQL database. It familiarizes them with creating databases, collections, and documents for efficient storage of unstructured and semi-structured data..

II Industry / Employer Expected Outcome(s)

Employers expect students to install, configure, and use MongoDB for real-world data storage. Students should be capable of creating collections, inserting documents, and managing NoSQL databases using CLI or GUI tools.

III Course Level Learning Outcomes(s)

CO3 - Apply NoSQL database concepts and architecture patterns to manage big data.

IV Laboratory Learning Outcome(s)

LLO11.1: Setup a MongoDB NoSQL database with collections.

V Relevant Affective Domain related Outcomes

1. Students **demonstrate** the ability to configure and use a NoSQL database for data storage
2. Students **explain** the difference between relational and non-relational databases.

VI Relevant Theoretical Background

Introduction

MongoDB is a document-oriented NoSQL database that stores data in flexible JSON-like documents (BSON). It is widely used for big data, analytics, real-time applications, and scalable environments.

Key Concepts

- **Database:** A container for collections
- **Collection:** A group of documents (similar to a table but schema-less)
- **Document:** A record stored as JSON-like key–value pairs
- **BSON:** Binary format used internally by MongoDB

Practical Steps

Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL).

Step 1: Add MongoDB Repo

```
curl -fsSL https://pgp.mongodb.com/server-7.0.asc | \
  sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg --dearmor

echo "deb [ signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ] \
https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 multiverse" \
| sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list
```

Step 2: Install MongoDB

```
sudo apt update
```

sudo apt install -y mongodb-org

Step 3: Start & Enable Service

sudo systemctl start mongod
sudo systemctl enable mongod
sudo systemctl status mongod
Step 4: Use MongoDB Shell

Mongosh

Step 5: Create a Database

use collegeDB
If the database doesn't exist, MongoDB will create it when a collection/document is added.

Step 6: Create Collections

Create collection (method 1 – explicit):
db.createCollection("students")

Create collection (method 2 – implicit by inserting a document):
db.teachers.insertOne({name: "Ravi", subject: "DBMS"})

Step 7: Insert Documents

Insert a single document:
```
db.students.insertOne({
   name: "Amit",
   roll: 21,
   branch: "Computer",
   marks: 85
})
```
Insert multiple documents:
```
db.students.insertMany([
   {name: "Sneha", roll: 22, marks: 90},
   {name: "Rahul", roll: 23, marks: 88}
])
```
Step 6: View Collections and Documents
Show all collections:
show collections
View documents:
db.students.find().pretty()

Step 8: Verify Data Storage

Use the following command:
db.students.countDocuments()

This confirms that documents are stored properly.

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	MongoDB Version 4.x or 5.x Mongo Shell (mongosh) Text Editor		

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Create a MongoDB database for a library system. Add two collections (books, members) and insert at least five documents in each.
2. Create a collection named faculty and insert multiple faculty records, then display them using find().pretty().
3. Differentiate between a MongoDB collection and a SQL table?
4. Explain why is MongoDB considered schema-less
5. Explain the purpose of the insertOne() and insertMany() commands.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

X References:

1. https://www.mongodb.com/docs/
2. https://www.tutorialspoint.com/mongodb/
3. https://www.w3schools.com/mongodb/
4. https://www.geeksforgeeks.org/mongodb-tutorial/
5. How to Install MongoDB 8 on Ubuntu 24.04 LTS Linux
   https://www.youtube.com/watch?v=SNgaUYu5o1o
6. Learn MongoDB in 1 hour https://www.youtube.com/watch?v=c2M-rlkkT5o

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 12: Create a Schema for Unstructured Data in MongoDB.

I Practical Significance

This practical teaches students how to analyze unstructured JSON data and convert it into a structured MongoDB schema. It helps them understand schema design concepts used in NoSQL databases to store flexible and semi-structured information.

II Industry / Employer Expected Outcome(s)

Employers expect students to examine raw unstructured data, identify patterns, and design efficient MongoDB schemas.

III Course Level Learning Outcomes(s)

CO3 - Apply NoSQL database concepts and architecture patterns to manage big data.

IV Laboratory Learning Outcome(s)

LLO12.1: Create a schema for unstructured data in MongoDB using any dataset.

V Relevant Affective Domain related Outcomes

1. Students **analyze** unstructured datasets to identify patterns and key attributes.
2. Students **demonstrate** proper schema design principles while organizing JSON data.

VI Relevant Theoretical Background

MongoDB is a NoSQL database designed to store **unstructured and semi-structured data**. Unlike relational databases, MongoDB uses **flexible schemas** and stores records as **documents** in **BSON** format.

Why schema design is needed?

Even though MongoDB is schema-less, defining a **logical schema** helps:

- Improve query performance
- Maintain consistency
- Ensure better indexing
- Organize complex data such as logs, reviews, profiles, metadata

Practical Steps

Step 1: Open MongoDB Shell

Mongosh

Step 2: Create / Select Database

use jsonDB

Step 3: Create reviews Collection With Schema Validation

```
db.createCollection("reviews", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["user_id", "rating", "comment"],
      properties: {
        user_id:   { bsonType: "string" },
```

```
        rating:    { bsonType: "int" },
        comment:   { bsonType: "string" },
        timestamp: { bsonType: "date" }
      }
    }
  }
})
```

Step 4: Insert Sample JSON Document

```
db.reviews.insertOne({
  user_id: "U123",
  username: "rahul_k",
  rating: 4,
  comment: "Very useful product!",
  likes: 12,
  timestamp: ISODate("2024-02-15T10:30:00Z"),
  metadata: {
    location: "India",
    device: "Android"
  }
})
```

Step 5: Insert Multiple Records (if needed)

```
db.reviews.insertMany([
  {
    user_id: "U124",
    username: "sneha_m",
    rating: 5,
    comment: "Excellent!",
    timestamp: ISODate("2024-02-16T11:20:00Z")
  },
  {
    user_id: "U125",
    username: "rahul_m",
    rating: 3,
    comment: "Average product",
    likes: 2,
    timestamp: ISODate("2024-02-17T09:45:00Z")
  }
])
```

Step 6: View Data

```
show collections
db.reviews.find().pretty()
```

Step 7: Count Documents

```
db.reviews.countDocuments()
```

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	MongoDBText Editor		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Choose a JSON dataset of website logs and design a MongoDB schema with nested subdocuments (e.g., IP, browser, device).
2. Import a JSON dataset into MongoDB using mongoimport and verify the document structure using find().pretty().
3. Explain why schema design is important even though MongoDB is schema-less.
4. Describe the benefits of using sub-documents in MongoDB schemas.
5. Describe the way MongoDB manages optional fields within documents.

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……

X References:

1. https://www.mongodb.com/docs/manual/data-modeling/
2. https://www.mongodb.com/json-and-bson
3. https://www.geeksforgeeks.org/mongodb-schema-design/
4. Different types of schema like Structured, Semi-Structured, Unstructured documents https://www.youtube.com/watch?v=IaqwlKWEhEk

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No.13: *Run Basic Aggregation Queries on Unstructured Dataset in MongoDB.

I Practical Significance

This practical teaches students how to analyze unstructured data using MongoDB's aggregation pipeline. It allows learners to validate schema design and extract meaningful insights such as counts, groups, and averages.

II Industry / Employer Expected Outcome(s)

Employers expect students to perform basic analytical operations on NoSQL data, including grouping, filtering, and computing summary statistics.

III Course Level Learning Outcomes(s)

CO3 - Apply NoSQL database concepts and architecture patterns to manage big data.

IV Laboratory Learning Outcome(s)

LLO13.1: Run basic aggregation queries on Unstructured dataset in MongoDB.

V Relevant Affective Domain related Outcomes

1. Students **interpret** and **analyze** unstructured data using aggregation operations.
2. Students **demonstrate** understanding of schema validation through meaningful queries

VI Relevant Theoretical Background

Introduction

While Spark is optimized for distributed computation, visualization is typically a single-machine task performed by the driver program using libraries like Matplotlib or Seaborn. The process requires converting the final clustered Spark DataFrame into a local Pandas DataFrame (.toPandas()) for plotting. The resulting Scatter Plot uses the two selected features as axes, and the color of each point represents the cluster ID (prediction column).

Step 1: Import JSON Dataset

(Assuming your file is reviews.json and contains a JSON array)
mongoimport --db analyticsDB --collection reviews --file reviews.json --jsonArray

Verify:
mongosh
use analyticsDB
db.reviews.countDocuments()

Step 2: Basic Aggregation Queries

Inside mongosh:

A) Count total records
db.reviews.countDocuments()

B) Count how many entries per rating

```
db.reviews.aggregate([
  { $group: { _id: "$rating", total: { $sum: 1 } } }
])
```

C) Calculate average rating

```
db.reviews.aggregate([
  { $group: { _id: null, avgRating: { $avg: "$rating" } } }
])
```

D) Count reviews submitted by each user

```
db.reviews.aggregate([
  { $group: { _id: "$user_id", reviewCount: { $sum: 1 } } }
])
```

E) Find maximum likes

```
db.reviews.aggregate([
  { $group: { _id: null, maxLikes: { $max: "$likes" } } }
])
```

Step 3: Validate Schema Using Results

Run these checks:
Missing or inconsistent fields:

```
db.reviews.find({ rating: { $exists: false } })
db.reviews.find({ comment: { $exists: false } })
db.reviews.find({ timestamp: { $type: "string" } })   // detect wrong date types
```

Check field types:

```
db.reviews.aggregate([
  { $project: {
     ratingType: { $type: "$rating" },
     likesType: { $type: "$likes" },
     timestampType: { $type: "$timestamp" }
  }}
])
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	MongoDB Text Editor JSON Dataset		

VIII Conclusion

……..

……

……

……

IX Practical related questions

1. Write aggregation queries to find the top 3 most active users based on number of reviews.

2. Find the average, minimum, and maximum likes for reviews grouped by rating.
3. Find the average, minimum, and maximum likes for reviews grouped by rating.
4. Explain the role of $group and $match in aggregation
5. Describe the importance of schema validation through aggregation queries

Space for Answer

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………

X References:

1. https://www.mongodb.com/docs/manual/aggregation/
2. https://www.w3schools.com/mongodb/mongodb_aggregation.asp
3. https://www.geeksforgeeks.org/mongodb-aggregation-framework/
4. https://www.mongodb.com/docs/manual/reference/operator/aggregation/
5. The Ultimate MongoDB Aggregation Guide: Make Your Queries Soar in One Video https://www.youtube.com/watch?v=MWmMvudBgFU

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 14: Perform Basic MongoDB Operations Demonstrating CAP Theorem Behavior.

I Practical Significance

This practical demonstrates how MongoDB replica sets behave under normal and disrupted network conditions, helping students understand the CAP theorem through real-time database operations.

II Industry / Employer Expected Outcome(s)

Employers expect students to configure replica sets, perform distributed database operations, and analyze system behavior when consistency, availability, or partition tolerance is challenged.

III Course Level Learning Outcomes(s)

CO3 - Apply NoSQL database concepts and architecture patterns to manage big data.

IV Laboratory Learning Outcome(s)

LLO14.1: Apply basic operations in MongoDB to observe the behavior of CAP theorem properties.

V Relevant Affective Domain related Outcomes

1. Students **demonstrate** awareness of distributed system behavior by performing replica set operations.
2. Students **explain** how network disruptions influence consistency and availability in NoSQL databases.

VI Relevant Theoretical Background

Introduction to CAP Theorem

The CAP Theorem states that a distributed database system can guarantee only two of the following at any given time:

- **Consistency (C):** Every read receives the most recent write.
- **Availability (A):** Every request receives a response, even if some nodes are down.
- **Partition Tolerance (P):** System continues to operate despite network failures.

MongoDB replica sets help provide high availability and fault tolerance. Under network partition, MongoDB prioritizes CP behavior (Consistency + Partition Tolerance), meaning some write operations may pause.

Practical Steps

Step 1: Create 3 MongoDB Replica-Set Instances

Create data directories

```
sudo mkdir -p /data/rs0 /data/rs1 /data/rs2
```

Start each instance on a different port

```
mongod --replSet rs0 --port 27017 --dbpath /data/rs0 --bind_ip localhost --fork --logpath /data/rs0/mongod.log
mongod --replSet rs0 --port 27018 --dbpath /data/rs1 --bind_ip localhost --fork --logpath /data/rs1/mongod.log
mongod --replSet rs0 --port 27019 --dbpath /data/rs2 --bind_ip localhost --fork --logpath /data/rs2/mongod.log
```

Step 2: Initialize Replica Set

Open shell to the first node:
mongosh --port 27017

Inside mongosh:
```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "localhost:27017" },
    { _id: 1, host: "localhost:27018" },
    { _id: 2, host: "localhost:27019" }
  ]
})
rs.status()
```

Step 3: Basic Write / Read Operations

In primary (usually port 27017):
```
use testdb
db.items.insertOne({ name: "Laptop", price: 50000 })
db.items.find()
```

Connect to secondary:
mongosh --port 27018

Inside:
```
rs.slaveOk()
db.items.find()
```

Step 4: Simulate Failures

Stop a secondary:
sudo pkill -f "port 27018"

Stop the primary:
sudo pkill -f "port 27017"

Simulate network block:
sudo iptables -A INPUT -p tcp --dport 27017 -j DROP

Step 5: Test Behavior During Failure

Writes:
db.items.insertOne({ name: "Phone", price: 25000 })

Reads:
db.items.find()

Check status:
rs.status()

Step 6: Restore Normal State

Remove iptables rule:

sudo iptables -D INPUT -p tcp --dport 27017 -j DROP

Restart failed node (example for port 27018):
mongod --replSet rs0 --port 27018 --dbpath /data/rs1 --bind_ip localhost --fork --logpath /data/rs1/mongod.log
Re-check:
rs.status()

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	MongoDB 3 MongoDB instances iptables or service control for disruption		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Simulate failure of the primary node and record how long it takes for a secondary to become the new primary.
2. Perform multiple write operations during a simulated network delay and analyze which operations succeed or fail.
3. Explain how does MongoDB prioritize CAP theorem components during a network partition
4. Describe the impact on write operations if the primary node is not reachable.
5. Why is replica set election important for high availability?

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

X References:

1. https://www.mongodb.com/docs/manual/replication/
2. https://www.mongodb.com/docs/manual/administration/replica-set-member-configuration/
3. https://www.geeksforgeeks.org/mongodb-replication/
4. https://www.mongodb.com/cap-theorem
5. Explain CAP Theorem Consistency, Availability and the Partition Tolerance - MongoDB https://www.youtube.com/watch?v=z_dVOenbIy0

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 15: *Install and Configure Hive.

I Practical Significance

This practical introduces students to Hive, enabling them to perform SQL-like queries on large datasets stored in Hadoop.

II Industry / Employer Expected Outcome(s)

Employers expect students to install Hive, create tables, load data, and execute HiveQL queries for big data analytics.

III Course Level Learning Outcomes(s)

CO4: Use Hive and Pig for data processing and transformation within big data environments.

IV Laboratory Learning Outcome(s)

LLO15.1: Install Hive within aHadoop environment..

V Relevant Affective Domain related Outcomes

1. Students **demonstrate** the ability to configure Hive and work with SQL-like queries on big data.
2. Students **explain** the use of Hive in simplifying complex data processing tasks.
3. Students **organize** datasets and **value** accuracy while executing data operations in Hive.

VI Relevant Theoretical Background

Introduction

The Hadoop ecosystem uses several XML configuration files to define cluster behavior. In a local (pseudo-distributed) setup, Hadoop runs multiple daemons—NameNode, DataNode, ResourceManager, and NodeManager—on a single machine. Configuring these files correctly is essential for HDFS and MapReduce to work.

Key Concepts

- **Database:** Logical grouping of tables
- **Table:** Stores structured data
- **Internal Table:** Hive manages the data location
- **External Table:** Data stored outside Hive (in HDFS)
- **HiveQL: SQL-like query language**\

Practical Steps

Students are required to use an Ubuntu system to carry out this experiment (via native installation, VMware, or WSL).

Step 1: Install Hive on Hadoop Environment

Install required packages:
sudo apt update
sudo apt install openjdk-8-jdk -y

Download Hive:
wget -c https://archive.apache.org/dist/hive/hive-3.1.3/apache-hive-3.1.3-bin.tar.gz
tar -xvzf apache-hive-3.1.3-bin.tar.gz

sudo mv apache-hive-3.1.3-bin /usr/local/hive

Set environment variables:
nano ~/.bashrc

export HIVE_HOME=/usr/local/hive
export PATH=$PATH:$HIVE_HOME/bin
Apply changes:
source ~/.bashrc

Configure Hive metastore:

Initialize schema:
schematool -dbType derby -initSchema

Paste the csv file in home directory:
cp /mnt/c/Users/<username>/Downloads/students.csv ~/

Before loading put the data set in input folder:
hdfs dfs -mkdir /input
hdfs dfs -put ~/students.csv /input/

Start Hive:
hive

Step 2: Load a Dataset into Hive

Create database:
CREATE DATABASE collegeDB;
USE collegeDB;
Create table structure:
```
CREATE TABLE students (
   roll INT,
   name STRING,
   marks INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```
Load CSV file from HDFS:
LOAD DATA INPATH '/input/students.csv' INTO TABLE students;

Step 3: Run Basic HiveQL Queries

Select records:
SELECT * FROM students;

Insert new record:
INSERT INTO TABLE students VALUES (105, 'Riya', 88);

Delete a record (Hive 3+ supports delete):
DELETE FROM students WHERE roll = 105;

Filter with WHERE clause:
SELECT name, marks FROM students WHERE marks > 80;

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Computer System	Minimum 8 GB RAM,250 GB Hard disk ,Quad Core Processor with Stable internet connection	As per Batch Size	
2.	Operating System	Ubuntu 20.04 LTS		
3.	Software	OpenJDK 8 or OpenJDK 11Hadoop 3.3.x (Latest stable version) Apache Hive 3.x		

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Create a new table for employee records, load data, and run SELECT, INSERT, and DELETE operations.
2. Create an external table that reads data directly from an HDFS directory and run queries on it.
3. Differentiate between an internal and external table in Hive
4. Explain why does Hive use HDFS as its primary storage layer
5. Explain how Hive converts SQL queries into Hadoop jobs

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://hive.apache.org/
2. https://cwiki.apache.org/confluence/display/Hive/Home
3. https://www.tutorialspoint.com/hive/
4. https://www.geeksforgeeks.org/hive-tutorial/
5. Configure Hive on Hadoop in Ubuntu | Data Engineering Tutorial for Beginners https://www.youtube.com/watch?v=LMrW2BFeRh8

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 16: *Execute Data Queries using HiveQL.

I Practical Significance

This is crucial for demonstrating the power of Apache Hive, which allows data analysts to use familiar SQL syntax (HiveQL) to query vast datasets stored in HDFS. It fulfills the mandatory task of executing queries after the initial Hive setup.

II Industry / Employer Expected Outcome(s)

Define and Manage schemas for large-scale data warehouses, enabling efficient storage and retrieval of structured data for reporting (Data Warehousing).

III Course Level Learning Outcomes(s)

CO4: Use Hive and Pig for data processing and transformation within big data environments.

IV Laboratory Learning Outcome(s)

LLO16.1: Execute after writing complex data queries using HiveQL.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Hive is a data warehouse software built on top of Hadoop. It facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Hive transforms HiveQL queries into a series of MapReduce or Spark jobs for execution.

Stepwise Procedure

1. Load any Dataset into Hive (Extraction & Schema Definition)

This section details the steps to prepare the data in HDFS and create the necessary table structures in the Hive Metastore.

Step 1: Start Following Services

Command: Use shell commands to start the Hadoop cluster and launch the Hive client.

Example:

1. **start-dfs.sh** (To start HDFS NameNode and DataNodes)
2. **hive** (To launch the Hive command line interface)

Step 2: Upload Data to HDFS

Action: Upload the raw dataset (Ex. employees.csv) from the local machine into a dedicated HDFS directory.

Command: Use the HDFS shell utility.

Example:

1. **hdfs dfs -mkdir -p /user/hive/data/org** (Create the directory)
2. **hdfs dfs -put /local/path/employees.csv /user/hive/data/org/employees/**

Step 3: Define the Table Schema

Action: Create an External Table in Hive. This defines the schema but leaves the data file in place in HDFS.

Command: Use CREATE EXTERNAL TABLE and specify the column types, data delimiter,

and HDFS location.

Example:

```
CREATE EXTERNAL TABLE IF NOT EXISTS employees (
   emp_id INT,
   name STRING,
   dept_id INT,
   salary FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/hive/data/org/employees/';
```

Step 4: Define the Departments Table Schema

Action: Repeat the upload and schema definition for the second dataset (departments.csv).
Command: Create the second table, ensuring the join key (dept_id) has the correct data type.

Example:

```
Assume data is in /user/hive/data/org/departments/
CREATE EXTERNAL TABLE IF NOT EXISTS departments (
   dept_id INT,
   dept_name STRING,
   city STRING
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/hive/data/org/departments/';
```

2. Execute Joins, Group By, and Aggregate Queries (Transformation & Analysis)

This section demonstrates analytical queries using the tables created in Part 1.

Step 5: Execute Simple Aggregate Queries

Action: Use built-in aggregate functions to summarize the data.
Query: Find the total number of employees and the overall average salary.

Example:

```
SELECT
   COUNT(*) AS total_employees,
   AVG(salary) AS average_salary_org
FROM employees;
```

Step 6: Execute Group by Queries

Action: Group records by a specific dimension and calculate an aggregate value for each group.
Query: Find the minimum, maximum, and total salary expenditure for each department ID.

Example:

```
SELECT
   dept_id,
   MIN(salary) AS min_salary,
   MAX(salary) AS max_salary,
   SUM(salary) AS total_expenditure
FROM employees
```

```
GROUP BY dept_id
HAVING SUM(salary) > 100000.0 // Optional: Filter the groups
ORDER BY total_expenditure DESC;
```

Step 7: Execute Join Queries

Action: Combine data from the two tables (employees and departments) using the common key (dept_id).

Query: List all employee names, their salaries, and the full department name.

Example:

```
SELECT
   e.name,
   e.salary,
   d.dept_name
FROM employees e
INNER JOIN departments d
ON e.dept_id = d.dept_id
WHERE d.city = 'New York'; //Filter after the join
```

Step 8: Verify Query Execution

Action: Review the query execution details to ensure that Hive is using the most efficient execution engine (MapReduce/Tez/Spark).

Command: Prefix the query with the EXPLAIN command to view the query plan.

Example:

```
EXPLAIN SELECT * FROM employees;
```

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Hive	Latest stable release (Integrated with the installed Hadoop version)	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	

VIII Conclusion

……..

……..

……..

IXPractical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. If the raw data file (employees.csv) is currently located in the HDFS directory /user/raw/data, write the full HiveQL command required to create an External Table named staging_employees that references this data. Assume the fields are separated by a comma (,).
2. Explain in one sentence why we use the JOIN operation in HiveQL when analyzing the employees table and the departments table. Identify the specific common column (key) that must exist in both tables to make this join possible.
3. Write the single HiveQL query that will calculate the total salary expenditure for all employees working in Department ID 101 and output only the final calculated sum.
4. After successfully loading the employee dataset, what two different HiveQL commands would you execute to verify (a) the table's schema structure (column names and data types) and (b) the total number of records successfully loaded?

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

X References:

1. https://hive.apache.org/
2. https://www.tutorialspoint.com/hive/index.htm
3. https://onlinecourses.nptel.ac.in/noc20_cs92/preview

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 17: Run Pig Scripts for data transformation.

I Practical Significance

This will provide hands-on experience with Apache Pig and its high-level scripting language, Pig Latin. It demonstrates how to perform fundamental ETL (Extract, Transform, Load) operations like filtering and grouping on large datasets using a simpler, sequential flow model than complex Java MapReduce code.

II Industry / Employer Expected Outcome(s)

Develop rapid ETL (Extract, Transform, Load) scripts for iterative data cleansing and manipulation on unstructured data.

III Course Level Learning Outcomes(s)

CO4: Use Hive and Pig for data processing and transformation within big data environments.

IV Laboratory Learning Outcome(s)

LLO17.1: Run after writing Pig scripts for data transformation.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Apache Pig is a platform for analyzing large datasets that compiles Pig Latin scripts into MapReduce jobs. It simplifies common data processing tasks like joining, filtering, and grouping. Data is processed in two modes: Local (for testing on the local file system) or MapReduce (for production on the Hadoop cluster).

Stepwise Procedure

This procedure details the execution of a Pig Latin script (Ex. data_analysis.pig) to transform a dataset (Ex. sales_data.csv).

Part 1: Load any Dataset into Pig (Extraction)

This part focuses on preparing the data in HDFS and using the LOAD operator to bring it into the Pig execution environment.

Step 1: Start Services and Launch Pig

Action: Ensure the Hadoop cluster is running and launch the Pig shell in MapReduce mode.
Command: Use the shell command pig (or pig -x local for testing).

Step 2: Prepare Data in HDFS

Action: Upload the sample dataset (sales_data.csv) into HDFS.

HDFS Command Example:

hdfs dfs –put/local/path/sales_data.csv/user/pig/input/

Step 3: Write the Pig Script (LOAD)

Action: Use the LOAD operator to extract data from HDFS and define the schema or the PigStorage function is used for delimited files.

Pig Latin Script Line 1:

```
sales = LOAD '/user/pig/input/sales_data.csv' USING PigStorage(',')
        AS (transaction_id:INT, customer_id:INT, region:CHARARRAY, amount:FLOAT,
date:CHARARRAY);
```

Step 4: Verify Schema and Content

Action: Inspect the structure and view a sample of the loaded data to confirm successful loading and parsing.

Pig Latin Commands:

```
DESCRIBE sales;
DUMP sales; //Use LIMIT for large data: ILLUSTRATE sales;
```

Part 2: Apply Filtering and Grouping Operations (Transformation)

This part focuses on the core data manipulation required by the experiment, using the FILTER and GROUP operators.

Step 5: Apply Filtering Operation

Action: Use the FILTER operator to select a subset of records based on a condition, mimicking a quality check or specific business requirement.

Query: Filter the sales data to keep only transactions where the amount is greater than 500 and the region is 'West'.

Pig Latin Script Line 2:

```
filtered_sales = FILTER sales BY (amount > 500.0) AND (region == 'West');
```

Step 6: Apply Grouping Operation

Action: Use the GROUP operator to collect all records with the same value for a specified column, preparing for aggregation.

Query: Group the filtered sales data by customer_id.

Pig Latin Script Line 3:

```
grouped_customers = GROUP filtered_sales BY customer_id;
```

Step 7: Apply Aggregation (Optional but Essential for Grouping)

Action: Use the FOREACH...GENERATE block with aggregate functions (like SUM or COUNT) to process the grouped data.

Query: Calculate the total number of transactions and the total amount spent for each customer group.

Pig Latin Script Line 4:

```
customer_summary = FOREACH grouped_customers GENERATE
group AS customer_id,
COUNT(filtered_sales) AS total_transactions,
SUM(filtered_sales.amount) AS total_amount_spent;
```

Step 8: Output the Final Result

Action: Output the transformed data either to the console (DUMP) or back to HDFS (STORE).

Pig Latin Commands:

```
DUMP customer_summary;
OR
```

```
STORE customer_summary INTO '/user/pig/output/west_sales_summary' USING
PigStorage(',');
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Hive	Latest stable release (Integrated with the installed Hadoop version)	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What are the two main Pig Latin operators used to achieve the following: (a) Selecting only those records where a field meets a condition, and (b) Bringing together all records with the same value in a specific column?
2. When you execute the Pig Latin command DUMP customer_summary;, briefly explain in your own words what happens *after* this command is given, but *before* you see the results on the screen (i.e., which underlying technology is involved)?
3. Write the single line of Pig Latin code that would load the alias my_data and then create a new alias high_income containing only records where the salary column (of type FLOAT) is greater than 60000.00.
4. Suppose you want to find the number of unique product categories purchased by each customer. Which Pig Latin operator would you use as the first step after loading the data, and by which column would you group the data?

Space for Answer

……..

……..

……..

……..

……..

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

……

……..

……

……

……

……

……

X References:

1. https://pig.apache.org/
2. https://www.tutorialspoint.com/apache_pig/index.htm
3. https://www.tutorialspoint.com/hadoop/hadoop_file_system.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 18: *Execute User Defined Functions (UDF) in Pig for Data Normalization.

I Practical Significance

This demonstrates the flexibility of Apache Pig. Students learn how to extend Pig's functionality by integrating custom code (User Defined Functions - UDFs), which is essential for performing advanced, custom ETL logic like data normalization, complex cleansing, or proprietary business rules not available in standard Pig Latin operators.

II Industry / Employer Expected Outcome(s)

Implement custom data quality and normalization logic via UDFs to handle complex business rules beyond standard ETL operations.

III Course Level Learning Outcomes(s)

CO4: Use Hive and Pig for data processing and transformation within big data environments.

IV Laboratory Learning Outcome(s)

LLO18.1: Use after writing User Defined Functions (UDF) in Pig.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

User Defined Functions (UDFs) allow programmers to write custom code to extend Pig's capabilities. A UDF is necessary when the required transformation is too complex or not available in the standard Pig Latin operators (e.g., Min-Max normalization requires global min/max values, which is complex). Data Normalization scales numerical values to a standard range (e.g., 0 to 1), preventing features with larger values from dominating machine learning models. The formula for Min-Max normalization is: $X_{normalized} = (X - X_{min}) / (X_{max} - X_{min})$.

Stepwise Procedure

The procedure is divided into UDF Development and Pig Script Execution.

Part 1: Develop UDF and Prepare Data

This part focuses on coding the custom logic and loading the raw data.

Step 1: Develop the Custom UDF Script

Action: Write the UDF code (e.g., Normalizer.py using **Jython**) that accepts a numerical value and the pre-calculated minimum and maximum values, then returns the normalized result.

UDF Example (Python): Define a function in python, perhaps normalize_value(x, x_min, x_max), that calculates (x - x_min) / (x_max - x_min).

Step 2: Prepare and Upload Data to HDFS

Action: Create a dataset with numerical readings (e.g., sensor data) and upload both the dataset and the UDF script to HDFS.

HDFS Command Example:

```
hdfs dfs -put /local/path/readings.csv /user/pig/input/
hdfs dfs -put /local/path/Normalizer.py /user/pig/udf/
```

Step 3: Load the Numerical Dataset into Pig

Action: Launch the Pig client and use the `LOAD` operator to bring the numerical data into an alias.

Pig Latin Command Example:

```
data = LOAD '/user/pig/input/readings.csv' USING PigStorage(',')
AS (id:INT, reading:FLOAT);
```

Part 2: Register UDF and Apply Normalization

This part focuses on integrating and executing the custom function within the Pig script.

Step 4: Register the UDF Script

Action: Use the `REGISTER` command to make the UDF file accessible to the Pig execution environment.

Pig Latin Command Example:

```
REGISTER '/user/pig/udf/Normalizer.py' USING jython AS my_udf;
```

(Note: If using a Java UDF, register the compiled JAR file.)

Step 5: Apply the UDF for Normalization

Action: Use the `FOREACH...GENERATE` operator to iterate through each record and apply the custom UDF to the numerical column. (Assume Min=0 and Max=100 for a simplified demonstration).

Query: Apply the `normalize_value` function from the registered UDF to the `reading` column.

Pig Latin Command Example:

NOTE: In a real scenario, you'd calculate Min/Max dynamically first, but here we hardcode for demonstration.

```
normalized_data = FOREACH data GENERATE
id,
my_udf.normalize_value(reading, 0.0, 100.0) AS normalized_reading;
```

Step 6: Output and Verify Normalized Data

Action: Review the transformed data to confirm that the output values are now scaled between 0 and 1.

Pig Latin Command Example:

```
DUMP normalized_data;
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Hive	Latest stable release (Integrated with the installed Hadoop version)	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the specific Pig Latin command used to tell the Pig environment where the Python script or Java JAR containing the custom UDF is located?
2. Explain in simple terms why a custom UDF is necessary to perform Min-Max normalization (like X_normalized = (X - 0) / (100 - 0)) in Pig, instead of just using the standard arithmetic operators available in Pig Latin.
3. Assume you have successfully registered a UDF named StringTools.CleanText as the alias text_udf. Write the single line of Pig Latin code that uses this UDF to apply the cleaning function to a column named tweet_body within the alias raw_tweets.
4. If the highest value in your dataset is 500 and the lowest is 0, what would be the normalized value (range 0 to 1) of a reading that is exactly 125? Show your simple calculation.

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

X References:

1.https://pig.apache.org/docs/latest/udf.html
2.https://pig.apache.org/docs/r0.11.1/udf.html#python-udfs
3.https://www.tutorialspoint.com/apache_pig/apache_pig_udfs.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 19: *Install Spark on Cluster environment and verify installation with any dataset.

I Practical Significance

This is a foundational experiment which introduces students to Apache Spark, the unified engine for large-scale data processing, which runs significantly faster than traditional MapReduce by leveraging in-memory computing. Successful installation and verification are the prerequisites for performing all subsequent experiments in Spark.

II Industry / Employer Expected Outcome(s)

Configure and maintain the Big Data processing environment, ensuring Apache Spark is correctly integrated with the cluster infrastructure for high-speed computation.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO19.1: Install Spark on a cluster and verify installation.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Apache Spark is a fast and general-purpose cluster computing system that uses the Resilient Distributed Dataset (RDD) as its core abstraction. Spark can run on a cluster managed by its own Standalone Manager or by resource managers like YARN or Mesos. The Pseudo-Distributed or Standalone mode is typically used. Verification involves using the Spark console to perform simple, distributed data operations.

Stepwise Procedure

The procedure is broken down into the four parts: Installation, Shell Initiation, Data Loading, and Verification.

Part 1: Install Spark on the Cluster (Setup)

1. Download and Extract:

Action: Download the pre-built Apache Spark package (e.g., *Spark 3.x for Hadoop 3.3*) from the official website.

Command: Extract the compressed file to a suitable location.

```
tar -xvf spark-*-bin-hadoop*.tgz
```

2. Configure Environment Variables:

Action: Set the environment variables, which tell the system where Spark and Java are located.

Command: Edit the .bashrc or .zshrc file.

```
export SPARK_HOME=/path/to/spark-directory
export PATH=$PATH:$SPARK_HOME/bin
source ~/.bashrc (To apply changes)
```

3. Configure Logging (Optional but Recommended):

Action: Create a log4j configuration file (log4j.properties) in the Spark configuration directory to limit verbose output.

Goal: Set the default logging level to WARN to minimize console clutter.

Part 2: Start Spark Shell (Launch)

4. Launch the Spark Shell:

Action: Start the interactive Spark console, which automatically initializes the **SparkSession** object (named spark) the entry point for all Spark functionality.
Command: Launch the Python-based shell (**PySpark**) or the Scala shell.

5. Verify Spark Context:

Action: Confirm that the Spark Web UI is accessible and the SparkSession object exists.
Verification: Check the terminal output for the **Spark Web UI URL** (e.g., http://hostname:4040).

Part 3: Load a Sample Dataset (Extraction)

6. Prepare Sample Data:

Action: Create a small text file (e.g., sample.txt) containing a few lines of text. Upload it to a local folder or HDFS.

Data Example:
Big Data is big.
Spark is faster than Big Data.

7. Load Data into an RDD/DataFrame:

Action: Use the SparkSession object (spark) to load the text file into a distributed dataset.
PySpark Command**:** Load as a basic **RDD** (Resilient Distributed Dataset).

Python Code:

```
# Load the text file from the HDFS or local file system
sample_rdd = spark.sparkContext.textFile("file:///path/to/sample.txt")
```

Part 4: Verify Basic Operations (Transformation & Action)

8. Verify Basic Transformation (map):

Action: Apply a simple **Transformation** (like map) to process each element in the RDD.
PySpark Command: Convert all lines to lowercase.

Python Code:

```
lowercase_rdd = sample_rdd.map(lambda line: line.lower())
```

9. Verify Basic Action (count):

Action: Apply a simple **Action** (like count) which triggers the actual distributed computation and returns a result to the driver.
PySpark Command: Count the number of lines in the file.

Python Code:

```
line_count = sample_rdd.count()
print("Total Lines: ", line_count)
```

10. Final Verification:

Action: Use the collect() action to view the final transformed data on the console.

Python Code:

```
print(lowercase_rdd.collect())
```

Expected Output: The count should match the number of lines, and the output of collect() should show the lines in lowercase, confirming Spark's distributed processing is active.

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the fundamental, fault-tolerant, and distributed collection abstraction (data structure) that Apache Spark is built upon?
2. Spark is often much faster than Hadoop MapReduce. Explain the primary difference in how Spark handles data processing that leads to this speed advantage.
3. The Spark method filter() is a Transformation, while count() is an Action. Briefly state what happens in the Spark cluster only when you call the count() method, but **not** when you call the filter() method.
4. If a student wants to launch the interactive Spark shell using the Python API, what is the single command they should execute in the terminal?

Space for Answer

……..

……..

……..

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

……

……

X References:

1. https://spark.apache.org/docs/latest/
2. https://onlinecourses.nptel.ac.in/noc20_cs92/preview
3. https://www.tutorialspoint.com/apache_spark/apache_spark_installation.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 20: *Perform Data Transformations with RDDs and apply map, filter, reduce operations.

I Practical Significance

This experiment reinforces the fundamental programming model of Apache Spark by using the core abstraction, Resilient Distributed Dataset (RDD). Students gain practical experience with Transformations (map, filter) and Actions (reduce), understanding the concepts of lazy evaluation and distributed execution which are foundational to efficient Spark development.

II Industry / Employer Expected Outcome(s)

Write core distributed logic using RDD primitives to process complex, low-level data structures that require custom partitioning or low-latency operations.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO20.1: Perform data transformations with RDDs in Spark.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

The RDD is an immutable, distributed collection of objects. Transformations (like map and filter) define a new RDD but do not execute immediately (lazy evaluation). Actions (like reduce, count, collect) trigger the distributed computation across the cluster. The goal is to perform a simplified Word Count or numerical analysis pipeline using these core operations.

Stepwise Procedure

This procedure demonstrates the use of map, filter, and reduce on a text dataset using **PySpark**.

Part 1: Load any Dataset into an RDD

1. Launch PySpark Session:
Action: Start the Spark interactive shell to gain access to the SparkContext (sc).
Command: pyspark

2. Prepare and Load Text Data:
Action: Load a text file (e.g., sentences.txt) into an RDD. The file should contain several lines of text.
PySpark Command:
Python Code:

```
# Load the text file from the local file system
text_rdd = sc.textFile("file:///path/to/sentences.txt")
print("Initial partitions: ", text_rdd.getNumPartitions())
```

Input Data Example:
Big data processing is fast.
Spark uses in-memory computing.

It is fun to code.

3. Initial Verification:
Action: Check the number of records loaded (an Action).
PySpark Command:

Python Code:
```
line_count = text_rdd.count()
print("Total lines loaded:", line_count)
```

Part 2: Apply Map, Filter, and Reduce Operations

4. Apply map Operation (Transformation):
Action: Use the flatMap transformation to split each line of text into individual words, creating a new RDD where each element is a single word.
Query: Convert all words to lowercase and tokenize the lines.

PySpark Command:
Python Code:
```
words_rdd = text_rdd.flatMap(lambda line: line.lower().split(" "))
print("Words RDD is defined (lazy evaluation).")
```

5. Apply filter Operation (Transformation):
Action: Use the filter transformation to select a subset of the data based on a condition (e.g., removing short, common words).
Query: Keep only words that have a length greater than 3 characters.

PySpark Command:
Python Code:
```
filtered_words_rdd = words_rdd.filter(lambda word: len(word) > 3)
print("Filtered RDD is defined (lazy evaluation).")
```

6. Apply reduce Operation (Action):
Action**:** Use the reduce action to perform an associative and commutative aggregation across all elements of the RDD.
Query: Concatenate all the remaining words into a single string (simple example of aggregation).

PySpark Command:
Python Code:
Note: For realistic analysis, reduceByKey is used, but for the basic reduce demonstration:

```
combined_string = filtered_words_rdd.reduce(lambda a, b: a + " " + b)
print("\n--- Final Output (Action Triggered) ---")
print("Combined Filtered String:", combined_string)
```

7. Final Verification (Collect):
Action: Alternatively, use collect() to bring all the elements of the final RDD back to the driver program for inspection.

PySpark Command:
Python Code:
```
print("Total number of resulting words:", filtered_words_rdd.count())
```

```
print("Sample final words (Collect):", filtered_words_rdd.collect()[:10])
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. RDD stands for Resilient Distributed Dataset. Name one of the two core properties (beginning with 'I' and 'F') that defines an RDD, making it suitable for fault-tolerant distributed computing.
2. Explain why executing the map operation in Spark does not immediately start processing data on the cluster workers. What command must be executed to force the processing?
3. Write the single line of PySpark code that uses the filter transformation on an existing RDD named sensor_data_rdd to keep only the temperature readings (column index 1) that are above 35.0.
4. An RDD named price_data contains tuples of (item_name, old_price). Write the map transformation that calculates a new RDD where the new price is 10% higher than the old price, keeping the item name.

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……..
……
……
……..
……
……..
……
……
……..
……
……..
……
……
……..
……
……..
……
……..
……
……..
……..
……..

……..

X References:

1. https://spark.apache.org/docs/latest/rdd-programming-guide.html
2. https://onlinecourses.nptel.ac.in/noc20_cs92/preview
3. https://www.tutorialspoint.com/apache_spark/apache_spark_rdd_operations.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 21: *Perform ETL Process on Big Data Using Spark.

I Practical Significance

This covers the complete ETL (Extract, Transform, Load) lifecycle using Spark Data Frames, the industry standard for structured data processing in Big Data. This is a critical skill for Data Engineers, demonstrating how Spark is used to clean, aggregate, and prepare raw data for downstream analysis or machine learning.

II Industry / Employer Expected Outcome(s)

Design and execute end-to-end data pipelines using Spark DataFrames to transform raw data into optimized, structured analytical tables.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO21.1: Execute an end-to-end ETL process using Spark.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Spark Data Frames are distributed collections of data organized into named columns, providing an optimized way to perform ETL compared to RDDs. The Catalyst Optimizer automatically creates an efficient execution plan for Data Frame operations. The ETL process involves: Extract (reading data), Transform (cleaning, aggregating, filtering), and Load (writing the result).

Stepwise Procedure

This procedure demonstrates a standard ETL process using **PySpark** DataFrames on a hypothetical sales dataset.

Part 1: Load Dataset from Local Storage or HDFS (Extract)

1. Start Spark Session:

Action: Launch the PySpark shell or an environment that initializes the SparkSession object, named spark.
Command: pyspark

2. Prepare and Load Raw Data:

Action: Load the raw data file (e.g., raw_sales.csv) into a Spark DataFrame. It is essential to infer the schema and read the header.

PySpark Command Example:

Python Code:

```
# Load data from HDFS or local file system
df_raw = spark.read.csv(
    "hdfs:///data/raw_sales.csv",#or "file:///local/path/raw_sales.csv"
    header=True,
    inferSchema=True
```

```
)
df_raw.printSchema()
df_raw.show(5)
```

Part 2: Apply Basic Transformations (Filtering and Aggregation) (Transform)

3. Apply Filtering Transformation (Data Cleansing):

Action: Filter the dataset to remove invalid records (e.g., transactions with a negative or zero amount, or null region).

Query: Filter out records where Amount is less than or equal to zero.

PySpark Command Example:

Python Code:

```
df_cleaned = df_raw.filter(df_raw.Amount > 0)
print("Filtered records:", df_raw.count() - df_cleaned.count())
```

4. Apply Aggregation Transformation (Data Summarization):

Action: Summarize the data by grouping records based on a dimension (e.g., Region) and calculating a metric (e.g., total sales).

Query: Calculate the total sum of Amount and the total Count of transactions per Region.

PySpark Command Example:

Python Code:

```
from pyspark.sql.functions import sum, count

df_summary = df_cleaned.groupBy("Region").agg(
   sum("Amount").alias("TotalSales"),
   count("*").alias("TotalTransactions")
)
df_summary.show()
```

5. Apply Ordering (Refinement):

Action: Sort the final results to present the data logically (e.g., highest sales first).

PySpark Command Example:

Python Code:

```
df_final = df_summary.orderBy("TotalSales", ascending=False)
```

Part 3: Store the Transformed Data (Load)

6. Store Transformed Data to Storage:

Action: Write the final, refined DataFrame (df_final) back to persistent storage. **Parquet format** is highly recommended for optimized columnar storage in Big Data environments.

Query: Write the DataFrame to an HDFS path in Parquet format, overwriting if the path exists.

PySpark Command Example:

Python Code:

```
# Load the data back into HDFS, overwriting the destination if it exists
df_final.write.mode("overwrite").parquet("hdfs:///data/processed/sales_summary_parquet
")
print("Transformed data loaded to HDFS.")
```

```
# Alternative: Write to local storage (e.g., for reporting)
# df_final.write.mode("overwrite").csv("file:///local/output/sales_summary.csv")
```

7. Final Verification (Optional):

Action: Create a new DataFrame by reading the saved data to confirm the load was successful and the schema is correct.

PySpark Command Example:

Python Code:

```
df_verify= spark.read.parquet("hdfs:///data/processed/sales_summary_parquet")
df_verify.show(5)
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the single main object (the entry point) that you must use to start all structured data operations, such as reading a CSV file into a DataFrame, in modern Spark?
2. When performing the Load step, it is recommended to save the data in Parquet format instead of CSV. Explain why Parquet is a better choice for storing data in a Big Data environment, based on its internal structure.
3. Write the single line of PySpark code that filters a DataFrame named df_sensors to keep only the records where the device_status column is exactly equal to the string 'FAILED'.
4. A DataFrame df_orders contains columns product_id and quantity. Write the PySpark code necessary to group this DataFrame by product_id and calculate the sum of all quantity for each product.

Space for Answer

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://spark.apache.org/docs/latest/sql-getting-started.html
2. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html
3. https://onlinecourses.nptel.ac.in/noc20_cs92/preview

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 22: Load Structured Dataset and Create temporary views in spark SQL.

I Practical Significance

This is crucial for bridging the gap between distributed data processing (Spark) and traditional relational analysis (SQL). Spark SQL allows analysts to use familiar SQL syntax to query vast datasets managed by Spark DataFrames. Creating a Temporary View enables direct, high-performance querying, which is essential for data exploration and reporting.

II Industry / Employer Expected Outcome(s)

Establish a performant data access layer by making distributed datasets accessible to BI tools and data analysts via standard SQL queries.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO22.1: Load structured data and create temporary views in SparkSQL.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Spark SQL is a Spark module for structured data processing. The core element is the DataFrame, which is optimized by the Catalyst Optimizer. A Temporary View is a short-lived, logical table definition over a DataFrame, accessible only within the current SparkSession. This allows data analysts to use standard SQL for querying distributed data, leveraging Spark's speed.

Stepwise Procedure

This procedure demonstrates the process using **PySpark** on a hypothetical customer dataset.

Part 1: Select and Load any Structured Dataset into Spark

1. Start Spark Session:

Action: Launch the PySpark shell, which automatically initializes the **SparkSession** object (spark).

Command: pyspark

2. Load the Structured Dataset:

Action: Use the spark.read API to load a structured file (e.g., customers.csv). Ensure the header is read and the schema is automatically detected (inferSchema=True).

PySpark Command Example:

Python Code:

```
# Load customer data from a file
df_customers = spark.read.csv(
    "file:///path/to/customers.csv",
    header=True,
    inferSchema=True
)
```

print("Dataset loaded into DataFrame.")

Part 2: Inspect the Dataset to Understand its Structure

3. Inspect Schema:

Action: Print the schema to verify that column names are correct and that Spark has inferred appropriate data types (e.g., INT, STRING, DOUBLE).

PySpark Command Example:

Python Code:

```
df_customers.printSchema()
# Expected: Root |-- id: integer (nullable = true)
```

4. Inspect Records:

Action: Display the first few rows of the DataFrame to confirm the data was parsed correctly (e.g., delimiters handled, no missing values in initial rows).

PySpark Command Example:

Python Code:

```
df_customers.show(5)
```

Part 3: Create a Temporary View from the Loaded Dataset

5. Create the Temporary View:

Action: Use the createOrReplaceTempView method on the DataFrame. This registers the DataFrame as a logical table that can be queried using standard SQL within the current session.

PySpark Command Example:

Python Code:

```
df_customers.createOrReplaceTempView("customer_data_view")
print("Temporary SQL view 'customer_data_view' created successfully.")
```

Part 4: Display Basic Records from the View to Verify Successful Creation

6. Query the View using Spark SQL:

Action: Use the spark.sql() command to execute a basic SQL query against the newly created temporary view.

Query: Select all columns and limit the output to the first 10 records.

PySpark Command Example:

Python Code:

```
# Execute SQL query against the temporary view
sql_result_df = spark.sql("SELECT CustomerID, Name, City FROM customer_data_view
LIMIT 10")
```

7. Display the Result:

Action: Show the resulting DataFrame to confirm that the SQL query executed successfully against the distributed data.

PySpark Command Example:

Python Code:

```
sql_result_df.show()
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is the specific PySpark object that is used to execute a pure SQL query (e.g., SELECT * FROM table_name) against a temporary view?
2. In Spark, why must you create a Temporary View from a DataFrame before you can use a pure SQL command on that data?
3. A DataFrame named df_inventory has been loaded. Write the single line of PySpark code to register this DataFrame as a SQL view named product_details.
4. After loading a dataset into a DataFrame named df_students, what two different methods would you call on this DataFrame to check (a) the columns and data types, and (b) the actual values of the first few rows?

Space for Answer

……..

……..

……..

……..

……..

……..

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……..

……

……

……

……..

……

……

……

……..

……

X References:

1. https://spark.apache.org/docs/latest/sql-getting-started.html#creating-dataframes
2. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.SparkSession.sql.html
3. https://www.tutorialspoint.com/apache_spark/apache_spark_sql.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 23: Perform Select, Join, and Group by Queries in Spark SQL.

I Practical Significance

This experiment solidifies the analyst's ability to perform complex data manipulation and aggregation on Big Data using Spark SQL. By executing SELECT, JOIN, and GROUP BY operations, students simulate real-world Business Intelligence (BI) tasks, directly addressing the core analytical needs.

II Industry / Employer Expected Outcome(s)

Generate complex analytical reports and business insights by querying and aggregating massive datasets using distributed SQL.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO23.1: Run after writing basic SQL queries on datasets using SparkSQL.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Spark SQL leverages the Catalyst Optimizer to efficiently process standard SQL queries against DataFrames registered as views. Complex queries often require combining data from multiple sources (JOIN), filtering results based on criteria (SELECT/WHERE), and calculating summary statistics (GROUP BY/Aggregate Functions). This process generates the core analytical reports needed for decision-making.

Stepwise Procedure

This procedure assumes two datasets have been loaded and registered as temporary views in a previous experiment: customer_data_view and order_data_view.

Part 1: Use the Temporary View Created from a Structured Dataset

1. Verify View Existence:

Action: Check that the necessary temporary views are available in the current SparkSession.

PySpark Command Example:

Python Code:

```
spark.sql("SHOW VIEWS").show()
# Ensure 'customer_data_view' and 'order_data_view' are listed.
```

Part 2: Execute SELECT Queries to Retrieve Specific Columns

2. Execute Basic Column Selection:

Action: Use a SELECT query with a WHERE clause to retrieve a filtered list of specific columns from one view.

Query: Select the customer ID, Name, and City for all customers located in 'Mumbai'.

PySpark Command Example:
SQL Code:
```
SELECT CustomerID, Name, City
FROM customer_data_view
WHERE City = 'Mumbai'
```

Part 3: Perform JOIN Operations Between Two Views or Datasets

3. Execute INNER JOIN:
Action: Join the two views (customer_data_view and order_data_view) based on a common key (CustomerID) to link customer details with their orders.
Query: Retrieve the customer name and the order amount for all orders.

PySpark Command Example:
SQL Code:
```
SELECT c.Name, o.OrderID, o.Amount
FROM customer_data_view c
INNER JOIN order_data_view o ON c.CustomerID = o.CustomerID
```

Part 4: Apply GROUP BY Queries to Aggregate Data

4. Execute Aggregation with GROUP BY:
Action: Use the joined result to calculate aggregated metrics based on a grouping dimension.
Query: Calculate the **Total Amount Spent (SUM)** and the **Average Order Value (AVG)** for each **City**.

PySpark Command Example:
Python Code:
```
spark.sql("""
   SELECT
      c.City,
      SUM(o.Amount) AS TotalCitySales,
      AVG(o.Amount) AS AverageOrderValue,
      COUNT(o.OrderID) AS TotalOrders
   FROM customer_data_view c
   INNER JOIN order_data_view o ON c.CustomerID = o.CustomerID
   GROUP BY c.City
   ORDER BY TotalCitySales DESC
""").show()
```

Part 5: Display and Save the Query Results

5. Execute and Capture Final Results:
Action: Run the final complex query and capture the resulting DataFrame object.

PySpark Command Example:
Python Code:
```
df_analytical_report = spark.sql("""
   SELECT c.City, SUM(o.Amount) AS TotalSales
   FROM customer_data_view c
```

```
    INNER JOIN order_data_view o ON c.CustomerID = o.CustomerID
    GROUP BY c.City
    ORDER BY TotalSales DESC
""")
```

6. Save the Transformed Report:

Action: Write the final analytical DataFrame to a persistent, optimized storage format (like Parquet) in HDFS for reporting or visualization tools.

PySpark Command Example:

Python Code:

```
df_analytical_report.write.mode("overwrite").parquet("hdfs:///reports/city_sales_summary.parquet")
print("Analytical Report saved to HDFS in Parquet format.")
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. In a standard SQL query, which clause must appear after the FROM clause but before the GROUP BY clause?
2. When Spark performs an INNER JOIN between two very large views (customer_data_view and order_data_view), explain the main job that the Spark cluster must do to ensure that matching rows (based on CustomerID) are brought together before the join logic can be applied.
3. Write the SQL query snippet that uses the GROUP BY result from Step 4 to filter the final output, showing only those cities where the TotalCitySales is greater than 50,000.
4. Write the single line of Spark SQL code that calculates the average (AVG) value of the column price from the view named product_view.

Space for Answer

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://spark.apache.org/docs/latest/sql-ref.html
2. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.SparkSession.sql.html
3. https://www.w3schools.com/sql/

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 24: Perform Real-Time Word Count Using Spark Streaming.

I Practical Significance

This introduces students to Spark Streaming, demonstrating its capability for real-time processing of high-velocity, continuous data streams. This technique is vital for applications like fraud detection, log monitoring, and live analytics, directly fulfilling the requirements of CO5 by applying Spark to complex scenarios.

II Industry / Employer Expected Outcome(s)

Develop real-time monitoring applications to ingest and process high-velocity data streams for immediate pattern detection and trending.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO24.1: Stream real- time text and count words using Spark streaming.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Spark Streaming (or the more modern Structured Streaming) processes data streams by dividing them into small batches (micro-batches) and processing them using the Spark engine. The core abstraction is the DStream (Discretized Stream) or a Streaming DataFrame. The Word Count example is the 'Hello World' of streaming, demonstrating the continuous cycle of receiving data, transforming it (splitting/grouping), and updating the count.

Stepwise Procedure

This procedure outlines the steps using PySpark (Structured Streaming recommended for modern Spark, but the DStream concept is used for clarity) alongside the essential Netcat utility for simulating the data stream.

Part 1: Connect to Socket or Stream Text Data

1. Set up the Data Stream Source (Netcat):

Action: Open a terminal window and start the netcat utility, which acts as a server sending continuous data over a TCP port (e.g., 9999).

Terminal Command: nc -lk 9999

Note: This terminal will be used to type lines of text that Spark will consume.

2. Start PySpark and Initialize Streaming Context:

Action: Launch the PySpark environment and define the source of the data as the network socket created in Step 1.

PySpark Command:

Python Code:

```
from pyspark.streaming import StreamingContext
# Initialize StreamingContext with batch interval (e.g., 5 seconds)
```

```
ssc = StreamingContext(spark.sparkContext, 5)

# Create a DStream that will connect to the server (Netcat)
lines = ssc.socketTextStream("localhost", 9999)
print("DStream initialized and connected to socket.")
```

Part 2: Process Incoming Lines

3. Split Lines into Words (Transformation):

Action: Use the RDD flatMap transformation, applied to the DStream, to split each incoming line into individual words.

Query: Flatten the lines, and ensure words are handled uniformly (e.g., convert to lowercase).

PySpark Command:

Python Code:

```
# Apply flatMap to split each line into words
words = lines.flatMap(lambda line: line.split(" "))
```

Part 3: Count Word Frequency in Real-Time

4. Map Words to Pairs:

Action: Prepare the words for aggregation by mapping each word to a key-value pair of (word, 1).

PySpark Command:

Python Code:

```
# Map each word to a (word, 1) pair
pairs = words.map(lambda word: (word, 1))
```

5. Calculate Running Count (Aggregation):

Action: Use the reduceByKeyAndWindow or simply reduceByKey (in a simple streaming context) to aggregate the counts for each unique word across the batches.

Query: Sum the '1's for all identical word keys.

PySpark Command:

Python Code:

```
# Sum the counts for each key (word)
wordCounts = pairs.reduceByKey(lambda x, y: x + y)
print("Word Count DStream defined.")
```

Part 4: Display Live Results

6. Define Output Action:

Action: Use the pprint() action to print the results of the wordCounts DStream to the console every batch interval.

PySpark Command:

Python Code:

```
# Print the running counts to the console
wordCounts.pprint()
```

7. Start and Monitor the Stream:

Action: Start the StreamingContext and keep the process running until explicitly terminated. The cluster will now actively process any input received by Netcat.

PySpark Command:
Python Code:
ssc.start()
ssc.awaitTermination() # Keeps the job running indefinitely

8. Verification:
Action: Go back to the Netcat terminal and type sentences. Observe the Spark terminal to see the word counts update in real-time every 5 seconds.

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Netcat Utility	**nc** command-line tool (often pre-installed on Linux distributions). Acts as the data source (TCP Socket Server).	1	

VIII Conclusion

……..

…….

…….

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the core data structure abstraction in classic Spark Streaming that represents a continuous stream of data as a sequence of small, fault-tolerant RDDs? (It begins with 'D').
2. Explain in simple terms how Spark Streaming achieves a "real-time" feel, even though it is not true, continuous, per-event processing.
3. To successfully run a Spark Streaming job, the student must have the correct package dependencies. What are the two necessary packages/modules that must be imported at the beginning of the PySpark script to run the socket connection (Step 2)?
4. A DStream named lines is created. If the goal is to count the frequency of each unique word,

what is the first transformation that must be applied to lines?

Space for Answer

……..
……
……
……..
……
……
……..
……
……

……

……..

……

……

……..

……

……

……..

X References:

1. https://spark.apache.org/docs/latest/streaming-programming-guide.html
2. https://www.tutorialspoint.com/apache_spark/apache_spark_streaming.htm
3. https://linux.die.net/man/1/nc

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 25: Store Real-Time Streamed Data from Spark streaming into HDFS.

I Practical Significance

This is critical as it completes the real-time data ingestion pipeline. It teaches students how to use Spark Streaming not just for immediate analysis, but for persistent storage in HDFS. This process is vital in the industry for tasks like archiving log data, creating a centralized data lake, and ensuring data durability for future batch analysis.

II Industry / Employer Expected Outcome(s)

Build a robust data ingestion and archiving mechanism for a centralized Data Lake, ensuring the durability and fault tolerance of streaming data.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO25.1: Save after extracting streaming data into persistent storage using Spark Streaming.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

The process involves treating the continuous stream as a sequence of RDDs (DStreams) and applying a persistent Output Operation at the end of each micro-batch interval. The saveAsTextFiles() operation is the primary mechanism to write the data from each batch's RDD to unique, time-stamped directories in HDFS. This ensures no data is lost and provides a fault-tolerant way to archive the stream.

Stepwise Procedure

This procedure demonstrates the persistent archiving of stream data using **PySpark** and **Netcat**.

Part 1: Connect to a Text Stream (Capture Real-Time Data)

1. Set up the Data Stream Source (Netcat):

Action: Start the netcat utility on a designated port (e.g., 9998) to simulate the continuous influx of data (e.g., system logs).

Terminal Command: nc -lk 9998

2. Start PySpark and Initialize Streaming Context:

Action: Launch the PySpark environment and initialize the StreamingContext (ssc) with a defined batch interval (e.g., 10 seconds).

PySpark Command:

Python Code:

```
from pyspark.streaming import StreamingContext
# Initialize StreamingContext with a 10-second batch interval
ssc = StreamingContext(spark.sparkContext, 10)
# Connect DStream to the Netcat server
log_stream = ssc.socketTextStream("localhost", 9998)
print("DStream connected and ready to receive data...")
```

Part 2: Capture Real-Time Data Using Spark Streaming

3. Apply Simple Transformation (Data Cleaning):

Action: Apply a minor transformation to the incoming stream to demonstrate processing before storage (e.g., filter out blank lines or add a timestamp).

Query: Filter the stream to remove any empty lines.

PySpark Command:

Python Code:

```
# Filter out empty lines from the stream
cleaned_stream = log_stream.filter(lambda line: len(line.strip()) > 0)
```

Part 3: Write the Incoming Stream to HDFS in Regular Intervals

4. Define Output Directory:

Action: Define the root directory in HDFS where the stream data will be archived.

HDFS Command Example:

```
hdfs dfs -mkdir -p /data/stream_archive/logs
```

5. Apply HDFS Output Operation:

Action: Use the saveAsTextFiles() DStream output operation. Spark will create a new, time-stamped sub-directory within the base path for every 10-second batch and write the data there.

PySpark Command:

Python Code:

```
# Save the content of each RDD in the DStream to HDFS
# The base path will be followed by a time-stamp suffix (e.g., base_path-1609459200000)
base_path = "hdfs:///data/stream_archive/logs/batch"
cleaned_stream.saveAsTextFiles(base_path)
print("Output operation defined: saving to HDFS every 10 seconds.")
```

6. Start and Verify Archiving:

Action: Start the streaming job and then use the HDFS shell command from a separate terminal to monitor the creation of new subdirectories.

PySpark Command: ssc.start()

HDFS Monitoring Command (Run in a separate terminal):

```
hdfs dfs -ls /data/stream_archive/logs/
```

Verification: Type data into the Netcat terminal. Every 10 seconds, a new directory should appear in the HDFS output location, containing the data received in that batch.

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Netcat Utility	**nc** command-line tool (often pre-installed on Linux distributions). Acts as the data source (TCP Socket Server).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the specific Spark Streaming DStream output function that is used to write the contents of each micro-batch (RDD) to HDFS?
2. If the Spark streaming application fails after processing the data in a batch but *before* saving it to HDFS, how does the underlying mechanism of the DStream ensure that this data is not permanently lost?
3. If you set the base path to hdfs:///archive/stream and the batch interval is 5 seconds, what is the key difference between the directory created by Spark in HDFS for the batch starting at 10:00:00 and the one created for the batch starting at 10:00:05?

Space for Answer

……..

……..

……..

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

X References:

1. https://spark.apache.org/docs/latest/streaming-programming-guide.html#output-operations
2. https://hadoop.apache.org/docs/r3.3.0/hadoop-project-dist/hadoop-common/FileSystemShell.html
3. https://onlinecourses.nptel.ac.in/noc20_cs92/preview

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 26: *Apply Simple Regression on Large Dataset in Spark.

I Practical Significance

This experiment is a fundamental introduction to distributed Machine Learning (ML) using Spark MLlib. Students learn how to apply Simple Linear Regression a core predictive modeling technique on a large dataset. This capability is essential for tasks like trend forecasting, price prediction, and analyzing linear relationships in big data.

II Industry / Employer Expected Outcome(s)

Build and deploy basic predictive models to forecast numerical outcomes, such as sales trends or resource needs, using distributed machine learning.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO26.1: Run after creating a regression model on a large dataset in Spark.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Spark MLlib is the machine learning library for Spark. Simple Linear Regression models the relationship between a single independent feature and a dependent label by fitting a linear equation. Before training, Spark MLlib requires all features to be combined into a single vector column using the VectorAssembler transformer. The process involves splitting data into training and testing sets, training the model, and then evaluating its performance (e.g., using RMSE - Root Mean Squared Error).

Stepwise Procedure

This procedure uses **PySpark MLlib** DataFrames to build a regression model.

Part 1: Load any Numerical Dataset and Prepare Data

1. Load the Numerical Dataset:

Action: Load a structured dataset (e.g., containing house size and price) into a Spark DataFrame.

PySpark Command Example:

Python Code:

```
df_data = spark.read.csv(
    "file:///path/to/house_prices.csv",
    header=True,
    inferSchema=True
).select("Size_SqFt", "Price") # Select the feature and the label
df_data.printSchema()
```

2. Build the Feature Vector:

Action: MLlib requires all features to be in a single vector column. Use

VectorAssemblerto combine the feature column (Size_SqFt) into a new column named features.

PySpark Command Example:
Python Code:

```
from pyspark.ml.feature import VectorAssembler

# Define the assembler to group the input feature column(s)
assembler = VectorAssembler(
    inputCols=["Size_SqFt"],
    outputCol="features"
)

# Apply the transformation to the DataFrame
df_assembled = assembler.transform(df_data)
df_assembled.show(5, truncate=False)
```

Part 2: Build and Train a Simple Regression Model

3. Split Data into Training and Testing Sets:

Action: Divide the assembled dataset into two parts: a large portion for training the model and a smaller portion for testing its accuracy.

PySpark Command Example:
Python Code:

```
# Split the data 70% for training, 30% for testing
(trainingData, testData) = df_assembled.randomSplit([0.7, 0.3], seed=1234)
print(f"Training size: {trainingData.count()}, Test size: {testData.count()}")
```

4. Define the Regression Model:

Action: Instantiate the **LinearRegression** algorithm from the MLlib library. The label column must be specified.

PySpark Command Example:
Python Code:

```
from pyspark.ml.regression import LinearRegression

# Define the algorithm, specifying the label and feature columns
lr = LinearRegression(featuresCol='features', labelCol='Price')
print("Linear Regression model defined.")
```

Part 3: Train and Apply the Model

5. Train the Model:

Action: Use the fit() method on the training data. This is the computationally intensive step where the distributed algorithm calculates the optimal line (coefficients).

PySpark Command Example:
Python Code:

```
# Train the model to find the best-fit line
lrModel = lr.fit(trainingData)
print("Model training complete.")
```

6. Evaluate the Model (Optional but Recommended):

Action: Apply the trained model to the testing data to generate predictions, and calculate

metrics like RMSE.

PySpark Command Example:
Python Code:
```
# Generate predictions on the test set
predictions = lrModel.transform(testData)
predictions.select("Price", "prediction", "features").show(10)
```

Part 4: Predict Target Values

7. Extract Model Metrics and Coefficients:

Action: Display the learned model parameters (intercept and coefficient).

PySpark Command Example:
Python Code:
```
print(f"Intercept: {lrModel.intercept}")
print(f"Coefficient: {lrModel.coefficients[0]}")
```

8. Use the Model for Prediction:

Action: Use the trained model (lrModel) to make a final prediction on a new, unseen data point (or the testData set).
Query: The final output is the prediction column alongside the actual Price.

PySpark Command Example:
Python Code:
```
# Calculate performance metrics on the test data
summary = lrModel.evaluate(testData)
print(f"Root Mean Squared Error (RMSE) on test data: {summary.rootMeanSquaredError}")
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	

VIII Conclusion

……..

…….

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. In Spark MLlib, what is the name of the special column (created by the VectorAssembler) that must contain all the numerical input features before the training algorithm can be run?
2. Explain the fundamental difference between a Regression problem (like the one performed here) and a Classification problem based on the type of value the model is designed to predict.
3. A Linear Regression algorithm is defined as lr_model. You have prepared a DataFrame named final_training_set. Write the single line of PySpark code required to execute the training process and store the result in a variable named trained_model.
4. If you have an input DataFrame with columns A, B, and C, and you want to use only columns A andC as features for the regression model, write the PySpark code to define the VectorAssembler for this purpose.

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……..
……
……
……

X References:

1. https://spark.apache.org/docs/latest/ml-linear-regression.html
2. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.VectorAssembler.html
3. https://www.tutorialspoint.com/apache_spark/apache_spark_machine_learning.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 27: *Apply K-Means Clustering on any Dataset using Spark MLlib.

I Practical Significance

This experiment introduces students to Unsupervised Machine Learning using Spark MLlib. K-Means Clustering is a foundational technique used widely in industry for tasks like market segmentation, anomaly detection, and customer profiling. By applying this algorithm, students learn to discover inherent groupings and patterns within large datasets without prior labeled data.

II Industry / Employer Expected Outcome(s)

Segment customer bases or categorize products automatically to discover hidden patterns and inform targeted marketing or operational strategies.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO27.1: Apply the K-Means clustering algorithm in Spark MLlib.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

K-Means is an iterative, partitioning clustering algorithm that aims to divide N data points into *K* clusters. It minimizes the Within-Cluster Sum of Squares (WCSS). As with regression, Spark MLlib requires features to be combined into a single vector column using VectorAssembler. The key step is defining the number of clusters (*K*), which is determined beforehand (e.g., based on domain knowledge or methods like the Elbow Method).

Stepwise Procedure

This procedure uses **PySpark MLlib** DataFrames to perform K-Means clustering on a hypothetical customer segmentation dataset.

Part 1: Select and Load Dataset and Prepare Features

1. Load the Structured Dataset:

Action: Load a dataset (e.g., customer_data.csv) containing numerical features suitable for clustering (e.g., Annual Income, Spending Score) into a Spark DataFrame.

PySpark Command Example:

Python Code:

```
df_data = spark.read.csv(
    "file:///path/to/customer_data.csv",
    header=True,
    inferSchema=True
).select("CustomerID", "Annual_Income", "Spending_Score") # Select ID and features
df_data.show(5)
```

2. Build the Feature Vector:

Action: Use **VectorAssembler** to combine all relevant numerical feature columns (Annual_Income, Spending_Score) into a single vector column named features.

PySpark Command Example:
Python Code:

```
from pyspark.ml.feature import VectorAssembler

# Group the two selected features into a single vector
assembler = VectorAssembler(
    inputCols=["Annual_Income", "Spending_Score"],
    outputCol="features"
)

# Apply the transformation
df_assembled = assembler.transform(df_data)
df_assembled.show(5, truncate=False)
```

Part 2: Apply the K-Means Clustering Algorithm

3. Define the K-Means Model:

Action: Instantiate the **KMeans** algorithm from MLlib and specify the desired number of clusters, **K** (e.g., *K*=5).

PySpark Command Example:
Python Code:

```
from pyspark.ml.clustering import KMeans

K = 5 # Define the desired number of clusters
kmeans = KMeans(featuresCol='features', k=K)
print(f"K-Means model defined with K={K}.")
```

4. Train the Model:

Action: Use the fit() method on the assembled DataFrame. The distributed algorithm will run iterations to find the optimal cluster centers (centroids).

PySpark Command Example:
Python Code:

```
# Train the model
model = kmeans.fit(df_assembled)
print("K-Means training complete.")
```

Part 3: Output the Cluster Assignments for Each Data Point

5. Apply the Model and Get Assignments:

Action: Use the transform() method of the trained model to assign each data point in the original dataset to one of the *K* clusters. A new column, typically named prediction, is added with the cluster ID.

PySpark Command Example:
Python Code:

```
# Generate cluster assignments (predictions)
clustered_data = model.transform(df_assembled)
```

6. Inspect Cluster Centers and Assignments:

Action: Display the coordinates of the calculated cluster centers and show a sample of the data with their assigned cluster IDs.

PySpark Command Example:

Python Code:

```
# Show the coordinates of the calculated cluster centers
centers = model.clusterCenters()
print("Cluster Centers (Coordinates):")
for center in centers:
    print(center)

# Show data points with their assigned cluster ID (prediction column)
clustered_data.select("CustomerID", "Annual_Income", "Spending_Score", "prediction").show(10)
```

Part 4: Save the Clustering Result for Further Use

7. Save the Cluster Assignments:

Action: Save the resulting DataFrame, which now contains the original data plus the cluster assignment column (prediction), to HDFS in an optimized format (e.g., Parquet).

PySpark Command Example:

Python Code:

```
# Write the final result (data + cluster assignment) to HDFS
clustered_data.write.mode("overwrite").parquet("hdfs:///ml_output/customer_segments_k5.parquet")
print("Clustering results saved to HDFS.")
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Hardware (Cluster/VM)	A single Virtual Machine or cluster node with Spark configured (minimum 8GB RAM recommended for MLlib processing).	1	

VIII Conclusion

……..

……

……

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. K-Means is a form of Unsupervised Learning. What is the key characteristic of the input dataset that defines Unsupervised Learning, distinguishing it from Supervised Learning?
2. If the K-Means algorithm needs to calculate the distance between data points, explain why the VectorAssembler step is critical for combining multiple feature columns before training the model.
3. A K-Means model is defined as kmeans_algo with K=3. You have prepared a DataFrame named prepped_data. Write the single line of PySpark code required to execute the training process and store the resulting model in a variable named final_model.
4. After the K-Means model is trained and the transform() method is applied to the data, what is the default name of the new column added to the DataFrame that contains the assigned cluster ID for each data point?

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

……

……

……

……

……

……

……

X References:

1. https://spark.apache.org/docs/latest/ml-clustering.html#k-means
2. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.clustering.KMeans.html
3. https://www.tutorialspoint.com/apache_spark/apache_spark_machine_learning.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 28: *Visualize K-Means Clustering Results Using Spark MLlib.

I Practical Significance

This experiment is crucial for the interpretation and validation of the K-Means model trained in the previous practical. Visualization allows students to qualitatively assess the effectiveness of the clustering algorithm, visually confirm the separation of the groups (e.g., customer segments), and understand the characteristics of each cluster a critical step in any data science workflow.

II Industry / Employer Expected Outcome(s)

Communicate complex model results effectively by creating clear, visual representations of data groupings and clusters for non-technical stakeholders.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO28.1: Visualize clustering results of a dataset processed with SparkMLlib.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

While Spark is optimized for distributed computation, visualization is typically a single-machine task performed by the driver program using libraries like Matplotlib or Seaborn. The process requires converting the final clustered Spark DataFrame into a local Pandas DataFrame (.toPandas()) for plotting. The resulting Scatter Plot uses the two selected features as axes, and the color of each point represents the cluster ID (prediction column).

Stepwise Procedure

This procedure integrates PySpark with the Matplotlib visualization library.

Part 1: Load the Cluster Assignment Results

1. Start PySpark Session and Import Libraries:

Action: Launch the PySpark environment and import the necessary libraries for data processing and visualization.

PySpark Command Example:

Python Code:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd # Used for the intermediate conversion

# Load the clustered data from the previous experiment (Parquet format)
df_clustered = spark.read.parquet("hdfs:///ml_output/customer_segments_k5.parquet")
df_clustered.printSchema()
```

2. Select and Prepare Data for Plotting:

Action: Select only the essential columns (the two input features and the cluster

assignment) needed for the 2D plot.

PySpark Command Example:
Python Code:
```
# Select the features used for clustering and the prediction column
df_plot_data = df_clustered.select("Annual_Income", "Spending_Score", "prediction")
```

Part 2: Use Visualization Tools and Plot Clusters

3. Convert to Local Pandas DataFrame:

Action: Since Matplotlib runs locally, the distributed Spark DataFrame must be converted to a local Pandas DataFrame using the .toPandas() action.

PySpark Command Example:
Python Code:
```
# WARNING: This step should only be done on reasonably sized results
# (e.g., typically < 1 million records for local processing)
pd_plot_data = df_plot_data.toPandas()
print("Data converted to local Pandas DataFrame for visualization.")
```

4. Plot the Clusters (Scatter Plot):

Action: Use Matplotlib or Seaborn to generate a scatter plot using the two features as the X and Y axes.

PySpark Command Example:
Python Code:
```
# Ensure 'prediction' is treated as a categorical variable for color-coding
plt.figure(figsize=(10, 6))

# Use Seaborn's scatterplot with 'hue' set to the cluster ID ('prediction')
sns.scatterplot(
    x='Annual_Income',
    y='Spending_Score',
    hue='prediction',
    data=pd_plot_data,
    palette='viridis',
    legend='full',
    s=80 # size of points
)
```

Part 3: Color-Code and Export the Visualization

5. Add Labels and Title (Color-Code Verification):

Action: Add a title, axis labels, and a legend to interpret the visualization accurately. The hue parameter in Seaborn automatically color-codes the points based on the cluster ID (prediction).

PySpark Command Example:
Python Code:
```
plt.title('K-Means Customer Segmentation (K=5)')
plt.xlabel('Annual Income (Feature X)')
plt.ylabel('Spending Score (Feature Y)')
plt.grid(True)
```

6. Export the Visualization:

Action: Save the generated plot to a file (e.g., PNG) on the local file system for inclusion in the final lab report.

PySpark Command Example:
Python Code:
```
# Save the plot to the local file system
plt.savefig('kmeans_cluster_visualization.png')
plt.show() # Display the plot (if running in an interactive notebook/environment)
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Hardware (Cluster/VM)	A single Virtual Machine or cluster node with Spark configured (minimum 8GB RAM recommended for MLlib processing).	1	
7.	Visualization Libraries	Matplotlib and Seaborn (Must be installed on the machine running the PySpark driver program).	1	
8.	Data Manipulation Library	Pandas (Required for converting the Spark DataFrame to a local data structure for plotting).	1	

VIII Conclusion

……..

……

……

IX Practical related questions

1. What is the most common and standard Python library used in data analysis for plotting and creating visualizations, often used alongside Pandas for local plotting?
2. Explain the main reason why the .toPandas() method is required in this procedure before the

visualization code can be executed (i.e., why can't Matplotlib plot the Spark DataFrame directly)?

3. If the cluster assignment results DataFrame is named df_results and you want to plot the features Age (X-axis) and Salary (Y-axis), write the Seaborn code to create the scatter plot using the prediction column for color-coding.
4. After plotting the clusters, if you observe that Cluster 1 is located in the top-right section of the graph (high X-axis and high Y-axis), what specific characteristics (in terms of Annual Income and Spending Score) would you assign to the customers in this segment?

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……

……..

……

……

……

……

……

……..

……

……

……

……

……

X References:

1. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.toPandas.html
2. https://matplotlib.org/stable/gallery/index.html
3. https://seaborn.pydata.org/tutorial/regression.html

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 29: Apply Decision Tree Classification on any Dataset using Spark MLlib.

I Practical Significance

This experiment introduces students to Supervised Machine Learning for classification problems using Spark MLlib. The Decision Tree Classification algorithm is intuitive, powerful, and easy to interpret, making it an excellent tool for tasks like predicting outcomes (e.g., loan risk, customer churn, medical diagnosis). By performing this practical, students apply Spark to solve a core predictive analytics challenge.

II Industry / Employer Expected Outcome(s)

Develop automated classification systems to assess risk, approve applications, or categorize inputs based on large feature sets.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO29.1: Classify a dataset using the Decision Tree algorithm in SparkMLlib.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

Decision Tree Classification creates a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Spark MLlib requires the label column to be numerical (indexed), which is often achieved using StringIndexer for categorical labels. The process involves assembling features, indexing the label, training the model on a portion of the data, and using the trained model to predict the class label for new data points.

Stepwise Procedure

This procedure uses **PySpark MLlib** DataFrames to build a Decision Tree Classifier on a hypothetical loan approval dataset.

Part 1: Select and Load Dataset and Preprocess It

1. Load the Labeled Dataset:

Action: Load a structured dataset (e.g., loan_data.csv) containing features and a binary label (e.g., 'Approved'/'Denied') into a Spark DataFrame.

PySpark Command Example:

Python Code:

```
df_data = spark.read.csv(
    "file:///path/to/loan_data.csv",
    header=True,
    inferSchema=True
).select("CreditScore", "Income", "LoanStatus") # Select features and label
df_data.printSchema()
```

2. Index the Categorical Label:

Action: The Decision Tree algorithm requires the categorical label column (LoanStatus)

to be converted into a numerical index (e.g., 0 and 1). Use **StringIndexer** for this.

PySpark Command Example:
Python Code:
```
from pyspark.ml.feature import StringIndexer

indexer = StringIndexer(inputCol="LoanStatus", outputCol="label")
df_indexed = indexer.fit(df_data).transform(df_data)
df_indexed.show(5)
```

3. Build the Feature Vector:

Action: Use **VectorAssembler** to combine the numerical feature columns (CreditScore, Income) into the required single vector column named features.

PySpark Command Example:
Python Code:
```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(
    inputCols=["CreditScore", "Income"],
    outputCol="features"
)
df_assembled = assembler.transform(df_indexed)
df_final = df_assembled.select("features", "label")
```

Part 2: Apply Decision Tree Classification Algorithm to Train the Model

4. Split Data into Training and Testing Sets:

Action: Divide the preprocessed dataset into training and testing subsets to validate the model's accuracy on unseen data.

PySpark Command Example:
Python Code:
```
# Split the data 80% for training, 20% for testing
(trainingData, testData) = df_final.randomSplit([0.8, 0.2], seed=42)
```

5. Define and Train the Classifier:

Action: Instantiate the **DecisionTreeClassifier** and use the fit() method on the training data.

PySpark Command Example:
Python Code:
```
from pyspark.ml.classification import DecisionTreeClassifier

# Define the algorithm, specifying the feature and indexed label columns
dt = DecisionTreeClassifier(featuresCol='features', labelCol='label', maxDepth=5)

# Train the model
dtModel = dt.fit(trainingData)
```

```
print("Decision Tree Model training complete.")
```

Part 3: Generate Predictions for the Dataset

6. Apply the Model to Generate Predictions:

Action: Use the transform() method of the trained model to generate predictions (prediction column) for the test dataset.

PySpark Command Example:

Python

```
# Generate predictions on the test set
predictions = dtModel.transform(testData)
predictions.select("features", "label", "prediction").show(10)
```

7. Evaluate Model Accuracy (Optional but Recommended):

Action: Use the **MulticlassClassificationEvaluator** to calculate metrics like accuracy.

PySpark Command Example:

Python Code:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(
    labelCol="label",
    predictionCol="prediction",
    metricName="accuracy"
)
accuracy = evaluator.evaluate(predictions)
print(f"Model Accuracy on Test Data: {accuracy:.4f}")
```

Part 4: Save the Prediction Results for Further Processing

8. Save the Predictions:

Action: Write the DataFrame containing the original features, the true label, and the predicted label to persistent storage (e.g., Parquet in HDFS) for reporting or auditing.

PySpark Command Example:

Python Code:

```
# Select key columns and save the result
predictions.select("features", "label", "prediction", "probability").write.mode("overwrite").parquet("hdfs:///ml_output/dt_predictions_loan_approval.parquet")
print("Decision Tree prediction results saved to HDFS.")
```

VII Recourses Required:

Sr.No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Hardware (Cluster/VM)	A single Virtual Machine or cluster node with Spark configured (minimum 8GB RAM recommended for MLlib processing).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. Decision Tree Classification is a form of Supervised Learning. What specific requirement (related to the dataset) must be met for a problem to be solvable using Supervised Learning?
2. Explain the necessity of the StringIndexer step in this procedure when the original label column (LoanStatus) contains non-numerical strings (like 'Approved' or 'Denied').
3. A categorical feature named Gender needs to be indexed into a new column named Gender_Index. Write the PySpark code to define and apply the StringIndexer to a DataFrame named df_raw.
4. After running the dtModel.transform(testData) command, what is the default name of the new column added to the DataFrame that holds the final predicted class label (e.g., 0 or 1)?

Space for Answer

……..

……..

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..
……
……
……..
……
……
……..
……
……

……

……

……..

……

……

……

……..

……

……

……

……..

X References:

1. https://spark.apache.org/docs/latest/ml-classification-regression.html#decision-tree-classifier
2. https://spark.apache.org/docs/latest/ml-features.html
3. https://www.tutorialspoint.com/apache_spark/apache_spark_classification.htm

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	

Practical No. 30: Display Classification Results from Decision Tree Model in Spark MLlib.

I Practical Significance

This experiment closes the loop on the Decision Tree Classification task by focusing on the crucial step of result analysis and communication. Students learn to display, summarize, and interpret the model's predictions, comparing them against the true labels. This step is essential for assessing model quality and providing actionable insights for business users.

II Industry / Employer Expected Outcome(s)

Validate and report model accuracy and performance using essential metrics (like confusion matrices), ensuring the deployed model meets business requirements.

III Course Level Learning Outcomes(s)

CO5: Use Spark to process and analyze big data in real-time or archives.

IV Laboratory Learning Outcome(s)

LLO30.1: Display classification outputs from Decision Tree results.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

The final step in a classification pipeline is to analyze the DataFrame containing the original data, the true labels, and the prediction column generated by the model. Simple analysis involves counting the frequency of the predicted classes (e.g., how many loans were predicted 'Approved' vs. 'Denied'). This is achieved using DataFrame aggregation methods (groupBy and count) and then saving the final, enriched dataset for downstream use.

Stepwise Procedure

This procedure uses PySpark to analyze the Decision Tree results saved from the previous experiment (Practical No. 29).

Part 1: Load the Saved Classification Results

1. Load the Prediction Results:

Action: Start the PySpark session and load the Parquet file containing the saved predictions from the Decision Tree model. This DataFrame includes the original features, the numerical true label (label), and the predicted numerical label (prediction).

PySpark Command Example:

Python Code:

```
# Load the prediction results DataFrame from HDFS
df_results =
spark.read.parquet("hdfs:///ml_output/dt_predictions_loan_approval.parquet")
print("Prediction results loaded.")
```

Part 2: Display the Predicted Class Labels Along Side the Original Data

2. Convert Numerical Labels Back to Strings (Interpretation):

Action: Use the **IndexToString** transformer to convert the numerical label and prediction columns back into human-readable strings (e.g., 0 -> 'Denied', 1 -> 'Approved'). This step is crucial for clear communication.

PySpark Command Example:
Python Code:

```
from pyspark.ml.feature import IndexToString

# Assuming 'label' and 'prediction' are indexed (0, 1) and derived from 'LoanStatus'
# We use a placeholder for the labels; in reality, we'd retrieve the original mapping from StringIndexer metadata.
label_converter = IndexToString(
    inputCol="label",
    outputCol="True_LoanStatus",
    labels=['Approved', 'Denied'] # Example mapping
)
pred_converter = IndexToString(
    inputCol="prediction",
    outputCol="Predicted_LoanStatus",
    labels=['Approved', 'Denied']
)

# Apply both converters to the results DataFrame
df_display = label_converter.transform(df_results)
df_display = pred_converter.transform(df_display)
```

3. Display the Results:
Action: Show a sample of the enriched DataFrame, displaying the original feature vectors, the true status, and the model's predicted status.

PySpark Command Example:
Python Code:

```
df_display.select("features","True_LoanStatus", "Predicted_LoanStatus").show(15, truncate=False)
```

Part 3: Use Simple Metrics (such as counts of predicted classes) to Observe Results

4. Count Predicted Classes:
Action: Calculate the distribution of the predicted outcomes to understand the model's overall tendency (e.g., how many applicants were predicted 'Approved').

PySpark Command Example:
Python Code:

```
print("\n--- Predicted Class Distribution ---")
df_display.groupBy("Predicted_LoanStatus").count().orderBy("count", ascending=False).show()
```

5. Count True vs. Predicted Matches (Accuracy Check):
Action: Create a contingency table (Confusion Matrix) to see how many predictions were correct/incorrect.

PySpark Command Example:
Python Code:

```
print("\n--- Confusion Matrix (True vs. Predicted) ---")
df_display.groupBy("True_LoanStatus").pivot("Predicted_LoanStatus").count().show()
```

Part 4 & 5: Save the Final Classified Dataset and Generate Summary

6. Save the Final Classified Dataset:

Action: Save the fully enriched DataFrame, which now contains both the original data and the human-readable prediction results, to a final reporting location.

PySpark Command Example:

Python Code:

```
df_display.write.mode("overwrite").csv("hdfs:///reports/final_loan_classification_report.csv", header=True)
print("\nFinal classification report saved to HDFS as CSV.")
```

7. Generate a Basic Summary of the Output:

Action: Re-run the evaluation metric calculation or display a brief summary of the accuracy achieved.

PySpark Command Example:

Python Code:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(df_results) # Use the original df_results for metric calculation

print("\n--- Model Performance Summary ---")
print(f"Total Records Analyzed: {df_results.count()}")
print(f"Overall Classification Accuracy: {accuracy:.4f}")
```

VII Recourses Required:

Sr. No.	Name of Resource	Broad Specification	Quantity	Remarks
1.	Operating System	Linux Distribution (e.g.,Ubuntu, CentOS, Red Hat)	1	
2.	Apache Hadoop	Version 2.x or 3.x (Fully operational HDFS and YARN)	1	
3.	Apache Spark	Pre-built binary package (e.g., Spark 3.x for Hadoop 3.3).	1	
4.	Java Development Kit (JDK)	OpenJDK or Oracle JDK, Version 8 or higher.	1	
5.	Python	InterpreterPython 3.x (Required for the PySpark shell).	1	
6.	Hardware (Cluster/VM)	A single Virtual Machine or cluster node with Spark configured (minimum 8GB RAM recommended for MLlib processing).	1	

VIII Conclusion

……..

……..

……..

IX Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questionsto ensure the achievement of identified CO.

1. What is the specific Spark MLlib transformer used to convert the numerical cluster labels (0 and 1) back into their original string values (e.g., 'Approved' and 'Denied') for reporting purposes?
2. In the Confusion Matrix (Step 5), explain what the values in the cell where the row is True_LoanStatus='Approved' and the column is Predicted_LoanStatus='Denied' represent in terms of model error.
3. Write the PySpark DataFrame operation chain (starting from df_display) required to count how many records were assigned to each unique value in the Predicted_LoanStatus column.
4. If the final report DataFrame df_display needs to be easily opened and read by business analysts using Microsoft Excel, what is the best format to use when calling the write.csv() method, and which parameter must be set to True?

Space for Answer

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

X References:

1. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.IndexToString.html
2. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.groupBy.html
3. https://spark.apache.org/docs/latest/ml-evaluation.html

XI Assessment Scheme (25 Marks)

S. No.	Weightage- Process related: 60%	Marks-15
1.	Logic formation:30%	
2.	Debugging ability:20%	
3.	Follow ethical practices:10%	
	Weightage- Product related: 40%	**Marks-10**
4.	Expected output:15%	
5.	Timely Submission:15%	
6.	Answer to sample questions:10%	
	Total 25	
	Dated Signature of Course Teacher	