Dr. D. Y. Patil Educational Federation's

Dr. D. Y. PATIL COLLEGE OF ENGINEERING AND INNOVATION

Survey No. 27/A/1/2C, Village Varale, Near Talegaon Railway Station,
Tal. Maval, Dist. Pune 410507, Ph.No. 020 48522561
Web Site: www.dypcoei.edu.in, Email: principal.dypcoei@dypatilef.com

DEPARTMENT
OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Computer Laboratory-IV Manual

Subject Code: 417535

Prepared By

Mrs. Asmeeta Mali

Dr. D. Y. Patil Educational Federation's

Dr. D. Y. PATIL COLLEGE OF ENGINEERING AND INNOVATION

Survey No. 27/A/1/2C, Village Varale, Near Talegaon Railway Station,
Tal. Maval, Dist. Pune 410507, Ph.No. 020 48522561
Web Site: www.dypcoei.edu.in, Email: principal.dypcoei@dypatilef.com

DEPARTMENT
OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Lab Manual

Fourth Year Engineering, Semester-VIII

Course & Course Code: Computer Laboratory IV (417535)

Class: BE AI&DS

Academic Year 2024-25

Computer Laboratory IV
Subject Code: 417535

Teaching Scheme	Credit	Examination Scheme
PR:: 02 Hours/Week	02	PR: 25 Marks TW: 50 Marks

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant. The instructor's manual is to be developed as a hands-on resource and reference. The instructor's manual needs to include a prologue (about the University/program/ institute/ department/foreword/ preface etc.), University syllabus, conduction & Assessment guidelines, topics under consideration concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by students in the form of a journal. Journal consists of prologue, Certificate, table of contents, and handwritten write-up of each assignment (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks, and assessor's sign, Theory- Concept in brief, Database design, test cases, conclusion/analysis.

1. Students should submit term work in the form of the journal with write-ups based on a specified list of assignments.
2. Practical /Oral Examinations will be based on all the assignments in the lab manual.
3. Candidate is expected to know the theory involved in the experiment.
4. The practical/Oral examination should be conducted only if the journal of the candidate is complete in all respects.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work is done based on the overall performance and lab assignments performance of students. Each lab assignment assessment will assign grade/marks based on parameters (Attendance, conduction & viva). Suggested parameters for the overall evaluation as well as each lab assignment assessment include- timely completion, performance, innovation, efficient codes, punctuality, and neatness.

1. Examiners will assess the student based on the performance of students considering the parameters such as timely conduction of practical assignment, the methodology adopted for the implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of the implemented assignment, attendance, etc.
2. Examiners will judge the understanding of the practices performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. The concerned faculty member should check appropriate knowledge of the usage of software

and hardware related to the respective laboratory.

Guidelines for Practical Examination

Both internal and external examiners should jointly set problem statements. During the practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to test the students for advanced learning, understanding of the fundamentals, and effective and efficient implementation. So, encouraging efforts, transparent evaluation, and a fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of the student's academics.

Guidelines for Laboratory Conduction

Set of Suggested assignment lists are provided in Groups – A and B. Each Student must perform at least 10 assignments (8 from Group A, 2 from Group B i.e. 1 Mini Project from each elective).
Operating System Recommended: - 64-bit Open-source Linux or its derivative
Programming tools recommended: SQL, PL/SQL, Front End: Java/Perl/PHP/Python/Ruby/.net, Backend: Monod/MYSQL/Oracle, Database Connectivity: ODBC/JDBC.

ASSIGNMENT 1

PROBLEM STATEMENT: -

Set up and Configuration of Hadoop Using CloudEra/ Google Cloud BigQuery. Databricks Lakehouse Platform. Snowflake, Amazon Redshift

OBJECTIVE:

1. Set up a multi-node Hadoop cluster using Cloudera Manager And Configure Hadoop Distributed File System (HDFS) for distributed storage.
2. Create a Google Cloud project and enable BigQuery API and Configure billing and access controls.
3. Create a Databricks workspace and configure clusters.
4. Create a Snowflake account and configure access controls.
5. Launch and configure an Amazon Redshift cluster.

Software Requirements:

Cloudera Manager Server, Google Cloud Platform account and SDK for command-line tools , Databricks account and Apache Spark environment , Snowflake account and UI access , Amazon Redshift cluster , AWS account with permissions and JDBC/ODBC driver.

Hardware Requirements:

• Network connectivity, Storage for Hadoop Distributed File System (HDFS).

• Amazon Redshift, Snowflake is a cloud-based service, Databricks are provides a managed environment, no specific hardware requirements.

THEORY:

Setting up and configuring Hadoop using Cloudera, Google Cloud BigQuery, Databricks Lakehouse Platform, Snowflake, and Amazon Redshift involves different procedures for each platform. Below, I'll provide a brief overview and the general steps for setting up each one. Please note that detailed setup and configuration steps can vary based on the specific version and updates of these platforms.

• **Hadoop using Cloudera:**

1. **Cloudera Distribution**: Download and install Cloudera Distribution for Hadoop (CDH) from the Cloudera website.

2. **Cloudera Manager**: Use Cloudera Manager to set up and manage your Hadoop cluster. Follow the installation and configuration wizard in Cloudera Manager to define hosts, services, and configuration

parameters.

3. **HDFS and MapReduce**: Configure Hadoop Distributed File System (HDFS) and MapReduce based on your requirements.

4. **Cluster Deployment**: Deploy the configured cluster.

• Google Cloud Big Query:

1. **Google Cloud Console**: Log in to the [Google Cloud Console](https://console.cloud.google.com/).

2. **Enable Big Query API**: In the GCP Console, navigate to "APIs & Services" > "Library." Enable the Big Query API.

3. **Create a Dataset**: Create a BigQuery dataset to organize your tables.

4. **Create Tables**: Create tables within your dataset, defining the schema and specifying data sources.

5. **Access Control**: Configure access control for datasets and tables.

• Databricks Lakehouse Platform:

1. Databricks Account:

Sign up for a Databricks account: [Databricks](https://databricks.com/trydatabricks).

2. Create a Workspace: Create a Databricks workspace. Configure clusters, libraries, and storage settings.

3. Connect to Data Sources: Connect Databricks to your data sources, such as storage accounts or databases

• Snowflake:

1. Snowflake Account:

Sign up for a Snowflake account: [Snowflake](https://www.snowflake.com/free-trial/).

2. Snowflake UI: Use the Snowflake web UI to configure your account, create warehouses, databases, and schemas.

3. Connectivity: Configure network policies and connectivity settings.

4. User and Role Setup: Set up users and roles with appropriate privileges.

• Amazon Redshift:

1. AWS Account: Sign up for an AWS account: [AWS Signup](https://aws.amazon.com/).

2. Amazon Redshift Cluster: In the AWS Management Console, navigate to Amazon Redshift.

Create a new Redshift cluster.

3. Configure Redshift: Set up databases, tables, and users as needed. Configure security groups and network settings.

These are general steps, and each platform has detailed documentation that you should follow for the specific versions you are using. Ensure that you follow best practices for security, scalability, and performance in each case.

Applications –

• Hadoop Using Cloudera :

1. Data Warehousing.
2. Batch Processing.
3. Machine Learning.

• Google Cloud BigQuery :

1. Ad Hoc Analytics
2. Real-time Analytics

• Databricks Lakehouse Platform:

1. Unified Analytics.
2. ETL and Data Integration.
3. Data Exploration and Visualization.

• Snowflake:

1. Cloud Data Warehousing
2. Data Sharing
3. Time Travel Queries.

• Amazon Redshift:

1. Business Intelligence (BI).
2. Data Integration.

Setup - Setting up and configuring Hadoop, Google Cloud BigQuery, Databricks Lakehouse Platform, Snowflake, and Amazon Redshift involves different procedures for each platform. Below, I'll provide you with a brief overview and the steps for setting up each one. Keep in mind that detailed setup and configuration steps can vary based on the specific version and updates of these platforms.

• **Hadoop Setup:**

Download and Install Hadoop:

Visit the Apache Hadoop website and download the version you want to use: [Hadoop Downloads](http://hadoop.apache.org/releases.html) Follow the installation instructions provided with the Hadoop distribution.

Configure Hadoop:

Modify the `core-site.xml`, `hdfs-site.xml`, and other configuration files as needed. Configure the Hadoop environment variables.

Start Hadoop Services:
Use the `start-dfs.sh` and `start-yarn.sh` scripts to start the Hadoop Distributed File System (HDFS) and Yet Another Resource Negotiator (YARN).

• **Google Cloud BigQuery Setup:**

1. **Create a Google Cloud Platform (GCP) Project:**

   Go to the [Google Cloud Console](https://console.cloud.google.com/).
   Create a new project.

2. **Enable BigQuery API:**

   In the GCP Console, navigate to "APIs & Services" > "Library". Enable the BigQuery API.

3. **Set Up Authentication:**

   Create service account credentials with BigQuery access. Download the JSON key file and set the `GOOGLE_APPLICATION_CREDENTIALS` environment variable.

• **Databricks Lakehouse Platform Setup:**

1. Create a Databricks Account:
Sign up for a Databricks account: [Databricks](https://databricks.com/try-databricks)

2. Configure Databricks Workspace:
Create a Databricks workspace. Configure clusters, libraries, and other settings as needed.

3. Connect to Data Sources: Connect Databricks to your data sources, such as storage accounts or databases.

• **Snowflake Setup:**
1. Create a Snowflake Account: Sign up for a Snowflake account: [Snowflake](https://www.snowflake.com/free-trial/)

2. Configure Snowflake: Create a Snowflake warehouse, database, and schema Set up users and roles.

3. **Connect to Snowflake**: Use a Snowflake client or integration tool to connect to Snowflake.

• Amazon Redshift Setup:

1. **Create an AWS Account**: Sign up for an AWS account: [AWS Signup] (https://aws.amazon.com/)

2. **Create an Amazon Redshift Cluster**: In the AWS Management Console, navigate to Amazon Redshift. Create a new Redshift cluster.

3. **Configure Redshift**: Set up databases, tables, and users as needed. Configure security groups and network settings.

Each platform has its own documentation with detailed setup and configuration instructions. Follow the official documentation for the specific versions you are using. Additionally, consider security best practices and compliance requirements when setting up these platforms.

CONCLUSION:

In this way we have Set up and Configuration Hadoop Using CloudEra/ Google Cloud BigQuery, Databricks Lakehouse Platform, Snowflake, Amazon Redshift.

ORAL QUESTION:

1. How did you configure and set up Hadoop within the CloudEra/Google Cloud BigQuery environment?
2. What are the key advantages of using CloudEra or Google Cloud BigQuery for hosting Hadoop?
3. How did you ensure data integrity and security within this environment?
4. What are the key features of Databricks Lakehouse Platform that enhance data processing and analytics capabilities?

ASSIGNMENT 2

PROBLEM STATEMENT: -
Develop a MapReduce program to calculate the frequency of a given word in a given file.

OBJECTIVE:
1. Develop a MapReduce program to calculate the frequency of a given word.
2. Understand the MapReduce programming model and its key components (Mapper and Reducer).
3. Implement the logic for word frequency calculation in the Mapper and Reducer functions.

Hardware Requirements –

• A Hadoop cluster for distributed computing.

• Sufficient computational resources on the cluster to handle the input file.

Software Requirements –

• Hadoop Installation: Ensure that Hadoop is installed and configured on the cluster.

• Java Development Kit (JDK): MapReduce programs are typically written in Java, so JDK is required for development.

• Text Editor or Integrated Development Environment (IDE): Use a text editor or IDE to write and edit the MapReduce program.

• Input File: The file for which the word frequency is to be calculated.

THEORY:

• **Hadoop** is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. It was popularized by Google and is widely used in open-source implementations like Apache Hadoop.

• **A MapReduce** job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

• In the **Map phase**, the input data is divided into smaller chunks and processed independently in parallel across multiple nodes in a distributed computing environment. Each chunk is transformed or

"mapped" into key-value pairs by applying a user-defined function. The output of the Map phase is a set of intermediate key-value pairs.

• The **Reduce phase** follows the Map phase. It gathers the intermediate key-value pairs generated by the Map tasks, performs data shuffling to group together pairs with the same key, and then applies a user-defined reduction function to aggregate and process the data. The output of the Reduce phase is the final result of the computation.

• The MapReduce framework operates exclusively on pairs, that is, the framework views the input to the job as a set of pairs and produces a set of pairs as the output of the job, conceivably of different types.

• Input and Output types of a MapReduce job: (input) -> map -> -> combine -> -> reduce -> (output)

• **MapReduce** allows for efficient processing of large-scale datasets by leveraging parallelism and distributing the workload across a cluster of machines. It simplifies the development of distributed data processing applications by abstracting away the complexities of parallelization, data distribution, and fault tolerance, making it an essential tool for big data processing in the Hadoop ecosystem.

• **Word Count** is a simple application that counts the number of occurrences of each word in a given input set. This works with a local-standalone, pseudo-distributed or fully-distributed Hadoop installation.

• The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is '1'.

• For instance if you consider the sentence "An elephant is an animal". The mapper phase in the Word Count example will split the string into individual tokens i.e. words. In this case, the entire sentence will be split into 5 tokens (one for each word) with a value 1 as shown below –

• Key-Value pairs from Hadoop Map Phase Execution-

(an,1)

(elephant,1)

(is,1)

(an,1)

(animal,1)

• Hadoop WordCount Example- Shuffle Phase Execution: After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order. After the shuffle phase is executed from the WordCount example code, the output will look like this –

(an,1)

(an,1)

(animal,1)

(elephant,1)

(is,1)

•Hadoop WordCount Example- Reducer Phase Execution: In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word. It is like an aggregation phase for the keys generated by the map phase. The reducer phase takes the output of shuffle phase as input and then reduces the key-value pairs to unique keys with values added up. In our example "An elephant is an animal." is the only word that appears twice in the sentence. After the execution of the reduce phase of MapReduce WordCount example program, appears as a key only once but with a count of 2 as shown below –

(an, 2)

(animal, 1)

(elephant, 1)

(is, 1)

• This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file. An important point to note during the execution of the WordCount example is that the mapper class in the WordCount program will execute completely on the entire input file and not just a single sentence. Suppose if the input file has 15 lines then the mapper class will split the words of all the 15 lines and form initial key value pairs for the entire dataset. The reducer execution will begin only after the mapper phase is executed successfully.

Code –

Mapper Code –

```
// Importing libraries import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

// Map function

public void map(LongWritable key, Text value, OutputCollector<Text

IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces

for (String word : line.split(" "))

{

if (word.length() > 0)

{

output.collect(new Text(word), new IntWritable(1));

} } } }
```

Reducer Code:

```
// Importing libraries import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import  org.apache.hadoop.mapred.MapReduceBase;

import   org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

// Reduce function

public void reduce(Text key, Iterator value,

OutputCollector output, Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words

while (value.hasNext())

{

IntWritable i = value.next();

count += i.get();

 }

output.collect(key, new IntWritable(count));

}}
```

Driver Code:

```
// Importing libraries

import java.io.IOException;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool

{

public int run(String args[]) throws IOException

{

if (args.length < 2)

{

System.out.println("Please give valid inputs");

return -1;

}

JobConf conf = new JobConf(WCDriver.class);

FileInputFormat.setInputPaths(conf,newPath(args[0]));
```

```
FileOutputFormat.setOutputPath(conf,newPath(args[1])); conf.setMapperClass(WCMapper.class);

conf.setReducerClass(WCReducer.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(IntWritable.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

return 0;

}


// Main Method

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

{

int exitCode = ToolRunner.run(new WCDriver(), args);

System.out.println(exitCode);

}

}
```

CONCLUSION:

In this way we have develop a MapReduce program to calculate the frequency of a given word in a given file.

ORAL QUESTION:

1. What are the key phases in a MapReduce program?
2. How do you handle intermediate key-value pairs between the Mapper and Reducer?
3. What challenges might arise when dealing with very large files in MapReduce?
4. Can you explain the concept of shuffling and sorting in MapReduce and its relevance here?

ASSIGNMENT 3

PROBLEM STATEMENT:

Implement Matrix Multiplication using Map-Reduce.

OBJECTIVE:

1. Implement Matrix Multiplication using the Map-Reduce programming model.
2. Understand the parallel processing capabilities of Map-Reduce for large-scale data.

Software Requirements

Hadoop (or any other Map-Reduce framework)

Hardware Requirements

A cluster of machines for distributed processing (number of nodes depends on the scale of data)

Data set

Input matrices for multiplication. This could be generated or sourced from real-world scenarios.

Libraries/modules used

Hadoop Map-Reduce libraries.

Theory

Matrix multiplication is a fundamental operation in linear algebra and is often used in various computational tasks. In the context of big data analysis, implementing matrix multiplication using MapReduce is an interesting and challenging task. MapReduce is a programming model commonly used for processing and generating large datasets in a distributed and parallel manner. Here's an overview of how you can implement matrix multiplication using MapReduce.

Matrix Multiplication using MapReduce

Input Data:

Assume you have two matrices, A and B, that you want to multiply.

Matrix A:

```
A = | a11 a12 |
    | a21 a22 |
```

Matrix B:

```

```

B = | b11 b12 |
| b21 b22 |

```

**MapReduce Steps**

1. Map Step: -

- Each Mapper takes a portion of Matrix A and Matrix B as input.

- The key is the row index for Matrix A and column index for Matrix B.

- The value is the corresponding element.

Mapper 1:

```
Input: (i, A, aij)

Output: [(j, ('A', i, aij))] # j is the column index of matrix B
```

Mapper 2:

```
Input: (j, B, bjk)

Output: [(i, ('B', j, bjk))] # i is the row index of matrix A
```

2. Shuffle and Sort:

- The framework groups the intermediate key-value pairs by key.

3. Reduce Step: - Each Reducer receives a group of key-value pairs with the same key.

- It multiplies the corresponding elements and sums up the products.

Reducer:

```
Input: (j, [('A', i, aij), ('B', j, bjk), ...])

Output: [(('A', i, aij), ('B', j, bjk)), ...]
```

4. Final Output:

- The final output is a set of key-value pairs representing the result matrix.

```
Output: (i, j, Cij) # Cij is the element at row i and column j of the result matrix C
```

Example: Let's say we want to multiply two 2x2 matrices: Matrix A:
```

```
| 1 2 |
| 3 4 |
```

Matrix B:

```
 | 5 6 | | 7 8 |
```

The MapReduce process would involve breaking down the matrices into key-value pairs, shuffling and sorting, and then performing the multiplication and summing in the Reducer. The final output would be the result matrix C:

Result Matrix C:

```
| 19 22 | | 43 50 |
```

This process showcases how MapReduce can be leveraged for distributed and parallel computation of matrix multiplication, making it suitable for handling large datasets in big data scenarios.

Algorithm of work –

• Divide the matrix multiplication into Map and Reduce phases.
• Map phase: Emit intermediate key-value pairs for each element in the input matrices.
• Shuffle and Sort phase: Group intermediate pairs by key (matrix element position).
• Reduce phase: Perform the actual multiplication and summing to get the result.

Applications

• Use cases include large-scale scientific computations, machine learning algorithms, and data processing tasks where matrices are involved.

Code

Mapper:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MatrixMapper extends Mapper {
private Text outKey = new Text();
private Text outValue = new Text();
protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
int i = Integer.parseInt(tokenizer.nextToken());
int j = Integer.parseInt(tokenizer.nextToken());
int valueIJ = Integer.parseInt(tokenizer.nextToken());
// Emit key-value pairs for multiplication in the reducer
for (int k = 0; k < context.getNumReduceTasks(); k++) {
outKey.set(j + "," + k); // Group elements by column index (j)
outValue.set(i + "," + valueIJ);
context.write(outKey, outValue);
} }
```

Reducer:

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MatrixReducer extends Reducer {
private IntWritable result = new IntWritable();
protected void reduce(Text key, Iterable values, Context context) throws IOException, InterruptedException {
List iValues = new ArrayList<>();
List AValues = new ArrayList<>();
for (Text value : values)
```

```
{
String[] parts = value.toString().split(",");
int i = Integer.parseInt(parts[0]);
int Aij = Integer.parseInt(parts[1]);
iValues.add(i); AValues.add(Aij);
}
// Perform multiplication and aggregation
for (int i = 0; i < iValues.size(); i++)
{
int product = AValues.get(i); // Assuming corresponding Bjk values are available in the reducer context
result.set(product);
context.write(new Text(iValues.get(i) + "," + key.toString().split(",")[0]), result); // Emit final (i,j) result
} } }
```

Driver:

```
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MatrixMultiplyDriver
{
public static void main(String[] args) throws Exception
{
if (args.length != 3)
{
System.err.println("Usage: MatrixMultiplyDriver <input path matrix A> <input path matrix B>
<output path>");
System.exit(-1);
}
Configuration conf = new Configuration();
```

```
Job job = Job.getInstance(conf, "Matrix Multiplication");
job.setJarByClass(MatrixMultiplyDriver.class);
job.setMapperClass(MatrixMapper.class);
job.setReducerClass(MatrixReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0])); // Input path for matrix A // Assuming matrix B is accessible within the reducer context
FileOutputFormat.setOutputPath(job, new Path(args[2])); // Output path for result matrix
System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

Steps to run the code

Here are the steps to run the matrix multiplication MapReduce program using the provided code:

1. Prerequisites:

• Hadoop environment: Ensure you have a Hadoop environment set up and configured on your system. This includes setting up HDFS, Yarn, and necessary environment variables.

• Java compiler: You'll need a Java compiler like javac to compile the provided code.

• Input files: Prepare two text files containing your matrices (matrix A and matrix B) in the format described earlier (rows separated by newlines, elements within rows separated by spaces).

2. Compile the code:

Open a terminal in your Hadoop environment directory and compile the Java files for the mapper, reducer, and driver: javac MatrixMapper.java MatrixReducer.java MatrixMultiplyDriver.java

3. Submit the MapReduce job:

Run the driver class with the path to your input files (matrix A and matrix B) and the desired output directory:

hadoop jar <your_jar _file name>.jar MatrixMultiplyDriver <input_path_matrix A> <input_path_matrix B> <output_path>

Replace <your_jar _file name>.jar with the actual name of the JAR file created after compiling the code.

Replace <input_path_matrix A>, <input_path_matrix B> and <output_path>with the actual paths to your input and output directories.

4. Monitoring and Output:

The job will be submitted to the Hadoop cluster and start processing. You can monitor the progress using the Hadoop web interface or command-line tools. Once the job completes successfully, check the output directory for the resulting matrix with elements (i, j) calculated by the program.

Running Example:

Assume you have two text files matrix_a.txt and matrix_b.txt containing your matrices and an output directory named result. you compile the code and run the driver with the following command: hadoop jar matrix_multiply.jar MatrixMultiplyDriver matrix_a.txt matrix_b.txt result.

If the job finishes successfully, you'll find the resulting product matrix with elements (i, j) in the result directory.

Note: This is a basic example. You might need to adapt the code and steps based on your specific matrices, Hadoop configuration, and desired output format. Remember to modify the reducer logic if matrix B isn't directly accessible within the reducer context.

CONCLUSION:

In this way we have Implemented Matrix Multiplication using Map-Reduce.

ORAL QUESTION:

1. Can you explain how MapReduce can be used to parallelize Matrix Multiplication?
2. What are the key components of a MapReduce implementation for Matrix Multiplication?
3. What challenges might arise when dealing with very large matrices in MapReduce?
4. What optimizations could you implement to improve the performance of Matrix Multiplication in MapReduce?

ASSIGNMENT 4

PROBLEM STATEMENT:

Develop a MapReduce program to find the grades of students.

OBJECTIVE:

1. Implement a MapReduce program to calculate grades for students based on their scores.
2. Understand the distributed processing nature of MapReduce in the context of grading.

Hardware Requirements

• A cluster of machines for distributed processing (number of nodes depends on the scale of data).

Software Requirements:

• Hadoop (or any other MapReduce framework).

Data set

• Student data with scores (e.g., CSV file with student ID, name, and scores in different subjects).

Libraries or modules used

• Hadoop MapReduce libraries.

Theory

• **MapReduce Paradigm**

MapReduce is a programming model and processing technique designed for large-scale data processing across distributed clusters. It divides a computational task into two main phases: the Map phase and the Reduce phase. In the context of grading students, MapReduce can efficiently handle the computation of average scores and assignment of grades across a large dataset.

• **Mapper Function**

The mapper function is responsible for processing each input record and emitting intermediate keyvalue pairs. In the case of student grading, the input data likely consists of records in the form of "student_name, subject, score." The mapper function extracts the relevant information, specifically the student name, subject, and score, and emits key-value pairs where the student name is the key and a tuple containing the subject and score is the value.

• **Shuffling and Sorting**

The framework automatically shuffles and sorts the intermediate key-value pairs, grouping them by key. In our student grading example, this means that all records related to a specific student are brought together, ready for processing in the next phase.

• **Reducer Function**

The reducer function takes the grouped key-value pairs and performs the necessary computations to determine the final output. In the context of grading students, the reducer calculates the average score for each student by iterating through the tuples of subjects and scores associated with that student. Once the average score is computed, the reducer assigns a grade based on predefined criteria (e.g., A for scores above 90, B for scores between 80 and 90, and so on). The final output of the reducer is a key-value pair where the key is the student name, and the value is the assigned grade.

• **Scalability and Parallel Processing**

One of the key strengths of MapReduce is its ability to scale horizontally, meaning it can efficiently process large datasets by distributing the computation across multiple nodes in a cluster. Each mapper and reducer operates independently on its subset of data, allowing for parallel processing and significantly reducing the time required for computation.

• **Flexibility and Extensibility**

The MapReduce model is flexible and extensible, making it suitable for a wide range of data processing tasks. In the student grading example, the same MapReduce framework can be adapted for different datasets and grading criteria by modifying the mapper and reducer functions.

Algorithm of work

• **Map phase**

Read student data, emit student ID as the key, and scores as values.

• **Reduce phase**

Calculate grades based on scores and emit the final result.

Applications

• Use cases include automated grading systems for educational institutions.

Code

1. Mapper:

```
Public class Mapper extends Mapper
{
Public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
String line = value.toString();
String[] fields = line.split(","); // assuming comma-separated values
```

```
String studentId = fields[0];
int marks = Integer.parseInt(fields[1]); // assuming second field is marks
Text outputKey = new Text(studentId);
IntWritable outputValue = new IntWritable(marks);
context.write(outputKey, outputValue);
} }
```

This mapper reads each line of the input data containing student ID and marks separated by a comma. It then:

- Splits the line into fields based on the comma separator.
- Extracts the student ID and parses the marks as an integer.
- Creates a key-value pair where the key is the student ID and the value is the marks.
- Emits the key-value pair to the reducer.

2. Reducer:

```
Java
public class Reducer extends Reducer
{
public void reducer (Text key, Iterable values, Context context) throws IOException, InterruptedException {
int totalMarks = 0;
for (IntWritable mark : values)
{
totalMarks += mark.get();
}
String grade = calculateGrade(totalMarks); // Replace with your logic for calculating grades
context.write(key, new Text(grade));
}
private String calculateGrade(int totalMarks)
{ // Replace with your specific grade calculation logic
if (totalMarks >= 90) {
return
"A";
} else
```

```
if (totalMarks >= 80) {
return
"B";
} else
if (totalMarks >= 70) {
return
"C"; } else
if (totalMarks >= 60) {
return
"D"; } else {
return "F";
} } }
```

This reducer receives key-value pairs from the mapper, where the key is the student ID and the values are individual marks. It:

• Iterates through the marks for each student and calculates the total marks.

• Uses your logic (replace calculateGrade(int totalMarks)) to determine the grade based on the total marks.

• Creates a new key-value pair where the key is the student ID and the value is the calculated grade.

3. Driver:

```
public class Driver
{
Public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Student Grades");
job.setJarByClass(Driver.class);
job.setMapperClass(Mapper.class);
job.setReducerClass(Reducer.class);
job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job,newPath(args[1])); System.exit(job.waitForCompletion(true) ? 0 :
1);
}
```

}

This driver configures and submits the MapReduce job:

• Sets the job name and JAR file containing the mapper and reducer classes.

• Specifies the mapper and reducer classes to be used.

• Defines the input format (text file) and output format (text file).

• Sets the input and output paths for the job based on command-line arguments.

• Submits the job and waits for completion, exiting with an appropriate code.

4. **Running the program**:

• Save the code in separate files for Mapper, Reducer, and Driver.

• Compile the files using javac command.

• Run the program using the hadoop jar command, specifying the JAR file, input path, and output path as arguments.

• The output file will contain student IDs and their respective grades.

5. Adapting the program:

• Replace the comma-separated assumption with your actual data format.

• Modify the calculateGrade method to reflect your specific grading system.

• Adjust the input and output formats based on your needs.

CONCLUSION:

In this way we have develop a MapReduce program to find the grades of students.

ORAL QUESTION:

1. How do you handle scenarios where the grading criteria need to be adjusted or updated?
2. How do you ensure the correctness and efficiency of the MapReduce program?
3. What would be the output format of this MapReduce job?

ASSIGNMENT 5

PROBLEM STATEMENT:

To develop a MapReduce program to analyse Titanic ship data, focusing on calculating the average age of deceased males and determining the count of female casualties in each class.

OBJECTIVE:

• Develop a MapReduce program for analysing Titanic ship data.

• Compute the average age of males who died in the tragedy.

• Determine the number of female casualties in each class.

• Understand and implement the key components of MapReduce: Mapper and Reducer functions.

Hardware Requirements

• A Hadoop cluster for distributed computing.

• Sufficient computational resources on the cluster to handle the input file.

Software Requirements

• Hadoop Installation: Ensure that Hadoop is installed and configured on the cluster.

• Java Development Kit (JDK): MapReduce programs are typically written in Java, so JDK is required for development.

• Text Editor or Integrated Development Environment (IDE): Use a text editor or IDE to write and edit the MapReduce program.

• Titanic Ship Data: The dataset containing information about passengers on the Titanic.

Theory

• Hadoop, a pioneering open-source framework, serves as the bedrock for processing vast datasets in parallel across expansive clusters of commodity hardware. It derives inspiration from Google's infrastructure and is widely utilized, especially in the realm of big data applications.

• In the context of MapReduce, a pivotal paradigm within Hadoop, the workflow involves disassembling colossal datasets into manageable fragments. This segmentation facilitates independent processing through map tasks, fostering efficient parallel computation. Subsequently, a reduction phase orchestrates the synthesis of these processed chunks, employing techniques such as speculative execution for continuous progress and checkpointing for restarting from intermediate stages in case of

failures. In the Map phase, the input data is divided into smaller chunks and processed independently in parallel across multiple nodes in a distributed computing environment. Each chunk is transformed or "mapped" into key-value pairs by applying a user-defined function. The output of the Map phase is a set of intermediate key-value pairs.

• The Reduce phase follows the Map phase. It gathers the intermediate key-value pairs generated by the Map tasks, performs data shuffling to group together pairs with the same key, and then applies a user-defined reduction function to aggregate and process the data. The output of the Reduce phase is the final result of the computation.

• The Map phase, a crucial precursor, acts as the engine for data transformation. Input data undergoes a metamorphosis into key-value pairs, steered by user-defined mapping functions (Fmap). This phase yields intermediate key-value pairs, laying the groundwork for the impending reduction.

• Contrastingly, the Reduction phase assumes the role of data aggregation and synthesis. It assimilates intermediate key-value pairs generated by map tasks, employing techniques such as in-memory merging (Mr) for efficient data aggregation. The orchestrated process involves key grouping through hashing (H) or sorting algorithms, showcasing the intricacies of distributed data processing tasks under the orchestration of Hadoop's job scheduler (JS).

• Input and Output types of a MapReduce job: (input) -> map -> -> combine -> -> reduce -> (output)

• MapReduce allows for efficient processing of large-scale datasets by leveraging parallelism and distributing the workload across a cluster of machines. It simplifies the development of distributed data processing applications by abstracting away the complexities of parallelization, data distribution, and fault tolerance, making it an essential tool for big data processing in the Hadoop ecosystem.

• The Titanic dataset encapsulates a rich array of information about passengers on the ill-fated voyage. Key attributes include passenger class, gender, age, and survival status. This multifaceted dataset serves as a poignant historical record, offering insights into the societal impact of the tragedy. Our MapReduce exploration aims to glean nuanced details from this dataset, shedding light on the human dimension of this historical event.

Code

```
// import libraries import java.io.IOException;
 import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

// Making a class with name Average_age
public class Average_age {
public static class Map extends Mapper<long writable, Text, Text, Int Writable> {
// private text gender variable which
// stores the gender of the person
// who died in the Titanic Disaster private Text gender = new Text();
// private IntWritable variable age will store
// the age of the person for MapReduce. where
// key is gender and value is age
private IntWritable age = new IntWritable();
// overriding map method(run for one time for each record in dataset)
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
{
// storing the complete record
// in a variable name line
String line = value.toString();
// splitting the line with ', ' as the
// values are separated with this
// delimiter
String str[] = line.split(", ");

/* checking for the condition where the number of columns in our dataset has to be more than 6. This helps in eliminating the ArrayIndexOutOfBoundsException when the data sometimes is incorrect in our dataset*/

if (str.length > 6)
{
// storing the gender
```

```
// which is in 5th column
gender.set(str[4]);
// checking the 2nd column value in
// our dataset, if the person is
// died then proceed. if ((str[1].equals("0"))) {
// checking for numeric data with
// the regular expression in this column if (str[5].matches("\\d+")) {
// converting the numeric
// data to INT by typecasting int i = Integer.parseInt(str[5]);
// storing the person of age age.set(i);
} } }
// writing key and value to the context
// which will be output of our map phase
context.write(gender, age);
} }
public static class Reduce extends Reducer {
// overriding reduce method(runs each time for every key )
public void reduce(Text key, Iterable values, Context context) throws IOException, InterruptedException
{
// declaring the variable sum which
// will store the sum of ages of people int sum = 0; // Variable l keeps incrementing for // all the value of that key.
 int l = 0;
// foreach loop
for (IntWritable val : values) {
l += 1;
// storing and calculating
// sum of values
sum += val.get();
 } sum = sum / l;
context.write(key, new IntWritable(sum));
} }
```

```
public static void main(String[] args) throws Exception
{
Configuration conf = new Configuration();
@SuppressWarnings("deprecation")
Job job = new Job(conf, "Averageage_survived");
job.setJarByClass(Average_age.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
//job.setNumReduceTasks(0);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
Path out = new Path(args[1]);
out.getFileSystem(conf).delete(out);
job.waitForCompletion(true);
}
 }
```

CONCLUSION:

In this way we have develop a MapReduce program to analyse Titanic ship data, focusing on calculating the average age of deceased males and determining the count of female casualties in each class.

ORAL QUESTION:

1. How do you handle scenarios where there are inconsistencies or errors in the Titanic ship data?
2. How do you handle scenarios where the Titanic ship data is distributed across multiple files or partitions?
3. What approach would you take to determine the count of female casualties in each class?

ASSIGNMENT 6

PROBLEM STATEMENT: -

Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

OBJECTIVE:

1. To introduce the concepts and components of Business Intelligence (BI).
2. To understand how to import the data from different sources and load in the target system

THEORY:

Power BI

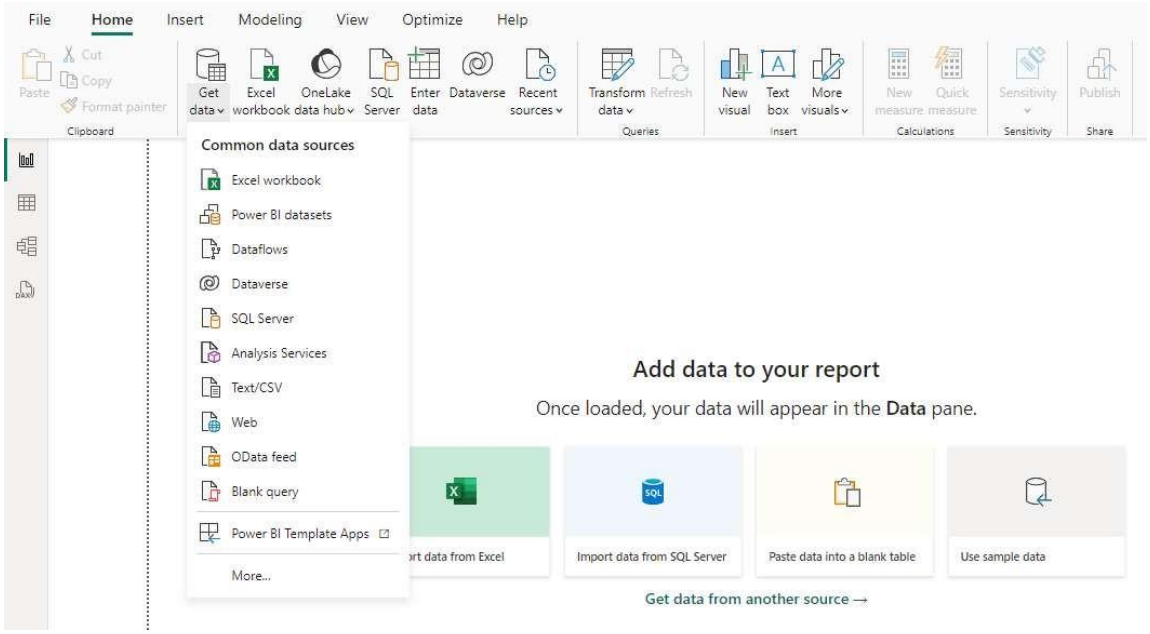
- Microsoft Power BI is a complete reporting solution that offers data preparation, data visualization, distribution, and management through development tools and an online platform.
- Power BI can scale from simple reports using a single data source to reports requiring complex data modeling and consistent themes. Use Power BI to create visually stunning, interactive reports to serve as the analytics and decision engine behind group projects, divisions, or entire organizations.
- Power BI is an essential tool to data analysts and their organization; however, all data professionals benefit from understanding how Power BI works to explore and present data insights within organizations.
- There are three primary components to Power BI:
    - Power BI Desktop (desktop application)
    - Power BI service (online platform)
    - Power BI Mobile (cross-platform mobile app)
- Power BI Desktop is the development tool available to data analysts and other report creators. While the Power BI service allows you to organize, manage, and distribute your reports and other Power BI items. Power BI Desktop is available to download for free either through the Windows store or directly online.
- You can access the Power BI service at app.powerbi.com with a school or work account.
- Power BI Mobile allows consumers to view reports in a mobile-optimized format. You can create these optimized report views in Power BI Desktop.

The flow of Power BI is:

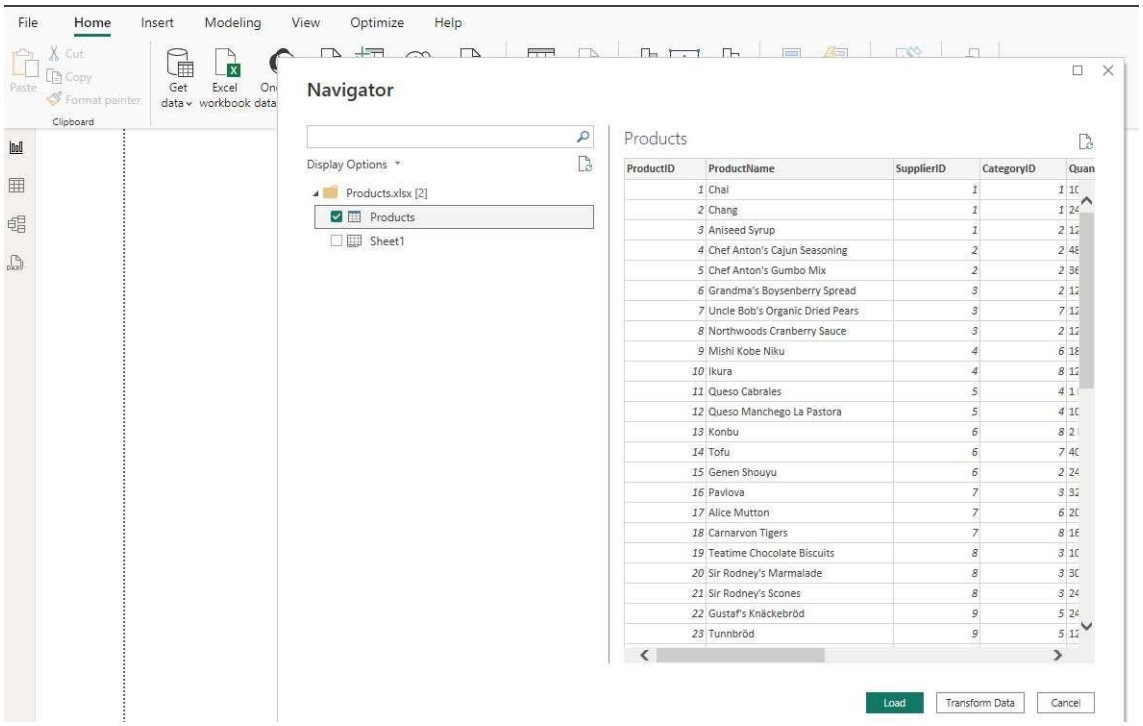
1. Connect to data with Power BI Desktop.
2. Transform and model data with Power BI Desktop.
3. Create visualizations and reports with Power BI Desktop.
4. Publish report to Power BI service.
5. Distribute and manage reports in the Power BI service.

Importing Excel Data

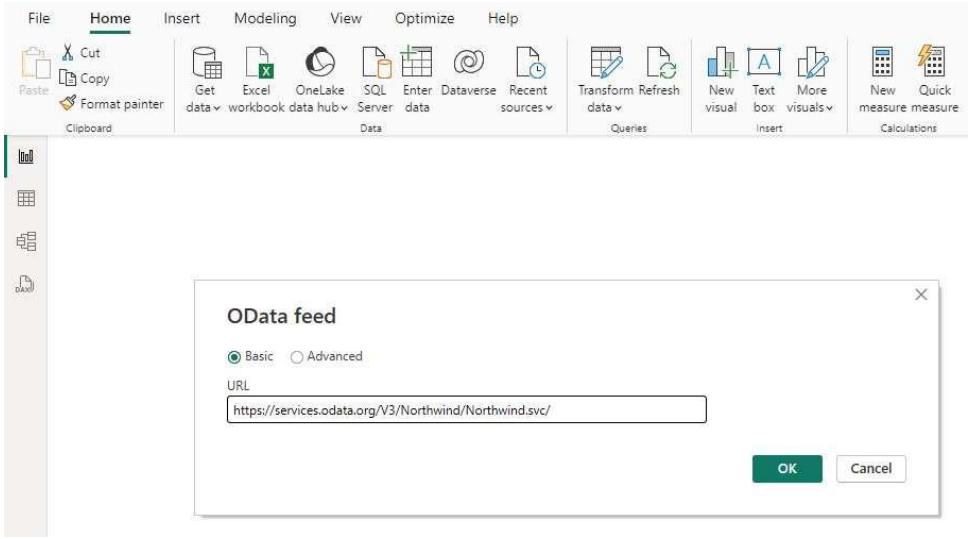
1. Launch Power BI Desktop.
2. From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu.



3. In the Open File dialog box, select the Products.xlsx file (you choose your file).
4. In the Navigator pane, select the Products table and click on the Load button.



Importing Data from OData Feed

1. From the Home ribbon tab in Query Editor, select Get Data.
2. Browse to the OData Feed data source.
3. In the OData Feed dialog box, paste the URL for the Northwind OData feed. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below: http://services.odata.org/V3/Northwind/Northwind.svc/
4. 

5. Select OK.
6. In the Navigator pane, select the Orders table and load the table.

CONCLUSION:

In this way we have explored the Power BI and imported data from different sources to target system.

ORAL QUESTION

1. Explain Business Intelligence.
2. List the different types of data sources supported by Power BI.
3. Explain different components to Power BI.

ASSIGNMENT 7

PROBLEM STATEMENT:

Data Visualization from Extraction Transformation and Loading (ETL) Process.

OBJECTIVE:

1. To understand the concept of Data Visualization.
2. To understand the Data Visualization using Power BI.

THEORY:

Data Visualization

- Data visualization aims to communicate data clearly and effectively through graphical representation.
- Data visualization has been used extensively in many applications for example, at work for reporting, managing business operations, and tracking progress of tasks.
- More popularly, we can take advantage of visualization techniques to discover data relationships that are otherwise not easily observable by looking at the raw data.
- Three major trends have shaped the direction of data visualization software.
    - More Chart Types. Most data visualizations are in the form of some standard chart type. The numerical results are converted into a pie chart, a scatter plot, or another chart type. Now the list of chart types supported by data visualization software has grown much longer.
    - Interactive Visualization. Visualizations are no longer static. Dynamic chart types are themselves user interfaces. Your users can review a result chart, manipulate it, and then see newer views online.
    - Visualization of Complex and Large Result Sets. You users can view a simple series of numeric result points as a rudimentary pie or bar chart. But newer visualization software can visualize thousands of result points and complex data structures.

Drive better decision making with data visualization

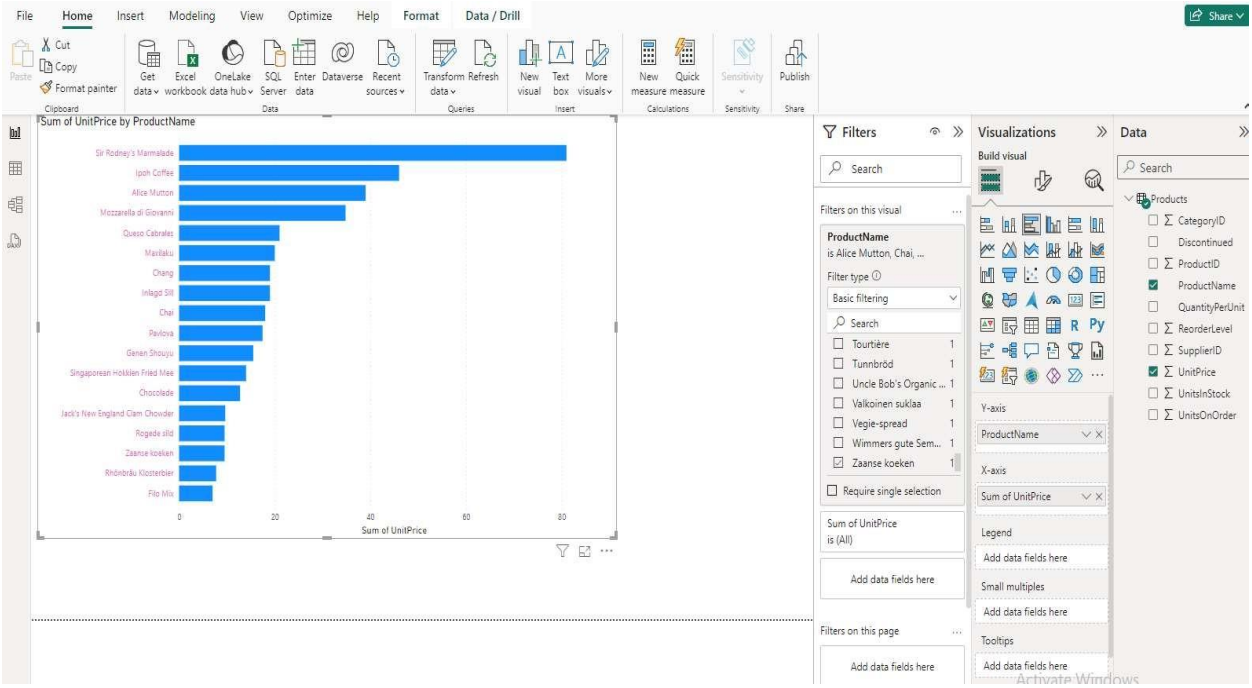
- **See the big picture.** There's a clear picture of performance buried within the transaction, interaction, process, and behavioral data stored in your systems. Data visualization allows you to recognize the broader context and higher-level scenario within it. As a result, you'll notice trends and spot patterns you wouldn't be able to see if you were looking at numbers on their own.

- **Identify the significance.** Bringing visual clarity to the story told within your data helps you identify insights that lead to better decision making, planning, strategies, and actions. How is your business performing, what needs to be modified, and where should you focus your resources? The ability to understand the significance of your data drives more effective operations and decisions.

- **Make informed decisions.** With concrete numbers and tangible insights, you can be confident your decisions are backed by data. Having clear insight into performance metrics empowers you with the knowledge and arms you with the tools to make the right decisions at the right time.

- **Track trends over time.** Once you've established a baseline, trends will begin to emerge. Track progress, spot trends, and begin using your insights to drive informed, strategic decisions. As you build your trends, shifts in patterns indicate if things drift off track, allowing you to immediately address any sign of lowered performance.

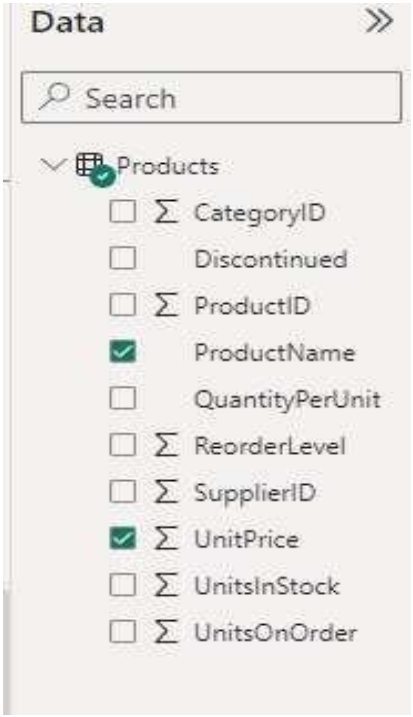
Power BI Desktop lets you create a variety of visualizations to gain insights from your data. You can build reports with multiple pages and each page can have multiple visuals. You can interact with your visualizations to help analyze and understand your data in this task, you create a report based on the data previously loaded. You use the Fields pane to select the columns from which you create the visualizations.



Following are the steps we have to follow to visualize data in Power BI

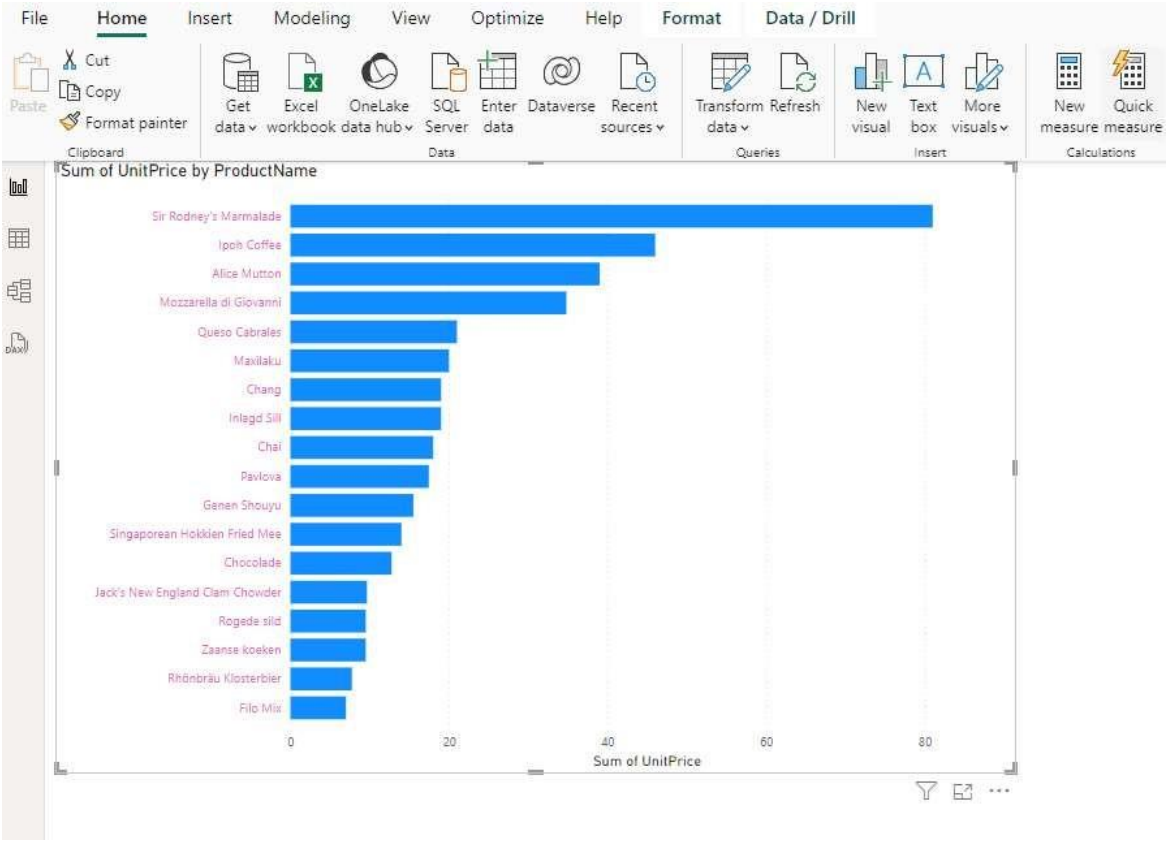
- Load the data in Power BI by applying ETL process.

- Select columns from Data section as shown in above figure which you want to display on chart.



- Select the required chart from Visualization section.

- Following output will get after selecting required visualization type



CONCLUSION

In this way we have explored the concept of Data Visualization using Power BI.

ORAL QUESTION

1. What is the importance of data visualization in the context of the ETL process?
2. How does data visualization enhance the understanding of extracted and transformed data?
3. Can you name some common data visualization tools used in conjunction with ETL processes?
4. Explain the significance of creating dashboards as part of the data visualization process in ETL.

ASSIGNMENT 8

PROBLEM STATEMENT: -

Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

OBJECTIVE:

1. To understand the concept of Extraction Transformation and Loading (ETL) process.
2. Construction of database in Power BI by performing ETL process.

THEORY:

Extraction Transformation and Loading (ETL)

- **Extraction**
  - During the first phase, data are extracted from the available internal and external sources.
  - The selection of data to be imported is based upon the data warehouse design, which in turn depends on the information needed by business intelligence analyses and decision support systems operating in a specific application domain.
  - Types of Data Extraction:
    - Full Extraction: All the data from source systems or operational systems gets extracted to staging area. (Initial Load)
    - Partial Extraction: Sometimes we get notification from the source system to update specific date. It is called as Delta load.
- **Transformation**
  - The goal of the cleaning and transformation phase is to improve the quality of the data extracted from the different sources, through the correction of inconsistencies, inaccuracies and missing values.

- During the transformation phase, additional data conversions occur in order to guarantee homogeneity and integration with respect to the different data sources.

- Examples

  - Standardizing data: Data is fetched from multiple sources so it needs to be standardized as per the target system.
  - Character set conversion: Need to transform the character sets as per the target systems.(First name and last name example)
  - Data Conversion in different formats: If in source system date in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.

- **Loading**

  Finally, after being extracted and transformed, data are loaded into the tables of the data warehouse to make them available to analysts and decision support applications.

There are many reasons for adopting ETL in the organization:

- It helps companies to analyze their business data for taking critical business decisions.
- ETL provides a method of moving the data from various sources into a data warehouse.
- As data sources change, the Data Warehouse will automatically update.
- Well-designed and documented ETL system is almost essential to the success of a Data Warehouse project.
- Allow verification of data transformation, aggregation and calculations rules.
- ETL helps to Migrate data into a Data Warehouse. Convert to the various formats and types to adhere to one consistent system.
- ETL in data warehouse offers deep historical context for the business.

CONCLUSION:

In this way we have performed Extraction Transformation and Loading (ETL) process to construct the database in Power BI.

ORAL QUESTION

1. What is the purpose of the Extraction phase in the ETL process?
2. What factors should you consider when selecting data for extraction?
3. What is meant by the Transformation phase in the ETL process?
4. Explain the concept of loading data in the context of the ETL process.

ASSIGNMENT 9

PROBLEM STATEMENT:

Perform the data classification algorithm using any Classification algorithm.

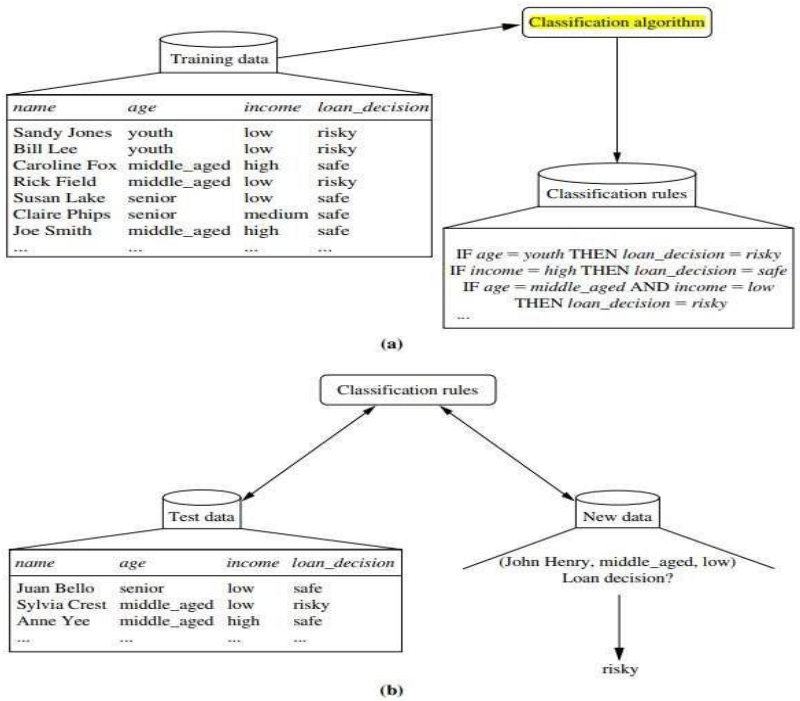
OBJECTIVE:

Students should be able understand the concept of data classification and different data classification algorithm.

THEORY:

Data Classification

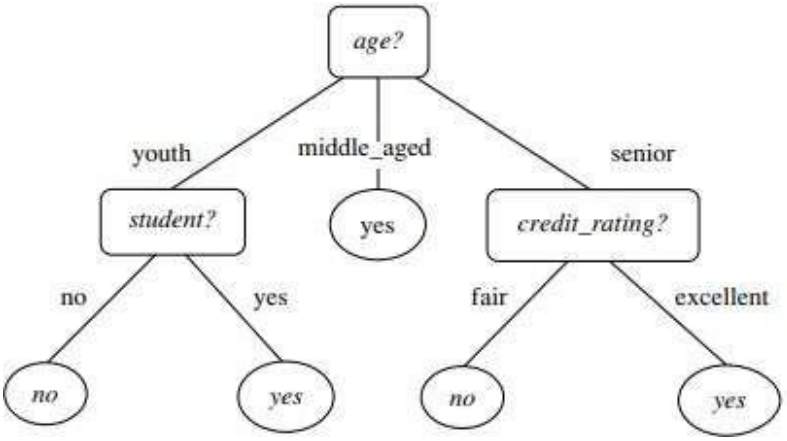
- Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels.
- For example, we can build a classification model to categorize bank loan applications as either safe or risky.
- Data classification is a two-step process, consisting of a learning step (where a classification model is constructed) and a classification step (where the model is used to predict class labels for given data).
- The process is shown for the loan application data in the following figure:



- The data classification process for above example:
    - Learning: Training data are analyzed by a classification algorithm. Here, the class label attribute is loan decision, and the learned model or classifier is represented in the form of classification rules.
    - Classification: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

Decision Tree Induction

- Decision tree induction is the learning of decision trees from class-labeled training tuples.
- A decision tree is a flowchart-like tree structure, where each internal node (non leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node.
- A typical decision tree is shown in figure



- A decision tree algorithm known as ID3 (Iterative Dichotomiser).
- ID3, C4.5(a successor of ID3), and CART(Classification and Regression Trees) adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.

Bayes Classification Methods

- Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' theorem.
- A simple Bayesian classifier known as the na¨ıve Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.

Rule-Based Classification

- Rule-based classifiers, where the learned model is represented as a set of IF-THEN rules.
- An IF-THEN rule is an expression of the form

  IF condition THEN conclusion.

  An example is rule R1

  R1: IF age = youth AND student = yes THEN buys computer = yes.
- The "IF" part (or left side) of a rule is known as the rule antecedent or precondition. The "THEN" part (or right side) is the rule consequent.

Following are the various advanced classification methods:

- **Bayesian belief networks:** which unlike na¨ıve Bayesian classifiers, do not assume class conditional independence.
- **Classification by Backpropagation**: Backpropagation is a neural network learning algorithm. A neural network is a set of connected input/output units in which each connection has a weight associated with it.
- **Support Vector Machines**: a method for the classification of both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors).
- **Classification Using Frequent Patterns**: Frequent patterns show interesting relationships between attribute–value pairs that occur frequently in a given data set.

CONCLUSION

In this way we have explored the concept of data classification and implemented the data classification algorithm.

ORAL QUESTION

1. What is the primary objective of a classification algorithm?
2. What are some common types of classification algorithms?
3. How do you evaluate the performance of a classification algorithm?

ASSIGNMENT 10

PROBLEM STATEMENT

Perform the data clustering algorithm using any Clustering algorithm.

OBJECTIVE

Students should be able understand the concept of data clustering and different data clustering algorithm.

THEORY

Data clustering

- Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets.
- Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters.
- The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clustering on the same data set. The partitioning is not performed by humans, but by the clustering algorithm.
- Cluster analysis has been widely used in many applications such as image pattern recognition, Web search, biology, and security.

Partitioning methods

- Given a set of n objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$.
- That is, it divides the data into k groups such that each group must contain at least one object. In other words, partitioning methods conduct one-level partitioning on data sets. The basic partitioning methods typically adopt exclusive cluster separation. That is, each object must belong to exactly one group.
- Most partitioning methods are distance-based. Given k, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are “close” or related to each other, whereas objects in different clusters are “far apart” or very different.
- Most applications adopt popular heuristic methods, such as greedy approaches like the k-means and the k-medoids algorithms, which progressively improve the clustering quality and approach a local optimum.

Hierarchical methods

- A hierarchical method creates a hierarchical decomposition of the given set of data objects.
- A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed.
- The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group.
- It successively merges the objects or groups close to one another, until all the groups are merged into one (the topmost level of the hierarchy), or a termination condition holds.
- The divisive approach, also called the top-down approach, starts with all the objects in the same cluster.
- In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds. Hierarchical clustering methods can be distance-based or density- and continuity based.

Method	General Characteristics
Partitioning methods	– Find mutually exclusive clusters of spherical shape – Distance-based – May use mean or medoid (etc.) to represent cluster center – Effective for small- to medium-size data sets
Hierarchical methods	– Clustering is a hierarchical decomposition (i.e., multiple levels) – Cannot correct erroneous merges or splits – May incorporate other techniques like microclustering or consider object "linkages"

CONCLUSION

In this way we have explored the concept of data clustering and implemented the data clustering algorithm.

ORAL QUESTION

1. What is the primary objective of clustering algorithms?
2. What are some common types of clustering algorithms?
3. Describe the process of preparing data for clustering.
4. Can you provide an example of a real-world application where clustering algorithms are commonly used?