

Practical - 1 :-

Step 1 :- Start VMware -> Start Ubuntu/Linux based Machine

Step 2 :- Start Quick Terminal and perform below commands

1) Hadoop Version:

Description: The Hadoop fs shell command version prints the Hadoop version.

Command: `hadoop version`

2) Make Directory:

Description: This command creates the directory in HDFS if it does not already exist.

Note: If the directory already exists in HDFS, then we will get an error message that the file already exists.

Command: `hadoop fs -mkdir /path/directory_name`

3) Listing Directories: **Description:** Using the ls command, we can check for the directories in HDFS.

Command: `hadoop fs -ls /`

4) copyFromLocal or put:

Description:

The Hadoop fs shell command is similar to the copyFromLocal, which copies files or directory from the local filesystem to the destination in the Hadoop filesystem.

Command: `hadoop fs -put ~/localfilepath /fileofdestination`

5) count: **Command:** `hadoop fs -count /`

6) cat :- **Description:** The cat command reads the file in HDFS and displays the content of the file on console or stdout.

Command: `hadoop fs -cat /path`

7) touchz

Description: touchz command creates a file in HDFS with file size equal to 0 byte. The directory is the name of the directory where we will create the file, and filename is the name of the new file we are going to create

Command: `hadoop fs -touchz /directory/file`

8) stat: Description: The Hadoop fs shell command stat prints the statistics about the file or directory in the specified format.

Formats:

%b - file size in bytes

%g - group name of owner

%n - file name

%0 - block size

%r - replication

%u - user name of owner

%y - modification date

If the format is not specified then %y is used by default.

Command: `hadoop fs -stat %format /path`

9) checksum :

Description: The Hadoop fs shell command checksum returns the checksum information of a file.

Command: `hadoop fs -checksum /path`

10) usage :

Description: The Hadoop fs shell command usage returns the help for an individual command.

Command: `hadoop fs -usage [command]`

11) help : Description: The Hadoop fs shell command help shows help for all the commands or the specified command.

Command: `hadoop fs -help [command] -rw-r -- r -`

14) mv : Description: The HDFS mv command moves the files or directories from the source to a destination within HDFS.

Command: `hadoop fs -mv <sre> <des>`

13) cp : Description: The cp command copies a file from one directory to another directory within the HDFS.

Command: `hadoop fs -cp <src> <des>`

Practical - 2 :-

1. Wordcount :

- Step 1 :- Start VMware -> Start Ubuntu/Linux based Machine
- Step 2 :- create a text file with words
- Step 3 :- Start Quick Terminal
- Step 4 :- ls
- Step 5 :- pwd
- Step 6 :- cat > /path of the text file created(eg:-file.txt)
- Step 7 :- hdfs dfs -ls /
- Step 8 :- hdfs dfs -mkdir /directory(folder)name(eg:-newdir)
- Step 9 :- hdfs dfs -put /home/cloudera/file.txt /newdir/
- Step 10 :- hdfs dfs -cat /newdir/file.txt
- Step 11 :- Start Eclipse and create a simple java project
- Step 12 :- Create a class file name : WordCount.java (Code provided below)
- Step 13 :- Save and Export the project into jar file(WC.jar)
- Step 14 :- hadoop jar /home/cloudera/WC.jar WordCount /newdir/file.txt /out1
- Step 15 :- hdfs dfs -ls /out1
- Step 16 :- hdfs dfs -cat /out1/part-r-00000

WordCount.java :-

```
import java.io.IOException;
import java.util.StringTokenizer;
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 {
```

```

StringTokenizer itr = new StringTokenizer(value.toString());
while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
}
}
}

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

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);
}
}

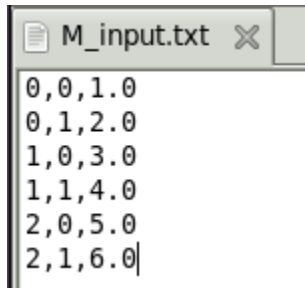
```

2. Matrix Multiplication :-

Step 1 :- Start VMware -> Start Ubuntu/Linux based Machine

Step 2 :- create 2 input text file with numbers

Sample for text file :-



```
0,0,1.0
0,1,2.0
1,0,3.0
1,1,4.0
2,0,5.0
2,1,6.0
```

Step 3 :- Start Quick Terminal

Step 4 :- Make Directories :

```
hadoop fs -mkdir /BdavPrac/input
hadoop fs -mkdir /BdavPrac/output
hadoop fs -ls /BdavPrac
```

Step 5 :- `hadoop fs -put /home/cloudera/Desktop/M_input.txt /BdavPrac/input`

`hadoop fs -put /home/cloudera/Desktop/N_input.txt /BdavPrac/input`

Step 6 :- Start Eclipse and create a simple java project

Step 7 :- Create a class file name : MatrixMultiplication.java (Code provided below)

Step 8 :- Save and Export the project into jar file(MM.jar)

Step 9 :- `hadoop jar /home/cloudera/MM.jar MatrixMultiply /BdavPrac/input/* /BdavPrac/output`

Step 10 :- `hadoop fs -ls /BdavPrac/output`

Step 11 :- `hadoop fs -cat /BdavPrac/output/part-00000`

MatrixMultiplication.java :-

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.ArrayList;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
```

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

```
class Element implements Writable {
    int tag;
    int index;
    double value;

    Element() {
        tag = 0;
        index = 0;
        value = 0.0;
    }

    Element(int tag, int index, double value) {
        this.tag = tag;
        this.index = index;
        this.value = value;
    }

    @Override
    public void readFields(DataInput input) throws IOException {
        tag = input.readInt();
        index = input.readInt();
        value = input.readDouble();
    }

    @Override
    public void write(DataOutput output) throws IOException {
        output.writeInt(tag);
        output.writeInt(index);
    }
}
```

```

        output.writeDouble(value);
    }
}

```

```

class Pair implements WritableComparable<Pair> {
    int i;
    int j;

    Pair() {
        i = 0;
        j = 0;
    }

    Pair(int i, int j) {
        this.i = i;
        this.j = j;
    }

    @Override
    public void readFields(DataInput input) throws IOException {
        i = input.readInt();
        j = input.readInt();
    }

    @Override
    public void write(DataOutput output) throws IOException {
        output.writeInt(i);
        output.writeInt(j);
    }

    @Override
    public int compareTo(Pair compare) {
        if (i > compare.i) {
            return 1;
        } else if (i < compare.i) {
            return -1;
        } else {
            if (j > compare.j) {
                return 1;
            } else if (j < compare.j) {

```

```

        return -1;
    }
}
return 0;
}

@Override
public String toString() {
    return i + " " + j + " ";
}
}

public class MatrixMultiply {
    public static class MatrixMapperM extends
        Mapper<Object, Text, IntWritable, Element> {
        @Override
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            String readLine = value.toString();
            String[] tokens = readLine.split(",");
            int index = Integer.parseInt(tokens[0]);
            double elementVal = Double.parseDouble(tokens[2]);
            Element e = new Element(0, index, elementVal);
            IntWritable keyval = new IntWritable(Integer.parseInt(tokens[1]));
            context.write(keyval, e);
        }
    }

    public static class MatrixMapperN extends
        Mapper<Object, Text, IntWritable, Element> {
        @Override
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            String readLine = value.toString();
            String[] tokens = readLine.split(",");
            int index = Integer.parseInt(tokens[1]);
            double elementVal = Double.parseDouble(tokens[2]);
            Element e = new Element(1, index, elementVal);
            IntWritable keyval = new IntWritable(Integer.parseInt(tokens[0]));
            context.write(keyval, e);
        }
    }
}

```



```

    }
}

public static class ReducerMN extends
    Reducer<IntWritable, Element, Pair, DoubleWritable> {
    @Override
    public void reduce(IntWritable key, Iterable<Element> values,
        Context context) throws IOException,
        InterruptedException {
        ArrayList<Element> M = new ArrayList<Element>();
        ArrayList<Element> N = new ArrayList<Element>();
        Configuration conf = context.getConfiguration();
        for (Element element : values) {
            Element temp =
                ReflectionUtils.newInstance(Element.class, conf);
            ReflectionUtils.copy(conf, element, temp);
            if (temp.tag == 0) {
                M.add(temp);

            } else if (temp.tag == 1) {
                N.add(temp);
            }
        }
        for (int i = 0; i < M.size(); i++) {
            for (int j = 0; j < N.size(); j++) {
                Pair p = new Pair(M.get(i).index, N.get(j).index);
                double mul = M.get(i).value * N.get(j).value;
                context.write(p, new DoubleWritable(mul));
            }
        }
    }
}

public static class MapMN extends
    Mapper<Object, Text, Pair, DoubleWritable> {
    @Override
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        String readLine = value.toString().trim();
        // Replace all tabs and multiple spaces with a single space

```

```

        readLine = readLine.replaceAll("\\s+", " ");
        // Split based on a single space
        String[] pairValue = readLine.split(" ");
        if (pairValue.length == 3) {
            try {
                int first = Integer.parseInt(pairValue[0]);
                int second = Integer.parseInt(pairValue[1]);
                double val = Double.parseDouble(pairValue[2]);
                Pair p = new Pair(first, second);
                context.write(p, new DoubleWritable(val));
            } catch (NumberFormatException e) {
                // Handle parsing errors, log them if necessary
                System.err.println("Error parsing input: " +
readLine);
            }
        } else {
            // Handle the case where the input format is incorrect
            throw new IOException("Invalid input format: " +
readLine);
        }
    }
}

```

```

public static class ReduceMN extends
    Reducer<Pair, DoubleWritable, Pair, DoubleWritable> {
    @Override
    public void reduce(Pair key, Iterable<DoubleWritable> values,
        Context context) throws IOException,
InterruptedException {
        double sum = 0.0;
        for (DoubleWritable value : values) {
            sum += value.get();
        }
        context.write(key, new DoubleWritable(sum));
    }
}

```

```

public static void main(String[] args) throws Exception {
    Path MPath = new Path("/bdavMM/input/M_input.txt");
    Path NPath = new Path("/bdavMM/input/N_input.txt");
}

```

```

        Path intermediatePath = new Path("/bdavMM/input/harshal");
        Path outputPath = new Path("/bdavMM/input/output");
        Job job1 = Job.getInstance(new Configuration(), "Map Intermediate");
        job1.setJarByClass(MatrixMultiply.class);
        MultipleInputs.addInputPath(job1, MPath, TextInputFormat.class,
MatrixMapperM.class);
        MultipleInputs.addInputPath(job1, NPath, TextInputFormat.class,
MatrixMapperN.class);
        job1.setReducerClass(ReducerMN.class);
        job1.setMapOutputKeyClass(IntWritable.class);
        job1.setMapOutputValueClass(Element.class);
        job1.setOutputKeyClass(Pair.class);
        job1.setOutputValueClass(DoubleWritable.class);
        job1.setOutputFormatClass(TextOutputFormat.class);
        FileOutputFormat.setOutputPath(job1, intermediatePath);
        job1.waitForCompletion(true);
        Job job2 = Job.getInstance();
        job2.setJobName("Map Final Output");
        job2.setJarByClass(MatrixMultiply.class);
        job2.setMapperClass(MapMN.class);
        job2.setReducerClass(ReduceMN.class);
        job2.setOutputKeyClass(Pair.class);
        job2.setOutputValueClass(DoubleWritable.class);
        job2.setInputFormatClass(TextInputFormat.class);
        job2.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job2, intermediatePath);
        FileOutputFormat.setOutputPath(job2, outputPath);
        job2.waitForCompletion(true);
    }
}

```

Practical - 3 :-

Step 1 :- Start MongoDB Compass app if installed

Step 2 :- Create a new Database(bdavdb) and a collection(students)

Step 3 :- Open Visual Studio and Start the terminal

Step 4 :- Run following Commands :

1. *mongosh*
2. *use [DB_NAME](employee)*
3. *show employee*
4. **Insert new documents :**
*db.employee.insertMany([
 {id:1, name:"Harshal",gender: "male"},
 {id:2, name:"Jhon",gender: "male"},
 {id:3, name: "Pradeep", gender: "male"},
 {id:1, name: "peeti", gender: "female"}])*
5. **Verify the Collection :**
db.myCollection.find()
6. **Insert new single document :**
*db.employee.insertOne(
 {id:1, name:"Harshal",gender: "male"})*
7. **Verify the Collection :**
db.myCollection.findOne()
8. **Update :**
*db.myCollection.updateOne(
 { "name": "Jhon" },
 { \$set: { "gender": "Trans" } })*
9. **Delete :**
db.myCollection.deleteOne({"name":"peeti"})
10. *db.myCollection.find().pretty()*
11. **Indexing :**
 - a. **Create :**
db.myCollection.createIndex ({ "name": 1})
db.myCollection.getIndexes ()
 - b. **Drop :**
db.myCollection.dropIndex("name_1")
- 12.

Practical - 4 :-

Code and Output :-

1. Start Hive :-

- *sudo service zookeeper-server start*
- *sudo hive;*

2. Create and Show Database :-

- *Create database bdav;*
- *Show databases;*

3. Create and show tables :-

- *CREATE TABLE emp_36 (
id INT,
name STRING,
city STRING,
dept STRING,
salary INT,
Domain STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';*
- *SHOW TABLES;*

4. Load Data into table epm_36 using .csv file and Display it :-

- *LOAD DATA LOCAL INPATH 'employee.csv' INTO TABLE emp_36;*
- *SELECT * FROM emp_36;*

5. Create and show Partition Table :-

- *CREATE TABLE employee_part(
emp_id INT,*

*emp_name STRING,
emp_salary FLOAT)
PARTITIONED BY (dept STRING)*

- *SHOW TABLES;*

6. Creating View :-

- *CREATE VIEW high_salary_employees AS
SELECT id, name, city, dept, salary
FROM emp_36 WHERE salary > 50000;*
- *Select * FROM high_salary_employees;*

7. In-Built Functions :-

- *SELECT id, CONCAT(name, ' from ', city) AS name_with_city FROM
emp_36;*
- *SELECT id, name, DATE_ADD(CURRENT_DATE, 365) AS
next_year_date FROM emp_36;*
- *SELECT AVG(salary) AS average_salary FROM emp_36;*
- *SELECT dept, SUM(salary) AS total_salary FROM emp_36 GROUP BY
dept;*
- *select max(salary) from emp_36;*

8. Join Operation in Hive :-

- *CREATE TABLE sales (*
- *sale_id INT,*
- *product_id INT,*
- *quantity INT,*
- *sale date DATE,*
- *total amount DECIMAL(10, 2));*
- *CREATE TABLE product (*

product_id INT,
product_name STRING,
category STRING,
price DECIMAL(10, 2));

- *select * from sales;*
- *select * from product;*

a. Inner join :

- *SELECT s.sale_id, p.product_name, s.quantity, s.total_amount
FROM sales s
JOIN product p ON s.product_id = p.product_id;*

b. Left join :

- *SELECT s.sale_id, p.product_name, s.quantity, s.total_amount
FROM sales s
LEFT JOIN product p ON s.product_id = p.product_id;*

c. Right Join :

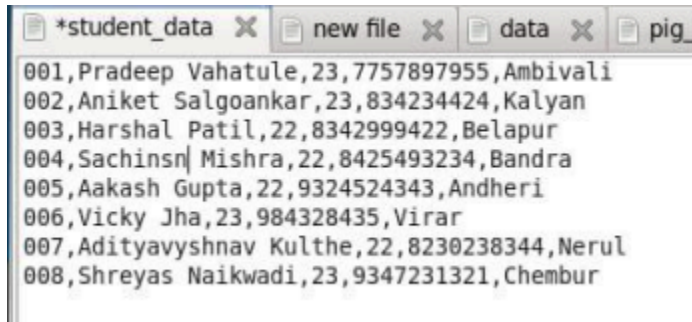
- *SELECT s.sale_id, p.product_name, s.quantity, s.total_amount
FROM sales s
RIGHT JOIN product p ON s.product_id = p.product_id;*

d. Full-Outer Join :

- *SELECT s.sale_id, p.product_name, s.quantity, s.total_amount
FROM sales s
FULL OUTER JOIN product p ON s.product_id = p.product_id;*

Practical 5 -

student_data' :



1. pig -x local
student_info = LOAD '/home/cloudera/Desktop/student_data' USING PigStorage(',') AS
(
id: int,
first_name: chararray,
last_name: chararray,
age: int,
contact: chararray,
city: chararray
);
2. Dump **student_info**;
3. Store **Student_info**:
Code: Store **student_info** into 'sampleoutput' using PigStorage('|');
4. Describe :
describe **student_info**
5. Filter :
Filter_stud = filter **student_info** by age>22;
DUMP **Filter_stud**
6. Explain :
explain **Filter_stud**
7. ForEach :
foreachstudent = foreach **filterstudent** generate id, first_name, age, city;
DUMP **foreachstudent**

8. Groupstudent :

- a. **grpstud** = group **student_info** by city;
DUMP **grpstud**
- b. **groupstudent2** = group **student_info** by (city,age);
dump **groupstudent2**;

9. Illustrate **foreachstudent**

10. Joins :

- a. Inner Join :
InnerJoin = Join **student_info** by city, I by city;
dump InnerJoin
- b. Left Join :
LeftJoin = Join **student_info** by city LEFT, I by city;
dump LeftJoin
- c. Right Join :
RightJoin = Join **student_info** by city right outer, I by city;
dump RightJoin
- d. Full Join :
FullJoin = Join **student_info** by city full outer, I by city;
dump FullJoin

11. Order :

- a. Ascending :
A1 = order **student_info** by id asc;
dump A1
- b. Descending :
B5 = order **student_info** by id desc;
dump B5

12. Limit : **limited_data** = LIMIT **student_info** 3;

13. Union : **Combined_Data** = union **student_info**, **limited_data**;

14. Split :

split **student_info** into *younger_students* if age<23, *older_students* if age>=23;
dump **student_info**;

Practical 6 -

1. gedit pgtest
2. cat pgtest
3. hdfs dfs -put pgtest /Pgroup
4. hdfs dfs -ls /Pgroup
5. Start Spark : *spark-shell*
6. Load Data into RDD(Resilient Distributed Dataset) : *val mydata = sc.textFile("pgtest")*
7. Count lines in RDD : *mydata.count()*
8. Check Application Name : *sc.appName*
9. Iterates to print first lines from data : *for (line <- mydata.take(2)) { println(line) }*
10. *mydata.map(line => line.toUpperCase)*: Transforms each line in the RDD to uppercase.
11. *mydata.filter(line => line.endsWith("H"))* : Filters lines that end with "H."
12. *datanum.parallelize(List(10, 20, 30))* : Creates an RDD from a local collection of numbers.
13. *datanum.collect()* : Collects the data from the RDD into an array.
14. *datanum.map(x => x + 10)* : Adds 10 to each element in the RDD.
15. *val name = Seq("Krushna", "Jadhav")* : Creates a sequence of names.
16. *name.map(_._toLowerCase)* : Converts each name to lowercase.
17. *name.flatMap(_._toLowerCase)* : Flattens the lowercase characters into a single list.
18. *sc.textFile("a1.txt")* : Loads the content of "a1.txt" into an RDD.
19. *a1.collect()* : Collects the data from the RDD.
20. *a1.count()* : Counts the number of lines in the RDD.
21. *Val splitdata = a1.flatMap(line => line.split(" "))* : Splits each line into words and flattens them.
22. *splitdata.collect()*
23. *Val mapdata = splitdata.map(word => (word, 1))*: Maps each word to a key-value pair with a count of 1.
24. *mapdata.collect()*
25. *Val reducedata = mapdata.reduceByKey(_+_)* : Reduces data by summing values for each key.
26. *reducedata.collect()*

Practical -7 :-

Installation of Power BI

Practical - 8 :-

Data Preprocessing activities in Power BI

Dataset Link - https://github.com/Ayushi0214/Datasets/tree/main/pizza_sales

1. Load the dataset into PowerBI via Folder and click on “Transform Data” and Power Query Editor will be opened.
2. **Basic Data Transformations:**
Example: Splitting a "Full Name" column into separate "First Name" and "Last Name" columns.
Right Click on Binary of orders.csv > Add as new Query.
Double tap the orders.csv
Do this for all files
To Transform data, go to the Transform tab at the top and then click on “Statistics”
Transform a column by multiplying or other options use the “Standard” option given besides Statistics
3. **Dealing With Text Tools:**
Dataset - https://github.com/Ayushi0214/Power_BI_course/blob/main/classicmodelszip.zip
Import and click on “Transform Data”
Add as a new query, customers and employee files.
Merge the columns by selecting multiple with shift key.
In the transform tab, click on Merge columns. Click ok.
Trim whitespaces by using (Transform -> Format -> Trim)
Split => By delimiter (Transform -> Split Column)
Extract Email before @ (Add Column -> Extract)
4. **Handling Unwanted Columns and Null Values:**
Remove duplicates by right clicking on the column
Replace NULL values from transform tab (Transform -> Replace Values)
5. **Dealing with Numerical Tools:**
Add Column => Custom Column => = [priceEach]*[quantityOrdered]
Rounding Up => (Transform -> Rounding)
Explore in Scientific Formats in Transform Tab

6. Dealing with Date and Time:

Select any column with dates and replace NULL with 0.

Then change the data type: text to date in Transform Tab

And Remove errors by right clicking on the column

Adding column of Quarter (Add Column -> Date)

Custom column => [shippedDate]-[orderDate] to get the duration

Total days required to ship product (Add Column -> Duration -> Total Days)

Practical - 9 :-

Handling of tables and Queries in Power BI

Using dataset https://github.com/Ayushi0214/Power_BI_course/blob/main/classicmodelszip.zip

1. Merge Queries and Append Queries

Merge Queries

Select the primary table you want to merge, here order details. Merging products and order details table based on common Product Code.

In the Home Tab, click on “Merge Queries”.

In the Merge dialog box:

- Select the secondary table you want to merge with. Select product table
- Choose the columns in both tables that you want to join on.
- Select the type of join (e.g., Inner, Left Outer, Right Outer, Full Outer). Here: Inner Join

Click OK to create the merged column.

Calculating margin from Buy Price and quantity by creating custom column

Calculating Profit using Custom Columns => sales - margin

Append Queries

Go to the Home tab. Click on Append Queries.

Choose whether to append two tables or multiple tables. Select the tables you want to append.

Click OK to combine the tables into a new query

2. Column Formats

Use the Column tools to set additional formatting options (e.g., currency, percentage). Steps to Format Columns:

1. In the Power Query Editor, select the column you want to format
2. Changing CreditLimit format from whole number to fixed decimal Number in Customers Table (Transform -> Data Type)
3. Icon changes to \$
4. Click on using locale by clicking the column icon of requiredDate in orders table.
5. Changing Indian Date format to USA Date Format.

3. Creating a Table

1. In Power BI Desktop, navigate to the Home tab. Click on Enter Data.
2. In the data entry window, input your column names and data.
3. Click OK to create a new table.

4. Pivoting and Unpivoting of Data

Unpivoting Data => transforming rows to columns

1. Suppose we have data like this and we are asked to convert this table to vertical table. In the Power Query Editor, select the columns you want to unpivot.
2. Go to Transform > Unpivot other columns
3. This will create two new columns: Attribute and Value.

Pivoting Data

1. In the Power Query Editor, select the column you want to pivot.
2. Now click on column and click on “Pivot Column” in Transform Tab
3. Choose the value column for aggregation.
4. Click OK to create the pivoted data structure.

5. Managing Data Relationships

1. Navigate to the Model view in Power BI Desktop.
2. Close & Apply the Query editor. A model with all tables will be formed.
3. Drag and drop fields between tables to create a relationship. Since the Orders table and Customers table have a common column “CustomerNumber”. We can drag this column from Orders and put it in Customers table and it will create a relationship.

4. In the relationship settings, specify the cardinality (one-to-many, many-to-one, etc.).
5. Set the cross-filter direction (single or both).

6. Cardinality and Cross-Filter Direction

1. In the Model view, select the relationship line connecting two tables.
2. Right-click and select Properties.
3. In the Edit Relationship dialog, set the Cardinality:
 - a. One-to-Many: One value in a column corresponds to many values in another column. This is the most common relationship type.
 - b. Many-to-One: Reverse of one-to-many, where multiple records in one table are linked to a single record in another.
 - c. Many-to-Many: Both tables can have multiple matching records, commonly used when there is no single unique key.
4. Choose the Cross-filter direction:
 - a. Single Direction: Filters flow in one direction only. This is useful when one table is dependent on another (e.g., filtering products by product category).
 - b. Both Directions: Filters flow in both directions, which means a change in one table can filter data in both related tables. This is useful for complex models where both tables need to influence each other

Practical - 10 :-

Data Visualization and dashboard creation in Power BI

Launch Power BI Desktop and load data.