```
create database modeldetail;
use modeldetail;
CREATE TABLE employee (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    department_id INT,
    salary INT,
    manager_id INT,
    FOREIGN KEY (department_id) REFERENCES
department(department_id),
    FOREIGN KEY (manager_id) REFERENCES employee(employee_id)
);

CREATE TABLE department (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);

CREATE TABLE project (
    project_id INT PRIMARY KEY,
    project_name VARCHAR(100)
);

CREATE TABLE employee_project (
    employee_id INT,
    project_id INT,
    PRIMARY KEY (employee_id, project_id),
    FOREIGN KEY (employee_id) REFERENCES employee(employee_id),
```

```
    FOREIGN KEY (project_id) REFERENCES project(project_id)
);

CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100),
    email VARCHAR(50) UNIQUE,
    phone VARCHAR(20),
    address VARCHAR(100)
);

CREATE TABLE product (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price INT,
    stock INT
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    order_total INT,
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);
```

```
CREATE TABLE order_details (
    order_id INT,
    product_id INT,
    quantity INT,
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id)
);


CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY,
    order_id INT,
    transaction_date DATE,
    payment_method VARCHAR(50),
    amount INT,
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
);


CREATE TABLE students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(100),
    email VARCHAR(100),
    department_id INT,
    year INT,
    FOREIGN KEY (department_id) REFERENCES
department(department_id)
```

);

INSERT INTO department VALUES

(3001,'it'),

(3002, 'hr');

INSERT INTO employee VALUES

(2, 'sita', 3001, 60000, NULL),

(1, 'ram', 3001, 70000, 2),

(3, 'gopal', 3002, 75000, 2);

INSERT INTO project VALUES

(11,'website'),

(12, 'mobile app');

INSERT INTO customer VALUES

(1, 'johnson', 'johnson@gmail.com', 123456789,'chennai'),

(2, 'scarlet', 'scarlet@gmail.com', 987654321, 'vellore');

INSERT INTO product VALUES

(101, 'laptop', 'electronics', 40000,20),(102, 'phone', 'electronics', 500,100),(103, 'shoes', 'apparel', 500,150);

INSERT INTO students VALUES

(1, 'bond', 'electronics', 3001,20),(2, 'allen', 'electronics', 3001,100),(3, 'harris', 'apparel', 3002,150);

```
INSERT INTO orders VALUES
(1001, 1, '2025-03-10', 1500),
(1002, 2, '2025-03-11', 600);

INSERT INTO orders VALUES (1003, 3, '2025-03-12', 160);

INSERT INTO transactions VALUES
(2001,1001,  '2025-03-10','credit card', 1500),
(2002,1002,  '2025-03-11', 'debit card',600);

#write a pl\sql function that accepts an employeeid and returns full name of the employee from employee table
DELIMITER //

CREATE FUNCTION get_employee_name(p_employee_id INT)
RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
   DECLARE v_employee_name VARCHAR(100);

   SELECT employee_name INTO v_employee_name
   FROM employee
   WHERE employee_id = p_employee_id;

   RETURN IFNULL(v_employee_name, 'Employee Not Found');
END;
//
```

```
DELIMITER ;

#write a pl\sql triggers that automatically updates the last updated timestamp whenever an employee salary is modified
ALTER TABLE employee ADD last_updated DATETIME;
DELIMITER //

CREATE TRIGGER trg_update_salary_timestamp
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
   IF OLD.salary != NEW.salary THEN
      SET NEW.last_updated = NOW();
   END IF;
END;
//

DELIMITER ;

#write pl\sql procedure that accepts product id and quantity then updattes the inventory in product table accordingly
DELIMITER //

CREATE PROCEDURE update_inventory (
   IN p_product_id INT,
   IN p_quantity INT
)
```

```
BEGIN
    DECLARE v_stock INT;

    -- Get current stock
    SELECT stock INTO v_stock
    FROM product
    WHERE product_id = p_product_id;

    -- Check stock and update
    IF v_stock < p_quantity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Not enough stock available.';
    ELSE
        UPDATE product
        SET stock = stock - p_quantity
        WHERE product_id = p_product_id;
    END IF;
END;
//

DELIMITER ;
CALL update_inventory(101, 5);

#create pl \sql cursor that retrives all employee in "it" department and
their names and salaries
DELIMITER //

CREATE PROCEDURE get_it_employees()
```

```
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_name VARCHAR(100);
    DECLARE v_salary INT;

    -- Cursor for selecting IT department employees
    DECLARE it_cursor CURSOR FOR
        SELECT e.employee_name, e.salary
        FROM employee e
        JOIN department d ON e.department_id = d.department_id
        WHERE LOWER(d.department_name) = 'it';

    -- Handler for end of data
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Open and loop through the cursor
    OPEN it_cursor;

    read_loop: LOOP
        FETCH it_cursor INTO v_name, v_salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- You can do any processing here; for now, just select output
        SELECT CONCAT('Name: ', v_name, ', Salary: ', v_salary) AS
employee_info;
    END LOOP;
```

```
    CLOSE it_cursor;
END;
//

DELIMITER ;
CALL get_it_employees();

#pl/sql procedure that print all employee names from employee table
DELIMITER //

CREATE PROCEDURE print_employee_names()
BEGIN
    SELECT employee_name FROM employee;
END;
//

DELIMITER ;
CALL print_employee_names();
```

QUERY:

2. a.Write a SQL query to retrieve all employees who have worked on more than 2 projects.

SELECT Employee_ID, Employee_Name FROM Employee GROUP BY Employee_ID, Employee_Name HAVING COUNT(DISTINCT Project_ID) > 2;

b. Write a SQL query to show the number of projects they worked on.

SELECT Employee_ID, Employee_Name, COUNT (DISTINCT Project_ID) AS Project_Count FROM Employee GROUP BY Employee_ID, Employee_Name;

3. Write a SQL query using Nested Queries to find employees who earn more than the average salary of their department.

SELECT P.Product_Name, SUM(T.Amount) AS Total_Sales FROM Product P JOIN OrderTable O ON P.Product_ID = O.Order_ID JOIN TransactionTable T ON O.Order_ID = T.Order_ID GROUP BY P.Product_Name ORDER BY Total_Sales DESC LIMIT 5

4. Write a SQL query to find the top 5 products by total sales using aggregate functions and grouping.

SELECT P.Product_Name, SUM(T.Amount) AS Total_Sales FROM Product P JOIN OrderTable O ON P.Product_ID = O.Order_ID JOIN TransactionTable T ON O.Order_ID = T.Order_ID GROUP BY P.Product_Name ORDER BY Total_Sales DESC LIMIT 5;

5. Write a SQL query that uses a JOIN to list all customers and their orders. Ensure to only show customers who have placed orders.

SELECT C.Customer_ID, C.Customer_Name, O.Order_ID, O.Order_Date, O.Order_Total FROM Customer C JOIN OrderTable O ON C.Customer_ID = O.Customer_ID;

6. Write a SQL query that uses a VIEW to display the names of employees, their department, and their total salary (salary + bonus).

CREATE VIEW EmployeeSalaryView AS SELECT Employee_Name, Department, Salary, (Salary * 0.10) AS Bonus, (Salary + Salary * 0.10) AS Total_Salary FROM Employee; SELECT * FROM EmployeeSalaryView;

SELECT * FROM customer WHERE customer_name LIKE 'A%';

write a Python program using MongoDB to fetch all documents from a users collection where the user's age is greater than 25 and sort them by their last login date

from pymongo import MongoClient

from datetime import datetime

Step 1: Connect to MongoDB

```
client = MongoClient("mongodb://localhost:27017/")  # Replace with your MongoDB connection string
db = client["blog_database"]  # Replace with your database name
users_collection = db["Users"]  # Replace with your collection name

# Step 2: Query to fetch users where age > 25 and sort by last_login_date
query = {"age": {"$gt": 25}}
sort_order = [("last_login_date", 1)]  # 1 for ascending, -1 for descending

# Step 3: Execute the query and fetch documents
users = users_collection.find(query).sort(sort_order)

# Step 4: Print the results
for user in users:
    print(f"Username: {user['username']}, Age: {user['age']}, Last Login: {user['last_login_date']}")
pip install pymongo

{
  "_id": ObjectId("..."),
  "username": "john_doe",
  "age": 28,
  "last_login_date": ISODate("2025-04-21T10:00:00Z")
}
```

write javaprogram using jdbc to connect databasevand retrieve list of all employees working in a specific department display results

```
import java.sql.*;


public class EmployeeByDepartment {
    public static void main(String[] args) {
        // Replace with your actual DB details
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
        String dbUser = "your_username";
        String dbPassword = "your_password";


        String department = "Sales"; // Example department
        String sql = "SELECT employee_id, name, position FROM employees
WHERE department = ?";


        try (
            Connection connection = DriverManager.getConnection(jdbcURL,
dbUser, dbPassword);
            PreparedStatement statement = connection.prepareStatement(sql)
        ) {
            statement.setString(1, department); // Set department value in query
            ResultSet resultSet = statement.executeQuery();


            System.out.println("Employees in " + department + " Department:");
            System.out.println("----------------------------------------");


            while (resultSet.next()) {
                int id = resultSet.getInt("employee_id");
```

```
            String name = resultSet.getString("name");
            String position = resultSet.getString("position");

            System.out.println("ID: " + id + ", Name: " + name + ", Position: " +
position);
        }

    } catch (SQLException e) {
        System.out.println("Database error:");
        e.printStackTrace();
    }
  }
}
```

Write a JDBC program to insert a new record into the Students tahle. Ensure that the program checks for any duplicate records before inserting

```
import java.sql.*;

public class InsertStudentWithCheck {
  public static void main(String[] args) {
    // Database credentials and URL
    String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
    String dbUser = "your_username";
    String dbPassword = "your_password";

    // Student data to insert
    int studentId = 101;
    String name = "John Doe";
```

```
        String email = "john.doe@example.com";


        try (
            Connection connection = DriverManager.getConnection(jdbcURL,
dbUser, dbPassword)
        ) {
            // Step 1: Check for duplicate
            String checkSQL = "SELECT * FROM Students WHERE student_id =
?";
            PreparedStatement checkStmt =
connection.prepareStatement(checkSQL);
            checkStmt.setInt(1, studentId);


            ResultSet resultSet = checkStmt.executeQuery();


            if (resultSet.next()) {
                System.out.println("Student with ID " + studentId + " already
exists.");
            } else {
                // Step 2: Insert new student
                String insertSQL = "INSERT INTO Students (student_id, name,
email) VALUES (?, ?, ?)";
                PreparedStatement insertStmt =
connection.prepareStatement(insertSQL);
                insertStmt.setInt(1, studentId);
                insertStmt.setString(2, name);
                insertStmt.setString(3, email);


                int rowsInserted = insertStmt.executeUpdate();
```

```
            if (rowsInserted > 0) {
                System.out.println("New student inserted successfully.");
            }
        }

    } catch (SQLException e) {
        System.out.println("Database error:");
        e.printStackTrace();
    }
  }
}
```

Using JDBC, write a program to delete a specific record from the orders table based on the OrderlD provided by the user.

```
import java.sql.*;
import java.util.Scanner;

public class DeleteOrder {
   public static void main(String[] args) {
      // Database connection details
      String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
      String dbUser = "your_username";
      String dbPassword = "your_password";

      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter OrderID to delete: ");
      int orderId = scanner.nextInt();
```

```
        String deleteSQL = "DELETE FROM orders WHERE order_id = ?";


        try (
            Connection connection = DriverManager.getConnection(jdbcURL,
dbUser, dbPassword);
            PreparedStatement statement =
connection.prepareStatement(deleteSQL)
        ) {
            statement.setInt(1, orderId);
            int rowsAffected = statement.executeUpdate();


            if (rowsAffected > 0) {
                System.out.println("Order with ID " + orderId + " deleted
successfully.");
            } else {
                System.out.println("Order with ID " + orderId + " not found.");
            }


        } catch (SQLException e) {
            System.out.println("Database error:");
            e.printStackTrace();
        }
    }
}
```

14. Write a JDBC code to perform a transaction where you insert a new order and update the corresponding customer's order count. Ensure to handle commit and rollback properly.

```
import java.sql.*;

public class OrderTransaction {
    public static void main(String[] args) {
        // Database credentials
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
        String dbUser = "your_username";
        String dbPassword = "your_password";

        // Sample data
        int orderId = 1001;
        int customerId = 501;
        double orderAmount = 250.00;

        try (
            Connection connection = DriverManager.getConnection(jdbcURL,
dbUser, dbPassword)
        ) {
            // Disable auto-commit for manual transaction control
            connection.setAutoCommit(false);

            try {
                // 1. Insert new order
                String insertOrderSQL = "INSERT INTO orders (order_id,
customer_id, amount) VALUES (?, ?, ?)";
                PreparedStatement insertOrderStmt =
connection.prepareStatement(insertOrderSQL);
                insertOrderStmt.setInt(1, orderId);
```

```
            insertOrderStmt.setInt(2, customerId);
            insertOrderStmt.setDouble(3, orderAmount);
            insertOrderStmt.executeUpdate();

            // 2. Update customer's order count
            String updateCustomerSQL = "UPDATE customers SET order_count = order_count + 1 WHERE customer_id = ?";
            PreparedStatement updateCustomerStmt = connection.prepareStatement(updateCustomerSQL);
            updateCustomerStmt.setInt(1, customerId);
            updateCustomerStmt.executeUpdate();

            // If both succeed, commit the transaction
            connection.commit();
            System.out.println("Transaction completed: Order inserted and customer updated.");

        } catch (SQLException e) {
            // On any failure, rollback
            connection.rollback();
            System.out.println("Transaction failed. Rolled back.");
            e.printStackTrace();
        } finally {
            // Restore auto-commit to true (good practice)
            connection.setAutoCommit(true);
        }

    } catch (SQLException e) {
```

```
            System.out.println("Connection error:");
            e.printStackTrace();
        }
    }
}
```

WORD COUNT in hadoop

Cmd: hdfs namenode -format

Cmd: D:\hadoop\hadoop-2.8.0\sbin--start-dfs.cmd

Cmd: start-yarn.cmd

http://localhost:50070

Create a Sample Input File

1. On Z: drive, create folder and file:

Cmd: mkdir Z:\wordcount_input

Cmd: notepad Z:\wordcount_input\sample.txt

SAMPLE:

Hello Hadoop

Welcome to Big Data

Hadoop is cool

Create Input Directory in HDFS

In **CMD**, from D:\hadoop\hadoop-2.8.0\bin, run:

Cmd: hdfs dfs -mkdir /wordcount_input

Cmd: hdfs dfs -put Z:\wordcount_input\sample.txt /wordcount_input

Cmd: hdfs dfs -ls /wordcount_input

Cmd: hadoop jar D:\hadoop\hadoop-2.8.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.8.0.jar wordcount /wordcount_input /wordcount_output

Cmd: hdfs dfs -cat /wordcount_output/part-r-00000

WORDCOUNT IN MAPPER REDUCERS

cd D:\hadoop\hadoop-2.8.0\sbin

Cmd: start-dfs.cmd

Cmd: start-yarn.cmd

Z:

Cmd: cd wordcount

Cmd: notepad input.txt

D:

Cmd: hdfs dfs -mkdir /input

Cmd: hdfs dfs -put input.txt /input

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); // Optional
        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);
    }
}
```

Cmd: javac -classpath "D:\hadoop\hadoop-2.8.0\share\hadoop\common\hadoop-common-2.8.0.jar;D:\hadoop\hadoop-2.8.0\share\hadoop\mapreduce\hadoop-mapreduce-client-core-2.8.0.jar" -d wordcount_classes WordCount.java

Cmd: jar -cvf wordcount.jar -C wordcount_classes/ .

Cmd: hadoop jar wordcount.jar WordCount /input /output

Cmd: hdfs dfs -cat /output/part-r-00000

18. Write a MapReduce program to calculate the total revenue from a fist of tranactions

The transactions contain fields for the transaction ID, date, and amount

Cmd: start-dfs.sh

Cmd: start-yarn.sh

Cmd: D:\hadoop\hadoop-2.8.0\bin ==notepad transactions.txt

1,2024-03-21,100.50

2,2024-03-22,200.75

3,2024-03-23,150.00

Cmd: cd D:\hadoop\scripts==notepad mapper.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    line = line.strip()
    fields = line.split(",")
    if len(fields) != 3:
        continue
    try:
        amount = float(fields[2])
        print(f"Total\t{amount}")
    except ValueError:
        continue
```

Cmd: notepad reducer.py:

```
#!/usr/bin/env python3
import sys

total_revenue = 0.0

for line in sys.stdin:
    line = line.strip()
    key, value = line.split("\t")
    try:
```

```
        total_revenue += float(value)
    except ValueError:
        continue
print(f"Total Revenue: {total_revenue}")
```

Cmd: hdfs dfs -mkdir -p /user/hadoop/input

Cmd: hdfs dfs -put transactions.txt /user/hadoop/input/

Cmd: hdfs dfs -ls /user/hadoop/input

Cmd: python mapper.py

Cmd: python reducer.py

Cmd: hdfs dfs -ls /user/hadoop/output

Cmd: type transactions.txt | python mapper.py

Cmd: type transactions.txt | python mapper.py | python reducer.py

Cmd: hadoop jar %HADOOP_HOME%\share\hadoop\tools\lib\hadoop-streaming-*.jar ^

-input /user/hadoop/input/transactions.txt ^

-output /user/hadoop/output ^

-mapper "python mapper.py" ^

-reducer "python reducer.py"

Cmd:hdfs dfs -cat /user/hadoop/output/part-00000

In Hadoop.implement MapReduce job that finds the average sales amount per product from a dataset of sales records stored in hdfs

Cmd:Notepad mapper.py

```
#!/usr/bin/env python3
import sys

for line in sys.stdin:
    line = line.strip()
```

```
    fields = line.split(',')
    if len(fields) != 3:
        continue
    try:
        product = fields[1]
        amount = float(fields[2])
        print(f"{product}\t{amount}")
    except ValueError:
        continue
```

Cmd:notepadreducer.py

```
#!/usr/bin/env python3
import sys

current_product = None
total = 0.0
count = 0

for line in sys.stdin:
    line = line.strip()
    product, value = line.split('\t')

    try:
        amount = float(value)
    except ValueError:
        continue

    if current_product == product:
```

```
        total += amount
        count += 1
    else:
        if current_product:
            avg = total / count
            print(f"{current_product}\t{avg:.2f}")
        current_product = product
        total = amount
        count = 1

if current_product:
    avg = total / count
    print(f"{current_product}\t{avg:.2f}")
```

sales.txt:

```
1,Soap,50.0
2,Shampoo,100.0
3,Soap,70.0
4,Shampoo,130.0
```

Cmd:D:\hadoop\scripts\Put mapper.py, reducer.py, and sales.txt

Cmd:cd D:\hadoop\scripts

Cmd:hdfs dfs -mkdir -p /user/hadoop/input

Cmd:hdfs dfs -put sales.txt /user/hadoop/input/

Cmd:hadoop jar %HADOOP_HOME%\share\hadoop\tools\lib\hadoop-streaming-*.jar ^

```
 -input /user/hadoop/input/sales.txt ^
 -output /user/hadoop/output_avg ^
 -mapper "python mapper.py" ^
```

-reducer "python reducer.py"

-mapper "python D:/hadoop/scripts/mapper.py"

-reducer "python D:/hadoop/scripts/reducer.py"

Cmd:hdfs dfs -cat /user/hadoop/output_avg/part-00000

SANTHOSH S
3122246002011

EX.NO:1	**DATABASE DESIGN AND NORMALIZATION**
03.01.25	

AIM:

To create and normalize an employee database, ensuring it adheres to the Second Normal Form (2NF) and Third Normal Form (3NF). Additionally, it includes various SQL queries and operations to retrieve meaningful insights from the database, such as nested queries, aggregate functions, and views.

PROBLEM 1:

Normalize the unnormalized table

EmployeeID	Name	Department	Projects	Salary	DepartmentHead
1	Alice	HR	Payroll, Hiring	50000	Susan
2	Bob	IT	Website, Network	60000	James

SOLUTION:

1NF – Ensure that each cell contains atomic values (no multiple values in a single column). The projects column has multiple values, split into multiple rows.

```
CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY,
Name VARCHAR(100),
DepartmentID INT,
Salary DECIMAL(10, 2),
DepartmentHeadVARCHAR(100)
);
CREATE TABLE Projects (
ProjectID INT PRIMARY KEY AUTO_INCREMENT,
EmployeeID INT,
ProjectNameVARCHAR(100),
```

```
FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
INSERT INTO Employees (EmployeeID, Name, DepartmentID, Salary, DepartmentHead)
VALUES (1, 'Alice', 1, 50000, 'Susan'),
(2, 'Bob', 2, 60000, 'James');
INSERT INTO Projects (EmployeeID, ProjectName)
VALUES (1, 'Payroll'),
(1, 'Hiring'),
 (2, 'Website'),
(2, 'Network');
```

OUTPUT:

Action Output

#	Time	Action	Message
1	19:06:41	create database t1	1 row(s) affected
2	19:06:49	use t1	0 row(s) affected
3	19:07:27	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, Name VARCHAR(100), DepartmentID INT, ...	0 row(s) affected
4	19:07:43	CREATE TABLE Projects (ProjectID INT PRIMARY KEY AUTO_INCREMENT, EmployeeID INT, ProjectNa...	0 row(s) affected
5	19:08:02	INSERT INTO Employees (EmployeeID, Name, DepartmentID, Salary, DepartmentHead) VALUES (1, 'Alice', 1, 5...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
6	19:08:18	INSERT INTO Projects (EmployeeID, ProjectName) VALUES (1, 'Payroll'), (1, 'Hiring'), (2, '...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0

2NF – remove partial dependencies. The DepartmentHead depends only on DepartmentID, not on EmployeeID. Create a new table for departments.

```
CREATE TABLE Departments (
DepartmentID INT PRIMARY KEY,
DepartmentNameVARCHAR(100),
DepartmentHeadVARCHAR(100)
);
INSERT INTO Departments (DepartmentID, DepartmentName, DepartmentHead)
VALUES (1, 'HR', 'Susan'),
(2, 'IT', 'James');
```

OUTPUT:

```
7  19:10:16  CREATE TABLE Departments (  DepartmentID INT PRIMARY KEY,  DepartmentName VARCHAR(100),  Dep...  0 row(s) affected
8  19:10:34  INSERT INTO Departments (DepartmentID, DepartmentName, DepartmentHead) VALUES (1, 'HR', 'Susan'),  ...  2 row(s) affected Records: 2  Duplicates: 0  Warnings: 0
```

3NF – The database is already in 3NF because all non-key attributes are fully dependent on the primary key and there are no transitive dependencies.

PROBLEM 2

Nested Queries: Find the names of employees who work in the same department as an employee named "Alice".

SOLUTION:

```
SELECT NameFROM Employees
WHERE DepartmentID = (
SELECT DepartmentID
FROM Employees
WHERE Name = 'Alice'
);
```

OUTPUT:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name
Alice

PROBLEM 3:

Find employees whose salary is greater than the average salary of their department.

SOLUTION:

```
SELECT Name, SalaryFROM Employees E
WHERE Salary > (
 SELECT AVG(Salary)
 FROM Employees
 WHERE DepartmentID = E.DepartmentID
);
```

SANTHOSH S
3122246002011

OUTPUT:

Name	Salary
Alice	50000.00
Bob	60000.00

PROBLEM 4:

Fetch the names of employees along with their department names.

SOLUTION:

```
SELECT E.Name, D.DepartmentName
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID;
```

OUTPUT:

Name	DepartmentName
Alice	HR
Bob	IT

PROBLEM 5:

Calculate the total salary and the average salary for each department.

SOLUTION:

```
SELECT DepartmentID, SUM(Salary) AS TotalSalary, AVG(Salary) AS AvgSalary
FROM Employees
GROUP BY DepartmentID;
```

OUTPUT:

DepartmentID	TotalSalary	AvgSalary
1	50000.00	50000.000000
2	60000.00	60000.000000

SANTHOSH S
3122246002011

PROBLEM 6:

Find the highest salary in each department.

SOLUTION:

```
SELECT DepartmentID, MAX(Salary) AS HighestSalary
FROM Employees
GROUP BY DepartmentID;
```

OUTPUT:

DepartmentID	HighestSalary
1	50000.00
2	60000.00

PROBLEM 7:

Create a view called HighSalaryEmployees that includes the names and salaries of employees earning more than 60,000.

SOLUTION:

```
CREATE VIEW HighSalaryEmployees AS
SELECT Name, Salary
FROM Employees
WHERE Salary > 60000;
```

OUTPUT:

RESULT:

Thus, database has been structured into three tables Employees, Departments, and Projects, ensuring normalization and elimination of partial and transitive dependencies. SQL queries and operations are retrieved meaningful insights from the database.

SANTHOSH S
3122246002011

EX.NO:2	**QUERY THE DATABASE USING ADVANCED SQL**
09.01.25	

AIM:

To normalize the database to 3NF, ensuring data integrity and execute advanced SQL queries to extract meaningful insights.

PROBLEM 1:

A university database needs to track the following:

Students' details (StudentID, Name, Department).

Courses they enroll in (CourseID, CourseName, Department).

Faculty teaching the courses (FacultyID, FacultyName, Department).

Grades assigned to students for each course.

The data is currently unstructured and has redundancy. Normalize this data to 3NF.

SOLUTION:

#TABLE 1

```
CREATE TABLE Students (
StudentID INT PRIMARY KEY AUTO_INCREMENT,
Name VARCHAR(100),
Department VARCHAR(100)
);
INSERT INTO Students (Name, Department)
VALUES('John', 'CS'),
('Alice', 'IT'),
('Bob', 'CS');
```

#TABLE 2

```
CREATE TABLE Courses (
CourseID INT PRIMARY KEY AUTO_INCREMENT,
CourseNameVARCHAR(100),
```

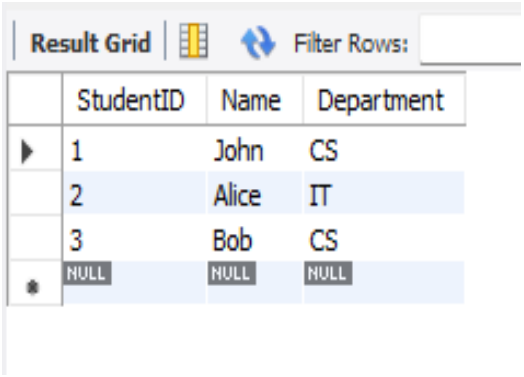
```
Department VARCHAR(100)
);
INSERT INTO Courses (CourseName, Department)
VALUES
('Data Science', 'CS'),
('AI', 'CS'),
('Networks', 'IT');
```

#TABLE 3

```
CREATE TABLE Faculty (
FacultyID INT PRIMARY KEY AUTO_INCREMENT,
FacultyNameVARCHAR(100),
Department VARCHAR(100)
);
INSERT INTO Faculty (FacultyName, Department)
VALUES
('Dr. Smith', 'CS'),
('Dr. Brown', 'CS'),
('Dr. Green', 'IT');
```

OUTPUT:

Students



Courses

Result Grid | Filter Rows:

CourseID	CourseName	Department
1	Data Science	CS
2	AI	CS
3	Networks	IT
NULL	NULL	NULL

Faculty

Result Grid | Filter Rows:

FacultyID	FacultyName	Department
1	Dr. Smith	CS
2	Dr. Brown	CS
3	Dr. Green	IT
NULL	NULL	NULL

PROBLEM 2:

Given a database with the following tables:

Orders table (OrderId, CustomerID, OrderDate, TotalAmount)

Customers table (CustomerID, CustomerName, City)

Find the second highest order amount.

Retrieve the customer name and the total number of orders they placed.

SOLUTION:

```
CREATE TABLE Orders (
OrderID INT PRIMARY KEY AUTO_INCREMENT,
CustomerID INT,
OrderDate DATE,
TotalAmountDECIMAL(10, 2),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
INSERT INTO Orders (OrderId, CustomerID, OrderDate, TotalAmount)
VALUES
(1,1, '2025-03-01', 150.50),
```

(2,2, '2025-03-02', 250.75),

(3,3, '2025-03-03', 300.00),

(4,4, '2025-03-04', 500.25),

(1, 5,'2025-03-05', 800.00);

#TABLE 2

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO_INCREMENT,

CustomerNameVARCHAR(100),

City VARCHAR(100)

);

INSERT INTO Customers (CustomerID, CustomerName, City)

VALUES

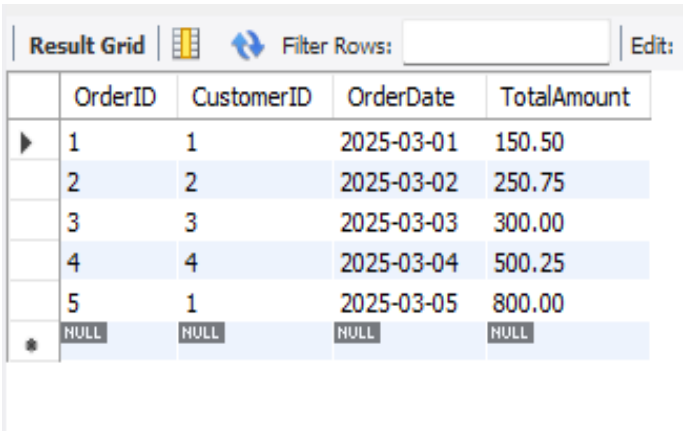
(1,'John Doe', 'New York'),

(2,'Jane Smith', 'Los Angeles'),

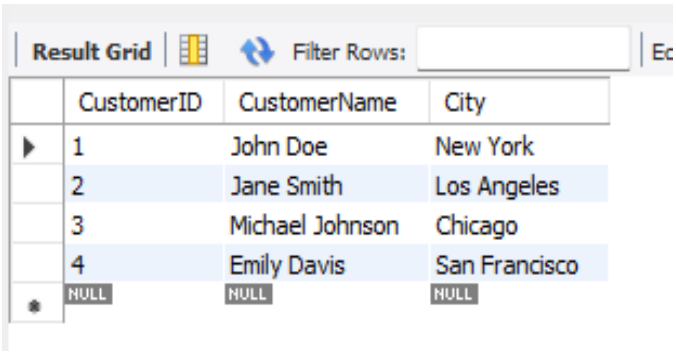
(3,'Michael Johnson', 'Chicago'),

(4,'Emily Davis', 'San Francisco');

OUTPUT:



OrderID	CustomerID	OrderDate	TotalAmount
1	1	2025-03-01	150.50
2	2	2025-03-02	250.75
3	3	2025-03-03	300.00
4	4	2025-03-04	500.25
5	1	2025-03-05	800.00
NULL	NULL	NULL	NULL



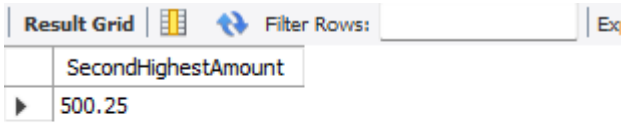
CustomerID	CustomerName	City
1	John Doe	New York
2	Jane Smith	Los Angeles
3	Michael Johnson	Chicago
4	Emily Davis	San Francisco
NULL	NULL	NULL

SANTHOSH S
3122246002011

Find the second highest order amount.

```
SELECT MAX(TotalAmount) AS SecondHighestAmount
FROM Orders
WHERE TotalAmount< (SELECT MAX(TotalAmount) FROM Orders);
```

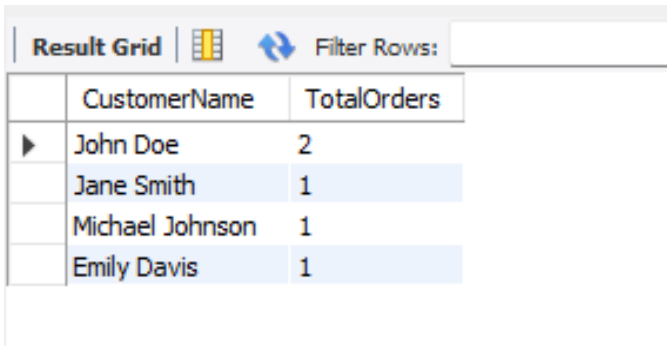
OUTPUT:



Retrieve the customer name and the total number of orders they placed.

```
SELECT Customers.CustomerName, COUNT(Orders.OrderID) AS TotalOrders
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerName;
```

OUTPUT:



PROBLEM 3:

Write a trigger to update the StockLevel in the Products table after an insertion into the Orders table.

Products table(ProductID, ProductName, StockLevel)

Orders table(OrderID, ProductID, Quantity)

SOLUTION:

 # TABLE 1

SANTHOSH S
3122246002011

```
CREATE TABLE Products (
ProductID INT PRIMARY KEY,
ProductName VARCHAR(255),
StockLevel INT
);
INSERT INTO Products (ProductID, ProductName, StockLevel) VALUES
(1, 'Product A', 50),
(2, 'Product B', 40),
(3, 'Product C', 30);
```

 # TABLE 2

```
CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
ProductID INT,
Quantity INT,
FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
INSERT INTO Orders (OrderID, ProductID,  Quantity) VALUES
(101, 1,5),
(102, 2, 3),
(102, 3, 2),
```

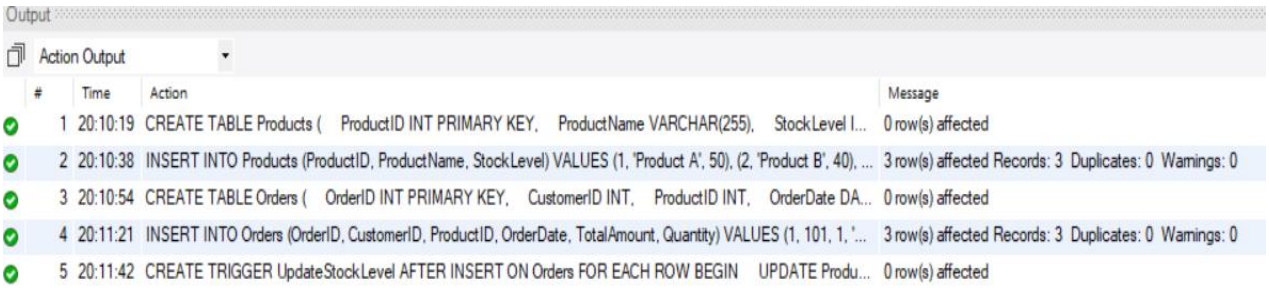
#TRIGGER

```
DELIMITER $$
CREATE TRIGGER UpdateStockLevel
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
   UPDATE Products
   SET StockLevel = StockLevel - NEW.Quantity
   WHERE ProductID = NEW.ProductID;
```

END $$

DELIMITER ;

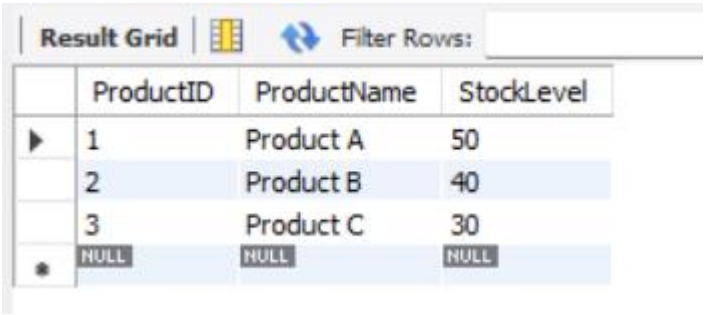
OUTPUT:



Output

Action Output

#	Time	Action	Message
1	20:10:19	CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(255), StockLevel I...	0 row(s) affected
2	20:10:38	INSERT INTO Products (ProductID, ProductName, StockLevel) VALUES (1, 'Product A', 50), (2, 'Product B', 40), ...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
3	20:10:54	CREATE TABLE Orders (OrderID INT PRIMARY KEY, CustomerID INT, ProductID INT, OrderDate DA...	0 row(s) affected
4	20:11:21	INSERT INTO Orders (OrderID, CustomerID, ProductID, OrderDate, TotalAmount, Quantity) VALUES (1, 101, 1, '...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
5	20:11:42	CREATE TRIGGER UpdateStockLevel AFTER INSERT ON Orders FOR EACH ROW BEGIN UPDATE Produ...	0 row(s) affected



Result Grid | Filter Rows:

ProductID	ProductName	StockLevel
1	Product A	50
2	Product B	40
3	Product C	30
NULL	NULL	NULL

PROBLEM 4:

A logistics company wants to identify customers who placed orders totaling more than $1,000 in a single month.

Orders table (OrderId, CustomerID, OrderDate, TotalAmount)

SOLUTION:

CREATE TABLE Orders (

OrderID INT PRIMARY KEY AUTO_INCREMENT,

CustomerID INT,

OrderDate DATE,

TotalAmountDECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)

VALUES

(1, 1, '2025-03-01', 150.50),

(2, 2, '2025-03-02', 250.75),

(3, 3, '2025-03-03', 300.00),

(4, 4, '2025-03-04', 500.25),

(1, 5, '2025-03-05', 800.00);

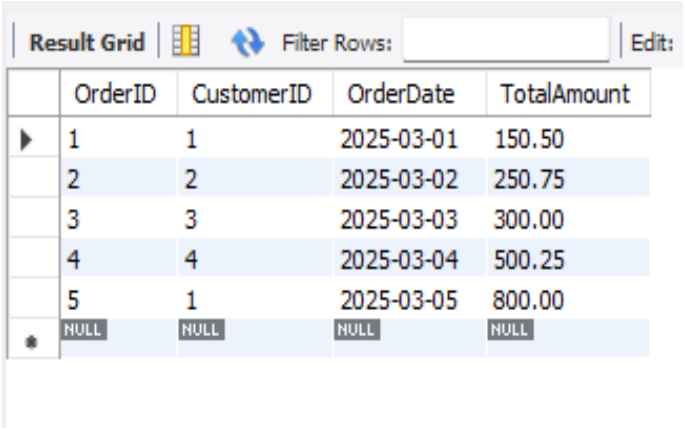
SELECT CustomerID, MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS TotalSpent

FROM Orders

GROUP BY CustomerID, MONTH(OrderDate)

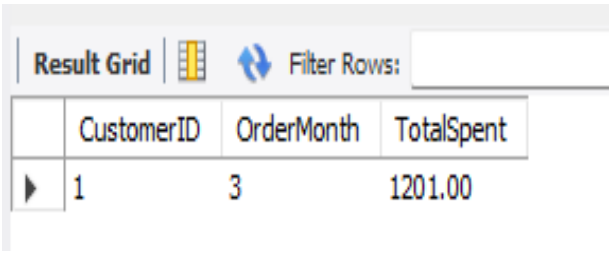
HAVING SUM(TotalAmount) > 1000;

OUTPUT:



Result Grid | Filter Rows: | Edit:

OrderID	CustomerID	OrderDate	TotalAmount
1	1	2025-03-01	150.50
2	2	2025-03-02	250.75
3	3	2025-03-03	300.00
4	4	2025-03-04	500.25
5	1	2025-03-05	800.00
NULL	NULL	NULL	NULL



Result Grid | Filter Rows:

CustomerID	OrderMonth	TotalSpent
1	3	1201.00

PROBLEM 5:

Retrieve all employees in the "IT" department from the MySQL Employees table and display their names and salaries.

Install mysql-connector-python

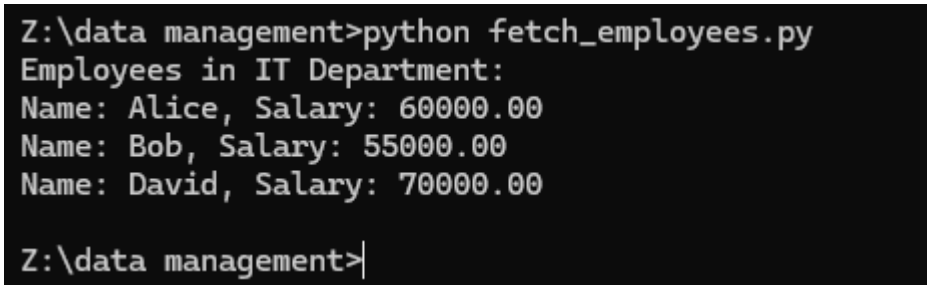
import mysql.connector

Include database connection parameters

```
db_config = {
    "host": "localhost",
    "user": "root",
    "password": "your_password",
    "database": "company",
```

```
    "port": 3306
}
try:
    # Establish the connection
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()

    # SQL query to fetch employees in the "IT" department
    query = "SELECT Name, Salary FROM Employees WHERE Department = 'IT';"
cursor.execute(query)

    # Fetch and display the results
print("Employees in IT Department:")
    for name, salary in cursor.fetchall():
print(f"Name: {name}, Salary: {salary}")

except mysql.connector.Error as e:
print("Database error:", e)
finally:
    # Close the connection
    if cursor:
cursor.close()
    if conn:
conn.close()
```

OUTPUT:



SANTHOSH S
3122246002011

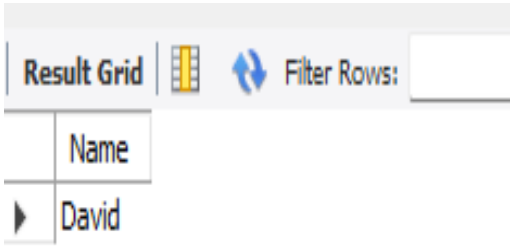
PROBLEM 6:

Retrieve the names of employees in the "IT" department with salaries greater than the average salary of all "IT" department employees.

SOLUTION:

```
SELECT Name

FROM Employees

WHERE Department = 'IT'

AND Salary > (SELECT AVG(Salary) FROM Employees WHERE Department = 'IT');
```

OUTPUT:



RESULT:

The database is structured efficiently, minimizing redundancy. Queries successfully retrieve employees by department, identify high-earning employees, and list customers spending over $1,000 monthly. The HighSalaryEmployees view filters employees earning above $60,000, and Python interacts seamlessly with MySQL.

SANTHOSH S
3122246002011

EX.NO : 3	PL/SQL PROGRAMS
DATE : 24/1/2025	

AIM :

To create a procedure for inserting employee records, and triggers to log employee insertions and updates in the employee_log table.

PROBLEM 1:

Create a procedure that accepts an employee ID and employee name as parameters and inserts a new record into the employees table.

PROGRAM :

1.CREATE A TABLE USING DATABASE

```
CREATE DATABASE EX3;
USE EX3;
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    emp_salary DECIMAL(10, 2)
);
```

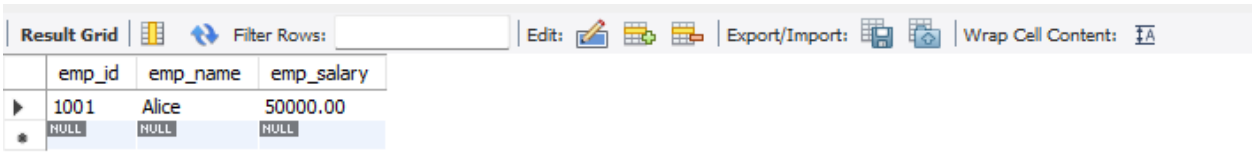
2.#Create the procedure again

```
DELIMITER $$
CREATE PROCEDURE add_employees(
    IN p_emp_id INT,
    IN p_emp_name VARCHAR(50),
    IN p_emp_salary DECIMAL(10, 2)
)
BEGIN
    -- Insert new employee into the employees table
```

SANTHOSH S
3122246002011

```
    INSERT INTO employees (emp_id, emp_name, emp_salary)
    VALUES (p_emp_id, p_emp_name, p_emp_salary);
END$$
DELIMITER ;
3.#Confirm the procedure creation
SHOW PROCEDURE STATUS WHERE Db = 'EX3';
4.#call the procedure
CALL add_employees(1001, 'Alice', 50000.00);
5.#check the employee table
SELECT * FROM employees;
```

OUTPUT :



SANTHOSH S
3122246002011

PROBLEM 2:

Create the employee_log table with columns for log_id, emp_id, action, and timestamp. Write a trigger log_employee_insert that inserts an action log into the employee_log table when a new employee is inserted.

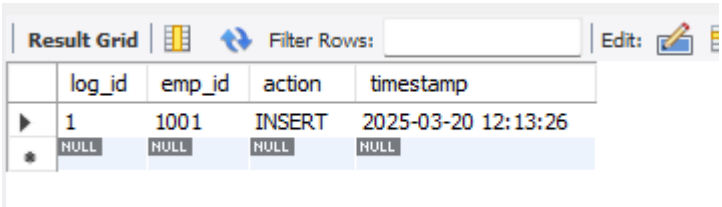
PROGRAM :

```
CREATE DATABASE company1;

USE company1;

CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    emp_salary DECIMAL(10, 2)
);


DROP TABLE IF EXISTS employees;
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    emp_salary DECIMAL(10, 2)
);


CREATE TABLE employee_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_id INT,
    action VARCHAR(50),
    timestamp DATETIME
);
```

SANTHOSH S
3122246002011

```
SHOW PROCESSLIST;
DROP TABLE IF EXISTS employee_log;

SHOW TABLES LIKE 'employee_log';
CREATE TABLE employee_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    emp_id INT,
    action VARCHAR(50),
    timestamp DATETIME
);
DELIMITER $$
CREATE TRIGGER log_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
    INSERT INTO employee_log (emp_id, action, timestamp)
    VALUES (NEW.emp_id, 'INSERT', NOW());
END$$
DELIMITER ;

INSERT INTO employees (emp_id, emp_name, emp_salary)
VALUES (1001, 'John Doe', 45000);
SELECT * FROM employee_log;
```

OUTPUT :



SANTHOSH S
3122246002011

PROBLEM 3 :

Create a trigger log_employee_update that fires after any update to the emp_name column.The trigger should log the employee ID, action (UPDATE), and timestamp in the employee_log table.

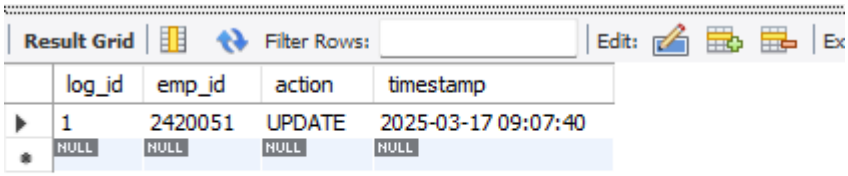
PROGRAM :

```
CREATE DATABASE company7;

USE company7;

#Create the Employee Table

CREATE TABLE employee7 (

   emp_id INT PRIMARY KEY,

   emp_name VARCHAR(50)

);


#Insert Data into Employee Table

INSERT INTO employee7 (emp_id, emp_name) VALUES (2420051, 'Abi');


#Create Employee Log Table

CREATE TABLE employee_log (

   log_id INT AUTO_INCREMENT PRIMARY KEY,

   emp_id INT,

   action VARCHAR(50),

   timestamp DATETIME DEFAULT CURRENT_TIMESTAMP

);

# Create the Trigger log_employee_update

DELIMITER //

CREATE TRIGGER log_employee_update

AFTER UPDATE ON employee7

FOR EACH ROW
```

SANTHOSH S
3122246002011

```
BEGIN
    -- Check if emp_name has changed before logging
    IF OLD.emp_name <> NEW.emp_name THEN
        INSERT INTO employee_log (emp_id, action, timestamp)
        VALUES (NEW.emp_id, 'UPDATE', NOW());
    END IF;
END;
DELIMITER ;
#Test the Trigger (Update Employee Name)
UPDATE employee7 SET emp_name = 'Jane Doe' WHERE emp_id = 2420051;
#Test the Trigger (Update Employee Name)
SELECT * FROM employee_log;
```

OUTPUT :



SELECT * FROM employee7;

OUTPUT :

RESULT:

Successfully implemented the procedure and triggers to log actions when new employees are inserted or their names are updated in the database.

SANTHOSH S
3122246002011

EX.NO:5	**DJANGO WEB APP WITH MONGODB**
DATE:	

AIM:

To develop a **Student Management System** using **Django and MongoDB** to perform CRUD operations. It demonstrates database integration, model creation, and web application functionality.

PREREQUISITES:

Python installed on your system.
Django framework.
MongoDB installed on your system and running.
Basic knowledge of Python and Django.
A text editor or IDE (such as VSCode or PyCharm).

ALGORITHM:

Step 1: **Set Up the Environment**
1.1. Install Django
pip install django
1.2. Install MongoDB Dependencies
Django does not natively support MongoDB. To connect Django with MongoDB, so need a special package called Djongo.
Install Djongo using pip:
pip install djongo
Djongo allows you to use MongoDB as the backend database in Django.

Step 2: Create a Django Project
2.1. Start a New Django Project
Once Django and Djongo are installed, let's create a new Django project:
django-admin startproject student_management
This creates a new directory called student_management containing the basic structure of a Django project.
2.2. Move into the Project Directory
Navigate into the project directory:
cd student_management
2.3. Create a Django App
In Django, an app is a component of the project. Create an app called students to handle student data.
Run the following command to create the app:
python manage.py startapp students
Now, check students directory inside the project.

Step 3: Configure MongoDB in Django

3.1. Update settings.py

Configure Django to use MongoDB. Open the settings.py file in your project directory (student_management/student_management/settings.py).

Find the DATABASES setting and update it to use Djongo with MongoDB.

```
DATABASES = {
'default': {
'ENGINE': 'djongo',
'NAME': 'student_db', # Name of the MongoDB database
'CLIENT': {
'host': 'mongodb://localhost:27017', # MongoDB connection string
}
}
}
```

Note:

ENGINE: Specifies the use of Djongo.

NAME: The name of the MongoDB database.

CLIENT: The connection string to MongoDB (default is localhost:27017).

3.2. Install MongoDB (If Not Installed)

If MongoDB is not installed, install it by following the instructions provided on the MongoDB LMS. After installation, make sure the MongoDB server is running:

```
mongod
```

Step 4: Create a Model for Students

In Django, models define the structure of the database. Create a model for the Student in the students/models.py file.

4.1. Define the Student Model

Open students/models.py and define a Student model like this:

```
from djongo import models
class Student(models.Model):
name = models.CharField(max_length=100)
age = models.IntegerField()
major = models.CharField(max_length=100)
grade = models.CharField(max_length=5)
def __str__(self):
return self.name
```

Note:

name: A string field for the student's name.

age: An integer field for the student's age.

major: A string field for the student's major.

grade: A string field for the student's grade (e.g., A, B, C).

4.2. Make Migrations

Once the model is defined, run the following commands to create the database structure in MongoDB:

```
python manage.py makemigrations
python manage.py migrate
```

This will create the necessary collections in MongoDB for storing student data.

Step 5: Create Views and Templates

Create views to interact with our students' data.

5.1. Create Views for CRUD Operations

In students/views.py, we'll define views to create, list, update, and delete students.

```
from django.shortcuts import render, redirect
from .models import Student
# View to list all students
def student_list(request):
students = Student.objects.all()
return render(request, 'student_list.html', {'students': students})
# View to add a new student
def add_student(request):
if request.method == 'POST':
name = request.POST['name']
age = request.POST['age']
major = request.POST['major']
grade = request.POST['grade']
student = Student(name=name, age=age, major=major, grade=grade)
student.save()
return redirect('student_list')
return render(request, 'add_student.html')
# View to update student details
def update_student(request, id):
student = Student.objects.get(id=id)
if request.method == 'POST':
student.name = request.POST['name']
student.age = request.POST['age']
student.major = request.POST['major']
student.grade = request.POST['grade']
student.save()
return redirect('student_list')
return render(request, 'update_student.html', {'student': student})
# View to delete a student
def delete_student(request, id):
student = Student.objects.get(id=id)
student.delete()
return redirect('student_list')
```

5.2. Create Templates for the Views

Create HTML templates to display these views.

student_list.html: List all students.

```
<!DOCTYPE html>
<html>
<head>
<title>Student List</title>
</head>
<body>
<h1>Student List</h1>
<a href="{% url 'add_student' %}">Add New Student</a>
<table border="1">
```

```
<tr>
<th>Name</th>
<th>Age</th>
<th>Major</th>
<th>Grade</th>
<th>Actions</th>
</tr>
{% for student in students %}
<tr>
<td>{{ student.name }}</td>
<td>{{ student.age }}</td>
<td>{{ student.major }}</td>
<td>{{ student.grade }}</td>
<td>
<a href="{% url 'update_student' student.id %}">Edit</a>
<a href="{% url 'delete_student' student.id %}">Delete</a>
</td>
</tr>
{% endfor %}
</table>
</body>
</html>
```

add_student.html: Add a new student.

```
<!DOCTYPE html>
<html>
<head>
<title>Add Student</title>
</head>
<body>
<h1>Add Student</h1>
<form method="post">
{% csrf_token %}
Name: <input type="text" name="name"><br>
Age: <input type="number" name="age"><br>
Major: <input type="text" name="major"><br>
Grade: <input type="text" name="grade"><br>
<input type="submit" value="Add Student">
</form>
<br>
<a href="{% url 'student_list' %}">Back to Student List</a>
</body>
</html>
```

Step 6: Set Up URLs

In students/urls.py, define the URL routes for the views:

```
from django.urls import path
from . import views
urlpatterns = [
path('', views.student_list, name='student_list'),
path('add/', views.add_student, name='add_student'),
```

path('update/<int:id>/', views.update_student, name='update_student'),
path('delete/<int:id>/', views.delete_student, name='delete_student'),
]
Then, include these URLs in the main urls.py of the project:
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
path('admin/', admin.site.urls),
path('students/', include('students.urls')),
]

Step 7: Run the Project
7.1. Start the Development Server
Run the following command to start the Django development server:
python manage.py runserver
Visit http://127.0.0.1:8000/students/ in Web browser. Check to find the student list, add a new student, edit existing students, and delete students.

OUTPUT:

MongoDB Connection:

```
74  # Database
75  # https://docs.djangoproject.com/en/3.1/ref/settings/#databases
76
77  DATABASES = {
78      'default': {
79          'ENGINE': 'djongo',
80          'NAME': 'student_db',
81          'CLIENT': {
82              'host': 'mongodb://localhost:27017', # MongoDB connection string
83          }
84      }
85  }
86
```

Models.py File:

```
1   from django.db import models
2   class Student(models.Model):
3       name = models.CharField(max_length=100)
4       age = models.IntegerField()
5       major = models.CharField(max_length=100)
6       grade = models.CharField(max_length=5)
7       def __str__(self):
8           return self.name
9
10  # Create your models here.
11
```

SANTHOSH S
3122246002011

Views.py File:

```python
from django.shortcuts import render, get_object_or_404, redirect
from .models import Student

# View to list all students
def student_list(request):
    students = Student.objects.all()
    return render(request, 'student_list.html', {'students': students})


# View to add a new student
def add_student(request):
    if request.method == 'POST':
        name = request.POST['name']
        age = request.POST['age']
        major = request.POST['major']
        grade = request.POST['grade']

        student = Student(name=name, age=age, major=major, grade=grade)
        student.save()
        return redirect('student_list')
    return render(request, 'add_student.html')


# View to update student details
def update_student(request, id):
    student = get_object_or_404(Student, id=id)

    if request.method == "POST":
        student.name = request.POST.get("name")
        student.age = request.POST.get("age")
        student.major = request.POST.get("major")
        student.grade = request.POST.get("grade")
        student.save()
        return redirect("student_list")

    return render(request, "students/update_student.html", {"student": student})


# View to delete a student
def delete_student(request, id):
    student = get_object_or_404(Student, id=id)

    if request.method == "POST":
        student.delete()
        return redirect("student_list")

    return render(request, "students/delete_student.html", {"student": student})
```

Urls.py File:

```python
from django.urls import path
from . import views
urlpatterns = [
    path('', views.student_list, name='student_list'),
    path('add/', views.add_student, name='add_student'),
    path('update/<int:id>/', views.update_student, name='update_student'),
    path('delete/<int:id>/', views.delete_student, name='delete_student'),
]

```

SANTHOSH S
3122246002011

Manage.py File:

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'student_management.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

```

Student.html File:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Student List</title>
</head>
<body>
    <h1>Student List</h1>
    <a href="{% url 'add_student' %}">Add New Student</a>
    <table border="1">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Major</th>
            <th>Grade</th>
            <th>Actions</th>
        </tr>
        {% for student in students %}
        <tr>
            <td>{{ student.name }}</td>
            <td>{{ student.age }}</td>
            <td>{{ student.major }}</td>
            <td>{{ student.grade }}</td>
            <td>
                {% if student.id %}
                    <a href="{% url 'update_student' student.id %}">Edit</a>
                    <a href="{% url 'delete_student' student.id %}">Delete</a>
                {% else %}
                    No Actions Available
                {% endif %}
            </td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```

SANTHOSH S
3122246002011

Add Student html File:

```
<!DOCTYPE html>
<html>
<head>
    <title>Add Student</title>
</head>
<body>
    <h1>Add Student</h1>
    <form method="post">
        {% csrf_token %}
        Name: <input type="text" name="name"><br>
        Age: <input type="number" name="age"><br>
        Major: <input type="text" name="major"><br>
        Grade: <input type="text" name="grade"><br>
        <input type="submit" value="Add Student">
    </form>
    <br>
    <a href="{% url 'student_list' %}">Back to Student List</a>
</body>
</html>
```

Update Student html File:

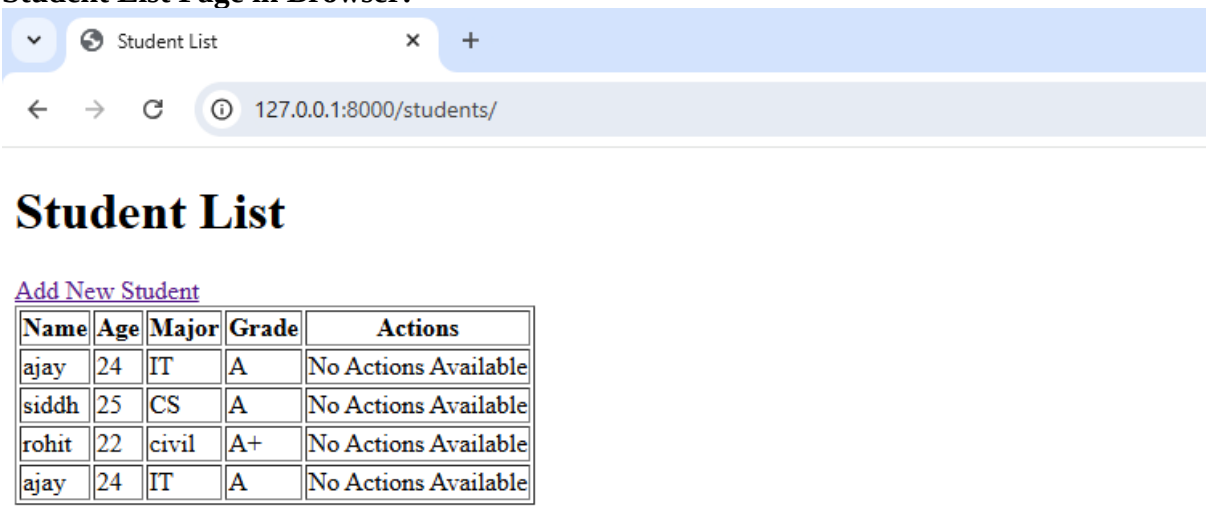
```
<!DOCTYPE html>
<html>
<head>
    <title>Update Student</title>
</head>
<body>
    <h1>Update Student</h1>
    <form method="post">
        {% csrf_token %}
        Name: <input type="text" name="name" value="{{ student.name }}"><br>
        Age: <input type="number" name="age" value="{{ student.age }}"><br>
        Major: <input type="text" name="major" value="{{ student.major }}"><br>
        Grade: <input type="text" name="grade" value="{{ student.grade }}"><br>
        <input type="submit" value="Update Student">
    </form>
    <br>
    <a href="{% url 'student_list' %}">Back to Student List</a>
</body>
</html>

```

Delete Student html File:

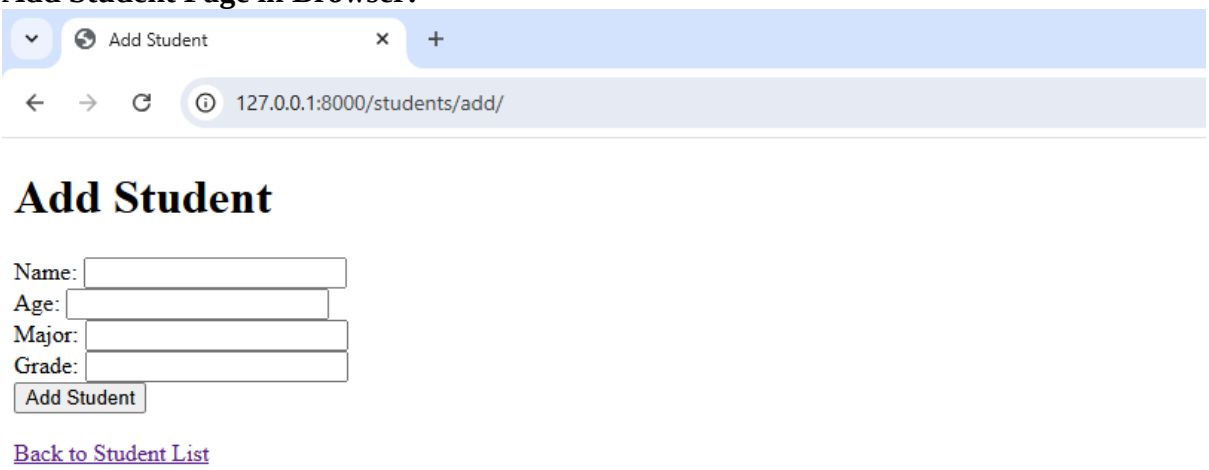
```
<!DOCTYPE html>
<html>
<head>
    <title>Delete Student</title>
</head>
<body>
    <h1>Delete Student</h1>
    <p>Are you sure you want to delete {{ student.name }}?</p>
    <form method="post">
        {% csrf_token %}
        <input type="submit" value="Yes, Delete">
    </form>
    <br>
    <a href="{% url 'student_list' %}">Cancel</a>
</body>
</html>

```

SANTHOSH S
3122246002011

Student List Page in Browser:



Add Student Page in Browser:



RESULT:

The experiment successfully implemented a Student Management System using Django and MongoDB, enabling CRUD operations. The system allows users to add, view, update, and delete student records through a web interface.

NAME: SANTHOSH S
REG NO: 3122246002011

EX. NO: 8	**PYSPARK**
DATE:07/03/25	

AIM:

To write a PySpark program that reads customer transaction data, computes the total number of items purchased by each customer, and writes the result to a CSV file using Spark DataFrame operations.

ALGORITHM:

- Start the SparkSession using SparkSession.builder.
- Define the transaction data as a Python list, where each entry contains a customer name and item name.
- Specify column names for the DataFrame (e.g., "Customer" and "Item").
- Create a DataFrame from the defined data using spark.createDataFrame().
- Group the data by the "Customer" column using groupBy().STEP 6 : Send a POST request to the same API using requests.post() with the created data.
- Aggregate the count of items purchased by each customer using count() function.
- Rename the count column to "Items_Count" using withColumnRenamed().
- Display the result using show().
- Write the result to a CSV file using .write.mode("overwrite").csv() with header option set to True.
- End the program.

NAME: SANTHOSH S
REG NO: 3122246002011

PROGRAM:

```
# Step 1: Import and Create Spark Session
from pyspark.sql import SparkSession
# Create or get Spark session
spark = SparkSession.builder.appName("CustomerItems").getOrCreate()
# Step 2: Load Data into DataFrames
customer_df = spark.read.csv("customers.csv", header=True, inferSchema=True)
items_df = spark.read.csv("items.csv", header=True, inferSchema=True)
# Step 3: Basic Data Exploration
# First few rows of customer data
customer_df.show()
# Summary statistics for items data
items_df.describe().show()
```

```
+--------------+--------------------+--------------+---------------+-----------------+--------------------+--------------------+-------------+--------
+----------+---------+----------------------+-----------+
|customerNumber|        customerName|contactLastName|contactFirstName|            phone|        addressLine1|        addressLine2|         city|   state
|postalCode|  country|salesRepEmployeeNumber|creditLimit|
+--------------+--------------------+--------------+---------------+-----------------+--------------------+--------------------+-------------+--------
+----------+---------+----------------------+-----------+
|           103|   Atelier graphique|       Schmitt|        Carine |       40.32.2555|      54, rue Royale|                NULL|       Nantes|    NULL
|     44000|   France|                  1370|      21000|
|           112|  Signal Gift Stores|          King|           Jean|       7025551838|     8489 Strong St.|                NULL|    Las Vegas|      NV
|     83030|      USA|                  1166|      71800|
|           114|Australian Collec...|      Ferguson|          Peter|     03 9520 4555|   636 St Kilda Road|            Level 3|    Melbourne|Victoria
|      3004|Australia|                  1611|     117300|
|           119|   La Rochelle Gifts|       Labrune|         Janine |       40.67.8555|67, rue des Cinqu...|                NULL|       Nantes|    NULL
|     44000|   France|                  1370|     118200|
|           121|  Baane Mini Imports|    Bergulfsen|          Jonas |       07-98 9555|Erling Skakkes ga...|                NULL|      Stavern|    NULL
|      4110|   Norway|                  1504|      81700|
|           124|Mini Gifts Distri...|        Nelson|          Susan|       4155551450|     5677 Strong St.|                NULL|   San Rafael|      CA
|     97562|      USA|                  1165|     210500|
|           125|  Havel & Zbyszek Co|Piestrzeniewicz|       Zbyszek |    (26) 642-7555|     ul. Filtrowa 68|                NULL|     Warszawa|    NULL
|    01-012|   Poland|                  NULL|          0|
|           128|Blauer See Auto, Co.|        Keitel|         Roland|+49 69 66 90 2555|      Lyonerstr. 34|                NULL|    Frankfurt|    NULL
|     60528|  Germany|                  1504|      59700|
|           129|     Mini Wheels Co.|       Murphy|          Julie|       6505555787|5557 North Pendal...|                NULL|San Francisco|      CA
|     94217|      USA|                  1165|      64600|
|           131|   Land of Toys Inc.|           Lee|          Kwai|       2125557818|897 Long Airport ...|                NULL|          NYC|      NY
|     10022|      USA|                  1323|     114900|
|           141|Euro+ Shopping Ch...|        Freyre|          Diego |   (91) 555 94 44|  C/ Moralzarzal, 86|                NULL|       Madrid|    NULL
|     28034|    Spain|                  1370|     227600|
|           144|Volvo Model Repli...|      Berglund|     Christina |    0921-12 3555|   Berguvsvï¿½gen  8|                NULL|      Luleï¿½|    NULL
|  S-958 22|   Sweden|                  1504|      53100|
|           145|Danish Wholesale ...|      Petersen|         Jytte |       31 12 3555|      Vinbï¿½ltet 34|                NULL|    Kobenhavn|    NULL
|      1734|  Denmark|                  1401|      83400|
|           146|Saveley & Henriot...|       Saveley|          Mary |       78.32.5555|  2, rue du Commerce|                NULL|         Lyon|    NULL
|     69004|   France|                  1337|     123900|
|           148|Dragon Souveniers...|     Natividad|           Eric|     +65 221 7555|         Bronz Sok.|Bronz Apt. 3/6 Te...|    Singapore|    NULL
|   079903|Singapore|                  1621|     103800|
|           151|  Muscle Machine Inc|         Young|           Jeff|       2125557413|   4092 Furth Circle|          Suite 400|          NYC|      NY
|     10022|      USA|                  1286|     138500|
|           157|Diecast Classics ...|        Leong|         Kelvin|       2155551555|    7586 Pompton St.|                NULL|    Allentown|      PA
|     70267|      USA|                  1216|     100600|
|           161|Technics Stores Inc.|     Hashimoto|           Juri|       6505556809|   9408 Furth Circle|                NULL|   Burlingame|      CA
|     94217|      USA|                  1165|      84600|
|           166|    Handji Gifts& Co|      Victorino|          Wendy|     +65 224 1555|106 Linden Road S...|            2nd Floor|    Singapore|    NULL
|   069045|Singapore|                  1612|      97900|
|           167|        Herkku Gifts|        Oeztan|          Veysel|    +47 2267 3215|     Brehmen St. 121|     PR 334 Sentrum|       Bergen|    NULL
|    N 5804| Norway  |                  1504|      96800|
+--------------+--------------------+--------------+---------------+-----------------+--------------------+--------------------+-------------+--------
+----------+---------+----------------------+-----------+
only showing top 20 rows
```

```
+-------+------------------+----------+--------------------+
|summary|     Member_number|      Date|     itemDescription|
+-------+------------------+----------+--------------------+
|  count|             38765|     38765|               38765|
|   mean|  3003.64186766413|      NULL|                NULL|
| stddev|1153.6110310565457|      NULL|                NULL|
|    min|              1000|01-01-2014|Instant food prod...|
|    max|              5000|31-10-2015|            zwieback|
+-------+------------------+----------+--------------------+
```

NAME: SANTHOSH S
REG NO: 3122246002011

# Step 4: Join customer_df and items_df
joined_df = customer_df.join(items_df, customer_df["customerNumber"] ==
items_df["Member_number"])
joined_df.show()

```
+--------------+------------+---------------+----------------+-----+------------+------------+----+-----+----------+-------+--
--------------------+-----------+-------------+----+---------------+
|customerNumber|customerName|contactLastName|contactFirstName|phone|addressLine1|addressLine2|city|state|postalCode|country|sa
lesRepEmployeeNumber|creditLimit|Member_number|Date|itemDescription|
+--------------+------------+---------------+----------------+-----+------------+------------+----+-----+----------+-------+--
--------------------+-----------+-------------+----+---------------+
+--------------+------------+---------------+----------------+-----+------------+------------+----+-----+----------+-------+--
--------------------+-----------+-------------+----+---------------+
```

# Step 5: Add Dummy Price Column (Assuming each item costs 10.0)
print("Customer DF Columns:", customer_df.columns)
print("Items DF Columns:", items_df.columns)

```
Customer DF Columns: ['customerNumber', 'customerName', 'contactLastName', 'contactFirstName', 'phone', 'addressLine1', 'addre
ssLine2', 'city', 'state', 'postalCode', 'country', 'salesRepEmployeeNumber', 'creditLimit']
Items DF Columns: ['Member_number', 'Date', 'itemDescription']
```

# Step 6: Filter Privileged vs. Unprivileged Customers
privileged_df = customer_df.filter(customer_df['creditLimit'] > 100)
unprivileged_df = customer_df.filter(customer_df['creditLimit'] <= 100)

privileged_df.show()
unprivileged_df.show()

```
+--------------+--------------------+-------------+---------------+---------------+--------------------+------------+----------+--------+----------
+------------+----------------------+-----------+
|customerNumber|        customerName|contactLastName|contactFirstName|          phone|        addressLine1|addressLine2|      city|   state|postalCode
|     country|salesRepEmployeeNumber|creditLimit|
+--------------+--------------------+-------------+---------------+---------------+--------------------+------------+----------+--------+----------
+------------+----------------------+-----------+
|           125|  Havel & Zbyszek Co|Piestrzeniewicz|       Zbyszek |   (26) 642-7555|     ul. Filtrowa 68|        NULL|   Warszawa|    NULL|    01-012
|      Poland|                  NULL|          0|
|           168|American Souvenir...|        Franco|          Keith|     2035557845|   149 Spinnaker Dr.|   Suite 101|  New Haven|      CT|     97823
|         USA|                  1286|          0|
|           169|   Porto Imports Co.|      de Castro|         Isabel |    (1) 356-5555|Estrada da saï¿½d...|        NULL|    Lisboa|    NULL|      1756
|    Portugal|                  NULL|          0|
|           206|Asian Shopping Ne...|         Walker|         Brydey|  +612 9411 1555|  Suntec Tower Three|   8 Temasek|  Singapore|    NULL|    038988
|   Singapore|                  NULL|          0|
|           223|   Natï¿½rlich Autos|         Kloss|         Horst |     0372-555188|  Taucherstraï¿½e 10|        NULL|  Cunewalde|    NULL|     01307
|     Germany|                  NULL|          0|
|           237|       ANG Resellers|        Camino|     Alejandra |   (91) 745 6555|       Gran Vï¿½a, 1|        NULL|     Madrid|    NULL|     28001
|       Spain|                  NULL|          0|
|           247|Messner Shopping ...|       Messner|        Renate |     069-0555984|        Magazinweg 7|        NULL|  Frankfurt|    NULL|     60528
|     Germany|                  NULL|          0|
|           273|   Franken Gifts, Co|       Franken|         Peter |     089-0877555|   Berliner Platz 43|        NULL|  Mï¿½nchen|    NULL|     80805
|     Germany|                  NULL|          0|
|           293|   BG&E Collectables|       Harrison|             Ed|+41 26 425 50 01|Rte des Arsenaux 41 |        NULL|   Fribourg|    NULL|      1700
| Switzerland|                  NULL|          0|
|           303|    Schuyler Imports|       Schuyler|        Bradley| +31 20 491 9555|    Kingsfordweg 151|        NULL|  Amsterdam|    NULL|   1043 GR
| Netherlands|                  NULL|          0|
|           307|    Der Hund Imports|       Andersen|            Mel|     030-0074555|       Obere Str. 57|        NULL|     Berlin|    NULL|     12209
|     Germany|                  NULL|          0|
|           335|Cramer Spezialitï...|         Cramer|         Philip |      0555-09555|       Maubelstr. 90|        NULL|Brandenburg|    NULL|     14776
|     Germany|                  NULL|          0|
|           348|Asian Treasures, ...|       McKenna|       Patricia |        2967 555|    8 Johnstown Road|        NULL|       Cork|Co. Cork|      NULL
|     Ireland|                  NULL|          0|
|           356|SAR Distributors, Co|         Kuger|          Armand| +27 21 550 3555|1250 Pretorius St...|        NULL|   Hatfield|Pretoria|      0028
|South Africa|                  NULL|          0|
|           361|     Kommission Auto|       Josephs|          Karin|     0251-555259|       Luisenstr. 48|        NULL|  Mï¿½nster|    NULL|     44087
|     Germany|                  NULL|          0|
|           369|Lisboa Souveniers...|     Rodriguez|          Lino |    (1) 354-2555|Jardim das rosas ...|        NULL|    Lisboa|    NULL|      1675
|    Portugal|                  NULL|          0|
|           376|Precious Collecta...|           Urs|          Braun|     0452-076555|        Hauptstr. 29|        NULL|      Bern|    NULL|      3012
| Switzerland|                  1702|          0|
|           409|Stuttgart Collect...|      Mï¿½ller|          Rita |     0711-555361|   Adenauerallee 900|        NULL|  Stuttgart|    NULL|     70563
|     Germany|                  NULL|          0|
|           443|Feuer Online Stor...|         Feuer|      Alexander |     0342-555176|         Heerstr. 22|        NULL|    Leipzig|    NULL|     04179
|     Germany|                  NULL|          0|
|           459|    Warburg Exchange|       Ottlieb|          Sven |     0241-039123|       Walserweg 21|        NULL|    Aachen|    NULL|     52066
|     Germany|                  NULL|          0|
+--------------+--------------------+-------------+---------------+---------------+--------------------+------------+----------+--------+----------
+------------+----------------------+-----------+
only showing top 20 rows
```

NAME: SANTHOSH S
REG NO: 3122246002011

```
# Step 7: Count Items Bought by Each Customer
from pyspark.sql import functions as F

customer_items_count = items_df.groupBy("Member_number").agg(
    F.count("itemDescription").alias("items_bought")
)
customer_items_count.show()
```

```
+-------------+------------+
|Member_number|items_bought|
+-------------+------------+
|         1959|          19|
|         1088|           9|
|         4818|           8|
|         2659|          16|
|         1580|          16|
|         2122|          10|
|         3997|          10|
|         3175|          14|
|         4519|           5|
|         2142|          19|
|         3794|          11|
|         2366|          13|
|         3749|           9|
|         4935|           8|
|         4101|           8|
|         1342|          13|
|         3918|          13|
|         1591|          10|
|         1238|           2|
|         1829|           9|
+-------------+------------+
only showing top 20 rows
```

```
items_df = items_df.withColumn("Price", F.lit(10.0))

# Step 8: Calculate Total Spending per Customer
total_spent_df = items_df.groupBy("Member_number").agg(
    F.sum("Price").alias("total_spent")
)
total_spent_df.show()
```

NAME: SANTHOSH S
REG NO: 3122246002011

```
+-------------+-----------+
|Member_number|total_spent|
+-------------+-----------+
|         1959|      190.0|
|         1088|       90.0|
|         4818|       80.0|
|         2659|      160.0|
|         1580|      160.0|
|         2122|      100.0|
|         3997|      100.0|
|         3175|      140.0|
|         4519|       50.0|
|         2142|      190.0|
|         3794|      110.0|
|         2366|      130.0|
|         3749|       90.0|
|         4935|       80.0|
|         4101|       80.0|
|         1342|      130.0|
|         3918|      130.0|
|         1591|      100.0|
|         1238|       20.0|
|         1829|       90.0|
+-------------+-----------+
only showing top 20 rows
```

Step 9: Map/Reduce Using RDD - Items Bought per Customer

```
from pyspark.sql import SparkSession
# Create a Spark session
spark = SparkSession.builder.appName("CustomerItemsAnalysis").getOrCreate()
# Load the items.csv file
df = spark.read.csv("items.csv", header=True, inferSchema=True)
# Show sample data to verify
df.show(5)
```

```
+-------------+----------+----------------+
|Member_number|      Date| itemDescription|
+-------------+----------+----------------+
|         1808|21-07-2015|  tropical fruit|
|         2552|05-01-2015|      whole milk|
|         2300|19-09-2015|       pip fruit|
|         1187|12-12-2015|other vegetables|
|         3037|01-02-2015|      whole milk|
+-------------+----------+----------------+
only showing top 5 rows
```

NAME: SANTHOSH S
REG NO: 3122246002011

```
import os
from pyspark.sql import SparkSession

# Step 1: Set environment variables
os.environ["HADOOP_HOME"] = "C:\\hadoop\\winutils-master\\hadoop-3.2.2"
os.environ["PATH"] += os.pathsep + os.path.join(os.environ["HADOOP_HOME"], "bin")
os.environ["spark.hadoop.fs.file.impl"] = "org.apache.hadoop.fs.LocalFileSystem"

# Step 2: Start Spark session
spark = SparkSession.builder.appName("CustomerItemsAnalysis").getOrCreate()

# Step 3: Create a dummy DataFrame (you can replace this with your actual logic)
data = [("Alice", 5), ("Bob", 3), ("Charlie", 8)]
columns = ["Customer", "ItemsCount"]
customer_items_count = spark.createDataFrame(data, columns)

# Step 4: Save to CSV
customer_items_count.write.mode("overwrite").csv("output/customer_items_count.csv",
header=True)
```

```
data = [("Alice", 5), ("Bob", 3), ("Charlie", 8)]
columns = ["Customer", "ItemsCount"]
```

```
+--------+-----------+
|Customer|ItemsCount|
+--------+-----------+
|   Alice|          5|
|     Bob|          3|
| Charlie|          8|
+--------+-----------+
```

RESULT:

Thus, the PySpark program has been successfully completed.

NAME: SANTHOSH S
REG NO: 3122246002011

NAME:SANTHOSH S
REG NO: 3122246002011

EX NO: 6	**HADOOP DISTRIBUTED FILE SYSTEM (HDFS) COMMANDS**
DATE:21/02/25	

AIM:

To understand and execute basic HDFS commands to manage files and directories in Hadoop Distributed File System (HDFS).

ALGORITHM:

1. **Start Hadoop Daemons**:
    - Use commands like start-dfs.sh and start-yarn.sh to start Hadoop services.
2. **Create Directories in HDFS** using hdfs dfs -mkdir.
3. **Upload Files from Local to HDFS** using hdfs dfs -put.
4. **View Files and Directories** using hdfs dfs -ls.
5. **Read File Contents** using hdfs dfs -cat.
6. **Download Files from HDFS to Local** using hdfs dfs -get.
7. **Remove Files or Directories from HDFS** using hdfs dfs -rm and -rmdir.
8. **Check HDFS File System Usage** using hdfs dfs -du, -count, or -df.
9. **Stop Hadoop Daemons** after usage using stop-dfs.sh and stop-yarn.sh.

PROGRAM:

Step 1: Start Hadoop Services

```
start-dfs.sh        # Starts HDFS daemons: NameNode and DataNode
start-yarn.sh       # Starts YARN daemons: ResourceManager and NodeManager
```

Step 2: Create Directory in HDFS

```
hdfs dfs -mkdir /student      # Creates a directory named 'student' in the root of HDFS
```

Step 3: Upload File to HDFS

```
hdfs dfs -put sample.txt /student     # Uploads the local file 'sample.txt' to '/student' directory in HDFS
```

Step 4: List HDFS Directory

```
hdfs dfs -ls /student         # Lists all files and folders inside '/student' in HDFS
```

Step 5: View File Contents

```
hdfs dfs -cat /student/sample.txt     # Displays the contents of 'sample.txt' from HDFS
```

NAME:SANTHOSH S
REG NO: 3122246002011

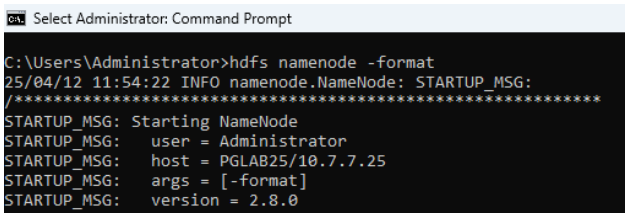
Step 6: Copy File to Local
hdfs dfs -get /student/sample.txt ./ # Downloads 'sample.txt' from HDFS to current local directory

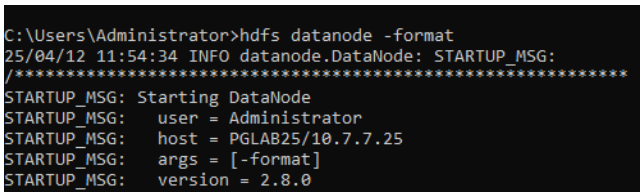
Step 7: Delete File from HDFS
hdfs dfs -rm /student/sample.txt # Deletes 'sample.txt' from '/student' in HDFS

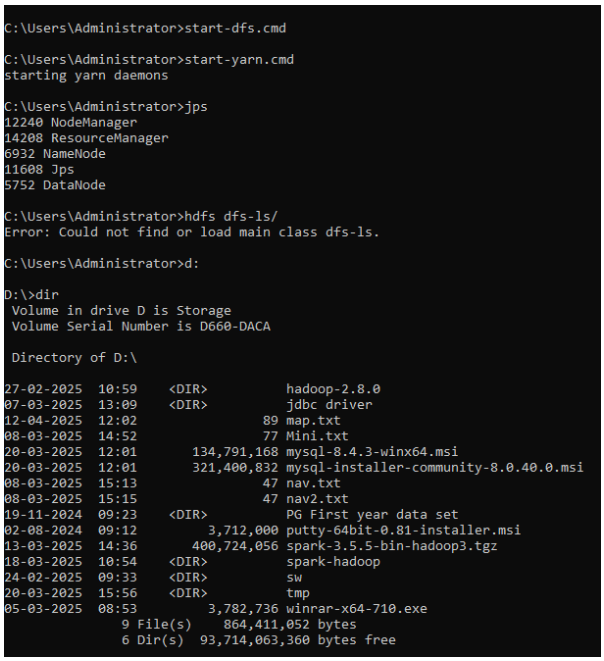
Step 8: Delete Directory
hdfs dfs -rmdir /student # Removes the empty '/student' directory from HDFS

Step 9: Check HDFS Report
hdfs dfsadmin -report # Displays the HDFS cluster report including storage, health, and nodes info

Step 10: Stop Hadoop Services
stop-dfs.sh # Stops HDFS daemons: NameNode and DataNode
stop-yarn.sh # Stops YARN daemons: ResourceManager and NodeManager







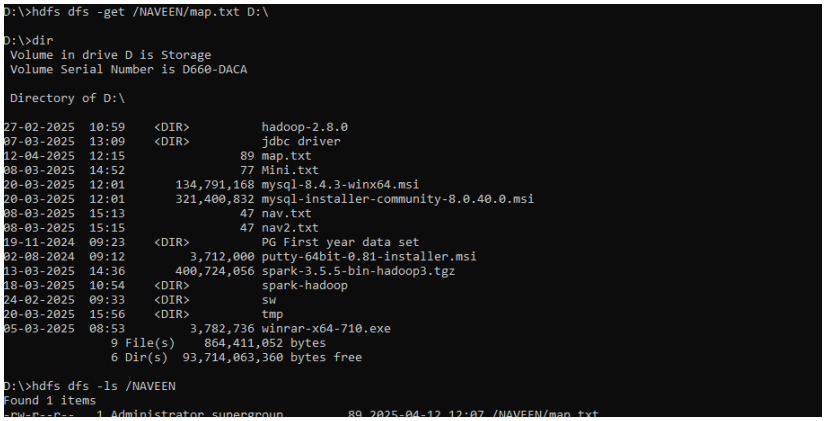
NAME:SANTHOSH S
REG NO: 3122246002011

```
D:\>hdfs dfs -mkdir /MAPREDUDE

D:\>hdfs dfs -put map.txt /MAPREDUCE

D:\>DIR
 Volume in drive D is Storage
 Volume Serial Number is D660-DACA

 Directory of D:\

27-02-2025  10:59    <DIR>          hadoop-2.8.0
07-03-2025  13:09    <DIR>          jdbc driver
12-04-2025  12:02                89 map.txt
08-03-2025  14:52                77 Mini.txt
20-03-2025  12:01       134,791,168 mysql-8.4.3-winx64.msi
20-03-2025  12:01       321,400,832 mysql-installer-community-8.0.40.0.msi
08-03-2025  15:13                47 nav.txt
08-03-2025  15:15                47 nav2.txt
19-11-2024  09:23    <DIR>          PG First year data set
02-08-2024  09:12         3,712,000 putty-64bit-0.81-installer.msi
13-03-2025  14:36       400,724,056 spark-3.5.5-bin-hadoop3.tgz
18-03-2025  10:54    <DIR>          spark-hadoop
24-02-2025  09:33    <DIR>          sw
20-03-2025  15:56    <DIR>          tmp
05-03-2025  08:53         3,782,736 winrar-x64-710.exe
               9 File(s)    864,411,052 bytes
               6 Dir(s)  93,714,063,360 bytes free
```

```
D:\>hdfs dfs -mkdir /MAPREDUDE

D:\>hdfs dfs -put map.txt /MAPREDUCE

D:\>DIR
 Volume in drive D is Storage
 Volume Serial Number is D660-DACA

 Directory of D:\

27-02-2025  10:59    <DIR>          hadoop-2.8.0
07-03-2025  13:09    <DIR>          jdbc driver
12-04-2025  12:02                89 map.txt
08-03-2025  14:52                77 Mini.txt
20-03-2025  12:01       134,791,168 mysql-8.4.3-winx64.msi
20-03-2025  12:01       321,400,832 mysql-installer-community-8.0.40.0.msi
08-03-2025  15:13                47 nav.txt
08-03-2025  15:15                47 nav2.txt
19-11-2024  09:23    <DIR>          PG First year data set
02-08-2024  09:12         3,712,000 putty-64bit-0.81-installer.msi
13-03-2025  14:36       400,724,056 spark-3.5.5-bin-hadoop3.tgz
18-03-2025  10:54    <DIR>          spark-hadoop
24-02-2025  09:33    <DIR>          sw
20-03-2025  15:56    <DIR>          tmp
05-03-2025  08:53         3,782,736 winrar-x64-710.exe
               9 File(s)    864,411,052 bytes
               6 Dir(s)  93,714,063,360 bytes free
```

```
D:\>hdfs dfs -get /NAVEEN/map.txt D:\

D:\>dir
 Volume in drive D is Storage
 Volume Serial Number is D660-DACA

 Directory of D:\

27-02-2025  10:59    <DIR>          hadoop-2.8.0
07-03-2025  13:09    <DIR>          jdbc driver
12-04-2025  12:15                89 map.txt
08-03-2025  14:52                77 Mini.txt
20-03-2025  12:01       134,791,168 mysql-8.4.3-winx64.msi
20-03-2025  12:01       321,400,832 mysql-installer-community-8.0.40.0.msi
08-03-2025  15:13                47 nav.txt
08-03-2025  15:15                47 nav2.txt
19-11-2024  09:23    <DIR>          PG First year data set
02-08-2024  09:12         3,712,000 putty-64bit-0.81-installer.msi
13-03-2025  14:36       400,724,056 spark-3.5.5-bin-hadoop3.tgz
18-03-2025  10:54    <DIR>          spark-hadoop
24-02-2025  09:33    <DIR>          sw
20-03-2025  15:56    <DIR>          tmp
05-03-2025  08:53         3,782,736 winrar-x64-710.exe
               9 File(s)    864,411,052 bytes
               6 Dir(s)  93,714,063,360 bytes free

D:\>hdfs dfs -ls /NAVEEN
Found 1 items
-rw-r--r--   1 Administrator supergroup         89 2025-04-12 12:07 /NAVEEN/map.txt
```

NAME:SANTHOSH S
REG NO: 3122246002011



RESULT:

Thus, the HDFS Commands has been executed successfully

NAME: SANTHOSH S
REG NO: 3122246002011

EX NO: 7	**HADOOP MAPREDUCE**
DATE:08/03/25	

AIM:

To write a simple Hadoop MapReduce program to count the number of occurrences of each word in a text file (e.g., map.txt) using basic MapReduce principles.

ALGORITHM:

1. Prepare the Input File (map.txt):
    - Contains a list of words or sentences (e.g., "apple banana apple orange").
2. Mapper Function:
    - Read each line.
    - Split the line into individual words.
    - Emit each word with the value 1.
3. Reducer Function:
    - Receive each word as key with list of 1s as values.
    - Sum all values to get total count.
    - Emit word and its total count.
4. Driver Program:
    - Set up job configuration.
    - Define input/output paths.
    - Set Mapper, Reducer, and data types.
5. Run the program and view output in the Hadoop output directory.

PROGRAM:

SAMPLE TEXT(map.txt):

apple banana apple

orange banana apple

NAME: SANTHOSH S
REG NO: 3122246002011

MAPPER CLASS (WordMapper.java):

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;
public class WordMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String[] words = value.toString().split(" ");
      for (String w : words) {
         word.set(w);
         context.write(word, one);
      }
   }
}
```

REDUCER CLASS (WordReducer.java):

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;
public class WordReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
   public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      context.write(key, new IntWritable(sum));
   }
}
```

DRIVER CLASS (WordCountDriver.java):

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCountDriver {
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "Word Count");
      job.setJarByClass(WordCountDriver.class);
      job.setMapperClass(WordMapper.class);
```

NAME: SANTHOSH S
REG NO: 3122246002011

```
        job.setReducerClass(WordReducer.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);
    }
}
```

EXECUTION COMMANDS:

javac -classpath `hadoop classpath` -d . WordMapper.java WordReducer.java WordCountDriver.java

Create JAR file
jar -cvf wordcount.jar *.class

Run the MapReduce program
hadoop jar wordcount.jar WordCountDriver /input/map.txt /output

```
D:\>hadoop jar C:\hadoop\hadoop-2.8.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.8.0.jar wordcount /NAVEEN/map.txt /part_0
25/04/12 12:19:09 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
25/04/12 12:19:09 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
25/04/12 12:19:09 INFO input.FileInputFormat: Total input files to process : 1
25/04/12 12:19:09 INFO mapreduce.JobSubmitter: number of splits:1
```

```
        HDFS: Number of write operations=4
    Map-Reduce Framework
            Map input records=1
            Map output records=15
            Map output bytes=149
            Map output materialized bytes=148
            Input split bytes=101
            Combine input records=15
            Combine output records=12
            Reduce input groups=12
            Reduce shuffle bytes=148
            Reduce input records=12
            Reduce output records=12
            Spilled Records=24
            Shuffled Maps =1
            Failed Shuffles=0
            Merged Map outputs=1
            GC time elapsed (ms)=7
            Total committed heap usage (bytes)=713031680
    Shuffle Errors
            BAD_ID=0
            CONNECTION=0
            IO_ERROR=0
            WRONG_LENGTH=0
            WRONG_MAP=0
            WRONG_REDUCE=0
    File Input Format Counters
            Bytes Read=89
    File Output Format Counters
            Bytes Written=94
```

NAME: SANTHOSH S
REG NO: 3122246002011

OUTPUT:

/output/part-r-00000):

apple 3
banana 2
orange 1

RESULT:

Thus, the MapReduce program has been executed successfully.