10.**Create the tables CUSTOMER (C_ID, Name, Address, City, Mobile_No) and ORDER (C_ID, P_ID, P_Name, P_COST)**

Ans:
```
CREATE TABLE CUSTOMER (
   C_ID INT PRIMARY KEY,
   Name VARCHAR(100),
   Address VARCHAR(200),
   City VARCHAR(50),
   Mobile_No VARCHAR(20)
);

CREATE TABLE ORDER (
   C_ID INT,
   P_ID INT,
   P_Name VARCHAR(100),
   P_COST DECIMAL(10, 2),
   FOREIGN KEY (C_ID) REFERENCES CUSTOMER(C_ID)
);
```

a.List the names and addresses of all the customers who have ordered products of costmore than 500.

ANS:
```
SELECT c.Name, c.Address
FROM CUSTOMER c
JOIN ORDER o ON c.C_ID = o.C_ID
WHERE o.P_COST > 500;
```

b) List the names of all the products ordered whose cost is 1,000 or more.

ANS:
```
SELECT P_Name
FROM `ORDER`
WHERE P_COST >= 1000;
```

c) List the product names which are ordered by customers of "City = Delhi".

ANS:
```
SELECT o.P_Name
FROM CUSTOMER c
JOIN `ORDER` o ON c.C_ID = o.C_ID
WHERE c.City = 'Delhi';
```

d) Add column "Email_id" in the CUSTOMER table.

ANS:
```
ALTER TABLE CUSTOMER
ADD Email_id VARCHAR(100);
```

e) Demonstrate the user defined function for the above tables.

ANS:
```
DELIMITER //

CREATE FUNCTION CalculateTotalCost(CustID INT) RETURNS DECIMAL(10, 2)
BEGIN
```

```
    DECLARE TotalCost DECIMAL(10, 2);

    SELECT SUM(P_COST) INTO TotalCost
    FROM `ORDER`
    WHERE C_ID = CustID;

    RETURN TotalCost;
END //

DELIMITER ;
```

to execute this function use this query

```
SELECT C_ID, CalculateTotalCost(C_ID) AS TotalCost
FROM CUSTOMER;
```

11.Create the tables SALESMAN (Salesman_id, Name, City, Commission), CUSTOMER (Customer_id, Cust_Name, City, Grade,Salesman_id), ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

ANS:

```
CREATE TABLE SALESMAN (
    Salesman_id INT PRIMARY KEY,
    Name VARCHAR(50),
    City VARCHAR(50),
    Commission DECIMAL(8,2)
);

CREATE TABLE CUSTOMER (
    Customer_id INT PRIMARY KEY,
    Cust_Name VARCHAR(50),
    City VARCHAR(50),
    Grade CHAR(1),
    Salesman_id INT,
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id)
);

CREATE TABLE ORDERS (
    Ord_No INT PRIMARY KEY,
    Purchase_Amt DECIMAL(10,2),
    Ord_Date DATE,
    Customer_id INT,
    Salesman_id INT,
    FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id),
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id)
);
```

a) Find the name and numbers of all salesmen who had more than one customer.

ANS:

```
SELECT s.Name, COUNT(c.Customer_id) AS Num_Customers
FROM SALESMAN s
INNER JOIN CUSTOMER c ON s.Salesman_id = c.Salesman_id
GROUP BY s.Salesman_id
HAVING COUNT(c.Customer_id) > 1;
```

b) List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

AnS:

```
SELECT s.Name AS Salesman_Name, s.City AS Salesman_City, 'Has Customers' AS Status
FROM SALESMAN s
INNER JOIN CUSTOMER c ON s.Salesman_id = c.Salesman_id
GROUP BY s.Salesman_id

UNION

SELECT s.Name AS Salesman_Name, s.City AS Salesman_City, 'No Customers' AS Status
FROM SALESMAN s
LEFT JOIN CUSTOMER c ON s.Salesman_id = c.Salesman_id
WHERE c.Customer_id IS NULL;
```

c) Create a view that finds the salesman who has the customer with the highest order of a day.

ANS:

```
CREATE VIEW Salesman_MaxOrder AS
SELECT o.Salesman_id, MAX(o.Purchase_Amt) AS Max_Order
FROM ORDERS o
INNER JOIN (
   SELECT MAX(Ord_Date) AS Max_Date
   FROM ORDERS
   GROUP BY DATE(Ord_Date)
) t ON o.Ord_Date = t.Max_Date
GROUP BY o.Salesman_id;
```

d) Perform the DELETE operation by removing salesman with id 1000. All his orders

ANS:

```
DELETE FROM ORDERS WHERE Salesman_id = 1000;
DELETE FROM SALESMAN WHERE Salesman_id = 1000;
```

e) Demonstrate the Triggers for the above table.

ANS:

----------INSERT-----

```
CREATE TRIGGER update_total_purchase
AFTER INSERT ON ORDERS
FOR EACH ROW
BEGIN
   UPDATE CUSTOMER
   SET Total_Purchase = Total_Purchase + NEW.Purchase_Amt
   WHERE Customer_id = NEW.Customer_id;
```

```
END;

----------UPDATE-----------

CREATE TRIGGER update_total_purchase
AFTER UPDATE ON ORDERS
FOR EACH ROW
BEGIN
   DECLARE purchase_diff DECIMAL(10,2);
   SET purchase_diff = NEW.Purchase_Amt - OLD.Purchase_Amt;

   UPDATE CUSTOMER
   SET Total_Purchase = Total_Purchase + purchase_diff
   WHERE Customer_id = NEW.Customer_id;
END;

----------DELETE-----------

CREATE TRIGGER delete_customer_orders
AFTER DELETE ON CUSTOMER
FOR EACH ROW
BEGIN
   DELETE FROM ORDERS
   WHERE Customer_id = OLD.Customer_id;
END;
```

12 . Develop a simple GUI based Inventory Management for a EMart Grocery Shop database application and incorporate all the Database features.

ANS:

```
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.*;

public class InventoryManagementApp extends Application {

   private static final String DB_URL = "jdbc:mysql://localhost:3306/inventory";
   private static final String DB_USERNAME = "your_username";
   private static final String DB_PASSWORD = "your_password";

   private ObservableList<Product> products;

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

```
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("EMart Inventory Management");

        // Create UI components
        TableView<Product> tableView = new TableView<>();
        Label nameLabel = new Label("Name:");
        TextField nameField = new TextField();
        Label quantityLabel = new Label("Quantity:");
        TextField quantityField = new TextField();
        Button addButton = new Button("Add Product");
        Button refreshButton = new Button("Refresh");

        // Configure table columns
        TableColumn<Product, String> nameColumn = new TableColumn<>("Name");
        nameColumn.setCellValueFactory(cellData -> cellData.getValue().nameProperty());

        TableColumn<Product, Integer> quantityColumn = new TableColumn<>("Quantity");
        quantityColumn.setCellValueFactory(cellData ->
cellData.getValue().quantityProperty().asObject());

        tableView.getColumns().addAll(nameColumn, quantityColumn);

        // Configure button actions
        addButton.setOnAction(event -> {
            String name = nameField.getText();
            int quantity = Integer.parseInt(quantityField.getText());

            addProduct(name, quantity);
            refreshTable();
        });

        refreshButton.setOnAction(event -> refreshTable());

        // Create a grid pane and add UI components
        GridPane gridPane = new GridPane();
        gridPane.setPadding(new Insets(10));
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.add(nameLabel, 0, 0);
        gridPane.add(nameField, 1, 0);
        gridPane.add(quantityLabel, 0, 1);
        gridPane.add(quantityField, 1, 1);
        gridPane.add(addButton, 0, 2);
        gridPane.add(refreshButton, 1, 2);
        gridPane.add(tableView, 0, 3, 2, 1);
```

```
        // Set up the scene
        Scene scene = new Scene(gridPane, 400, 300);
        primaryStage.setScene(scene);
        primaryStage.show();

        // Initialize the database connection and load initial data
        initializeDatabase();
        refreshTable();
    }

    private void initializeDatabase() {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();

            // Create the products table if it doesn't exist
            String createTableQuery = "CREATE TABLE IF NOT EXISTS products (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY," +
                    "name VARCHAR(50) NOT NULL," +
                    "quantity INT NOT NULL" +
                    ")";
            statement.executeUpdate(createTableQuery);

            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void addProduct(String name, int quantity) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO products
(name, quantity) VALUES (?, ?)");

            preparedStatement.setString(1, name);
            preparedStatement.setInt(2, quantity);

            preparedStatement.executeUpdate();

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
    }

    private void refreshTable() {
        products = FXCollections.observableArrayList();

        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM products");

            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                int quantity = resultSet.getInt("quantity");

                products.add(new Product(id, name, quantity));
            }

            resultSet.close();
            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        TableView<Product> tableView = (TableView<Product>)
primaryStage.getScene().getRoot().lookup("#tableView");
        tableView.setItems(products);
    }

    public static class Product {
        private final int id;
        private final String name;
        private final int quantity;

        public Product(int id, String name, int quantity) {
            this.id = id;
            this.name = name;
            this.quantity = quantity;
        }

        public int getId() {
            return id;
        }

        public String getName() {
            return name;
        }
```

```
    public int getQuantity() {
        return quantity;
    }

    public StringProperty nameProperty() {
        return new SimpleStringProperty(name);
    }

    public IntegerProperty quantityProperty() {
        return new SimpleIntegerProperty(quantity);
    }
  }
}
```

13.Create an XML database for the student profile and validate it using XML schema.
ANS:

-----students.xml-------

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
   <student>
      <id>101</id>
      <name>binary shade</name>
      <age>20</age>
      <major>Computer Science</major>
   </student>
   <student>
      <id>102</id>
      <name>Dj dark cyber</name>
      <age>22</age>
      <major>Business Administration</major>
   </student>
</students>
```

-------students.xsd----------

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.google.com/XMLSchema">
   <xs:element name="students">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="student" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="id" type="xs:integer"/>
                     <xs:element name="name" type="xs:string"/>
                     <xs:element name="age" type="xs:integer"/>
                     <xs:element name="major" type="xs:string"/>
```

```
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

14. Develop a simple GUI based Cop Friendly App – Eseva database application and incorporate all the
Database features.

ANS;

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.*;

public class CopFriendlyApp extends Application {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/eseva";
    private static final String DB_USERNAME = "your_username";
    private static final String DB_PASSWORD = "your_password";

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Cop Friendly App - Eseva");

        // Create UI components
        Label nameLabel = new Label("Name:");
        TextField nameField = new TextField();
        Label badgeLabel = new Label("Badge Number:");
        TextField badgeField = new TextField();
        Button registerButton = new Button("Register");
        Button loginButton = new Button("Login");

        // Configure button actions
        registerButton.setOnAction(event -> {
            String name = nameField.getText();
            String badgeNumber = badgeField.getText();

            registerUser(name, badgeNumber);
```

```
        });

        loginButton.setOnAction(event -> {
            String badgeNumber = badgeField.getText();

            loginUser(badgeNumber);
        });

        // Create a grid pane and add UI components
        GridPane gridPane = new GridPane();
        gridPane.setPadding(new Insets(10));
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.add(nameLabel, 0, 0);
        gridPane.add(nameField, 1, 0);
        gridPane.add(badgeLabel, 0, 1);
        gridPane.add(badgeField, 1, 1);
        gridPane.add(registerButton, 0, 2);
        gridPane.add(loginButton, 1, 2);

        // Set up the scene
        Scene scene = new Scene(gridPane, 300, 150);
        primaryStage.setScene(scene);
        primaryStage.show();

        // Initialize the database connection and create the necessary table
        initializeDatabase();
    }

    private void initializeDatabase() {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();

            // Create the users table if it doesn't exist
            String createTableQuery = "CREATE TABLE IF NOT EXISTS users (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY," +
                    "name VARCHAR(50) NOT NULL," +
                    "badge_number VARCHAR(20) NOT NULL UNIQUE" +
                    ")";
            statement.executeUpdate(createTableQuery);

            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    private void registerUser(String name, String badgeNumber) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO users
(name, badge_number) VALUES (?, ?)");

            preparedStatement.setString(1, name);
            preparedStatement.setString(2, badgeNumber);

            preparedStatement.executeUpdate();

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void loginUser(String badgeNumber) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM users
WHERE badge_number = ?");

            preparedStatement.setString(1, badgeNumber);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                String name = resultSet.getString("name");
                System.out.println("Welcome, " + name + "!");
            } else {
                System.out.println("Invalid badge number. Please try again.");
            }

            resultSet.close();
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

15.Create a Employee database and write SQL Triggers for insert, delete, and update operations in a

database table.
ANS;

```
-- Create the Employee table
CREATE TABLE Employee (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10, 2)
);

-- Create the trigger for insert operation
CREATE TRIGGER insert_employee_trigger
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
    -- Perform desired actions
    -- Here, we are simply printing a message
    SELECT CONCAT('New employee inserted: ', NEW.name) AS message;
END;

-- Create the trigger for delete operation
CREATE TRIGGER delete_employee_trigger
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
    -- Perform desired actions
    -- Here, we are simply printing a message
    SELECT CONCAT('Employee deleted: ', OLD.name) AS message;
END;

-- Create the trigger for update operation
CREATE TRIGGER update_employee_trigger
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
    -- Perform desired actions
    -- Here, we are simply printing a message
    SELECT CONCAT('Employee updated: ', NEW.name) AS message;
END;
```

16. Create a table Supplier (Sup_No, Sup_Name ,Item_Supplied , Item_Price , City)
ANS;

```
CREATE TABLE Supplier (
    Sup_No INT PRIMARY KEY,
    Sup_Name VARCHAR(50),
    Item_Supplied VARCHAR(50),
    Item_Price DECIMAL(10, 2),
    City VARCHAR(50)
);
```

a)Write sql query to display Suplier numbers and Supplier names whose name starts with ‘S’
ANS; SELECT Sup_No, Sup_Name
FROM Supplier
WHERE Sup_Name LIKE 'S%';

b) Write sql query to add a new column called CONTACTNO.
ANS;
ALTER TABLE Supplier
ADD CONTACTNO VARCHAR(20);

c) Write sql query to display supplier numbers, Suplier names and item price for suppliers in Chennai in the ascending order of item price
ANS;
SELECT Sup_No, Sup_Name, Item_Price
FROM Supplier
WHERE City = 'Chennai'
ORDER BY Item_Price ASC;

d) Create a view on the table which displays only supplier numbers and supplier names.
ANS:
CREATE VIEW SupplierView AS
SELECT Sup_No, Sup_Name
FROM Supplier;

e) Demonstrate the procedure for the supplier table.
ANS;
DELIMITER //

CREATE PROCEDURE GetSuppliers()
BEGIN
   SELECT *
   FROM Supplier;
END //

DELIMITER ;

17. Develop a simple GUI based Banking System and incorporate all the Database features
ANS;
```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.*;

public class BankingSystem extends Application {
```

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/banking";
private static final String DB_USERNAME = "your_username";
private static final String DB_PASSWORD = "your_password";

public static void main(String[] args) {
    launch(args);
}

@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Banking System");

    // Create UI components
    Label nameLabel = new Label("Name:");
    TextField nameField = new TextField();
    Label accountNumberLabel = new Label("Account Number:");
    TextField accountNumberField = new TextField();
    Button balanceButton = new Button("Check Balance");
    Button depositButton = new Button("Deposit");
    Button withdrawButton = new Button("Withdraw");

    // Configure button actions
    balanceButton.setOnAction(event -> {
        String accountNumber = accountNumberField.getText();

        checkBalance(accountNumber);
    });

    depositButton.setOnAction(event -> {
        String accountNumber = accountNumberField.getText();
        double amount = getAmount("Deposit");

        deposit(accountNumber, amount);
    });

    withdrawButton.setOnAction(event -> {
        String accountNumber = accountNumberField.getText();
        double amount = getAmount("Withdraw");

        withdraw(accountNumber, amount);
    });

    // Create a grid pane and add UI components
    GridPane gridPane = new GridPane();
    gridPane.setPadding(new Insets(10));
    gridPane.setHgap(10);
    gridPane.setVgap(10);
    gridPane.add(nameLabel, 0, 0);
```

```
        gridPane.add(nameField, 1, 0);
        gridPane.add(accountNumberLabel, 0, 1);
        gridPane.add(accountNumberField, 1, 1);
        gridPane.add(balanceButton, 0, 2);
        gridPane.add(depositButton, 1, 2);
        gridPane.add(withdrawButton, 2, 2);

        // Set up the scene
        Scene scene = new Scene(gridPane, 300, 150);
        primaryStage.setScene(scene);
        primaryStage.show();

        // Initialize the database connection and create the necessary table
        initializeDatabase();
    }

    private void initializeDatabase() {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();

            // Create the accounts table if it doesn't exist
            String createTableQuery = "CREATE TABLE IF NOT EXISTS accounts (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY," +
                    "name VARCHAR(50) NOT NULL," +
                    "account_number VARCHAR(20) NOT NULL UNIQUE," +
                    "balance DECIMAL(10, 2) NOT NULL" +
                    ")";
            statement.executeUpdate(createTableQuery);

            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void checkBalance(String accountNumber) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("SELECT balance
FROM accounts WHERE account_number = ?");

            preparedStatement.setString(1, accountNumber);

            ResultSet resultSet = preparedStatement.executeQuery();
```

```
            if (resultSet.next()) {
                double balance = resultSet.getDouble("balance");
                System.out.println("Current balance: $" + balance);
            } else {
                System.out.println("Invalid account number. Please try again.");
            }

            resultSet.close();
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void deposit(String accountNumber, double amount) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("UPDATE accounts SET
balance = balance + ? WHERE account_number = ?");

            preparedStatement.setDouble(1, amount);
            preparedStatement.setString(2, accountNumber);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Deposit successful.");
            } else {
                System.out.println("Invalid account number. Please try again.");
            }

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void withdraw(String accountNumber, double amount) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("UPDATE accounts SET
balance = balance - ? WHERE account_number = ? AND balance >= ?");

            preparedStatement.setDouble(1, amount);
            preparedStatement.setString(2, accountNumber);
```

```
            preparedStatement.setDouble(3, amount);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Withdrawal successful.");
            } else {
                System.out.println("Insufficient balance or invalid account number. Please try again.");
            }

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private double getAmount(String operation) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle(operation);
        dialog.setHeaderText(null);
        dialog.setContentText("Enter amount:");

        String amountString = dialog.showAndWait().orElse("0");
        return Double.parseDouble(amountString);
    }
}
```

18.Develop a simple GUI based Employee Pay Roll System and incorporate all the Database features.

ANS:

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.*;

public class PayrollSystem extends Application {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/payroll";
    private static final String DB_USERNAME = "your_username";
    private static final String DB_PASSWORD = "your_password";

    public static void main(String[] args) {
        launch(args);
    }
```

```
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Employee Payroll System");

    // Create UI components
    Label empIdLabel = new Label("Employee ID:");
    TextField empIdField = new TextField();
    Label nameLabel = new Label("Name:");
    TextField nameField = new TextField();
    Label salaryLabel = new Label("Salary:");
    TextField salaryField = new TextField();
    Button addButton = new Button("Add Employee");
    Button viewButton = new Button("View Employees");

    // Configure button actions
    addButton.setOnAction(event -> {
        int empId = Integer.parseInt(empIdField.getText());
        String name = nameField.getText();
        double salary = Double.parseDouble(salaryField.getText());

        addEmployee(empId, name, salary);
    });

    viewButton.setOnAction(event -> viewEmployees());

    // Create a grid pane and add UI components
    GridPane gridPane = new GridPane();
    gridPane.setPadding(new Insets(10));
    gridPane.setHgap(10);
    gridPane.setVgap(10);
    gridPane.add(empIdLabel, 0, 0);
    gridPane.add(empIdField, 1, 0);
    gridPane.add(nameLabel, 0, 1);
    gridPane.add(nameField, 1, 1);
    gridPane.add(salaryLabel, 0, 2);
    gridPane.add(salaryField, 1, 2);
    gridPane.add(addButton, 0, 3);
    gridPane.add(viewButton, 1, 3);

    // Set up the scene
    Scene scene = new Scene(gridPane, 300, 200);
    primaryStage.setScene(scene);
    primaryStage.show();

    // Initialize the database connection and create the necessary table
    initializeDatabase();
}
```

```
    private void initializeDatabase() {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();

            // Create the employees table if it doesn't exist
            String createTableQuery = "CREATE TABLE IF NOT EXISTS employees (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY," +
                    "emp_id INT NOT NULL UNIQUE," +
                    "name VARCHAR(50) NOT NULL," +
                    "salary DECIMAL(10, 2) NOT NULL" +
                    ")";
            statement.executeUpdate(createTableQuery);

            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void addEmployee(int empId, String name, double salary) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO
employees (emp_id, name, salary) VALUES (?, ?, ?)");

            preparedStatement.setInt(1, empId);
            preparedStatement.setString(2, name);
            preparedStatement.setDouble(3, salary);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Employee added successfully.");
            } else {
                System.out.println("Failed to add employee. Please try again.");
            }

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void viewEmployees() {
```

```
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM employees");

            while (resultSet.next()) {
                int empId = resultSet.getInt("emp_id");
                String name = resultSet.getString("name");
                double salary = resultSet.getDouble("salary");

                System.out.println("Employee ID: " + empId);
                System.out.println("Name: " + name);
                System.out.println("Salary: $" + salary);
                System.out.println("-----------------------------");
            }

            resultSet.close();
            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

19.Develop a simple GUI based Movie Ticket Reservation System and incorporate all the Database features.

ANS;

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.sql.*;
import java.time.LocalDate;

public class MovieTicketReservationSystem extends Application {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/movie_ticket_reservation";
    private static final String DB_USERNAME = "your_username";
    private static final String DB_PASSWORD = "your_password";

    public static void main(String[] args) {
        launch(args);
    }
```

```
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Movie Ticket Reservation System");

        // Create UI components
        Label movieLabel = new Label("Movie:");
        TextField movieField = new TextField();
        Label dateLabel = new Label("Date:");
        DatePicker datePicker = new DatePicker();
        Label seatsLabel = new Label("Seats:");
        TextField seatsField = new TextField();
        Button reserveButton = new Button("Reserve");

        // Configure button action
        reserveButton.setOnAction(event -> {
            String movie = movieField.getText();
            LocalDate date = datePicker.getValue();
            int seats = Integer.parseInt(seatsField.getText());

            reserveSeats(movie, date, seats);
        });

        // Create a grid pane and add UI components
        GridPane gridPane = new GridPane();
        gridPane.setPadding(new Insets(10));
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.add(movieLabel, 0, 0);
        gridPane.add(movieField, 1, 0);
        gridPane.add(dateLabel, 0, 1);
        gridPane.add(datePicker, 1, 1);
        gridPane.add(seatsLabel, 0, 2);
        gridPane.add(seatsField, 1, 2);
        gridPane.add(reserveButton, 0, 3);

        // Set up the scene
        Scene scene = new Scene(gridPane, 300, 200);
        primaryStage.setScene(scene);
        primaryStage.show();

        // Initialize the database connection and create the necessary table
        initializeDatabase();
    }

    private void initializeDatabase() {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
```

```
            Statement statement = connection.createStatement();

            // Create the reservations table if it doesn't exist
            String createTableQuery = "CREATE TABLE IF NOT EXISTS reservations (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY," +
                    "movie VARCHAR(50) NOT NULL," +
                    "date DATE NOT NULL," +
                    "seats INT NOT NULL" +
                    ")";
            statement.executeUpdate(createTableQuery);

            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void reserveSeats(String movie, LocalDate date, int seats) {
        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USERNAME,
DB_PASSWORD);
            PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO
reservations (movie, date, seats) VALUES (?, ?, ?)");

            preparedStatement.setString(1, movie);
            preparedStatement.setDate(2, Date.valueOf(date));
            preparedStatement.setInt(3, seats);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Seats reserved successfully.");
            } else {
                System.out.println("Failed to reserve seats. Please try again.");
            }

            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

20.Develop a simple GUI based Super Market Stock Maintenance System and incorporate all the Database features

ANS;

```
import javafx.application.Application;
```

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class StockMaintenanceSystem extends Application {

    private TableView<Product> table;
    private TextField nameInput;
    private TextField quantityInput;

    private ObservableList<Product> products = FXCollections.observableArrayList();

    private static final String DB_URL = "jdbc:mysql://localhost:3306/your_database_name";
    private static final String DB_USER = "your_username";
    private static final String DB_PASSWORD = "your_password";

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Super Market Stock Maintenance System");

        // Create table columns
        TableColumn<Product, String> nameColumn = new TableColumn<>("Product Name");
        nameColumn.setCellValueFactory(cellData -> cellData.getValue().nameProperty());

        TableColumn<Product, Integer> quantityColumn = new TableColumn<>("Quantity");
        quantityColumn.setCellValueFactory(cellData ->
cellData.getValue().quantityProperty().asObject());

        // Create table
        table = new TableView<>();
        table.setItems(products);
        table.getColumns().addAll(nameColumn, quantityColumn);

        // Create input fields
```

```
        nameInput = new TextField();
        nameInput.setPromptText("Product Name");
        nameInput.setMinWidth(150);

        quantityInput = new TextField();
        quantityInput.setPromptText("Quantity");

        // Create buttons
        Button addButton = new Button("Add");
        addButton.setOnAction(e -> addProduct());

        Button deleteButton = new Button("Delete");
        deleteButton.setOnAction(e -> deleteProduct());

        // Create layout
        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(8);
        grid.setHgap(10);

        grid.add(new Label("Product Name:"), 0, 0);
        grid.add(nameInput, 1, 0);
        grid.add(new Label("Quantity:"), 0, 1);
        grid.add(quantityInput, 1, 1);
        grid.add(addButton, 2, 0);
        grid.add(deleteButton, 2, 1);

        HBox hBox = new HBox();
        hBox.getChildren().addAll(table, grid);

        // Load data from the database
        loadData();

        // Create scene and show the stage
        Scene scene = new Scene(hBox);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void addProduct() {
        String name = nameInput.getText();
        int quantity = Integer.parseInt(quantityInput.getText());

        Product product = new Product(name, quantity);
        products.add(product);

        saveProductToDatabase(product);

        nameInput.clear();
```

```
        quantityInput.clear();
    }

    private void deleteProduct() {
        Product product = table.getSelectionModel().getSelectedItem();
        if (product != null) {
            products.remove(product);
            deleteProductFromDatabase(product);
        }
    }

    private void saveProductToDatabase(Product product) {
        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String sql = "INSERT INTO products (name, quantity) VALUES (?, ?)";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, product.getName());
            statement.setInt(2, product.getQuantity());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void deleteProductFromDatabase(Product product) {
        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String sql = "DELETE FROM products WHERE name = ?";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, product.getName());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void loadData() {
        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
            String sql = "SELECT * FROM products";
            PreparedStatement statement = connection.prepareStatement(sql);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()) {
                String name = resultSet.getString("name");
                int quantity = resultSet.getInt("quantity");
                products.add(new Product(name, quantity));
            }
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    public static class Product {
        private final String name;
        private final Integer quantity;

        public Product(String name, Integer quantity) {
            this.name = name;
            this.quantity = quantity;
        }

        public String getName() {
            return name;
        }

        public Integer getQuantity() {
            return quantity;
        }

        public StringProperty nameProperty() {
            return new SimpleStringProperty(name);
        }

        public IntegerProperty quantityProperty() {
            return new SimpleIntegerProperty(quantity);
        }
    }
}
```