1.create a web page using the advanced features of css:Grid,Flexbox.And apply transition and animations on the contents of the webpage

HTML

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>CSS Grid and Flexbox Example</title>

   <link rel="stylesheet" href="styles.css">

</head>

<body>

   <header class="header">

      <h1>My Awesome Web Page</h1>

   </header>

   <main class="main-content">

      <section class="grid-container">

         <div class="grid-item">Item 1</div>

         <div class="grid-item">Item 2</div>

         <div class="grid-item">Item 3</div>

         <div class="grid-item">Item 4</div>

         <div class="grid-item">Item 5</div>

      </section>

   </main>

   <footer class="footer">
```

```
        <p>Footer Content</p>
    </footer>
</body>
</html>
```

CSS

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    height: 100vh;
}

.header {
    background-color: #4CAF50;
    color: white;
    text-align: center;
    padding: 20px;
}
```

```
.main-content {
    flex: 1;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #f4f4f4;
}

.grid-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    gap: 15px;
    padding: 20px;
}

.grid-item {
    background-color: #ffcc00;
    padding: 20px;
    border-radius: 5px;
    text-align: center;
    transition: transform 0.3s ease, background-color 0.3s ease;
}

.grid-item:hover {
```

```
    transform: scale(1.1);
    background-color: #ffd700;
}

.footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
}

@keyframes fadeIn {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

.header, .footer {
    animation: fadeIn 1s ease-in;
}
```

2.make the webpages created in the above experiment as responsive web page with Bootstrap

Framework

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Web Page with Bootstrap</title>
                                                    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header class="bg-success text-white text-center py-4">
    <h1>My Awesome Web Page</h1>
  </header>
  <main class="container my-5">
    <section class="row">
      <div class="col-lg-4 col-md-6 mb-4">
        <div class="card grid-item">
          <div class="card-body">
            <h5 class="card-title">Item 1</h5>
          </div>
        </div>
      </div>
```

```
<div class="col-lg-4 col-md-6 mb-4">
   <div class="card grid-item">
      <div class="card-body">
         <h5 class="card-title">Item 2</h5>
      </div>
   </div>
</div>
<div class="col-lg-4 col-md-6 mb-4">
   <div class="card grid-item">
      <div class="card-body">
         <h5 class="card-title">Item 3</h5>
      </div>
   </div>
</div>
<div class="col-lg-4 col-md-6 mb-4">
   <div class="card grid-item">
      <div class="card-body">
         <h5 class="card-title">Item 4</h5>
      </div>
   </div>
</div>
<div class="col-lg-4 col-md-6 mb-4">
   <div class="card grid-item">
      <div class="card-body">
         <h5 class="card-title">Item 5</h5>
```

```
            </div>
          </div>
        </div>
      </section>
    </main>
    <footer class="bg-dark text-white text-center py-3">
      <p>Footer Content</p>
    </footer>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

CSS

```
.grid-item {
    background-color: #ffcc00;
    transition: transform 0.3s ease, background-color 0.3s ease;
}

.grid-item:hover {
    transform: scale(1.05);
    background-color: #ffd700;
}
```

3.Validate the registration ,user login,user profile and payment pages using javascript .Make use of any needed javascrpt objects

HTML

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Registration</title>

   <link rel="stylesheet" href="styles.css">

</head>

<body>

   <h1>Register</h1>

   <form id="registrationForm">

      <input type="text" id="username" placeholder="Username" required>

      <input type="email" id="email" placeholder="Email" required>

      <input type="password" id="password" placeholder="Password" required>

      <button type="submit">Register</button>

      <div id="message" class="error-message"></div>

   </form>

   <script src="script.js"></script>

</body>

</html>
```

script.js

```
document.getElementById("registrationForm").addEventListener("submit", function(event) {
    event.preventDefault(); // Prevent form submission
    validateRegistration();
});

function validateRegistration() {
    const username = document.getElementById("username").value;
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value;
    const messageDiv = document.getElementById("message");
    messageDiv.textContent = "";

    // Simple validation
    if (username.length < 3) {
        messageDiv.textContent = "Username must be at least 3 characters.";
        return;
    }
    if (!/\S+@\S+\.\S+/.test(email)) {
        messageDiv.textContent = "Email is not valid.";
        return;
    }
    if (password.length < 6) {
        messageDiv.textContent = "Password must be at least 6 characters.";
        return;
    }
```

```
    messageDiv.textContent = "Registration successful!";
    // You can proceed to send the data to the server here
}
```

User Login Page(HTML)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Login</h1>
    <form id="loginForm">
        <input type="email" id="loginEmail" placeholder="Email" required>
        <input type="password" id="loginPassword" placeholder="Password" required>
        <button type="submit">Login</button>
        <div id="loginMessage" class="error-message"></div>
    </form>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.getElementById("loginForm").addEventListener("submit", function(event) {
    event.preventDefault(); // Prevent form submission
    validateLogin();
});

function validateLogin() {
    const email = document.getElementById("loginEmail").value;
    const password = document.getElementById("loginPassword").value;
    const loginMessageDiv = document.getElementById("loginMessage");
    loginMessageDiv.textContent = "";

    // Simple validation
    if (!/\S+@\S+\.\S+/.test(email)) {
        loginMessageDiv.textContent = "Email is not valid.";
        return;
    }
    if (password.length < 6) {
        loginMessageDiv.textContent = "Password must be at least 6 characters.";
        return;
    }

    loginMessageDiv.textContent = "Login successful!";
    // You can proceed to authenticate the user here
}
```

User Profile Page(HTML)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>User Profile</h1>
    <form id="profileForm">
        <input type="text" id="profileUsername" placeholder="Username" required>
        <input type="email" id="profileEmail" placeholder="Email" required>
        <button type="submit">Update Profile</button>
        <div id="profileMessage" class="error-message"></div>
    </form>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.getElementById("profileForm").addEventListener("submit", function(event) {
    event.preventDefault(); // Prevent form submission
```

```
    validateProfile();
});

function validateProfile() {
    const username = document.getElementById("profileUsername").value;
    const email = document.getElementById("profileEmail").value;
    const profileMessageDiv = document.getElementById("profileMessage");
    profileMessageDiv.textContent = "";

    // Simple validation
    if (username.length < 3) {
        profileMessageDiv.textContent = "Username must be at least 3 characters.";
        return;
    }
    if (!/\S+@\S+\.\S+/.test(email)) {
        profileMessageDiv.textContent = "Email is not valid.";
        return;
    }

    profileMessageDiv.textContent = "Profile updated successfully!";
    // You can proceed to update the user profile on the server here
}
```

Payment page(HTML)

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Payment</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Payment</h1>
    <form id="paymentForm">
        <input type="text" id="cardNumber" placeholder="Card Number" required>
        <input type="text" id="expiryDate" placeholder="MM/YY" required>
        <input type="text" id="cvv" placeholder="CVV" required>
        <button type="submit">Pay</button>
        <div id="paymentMessage" class="error-message"></div>
    </form>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
document.getElementById("paymentForm").addEventListener("submit", function(event) {
    event.preventDefault(); // Prevent form submission
    validatePayment();
});
```

```
function validatePayment() {
    const cardNumber = document.getElementById("cardNumber").value;
    const expiryDate = document.getElementById("expiryDate").value;
    const cvv = document.getElementById("cvv").value;
    const paymentMessageDiv = document.getElementById("paymentMessage");
    paymentMessageDiv.textContent = "";

    // Simple validation
    if (!/^\d{16}$/.test(cardNumber)) {
        paymentMessageDiv.textContent = "Card number must be 16 digits.";
        return;
    }
    if (!/^(0[1-9]|1[0-2])\/\d{2}$/.test(expiryDate)) {
        paymentMessageDiv.textContent = "Expiry date must be in MM/YY format.";
        return;
    }
    if (!/^\d{3}$/.test(cvv)) {
        paymentMessageDiv.textContent = "CVV must be 3 digits.";
        return;
    }

    paymentMessageDiv.textContent = "Payment successful!";
    // You can proceed to process the payment here
}
```

4.Build a scientific calculator

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Scientific Calculator</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="calculator">
        <input type="text" id="display" disabled>
        <div class="buttons">
            <button onclick="clearDisplay()">C</button>
            <button onclick="appendToDisplay('7')">7</button>
            <button onclick="appendToDisplay('8')">8</button>
            <button onclick="appendToDisplay('9')">9</button>
            <button onclick="appendToDisplay('/')">/</button>
            <button onclick="appendToDisplay('4')">4</button>
            <button onclick="appendToDisplay('5')">5</button>
            <button onclick="appendToDisplay('6')">6</button>
            <button onclick="appendToDisplay('*')">*</button>
            <button onclick="appendToDisplay('1')">1</button>
            <button onclick="appendToDisplay('2')">2</button>
            <button onclick="appendToDisplay('3')">3</button>
```

```
        <button onclick="appendToDisplay('-')">-</button>
        <button onclick="appendToDisplay('0')">0</button>
        <button onclick="appendToDisplay('.')">.</button>
        <button onclick="calculate()">=</button>
        <button onclick="appendToDisplay('+')">+</button>
        <button onclick="calculate('sqrt')">√</button>
        <button onclick="calculate('pow')">x²</button>
        <button onclick="calculate('sin')">sin</button>
        <button onclick="calculate('cos')">cos</button>
        <button onclick="calculate('tan')">tan</button>
      </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

CSS

```
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f4f4f4;
  font-family: Arial, sans-serif;
}
```

```
.calculator {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    width: 300px;
}

#display {
    width: 100%;
    height: 40px;
    text-align: right;
    font-size: 24px;
    border: 1px solid #ccc;
    border-radius: 5px;
    margin-bottom: 10px;
    padding: 5px;
}

.buttons {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}
```

```
button {
    height: 40px;
    font-size: 18px;
    border: none;
    border-radius: 5px;
    background-color: #007bff;
    color: white;
    cursor: pointer;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #0056b3;
}
```

script.js

```
function appendToDisplay(value) {
    document.getElementById("display").value += value;
}

function clearDisplay() {
    document.getElementById("display").value = "";
}

function calculate(operation) {
    const display = document.getElementById("display");
```

```
    let result;


    try {
        if (operation === 'sqrt') {
            result = Math.sqrt(eval(display.value));
        } else if (operation === 'pow') {
            result = Math.pow(eval(display.value), 2);
        } else if (['sin', 'cos', 'tan'].includes(operation)) {
            const angle = eval(display.value) * (Math.PI / 180); // Convert to radians
            result = Math[operation](angle);
        } else {
            result = eval(display.value);
        }


        display.value = result;
    } catch (error) {
        display.value = "Error!";
    }
}
```

5.Javascript program to demostrate working of prototypal inheritance ,closure,callbacks,promises and sync/await

```
// Prototypal Inheritance
function Animal(name) {
    this.name = name;
}
```

```
Animal.prototype.speak = function() {
  console.log(`${this.name} makes a noise.`);
};

function Dog(name) {
  Animal.call(this, name); // Call the parent constructor
}

Dog.prototype = Object.create(Animal.prototype);
Dog.prototype.constructor = Dog;

Dog.prototype.speak = function() {
  console.log(`${this.name} barks.`);
};

// Closure
function createCounter() {
  let count = 0; // Private variable

  return {
    increment: function() {
      count++;
      return count;
    },
```

```
    decrement: function() {
      count--;
      return count;
    },
    getCount: function() {
      return count;
    }
  };
}

// Callback
function fetchData(callback) {
  setTimeout(() => {
    const data = { message: "Data fetched!" };
    callback(data);
  }, 1000);
}

// Promise
function fetchDataPromise() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const data = { message: "Data fetched with Promise!" };
      resolve(data);
    }, 1000);
```

```
    });
}

// Async/Await
async function fetchDataAsync() {
    const data = await fetchDataPromise();
    console.log(data.message);
}

// Demonstration
function demo() {
    // Prototypal Inheritance
    const dog = new Dog('Buddy');
    dog.speak(); // Output: Buddy barks.

    // Closure
    const counter = createCounter();
    console.log(counter.increment()); // Output: 1
    console.log(counter.increment()); // Output: 2
    console.log(counter.decrement()); // Output: 1
    console.log(counter.getCount()); // Output: 1

    // Callback
    fetchData((data) => {
        console.log(data.message); // Output: Data fetched!
```

```
  });

  // Promise
  fetchDataPromise().then((data) => {
    console.log(data.message); // Output: Data fetched with Promise!
  });

  // Async/Await
  fetchDataAsync(); // Output: Data fetched with Promise!
}

// Run the demonstration
demo();
```

6.Write an xml file which will display the Book information with the following fields : Title of the book,Author name,ISBN number ,Publisher name,Edition,Price.

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <isbn>978-0743273565</isbn>
    <publisher>Scribner</publisher>
    <edition>1st</edition>
    <price>10.99</price>
```

```
</book>
<book>
  <title>To Kill a Mockingbird</title>
  <author>Harper Lee</author>
  <isbn>978-0061120084</isbn>
  <publisher>Harper Perennial Modern Classics</publisher>
  <edition>50th Anniversary Edition</edition>
  <price>7.19</price>
</book>
<book>
  <title>1984</title>
  <author>George Orwell</author>
  <isbn>978-0451524935</isbn>
  <publisher>Signet Classics</publisher>
  <edition>Anniversary Edition</edition>
  <price>9.99</price>
</book>
<book>
  <title>Moby Dick</title>
  <author>Herman Melville</author>
  <isbn>978-1503280786</isbn>
  <publisher>CreateSpace Independent Publishing Platform</publisher>
  <edition>1st</edition>
  <price>11.95</price>
</book>
```

```
  <book>
    <title>Brave New World</title>
    <author>Aldous Huxley</author>
    <isbn>978-0060850524</isbn>
    <publisher>Harper Perennial Modern Classics</publisher>
    <edition>Reissue</edition>
    <price>14.99</price>
  </book>
</library>
```

7.Define a Document Type Definition(DTD) and xml schema to validate the above created xml Documents

books.dtd

```
<!ELEMENT library (book+)>
<!ELEMENT book (title, author, isbn, publisher, edition, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

books.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="author" type="xs:string"/>
              <xs:element name="isbn" type="xs:string"/>
              <xs:element name="publisher" type="xs:string"/>
              <xs:element name="edition" type="xs:string"/>
              <xs:element name="price" type="xs:decimal"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

8.8.write a java program to establish a connection to a database and execute simple SQL queries

SQL

```
use itb;
create table students(name varchar(50),rollno int,branch varchar(20));
insert into students (name,rollno,branch) values
```

('niha',90,'it'),

('nishka',95,'it'),

('nishu',96,'it');

JAVA

```
package jdbc;

import java.sql.*;

public class Simple_Connection {

	Connection conn = null;

	public static void main(String[] args) {

		// TODO Auto-generated method stub

		try {

			Class.forName("com.mysql.cj.jdbc.Driver");

			Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/itb","root","root");

			Statement stmt  = conn.createStatement();

			ResultSet rs=stmt.executeQuery("Select * from students");

			while(rs.next()){

				System.out.println(rs.getString(1)+"
"+rs.getInt(2)+" "+rs.getString(3));

			}

			conn.close();


		}catch(Exception e) {

			e.printStackTrace();

		}

	}

}
```

9.write a java program to demonstrate the usage of JDBC in performing various DML statements

.Use prepared statements and callable statements

SQL

```
CREATE DATABASE sampledb;


USE sampledb;


CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100)
);
```

JAVA

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


public class JdbcDmlExample {


   private static final String URL = "jdbc:mysql://localhost:3306/sampledb";
   private static final String USERNAME = "your_username"; // Replace with your DB username
   private static final String PASSWORD = "your_password"; // Replace with your DB password
```

```
public static void main(String[] args) {
    Connection connection = null;

    try {
        // Establishing the connection
        connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        System.out.println("Connected to the database successfully.");

        // Inserting data using PreparedStatement
        insertUser(connection, "Alice Johnson", "alice@example.com");
        insertUser(connection, "Bob Smith", "bob@example.com");

        // Updating data using PreparedStatement
        updateUserEmail(connection, 1, "alice.new@example.com");

        // Calling stored procedure using CallableStatement
        callUserCountProcedure(connection);

    } catch (SQLException e) {
        System.err.println("SQL Exception: " + e.getMessage());
    } finally {
        // Closing the connection
        try {
            if (connection != null && !connection.isClosed()) {
                connection.close();
```

```
            System.out.println("Database connection closed.");
        }
      } catch (SQLException e) {
        System.err.println("Failed to close the connection: " + e.getMessage());
      }
    }
  }

  // Method to insert user using PreparedStatement
    private static void insertUser(Connection connection, String name, String email) throws SQLException {
    String insertSQL = "INSERT INTO users (name, email) VALUES (?, ?)";
    try (PreparedStatement pstmt = connection.prepareStatement(insertSQL)) {
      pstmt.setString(1, name);
      pstmt.setString(2, email);
      int rowsAffected = pstmt.executeUpdate();
      System.out.println(rowsAffected + " user(s) inserted.");
    }
  }

  // Method to update user email using PreparedStatement
   private static void updateUserEmail(Connection connection, int userId, String newEmail) throws SQLException {
    String updateSQL = "UPDATE users SET email = ? WHERE id = ?";
    try (PreparedStatement pstmt = connection.prepareStatement(updateSQL)) {
      pstmt.setString(1, newEmail);
```

```
            pstmt.setInt(2, userId);
            int rowsAffected = pstmt.executeUpdate();
            System.out.println(rowsAffected + " user(s) updated.");
        }
    }

    // Method to call a stored procedure using CallableStatement
    private static void callUserCountProcedure(Connection connection) throws SQLException {
        // Assuming there is a stored procedure named `GetUserCount` that returns the count of users
        String procedureCall = "{ CALL GetUserCount() }";
        try (CallableStatement cstmt = connection.prepareCall(procedureCall)) {
            try (ResultSet rs = cstmt.executeQuery()) {
                if (rs.next()) {
                    int userCount = rs.getInt(1);
                    System.out.println("Total users: " + userCount);
                }
            }
        }
    }
}
```

10.write a java based application to demonstrate the Updatable and Scrollable resultsets

SQL

```
CREATE DATABASE sampledb;

USE sampledb;
```

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100)
);
```

JAVA

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

public class UpdatableScrollableResultSetExample {
    private static final String URL = "jdbc:mysql://localhost:3306/sampledb";
    private static final String USERNAME = "your_username"; // Replace with your DB username
    private static final String PASSWORD = "your_password"; // Replace with your DB password

    public static void main(String[] args) {
        Connection connection = null;

        try {
            // Establishing the connection
            connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
```

```
        System.out.println("Connected to the database successfully.");

        // Inserting some sample data
        insertSampleData(connection);

        // Using Updatable and Scrollable ResultSet
        try (Statement stmt = connection.createStatement(
                ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE)) {

            String query = "SELECT * FROM users";
            ResultSet rs = stmt.executeQuery(query);

            // Moving to the last record
            if (rs.last()) {
                System.out.println("Last Record: ID: " + rs.getInt("id") + ", Name: " +
rs.getString("name") + ", Email: " + rs.getString("email"));
            }

            // Moving to the first record
            if (rs.first()) {
                System.out.println("First Record: ID: " + rs.getInt("id") + ", Name: " +
rs.getString("name") + ", Email: " + rs.getString("email"));
            }

            // Updating the first record
            if (rs.first()) {
```

```
            System.out.println("Updating record...");
            rs.updateString("name", "Updated Name");
            rs.updateRow();
            System.out.println("Record updated.");
        }

        // Displaying updated records
        System.out.println("Updated Records:");
        rs.beforeFirst(); // Move cursor to before the first record
        while (rs.next()) {
            System.out.println("ID: " + rs.getInt("id") + ", Name: " + rs.getString("name") + ", Email: " + rs.getString("email"));
        }

      }

    } catch (SQLException e) {
      System.err.println("SQL Exception: " + e.getMessage());
    } finally {
      // Closing the connection
      try {
        if (connection != null && !connection.isClosed()) {
          connection.close();
          System.out.println("Database connection closed.");
        }
```

```
            } catch (SQLException e) {
                System.err.println("Failed to close the connection: " + e.getMessage());
            }
        }
    }

    // Method to insert sample data into the users table
    private static void insertSampleData(Connection connection) throws SQLException {
        String insertSQL = "INSERT INTO users (name, email) VALUES (?, ?)";
        try (PreparedStatement pstmt = connection.prepareStatement(insertSQL)) {
            pstmt.setString(1, "John Doe");
            pstmt.setString(2, "john@example.com");
            pstmt.executeUpdate();
            pstmt.setString(1, "Jane Smith");
            pstmt.setString(2, "jane@example.com");
            pstmt.executeUpdate();
            System.out.println("Sample data inserted into users table.");
        }
    }
}
```

11.write a java program to access metadata of the SQL database

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.DatabaseMetaData;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;


public class DatabaseMetadataExample {

    private static final String URL = "jdbc:mysql://localhost:3306/sampledb"; // Replace with your
DB URL

   private static final String USERNAME = "your_username"; // Replace with your DB username

   private static final String PASSWORD = "your_password"; // Replace with your DB password


   public static void main(String[] args) {

      Connection connection = null;


      try {

         // Establishing the connection

         connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

         System.out.println("Connected to the database successfully.");


         // Accessing database metadata

         DatabaseMetaData metaData = connection.getMetaData();


         // Retrieve and display basic information

         System.out.println("Database Product Name: " + metaData.getDatabaseProductName());

         System.out.println("Database Product Version: " + metaData.getDatabaseProductVersion());

         System.out.println("Driver Name: " + metaData.getDriverName());

         System.out.println("Driver Version: " + metaData.getDriverVersion());

         System.out.println("SQL Syntax: " + metaData.getSQLKeywords());
```

```
    // Get and display the tables
    System.out.println("\nTables in the database:");
    ResultSet tables = metaData.getTables(null, null, "%", new String[]{"TABLE"});
    while (tables.next()) {
        String tableName = tables.getString("TABLE_NAME");
        System.out.println("Table: " + tableName);

        // Get and display columns for each table
        ResultSet columns = metaData.getColumns(null, null, tableName, null);
        System.out.println("  Columns in " + tableName + ":");
        while (columns.next()) {
            String columnName = columns.getString("COLUMN_NAME");
            String columnType = columns.getString("TYPE_NAME");
            int columnSize = columns.getInt("COLUMN_SIZE");
            System.out.printf("    %s (%s, Size: %d)%n", columnName, columnType, columnSize);
        }
    }

} catch (SQLException e) {
    System.err.println("SQL Exception: " + e.getMessage());
} finally {
    // Closing the connection
    try {
        if (connection != null && !connection.isClosed()) {
```

```
            connection.close();
            System.out.println("Database connection closed.");
        }
      } catch (SQLException e) {
        System.err.println("Failed to close the connection: " + e.getMessage());
      }
    }
  }
}
```

12. write a java program to accept request parameters a form and generate the response

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Form</title>
</head>
<body>
  <h1>User Information Form</h1>
  <form action="ResponseServlet" method="POST">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name" required><br>

    <label for="email">Email:</label><br>
```

```
    <input type="email" id="email" name="email" required><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

ResponseServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ResponseServlet")
public class ResponseServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // Set the response content type
    response.setContentType("text/html");

    // Get the parameters from the request
```

```
        String name = request.getParameter("name");
        String email = request.getParameter("email");

        // Generate the response
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>User Information</h2>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Email: " + email + "</p>");
        out.println("</body></html>");
    }
}
```

13.write a program to accept ServletConfig and ServletContext parameters

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ConfigServlet")
```

```
public class ConfigServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      // Set the response content type
      response.setContentType("text/html");

      // Get ServletConfig and ServletContext
      ServletConfig config = getServletConfig();
      ServletContext context = getServletContext();

      // Get parameters from ServletConfig
      String servletParam = config.getInitParameter("servletParam");

      // Get parameters from ServletContext
      String contextParam = context.getInitParameter("contextParam");

      // Generate the response
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      out.println("<h2>Servlet Config and Context Parameters</h2>");
      out.println("<p>Servlet Parameter: " + servletParam + "</p>");
      out.println("<p>Context Parameter: " + contextParam + "</p>");
      out.println("</body></html>");
```

```
    }
}
```