===================================================

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

on the contents of the webpage

===================================================

=========index.html====

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Advanced CSS Page</title>

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

</head>

<body>

<header class="header">

<h1>Welcome to My Advanced CSS Page</h1>

</header>

<nav class="nav">

<a href="#">Home</a>

<a href="#">About</a>

<a href="#">Services</a>

<a href="#">Contact</a>

</nav>

<main class="main">

<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>

</section>

</main>

<footer class="footer">

<p>Created with ❤️ using Grid, Flexbox, and Animations!</p>

</footer>

</body>

</html>

==============styles.css===

/\* General Reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Body styling \*/

body {

font-family: Arial, sans-serif;

line-height: 1.6;

display: grid;

grid-template-rows: auto 50px 1fr auto;

grid-template-areas:

"header"

"nav"

"main"

"footer";

min-height: 100vh;

}

/\* Header \*/

.header {

grid-area: header;

background: linear-gradient(45deg, #6b5b95, #feb236);

color: white;

text-align: center;

padding: 20px;

}

/\* Navigation Bar \*/

.nav {

grid-area: nav;

display: flex;

justify-content: space-around;

align-items: center;

background: #88d8b0;

}

.nav a {

color: white;

text-decoration: none;

font-weight: bold;

transition: color 0.3s ease-in-out;

}

.nav a:hover {

color: #d64161;

}

/\* Main Content \*/

.main {

grid-area: main;

padding: 20px;

}

/\* Grid Container \*/

.grid-container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

gap: 20px;

}

.grid-item {

background: #ff6f61;

color: white;

text-align: center;

padding: 40px;

border-radius: 10px;

font-size: 1.2rem;

transition: transform 0.3s ease-in-out, background 0.3s ease-in-out;

}

.grid-item:hover {

transform: scale(1.1);

background: #d64161;

}

/\* Footer \*/

.footer {

grid-area: footer;

text-align: center;

background: #6b5b95;

color: white;

padding: 10px;

}

/\* Adding Animation \*/

@keyframes fadeIn {

0% {

opacity: 0;

transform: translateY(20px);

}

100% {

opacity: 1;

transform: translateY(0);

}

}

.grid-item {

animation: fadeIn 1s ease-out;

}

===================================================

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

===================================================

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Responsive Webpage with Bootstrap</title>

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->

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

</head>

<body>

<!-- Header -->

<header class="header">

<h1>Welcome to My Responsive Webpage</h1>

</header>

<!-- Navigation Bar -->

<nav class="navbar navbar-expand-lg bg-primary">

<div class="container-fluid">

<a class="navbar-brand text-white" href="#">Brand</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav ms-auto">

<li class="nav-item"><a class="nav-link" href="#">Home</a></li>

<li class="nav-item"><a class="nav-link" href="#">About</a></li>

<li class="nav-item"><a class="nav-link" href="#">Services</a></li>

<li class="nav-item"><a class="nav-link" href="#">Contact</a></li>

</ul>

</div>

</div>

</nav>

<!-- Main Content -->

<main class="container my-5">

<div class="row g-4">

<div class="col-md-6 col-lg-3">

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

</div>

<div class="col-md-6 col-lg-3">

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

</div>

<div class="col-md-6 col-lg-3">

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

</div>

<div class="col-md-6 col-lg-3">

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

</div>

</div>

</main>

<!-- Footer -->

<footer>

<p>Created with ❤️ using Bootstrap!</p>

</footer>

<!-- Bootstrap JavaScript -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

===========css=========

/\* Reset and General Styles \*/

body {

font-family: Arial, sans-serif;

line-height: 1.6;

}

/\* Header Styles \*/

.header {

background: linear-gradient(45deg, #6b5b95, #feb236);

color: white;

padding: 20px;

text-align: center;

}

/\* Navigation Bar Styles \*/

.nav-link {

color: white !important;

font-weight: bold;

transition: color 0.3s ease-in-out;

}

.nav-link:hover {

color: #d64161 !important;

}

/\* Main Content (Grid Items) \*/

.grid-item {

background: #ff6f61;

color: white;

padding: 40px;

text-align: center;

border-radius: 10px;

font-size: 1.2rem;

transition: transform 0.3s ease-in-out, background 0.3s ease-in-out;

}

.grid-item:hover {

transform: scale(1.1);

background: #d64161;

}

/\* Footer Styles \*/

footer {

background: #6b5b95;

color: white;

text-align: center;

padding: 10px;

}

================================================

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

==============================================

index.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">

<h1>Scientific Calculator</h1>

<!-- Display Screen -->

<input type="text" id="display" readonly>

<!-- Buttons -->

<div class="buttons">

<!-- Basic Operations -->

<button onclick="appendValue('7')">7</button>

<button onclick="appendValue('8')">8</button>

<button onclick="appendValue('9')">9</button>

<button onclick="performOperation('/')">/</button>

<button onclick="appendValue('4')">4</button>

<button onclick="appendValue('5')">5</button>

<button onclick="appendValue('6')">6</button>

<button onclick="performOperation('\*')">\*</button>

<button onclick="appendValue('1')">1</button>

<button onclick="appendValue('2')">2</button>

<button onclick="appendValue('3')">3</button>

<button onclick="performOperation('-')">-</button>

<button onclick="appendValue('0')">0</button>

<button onclick="appendValue('.')">.</button>

<button onclick="calculate()">=</button>

<button onclick="performOperation('+')">+</button>

<!-- Clear and Scientific Functions -->

<button onclick="clearDisplay()">C</button>

<button onclick="calculateSquareRoot()">√</button>

<button onclick="calculatePower()">x²</button>

<button onclick="calculateSin()">sin</button>

<button onclick="calculateCos()">cos</button>

<button onclick="calculateTan()">tan</button>

</div>

</div>

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

</body>

</html>

script.js

// Display Screen

const display = document.getElementById("display");

// Append values to the display

function appendValue(value) {

display.value += value;

}

// Perform basic operations

function performOperation(operator) {

display.value += operator;

}

// Clear the display

function clearDisplay() {

display.value = "";

}

// Calculate the result of the expression

function calculate() {

try {

display.value = eval(display.value);

} catch (error) {

display.value = "Error!";

}

}

// Calculate the square root

function calculateSquareRoot() {

const value = parseFloat(display.value);

if (!isNaN(value)) {

display.value = Math.sqrt(value).toFixed(2);

} else {

display.value = "Error!";

}

}

// Calculate the square

function calculatePower() {

const value = parseFloat(display.value);

if (!isNaN(value)) {

display.value = Math.pow(value, 2).toFixed(2);

} else {

display.value = "Error!";

}

}

// Calculate sine

function calculateSin() {

const value = parseFloat(display.value);

if (!isNaN(value)) {

display.value = Math.sin(value \* Math.PI / 180).toFixed(2);

} else {

display.value = "Error!";

}

}

// Calculate cosine

function calculateCos() {

const value = parseFloat(display.value);

if (!isNaN(value)) {

display.value = Math.cos(value \* Math.PI / 180).toFixed(2);

} else {

display.value = "Error!";

}

}

// Calculate tangent

function calculateTan() {

const value = parseFloat(display.value);

if (!isNaN(value)) {

display.value = Math.tan(value \* Math.PI / 180).toFixed(2);

} else {

display.value = "Error!";

}

}

styles.csss

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background: #f4f4f4;

}

.calculator {

width: 300px;

padding: 20px;

background: #fff;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

.calculator h1 {

text-align: center;

margin-bottom: 20px;

color: #333;

}

#display {

width: 100%;

height: 50px;

font-size: 1.5rem;

margin-bottom: 10px;

padding: 5px;

text-align: right;

border: 1px solid #ccc;

border-radius: 5px;

background: #f9f9f9;

}

.buttons {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 10px;

}

button {

padding: 15px;

font-size: 1rem;

border: none;

border-radius: 5px;

background: #007bff;

color: #fff;

cursor: pointer;

transition: background 0.3s ease;

}

button:hover {

background: #0056b3;

}

button:active {

background: #003f7f;

}

=========================================================================

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 = newDog('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!

}

// Runthe 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 1 -->

<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 2 -->

<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 3 -->

<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 4 -->

<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 5 -->

<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

Document

====================================================

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

<?xmlversion="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..write a java program to establish a connection to a database and execute simple SQL queries

SQL

=====================================

package newJDBC;

import java.sql.\*;

import java.util.\*;

public class mani {

public static void main(String[] args) throws ClassNotFoundException, SQLException {

// TODO Auto-generated method stub

try {

// Class.forName("oracle.jdbc.driver.OracleDriver");

// System.out.println("driver loaded");

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

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

//Connection conn=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","mani");

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

System.out.println("connection done");

String sql="insert into studentss values(?,?,?)";

PreparedStatement pst=conn.prepareStatement(sql);

Scanner sc=new Scanner(System.in);

System.out.println("name----->");

String name=sc.nextLine();

pst.setString(1, name);

System.out.println("id----->");

Integer id=Integer.parseInt(sc.nextLine());

pst.setInt(2, id);

System.out.println("age---->");

Integer age=Integer.parseInt(sc.nextLine());

pst.setInt(3, age);

int res=pst.executeUpdate();

if(res==1) {

System.out.println("vachindhi royyy");

}

else {

System.out.println("nooooooooooo");

}

conn.close();

pst.close();

sc.close();

}

catch(ClassNotFoundException e)

{

System.out.print(e);

}

catch(SQLException e)

{

System.out.print(e);

}

}

}

===============================================================

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

===================================================================

package newJDBC;

import java.sql.\*;

import java.util.\*;

public class man {

public static void main(String[] args) {

// Database connection details

String dbUrl = "jdbc:mysql://localhost:3306/datab"; // Change as per your DB

String dbUser = "root"; // Change to your MySQL username

String dbPassword = "mani"; // Change to your MySQL password

// Establish connection to the database

try (Connection conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword)) {

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

insertData(conn);

// Demonstrating the UPDATE operation

updateData(conn);

// Demonstrating the DELETE operation

deleteData(conn);

// Demonstrating the SELECT operation

selectData(conn);

} catch (SQLException e) {

System.out.println("Database connection failed: " + e.getMessage());

}

}

// Method to insert data into the table

public static void insertData(Connection conn) {

String insertSQL = "INSERT INTO studentss (name, id, age) VALUES (?, ?, ?)";

try (PreparedStatement pst = conn.prepareStatement(insertSQL)) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter student name: ");

String name = sc.nextLine();

pst.setString(1, name);

System.out.println("Enter student ID: ");

int id = Integer.parseInt(sc.nextLine());

pst.setInt(2, id);

System.out.println("Enter student age: ");

int age = Integer.parseInt(sc.nextLine());

pst.setInt(3, age);

int rowsAffected = pst.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Data inserted successfully!");

} else {

System.out.println("Failed to insert data.");

}

} catch (SQLException e) {

System.out.println("Error executing INSERT query: " + e.getMessage());

}

}

// Method to update data in the table

public static void updateData(Connection conn) {

String updateSQL = "UPDATE studentss SET age = ? WHERE id = ?";

try (PreparedStatement pst = conn.prepareStatement(updateSQL)) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter student ID to update: ");

int id = Integer.parseInt(sc.nextLine());

System.out.println("Enter new age for the student: ");

int age = Integer.parseInt(sc.nextLine());

pst.setInt(1, age);

pst.setInt(2, id);

int rowsAffected = pst.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Data updated successfully!");

} else {

System.out.println("Failed to update data or no matching ID found.");

}

} catch (SQLException e) {

System.out.println("Error executing UPDATE query: " + e.getMessage());

}

}

// Method to delete data from the table

public static void deleteData(Connection conn) {

String deleteSQL = "DELETE FROM studentss WHERE id = ?";

try (PreparedStatement pst = conn.prepareStatement(deleteSQL)) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter student ID to delete: ");

int id = Integer.parseInt(sc.nextLine());

pst.setInt(1, id);

int rowsAffected = pst.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Data deleted successfully!");

} else {

System.out.println("No matching ID found to delete.");

}

} catch (SQLException e) {

System.out.println("Error executing DELETE query: " + e.getMessage());

}

}

// Method to select and display data from the table

public static void selectData(Connection conn) {

String selectSQL = "SELECT \* FROM studentss";

try (PreparedStatement pst = conn.prepareStatement(selectSQL);

ResultSet rs = pst.executeQuery()) {

System.out.println("Displaying all records from the 'studentss' table:");

while (rs.next()) {

System.out.println("ID: " + rs.getInt("id") + ", Name: " + rs.getString("name") + ", Age: " + rs.getInt("age"));

}

} catch (SQLException e) {

System.out.println("Error executing SELECT query: " + e.getMessage());

}

}

}

===========================================================

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

SQL

package newJDBC;

import java.sql.\*;

import java.util.\*;

public class ScrollableUpdatableResultSet {

public static void main(String[] args) {

// Database connection details

String dbUrl = "jdbc:mysql://localhost:3306/datab"; // Change as per your DB

String dbUser = "root"; // Change to your MySQL username

String dbPassword = "mani"; // Change to your MySQL password

// Establish connection to the database

try (Connection conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword)) {

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

// Create a scrollable and updatable result set

String query = "SELECT \* FROM studentss"; // Adjust this query to your database

try (Statement stmt = conn.createStatement(

ResultSet.TYPE\_SCROLL\_INSENSITIVE, // Make result set scrollable

ResultSet.CONCUR\_UPDATABLE)) { // Make result set updatable

// Execute the query

ResultSet rs = stmt.executeQuery(query);

// Demonstrate scrolling through the result set

System.out.println("Navigating through the ResultSet:");

// Move to the first record

if (rs.first()) {

System.out.println("First Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

}

// Move to the last record

if (rs.last()) {

System.out.println("Last Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

}

// Move to the previous record

if (rs.previous()) {

System.out.println("Previous Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

}

// Move to a specific record (example: 2nd row)

if (rs.absolute(2)) {

System.out.println("2nd Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

}

// Update the data in the result set (updatable result set)

rs.absolute(2); // Move to the 2nd record

System.out.println("Updating the 2nd record...");

rs.updateString("name", "Updated Name");

rs.updateInt("age", 30);

rs.updateRow(); // Commit the update

// Display updated record

System.out.println("Updated 2nd Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

// Insert a new row into the result set

rs.moveToInsertRow();

rs.updateString("name", "New Student");

rs.updateInt("id", 999);

rs.updateInt("age", 25);

rs.insertRow(); // Commit the insertion

// Move back to the last row to see the inserted row

rs.last();

System.out.println("Inserted Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

// Demonstrating the delete operation on the result set

rs.absolute(3); // Move to the 3rd row

System.out.println("Deleting 3rd Record: " + rs.getString("name") + ", " + rs.getInt("id") + ", " + rs.getInt("age"));

rs.deleteRow(); // Delete the current row

System.out.println("Deleted the 3rd Record.");

} catch (SQLException e) {

System.out.println("Error executing query or manipulating the result set: " + e.getMessage());

}

} catch (SQLException e) {

System.out.println("Database connection failed: " + e.getMessage());

}

}

}

===================================================

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

====================================================

package newJDBC;

import java.sql.\*;

public class man {

public static void main(String[] args) {

// Database connection details

String dbUrl = "jdbc:mysql://localhost:3306/datab"; // Change as per your DB

String dbUser = "root"; // Change to your MySQL username

String dbPassword = "mani"; // Change to your MySQL password

try (Connection conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword)) {

// Get Database MetaData

DatabaseMetaData dbMetaData = conn.getMetaData();

// Print basic database information

System.out.println("Database Information: ");

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

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

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

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

// Get information about the tables in the database

System.out.println("\nTables in the Database: ");

ResultSet tables = dbMetaData.getTables(null, null, "%", new String[] { "TABLE" });

while (tables.next()) {

String tableName = tables.getString("TABLE\_NAME");

System.out.println("Table Name: " + tableName);

}

// Get information about the columns in a specific table (e.g., "studentss")

System.out.println("\nColumns in the 'studentss' Table: ");

ResultSet columns = dbMetaData.getColumns(null, null, "studentss", "%");

while (columns.next()) {

String columnName = columns.getString("COLUMN\_NAME");

String columnType = columns.getString("TYPE\_NAME");

System.out.println("Column Name: " + columnName + ", Type: " + columnType);

}

// Get information about primary key of the "studentss" table

System.out.println("\nPrimary Key of the 'studentss' Table: ");

ResultSet primaryKey = dbMetaData.getPrimaryKeys(null, null, "studentss");

while (primaryKey.next()) {

String pkName = primaryKey.getString("COLUMN\_NAME");

System.out.println("Primary Key Column: " + pkName);

}

// Get information about foreign keys (if any) in the "studentss" table

System.out.println("\nForeign Keys in the 'studentss' Table: ");

ResultSet foreignKeys = dbMetaData.getExportedKeys(null, null, "studentss");

while (foreignKeys.next()) {

String fkName = foreignKeys.getString("FKCOLUMN\_NAME");

System.out.println("Foreign Key Column: " + fkName);

}

// Get supported SQL keywords

System.out.println("\nSupported SQL Keywords: ");

String supportedSQLKeywords = dbMetaData.getSQLKeywords();

System.out.println(supportedSQLKeywords);

} catch (SQLException e) {

System.out.println("Error accessing database metadata: " + e.getMessage());

}

}

}

==========================================================

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

=============================================================

==========index.html

<!DOCTYPEhtml>

<html lang="en">

<head>

<metacharset="UTF-8">

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

<title>User Form</title>

</head>

<body>

<h1>UserInformation 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=======

package com.mani;

import java.io.IOException;

import java.io.PrintWriter;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.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 ServletConig and ServletContext parameters

=========ServletConfig.java=====

package com.mani;

import java.io.IOException;

import java.io.PrintWriter;

import jakarta.servlet.ServletConfig;

import jakarta.servlet.ServletContext;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.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>");

}

}