1-10 MongoDB Practical Assignment

1. Create a Employee collection with mentioned fields Employee (eno,ename,salary,desig,dept:{deptno,deptname,location}, project:{pname,hrs})
2. Insert 10 documents in Employee collection
3. Display all the documents from Employee collection
4. Display all employees whose name starts with „S“
5. Display all Employee with the designation „Manager“
6. Display all employees with salary >50000 and salary <80000
7. Update no. of hrs to 7 for pname=_____
8. Add bonus Rs. 5000 for all employees with salary >50000 and salary <150000
9. Increase salary by 20% of employees working in deptname=_____
10. Remove all employees working on pname=_____

- **Solution-**

// **Define the Employee collection schema**
db.createCollection("Employee");
// **Insert 10 documents into the Employee collection**
db.Employee.insertMany([
{
eno: 1,
ename: "John Doe",
salary: 75000,
desig: "Software Engineer",
dept: {
deptno: 101,
deptname: "Development",
location: "New York"
},
project: {
pname: "Project A",
hrs: 160
}
},
// Add more documents here...
{
eno: 10,
ename: "Jane Smith",
salary: 80000,
desig: "Product Manager",
dept: {
deptno: 102,
deptname: "Product Management",
location: "San Francisco"
},

project: {
pname: "Project B",

hrs: 200
}
}
]);
2. Display all the documents from Employee collection
// Verify that the documents have been inserted
db.Employee.find().pretty();
4. Display all employees whose name starts with „S“
// Find employees whose names start with "S"
db.Employee.find({ "ename": { $regex: /^S/ } }).pretty();
5. Display all Employee with the designation „Manager“
// Find employees with the designation "Manager"
db.Employee.find({ "desig": "Manager" }).pretty();
6. Display all employees with salary >50000 and salary <80000
// Find employees with salary > $50,000 and salary < $80,000
db.Employee.find({
"salary": {
$gt: 50000,
$lt: 80000
}
}).pretty();
7. Update no. of hrs to 7 for pname=_____
// Update the number of hours to 7 for pname "Project B"
db.Employee.updateOne(
{ "project.pname": "Project B" },
{ $set: { "project.hrs": 7 } }
);
8. Add bonus Rs. 5000 for all employees with salary >50000 and salary
// Add a bonus of Rs. 5000 for employees with salary > Rs. 50,000 and salary < Rs. 150,000
db.Employee.updateMany(
{
"salary": {
$gt: 50000,
$lt: 150000
}
},
{
$inc: { "salary": 5000 }
}
);

9. Increase salary by 20% of employees working in deptname=_____

```
// Increase salary by 20% for employees in deptname &quot;Development&quot;
db.Employee.updateMany(
{
&quot;dept.deptname&quot;: &quot;Development&quot;
},
{
$mul: { &quot;salary&quot;: 1.2 }
}
);
```

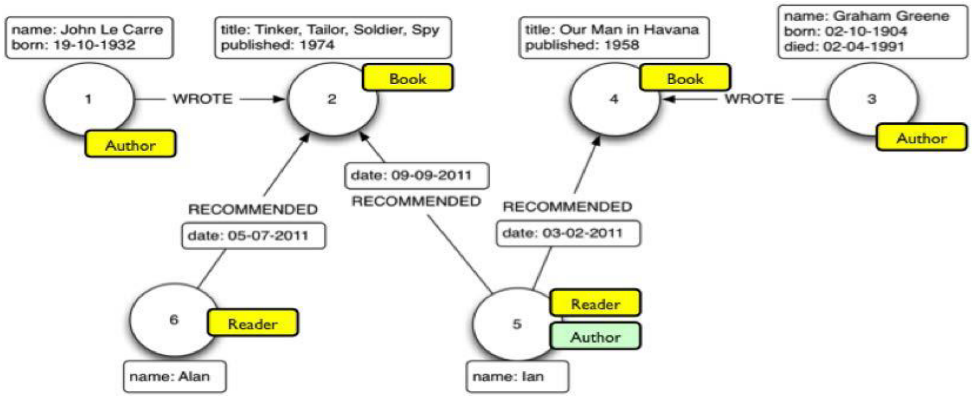
10. Remove all employees working on pname=_____

```
// Remove all employees working on a project with pname
&quot;Your_Project_Name&quot;
db.Employee.deleteMany({ &quot;project.pname&quot;:
&quot;Your_Project_Name&quot; });
```

Neo4j Assignment 2

1. Library Database :

Create a library database , as given below-



Step 1:Create Author and Reader nodes:

```
CREATE (:Author {name: "AuthorName"})
CREATE (:Reader {name: "ReaderName"})
```

Step 2:Create Book nodes:

```
CREATE (:Book {title: "BookTitle"})
```

Step 3: Create the "Author-wrote->book" relationship:

```
MATCH (a:Author {name: "AuthorName"})
MATCH (b:Book {title: "BookTitle"})
CREATE (a)-[:WROTE]->(b)
```

Step 4:Create the "Reader-recommended->book" relationship:

```
MATCH (r:Reader {name: "ReaderName"})
MATCH (b:Book {title: "BookTitle"})
CREATE (r)-[:RECOMMENDED]->(b)
```

Queries:

1. **List all reader, who have read a book "……"**

   ```
   MATCH (b:Book {title: "..."})<-[:ISSUED]-(r:Reader)
   RETURN r
   ```

2. **Count the number of reader who have read " …."**

   ```
   MATCH (:Book {title: "..."})<-[:READ]-(r:Reader)
   RETURN COUNT(r) AS numberOfReaders
   ```

3. **Add a property “Number of books read “ for Mr. Joshi and set its value as the count**

```
MATCH (r:Reader {name: "Mr. Joshi"})-[r:READ]->()
SET r.`Number of books read` = SIZE((r)-[:READ]->())
RETURN r
```

4. **List the names of readers from pune city..**

```
MATCH (r:Reader)
WHERE r.city = "Pune"
RETURN r.name
```

5. **List all readers who have recommended either book “…” or “……..” or “”**

```
MATCH (r:Reader)-[:RECOMMENDED]->(b:Book)
WHERE b.title IN ["Book1", "Book2", "Book3"]
RETURN DISTINCT r.name
```

6. **List the readers who haven‟t recommended any book**

```
MATCH (r:Reader)
WHERE NOT (r)-[:RECOMMENDED]->(:Book)
RETURN r.name
```

7. **List the authors who have written a book that has been read by maximum numberof readers.**

```
MATCH (a:Author)-[:WROTE]->(b:Book)<-[:READ]-(r:Reader)
WITH a, b, COUNT(r) AS readerCount
ORDER BY readerCount DESC
LIMIT 1
RETURN a.name AS authorName, b.title AS bookTitle, readerCount
```

8. **List the names of books recommended by “..... ” And read by at least one reader**

```
MATCH (r:Reader {name:
"..."})-[:RECOMMENDED]->(b:Book)<-[:READ]-(reader:Reader)
RETURN DISTINCT b.title
```

9. **List the names of books recommended by “..... ” and read by the maximum number of readers**.

```
MATCH (r:Reader {name:
"..."})-[:RECOMMENDED]->(b:Book)<-[:READ]-(reader:Reader)
WITH b, COUNT(DISTINCT reader) AS readerCount
ORDER BY readerCount DESC
LIMIT 1
RETURN b.title AS recommendedBook, readerCount AS numberOfReaders
```

10. **List the names of readers who haven‟t read any books written by authors fromPune and Mumbai.**

```
MATCH (r:Reader)
WHERE NOT EXISTS {
  MATCH (r)-[:READ_BY]->(:Book)-[:WRITTEN_BY]->(a:Author)
  WHERE a.location IN ['Pune', 'Mumbai']
}
RETURN r.name
```

11. **List the names of voracious readers in our library .....**

```
MATCH (r:Reader)-[:READ_BY]->(b:Book)
WITH r, COUNT(b) AS booksRead
WHERE booksRead > 20
RETURN r.name
```

2. Song database-

Consider a **Song database**, with labels as Artists, Song, Recording_company, Recoding_studio, song author etc.

Relationships can be as follows

1. **Artist → [Performs] → Song →[Written by] → Song_author.**
2. **Song → [Recorded in] → Recording Studio →[managed by] → Recording Company**
3. **Recording Company → [Finances] → Song**

You may add more labels and relationships and their properties, as per assumptions.

1. Artist → [Performs] → Song →[Written by] → Song_author.

```
CREATE (artist:Artist {name: "ArtistName"})

CREATE (song:Song {title: "SongTitle"})

CREATE (song_author:Song_author {name: "AuthorName"})

MATCH (artist:Artist {name: "ArtistName"})
MATCH (song:Song {title: "SongTitle"})
CREATE (artist)-[:Performs]->(song)

MATCH (song:Song {title: "SongTitle"})
MATCH (song_author:Song_author {name: "AuthorName"})
CREATE (song)-[:Written_by]->(song_author)

MATCH (artist:Artist)-[:Performs]->(song:Song)
RETURN artist, song

MATCH (song:Song)-[:Written_by]->(song_author:Song_author)
RETURN song, song_author
```

```
MATCH
(artist:Artist)-[:Performs]->(song:Song)-[:Written_by]->(song_author:Song_author)
RETURN artist, song, song_author
```

2. Song → [Recorded in] → Recording Studio →[managed by] → Recording Company

Step 1: Create Nodes

```
CREATE (:Song {title: "Song Title"})
CREATE (:RecordingStudio {name: "Studio Name"})
CREATE (:RecordingCompany {name: "Company Name"})
```

Step 2: Retrieve all nodes of a specific label:

1. To retrieve all `Song` nodes:

```
MATCH (song:Song)
RETURN song;
```

2. To retrieve all `Recording Studio` nodes:

```
MATCH (studio:RecordingStudio)
RETURN studio;
```

3. To retrieve all `Recording Company` nodes:

```
MATCH (company:RecordingCompany)
RETURN company;
```

Step 3: Retrieve nodes by their properties:

1. To find a specific song by its title:

```
MATCH (song:Song {title: "Song Title"})
RETURN song;
```

2. To find a specific recording studio by its name:

```
MATCH (studio:RecordingStudio {name: "Studio Name"})
RETURN studio;
```

3. To find a specific recording company by its name:

```
MATCH (company:RecordingCompany {name: "Company Name"})
RETURN company;
```

Step 4: Create Relationships

```
MATCH (song:Song {title: "Song Title"})
MATCH (studio:RecordingStudio {name: "Studio Name"})
MATCH (company:RecordingCompany {name: "Company Name"})
```

CREATE (song)-[:Recorded_in]->(studio)
CREATE (studio)-[:Managed_by]->(company)
Step 3:Retrieve specific relationships:

MATCH(song:Song)-[:Recorded_in]->(studio:RecordingStudio)-[:Managed_by]->(comp
any:RecordingCompany)

RETURN song, studio, company;

3. Recording Company → [Finances] → Song

Step 1: Create a Recording Company Node
CREATE (:RecordingCompany {name: "Company Name"})

Step 2: Create a Song Node
CREATE (:Song {title: "Song Title"})

Step 3: Create the "Finances" Relationship
MATCH (company:RecordingCompany {name: "Company Name"})
MATCH (song:Song {title: "Song Title"})
CREATE (company)-[:Finances]->(song)

Step 4: Verify the Relationship
MATCH (company:RecordingCompany)-[:Finances]->(song:Song)
RETURN company, song;

Queries:

1. **List the names of songs written by ":....."**

match(a:song)-[r:writtenby]->(b:song_author{name:"raj"}) return a.name

2. **List the names of record companies who have financed for the song "...."**

match(a:recording_company)-[r:finances]->(b:song{name:"duaa"}) return a.name

3. **List the names of artist performing the song "....."**

match(a:artist)-[r:performs]->(b:song{name:"duaa"}) return a.name

4. **Name the songs recorded by the studio " ....."**

match(a:song)-[r:recordedin]->(b:rec
ording_studio{name:"abcd"}) return a.name

14-18 Web Technology Assignment

14.Create an HTML5 program for the following input type

a. Date time

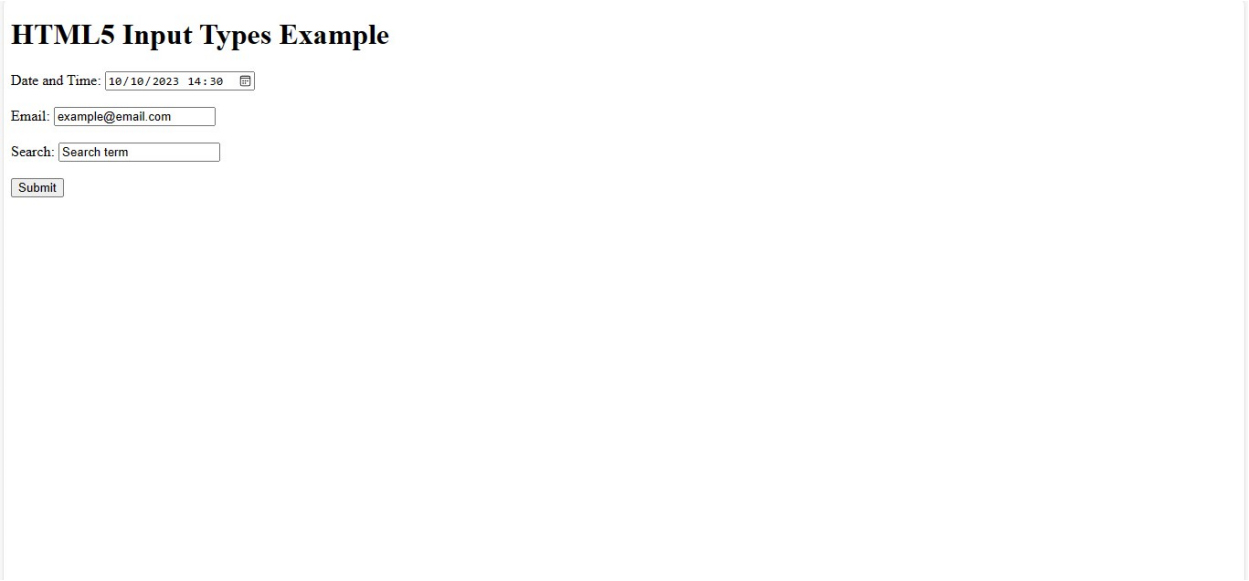
b. email input type

c. search input type

- **Solution-**

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 Input Types Example</title>
</head>
<body>
  <h1>HTML5 Input Types Example</h1>

  <!-- Date Time Input -->
  <label for="datetime">Date and Time:</label>
  <input type="datetime-local" id="datetime" name="datetime"
value="2023-10-10T14:30">
  <br><br>

  <!-- Email Input -->
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" value="example@email.com">
  <br><br>

  <!-- Search Input -->
  <label for="search">Search:</label>
  <input type="search" id="search" name="search" value="Search term">
  <br><br>

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

OUTPUT-


HTML5 Input Types Example
Date and Time: 10/10/2023 14:30
Email: example@email.com
Search: Search term
Submit

15.Write an HTML 5 program for student registration for college admission.

- **Solution-**

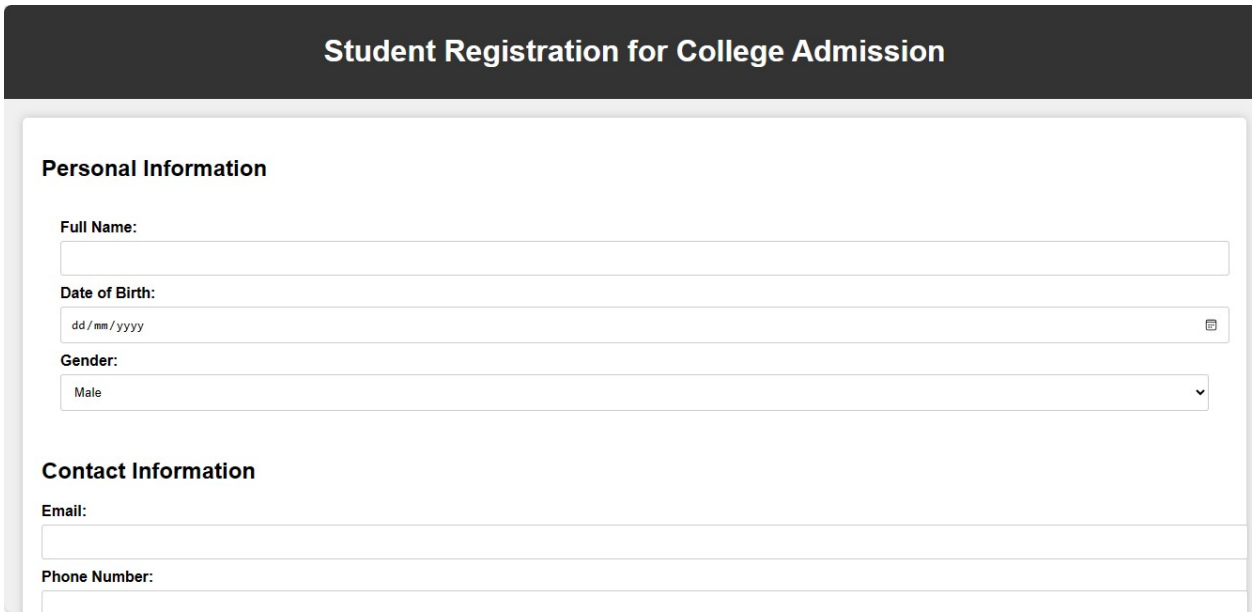
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Registration for College Admission</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #333;
            color: #fff;
            text-align: center;
            padding: 10px;
        }
        footer {
            background-color: #333;
            color: #fff;
            text-align: center;
            padding: 10px;
        }
        article {
            background-color: #fff;
            padding: 20px;
            margin: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
        }
        section {
            margin-bottom: 20px;
        }
        aside {
            background-color: #f5f5f5;
            padding: 10px;
```

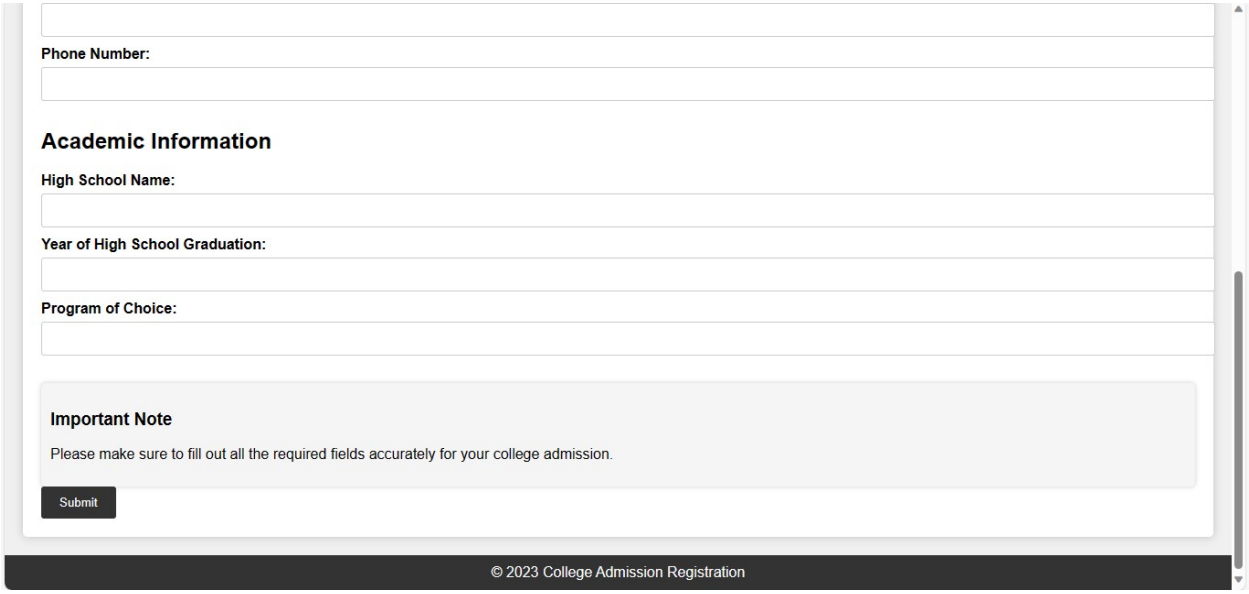
```
            border-radius: 5px;



 box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
        }
        form {
            padding: 20px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }
        input[type="text"],
        input[type="date"],
        input[type="email"],
        select,
        input[type="tel"],
        input[type="number"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 3px;
        }
        input[type="submit"] {
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
            border: none;
            border-radius: 3px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #555;
        }
    </style>
</head>
<body>
    <header>
        <h1>Student Registration for College Admission</h1>
```

```
    </header>

<article>
        <section>
            <h2>Personal Information</h2>
            <form action="process_registration.php" method="post">
                <label for="full_name">Full Name:</label>
                <input type="text" id="full_name" name="full_name" required>

                <label for="dob">Date of Birth:</label>
                <input type="date" id="dob" name="dob" required>

                <label for="gender">Gender:</label>
                <select id="gender" name="gender">
                    <option value="male">Male</option>
                    <option value="female">Female</option>
                    <option value="other">Other</option>
                </select>
            </section>

            <section>
                <h2>Contact Information</h2>
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>

                <label for="phone">Phone Number:</label>
                <input type="tel" id="phone" name="phone" required>
            </section>

            <section>
                <h2>Academic Information</h2>
                <label for="high_school">High School Name:</label>
                <input type="text" id="high_school" name="high_school" required>

                <label for="grad_year">Year of High School Graduation:</label>
                <input type="number" id="grad_year" name="grad_year" required>

                <label for="program_choice">Program of Choice:</label>
                <input type="text" id="program_choice" name="program_choice" required>
```

```
        </section>


  <aside>
            <h3>Important Note</h3>
            <p>Please make sure to fill out all the required fields accurately for your
college admission.</p>
        </aside>

        <input type="submit" value="Submit">
      </form>
    </article>

    <footer>
      &copy; 2023 College Admission Registration
    </footer>
</body>
</html>
```

OutPut




Phone Number:
Academic Information
High School Name:
Year of High School Graduation:
Program of Choice:
Important Note
Please make sure to fill out all the required fields accurately for your college admission.
Submit
© 2023 College Admission Registration

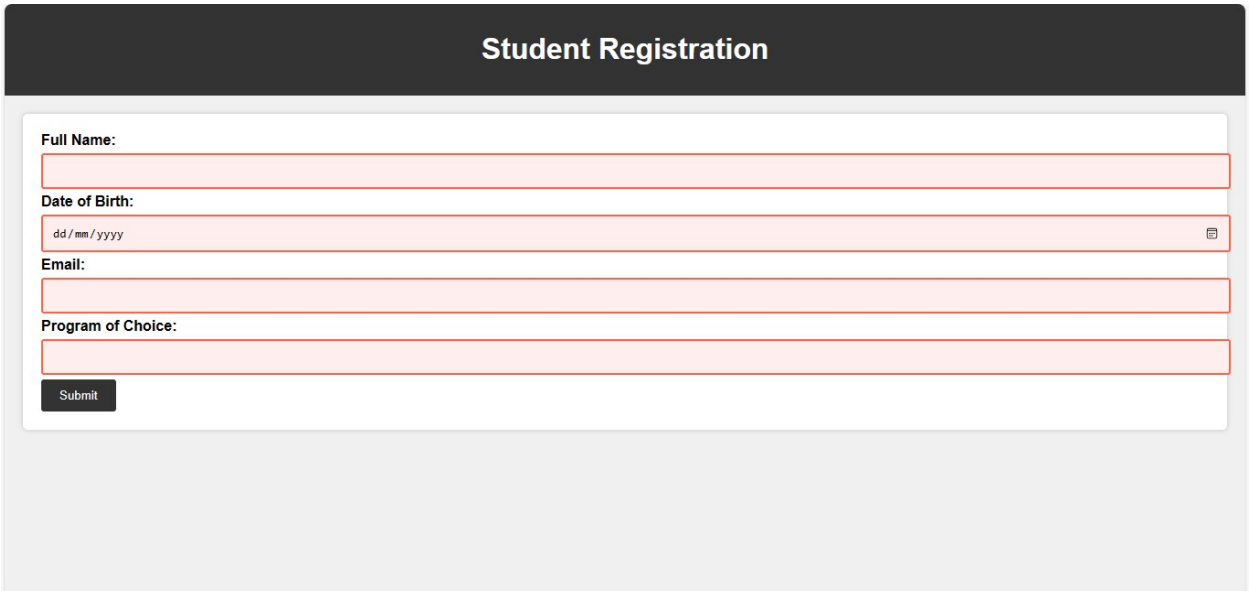
16.Write a css3 script for the above student registration form e.g. high lite compulsory fields in a different color

- **Solution-**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Registration</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #333;
            color: #fff;
            text-align: center;
            padding: 10px;
        }
        form {
            background-color: #fff;
            margin: 20px;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
        }
        label {
            font-weight: bold;
        }
        input[type="text"],
        input[type="date"],
        input[type="email"],
        select {
            width: 100%;
            padding: 10px;
            margin: 5px 0;
            border: 1px solid #ccc;
            border-radius: 3px;
```

```
        }
        input[required], /* Highlight required fields */
        select[required] {
            border: 2px solid #ff6347; /* Red border for required fields */
            background-color: #ffeeee; /* Light red background for required fields */
        }
        input[type="submit"] {
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
            border: none;
            border-radius: 3px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #555;
        }
    </style>
</head>
<body>
    <header>
        <h1>Student Registration</h1>
    </header>
    <form action="process_registration.php" method="post">
        <label for="full_name">Full Name:</label>
        <input type="text" id="full_name" name="full_name" required>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" required>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
            <label for="program_choice">Program of Choice:</label>
        <input type="text" id="program_choice" name="program_choice" required>
            <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Output


Student Registration
Full Name:
Date of Birth:
dd/mm/yyyy
Email:
Program of Choice:
Submit

17. Write a bootstrap program for the following "The .table class adds basic styling (light padding and only horizontal dividers) to a table" The table can have the first name, last name, and email id as columns.

- **Solution-**

```
<!DOCTYPE html>
<html>
<head>
  <title>Bootstrap Table Example</title>
  <!-- Include Bootstrap CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
  <div class="container">
    <h2>Bootstrap Table Example</h2>
    <table class="table">
      <thead>
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email ID</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>John</td>
          <td>Doe</td>
          <td>john.doe@example.com</td>
        </tr>
        <tr>
          <td>Jane</td>
          <td>Smith</td>
          <td>jane.smith@example.com</td>
        </tr>
        <tr>
          <td>Bob</td>
          <td>Johnson</td>
          <td>bob.johnson@example.com</td>
        </tr>
      </tbody>
```

```
    </table>
  </div>

  <!-- Include Bootstrap JS (optional) -->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></scrip
t>
</body>
</html>
```

OUTPUT-

Table with Bootstrap Styling

First Name	Last Name	Email ID	
John	Doe	johndoe@example.com	
Jane	Smith	janesmith@example.com	
Tom	Wilson	tomwilson@example.com	

18.Write a bootstrap application to display thumbnails of the images.

```
<!DOCTYPE html>
<html>
<head>
   <title>Image Thumbnails</title>
   <!-- Include Bootstrap CSS -->
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
   <div class="container">
      <h2>Image Thumbnails</h2>
      <div class="row">
         <div class="col-md-4">
            <div class="thumbnail">
               <a href="#">
                  <img src="image1.jpg" alt="Image 1">
                  <div class="caption">
                     <h3>Image 1</h3>
                     <p>Description of Image 1</p>
                  </div>
               </a>
            </div>
         </div>
         <div class="col-md-4">
            <div class="thumbnail">
               <a href="#">
                  <img src="image2.jpg" alt="Image 2">
                  <div class="caption">
                     <h3>Image 2</h3>
                     <p>Description of Image 2</p>
                  </div>
               </a>
            </div>
         </div>
         <div class="col-md-4">
            <div class="thumbnail">
               <a href="#">
                  <img src="image3.jpg" alt="Image 3">
                  <div class="caption">
                     <h3>Image 3</h3>
                     <p>Description of Image 3</p>
                  </div>
               </a>
            </div>
         </div>
```

```
  </div>
   </div>

   <!-- Include Bootstrap JS (optional) -->
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
   <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```