MongoDB Practical Assignment:1-10

Q.1.Create a Employee collection with mentioned fields Employee (eno,ename,salary,desig,dept:{deptno,deptname,location}, project:{pname,hrs})

Query:- db.Employee.insertMany({})

Q.2. Insert 10 documents in Employee collection.

Query:-

db.employee.insertMany([

{

"eno": 1,

"ename": "Sam",

"salary": 60000,

"desig": "Manager",

"dept": {

"deptno": 101,

"deptname": "Sales",

"location": "New York"

},

"project": [

{"pname": "ProjectA", "hrs": 5},

{"pname": "ProjectB", "hrs": 6}

]

},

{

"eno": 2,

"ename": "Alice",

"salary": 70000,

"desig": "Senior Developer",

"dept": {

"deptno": 102,

"deptname": "Engineering",

"location": "San Francisco"

},

"project": [

{"pname": "ProjectC", "hrs": 7},

{"pname": "ProjectD", "hrs": 8}

]

},

{

"eno": 3,

"ename": "Bob",

"salary": 50000,

"desig": "Junior Developer",

"dept": {

"deptno": 103,

"deptname": "Engineering",

"location": "San Francisco"

},

"project": [

{"pname": "ProjectE", "hrs": 4}

]

},

{

"eno": 4,

"ename": "Carol",

"salary": 65000,

"desig": "Marketing Specialist",

"dept": {

"deptno": 104,

"deptname": "Marketing",

"location": "Chicago"

},

"project": [

{"pname": "ProjectF", "hrs": 6},

{"pname": "ProjectG", "hrs": 5}

]

},

{

"eno": 5,

"ename": "Dave",

"salary": 55000,

"desig": "Sales Executive",

"dept": {

"deptno": 101,

"deptname": "Sales",

"location": "New York"

},

"project": [

{"pname": "ProjectH", "hrs": 6}

]

},

{

"eno": 6,

"ename": "Eve",

"salary": 72000,

"desig": "Product Manager",

"dept": {

"deptno": 105,

"deptname": "Product",

"location": "Boston"

},

"project": [

{"pname": "ProjectI", "hrs": 8},

{"pname": "ProjectJ", "hrs": 7}

]

},

{

"eno": 7,

"ename": "Frank",

"salary": 58000,

"desig": "Data Analyst",

"dept": {

"deptno": 106,

"deptname": "Data Science",

"location": "Seattle"

},

"project": [

{"pname": "ProjectK", "hrs": 5},

{"pname": "ProjectL", "hrs": 4}

]

},

{

"eno": 8,

"ename": "Grace",

"salary": 62000,

"desig": "HR Specialist",

"dept": {

"deptno": 107,

"deptname": "Human Resources",

"location": "Atlanta"

},

"project": [

{"pname": "ProjectM", "hrs": 6}

]

},

{

"eno": 9,

"ename": "Hank",

"salary": 53000,

"desig": "Graphic Designer",

"dept": {

"deptno": 108,

"deptname": "Design",

"location": "Los Angeles"

},

"project": [

{"pname": "ProjectN", "hrs": 7}

]

},

{

"eno": 10,

"ename": "Ivy",

"salary": 69000,

"desig": "Systems Analyst",

"dept": {

"deptno": 109,

"deptname": "IT",

"location": "Dallas"

},

"project": [

{"pname": "ProjectO", "hrs": 5},

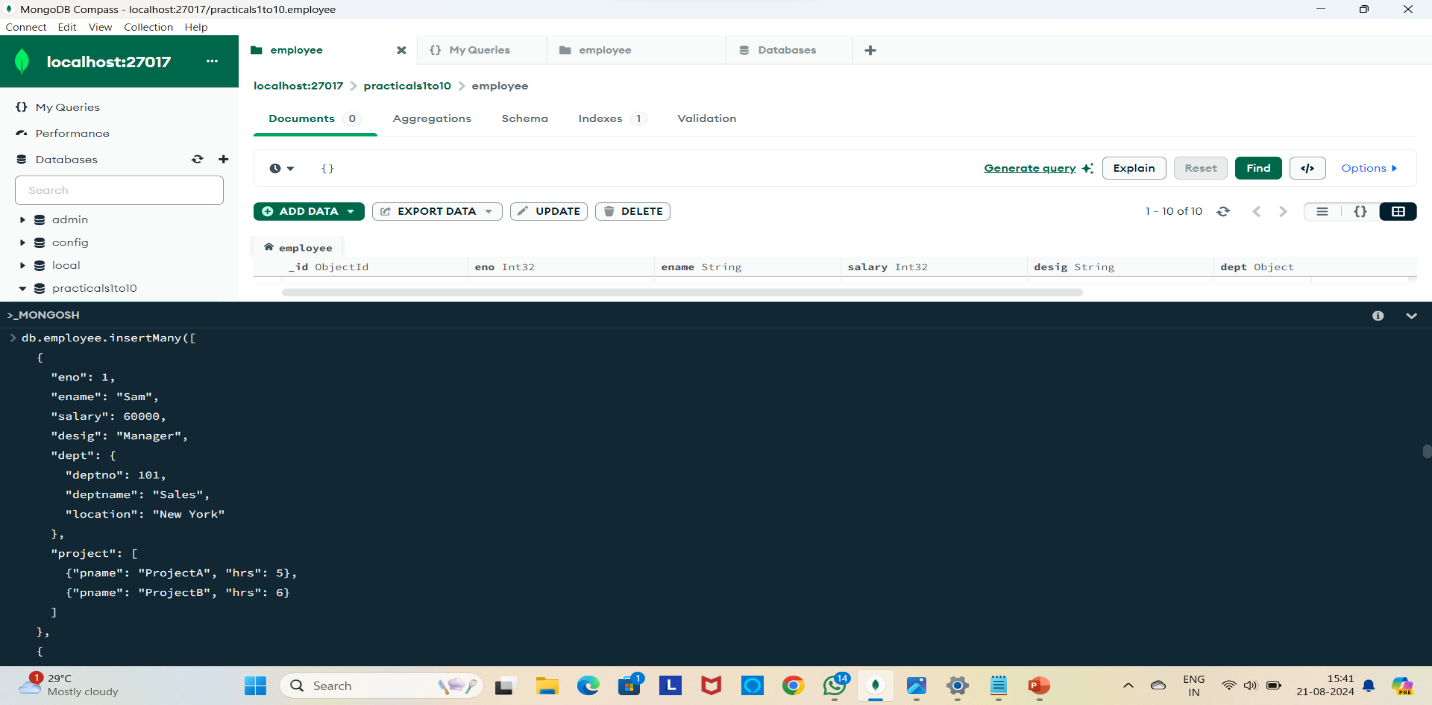
{"pname": "ProjectP", "hrs": 6}

]

}

]);

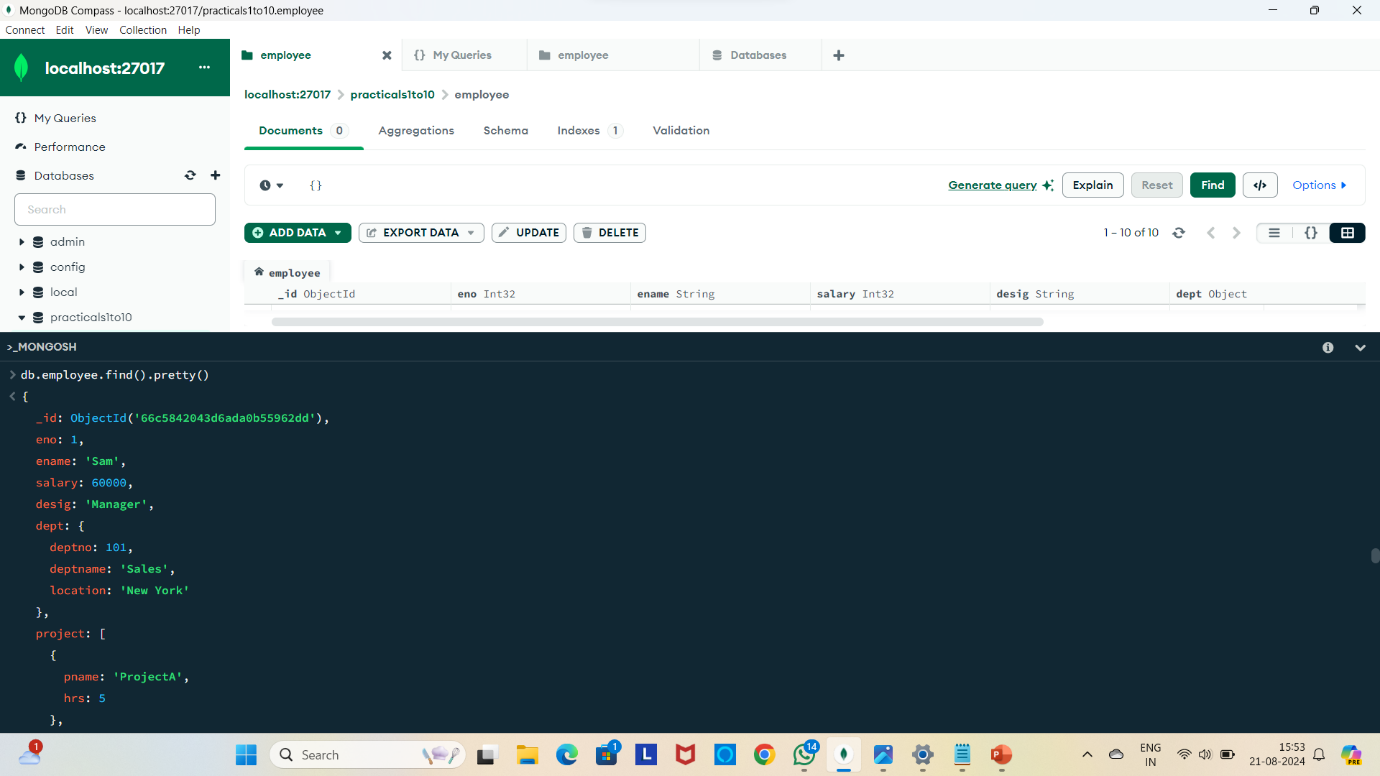
Output:-



Q.3. Display all the documents from Employee collection.

Query:- db.employee.find().pretty()

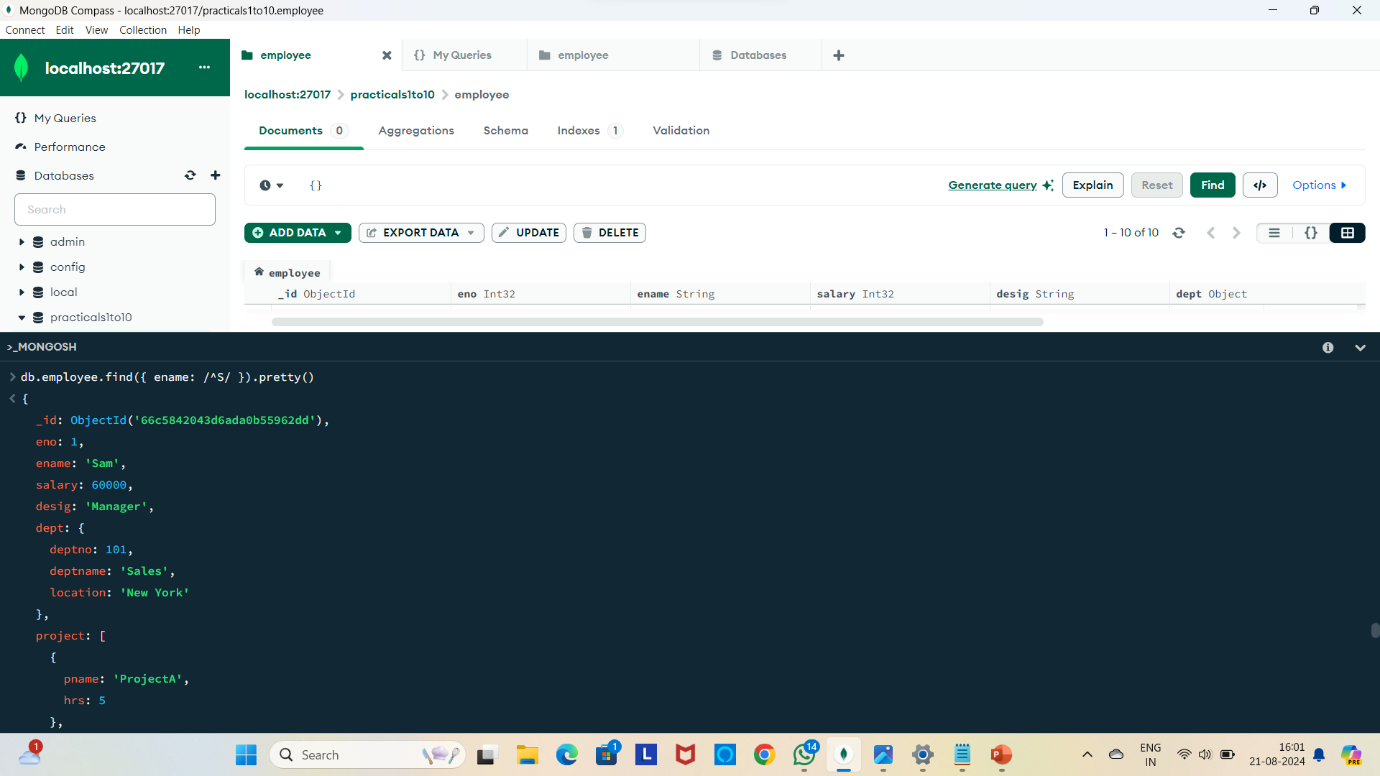
Output:-



Q.4. Display all employees whose name starts with “S”

Query:- db.employee.find({ ename: /^S/ }).pretty()

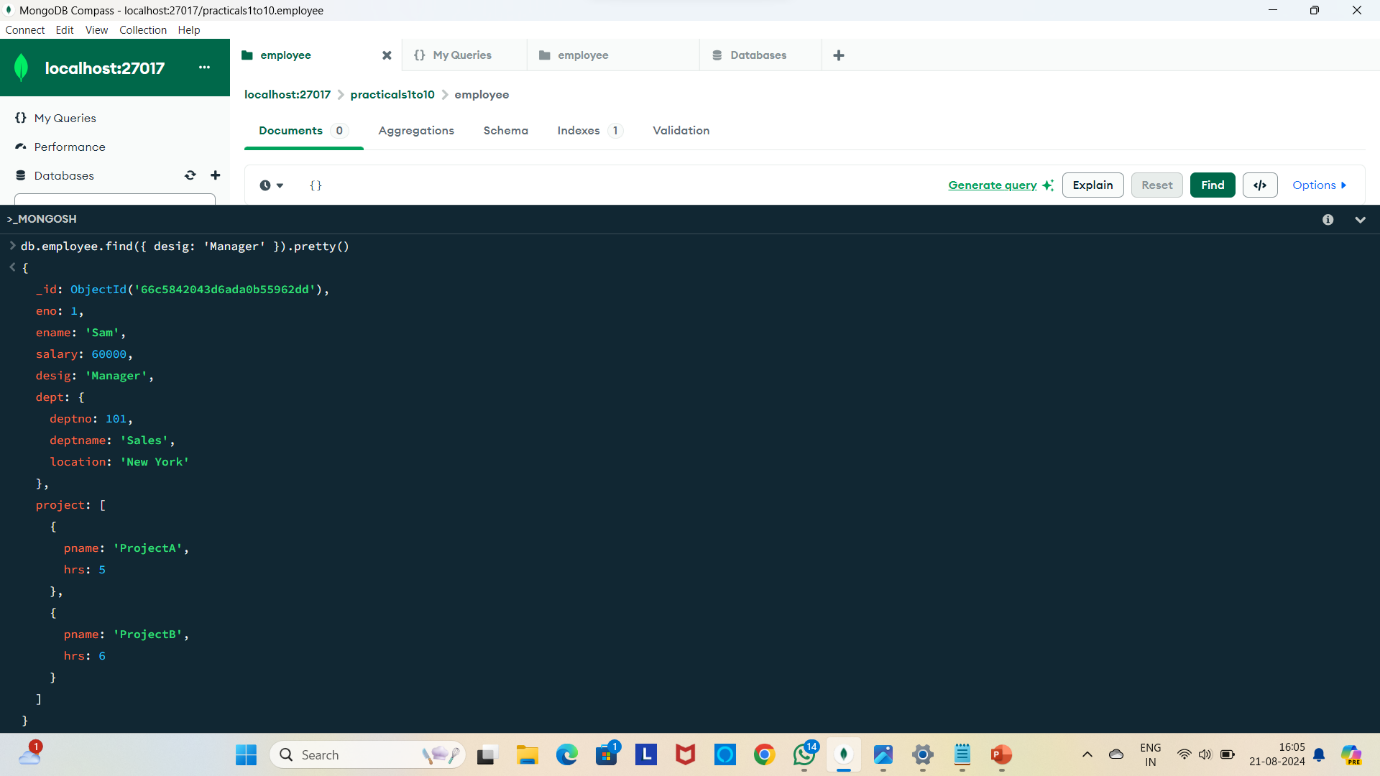
Output:-



Q.5.Display all employee with designation manager.

Query:-db.employee.find({ desig: 'Manager' }).pretty()

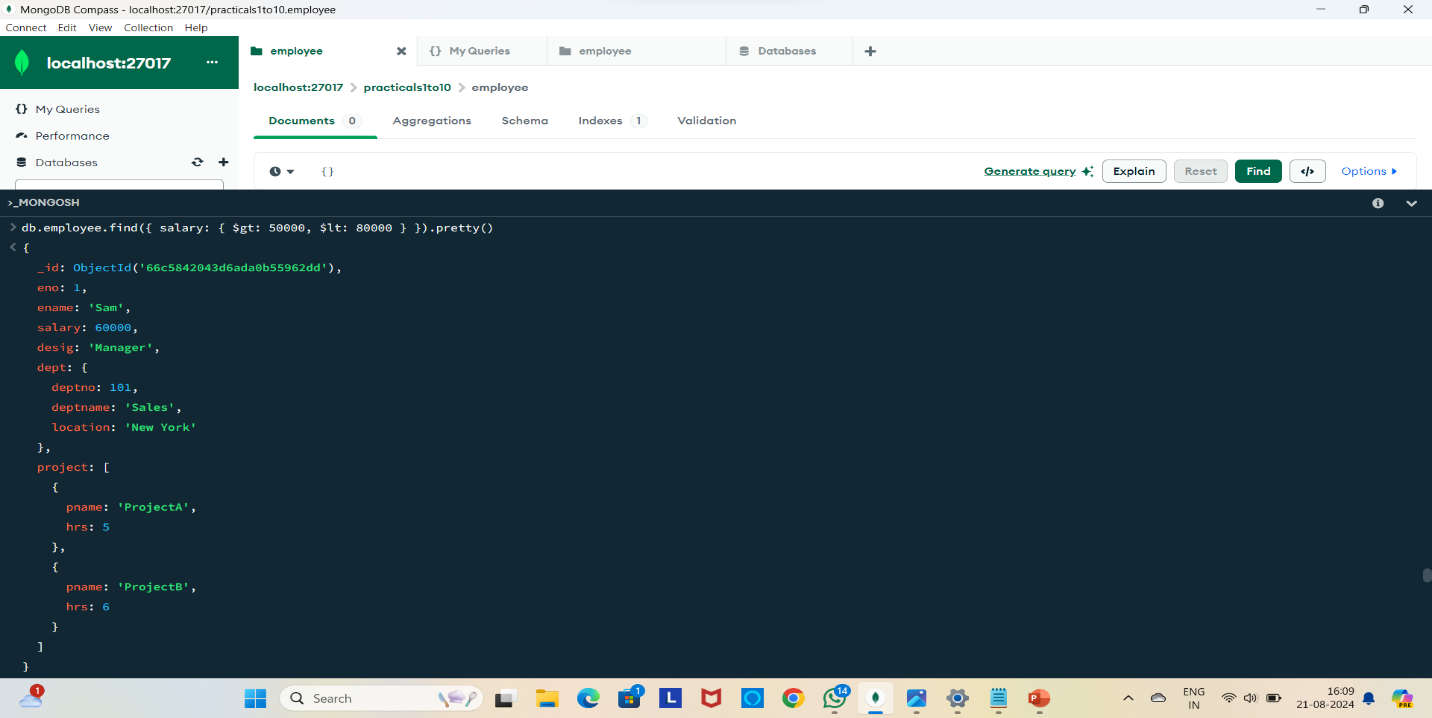
Output:-

****

Q.6. Display all employee with salary>50000 and salary<80000.

Query:- db.employee.find({ salary: { $gt: 50000, $lt: 80000 } }).pretty()

Output:-



Q.7.Update no. of hrs to 7 for pname=\_\_\_\_\_\_

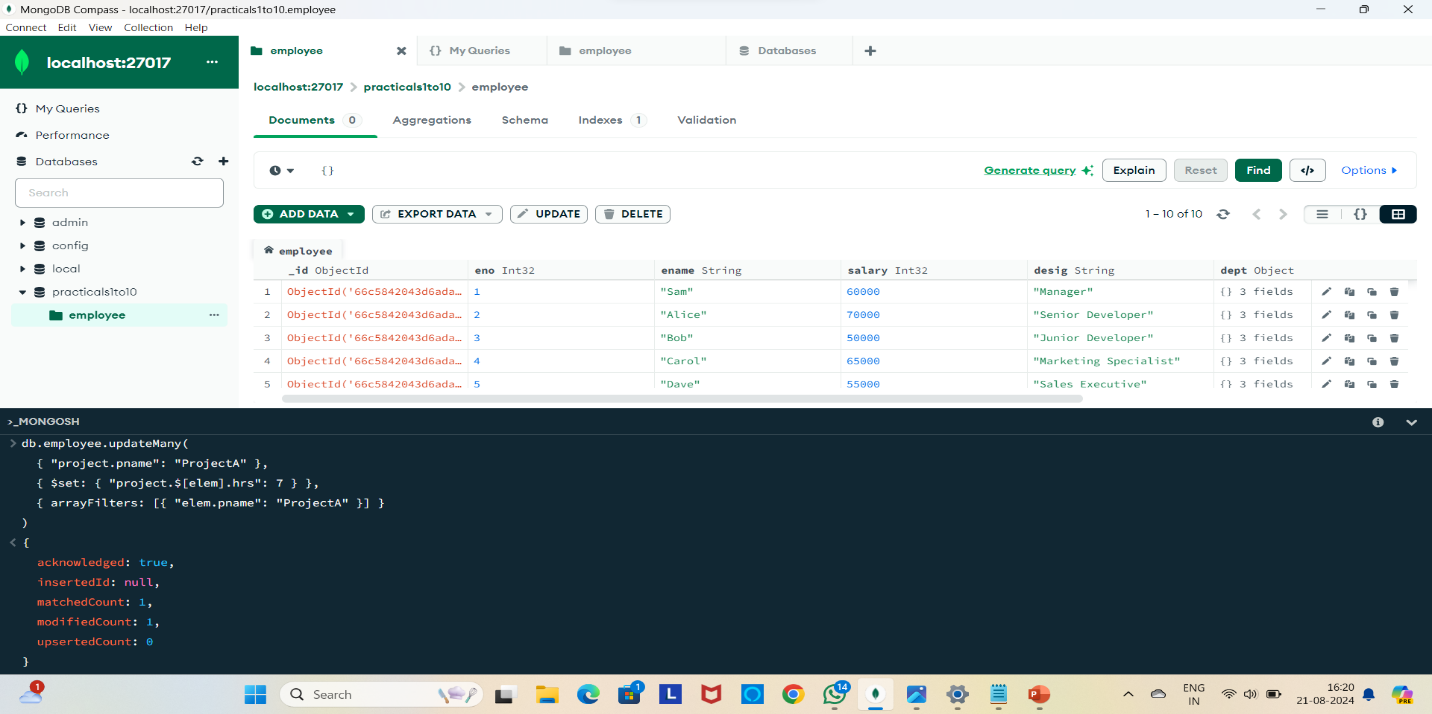
Query:-db.employee.updateMany(

{ "project.pname": "ProjectA" },

{ $set: { "project.$[elem].hrs": 7 } },

{ arrayFilters: [{ "elem.pname": "ProjectA" }])

Output:-



Q.8. . Add bonus Rs. 5000 for all employees with salary >50000 and salary <150000.

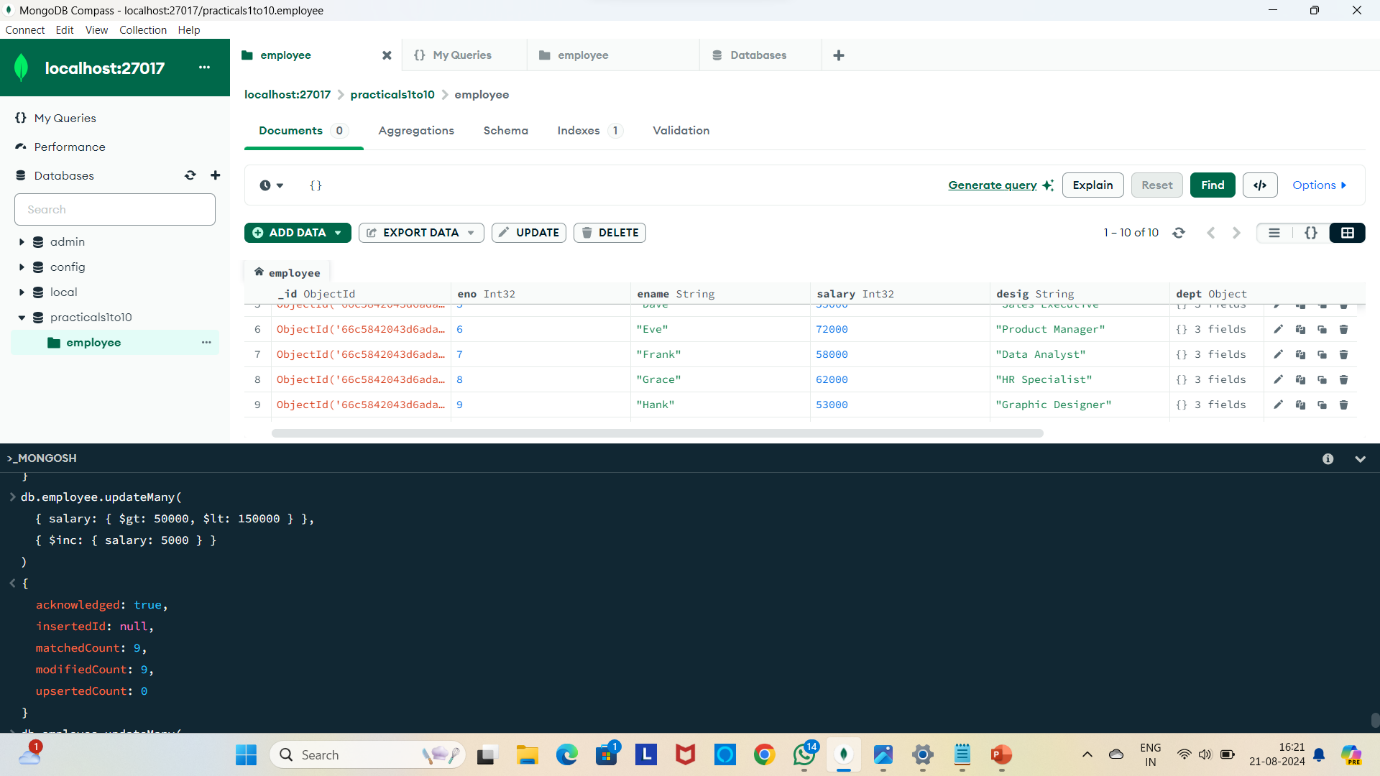
Query:-db.employee.updateMany(

{ salary: { $gt: 50000, $lt: 150000 } },

{ $inc: { salary: 5000 } }

)

Output:-



Q.9. Increase salary by 20% of employees working in deptname=\_\_\_\_\_\_\_\_

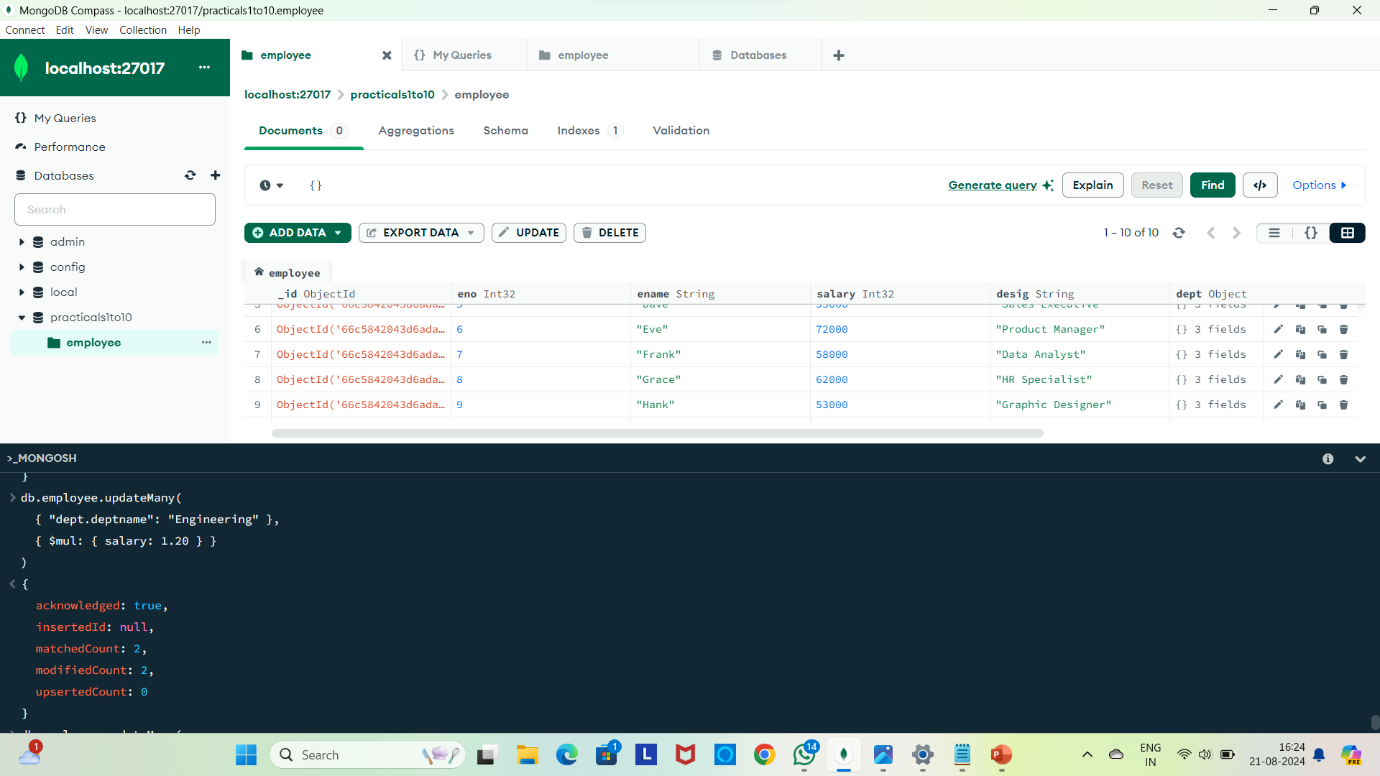
Query:-db.employee.updateMany(

{ "dept.deptname": "Engineering" },

{ $mul: { salary: 1.20 } }

)

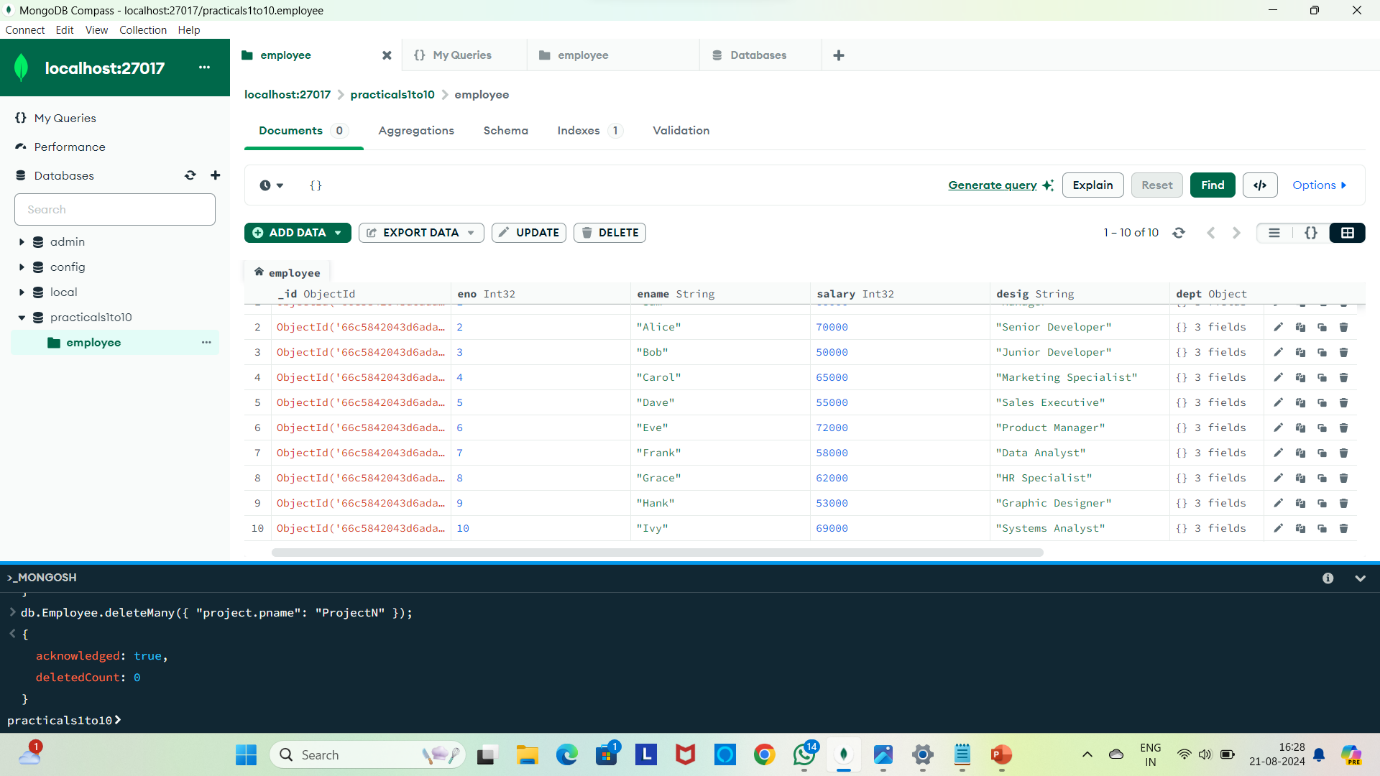
Output:-



Q.10. Remove all employees working on pname=\_\_\_\_\_\_\_\_

Query: db.Employee.deleteMany([“project.pname”:”ProjectN”]);

Output:-



**Neo4j Practical Assignment**

Below are the Cypher queries you can use for the given Neo4j practical assignment. These queries assume you have nodes and relationships in your Neo4j database representing people, books, publishers, songs, record companies, artists, and studios.

**11. Library Database**

**i. List all people who have issued a book with title “……”**

MATCH (person:Person)-[:ISSUED]->(book:Book {title: "……"})

RETURN person.name;

**ii. Count the number of people who have read “……”**

MATCH (person:Person)-[:READ]->(book:Book {title: "……"})

RETURN COUNT(person) AS numberOfPeople;

**iii. Add a property “Number of books issued” for Mr. Joshi and set its value as the count**

MATCH (person:Person {name: "Mr. Joshi"})-[:ISSUED]->(book:Book)

WITH person, COUNT(book) AS numberOfBooksIssued

SET person.numberOfBooksIssued = numberOfBooksIssued;

**iv. List the names of publishers from Pune city**

MATCH (publisher:Publisher {city: "Pune"})

RETURN publisher.name;

**12. Song Database**

**i. List the names of songs written by “……”**

MATCH (song:Song)-[:WRITTEN\_BY]->(writer:Artist {name: "……"})

RETURN song.title;

**ii. List the names of record companies who have financed the song “……”**

MATCH (recordCompany:RecordCompany)-[:FINANCED]->(song:Song {title: "……"})

RETURN recordCompany.name;

**iii. List the names of artists performing the song “……”**

MATCH (song:Song {title: "……"})<-[:PERFORMED\_BY]-(artist:Artist)

RETURN artist.name;

**iv. Name the songs recorded by the studio “……”**

MATCH (studio:Studio {name: "……"})<-[:RECORDED\_AT]-(song:Song)

RETURN song.title;

**13. Library Database**

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

MATCH (reader:Person)-[:RECOMMENDED]->(book:Book)

WHERE book.title IN ["…", "……..", "…"]

RETURN reader.name;

**b) List the readers who haven’t recommended any book**

MATCH (reader:Person)

WHERE NOT (reader)-[:RECOMMENDED]->(:Book)

RETURN reader.name;

**c) List the authors who have written a book that has been read/issued by the maximum number of readers**

MATCH (author:Author)-[:WRITTEN]->(book:Book)<-[:READ|ISSUED]-(reader:Person)

WITH author, COUNT(DISTINCT reader) AS readerCount

ORDER BY readerCount DESC

RETURN author.name, readerCount

LIMIT 1;

**d) List the names of books recommended by “......................” and read by at least one reader**

MATCH (recommender:Person {name: "......................"})-[:RECOMMENDED]->(book:Book)<-[:READ]-(reader:Person)

RETURN DISTINCT book.title;

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

MATCH (recommender:Person {name: "......................"})-[:RECOMMENDED]->(book:Book)<-[:READ]-(reader:Person)

WITH book, COUNT(DISTINCT reader) AS readerCount

ORDER BY readerCount DESC

RETURN book.title, readerCount

LIMIT 1;

**f) List the names of publishers who haven’t published any books written by authors from Pune and Mumbai**

MATCH (publisher:Publisher)-[:PUBLISHED]->(book:Book)<-[:WRITTEN\_BY]-(author:Author)

WHERE NOT (author)-[:FROM]->(:City {name: "Pune"}) AND NOT (author)-[:FROM]->(:City {name: "Mumbai"})

RETURN DISTINCT publisher.name;

**g) List the names of voracious readers in our library (i.e., those who have read the most books)**

MATCH (reader:Person)-[:READ]->(book:Book)

WITH reader, COUNT(DISTINCT book) AS booksRead

ORDER BY booksRead DESC

RETURN reader.name, booksRead

LIMIT 10;  // Adjust the limit as needed to find the top readers

**Web Technology Assignment**

**14.** **Create HTML5 Programs for Specific Input Types**

**a. Date Time Input Type**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Date Time Input</title>

</head>

<body>

<form>

<label for="datetime">Select Date and Time:</label>

<input type="datetime-local" id="datetime" name="datetime">

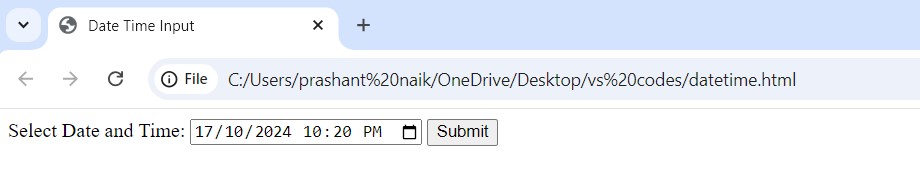
<input type="submit" value="Submit">

</form>

</body>

</html>

Output:-



**b. Email Input Type**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Email Input</title>

</head>

<body>

<form>

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

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

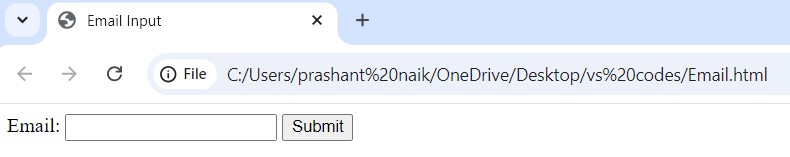
<input type="submit" value="Submit">

</form>

</body>

</html>

Output:-



**c. Search Input Type**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Search Input</title>

</head>

<body>

<form>

<label for="search">Search:</label>

<input type="search" id="search" name="search">

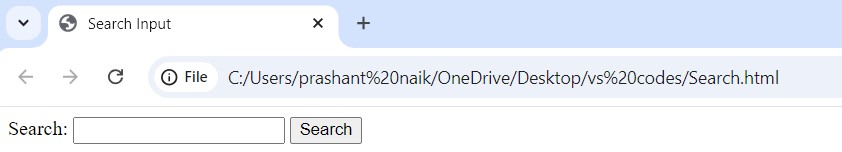
<input type="submit" value="Search">

</form>

</body>

</html>

Output:-



**15. HTML5 Program for Student Registration**

<!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>

</head>

<body>

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

<h1>Student Registration Form</h1>

<form>

<label for="firstName">First Name:</label>

<input type="text" id="firstName" name="firstName" required><br><br>

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName" required><br><br>

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required><br><br>

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

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

<label for="phone">Phone Number:</label>

<input type="tel" id="phone" name="phone"><br><br>

<label for="address">Address:</label>

<textarea id="address" name="address" rows="4" cols="50"></textarea><br><br>

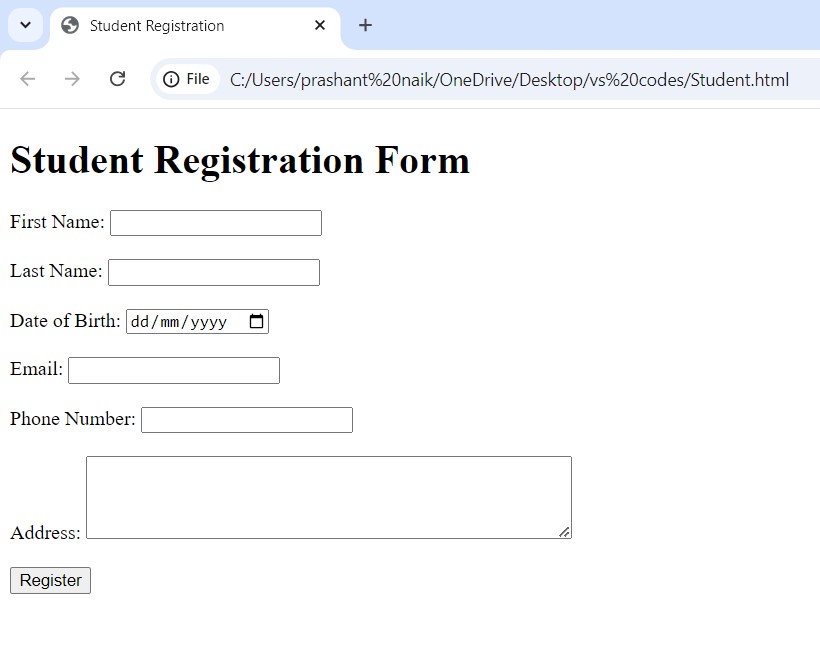
<input type="submit" value="Register">

</form>

</body>

</html>

Output:-



**16. CSS3 Script for Highlighting Compulsory Fields**

/\* Filename: styles.css \*/

body {

font-family: Arial, sans-serif;

}

form {

max-width: 600px;

margin: auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

}

label {

display: block;

margin: 10px 0 5px;

}

input[type="text"], input[type="email"], input[type="date"], input[type="tel"], textarea {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border-radius: 4px;

border: 1px solid #ccc;

}

input:required {

border: 2px solid #ff0000;

}

textarea {

resize: vertical;

}

input[type="submit"] {

background-color: #4CAF50;

color: white;

border: none;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin-top: 10px;

cursor: pointer;

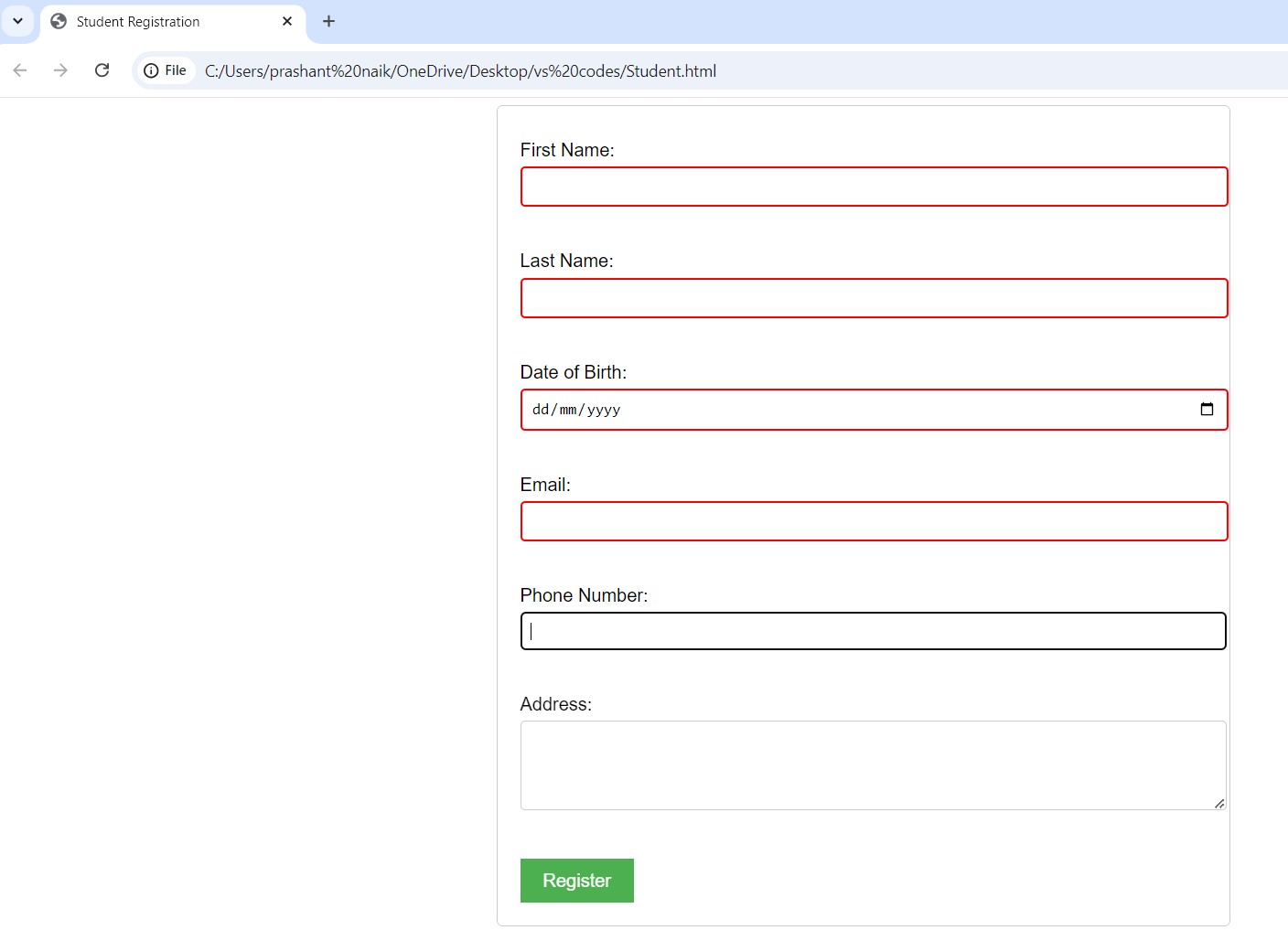
}

input[type="submit"]:hover {

background-color: #45a049;

}

Output:-



**17. Bootstrap Program for Table Styling**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Bootstrap Table</title>

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container">

<h1 class="mt-5">User Table</h1>

<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>

<!-- Add more rows as needed -->

</tbody>

</table>

</div>

<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://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>

**18. Bootstrap Application to Display Thumbnails of Images**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Image Thumbnails</title>

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container">

<h1 class="mt-5">Image Thumbnails</h1>

<div class="row">

<div class="col-md-4">

<div class="card">

<img src="https://via.placeholder.com/150" class="card-img-top" alt="Thumbnail Image">

<div class="card-body">

<p class="card-text">Description of the image.</p>

</div>

</div>

</div>

<div class="col-md-4">

<div class="card">

<img src="https://via.placeholder.com/150" class="card-img-top" alt="Thumbnail Image">

<div class="card-body">

<p class="card-text">Description of the image.</p>

</div>

</div>

</div>

<!-- Add more thumbnails as needed -->

</div>

</div>

<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://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>