Write the HTML5 code for generating the form as shown below. Apply the internal CSS to the following form to change the font size of the heading to 6pt and change the color to red and also change the background color to yellow.

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title>Project Management</title>

</head>

<style>

   h2 {

   font-family: Arial;

   font-size: 25px;

   font-style: normal;

   font-weight: bold;

   color: black;

   text-align: center;

   text-decoration: underline;

}

table {

   background-color: rgb(107, 250, 243);

   font-size: 18px;

   font-weight: bold;

   font-family: Arial, Helvetica, sans-serif;

   border-radius: 4px;
```

```
}
input[type=text] {
    box-sizing: border-box;
    width: 80%;
    padding: 10px 5px;
    margin-top: 15px;
    margin-right: 15px;
    border-radius: 5px;
}
input[type=date] {
    box-sizing: border-box;
    width: 80%;
    margin: 8px 0;
    padding: 8px 5px;
    border-radius: 5px;
}
select {
    box-sizing: border-box;
    width: 80%;
    margin: 8px 0;
    padding: 10px 5px;
    border-radius: 5px;
}
textarea {
    padding: 2px;
```

```
    line-height: 1;
    border-radius: 5px;
    box-sizing: border-box;
}
.reset1 {
    width: 30%;
    background-color: #547ef3;
    color: white;
    padding: 12px 8px;
    margin-right: 50px;
    margin-left: -50px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 15px;
    margin-top: auto;
}
.reset2:hover{
        background-color: #584ca0;
     }
     button[type=submit] {
    width: 65%;
    padding: 12px 8px;
    margin-left: 80px;
    margin-top: auto;
```

```
    margin-right: -20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #33d439;
    color: white;
    font-size: 15px;
}
button[type=submit]:hover {
    background-color: #45a049;
}
.row1{
    visibility: hidden;
    display: none;
}
</style>
<body>
    <h2>Project Management</h2>
    <table cellpadding="10" cellspacing="2" align="center">
        <th class="row1">Project Name</th>
        <tr>
            <td class="col1">Project Name</td>
            <td class="col2"><input type="text" name="project name" id="project name"></td>
        </tr>
        <th class="row1">Assigned to</th>
```

```
<tr>
   <td class="col1">Assigned to</td>
   <td class="col2">
      <select name="person" id="projectAssigned">
         <option value="Er Merry Petision">Er Merry Petision</option>
         <option value="Er Deepak Bhusan">Er Deepak Bhusan</option>
         <option value="Er Robert Watson">Er Robert Watson</option>
      </select>
   </td>
</tr>
<th class="row1">Start Date</th>
<tr>
   <td class="col1">Start Date</td>
   <td class="col2"><input type="date" name="Start Date" id="SDate" class="startDate"></td>
</tr>
<th class="row1">End Date</th>
<tr>
   <td class="col1">End Date</td>
   <td class="col2"><input type="date" name="End date" id="EDate" class="endDate"></td>
</tr>

<th class="row1">Priority</th>
<tr>
   <td class="col1">Priority</td>
   <td class="col2">
```

```
            <input type="radio" name="priority" id="high" value="High"> High
            <input type="radio" name="priority" id="average" value="Average"> Average
            <input type="radio" name="priority" id="low" value="Low"> Low
          </td>
        </tr>
        <th class="row1">Description</th>
        <tr>
          <td class="col1">Description</td>
          <td class="col2"><textarea name="description" id="description" cols="35"
rows="4"></textarea></td>
        </tr>
        <tr colspan="2" align="center">
          <td>
            <button type="submit" class="submit1" >Submit</button>
          </td>
          <td>
            <button type="reset" class="reset1" >Clear</button>
          </td>
        </tr>
    </table>
</body>
</html>
```

Model the following Property system as a document database. Consider a set of Property, Owner. One owner can buy many properties. 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 05 documents in each collection. 4. Answer the following Queries

```
use mydb
```

mydb> db.createCollection("property")

mydb>
db.property.insertMany([{"p_id":1,"area":"mumbai","rate":150000,"o_id":101},{"p_id":2,"area":"nashik","rate":90000,"o_id":102},{"p_id":3,"area":"pune","rate":120000,"o_id":103},{"p_id":4,"area":"mumbai","rate":180000,"o_id":101},{"p_id":5,"area":"nagpur","rate":8000,"o_id":104}])

db.createCollection("owner")

mydb> db.owner.insertMany([{"o_id":101,"name":"mr.patil","property_owned":[1, 4]},{"o_id":102,"name":"mr.deshmukh","property_owned":[2]},{"o_id":103,"name":"mrs.kulkarni","property_owned":[3]},{"o_id":104,"name":"mr.sharma","property_owned":[5]}])

a. Display area wise property details

db.property.aggregate([{$group:{_id:"$area",property:{$push:"$ROOT"}}}])

b. Display property owned by 'Mr.Patil' having minimum rate

db.property.find({o_id:101}).sort({rate:1}).limit(1)

c. Give the details of owner whose property is at “Nashik”

var property = db.property.findOne({area:"nashik"})

db.owner.findOne({property_owned:property.p_id})

d. Display area of property whose rate is less than 100000.
db.property.find({"rate":{$lt: 100000}},{"area":1})

SLIP 2

Create a container add row inside it and add 3 columns inside row using BootStrap.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
 <title>Bootstrap Container with Rows and Columns</title>
</head>
```

```
<body>

<div class="container">
 <!-- Row -->
 <div class="row">
  <!-- Column 1 -->
  <div class="col-md-4">
   <div class="card">
    <div class="card-body">
     <h5 class="card-title">Column 1</h5>
     <p class="card-text">Content for column 1.</p>
    </div>
   </div>
  </div>
  <!-- Column 2 -->
  <div class="col-md-4">
   <div class="card">
    <div class="card-body">
     <h5 class="card-title">Column 2</h5>
     <p class="card-text">Content for column 2.</p>
    </div>
   </div>
  </div>
  <!-- Column 3 -->
  <div class="col-md-4">
   <div class="card">
    <div class="card-body">
     <h5 class="card-title">Column 3</h5>
     <p class="card-text">Content for column 3.</p>
    </div>
   </div>
  </div>
 </div>
 <!-- End Row -->
</div>
</body>
</html>
```

Model the following system as a document database. Consider a database of newspaper, publisher, and city. Different publisher publishes various newspapers in different cities 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries

db.createCollection("newspaper")
mydb> db.newspaper.insertMany([{"n_id":1,"name":"The
Time","language":"English","p_id":101,"city":"nashik","state":"maharashtra","sales":5000},{"n_id":2,"name":"Lakmat","language":"marathi","p_id":102,"city":"pune","state":"maharashtra","sales":75000},{"n_id":3,"name":"gujarat
samachar","p_id":103,"city":"ahmedabad","state":"gujarat","sales":60000},{"n_id":4,"name":"hindustan
times","language":"english","p_id":104,"city":"mumbai","state":"maharashtra","sales":10000},{"n_id":5,"name":"prahhar","language":"marathi","p_id":105,"city":"nagpur","state":"maharashtra","sales":40000}])

db.createCollection("publishers")
mydb> db.publishers.insertMany([{"p_id":101,"name":"time
media","state":"maharashtra"},{"p_id":102,"name":"lokmat
group","state":"maharashtra"},{"p_id":103,"name":"gujarat
publications","state":"gujarat"},{"p_id":104,"name":"HT
media","state":"maharashtra"},{"p_id":105,"name":"prahaar
publications","state":"maharashtra"}])

. a. List all newspapers available “NASHIK” city

db.newspaper.find({city:"nashik"},{_id:0,name:1,language:1})

b. List all the newspaper of “Marathi” language
db.newspaper.find({language:"marathi"},{_id:0,name:1,city:1})

c. Count no. of publishers of “Gujrat” state
db.publishers.countDocuments({state:"gujarat"})

e. Write a cursor to show newspapers with highest sale in Maharashtra State
var cursor = db.newspaper.find({state:"maharashtra"}).sort({sales:-1})

while (cursor.hasNext()) { printjson(cursor.next()); }

SLIP 3

1 ques by image thumbnail in ower lab book manual
2 . Model the following system as a document database. Consider employee and department’s information. 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

db.employees.insertMany([{"e_id":1,"name":"dhanu","salary":750000,"d_id":101},{"e_id":2,"name":"abhi","salary":80000,"d_id":102},{"mydb>
db.employees.insertMany([{"e_id":1,"name":"dhanu","salary":750000,"d_id":101},{"e_id":2,"name":"abhi","salary":80000,"d_id":102},{"e_id":3,"name":"sani","salary":700000,"d_id":103},{"e_id":4,"name":"harsh","salary":90000,"d_id":101},{"e_id":5,"name":"charli","salary":85000,"d_id":104}])
db.createCollection("departments")
mydb>
db.departments.insertMany([{"d_id":101,"name":"sales"},{"d_id":102,"name":"marketing"},{"d_id":103,"name":"finance"},{"d_id":104,"name":"engineering"},{"d_id":105,"name":"HR"}])

a. Display name of employee who has highest salary

db.employees.find().sort({sort:-1}).limit(1,{_id:0,name:1})

b. Display biggest department with max. no. of employees

c. Write a cursor which shows department wise employee information

```
var cursor = db.departments.find()

while(cursor.hasNext()){

var department = cursor.next();

var employees = db.employees.find({d_id:department.department_id}).toArray();

printjson({department: department,employees:employees});

}
```

d. List all the employees who work in Sales dept and salary > 50000
db.employees.find({d_id:101,salary:{$gt: 50000}},{_id: 0, name:1})

SLIP 4

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
By practical book

2. Model the following information system as a document database. Consider hospitals around Nashik. Each hospital may have one or more specializations like Pediatric, Gynaec, Orthopedic, etc. A person can recommend/provide review for a hospital. A doctor can give service to one or more hospitals. 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 10 documents in each collection. 4. Answer the following Queries

```
db.createCollection("hospitals")
mydb> db.hospitals.insertMany([{"h_id":1,"name":"city general
hospital","city":"nashik","specializations":["pediatric","orthopedic"],"rating":4.2},{"h_id":2,"
name":"womens care
center","city":"nashik","specializations":["gynecology","maternity"],"ratring":4.8},{"h_id":3,
"name":"orthocare
clinic","city":"nashik","specializations":["orthopedic"],"rating":4.5},{"h_id":4,"name":"eye
clinic","city":"pune","specializations":["eyes"],"rating":4.5}])

db.createCollection("doctors")
db.doctors.insertMany([{"do_id":101,"name":"dr.patil","hospitals":[1,
3]},{"do_id":102,"name":"dr.deshmukh","hospitals":[2,
3]},{"do_id":103,"name":"dr.ahire","hospitals":[2, 1]}])

db.createCollection("reviews")
db.reviews.insertMany([{"r_id":1,"h_id":1,"reviewer":"User123","rating":4},{"r_id":2,"h_id":
2,"reviewer":"User456","rating":5},{"r_id":3,"h_id":3,"reviewer":"User356","rating":3},{"r_i
d":4,"h_id":4,"reviewer":"User678","rating":4}])
```

a. List the names of hospitals with…… specialization.

```
db.hospitals.find({specializations:"orthopedic"},{_id:0,name:1})
```

b. List the Names of all hospital located in ……. City

```
db.hospitals.find({city:"nashik"},{_id:0,name:1})
```

c. List the names of hospitals where Dr. Deshmukh visits

```
var doctor = db.doctors.findOne({name:"dr.deshmukh"});

db.hospitals.find({h_id:{$in: doctor.hospitals}},{_id:0,name:1})
```

d. List the names of hospitals whose rating >=4

```
db.hospitals.find({rating:{$gte: 4}},{_id:0,name:1})
```

SLIP 5

Write a HTML code, which generate the following output [Apply border, border radius tags]

```
<html>
<head>
  <title>Table</title>
</head>
<body>
  <table border ="1">
    <tr>
      <th colspan="4">List of persons</th>
      </tr>
    <tr>
      <td width = 80>srno</td>
      <td width = 80>Person Name</td>
      <td width = 80>Age</td>
      <td width = 80>Countery</td>
    </tr>
    <tr>
      <td>1</td>
      <td>dhanshree</td>
      <td>20</td>
      <td>Pune</td>
    </tr>
    <tr>
      <td>2</td>
      <td>abhi</td>
      <td>21</td>
      <td>pune</td>
    </tr>
    <tr>
      <td>3</td>
      <td>sani</td>
      <td>21</td>
      <td>pune</td>
    </tr>
  </table>
</body>
</html>
```

Model the following database. Many employees working on one project. A company has various ongoing projects. 2. Assume appropriate attributes and collections as per the query

requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries

```
db.createCollection("project")
mydb> db.project.insertMany([{"pro_id":1,"name":"project A","project_type":"software development","duration_mounths":5},{"pro_id":2,"name":"project B","project_type":"infrastructure","duration_months":4},{"pro_id":3,"name":"project c","project_type":"marketing","duration_months":2},{"pro_id":4,"name":"project d","project_type":"cloud computing","duration_months":5},{"pro_id":5,"name":"project e","project_type":"AI","duration_months":7}])

db.createCollection("emp")
mydb> db.emp.insertMany([{"emp_id":101,"name":"mr. patil","project":[1,2]},{"emp_id":102,"name":"mr. sharma","project":[2, 3]},{"emp_id":103,"name":"miss.ahire","projects":[4, 5]},{"emp_id":104,"name":"mr.waghmode","project":[4, 5]}])
```

a. List all names of projects where Project_type =….. b. List all the projects with duration

```
db.project.find({project_type:"software development"},{_id: 0,name: 1})
```

b. greater than 3 months

```
db.project.find({duration_months:{$gt:3}},{_id: 0, name: 1})
```

c. Count no. of employees working on ……..project

```
var projectEmployeesCount = db.emp.count({project:1});

print("number of employees working on project A:"+projectEmployeesCount);
```

d. List the names of projects on which Mr. Patil is working

SLIP 6

Create a web page being rendered in the browser consists of many things - logo, informative text, pictures, hyperlinks, navigational structure and table.

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

```
<title>Your Webpage Title</title>
<style>
  /* Add some basic styling to make it visually appealing */
  body {
    font-family: Arial, sans-serif;
    margin: 20px;
  }

  header {
    text-align: center;
    padding: 10px;
    background-color: #f2f2f2;
  }

  nav {
    margin: 10px 0;
  }

  nav a {
    margin-right: 20px;
    text-decoration: none;
    color: #333;
  }

  section {
```

```
      margin-bottom: 20px;
    }

    img {
      max-width: 100%;
      height: auto;
    }

    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }

    th, td {
      border: 1px solid #ddd;
      padding: 8px;
      text-align: left;
    }

    th {
      background-color: #f2f2f2;
    }
  </style>
</head>
```

```
<body>

  <header>
    <img src="logo.png" alt="Your Logo" width="100">
    <h1>Your Website Name</h1>
  </header>

  <nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Services</a>
    <a href="#">Contact</a>
  </nav>

  <section>
    <h2>Welcome to Our Website!</h2>
    <p>This is some informative text about your website. You can add more details and make it interesting.</p>
    <img src="image.jpg" alt="Description of the image">
    <p>Learn more <a href="#">about us</a>.</p>
  </section>

  <section>
    <h2>Our Services</h2>
    <p>Here, you can provide information about the services your website offers.</p>
  </section>
```

```
<section>
  <h2>Contact Us</h2>
  <p>Feel free to <a href="#">contact us</a> for any inquiries or feedback.</p>
</section>

<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Phone</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John Doe</td>
      <td>john@example.com</td>
      <td>(555) 123-4567</td>
    </tr>
    <tr>
      <td>Jane Doe</td>
      <td>jane@example.com</td>
      <td>(555) 987-6543</td>
    </tr>
```

</tbody>

</table>

</body>

</html>

Model the following information as a document database. A customer can take different policies and get the benefit. There are different types of policies provided by various companies 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

db.createCollection("customers")

{ ok: 1 }

mydb> db.customer.insertMany([{"c_id":1,"name":"john doe","age":30,"address":"123 main st","policies":["komal jeevan","life secure","health shield"]},{"c_id":2,"name":"jane smith","age":25,"address":"456 oak st","policies":["komal jeevan","travel guared"]},{"c_id":3,"name":"bob johnson","age":40,"address":"789 pine st","policies":["life secure","accident protect","health shield"]},{"c_id":4,"name":"alice brown","age":35,"address":"101 maple st","policies":["monthly saver","retirement plan"]},{"c_id":5,"name":"charlie wilson","age":28,"address":"202 cedar st","policies":["health shield","komal jeevan"]}])

db.policies.insertMany([{"p_id":101,"pname":"komal jeevan","cname":"abc insurance","type":"yearly","premiumAmount":1200,"benefits":["death benefit","maturity benefit"]},{"p_id":102,"pname":"life secure","cname":"xyz insurance","type":"half yearly","premiumAmount":800,"benefits":["death benefit","critical illness cover"]},{"p_id":103,"pname":"health shield","cname":"pqr insurance","type":"monthly","premiumAmount":100,"benefits":["health coverage","hospitalization benefit"]},{"p_id":104,"pname":"travel guard","cname":"lmn insurance","type":"yearly","premiumAmount":500,"benefits":["travel coverage","trip cancellation"]},{"p_id":105,"pname":"monthly saver","cname":"xyz insurance","type":"monthly","premiumAmount":150,"benefits":["savings","investment growth"]}])

a. List the details of customers who have taken “Komal Jeevan” Policy
   db.customer.find({"policies":"komal jeevan"})
b. Display average premium amount
   db.policies.aggregate([{$group:{_id:null,avgPremium:{$avg:"$premiumAmount"}}}])

c. Increase the premium amount by 5% for policy type=”Monthly”
db.policies.updateMany({"type":"monthly"},{$mul:{"premiumAmount":1.05}})
d. Count no. of customers who have taken policy type “half yearly”.

SLIP 7

Create a 3D text, apply appropriate font, style, color. Use : Hover in the style selector so that the 3D effects appear only when you hover over the text

. Model the following information as a document database. A customer operates his bank account, does various transactions and get the banking services 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

```
db.createCollection("cust")
{ ok: 1 }
mydb>
db.cust.insertMany([{"cust_id":1,"fname":"john","lname":"smith","amountOpenDate":"2020-
01-01","branch":"branchA","accountType":"saving","transactions":[1,
2]},{"cust_id":2,"fname":"sarah","lname":"jones","accountOpenDate":"2020-01-
01","branch":"branchB","accountType":"loan","transactions":[3]},{"cust_id":3,"fname":"samuel
","lname":"brown","accountOpenDate":"2020-02-
15","branch":"branchaA","accountType":"saving","transactions":[4,
5]},{"cust_id":4,"fname":"sara","lname":"williams","accountOpenDate":"2020-01-
01","branch":"branchC","accountType":"loan","transactions":[6]},{"cust_id":5,"fname":"mark","
lname":"johnson","accountOpenDate":"2020-03-
10","branch":"branchB","accountType":"saving","transactions":[7, 8]}])

db.createCollection("transactions")
{ ok: 1 }
mydb> db.transactions.insertMany([{"t_id":1,"cust_id":1,mydb>
db.transactions.insertMany([{"t_id":1,"cust_id":1,mydb> db.transactions.insertMany([{"mydb>
db.transactions.insertMany([{"t_id":1,"cust_id":1,"transactionDate":"2020-01-
02","amount":500,"transactionType":"deposit"},{"t_id":2,"cust_i":1,"transactionDate":"2020-
02-
01","amount":200,"transactionType":"withdrawal"},{"t_id":3,"cust_id":2,"transactionDate":"202
0-01-
01","amount":1000,"transactionType":"loan"},{"t_id":4,"cust_id":3,"transactionDate":"2020-02-
20","amount":300,"transactionType":"deposit"},{"t_id":5,"cust_id":3,"transactionDate":"2020-
03-
01","amount":150,"transactionType":"withdrawal"},{"t_id":6,"cust_id":4,"transactionDate":"202
0-01-
```

15","amount":800,"transactionType":"loan"},{"t_id":7,"cust_id":5,"transactionDate":"2020-03-15","amount":400,"transactionType":"deposit"},{"t_id":8,"cust_id":5,"transactionDate":"2020-03-25","amount":200,"transactionType":"withdrawal"}])

a. List names of all customers whose first name starts with a “S”
   db.cust.find({"fname":/^s/i},{"fname":1,"lname":1,"_id":0})
b. List all customers who has open an account on 1/1/2020 in ___branch
   db.cust.find({"amountOpenDate":"2020-01-01","branch":"branchA"})
c. List the names customers where acctype=”Saving”
   db.cust.find({"accountType":"saving"},{"fname":1,"lname":1,"_id":0})
d. Count total no. of loan account holder of …….branch
   db.cust.count({"branch":"branchC","accountType":"loan"})

SLIP 8

Create a button with different style (Secondary, Primary, Success, Error, Info, Warning, Danger) using BootStrap

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
 <title>Bootstrap Button Styles</title>
</head>
<body>

 <div class="container mt-5">
  <h2>Bootstrap Button Styles</h2>

  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-danger">Danger</button>
 </div>
```

</body>
</html>

Model the following inventory information as a document database. The inventory keeps track of various items. The items are tagged in various categories. Items may be kept in various warehouses and each warehouse keeps track of the quantity of the item. 2. Assume appropriate attributes and collections as per the query requirements 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

db.createCollection("item")
{ ok: 1 }
mydb>
db.item.insertMany([{"i_id":1,"iname":"laptop","tags":["electronics","portable"],"quantity":400,"status":"A","height":12},{"i_id":2,"iname":"desk chair","tags":["furniture","office"],"quantity":250,"status":"B","height":15},{"i_id":3,"iname":"planner","tags":["stationery","office"],"quantity":30,"status":"C","height":8},{"i_id":4,"iname":"printer","tags":["electronics","office"],"quantity":120,"status":"A","height":10},{"i_id":5,"iname":"bookshelf","tags":["furniture","storage"],"quantity":180,"status":"B","height":72}])

db.warehouse.insertMany([{"w_id":101,"wname":"main warehouse","items":[{"i_id":1,"quantity":150},{"i_id":2,"quantity":80},{"i_id":3,"quantity":20},{"i_id":4,"quantity":70},{"i_id":5,"quantity":100}]},{"w_id":102,"wname":"bakup warehouse","items":[{"i_id":1,"quantity":250},{"i_id":2,"quantity":170},{"i_id":3,"quantitiy":10},{"i_id":4,"quantity":50},{"i_id":5,"quantity":80}]},{"w_id":103,"wname":"remote warehouse","items":[{"i_id":1,"quantity":50},{"i_id":2,"quantity":30},{"i_id":3,"quantity":0},{"i_id":4,"quantity":0},{"i_id":5,"quantity":0}]}])

a. List all the items qty is greater than 300
   db.item.find({"quantity":{$gt:300}})
b. List all items which have tags less than 5

c. List all items having status equal to “B” or having quantity less than 50 and height of the product should be greater than 8

   db.item.find({$or:[{"status":"B"},{$and:[{"quantity":{$lt:50}},{"height":{$gt:8}}]}]})

d. Find all warehouse that keeps item “Planner” and having in stock quantity less than 20
   db.warehouse.find({"items":{$elemMatch:{"i_id":3,"quantity":{$lt:20}}}})

SLIP 9

Write an HTML 5 program for student registration form for college admission. Use input type like search, email, date etc

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Student Registration Form</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   background-color: #f4f4f4;
   margin: 20px;
  }

  form {
   max-width: 600px;
   margin: 0 auto;
   background-color: #fff;
   padding: 20px;
   border-radius: 5px;
   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  h2 {
   text-align: center;
  }

  label {
   display: block;
   margin-bottom: 8px;
  }

  input, select {
   width: 100%;
   padding: 8px;
   margin-bottom: 15px;
   box-sizing: border-box;
  }

  input[type="submit"] {
```

```
    background-color: #4caf50;
    color: #fff;
    cursor: pointer;
   }

   input[type="submit"]:hover {
    background-color: #45a049;
   }
  </style>
</head>
<body>

 <form>
  <h2>Student Registration Form</h2>

  <label for="fullName">Full Name:</label>
  <input type="text" id="fullName" name="fullName" required>

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" 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>

  <label for="address">Address:</label>
  <input type="text" id="address" name="address" required>

  <label for="city">City:</label>
  <input type="text" id="city" name="city" required>

  <label for="state">State:</label>
  <input type="text" id="state" name="state" required>

  <label for="zipCode">Zip Code:</label>
  <input type="text" id="zipCode" name="zipCode" required>
```

```
  <label for="phone">Phone Number:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required>

  <label for="admissionDate">Admission Date:</label>
  <input type="date" id="admissionDate" name="admissionDate" required>

  <label for="course">Course of Interest:</label>
  <input type="search" id="course" name="course" list="courses" required>
  <datalist id="courses">
   <option value="Computer Science">
   <option value="Electrical Engineering">
   <option value="Business Administration">
   <option value="Psychology">
   <!-- Add more options as needed -->
  </datalist>

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

</body>
</html>
```

Model the following Customer Loan information as a documentdatabase. Consider Customer Loan information system where the customer can take many types of loans. 2. Assume appropriate attributes and collections as per the query requirements 3. Insert at least 10 documents in each collection. 4. Answer the following Queries.

```
([
 {"CustomerID": 1, "Name": "Mr. Patil", "Address": "ABC Street", "City": "Pimpri",
"ContactNumber": "1234567890"},
 {"CustomerID": 2, "Name": "Ms. Deshmukh", "Address": "XYZ Street", "City": "Pune",
"ContactNumber": "9876543210"},
 // ... (8 more documents)
])
([
 {"LoanID": 1, "CustomerID": 1, "LoanType": "Home Loan", "LoanAmount": 150000,
"LoanCity": "Pimpri"},
 {"LoanID": 2, "CustomerID": 2, "LoanType": "Car Loan", "LoanAmount": 100000,
"LoanCity": "Pimpri"},
 // ... (8 more documents)
])
```

a. List all customers whose name starts with 'D' character

```
db.Customers.find({ "Name": /^D/ })
```

b.List the names of customer in descending order who has taken a loan from Pimpri city.

```
db.Customers.find({ "City": "Pimpri" }).sort({ "Name": -1 })
```

c.Display customer details having maximum loan amount.

```
db.Customers.aggregate([
 {
  $lookup: {
   from: "Loans",
   localField: "CustomerID",
   foreignField: "CustomerID",
   as: "customer_loans"
  }
 },
 {
  $unwind: "$customer_loans"
 },
 {
  $sort: { "customer_loans.LoanAmount": -1 }
 },
 {
  $limit: 1
 }
])
```

d.Update the address of customer whose name is “Mr. Patil” and loan_amt is greater than 100000.

```
db.Customers.update(
 { "Name": "Mr. Patil", "LoanAmount": { $gt: 100000 } },
 { $set: { "Address": "New Updated Address" } }
)
```

Create a web page that shows use of transition properties, transition delay and duration effect.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Transition Example</title>
```

```
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin: 50px;
    }
    .box {
      width: 100px;
      height: 100px;
      background-color: #3498db;
      margin: 20px;
      display: inline-block;
      transition: transform 1s ease-in-out;
    }
    .box:hover {
      transform: scale(1.2);
    }
    .delayed-box {
      width: 100px;
      height: 100px;
      background-color: #e74c3c;
      margin: 20px;
      display: inline-block;
      transition: transform 1s ease-in-out 0.5s;
    }
    .delayed-box:hover {
      transform: scale(1.2);
    }
    .duration-box {
      width: 100px;
      height: 100px;
      background-color: #2ecc71;
      margin: 20px;
      display: inline-block;
      transition: transform 0.5s ease-in-out;
    }
    .duration-box:hover {
      transform: scale(1.2);
    }
  </style>
</head>
<body>
```

```
  <h2>Transition Example</h2>
  <div class="box"></div>
  <div class="delayed-box"></div>
  <div class="duration-box"></div>
</body>
</html>
```

Model the following Online shopping information as a document database. Consider online shopping where the customer can get different products from different brands. Customers can rate the brands and products 2. Assume appropriate attributes and collections as per the query requirements 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

```
[
 {"ProductID": 1, "Name": "Laptop", "Brand": "HP", "WarrantyPeriod": "1 year",
"Rating": 4.5},
 {"ProductID": 2, "Name": "Smartphone", "Brand": "Samsung", "WarrantyPeriod": "2
years", "Rating": 4.2},
 {"ProductID": 3, "Name": "Headphones", "Brand": "Sony", "WarrantyPeriod": "1 year",
"Rating": 4.8},
 {"ProductID": 4, "Name": "Camera", "Brand": "Canon", "WarrantyPeriod": "1 year",
"Rating": 4.0},
 {"ProductID": 5, "Name": "Refrigerator", "Brand": "LG", "WarrantyPeriod": "3 years",
"Rating": 4.6}
]
[
 {"CustomerID": 1, "Name": "John Doe", "City": "New York", "BillAmount": 75000},
 {"CustomerID": 2, "Name": "Alice Smith", "City": "Los Angeles", "BillAmount": 60000},
 {"CustomerID": 3, "Name": "Bob Johnson", "City": "Chicago", "BillAmount": 45000},
 {"CustomerID": 4, "Name": "Emily White", "City": "San Francisco", "BillAmount":
80000},
 {"CustomerID": 5, "Name": "Charlie Brown", "City": "Houston", "BillAmount": 90000}
]
[
 {"PurchaseID": 1, "CustomerID": 1, "ProductID": 2, "PurchaseDate": "15/08/2023"},
 {"PurchaseID": 2, "CustomerID": 2, "ProductID": 4, "PurchaseDate": "15/08/2023"},
 {"PurchaseID": 3, "CustomerID": 3, "ProductID": 1, "PurchaseDate": "15/08/2023"},
 {"PurchaseID": 4, "CustomerID": 4, "ProductID": 5, "PurchaseDate": "16/08/2023"},
 {"PurchaseID": 5, "CustomerID": 5, "ProductID": 3, "PurchaseDate": "17/08/2023"}
]
```

a. List the names of product whose warranty period is one year
   db.Products.find({ "WarrantyPeriod": "1 year" }, { "Name": 1, "_id": 0 })

b. List the customers has done purchase on “15/08/2023”.

```
db.Customers.aggregate([
 {
  $lookup: {
   from: "Purchases",
   localField: "CustomerID",
   foreignField: "CustomerID",
   as: "customer_purchases"
  }
 },
 {
  $match: { "customer_purchases.PurchaseDate": "15/08/2023" }
 },
 {
  $project: { "Name": 1, "_id": 0 }
 }
])
```

c. Display the names of products with brand which have highest rating

```
db.Products.aggregate([
 {
  $group: {
   _id: "$Brand",
   maxRating: { $max: "$Rating" }
  }
 },
 {
  $lookup: {
   from: "Products",
   localField: "_id",
   foreignField: "Brand",
   as: "brand_products"
  }
 },
 {
  $unwind: "$brand_products"
 },
 {
  $match: { "brand_products.Rating": "$maxRating" }
 },
 {
  $project: { "Product": "$brand_products.Name", "Brand": 1, "_id": 0 }
 }
```

])

d. . d. Display customers who stay in …… city and billamt >50000 .
db.Customers.find({ "City": "New York", "BillAmount": { $gt: 50000 } })

Write a HTML code which will divide web page in three frames. First frame should consists of company name as heading. Second frame should consists of name of departments with hyperlink. Once click on any department, it should display information of that department in third frame.
Index.html

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

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Company Information</title>
</head>

<body>

 <!-- First Frame: Company Name -->
 <iframe src="company1.html" name="companyFrame" width="100%" height="10%"
frameborder="0"></iframe>

 <!-- Second Frame: Departments -->
 <iframe src="departments1.html" name="departmentsFrame" width="20%"
height="80%" frameborder="0"></iframe>

 <!-- Third Frame: Department Information -->
 <iframe src="blank.html" name="departmentInfoFrame" width="80%" height="80%"
frameborder="0"></iframe>

</body>

</html>
```

Department.html

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

```
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Departments</title>
</head>

<body>

 <h2>Departments</h2>
 <ul>
  <li><a href="javascript:void(0);" onclick="showDepartmentInfo('HR')">Human
Resources</a></li>
  <li><a href="javascript:void(0);" onclick="showDepartmentInfo('IT')">Information
Technology</a></li>
  <!-- Add more departments as needed -->
 </ul>

 <script>
  function showDepartmentInfo(department) {
   parent.departmentInfoFrame.location.href = department + ".html";
  }
 </script>

</body>

</html>
```

Company.html

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

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Company Name</title>
</head>

<body>

 <h1>Company Name</h1>

</body>
```

```
</html>
```

Department_info.html

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Department Information</title>
</head>

<body>

  <h2>Department Information</h2>
  <p>hello  in humane resoures how a day</p>
  <!-- Add content specific to the department -->

</body>

</html>
```

HR.html

```
<html>
   <head>
      <title>human resourece</title>
   </head>
   <body>
      <h1>Human resource</h1>
      <p>It is human resource department and  welcome here</p>
   </body>
</html>
```

IT.html

```
<html>
   <head>
      <title>Information technology</title>
   </head>
   <body>
      <h1>Information Technology</h1>
      <p>Hello welcome to the information technology department and here we do a
tecnoloy releted work</p>
   </body>
</html>
```

Blank.html asa blank

Model the following sales system as a document database. Consider a set of products, customers, orders and invoices. An invoice is generated when an order is processed. 2. Assume appropriate attributes and collections as per the query requirements. 3. Insert at least 5 documents in each collection. 4. Answer the following Queries.

```
> use mydb
switched to db mydb
mydb> db.createCollection("products")
{ ok: 1 }
mydb>
db.products.insertMany([{"product_id":1,"name":"laptop","price":1200},{"product_id":2,"name":"printer","price":300},{"product_id":3,"name":"smartphone","price":800},{"product_id":4,"name":"headphone","price":100},{"product_id":5,"name":"tablet","price":500}])

db.createCollection("custo")
{ ok: 1 }
mydb>
db.custo.insertMany([{"custo_id":1,"name":"Mr.Rajiv","email":"rajiv@example.com"},{"custo_id":2,"name":"Mrs.Meera","email":"meera@example.com"},{"custo_id":3,"name":"Mr.Suresh","email":"suresh@example.com"},{"custo_id":4,"name":"Ms.Anu","email":"anu@example.com"},{"custo_id":5,"name":"Dr.Priya","email":"priya@example.com"}]
)

db.createCollection("orders")
{ ok: 1 }
mydb>
db.orders.insertMany([{"o_id":1,"custo_id":1,"products":[{"product_id":1,"quantity":2},{"product_id":3,"quantity":1}],"total_value":2900,"proccessed":true},{"o_id":2,"custo_id":3,"products":[{"product_id":2,"quantity":1},{"product_id":4,"quantity":3}],"total_value":700,"proccessed":false},{"o_id":3,"custo_id":2,"products":[{"product_id":5,"quantity":1}],"total_value":500,"proccessed":true},{"o_id":4,"custo_id":4,"products":[{"product_id":1,"quantity":1},{"product_id":3,"quantity":2}],"total_value":2500,"proccessed":false},{"o_id":5,"custo_id":5,"products":[{"product_id":2,"quantity":2},{"product_id":4,"quantity":1},{"product_id":5,"quantity":1}],"total_value":1900,"proccessed":true}])

db.invoice.insertMany([{"in_id":1,"o_id":1,"amount":2900,"date":"2023-01-15"},{"in_id":2,"o_id":3,"amount":500,"date":"2023-02-01"},{"in_id":3,"o_id":5,"amount":1900,"date":"2023-03-05"}])
```

a. List all products in the inventory.

```
db.products.find()
```

b. List the details of orders with a value >20000.
db.orders.find({total_value:{$gt:2000}})

c. List all the orders which has not been processed (invoice not generated)

db.orders.find({proccessed:false})

d. List all the orders along with their invoice for “Mr. Rajiv”.
db.orders.aggregate([{$match:{"custo_id":1}},{$lookup:{from:"invoice",localField:"_id",foreignField:"o_id",as:"invoice"}}])

SLIP12

Design an appropriate HTML form for customer registration visiting a departmental store. Form should consist of fields such as name, contact no, gender, preferred days of purchasing, favorite item (to be selected from a list of items), suggestions etc. You should provide button to submit as well as reset the form contents.

```
<html>
 <body>
  <form>
   <h1>Customer Registration</h1>
   <table border=1>
    <tr>
     <td>name</td>
     <td><input type="text" size="20"></td>
    </tr>
    <tr>
     <td>contact no</td>
     <td><input type="text" size="20"></td>
    </tr>
    <tr>
     <td>gender</td>
     <td><input type="radio"size="20">male<input
type="radio"size="20">female</td>
    </tr>
    <tr>
     <td>Days of purchasing</td>
     <td><input type="text" size="20"></td>
    </tr>
    <tr>
     <td>favorite item</td>
     <td><select name="example">
        <option value="item1">item1</option>
        <option value="item2">item2</option>
        <option value="item3">item3</option>
```

```
      </select></td>
    </tr>
    <tr>
      <td>Suggetion</td>
      <td><textarea rows="4" cols="50"></textarea></td>
    </tr>
    <tr>
      <td><input type="reset" value="Reset"></td>
      <td><input type="submit" value="submit"></td>
    </tr>
  </table>
 </form>
</body>
</html>
```

SLIP 13

Create a useful web with the following information and structure using gs like: , , , ,

```
<!DOCTYPE html>

<html lang="en">


<head>

  <meta charset="UTF-8">

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

  <title>Movie Information</title>

  <!-- Add your CSS styles or external stylesheet links here -->

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 0;

      padding: 0;

      box-sizing: border-box;
```

```
    }

    header, footer, nav, section, aside {
      margin: 10px;
      padding: 10px;
      border: 1px solid #ccc;
    }

    nav ul {
      list-style-type: none;
      padding: 0;
      margin: 0;
    }

    nav li {
      display: inline;
      margin-right: 10px;
    }
  </style>
</head>

<body>
  <header>
    <h1>Movie Database</h1>
    <p>Welcome to our online movie information platform.</p>
```

```
</header>

<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Movies</a></li>
    <li><a href="#">Actors</a></li>
    <li><a href="#">Producers</a></li>
  </ul>
</nav>

<section>
  <h2>Featured Movies</h2>
  <p>Explore our curated list of must-watch movies.</p>
  <!-- Add more content as needed -->
</section>

<aside>
  <h2>Latest News</h2>
  <p>Stay updated with the latest movie industry news.</p>
  <!-- Add more content as needed -->
</aside>

<footer>
  <p>&copy; 2023 Movie Database. All rights reserved.</p>
```

</footer>

</body>

</html>

Model the following Student Competition information as a document database. Consider Student competition information where the student can participate in many competitions. 2. Assume appropriate attributes and collections as per the query requirements 3. Insert at least 10 documents in each collection. 4. Answer the following Queries.

db.createCollection("Stuents")

{ ok: 1 }

mydb>
db.Students.insertMany([{"stud_id":1,"name":"john","class":"FY","Competitions":[{"competition_name":"running","position":1},{"competition_name":"painting","position":2}]},{"stud_id":2,"name":"abc","class":"TY","Competitions":[{"competition_name":"hakethon","position":1},{"competition_name":"codevita","position":3}]},{"stud_id":3,"name":"xyz","class":"SY","Competitions":[{"competition_name":"swimming","position":2},{"competition_name":"climbing","position":1}]},{"stud_id":4,"name":"dhanu","class":"FY","Competitions":[{"competition_name":"sleeping","position":1},{"competition_name":"coding","position":2}]}])

db.Competition.insertMany([{"com_id":101,"name":"hakethon","participants":[{"stud_id":2,"position":1}]},{"com_id":102,"name":"sleepin","participants":[{"stud_id":4,"position":1}]},{"com_id":103,"name":"coding","participants":[{"stud_id":4,"position":2}]},{"com_id":104,"name":"codevita","participants":[{"stud_id":2,"position":2}]}])

a. Display average no. of students participating in each competition.
   db.Competition.aggregate([{$project:{name :1,winners:{$slice:["$participants",3]}}}])
b. Find the number of student for programming competition.
   db.Competition.findOne({name:"coding"}).participants.length;
c. Display the names of first three winners of each competition
   db.Competition.aggregate([{$project:{name :1,winners:{$slice:["$participants",3]}}}])
d. . d. Display students from class 'FY' and participated in 'E-Rangoli ' Competition.
   db.Students.find({class:"FY","Competition.competition_name":"running"})

SLIP 14

Design an HTML form to take the information of a customer for booking a travel plan consisting of fields such as name, address, contact no., gender, preferred season(Checkboxes), location type(to be selected from a list) etc. You should provide button to submit as well as reset the form contents

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Travel Plan Booking</title>
  <!-- Add your CSS styles or external stylesheet links here -->
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    form {
      max-width: 600px;
      margin: 0 auto;
    }

    label {
      display: block;
      margin-bottom: 5px;
    }

    input,
    select {
      width: 100%;
      padding: 8px;
      margin-bottom: 10px;
      box-sizing: border-box;
    }

    input[type="checkbox"] {
      margin-bottom: 5px;
    }

    button {
      background-color: #4CAF50;
```

```
      color: white;
      padding: 10px 15px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    button[type="reset"] {
      background-color: #f44336;
    }
  </style>
</head>

<body>
  <form action="#" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>

    <label for="address">Address:</label>
    <input type="text" id="address" name="address" required>

    <label for="contact">Contact Number:</label>
    <input type="tel" id="contact" name="contact" required>

    <label>Gender:</label>
    <label for="male"><input type="radio" id="male" name="gender" value="male"
required> Male</label>
    <label for="female"><input type="radio" id="female" name="gender" value="female"
required> Female</label>

    <label>Preferred Season:</label>
    <label for="spring"><input type="checkbox" id="spring" name="season" value="spring">
Spring</label>
    <label for="summer"><input type="checkbox" id="summer" name="season"
value="summer"> Summer</label>
    <label for="autumn"><input type="checkbox" id="autumn" name="season"
value="autumn"> Autumn</label>
    <label for="winter"><input type="checkbox" id="winter" name="season"
value="winter"> Winter</label>

    <label for="locationType">Location Type:</label>
    <select id="locationType" name="locationType" required>
```

```
        <option value="beach">Beach</option>
        <option value="mountain">Mountain</option>
        <option value="city">City</option>
        <option value="countryside">Countryside</option>
      </select>

      <button type="submit">Submit</button>
      <button type="reset">Reset</button>
    </form>
</body>

</html>
```

Model the following system as a graph model, and answer the queries using Cypher. Government provides various scholarships for students. A students apply for scholarship. Student can get benefits of more than one scholarship if satisfies the criteria. Student can recommend it for his friends or other family members 1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. 2. Create nodes and relationships, along with their properties, andvisualize your actual Graph model. 3. Answer the following Queries in Cypher
:
a. List the names of scholarship for OBC category.b. Count no. of students who are benefitted by ____ scholarship in year 2020-2021 c. Update the income limit for _____scholarship. d. List the most popular scholarship.