Slip 2

1. 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">
   <title>Bootstrap Grid Example</title>
   <!-- Bootstrap CSS -->
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

<div class="container">
   <div class="row">
      <!-- Column 1 -->
      <div class="col-md-4">
         <div class="p-3 border bg-light">Column 1</div>
      </div>
      <!-- Column 2 -->
      <div class="col-md-4">
         <div class="p-3 border bg-light">Column 2</div>
      </div>
      <!-- Column 3 -->
      <div class="col-md-4">
         <div class="p-3 border bg-light">Column 3</div>
      </div>
   </div>
```

```
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>
</html>
```

1. 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]

3. Insert at least 5 documents in each collection. [3]

4. Answer the following Queries.

a. List all newspapers available “NASHIK” city [3]

b. List all the newspaper of “Marathi” language [3]

c. Count no. of publishers of “Gujrat” state [4]

d. Write a cursor to show newspapers with highest sale in Maharashtra State [4]

→ Newspapers Collection (5 documents):

```
{
  "_id": 1,
  "name": "Daily Nashik Times",
  "language": "Marathi",
  "publisher_id": 1,
  "city_id": 1,
  "sales": 5000
},
{
  "_id": 2,
```

```
  "name": "Pune Mirror",
  "language": "English",
  "publisher_id": 2,
  "city_id": 2,
  "sales": 8000
},
{
  "_id": 3,
  "name": "Ahmedabad Chronicle",
  "language": "Gujarati",
  "publisher_id": 3,
  "city_id": 3,
  "sales": 12000
},
{
  "_id": 4,
  "name": "Mumbai Express",
  "language": "Hindi",
  "publisher_id": 1,
  "city_id": 4,
  "sales": 15000
},
{
  "_id": 5,
  "name": "Nashik Dainik",
  "language": "Marathi",
  "publisher_id": 4,
  "city_id": 1,
  "sales": 7000
```

}

Publishers Collection (5 documents):

{
  "_id": 1,
  "name": "Maharashtra Media",
  "state": "Maharashtra"
},
{
  "_id": 2,
  "name": "Pune Press",
  "state": "Maharashtra"
},
{
  "_id": 3,
  "name": "Gujarat News Corp",
  "state": "Gujarat"
},
{
  "_id": 4,
  "name": "Nashik Publishing House",
  "state": "Maharashtra"
},
{
  "_id": 5,
  "name": "Surat Daily",
  "state": "Gujarat"
}

Cities Collection (5 documents):

{

```
 "_id": 1,
 "name": "Nashik",
 "state": "Maharashtra"
},
{
 "_id": 2,
 "name": "Pune",
 "state": "Maharashtra"
},
{
 "_id": 3,
 "name": "Ahmedabad",
 "state": "Gujarat"
},
{
 "_id": 4,
 "name": "Mumbai",
 "state": "Maharashtra"
},
{
 "_id": 5,
 "name": "Surat",
 "state": "Gujarat"
}
```


List all newspapers available in **Nashik** city

```
db.newspapers.find({
  "city_id": (db.cities.findOne({"name": "Nashik"}))._id
})
```

List all newspapers of **Marathi** language

```
db.newspapers.find({
  "language": "Marathi"
})
```

Count the number of publishers in the **Gujarat** state

```
db.publishers.count({
  "state": "Gujarat"
})
```

Write a cursor to show newspapers with the highest sale in **Maharashtra** State

```
db.newspapers.find({
  "city_id": {
    $in: db.cities.find({ "state": "Maharashtra" }).map(city => city._id)
  }
}).sort({"sales": -1}).limit(1)
```

Slip 3

Write a bootstrap application to display thumbnails of the 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 using Bootstrap</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div class="container mt-5">
```

```
  <h2 class="text-center mb-4">Image Thumbnails</h2>

  <!-- Bootstrap Grid for Thumbnails -->
  <div class="row">
    <!-- Thumbnail 1 -->
    <div class="col-md-4">
      <div class="card">
        <img src="https://via.placeholder.com/300" class="card-img-top img-thumbnail"
alt="Thumbnail 1">
        <div class="card-body">
          <h5 class="card-title">Image 1</h5>
          <p class="card-text">This is a description of the first image.</p>
        </div>
      </div>
    </div>

    <!-- Thumbnail 2 -->
    <div class="col-md-4">
      <div class="card">
        <img src="https://via.placeholder.com/300" class="card-img-top img-thumbnail"
alt="Thumbnail 2">
        <div class="card-body">
          <h5 class="card-title">Image 2</h5>
          <p class="card-text">This is a description of the second image.</p>
        </div>
      </div>
    </div>

    <!-- Thumbnail 3 -->
```

```
      <div class="col-md-4">
        <div class="card">
          <img src="https://via.placeholder.com/300" class="card-img-top img-thumbnail"
alt="Thumbnail 3">
          <div class="card-body">
            <h5 class="card-title">Image 3</h5>
            <p class="card-text">This is a description of the third image.</p>
          </div>
        </div>
      </div>
    </div>

    <!-- More rows can be added similarly to display more images -->
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>
</html>
```

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]

3. Insert at least 5 documents in each collection. [3]

4. Answer the following Queries.

a. Display name of employee who has highest salary [3]

b. Display biggest department with max. no. of employees [3]

c. Write a cursor which shows department wise employee information [4]

d. List all the employees who work in Sales dept and salary > 50000 [4

Insert 5 documents in the Employees collection:

```
db.employees.insertMany([
   { _id: 1, name: "John Doe", salary: 60000, dept_id: 1 },
   { _id: 2, name: "Jane Smith", salary: 75000, dept_id: 2 },
   { _id: 3, name: "Emily Davis", salary: 55000, dept_id: 1 },
   { _id: 4, name: "Michael Brown", salary: 50000, dept_id: 3 },
   { _id: 5, name: "Sarah Wilson", salary: 48000, dept_id: 2 }
]);
```

Insert 5 documents in the Departments collection:

```
db.departments.insertMany([
   { _id: 1, dept_name: "Sales", employees: [1, 3], num_employees: 2 },
   { _id: 2, dept_name: "Marketing", employees: [2, 5], num_employees: 2 },
   { _id: 3, dept_name: "HR", employees: [4], num_employees: 1 }
]);
```

Display the name of the employee who has the highest salary

```
db.employees.find().sort({ salary: -1 }).limit(1).forEach(employee => {
   print("Employee with highest salary: " + employee.name);
});
```

Display the biggest department with the maximum number of employees

```
db.departments.find().sort({ num_employees: -1 }).limit(1).forEach(department => {
   print("Biggest department: " + department.dept_name);
});
```

Write a cursor which shows department-wise employee information

```
var cursor = db.departments.find();
cursor.forEach(dept => {
   print("Department: " + dept.dept_name);
   db.employees.find({ dept_id: dept._id }).forEach(emp => {
      print("  Employee: " + emp.name + ", Salary: " + emp.salary);
   });
});
```

List all the employees who work in the Sales department and have a salary greater than 50,000

```
var sales_dept = db.departments.findOne({ dept_name: "Sales" });

db.employees.find({ dept_id: sales_dept._id, salary: { $gt: 50000 } }).forEach(employee => {
   print("Employee: " + employee.name);
});
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Basic Styled Table using Bootstrap</title>
   <!-- Bootstrap CSS -->
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

<div class="container mt-5">
   <h2 class="text-center mb-4">User Information Table</h2>

   <!-- Bootstrap Table -->
   <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>Michael</td>
            <td>Brown</td>
            <td>michael.brown@example.com</td>
          </tr>
        </tbody>
    </table>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>
</html>
```

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]

3. Insert at least 10 documents in each collection. [3]

4. Answer the following Queries

a. List the names of hospitals with…… specialization. [3]

b. List the Names of all hospital located in ……. city [3]

c. List the names of hospitals where Dr. Deshmukh visits [4]

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

Insert 10 documents in the Hospitals collection:

```
db.hospitals.insertMany([

  { _id: 1, name: "Nashik Medical Centre", city: "Nashik", specializations: ["Pediatric", "Orthopedic"], rating: 4.5, reviews: [] },

  { _id: 2, name: "City Hospital", city: "Nashik", specializations: ["Gynaec", "Pediatric"], rating: 4.0, reviews: [] },

  { _id: 3, name: "Care Hospital", city: "Nashik", specializations: ["Orthopedic"], rating: 3.8, reviews: [] },

  { _id: 4, name: "Sunshine Hospital", city: "Nashik", specializations: ["Gynaec"], rating: 4.2, reviews: [] },

  { _id: 5, name: "Nashik Heart Institute", city: "Nashik", specializations: ["Cardiology"], rating: 4.7, reviews: [] },

  { _id: 6, name: "Eagle Eye Hospital", city: "Nashik", specializations: ["Ophthalmology"], rating: 4.1, reviews: [] },

  { _id: 7, name: "Hope Clinic", city: "Nashik", specializations: ["General"], rating: 3.5, reviews: [] },

  { _id: 8, name: "Vishwa Hospital", city: "Pune", specializations: ["Pediatric", "Gynaec"], rating: 4.6, reviews: [] },

  { _id: 9, name: "Rainbow Hospital", city: "Nashik", specializations: ["Orthopedic", "Cardiology"], rating: 4.0, reviews: [] },

  { _id: 10, name: "LifeCare Hospital", city: "Nashik", specializations: ["Gynaec", "General"], rating: 3.9, reviews: [] }

]);
```

2. Insert 10 documents in the Doctors collection:

```
db.doctors.insertMany([
  { _id: 1, name: "Dr. Deshmukh", specialization: "Pediatric", hospitals: [1, 2, 8] },
  { _id: 2, name: "Dr. Sharma", specialization: "Gynaec", hospitals: [2, 4, 10] },
  { _id: 3, name: "Dr. Kumar", specialization: "Orthopedic", hospitals: [1, 3, 9] },
  { _id: 4, name: "Dr. Verma", specialization: "Cardiology", hospitals: [5, 9] },
  { _id: 5, name: "Dr. Joshi", specialization: "Ophthalmology", hospitals: [6] },
  { _id: 6, name: "Dr. Rao", specialization: "General", hospitals: [7, 10] },
  { _id: 7, name: "Dr. Patel", specialization: "Pediatric", hospitals: [1, 4] },
  { _id: 8, name: "Dr. Singh", specialization: "Gynaec", hospitals: [2, 3] },
  { _id: 9, name: "Dr. Iyer", specialization: "Orthopedic", hospitals: [3, 9] },
  { _id: 10, name: "Dr. Desai", specialization: "Cardiology", hospitals: [5] }
]);
```

3. Insert 5 documents in the Reviews collection:

```
db.reviews.insertMany([
  { _id: 1, hospital_id: 1, reviewer_name: "Alice", rating: 5, comment: "Excellent pediatric care!" },
  { _id: 2, hospital_id: 2, reviewer_name: "Bob", rating: 4, comment: "Good service, but long wait times." },
  { _id: 3, hospital_id: 3, reviewer_name: "Charlie", rating: 3, comment: "Average experience." },
  { _id: 4, hospital_id: 4, reviewer_name: "David", rating: 4, comment: "Great for women's health." },
  { _id: 5, hospital_id: 5, reviewer_name: "Eve", rating: 5, comment: "Highly recommended for cardiac issues." }
]);
```

a. List the names of hospitals with a specific specialization (e.g., "Pediatric")

```
db.hospitals.find({ specializations: "Pediatric" }).forEach(hospital => {
  print("Hospital: " + hospital.name);
});
```

b. List the names of all hospitals located in a specific city (e.g., "Nashik")

```
db.hospitals.find({ city: "Nashik" }).forEach(hospital => {
```

```
    print("Hospital: " + hospital.name);
});
```

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

```
var doctor = db.doctors.findOne({ name: "Dr. Deshmukh" });
db.hospitals.find({ _id: { $in: doctor.hospitals } }).forEach(hospital => {
    print("Hospital: " + hospital.name);
});
```

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

```
db.hospitals.find({ rating: { $gte: 4 } }).forEach(hospital => {
    print("Hospital: " + hospital.name);
});
```

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>My Web Page</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .logo {
            height: 60px; /* Set the logo height */
        }
```

```
  </style>
</head>
<body>

<!-- Navigation Bar -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">
      <img src="https://via.placeholder.com/150x60" alt="Logo" class="logo"> My Website
    </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">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" 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>

<div class="container mt-5">
  <h1 class="text-center">Welcome to My Web Page</h1>
  <p class="lead text-center">This is a simple web page created using HTML and Bootstrap. It demonstrates various elements including a logo, informative text, pictures, and more!</p>

  <!-- Informative Text -->
  <h2>About Us</h2>
  <p>We are dedicated to providing high-quality services to our clients. Our team is experienced and ready to assist you with your needs.</p>

  <!-- Images -->
  <div class="row">
    <div class="col-md-6">
      <img src="https://via.placeholder.com/500" class="img-fluid" alt="Image 1">
    </div>
    <div class="col-md-6">
      <img src="https://via.placeholder.com/500" class="img-fluid" alt="Image 2">
    </div>
  </div>

  <!-- Hyperlinks -->
  <h3>Useful Links</h3>
  <ul>
    <li><a href="https://www.example.com" target="_blank">Example Link 1</a></li>
    <li><a href="https://www.example.com" target="_blank">Example Link 2</a></li>
```

```
    <li><a href="https://www.example.com" target="_blank">Example Link 3</a></li>
</ul>

<!-- Table -->
<h2 class="mt-5">Our Services</h2>
<table class="table">
    <thead>
        <tr>
            <th>Service</th>
            <th>Description</th>
            <th>Price</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Service 1</td>
            <td>Lorem ipsum dolor sit amet.</td>
            <td>$100</td>
        </tr>
        <tr>
            <td>Service 2</td>
            <td>Consectetur adipiscing elit.</td>
            <td>$200</td>
        </tr>
        <tr>
            <td>Service 3</td>
            <td>Sed do eiusmod tempor incididunt.</td>
            <td>$300</td>
        </tr>
```

```
      </tbody>
   </table>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</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]

3. Insert at least 5 documents in each collection. [3]

4. Answer the following Queries.

a. List the details of customers who have taken “Komal Jeevan” Policy [3]

b. Display average premium amount [3]

c. Increase the premium amount by 5% for policy type=”Monthly” [4]

d. Count no. of customers who have taken policy type “half yearly”. [4]’

1. Insert 5 documents in the Customers collection:

```
db.customers.insertMany([
   { _id: 1, name: "Amit Sharma", age: 30, address: "Nashik, Maharashtra", policies: [1, 2] },
   { _id: 2, name: "Sneha Patil", age: 25, address: "Nashik, Maharashtra", policies: [3] },
   { _id: 3, name: "Rohit Desai", age: 35, address: "Nashik, Maharashtra", policies: [2, 4] },
   { _id: 4, name: "Priya Joshi", age: 28, address: "Mumbai, Maharashtra", policies: [1, 5] },
   { _id: 5, name: "Rahul Verma", age: 40, address: "Nashik, Maharashtra", policies: [3, 4] }
]);
```

2. Insert 5 documents in the Policies collection:

```
db.policies.insertMany([
```

```
  { _id: 1, policy_name: "Komal Jeevan", company: "ABC Insurance", premium_amount: 1000, policy_type: "Monthly", customers: [1, 4] },

  { _id: 2, policy_name: "Secure Future", company: "XYZ Insurance", premium_amount: 2000, policy_type: "Yearly", customers: [1, 3] },

  { _id: 3, policy_name: "Health Shield", company: "ABC Insurance", premium_amount: 3000, policy_type: "Half-Yearly", customers: [2, 5] },

  { _id: 4, policy_name: "Family Protection", company: "XYZ Insurance", premium_amount: 2500, policy_type: "Half-Yearly", customers: [3, 5] },

  { _id: 5, policy_name: "Life Secure", company: "LMN Insurance", premium_amount: 1500, policy_type: "Monthly", customers: [4] }

]);
```

a. List the details of customers who have taken the “Komal Jeevan” Policy

```
var policy = db.policies.findOne({ policy_name: "Komal Jeevan" });

db.customers.find({ policies: policy._id }).forEach(customer => {

  print("Customer Name: " + customer.name + ", Age: " + customer.age + ", Address: " + customer.address);

});
```

b. Display average premium amount

```
var totalPremium = db.policies.aggregate([

  { $group: { _id: null, averagePremium: { $avg: "$premium_amount" } } }

]).toArray()[0].averagePremium;


print("Average Premium Amount: " + totalPremium);
```

c. Increase the premium amount by 5% for policy type = “Monthly”

```
db.policies.updateMany(

  { policy_type: "Monthly" },

  { $mul: { premium_amount: 1.05 } }

);
```

d. Count the number of customers who have taken policy type “Half-Yearly”

```
var halfYearlyPolicy = db.policies.find({ policy_type: "Half-Yearly" }).toArray();
var customerCount = new Set();

halfYearlyPolicy.forEach(policy => {
   policy.customers.forEach(customerId => {
      customerCount.add(customerId);
   });
});

print("Number of customers with Half-Yearly policy: " + customerCount.size);
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>3D Text with Hover Effect</title>
   <style>
      body {
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
         background-color: #f0f0f0; /* Light background for contrast */
         font-family: 'Arial', sans-serif; /* Font style */
      }
```

```
      .text {
         font-size: 60px; /* Font size */
         color: #3498db; /* Initial color */
         text-shadow: 1px 1px 0 rgba(0, 0, 0, 0.2); /* Initial shadow for depth */
         transition: transform 0.3s, color 0.3s; /* Smooth transition */
      }

      .text:hover {
         color: #e74c3c; /* Change color on hover */
         transform: translateY(-10px) rotateY(10deg); /* 3D effect on hover */
         text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.5); /* Enhanced shadow on hover */
      }
   </style>
</head>
<body>

<div class="text">3D Hover Text</div>

</body>
</html>
```

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]

3. Insert at least 5 documents in each collection. [3]

4. Answer the following Queries.

a. List names of all customers whose first name starts with a “S” [3]

b. List all customers who has open an account on 1/1/2020 in ___branch [3]

c. List the names customers where acctype=”Saving” [4]

d. Count total no. of loan account holder of …….branch [4]

1. Insert 5 documents in the Customers collection:

```
db.customers.insertMany([
  { _id: 1, first_name: "Sanjay", last_name: "Sharma", email: "sanjay@example.com", branch: "Nashik", account_ids: [1, 2] },
  { _id: 2, first_name: "Priya", last_name: "Kumar", email: "priya@example.com", branch: "Nashik", account_ids: [3] },
  { _id: 3, first_name: "Ravi", last_name: "Desai", email: "ravi@example.com", branch: "Mumbai", account_ids: [4] },
  { _id: 4, first_name: "Sneha", last_name: "Patel", email: "sneha@example.com", branch: "Pune", account_ids: [5] },
  { _id: 5, first_name: "Sita", last_name: "Verma", email: "sita@example.com", branch: "Nashik", account_ids: [6] }
]);
```

2. Insert 5 documents in the Accounts collection:

```
db.accounts.insertMany([
  { _id: 1, customer_id: 1, account_type: "Saving", balance: 5000, date_opened: new Date("2020-01-01") },
  { _id: 2, customer_id: 1, account_type: "Current", balance: 15000, date_opened: new Date("2021-06-15") },
  { _id: 3, customer_id: 2, account_type: "Saving", balance: 8000, date_opened: new Date("2020-01-01") },
  { _id: 4, customer_id: 3, account_type: "Loan", balance: 200000, date_opened: new Date("2019-05-10") },
  { _id: 5, customer_id: 4, account_type: "Saving", balance: 10000, date_opened: new Date("2020-07-20") },
  { _id: 6, customer_id: 5, account_type: "Saving", balance: 6000, date_opened: new Date("2020-01-01") }
]);
```

a. List names of all customers whose first name starts with an “S”

```
db.customers.find({ first_name: { $regex: "^S", $options: "i" } }).forEach(customer => {
```

```
    print("Customer Name: " + customer.first_name + " " + customer.last_name);
});
```

b. List all customers who opened an account on 1/1/2020 in a specific branch (e.g., "Nashik")

```
var specificBranch = "Nashik";
db.accounts.find({ date_opened: new Date("2020-01-01") }).forEach(account => {
    var customer = db.customers.findOne({ _id: account.customer_id, branch: specificBranch });
    if (customer) {
        print("Customer Name: " + customer.first_name + " " + customer.last_name);
    }
});
```

c. List the names of customers where acctype = “Saving”

```
db.accounts.find({ account_type: "Saving" }).forEach(account => {
    var customer = db.customers.findOne({ _id: account.customer_id });
    print("Customer Name: " + customer.first_name + " " + customer.last_name);
});
```

d. Count the total number of loan account holders in a specific branch (e.g., "Nashik")

```
var branch = "Nashik";
var loanAccountHolders = new Set();

db.accounts.find({ account_type: "Loan" }).forEach(account => {
    var customer = db.customers.findOne({ _id: account.customer_id, branch: branch });
    if (customer) {
        loanAccountHolders.add(customer._id);
    }
});

print("Total number of loan account holders in " + branch + ": " + loanAccountHolders.size);
```

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">

   <title>Bootstrap Button Styles</title>

   <!-- Bootstrap CSS -->

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

</head>

<body>


<div class="container mt-5">

   <h2 class="text-center">Bootstrap Button Styles</h2>

   <div class="row text-center mt-4">

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

         <button class="btn btn-secondary">Secondary Button</button>

      </div>

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

         <button class="btn btn-primary">Primary Button</button>

      </div>

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

         <button class="btn btn-success">Success Button</button>

      </div>

   </div>

   <div class="row text-center mt-4">
```

```
      <div class="col-md-4">
        <button class="btn btn-danger">Danger Button</button>
      </div>
      <div class="col-md-4">
        <button class="btn btn-warning">Warning Button</button>
      </div>
      <div class="col-md-4">
        <button class="btn btn-info">Info Button</button>
      </div>
    </div>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</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.

a. List all the items qty is greater than 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

d. Find all warehouse that keeps item “Planner” and having in stock quantity less than 20

1. Insert 5 documents in the Items collection:

```
db.items.insertMany([
  { _id: 1, name: "Notebook", category: "Stationery", tags: 4, quantity: 500, status: "A", height: 10 },
  { _id: 2, name: "Planner", category: "Stationery", tags: 6, quantity: 15, status: "B", height: 9 },
  { _id: 3, name: "Eraser", category: "Stationery", tags: 3, quantity: 1000, status: "A", height: 2 },
  { _id: 4, name: "Textbook", category: "Books", tags: 2, quantity: 250, status: "C", height: 12 },
  { _id: 5, name: "Markers", category: "Stationery", tags: 5, quantity: 30, status: "B", height: 8 }
]);
```

2. Insert 5 documents in the Warehouses collection:

```
db.warehouses.insertMany([
  { _id: 1, name: "Warehouse A", location: "Nashik", items: [{ item_id: 1, quantity: 200 }, { item_id: 2, quantity: 10 }] },
  { _id: 2, name: "Warehouse B", location: "Pune", items: [{ item_id: 3, quantity: 500 }, { item_id: 4, quantity: 100 }] },
  { _id: 3, name: "Warehouse C", location: "Mumbai", items: [{ item_id: 1, quantity: 300 }, { item_id: 5, quantity: 20 }] },
  { _id: 4, name: "Warehouse D", location: "Nashik", items: [{ item_id: 2, quantity: 15 }, { item_id: 4, quantity: 150 }] },
  { _id: 5, name: "Warehouse E", location: "Nashik", items: [{ item_id: 1, quantity: 50 }, { item_id: 3, quantity: 300 }] }
]);
```

a. List all the items where quantity is greater than 300

```
db.items.find({ quantity: { $gt: 300 } }).forEach(item => {
  print("Item Name: " + item.name + ", Quantity: " + item.quantity);
});
```

b. List all items which have tags less than 5

```
db.items.find({ tags: { $lt: 5 } }).forEach(item => {
  print("Item Name: " + item.name + ", Tags: " + item.tags);
});
```

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.items.find({
   $or: [
      { status: "B" },
      { $and: [{ quantity: { $lt: 50 } }, { height: { $gt: 8 } }] }
   ]
}).forEach(item => {
   print("Item Name: " + item.name + ", Status: " + item.status + ", Quantity: " + item.quantity + ", Height: " + item.height);
});
```

d. Find all warehouses that keep item “Planner” and having in-stock quantity less than 20

```
var plannerItem = db.items.findOne({ name: "Planner" });

db.warehouses.find({
   items: {
      $elemMatch: { item_id: plannerItem._id, quantity: { $lt: 20 } }
   }
}).forEach(warehouse => {
   print("Warehouse Name: " + warehouse.name + ", Location: " + warehouse.location);
});
```

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>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa; /* Light background color */
        }
        .container {
            margin-top: 50px;
            border: 1px solid #dee2e6; /* Border around the form */
            border-radius: 10px; /* Rounded corners */
            padding: 20px; /* Padding inside the container */
            background-color: white; /* White background for the form */
        }
    </style>
</head>
<body>

<div class="container">
    <h2 class="text-center">Student Registration Form</h2>
    <form>
        <!-- Full Name -->
        <div class="mb-3">
            <label for="fullName" class="form-label">Full Name</label>
            <input type="text" class="form-control" id="fullName" required>
        </div>

        <!-- Email Address -->
```

```
    <div class="mb-3">
      <label for="email" class="form-label">Email Address</label>
      <input type="email" class="form-control" id="email" required>
    </div>

    <!-- Date of Birth -->
    <div class="mb-3">
      <label for="dob" class="form-label">Date of Birth</label>
      <input type="date" class="form-control" id="dob" required>
    </div>

    <!-- Phone Number -->
    <div class="mb-3">
      <label for="phone" class="form-label">Phone Number</label>
      <input type="tel" class="form-control" id="phone" required>
    </div>

    <!-- Search for Course -->
    <div class="mb-3">
      <label for="course" class="form-label">Search for Course</label>
      <input type="search" class="form-control" id="course" placeholder="e.g. Computer Science, Business, etc." required>
    </div>

    <!-- Gender -->
    <div class="mb-3">
      <label class="form-label">Gender</label>
      <div>
        <input type="radio" id="male" name="gender" value="male" required>
```

```
            <label for="male">Male</label>
        </div>
        <div>
            <input type="radio" id="female" name="gender" value="female">
            <label for="female">Female</label>
        </div>
        <div>
            <input type="radio" id="other" name="gender" value="other">
            <label for="other">Other</label>
        </div>
    </div>

    <!-- Address -->
    <div class="mb-3">
        <label for="address" class="form-label">Address</label>
        <textarea class="form-control" id="address" rows="3" required></textarea>
    </div>

    <!-- Submit Button -->
    <div class="text-center">
        <button type="submit" class="btn btn-primary">Register</button>
    </div>
  </form>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</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]

3. Insert at least 10 documents in each collection. [3]

4. Answer the following Queries. a. List all customers whose name starts with 'D' character [3]

b.List the names of customer in descending order who has taken a loan from Pimpri city. [3]

c.Display customer details having maximum loan amount. [4]

d.Update the address of customer whose name is “Mr. Patil” and loan_amt is greater than 100000. [4]

1. Insert 10 documents in the Customers collection:

```
db.customers.insertMany([
   { _id: 1, name: "Dinesh Patil", address: "Nashik", email: "dinesh@example.com", phone:
"1234567890" },
   { _id: 2, name: "Sneha Kulkarni", address: "Pune", email: "sneha@example.com", phone:
"0987654321" },
   { _id: 3, name: "Ravi Joshi", address: "Mumbai", email: "ravi@example.com", phone: "1122334455" },
   { _id: 4, name: "Deepa Patil", address: "Pimpri", email: "deepa@example.com", phone: "2233445566"
},
   { _id: 5, name: "Amit Sharma", address: "Nashik", email: "amit@example.com", phone: "3344556677"
},
   { _id: 6, name: "Disha Shetty", address: "Nashik", email: "disha@example.com", phone: "4455667788"
},
   { _id: 7, name: "Mr. Patil", address: "Pimpri", email: "mrpatil@example.com", phone: "5566778899" },
   { _id: 8, name: "Suresh Verma", address: "Mumbai", email: "suresh@example.com", phone:
"6677889900" },
   { _id: 9, name: "Kiran Singh", address: "Pune", email: "kiran@example.com", phone: "7788990011" },
   { _id: 10, name: "Deepak Gupta", address: "Pimpri", email: "deepak@example.com", phone:
"8899001122" }
]);
```

2. Insert 10 documents in the Loans collection:

```
db.loans.insertMany([
   { _id: 1, customer_id: 1, loan_type: "Personal", loan_amount: 150000, city: "Nashik", date_taken: new
Date("2023-01-15") },
   { _id: 2, customer_id: 2, loan_type: "Home", loan_amount: 300000, city: "Pune", date_taken: new
Date("2022-03-10") },
   { _id: 3, customer_id: 3, loan_type: "Auto", loan_amount: 200000, city: "Mumbai", date_taken: new
Date("2022-05-22") },
   { _id: 4, customer_id: 4, loan_type: "Personal", loan_amount: 50000, city: "Pimpri", date_taken: new
Date("2023-04-01") },
```

```
  { _id: 5, customer_id: 5, loan_type: "Home", loan_amount: 400000, city: "Nashik", date_taken: new
Date("2021-12-15") },
  { _id: 6, customer_id: 6, loan_type: "Personal", loan_amount: 75000, city: "Nashik", date_taken: new
Date("2023-02-11") },
  { _id: 7, customer_id: 7, loan_type: "Auto", loan_amount: 120000, city: "Pimpri", date_taken: new
Date("2023-05-20") },
  { _id: 8, customer_id: 8, loan_type: "Personal", loan_amount: 30000, city: "Mumbai", date_taken: new
Date("2023-06-30") },
  { _id: 9, customer_id: 9, loan_type: "Home", loan_amount: 250000, city: "Pune", date_taken: new
Date("2023-07-14") },
  { _id: 10, customer_id: 10, loan_type: "Personal", loan_amount: 90000, city: "Pimpri", date_taken:
new Date("2023-08-05") }
]);
```

Step 3: Queries

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

```
db.customers.find({ name: { $regex: "^D", $options: "i" } }).forEach(customer => {
  print("Customer Name: " + customer.name);
});
```

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

```
db.customers.aggregate([
  {
    $lookup: {
      from: "loans",
      localField: "_id",
      foreignField: "customer_id",
      as: "loan_info"
    }
  },
  { $unwind: "$loan_info" },
  { $match: { "loan_info.city": "Pimpri" } },
  { $sort: { name: -1 } },
  { $project: { name: 1 } }
]).forEach(customer => {
  print("Customer Name: " + customer.name);
});
```

c. Display customer details having the maximum loan amount

```
var maxLoan = db.loans.find().sort({ loan_amount: -1 }).limit(1).toArray()[0];
var customer = db.customers.findOne({ _id: maxLoan.customer_id });
```

```
print("Customer Name: " + customer.name + ", Loan Amount: " + maxLoan.loan_amount + ", Address: "
+ customer.address);
```

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

```
db.customers.updateOne(
   { name: "Mr. Patil" },
   { $set: { address: "Updated Address, Pimpri" } },
   { $currentDate: { lastModified: true } }  // Optional: To track when the update was made
);

db.loans.updateMany(
   { customer_id: db.customers.findOne({ name: "Mr. Patil" })._id, loan_amount: { $gt: 100000 } },
   { $set: { updated: true } }  // Optional: If you want to mark the loans as updated
);
```

Slip 10

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 Effects</title>

   <style>

      body {

         display: flex;

         justify-content: center;

         align-items: center;

         height: 100vh;

         background-color: #f0f0f0; /* Light background color */

         font-family: Arial, sans-serif; /* Font style */

      }
```

```
.card {
    width: 300px;
    height: 200px;
    background-color: #007bff; /* Initial background color */
    color: white; /* Text color */
    display: flex;
    justify-content: center;
    align-items: center;
    border-radius: 10px; /* Rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Initial shadow */
    transition: background-color 0.5s ease, transform 0.5s ease, box-shadow 0.5s ease; /* Transition properties */
}

.card:hover {
    background-color: #0056b3; /* Background color on hover */
    transform: scale(1.05); /* Scale effect on hover */
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3); /* Shadow effect on hover */
}

.card:active {
    transition: transform 0.1s ease; /* Faster transition on click */
    transform: scale(0.95); /* Scale down effect on click */
}

.delayed-transition {
    transition: background-color 1s ease 0.5s; /* 1 second duration with 0.5 second delay */
}
```

```
      .delayed-transition:hover {
         background-color: #28a745; /* Change color on hover with delay */
      }
   </style>
</head>
<body>

<div class="card">
   <div class="content">
      Hover over me!
   </div>
</div>

<div class="card delayed-transition" style="margin-top: 20px;">
   <div class="content">
      Hover over me (with delay)!
   </div>
</div>

</body>
</html>
```

1. 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]

3. Insert at least 5 documents in each collection. [3]

4. Answer the following Queries.

a. List the names of product whose warranty period is one year [3]

b. List the customers has done purchase on “15/08/2023”. [3]

c. Display the names of products with brand which have highest rating. [4]

d. Display customers who stay in …… city and billamt >50000 .[4

1. Insert 5 documents in the Products collection:

```
db.products.insertMany([
   { _id: 1, name: "Smartphone", brand_id: 1, warranty_period: 1, rating: 4.5 },
   { _id: 2, name: "Laptop", brand_id: 2, warranty_period: 2, rating: 4.7 },
   { _id: 3, name: "Headphones", brand_id: 3, warranty_period: 1, rating: 4.2 },
   { _id: 4, name: "Smartwatch", brand_id: 1, warranty_period: 1, rating: 4.8 },
   { _id: 5, name: "Tablet", brand_id: 2, warranty_period: 1, rating: 4.1 }
]);
```

2. Insert 5 documents in the Brands collection:

```
db.brands.insertMany([
   { _id: 1, name: "TechBrand A", average_rating: 4.7 },
   { _id: 2, name: "TechBrand B", average_rating: 4.5 },
   { _id: 3, name: "AudioBrand C", average_rating: 4.2 },
   { _id: 4, name: "GadgetBrand D", average_rating: 4.6 },
   { _id: 5, name: "WearableBrand E", average_rating: 4.9 }
]);
```

3. Insert 5 documents in the Customers collection:

```
db.customers.insertMany([
   { _id: 1, name: "John Doe", city: "Pune", purchases: [{ product_id: 1, purchase_date: new Date("2023-
08-15"), bill_amount: 45000 }] },
   { _id: 2, name: "Jane Smith", city: "Mumbai", purchases: [{ product_id: 2, purchase_date: new
Date("2023-07-20"), bill_amount: 75000 }] },
   { _id: 3, name: "David Brown", city: "Nashik", purchases: [{ product_id: 3, purchase_date: new
Date("2023-08-15"), bill_amount: 15000 }] },
   { _id: 4, name: "Emily Davis", city: "Pune", purchases: [{ product_id: 4, purchase_date: new
Date("2023-06-15"), bill_amount: 25000 }] },
   { _id: 5, name: "Michael Wilson", city: "Mumbai", purchases: [{ product_id: 5, purchase_date: new
Date("2023-05-10"), bill_amount: 60000 }] }
]);
```

Step 3: Queries

a. List the names of products whose warranty period is one year

```
db.products.find({ warranty_period: 1 }).forEach(product => {
```

```
    print("Product Name: " + product.name);
});
```

b. List the customers who made purchases on “15/08/2023”

```
db.customers.find({ "purchases.purchase_date": new Date("2023-08-15") }).forEach(customer => {
    print("Customer Name: " + customer.name + ", City: " + customer.city);
});
```

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

```
var highestRatedBrand = db.brands.find().sort({ average_rating: -1 }).limit(1).toArray()[0];

db.products.find({ brand_id: highestRatedBrand._id }).forEach(product => {
    print("Product Name: " + product.name + ", Brand: " + highestRatedBrand.name);
});
```

d. Display customers who stay in a specific city and have a bill amount greater than 50000

```
var targetCity = "Pune";  // Change this to the desired city

db.customers.find({
    city: targetCity,
    "purchases.bill_amount": { $gt: 50000 }
}).forEach(customer => {
    print("Customer Name: " + customer.name + ", City: " + customer.city);
});
```

Slip 13

Create a useful web with the following information and structure using HTML5 tags like ,

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Online Bookstore</title>

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

    <style>

        body {
```

```
        background-color: #f8f9fa; /* Light background color */
    }
    footer {
        background-color: #343a40; /* Dark footer */
        color: white; /* White text */
        padding: 20px 0; /* Padding for footer */
    }
    nav {
        background-color: #007bff; /* Blue navigation */
    }
    nav a {
        color: white; /* White links */
        padding: 15px; /* Padding for nav links */
        text-decoration: none; /* No underline */
    }
    nav a:hover {
        background-color: #0056b3; /* Darker blue on hover */
    }
    .aside {
        background-color: #e9ecef; /* Light aside background */
        padding: 20px; /* Padding inside aside */
        border-radius: 5px; /* Rounded corners */
    }
  </style>
</head>
<body>

<header class="text-center py-4">
  <h1>Welcome to Our Online Bookstore</h1>
```

```
    <p>Your one-stop shop for all your reading needs!</p>
</header>

<nav class="navbar navbar-expand-lg">
    <div class="container">
        <a class="navbar-brand" href="#">Bookstore</a>
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link" href="#">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Categories</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Best Sellers</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Contact Us</a>
                </li>
            </ul>
        </div>
    </div>
</nav>

<main class="container mt-4">
    <section>
        <h2>Featured Books</h2>
        <div class="row">
```

```
<div class="col-md-4">
  <div class="card mb-4">
    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Book Cover">
    <div class="card-body">
      <h5 class="card-title">Book Title 1</h5>
      <p class="card-text">Author: Author Name 1</p>
      <p class="card-text">Price: $19.99</p>
      <a href="#" class="btn btn-primary">Add to Cart</a>
    </div>
  </div>
</div>
<div class="col-md-4">
  <div class="card mb-4">
    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Book Cover">
    <div class="card-body">
      <h5 class="card-title">Book Title 2</h5>
      <p class="card-text">Author: Author Name 2</p>
      <p class="card-text">Price: $14.99</p>
      <a href="#" class="btn btn-primary">Add to Cart</a>
    </div>
  </div>
</div>
<div class="col-md-4">
  <div class="card mb-4">
    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Book Cover">
    <div class="card-body">
      <h5 class="card-title">Book Title 3</h5>
      <p class="card-text">Author: Author Name 3</p>
      <p class="card-text">Price: $24.99</p>
```

```
                <a href="#" class="btn btn-primary">Add to Cart</a>
              </div>
            </div>
          </div>
        </div>
      </section>

      <aside class="mt-4">
        <h2>About Us</h2>
        <p>We are dedicated to providing a wide selection of books at great prices. Our mission is to promote reading and literacy in our community.</p>
      </aside>
    </main>

    <footer class="text-center">
      <p>&copy; 2024 Online Bookstore. All rights reserved.</p>
      <p>Contact us: info@onlinebookstore.com</p>
    </footer>

    <!-- Bootstrap JS and dependencies -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

  </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]

3. Insert at least 10 documents in each collection. [3]

4. Answer the following Queries.

a. Display average no. of students participating in each competition.[3]

b. Find the number of student for programming competition. [3]

c. Display the names of first three winners of each competition. [4]

d. Display students from class 'FY' and participated in 'E-Rangoli ' Competition. [4]

1. Insert 10 documents in the Students collection:

```
db.students.insertMany([
   { _id: 1, name: "Alice Johnson", class: "FY", email: "alice.j@example.com" },
   { _id: 2, name: "Bob Smith", class: "FY", email: "bob.s@example.com" },
   { _id: 3, name: "Charlie Brown", class: "SY", email: "charlie.b@example.com" },
   { _id: 4, name: "Daisy Miller", class: "FY", email: "daisy.m@example.com" },
   { _id: 5, name: "Ethan Wilson", class: "TY", email: "ethan.w@example.com" },
   { _id: 6, name: "Fiona Lee", class: "FY", email: "fiona.l@example.com" },
   { _id: 7, name: "George Taylor", class: "SY", email: "george.t@example.com" },
   { _id: 8, name: "Hannah Clark", class: "TY", email: "hannah.c@example.com" },
   { _id: 9, name: "Isaac Green", class: "FY", email: "isaac.g@example.com" },
   { _id: 10, name: "Jack White", class: "SY", email: "jack.w@example.com" }
]);
```

2. Insert 5 documents in the Competitions collection:

```
db.competitions.insertMany([
   { _id: 1, name: "Coding Challenge", date: new Date("2023-09-15"), winners: [] },
   { _id: 2, name: "E-Rangoli", date: new Date("2023-09-20"), winners: [] },
   { _id: 3, name: "Debate Competition", date: new Date("2023-09-25"), winners: [] },
   { _id: 4, name: "Art Fest", date: new Date("2023-09-30"), winners: [] },
   { _id: 5, name: "Sports Meet", date: new Date("2023-10-05"), winners: [] }
]);
```

3. Insert 10 documents in the Participation collection:

```
db.participation.insertMany([
   { _id: 1, student_id: 1, competition_id: 1, rank: 1 },
   { _id: 2, student_id: 2, competition_id: 1, rank: 2 },
   { _id: 3, student_id: 3, competition_id: 2, rank: 1 },
   { _id: 4, student_id: 1, competition_id: 2, rank: 3 },
   { _id: 5, student_id: 4, competition_id: 3, rank: 1 },
   { _id: 6, student_id: 5, competition_id: 3, rank: 2 },
   { _id: 7, student_id: 6, competition_id: 4, rank: 1 },
   { _id: 8, student_id: 7, competition_id: 4, rank: 2 },
   { _id: 9, student_id: 8, competition_id: 5, rank: 1 },
   { _id: 10, student_id: 9, competition_id: 5, rank: 2 }
]);
```

Step 3: Queries

a. Display the average number of students participating in each competition

```
db.competitions.aggregate([
  {
    $lookup: {
      from: "participation",
      localField: "_id",
      foreignField: "competition_id",
      as: "participants"
    }
  },
  {
    $project: {
      name: 1,
      average_participants: { $avg: { $size: "$participants" } }
    }
  }
]).forEach(competition => {
  print("Competition Name: " + competition.name + ", Average Participants: " +
competition.average_participants);
});
```

b. Find the number of students for the programming competition

```
var programmingCompetition = db.competitions.findOne({ name: "Coding Challenge" });
var participantCount = db.participation.countDocuments({ competition_id:
programmingCompetition._id });

print("Number of students in the 'Coding Challenge' competition: " + participantCount);
```

c. Display the names of the first three winners of each competition

```
db.competitions.find().forEach(competition => {
  print("Competition Name: " + competition.name);
  var winners = db.participation.find({ competition_id: competition._id }).sort({ rank: 1 }).limit(3);

  winners.forEach(winner => {
    var student = db.students.findOne({ _id: winner.student_id });
    print("Winner: " + student.name + ", Rank: " + winner.rank);
  });
});
```

d. Display students from class 'FY' who participated in 'E-Rangoli' competition

```
var eRangoliCompetition = db.competitions.findOne({ name: "E-Rangoli" });
```

```
db.students.find({ class: "FY", _id: { $in: db.participation.find({ competition_id: eRangoliCompetition._id
}).map(p => p.student_id) } })
.forEach(student => {
  print("Student Name: " + student.name + ", Class: " + student.class);
});
```

Slip 18

Create a web page, place an image in center and apply 2d transformation on it. (rotation, scaling, translation)

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>2D Transformations on Image</title>

  <style>

    body {

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      margin: 0;

      background-color: #f0f0f0; /* Light background color */

      font-family: Arial, sans-serif; /* Font style */

    }


    .image-container {

      position: relative; /* Allows for positioning child elements */

    }
```

```
      .image {
         width: 300px; /* Set image width */
         transition: transform 0.5s ease; /* Smooth transition for transformations */
      }

      /* Apply transformations on hover */
      .image-container:hover .image {
         transform: translate(20px, 20px) rotate(15deg) scale(1.2); /* Translation, rotation, and scaling */
      }

      .caption {
         position: absolute; /* Position caption over the image */
         top: 10px;
         left: 10px;
         color: white; /* White text */
         background-color: rgba(0, 0, 0, 0.7); /* Semi-transparent background */
         padding: 5px 10px; /* Padding for caption */
         border-radius: 5px; /* Rounded corners for caption */
      }
   </style>
</head>
<body>

<div class="image-container">
   <img src="https://via.placeholder.com/300" alt="Placeholder Image" class="image">
   <div class="caption">Hover over me!</div>
</div>

</body>
```

</html>

Model the following Doctor's information system as a graph model, and answer the following queries using Cypher. Consider the doctors in and around Pune. Each Doctor is specialized in some stream like Pediatric, Gynaec, Heart Specialist, Cancer Specialist, ENT, etc. A doctor may be a visiting doctor across many hospitals or he may own a clinic. A person can provide a review/can recommend a doctor.

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, and visualize your actual Graph model. [3]
3. Answer the Queries

a. List the Orthopedic doctors in ……. Area. [3]

b. List the doctors who has specialization in ____ [3]

c. List the most recommended Pediatrics in Seren Medows. [4]

d. List all the who visits more than 2 hospitals [4

High-Level Graph Model

```
 (:Doctor {name: "Dr. Smith", specialization: "Orthopedic", area: "Pune", experience_years: 10})
  -[:VISITS {days_per_week: 3}]->
(:Hospital {name: "HealthCare Hospital", location: "Pune", type: "multi-specialty"})

(:Doctor {name: "Dr. Johnson", specialization: "Pediatrics", area: "Pune", experience_years: 5})
  -[:OWNS]->
(:Clinic {name: "Kids Clinic", location: "Pune", type: "private"})

(:Patient {name: "Alice", age: 30, gender: "female"})
  -[:RECOMMENDS]->
(:Doctor {name: "Dr. Johnson"})
  -[:HAS_REVIEW]->
(:Review {rating: 5, comment: "Excellent care", date: "2023-10-01"})
```

Step 2: Create Nodes and Relationships

Assuming we are using Neo4j, below is how you can create nodes and relationships with properties.

```
// Create Doctors
CREATE (:Doctor {name: "Dr. Smith", specialization: "Orthopedic", area: "Pune", experience_years: 10}),
      (:Doctor {name: "Dr. Johnson", specialization: "Pediatrics", area: "Pune", experience_years: 5}),
      (:Doctor {name: "Dr. Patel", specialization: "Heart Specialist", area: "Pune", experience_years: 15}),
      (:Doctor {name: "Dr. Verma", specialization: "Cancer Specialist", area: "Pune", experience_years: 8}),
      (:Doctor {name: "Dr. Gupta", specialization: "ENT", area: "Pune", experience_years: 6});

// Create Hospitals
CREATE (:Hospital {name: "HealthCare Hospital", location: "Pune", type: "multi-specialty"}),
      (:Hospital {name: "City Hospital", location: "Pune", type: "general"}),
```

```
    (:Hospital {name: "Heart Institute", location: "Pune", type: "specialty"}),
    (:Hospital {name: "Cancer Center", location: "Pune", type: "specialty"});

// Create Clinics
CREATE (:Clinic {name: "Kids Clinic", location: "Pune", type: "private"}),
    (:Clinic {name: "Wellness Clinic", location: "Pune", type: "private"});

// Create Patients
CREATE (:Patient {name: "Alice", age: 30, gender: "female"}),
    (:Patient {name: "Bob", age: 40, gender: "male"});

// Create Relationships
MATCH (d:Doctor {name: "Dr. Smith"}), (h:Hospital {name: "HealthCare Hospital"})
CREATE (d)-[:VISITS {days_per_week: 3}]->(h);

MATCH (d:Doctor {name: "Dr. Johnson"}), (c:Clinic {name: "Kids Clinic"})
CREATE (d)-[:OWNS]->(c);

MATCH (d:Doctor {name: "Dr. Johnson"}), (p:Patient {name: "Alice"})
CREATE (p)-[:RECOMMENDS]->(d);

MATCH (d:Doctor {name: "Dr. Johnson"}), (r:Review {rating: 5, comment: "Excellent care", date: "2023-
10-01"})
CREATE (d)-[:HAS_REVIEW]->(r);

// Repeat similar CREATE statements for other doctors, clinics, and relationships.
```

Step 3: Answering Queries

a. List the Orthopedic doctors in a specific area

```
MATCH (d:Doctor {specialization: "Orthopedic", area: "Pune"})
RETURN d.name AS Orthopedic_Doctor;
```

b. List the doctors who have a specialization in a specific stream

```
MATCH (d:Doctor {specialization: "Gynaec"})
RETURN d.name AS Gynaec_Doctors;
```

c. List the most recommended Pediatrics in Seren Medows

```
MATCH (p:Patient)-[:RECOMMENDS]->(d:Doctor {specialization: "Pediatrics", area: "Seren Medows"})
RETURN d.name AS Most_Recommended_Pediatrics;
```

d. List all doctors who visit more than 2 hospitals

```
MATCH (d:Doctor)-[:VISITS]->(h:Hospital)
WITH d, COUNT(h) AS hospitalCount
```

```
WHERE hospitalCount > 2
RETURN d.name AS Doctors_Visiting_More_Than_2_Hospitals;
```

Slip 21

Write a css3 script for the student registration form with appropriate message display also high light compulsory fields in a different color

```
<!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>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css"
rel="stylesheet">
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f8f9fa; /* Light background color */
      margin: 0;
      padding: 20px;
    }

    .container {
      max-width: 600px;
      margin: 0 auto; /* Center the container */
      background-color: #fff; /* White background for the form */
      padding: 20px;
      border-radius: 8px; /* Rounded corners */
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1); /* Shadow for depth */
    }

    h2 {
      text-align: center; /* Center the heading */
      color: #333; /* Dark text color */
    }

    label {
      display: block;
      margin-bottom: 8px; /* Space between label and input */
      color: #333; /* Dark text color */
    }

    input[type="text"],
    input[type="email"],
    input[type="date"],
```

```
input[type="tel"],
select,
textarea {
    width: 100%; /* Full width inputs */
    padding: 10px;
    margin-bottom: 20px; /* Space below inputs */
    border: 1px solid #ccc; /* Light border */
    border-radius: 4px; /* Rounded corners */
    transition: border-color 0.3s; /* Transition effect for border color */
}

input:focus {
    border-color: #007bff; /* Blue border on focus */
    outline: none; /* Remove default outline */
}

.compulsory {
    color: red; /* Red color for compulsory fields */
}

.message {
    color: green; /* Green message text */
    text-align: center; /* Center the message */
    margin: 10px 0; /* Space around the message */
    display: none; /* Hidden by default */
}

.button-container {
    display: flex;
    justify-content: space-between; /* Space between buttons */
}

button {
    background-color: #007bff; /* Blue button */
    color: white; /* White text */
    padding: 10px 20px; /* Padding for buttons */
    border: none; /* No border */
    border-radius: 4px; /* Rounded corners */
    cursor: pointer; /* Pointer cursor */
    transition: background-color 0.3s; /* Transition effect for background */
}

button:hover {
    background-color: #0056b3; /* Darker blue on hover */
}

button[type="reset"] {
    background-color: #dc3545; /* Red button for reset */
```

```
      }

      button[type="reset"]:hover {
         background-color: #c82333; /* Darker red on hover */
      }
   </style>
</head>
<body>

<div class="container">
   <h2>Student Registration Form</h2>
   <form id="registrationForm">
      <label for="name">Name <span class="compulsory">*</span></label>
      <input type="text" id="name" required>

      <label for="email">Email <span class="compulsory">*</span></label>
      <input type="email" id="email" required>

      <label for="dob">Date of Birth <span class="compulsory">*</span></label>
      <input type="date" id="dob" required>

      <label for="contact">Contact No. <span class="compulsory">*</span></label>
      <input type="tel" id="contact" required>

      <label for="course">Preferred Course</label>
      <select id="course">
         <option value="BSc">BSc</option>
         <option value="BA">BA</option>
         <option value="BCom">BCom</option>
      </select>

      <label for="comments">Comments</label>
      <textarea id="comments" rows="4"></textarea>

      <div class="button-container">
         <button type="submit">Submit</button>
         <button type="reset">Reset</button>
      </div>
   </form>
   <div class="message" id="successMessage">Registration Successful!</div>
</div>

<script>
   // JavaScript to handle form submission
   document.getElementById('registrationForm').addEventListener('submit', function(event) {
      event.preventDefault(); // Prevent form from submitting normally
      document.getElementById('successMessage').style.display = 'block'; // Show success message
      this.reset(); // Reset form fields
```

```
   });
</script>

</body>
</html>
```

Model the following Medical information as a graph model, and answerthe following queries using Cypher. There are various brands of medicine like Dr. Reddy, Cipla, SunPharma etc. Their uses vary across different states in India. The uses of medicine is measuredas %, with a high use defined as >=90%, Medium Use between 50 to 90%, and Low UseEach medicine manufactures various types of medicine products like Tablet, Syrup, and Powder etc.

1. Identify the labels and relationships, along with their properties, and drawa high-level Graph model. [3]
2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [3]
3. Answer the following queries using Cypher:

a. List the names of different medicines considered in your graph [3]
b. List the medicine that are highly Used in Rajasthan. [3]
c. List the highly used tablet in Gujarat. [4]
d. List the medicine names manufacturing “Powder” [4]

High-Level Graph Model

```
 (:MedicineBrand {name: "Dr. Reddy"})
  -[:HAS_PRODUCT]->
(:MedicineProduct {type: "Tablet", use_percentage: 95})
  -[:USED_IN {usage_percentage: 95}]->
(:State {name: "Rajasthan"})

(:MedicineBrand {name: "Cipla"})
  -[:HAS_PRODUCT]->
(:MedicineProduct {type: "Syrup", use_percentage: 85})
  -[:USED_IN {usage_percentage: 85}]->
(:State {name: "Gujarat"})

(:MedicineBrand {name: "SunPharma"})
  -[:HAS_PRODUCT]->
(:MedicineProduct {type: "Powder", use_percentage: 40})
  -[:USED_IN {usage_percentage: 40}]->
(:State {name: "Maharashtra"})
```

Step 2: Create Nodes and Relationships

Using Neo4j, you can create the nodes and relationships with properties as follows:

```
// Create Medicine Brands
CREATE (:MedicineBrand {name: "Dr. Reddy"}),
       (:MedicineBrand {name: "Cipla"}),
       (:MedicineBrand {name: "SunPharma"}),
```

```
    (:MedicineBrand {name: "Zydus Cadila"}),
    (:MedicineBrand {name: "Mylan"});

// Create Medicine Products
CREATE (:MedicineProduct {type: "Tablet", use_percentage: 95}),
    (:MedicineProduct {type: "Syrup", use_percentage: 85}),
    (:MedicineProduct {type: "Powder", use_percentage: 40}),
    (:MedicineProduct {type: "Capsule", use_percentage: 75}),
    (:MedicineProduct {type: "Injection", use_percentage: 60});

// Create States
CREATE (:State {name: "Rajasthan"}),
    (:State {name: "Gujarat"}),
    (:State {name: "Maharashtra"}),
    (:State {name: "Karnataka"}),
    (:State {name: "Tamil Nadu"});

// Create Relationships between Brands and Products
MATCH (m:MedicineBrand {name: "Dr. Reddy"}), (p:MedicineProduct {type: "Tablet"})
CREATE (m)-[:HAS_PRODUCT]->(p);

MATCH (m:MedicineBrand {name: "Cipla"}), (p:MedicineProduct {type: "Syrup"})
CREATE (m)-[:HAS_PRODUCT]->(p);

MATCH (m:MedicineBrand {name: "SunPharma"}), (p:MedicineProduct {type: "Powder"})
CREATE (m)-[:HAS_PRODUCT]->(p);

// Create Relationships between Products and States with usage percentage
MATCH (p:MedicineProduct {type: "Tablet"}), (s:State {name: "Rajasthan"})
CREATE (p)-[:USED_IN {usage_percentage: 95}]->(s);

MATCH (p:MedicineProduct {type: "Syrup"}), (s:State {name: "Gujarat"})
CREATE (p)-[:USED_IN {usage_percentage: 85}]->(s);

MATCH (p:MedicineProduct {type: "Powder"}), (s:State {name: "Maharashtra"})
CREATE (p)-[:USED_IN {usage_percentage: 40}]->(s);

// Create additional relationships as needed.
```

Step 3: Answering Queries

a. List the names of different medicines considered in your graph

```
MATCH (m:MedicineBrand)
RETURN m.name AS Medicine_Brand;
```

b. List the medicines that are highly used in Rajasthan (usage >= 90%)

```
MATCH (p:MedicineProduct)-[:USED_IN]->(s:State {name: "Rajasthan"})
WHERE p.use_percentage >= 90
RETURN p.type AS Highly_Used_Medicine;
```

c. List the highly used tablet in Gujarat (usage >= 90%)

```
MATCH (p:MedicineProduct {type: "Tablet"})-[:USED_IN]->(s:State {name: "Gujarat"})
WHERE p.use_percentage >= 90
RETURN p.type AS Highly_Used_Tablet;
```

d. List the medicine names manufacturing “Powder”

```
MATCH (m:MedicineBrand)-[:HAS_PRODUCT]->(p:MedicineProduct {type: "Powder"})
RETURN m.name AS Medicine_Brand;
```

Slip 22

Create a web page to create 3D text. Apply all text effects like text shadow, text overflow, wordwrap etc

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>3D Text Effects</title>
  <style>
    body {
      display: flex;
      justify-content: center; /* Center content horizontally */
      align-items: center; /* Center content vertically */
      height: 100vh; /* Full viewport height */
      margin: 0;
      background-color: #282c34; /* Dark background color */
      color: white; /* White text color */
      font-family: Arial, sans-serif; /* Font style */
    }
```

```
.text-container {
    position: relative; /* Positioning for text shadow */
    text-align: center; /* Center text alignment */
    overflow: hidden; /* Hide overflow text */
    width: 300px; /* Fixed width to demonstrate overflow */
}

.text {
    font-size: 48px; /* Font size for the 3D effect */
    font-weight: bold; /* Bold text */
    text-shadow:
        1px 1px 0 rgba(0, 0, 0, 0.7), /* Basic shadow */
        2px 2px 0 rgba(0, 0, 0, 0.5), /* Slightly larger shadow */
        3px 3px 0 rgba(0, 0, 0, 0.3), /* Even larger shadow */
        4px 4px 0 rgba(0, 0, 0, 0.1); /* Faint shadow */
    transform: translateZ(0); /* Trigger GPU acceleration for smoother effects */
    white-space: nowrap; /* Prevent word wrapping */
    overflow: hidden; /* Hide overflow text */
}

.text-overflow {
    overflow: hidden; /* Hide overflow */
    text-overflow: ellipsis; /* Add ellipsis for overflow text */
    display: inline-block; /* Needed for text-overflow to work */
    width: 100%; /* Full width for demonstration */
}

.word-wrap {
```

```
        word-wrap: break-word; /* Allow word breaking */
        overflow-wrap: break-word; /* Break words if necessary */
        width: 100%; /* Full width for demonstration */
        margin-top: 20px; /* Space between text blocks */
        font-size: 24px; /* Font size for wrapped text */
        text-shadow: none; /* No shadow for wrapped text */
      }
  </style>
</head>
<body>

<div class="text-container">
  <div class="text">3D Text Effects</div>
  <div class="text-overflow">This text might overflow and will show an ellipsis if it does!</div>
  <div class="word-wrap">This is an example of a really long text that should wrap into multiple lines if it exceeds the width of the container. Let's see how it handles word wrapping!</div>
</div>

</body>
</html>
```

Model the following Car Showroom information as a graph model,and answer the queries using Cypher. Consider a car showroom with different models of cars like sofas Honda city, Skoda, Creta, Swift, Ertiga etc. Showroom is divided into different sections, onesection for each car model; each section is handled by a sales staff. Asales staff can handle one or more sections. Customer may enquire about car. An enquiry may result in a purchase by the customer.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. [3]

2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [3]
3. Answer the following queries:

a. List the types of cars available in the showroom. [3]

b. List the sections handled by Mr. Narayan. [3]

c. List the names of customers who have done only enquiry but not made any purchase. [4]

d. List the highly sale car model.

High-Level Graph Model

```
 (:CarModel {name: "Honda City", type: "Sedan", brand: "Honda"})
  -[:LOCATED_IN]->
(:Section {name: "Sedans", location: "Main Floor"})

(:SalesStaff {name: "Mr. Narayan", employee_id: "SS001"})
  -[:HANDLES]->
(:Section {name: "Hatchbacks", location: "Ground Floor"})

(:Customer {name: "Alice", customer_id: "C001"})
  -[:MAKES_ENQUIRY]->
(:Enquiry {date: "2023-10-01", status: "resolved"})
  -[:RESULTS_IN]->
(:Purchase {date: "2023-10-02", amount: 100000})

(:Customer {name: "Bob", customer_id: "C002"})
  -[:MAKES_ENQUIRY]->
(:Enquiry {date: "2023-10-05", status: "pending"})
```

Step 2: Create Nodes and Relationships

Using Neo4j, you can create the nodes and relationships with properties as follows:

```
// Create Car Models
CREATE (:CarModel {name: "Honda City", type: "Sedan", brand: "Honda"}),
      (:CarModel {name: "Skoda Octavia", type: "Sedan", brand: "Skoda"}),
      (:CarModel {name: "Hyundai Creta", type: "SUV", brand: "Hyundai"}),
      (:CarModel {name: "Maruti Swift", type: "Hatchback", brand: "Maruti"}),
      (:CarModel {name: "Maruti Ertiga", type: "MPV", brand: "Maruti"});

// Create Sections
CREATE (:Section {name: "Sedans", location: "Main Floor"}),
      (:Section {name: "Hatchbacks", location: "Ground Floor"}),
      (:Section {name: "SUVs", location: "Upper Floor"}),
      (:Section {name: "MPVs", location: "First Floor"});

// Create Sales Staff
CREATE (:SalesStaff {name: "Mr. Narayan", employee_id: "SS001"}),
      (:SalesStaff {name: "Ms. Sharma", employee_id: "SS002"});

// Create Customers
```

```
CREATE (:Customer {name: "Alice", customer_id: "C001"}),
    (:Customer {name: "Bob", customer_id: "C002"}),
    (:Customer {name: "Charlie", customer_id: "C003"});

// Create Enquiries
CREATE (:Enquiry {date: "2023-10-01", status: "resolved"}),
    (:Enquiry {date: "2023-10-05", status: "pending"});

// Create Purchases
CREATE (:Purchase {date: "2023-10-02", amount: 100000});

// Create Relationships
MATCH (s:SalesStaff {name: "Mr. Narayan"}), (sec:Section {name: "Hatchbacks"})
CREATE (s)-[:HANDLES]->(sec);

MATCH (m:CarModel {name: "Honda City"}), (sec:Section {name: "Sedans"})
CREATE (m)-[:LOCATED_IN]->(sec);

MATCH (c:Customer {name: "Alice"}), (e:Enquiry {date: "2023-10-01"})
CREATE (c)-[:MAKES_ENQUIRY]->(e);

MATCH (e:Enquiry {date: "2023-10-01"}), (p:Purchase {date: "2023-10-02"})
CREATE (e)-[:RESULTS_IN]->(p);

MATCH (c:Customer {name: "Alice"}), (m:CarModel {name: "Honda City"})
CREATE (c)-[:PURCHASED]->(m);

MATCH (c:Customer {name: "Bob"}), (e:Enquiry {date: "2023-10-05"})
CREATE (c)-[:MAKES_ENQUIRY]->(e);
```

Step 3: Answering Queries

a. List the types of cars available in the showroom

```
MATCH (m:CarModel)
RETURN DISTINCT m.type AS Car_Types;
```

b. List the sections handled by Mr. Narayan

```
MATCH (s:SalesStaff {name: "Mr. Narayan"})-[:HANDLES]->(sec:Section)
RETURN sec.name AS Handled_Sections;
```

c. List the names of customers who have done only enquiry but not made any purchase

```
MATCH (c:Customer)-[:MAKES_ENQUIRY]->(e:Enquiry)
WHERE NOT (c)-[:PURCHASED]->(:CarModel)
RETURN c.name AS Customers_With_Enquiry_Only;
```

d. List the highly sold car model

Assuming we define "highly sold" as a car model that has at least one purchase associated with it:

```
MATCH (c:Customer)-[:PURCHASED]->(m:CarModel)
RETURN m.name AS Highly_Sold_Car_Model, COUNT(c) AS Sales_Count
ORDER BY Sales_Count DESC
LIMIT 1;
```