Slips 1

Q1)

```
<!DOCTYPE html>
<html>
<head>
<style>
Body
{ font-family: Arial, sans-serif; }
  H1 {
    Font-size: 6pt;
    Color: black;
  }

  Form {
    Background-color: lightblue;
  }
</style>
</head>
<body>
<h1><b>Project Management</b></h1>
<form action=””> <label for=”projectname”>Project Name:</label>
<input type=”text” id=”projectname” name=”projectname”><br><br>
<label for=”assignedto”>Assigned to:</label>
<input type=”text” id=”assignedto” name=”assignedto”><br><br>

<label for=”startdate”>Start Date:</label>
```

```
<input type=”date” id=”startdate” name=”startdate”><br><br>


<label for=”enddate”>End Date:</label>

<input type=”date” id=”enddate” name=”enddate”><br><br>


<label for=”priority”>Priority:</label>

<select id=”priority” name=”priority”>

  <option value=”high”>High</option>

  <option value=”average”>Average</option>

  <option value=”low”>Low</option>

</select><br><br>


<label for=”description”>Description:</label>

<textarea id=”description” name=”description” rows=”4”
cols=”50”></textarea><br><br>


<input type=”submit” value=”Submit”>

<input type=”submit” value=”clear”>

</form>

</body>

</html>
```

Q2)

```
// Property Collection

[

 {
  “property_id”: 1,
  “area”: “Mumbai”,
  “rate”: 120000,
  “owner_id”: 101
},
{
  “property_id”: 2,
  “area”: “Nashik”,
  “rate”: 95000,
  “owner_id”: 102
},
{
  “property_id”: 3,
  “area”: “Pune”,
  “rate”: 105000,
  “owner_id”: 103
},
{
  “property_id”: 4,
  “area”: “Mumbai”,
  “rate”: 80000,
  “owner_id”: 104
},
{
  “property_id”: 5,
  “area”: “Nashik”,
  “rate”: 90000,
  “owner_id”: 105
```

```
 }
]

// Owner Collection
[
 {
  “owner_id”: 101,
  “name”: “Mr. Gupta”
 },
 {
  “owner_id”: 102,
  “name”: “Mr. Patil”
 },
 {
  “owner_id”: 103,
  “name”: “Mrs. Deshmukh”
 },
 {
  “owner_id”: 104,
  “name”: “Mr. Shah”
 },
 {
  “owner_id”: 105,
  “name”: “Mr. Patil”
 }
]
```

a. Display area-wise property details:

```
Db.property.find({}, { _id: 0, area: 1, rate: 1 })
```

b. Display property owned by ‘Mr. Patil’ having the minimum rate:

```
Db.property.find({ owner_id: db.owner.findOne({ name: “Mr. Patil” }).owner_id })
    .sort({ rate: 1 })

    .limit(1)
```

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

```
Db.owner.findOne({ owner_id: db.property.findOne({ area: “Nashik” }).owner_id })
```

d. Display the area of the property whose rate is less than 100000:

```
Db.property.find({ rate: { $lt: 100000 } }, { _id: 0, area: 1 })
```

Slip 2

Q1)

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

  <!—Bootstrap CSS →
  <link href=”path/to/bootstrap.min.css” rel=”stylesheet”>
</head>
<body>

  <div class=”container mt-5”>
    <!—Container with margin-top (mt-5) →

    <div class=”row”>
      <!—Row inside the container →

      <div class=”col-md-4”>
        <!—First Column (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>

      <div class=”col-md-4”>
        <!—Second Column (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>

      <div class=”col-md-4”>
        <!—Third Column (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 of Row →

  </div>
  <!—End of Container →
```

```
  <!—Bootstrap JS (Optional, for certain components) →
  <script src=”path/to/bootstrap.bundle.min.js”></script>
</body>
</html>
```

Q2)

```
// Newspaper Collection
[
 {
  “newspaper_id”: 1,
  “name”: “Times of India”,
  “language”: “English”,
  “publisher_id”: 101,
  “city”: “Mumbai”,
  “state”: “Maharashtra”,
  “sale_count”: 50000
 },
 {
  “newspaper_id”: 2,
  “name”: “Lokmat”,
  “language”: “Marathi”,
  “publisher_id”: 102,
  “city”: “Nashik”,
  “state”: “Maharashtra”,
  “sale_count”: 30000
},
{
  “newspaper_id”: 3,
  “name”: “Gujarat Samachar”,
  “language”: “Gujarati”,
  “publisher_id”: 103,
  “city”: “Ahmedabad”,
  “state”: “Gujarat”,
  “sale_count”: 45000
},
{
  “newspaper_id”: 4,
  “name”: “Pune Mirror”,
  “language”: “English”,
  “publisher_id”: 104,
  “city”: “Pune”,
  “state”: “Maharashtra”,
  “sale_count”: 25000
},
{
  “newspaper_id”: 5,
  “name”: “Nagpur Times”,
  “language”: “English”,
  “publisher_id”: 105,
  “city”: “Nagpur”,
  “state”: “Maharashtra”,
  “sale_count”: 20000
}
```

```
]

// Publisher Collection
[
 {
  “publisher_id”: 101,
  “name”: “Bennett, Coleman & Co. Ltd.”
 },
 {
  “publisher_id”: 102,
  “name”: “Lokmat Media Pvt. Ltd.”
 },
 {
  “publisher_id”: 103,
  “name”: “Gujarat Samachar Pvt. Ltd.”
 },
 {
  “publisher_id”: 104,
  “name”: “The Indian Express Group”
 },
 {
  “publisher_id”: 105,
  “name”: “Times Group”
 }
]

// City Collection
[
 {
  “city”: “Mumbai”,
  “state”: “Maharashtra”
 },
 {
  “city”: “Nashik”,
  “state”: “Maharashtra”
 },
 {
  “city”: “Ahmedabad”,
  “state”: “Gujarat”
 },
 {
  “city”: “Pune”,
  “state”: “Maharashtra”
 },
 {
  “city”: “Nagpur”,
  “state”: “Maharashtra”
 }
]
```

a. List all newspapers available in “NASHIK” city:

```
Db.newspaper.find({ city: “Nashik” })
```

b. List all newspapers of “Marathi” language:

```
Db.newspaper.find({ language: “Marathi” })
```

c. Count the number of publishers in “Gujarat” state:

```
Db.publisher.find({ _id: { $in: db.newspaper.distinct(“publisher_id”, { “city”:
“Ahmedabad” }) } }).count()
```

d. Write a cursor to show newspapers with the highest sale in Maharashtra state:

```
Var cursor = db.newspaper.find({ state: “Maharashtra” }).sort({ sale_count: -1 });
While (cursor.hasNext()) {
  Printjson(cursor.next());
}
```

Slip 3

Q1)

```
<!DOCTYPE html>
<html lang=”en”>
<head>
  <meta charset=”UTF-8”>
```

```
 <meta name=”viewport” content=”width=device-width, initial-scale=1, shrink-to-
fit=no”>
 <link rel=”stylesheet”
href=https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css>
 <title>Image Thumbnails</title>
</head>
<body>

<div class=”container mt-5”>
 <h2 class=”mb-4”>Image Thumbnails</h2>

 <div class=”row”>
  <!—Image 1 →
  <div class=”col-md-4”>
   <div class=”thumbnail”>
    <img src=”path/to/image1.jpg” alt=”Image 1” class=”img-fluid”>
    <div class=”caption”>
     <h4>Image 1</h4>
    </div>
   </div>
  </div>

  <!—Image 2 →
  <div class=”col-md-4”>
   <div class=”thumbnail”>
    <img src=”path/to/image2.jpg” alt=”Image 2” class=”img-fluid”>
    <div class=”caption”>
     <h4>Image 2</h4>
    </div>
   </div>
  </div>

  <!—Image 3 →
  <div class=”col-md-4”>
   <div class=”thumbnail”>
    <img src=”path/to/image3.jpg” alt=”Image 3” class=”img-fluid”>
    <div class=”caption”>
     <h4>Image 3</h4>
    </div>
   </div>
  </div>
 </div>
</div>

<!—Bootstrap JS and Popper.js (optional) →
<script src=https://code.jquery.com/jquery-3.5.1.slim.min.js></script>
<script src=https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js></script>
<script src=https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js></script>

</body>
</html>
```

Q2)

```
// Employee Collection
```

```
[
 {
  “employee_id”: 1,
  “name”: “John Doe”,
  “salary”: 75000,
  “department_id”: 101
 },
 {
  “employee_id”: 2,
  “name”: “Jane Smith”,
  “salary”: 80000,
  “department_id”: 102
 },
 {
  “employee_id”: 3,
  “name”: “Bob Johnson”,
  “salary”: 70000,
  “department_id”: 103
 },
 {
  “employee_id”: 4,
  “name”: “Alice Brown”,
  “salary”: 85000,
  “department_id”: 101
 },
 {
  “employee_id”: 5,
  “name”: “Chris Williams”,
  “salary”: 90000,
  “department_id”: 102
 }
]

// Department Collection
[
 {
  “department_id”: 101,
  “name”: “Sales”,
  “employees”: 2
 },
 {
  “department_id”: 102,
  “name”: “Marketing”,
  “employees”: 2
 },
 {
  “department_id”: 103,
  “name”: “Engineering”,
  “employees”: 1
 },
 {
  “department_id”: 104,
  “name”: “Finance”,
  “employees”: 0
 }
]
```

a. Display the name of the employee who has the highest salary:

```
Db.employee.find().sort({ salary: -1 }).limit(1).project({ _id: 0, name: 1 })
```

b. Display the biggest department with the maximum number of employees:

```
Db.department.find().sort({ employees: -1 }).limit(1)
```

c. Write a cursor which shows department-wise employee information:

```
Var cursor = db.department.find();
While (cursor.hasNext()) {
  Var department = cursor.next();
  Print(“Department: “ + department.name);
  Var employees = db.employee.find({ department_id: department.department_id });
  While (employees.hasNext()) {
    Printjson(employees.next());
  }
  Print(“------------“);
}
```

d. List all the employees who work in the Sales department and have a salary greater than 50000:

```
Db.employee.find({ department_id: 101, salary: { $gt: 50000 } })
```

Slip 4

Q1)

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

  <!—Bootstrap CSS →
  <link href=”path/to/bootstrap.min.css” rel=”stylesheet”>
</head>
<body>

  <div class=”container mt-5”>
    <!—Container with margin-top (mt-5) →

    <h2 class=”mb-4”>User Information Table</h2>

    <table class=”table”>
      <!—Bootstrap Table →

      <thead>
      <!—Table Header →
      <tr>
        <th scope=”col”>First Name</th>
        <th scope=”col”>Last Name</th>
        <th scope=”col”>Email ID</th>
      </tr>
    </thead>

    <tbody>
      <!—Table Body →
      <tr>
        <td>John</td>
        <td>Doe</td>
        <td>john.doe@example.com</td>
      </tr>
      <tr>
        <td>Jane</td>
        <td>Smith</td>
        <td>jane.smith@example.com</td>
      </tr>
      <!—Add more rows as needed →
    </tbody>

  </table>
  <!—End of Bootstrap Table →

</div>
<!—End of Container →
```

```
  <!—Bootstrap JS (Optional, for certain components) →
  <script src=”path/to/bootstrap.bundle.min.js”></script>
</body>
</html>
```

Q2)

```
// Hospital Collection
[
 {
  “hospital_id”: 1,
  “name”: “City Hospital”,
  “city”: “Nashik”,
  “specializations”: [“Pediatric”, “Gynaec”, “Orthopedic”],
  “rating”: 4.5
 },
 {
  “hospital_id”: 2,
  “name”: “Grace Medical Center”,
  “city”: “Nashik”,
  “specializations”: [“Cardiology”, “Dermatology”, “Oncology”],
  “rating”: 3.8
 },
 {
  “hospital_id”: 3,
  “name”: “Sunshine Hospital”,
  “city”: “Nashik”,
  “specializations”: [“Gynaec”, “Orthopedic”, “ENT”],
  “rating”: 4.2
 },
 // ... (additional hospitals)
]

// Person Recommendation Collection
[
 {
  “recommendation_id”: 1,
  “person_name”: “Mr. Patel”,
  “hospital_id”: 1,
  “review”: “Excellent service and facilities!”
 },
 {
  “recommendation_id”: 2,
  “person_name”: “Mrs. Sharma”,
  “hospital_id”: 2,
  “review”: “Great doctors, but parking is an issue.”
 },
 // ... (additional recommendations)
]

// Doctor Service Collection
[
 {
  “doctor_id”: 101,
  “doctor_name”: “Dr. Deshmukh”,
  “hospitals_served”: [1, 3]
 },
 {
  “doctor_id”: 102,
  “doctor_name”: “Dr. Sharma”,
  “hospitals_served”: [2]
 },
 // … (additional doctors)
]
```

a. List the names of hospitals with a particular specialization (e.g., Orthopedic):

```
Db.hospital.find({ specializations: “Orthopedic” }, { _id: 0, name: 1 })
```

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

```
Db.hospital.find({ city: “Nashik” }, { _id: 0, name: 1 })
```

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

```
Var hospitalsVisited = db.doctor_service.findOne({ doctor_name: “Dr. Deshmukh” }).hospitals_served;
Db.hospital.find({ hospital_id: { $in: hospitalsVisited } }, { _id: 0, name: 1 })
```

d. List the names of hospitals whose rating is greater than or equal to 4:

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

Slip 5

Q1)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>List of Persons</title>
 <style>
  Table {
   Border-collapse: collapse;
   Width: 50%;
   Margin: 20px;
   Border: 1px solid #ddd;
   Border-radius: 10px;
  }

  Th, td {
   Border: 1px solid #ddd;
   Text-align: center;
   Padding: 10px;
  }

  Th {
      Background-color: #f2f2f2;
    }
  </style>
</head>
<body>

<h2>List of Persons</h2>

<table>
  <thead>
    <tr>
      <th>Srno</th>
      <th>Person Name</th>
      <th>Age</th>
      <th>Country</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>John Doe</td>
      <td>30</td>
      <td>USA</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Jane Smith</td>
      <td>25</td>
```

```
   <td>Canada</td>
  </tr>
  <tr>
   <td>3</td>
   <td>Bob Johnson</td>
   <td>35</td>
   <td>UK</td>
  </tr>
 </tbody>
</table>

</body>
</html>
```

Q2)

```
// Project Collection
[
 {
  “project_id”: 1,
  “project_name”: “Sales Automation”,
  “project_type”: “IT”,
  “duration_months”: 5
 },
 {
  “project_id”: 2,
  “project_name”: “Marketing Campaign”,
  “project_type”: “Marketing”,
  “duration_months”: 2
 },
 {
  “project_id”: 3,
  “project_name”: “Infrastructure Upgrade”,
  “project_type”: “IT”,
  “duration_months”: 6
 },
 {
  “project_id”: 4,
  “project_name”: “Employee Training”,
  “project_type”: “HR”,
  “duration_months”: 4
 },
 {
  “project_id”: 5,
  “project_name”: “Product Launch”,
  “project_type”: “Marketing”,
  “duration_months”: 3
 }
]

// Employee Collection
[
 {
  “employee_id”: 101,
  “employee_name”: “Mr. Patil”,
  “projects_working_on”: [1, 3]
 },
 {
  “employee_id”: 102,
  “employee_name”: “Ms. Deshmukh”,
  “projects_working_on”: [2, 4]
 },
 {
  “employee_id”: 103,
  “employee_name”: “Mr. Shah”,
  “projects_working_on”: [1, 5]
 },
 {
  “employee_id”: 104,
  “employee_name”: “Mrs. Gupta”,
  “projects_working_on”: [3, 4]
 },
 {
  “employee_id”: 105,
  “employee_name”: “Ms. Joshi”,
  “projects_working_on”: [2, 5]
 }
]
```

a. List all names of projects where Project_type = “Marketing”:

```
Db.project.find({ project_type: “Marketing” }, { _id: 0, project_name: 1 })
```

b. List all the projects with a duration greater than 3 months:

```
Db.project.find({ duration_months: { $gt: 3 } })
```

c. Count the number of employees working on the “Sales Automation” project:

```
Db.employee.find({ projects_working_on: 1 }).count()
```

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

```
Var projectsWorkingOn = db.employee.findOne({ employee_name: “Mr. Patil” }).projects_working_on;

Db.project.find({ project_id: { $in: projectsWorkingOn } }, { _id: 0, project_name: 1 })
```

Slip 6

Q1)

```
<!DOCTYPE html>

<html lang=”en”>

<head>

 <meta charset=”UTF-8”>

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

 <title>Sample Web Page</title>

 <style>

  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: 10px;

  Text-decoration: none;

  Color: #333;

}


Section {

  Margin: 20px 0;

}


Img {

  Max-width: 100%;

  Height: auto;

  Border-radius: 5px;
```

```
    }

    Table {
      Border-collapse: collapse;
      Width: 100%;
      Margin-top: 20px;
    }

    Th, td {
      Border: 1px solid #ddd;
      Padding: 10px;
      Text-align: left;
    }

    Th {
      Background-color: #f2f2f2;
    }
  </style>
</head>
<body>

  <header>
    <img src=”logo.png” alt=”Company Logo” width=”150”>
    <h1>Welcome to Our Website</h1>
  </header>

  <nav>
    <a href=”#about”>About Us</a>
    <a href=”#services”>Services</a>
    <a href=”#contact”>Contact</a>
  </nav>

  <section id=”about”>
    <h2>About Us</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla facilisi. Proin a felis eget nisi consequat feugiat.</p>
    <img src=”about_image.jpg” alt=”About Us Image”>
  </section>

  <section id=”services”>
    <h2>Our Services</h2>
    <p>We offer a wide range of services to meet your needs.</p>
    <ul>
      <li>Service 1</li>
      <li>Service 2</li>
      <li>Service 3</li>
    </ul>
  </section>

  <section id=”contact”>
    <h2>Contact Us</h2>
    <p>Feel free to reach out to us:</p>
    <address>
      Email: <a href=mailto:info@example.com>info@example.com</a><br>
      Phone: +1 (123) 456-7890
    </address>
```

```
</section>

<table>
 <thead>
  <tr>
   <th>Name</th>
   <th>Age</th>
   <th>Country</th>
  </tr>
 </thead>
 <tbody>
  <tr>
   <td>John Doe</td>
   <td>30</td>
   <td>USA</td>
  </tr>
  <tr>
   <td>Jane Smith</td>
   <td>25</td>
   <td>Canada</td>
  </tr>
  <tr>
   <td>Bob Johnson</td>
   <td>35</td>
   <td>UK</td>
  </tr>
 </tbody>
</table>
```

```
</body>
</html>
```

Q2)

```
// Customer Collection
[
 {
  “customer_id”: 1,
  “customer_name”: “John Doe”,
  “policies_taken”: [
   { “policy_id”: 101, “policy_type”: “Komal Jeevan”, “premium_amount”: 15000 },
   { “policy_id”: 102, “policy_type”: “Health Insurance”, “premium_amount”: 8000 }
  ]
 },
 {
  “customer_id”: 2,
  “customer_name”: “Jane Smith”,
  “policies_taken”: [
   { “policy_id”: 103, “policy_type”: “Term Life”, “premium_amount”: 12000 },
   { “policy_id”: 104, “policy_type”: “Komal Jeevan”, “premium_amount”: 18000 }
  ]
 },
 // … (additional customers)
]

// Policy Collection
```

```
[
 {
  “policy_id”: 101,
  “policy_type”: “Komal Jeevan”,
  “company”: “XYZ Insurance”,
  “benefit”: “Financial support for family”,
  “premium_frequency”: “Monthly”
 },
 {
  “policy_id”: 102,
  “policy_type”: “Health Insurance”,
  “company”: “ABC Insurance”,
  “benefit”: “Coverage for medical expenses”,
  “premium_frequency”: “Yearly”
 },
 {
  “policy_id”: 103,
  “policy_type”: “Term Life”,
  “company”: “XYZ Insurance”,
  “benefit”: “Fixed payout in case of death”,
  “premium_frequency”: “Half Yearly”
 },
 // … (additional policies)
]
```

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

```
Db.customer.find({ “policies_taken.policy_type”: “Komal Jeevan” })
```

b. Display the average premium amount:

```
Var totalPremium = 0;
Var totalCustomers = db.customer.count();
Db.customer.find().forEach(function(customer) {
 Customer.policies_taken.forEach(function(policy) {
  totalPremium += policy.premium_amount;
 });
});
Var averagePremium = totalPremium / totalCustomers;
Print(“Average Premium Amount: “ + averagePremium);
```

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

```
Db.policy.update(
 { “premium_frequency”: “Monthly” },
 { $mul: { “premium_amount”: 1.05 } },
 { multi: true }
)
```

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

```
Db.customer.find({ “policies_taken.policy_type”: “Half Yearly” }).count()
```

Slip 7

Q1)

```
<!DOCTYPE html>
<html lang=”en”>
<head>
  <meta charset=”UTF-8”>
  <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
  <title>3D Text Effect</title>

  <style>
    Body {
      Font-family: ‘Arial’, sans-serif;
      Background-color: #f0f0f0;
      Display: flex;
      Justify-content: center;
      Align-items: center;
      Height: 100vh;
      Margin: 0;
    }

    .three-d-text {
      Font-size: 3em;
      Font-weight: bold;
      Color: #3498db;
      Text-shadow: 4px 4px 0 #2980b9, 7px 7px 0 #2c3e50;
      Transition: transform 0.3s ease-in-out;
    }

    .three-d-text:hover {
      Transform: translate(3px, 3px);
    }
  </style>
</head>
<body>

  <div class=”three-d-text”>Hover me!</div>

</body>
</html>
```

Q2)

```
// Customer Collection
[
 {
  “customer_id”: 1,
  “first_name”: “John”,
  “last_name”: “Smith”,
  “dob”: “1990-05-15”,
  “accounts”: [
   { “account_id”: 101, “account_type”: “Savings”, “branch”: “Main”, “open_date”: “2020-01-01” },
   { “account_id”: 102, “account_type”: “Checking”, “branch”: “Downtown”, “open_date”: “2021-03-10” }
  ]
 },
 {
  “customer_id”: 2,
  “first_name”: “Sara”,
  “last_name”: “Jones”,
  “dob”: “1985-08-22”,
  “accounts”: [
   { “account_id”: 103, “account_type”: “Savings”, “branch”: “Main”, “open_date”: “2020-01-01” },
   { “account_id”: 104, “account_type”: “Loan”, “branch”: “Downtown”, “open_date”: “2022-05-20” }
  ]
 },
 // … (additional customers)
]

// Transaction Collection
[
 { “transaction_id”: 1, “account_id”: 101, “amount”: 500, “transaction_type”: “Deposit”, “date”: “2022-01-15” },
 { “transaction_id”: 2, “account_id”: 102, “amount”: -200, “transaction_type”: “Withdrawal”, “date”: “2022-02-20” },
 // … (additional transactions)
]
```

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

Db.customer.find({ “first_name”: /^S/i }, { “_id”: 0, “first_name”: 1, “last_name”: 1 })

b. List all customers who have opened an account on 1/1/2020 in the “Main” branch:

Db.customer.find({ “accounts.open_date”: “2020-01-01”, “accounts.branch”: “Main” }, { “_id”: 0, “first_name”: 1, “last_name”: 1 })

c. List the names of customers where acctype is “Savings”:

Db.customer.find({ “accounts.account_type”: “Savings” }, { “_id”: 0, “first_name”: 1, “last_name”: 1 })

d. Count the total number of loan account holders in the “Downtown” branch:

Db.customer.find({ “accounts.account_type”: “Loan”, “accounts.branch”: “Downtown” }).count()

Slip 8

Q1)

```
<!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=”path/to/bootstrap.min.css” rel=”stylesheet”>
</head>
<body>

  <div class=”container mt-5”>
    <!—Container with margin-top (mt-5) →

    <h2 class=”mb-4”>Bootstrap Button Styles</h2>

    <!—Button Styles →
    <button type=”button” class=”btn btn-secondary mr-2”>Secondary</button>
    <button type=”button” class=”btn btn-primary mr-2”>Primary</button>
    <button type=”button” class=”btn btn-success mr-2”>Success</button>
    <button type=”button” class=”btn btn-danger mr-2”>Danger</button>
    <button type=”button” class=”btn btn-info mr-2”>Info</button>
    <button type=”button” class=”btn btn-warning mr-2”>Warning</button>
    <button type=”button” class=”btn btn-error”>Error</button>
    <!—End of Button Styles →

  </div>
  <!—End of Container →

  <!—Bootstrap JS (Optional, for certain components) →
  <script src=”path/to/bootstrap.bundle.min.js”></script>
</body>
</html>
```

Q2)

```
// Item Collection
[
 { “item_id”: 101, “item_name”: “Laptop”, “tags”: [“Electronics”, “Gadgets”], “status”: “A” },
 { “item_id”: 102, “item_name”: “Chair”, “tags”: [“Furniture”], “status”: “B” },
 { “item_id”: 103, “item_name”: “Planner”, “tags”: [“Stationery”], “status”: “C” },
 { “item_id”: 104, “item_name”: “Camera”, “tags”: [“Electronics”, “Photography”], “status”: “A” },
 { “item_id”: 105, “item_name”: “Printer”, “tags”: [“Electronics”, “Office”], “status”: “B” }
]
```

```
// Warehouse Collection
[
 { “warehouse_id”: 1, “warehouse_name”: “Main Warehouse”, “items_stock”: [{ “item_id”: 101, “quantity”: 400 }, { “item_id”: 103, “quantity”: 30 }] },
 { “warehouse_id”: 2, “warehouse_name”: “Secondary Warehouse”, “items_stock”: [{ “item_id”: 102, “quantity”: 100 }, { “item_id”: 105, “quantity”: 15 }] },
 { “warehouse_id”: 3, “warehouse_name”: “Backup Warehouse”, “items_stock”: [{ “item_id”: 104, “quantity”: 250 }, { “item_id”: 103, “quantity”: 10 }] }
]
```

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

```
Db.warehouse.find({ “items_stock.quantity”: { $gt: 300 } })
```

b. List all items which have tags less than 5:

```
Db.item.find({ “tags”: { $exists: true, $size: { $lt: 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 warehouses that keep the item “Planner” and have in-stock quantity less than 20:

```
Db.warehouse.find({ “items_stock”: { $elemMatch: { “item_id”: 103, “quantity”: { $lt: 20 } } } })
```

Slip 9

Q1)

```
<!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: 8px;
    Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  Label {
    Display: block;
    Margin-bottom: 8px;
    Font-weight: bold;
  }

  Input, select {
    Width: 100%;
    Padding: 10px;
    Margin-bottom: 16px;
```

```
      Border: 1px solid #ccc;
      Border-radius: 4px;
      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 action=”#” method=”post”>
    <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=”dateOfBirth”>Date of Birth:</label>
    <input type=”date” id=”dateOfBirth” name=”dateOfBirth” required>


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


    <label for=”course”>Select Course:</label>
    <select id=”course” name=”course” required>
      <option value=”computerScience”>Computer Science</option>
      <option value=”engineering”>Engineering</option>
      <option value=”biology”>Biology</option>
      <!—Add more options as needed →
    </select>


    <label for=”searchCollege”>Search for College:</label>
    <input type=”search” id=”searchCollege” name=”searchCollege”>


    <label for=”comments”>Additional Comments:</label>
    <textarea id=”comments” name=”comments” rows=”4”></textarea>


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


</body>
</html>
```

Q2)

```
// Customer Collection
[
 { “customer_id”: 101, “customer_name”: “John Doe”, “address”: “Main St, Pune” },
 { “customer_id”: 102, “customer_name”: “Alice Smith”, “address”: “Park St, Pimpri” },
 { “customer_id”: 103, “customer_name”: “Mr. Patil”, “address”: “Hill St, Pimpri” },
 // ... (additional customers)
]

// Loan Collection
[
 { “loan_id”: 201, “customer_id”: 101, “city”: “Pune”, “loan_type”: “Personal Loan”,
“loan_amt”: 50000 },
 { “loan_id”: 202, “customer_id”: 102, “city”: “Pimpri”, “loan_type”: “Home Loan”,
“loan_amt”: 150000 },
 { “loan_id”: 203, “customer_id”: 103, “city”: “Pimpri”, “loan_type”: “Car Loan”,
“loan_amt”: 120000 },
 // ... (additional loans)
]
```

a. List all customers whose name starts with ‘D’ character:

```
Db.customer.find({ “customer_name”: /^D/i })
```

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

```
Db.customer.find({ “address”: /Pimpri/i }).sort({ “customer_name”: -1 })
```

c. Display customer details having the maximum loan amount:

```
Var maxLoanAmount = db.loan.find().sort({ “loan_amt”: -1 }).limit(1).next().loan_amt;

Db.loan.aggregate([

 { $match: { “loan_amt”: maxLoanAmount } },

 { $lookup: { from: “customer”, localField: “customer_id”, foreignField: “customer_id”, as: “customer_info” } },

 { $unwind: “$customer_info” },

 { $project: { “customer_info.customer_id”: 1, “customer_info.customer_name”: 1, “customer_info.address”: 1, “loan_type”: 1, “loan_amt”: 1 } }

])
```

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

```
Db.customer.update(

 { “customer_name”: “Mr. Patil”, “customer_id”: { $in: db.loan.find({ “loan_amt”: { $gt: 100000 } }).distinct(“customer_id”) } },

 { $set: { “address”: “New Address” } },

 { multi: true }

)
```

Slip 10

```
<!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;
      Display: flex;
      Justify-content: center;
      Align-items: center;
      Height: 100vh;
      Margin: 0;
      Background-color: #f4f4f4;
    }

    Button {
      Padding: 10px 20px;
      Font-size: 16px;
      Border: none;
      Cursor: pointer;
      Background-color: #4caf50;
      Color: #fff;
      Border-radius: 4px;
      Transition-property: background-color, color;
      Transition-duration: 0.3s;
```

```
        Transition-delay: 0.1s;
      }

      Button:hover {
        Background-color: #45a049;
        Color: #e0e0e0;
      }
    </style>
</head>
<body>

    <button>Hover Me</button>

</body>
</html>
```

Q2)

```
// Product Collection
[
  { “product_id”: 101, “product_name”: “Laptop”, “brand”: “Dell”, “warranty_period”: “1 year”, “rating”: 4.5 },
  { “product_id”: 102, “product_name”: “Smartphone”, “brand”: “Samsung”, “warranty_period”: “2 years”, “rating”: 4.8 },
  { “product_id”: 103, “product_name”: “Headphones”, “brand”: “Sony”, “warranty_period”: “1 year”, “rating”: 4.2 },
  // … (additional products)
]
```

```
// Customer Collection
[
 { “customer_id”: 201, “customer_name”: “John Doe”, “city”: “New York” },
 { “customer_id”: 202, “customer_name”: “Alice Smith”, “city”: “Los Angeles” },
 { “customer_id”: 203, “customer_name”: “Bob Johnson”, “city”: “Chicago” },
 // … (additional customers)
]
```

```
// Purchase Collection
[
 { “purchase_id”: 301, “customer_id”: 201, “product_id”: 101, “purchase_date”: “2023-08-15”, “bill_amount”: 1200 },
 { “purchase_id”: 302, “customer_id”: 202, “product_id”: 102, “purchase_date”: “2023-08-15”, “bill_amount”: 800 },
 { “purchase_id”: 303, “customer_id”: 203, “product_id”: 103, “purchase_date”: “2023-08-15”, “bill_amount”: 150 },
 // … (additional purchases)
]
```

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

```
Db.product.find({ “warranty_period”: “1 year” }, { “_id”: 0, “product_name”: 1 })
```

b. List the customers who have made a purchase on “15/08/2023”:

```
Db.purchase.aggregate([
 { $match: { “purchase_date”: “2023-08-15” } },
 { $lookup: { from: “customer”, localField: “customer_id”, foreignField: “customer_id”,
as: “customer_info” } },
 { $unwind: “$customer_info” },
 { $project: { “customer_info.customer_id”: 1, “customer_info.customer_name”: 1,
“customer_info.city”: 1 } }
])
```

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

```
Var maxRating = db.product.find().sort({ “rating”: -1 }).limit(1).next().rating;
Db.product.find({ “rating”: maxRating }, { “_id”: 0, “product_name”: 1, “brand”: 1 })
```

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

```
Db.purchase.aggregate([
 { $match: { “bill_amount”: { $gt: 50000 } } },
 { $lookup: { from: “customer”, localField: “customer_id”, foreignField: “customer_id”,
as: “customer_info” } },
 { $unwind: “$customer_info” },
 { $match: { “customer_info.city”: “New York” } },
 { $project: { “customer_info.customer_id”: 1, “customer_info.customer_name”: 1,
“customer_info.city”: 1 } }
])
```

Slip 11

Q1)

```
<!DOCTYPE html>
<html lang=”en”>
<head>
 <meta charset=”UTF-8”>
 <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
 <style>
  Body {
   Margin: 0;
   Padding: 0;
   Font-family: Arial, sans-serif;
   Display: flex;
   Flex-direction: column;
   Height: 100vh;
  }

  Header {
   Background-color: #333;
   Color: #fff;
   Text-align: center;
   Padding: 10px;
  }

  Main {
   Display: flex;
   Flex: 1;
```

```
    }

    Nav {
     Width: 200px;
     Background-color: #f0f0f0;
     Padding: 10px;
     Box-shadow: 2px 0 5px rgba(0, 0, 0, 0.1);
    }

    Nav a {
     Display: block;
     Margin-bottom: 10px;
     Text-decoration: none;
     Color: #333;
    }

    Article {
     Flex: 1;
     Padding: 20px;
    }
  </style>
</head>
<body>

  <header>
    <h1>Company Name</h1>
  </header>
```

```
<main>
 <nav>
  <a href="#" onclick="showDepartment('department1')">Department 1</a>
  <a href="#" onclick="showDepartment('department2')">Department 2</a>
  <a href="#" onclick="showDepartment('department3')">Department 3</a>
  <!—Add more departments as needed →
 </nav>

 <article id="department-info">
  <!—Department information will be displayed here →
 </article>
</main>

<script>
 Function showDepartment(department) {
  // You can replace the following line with an AJAX request to fetch department information from the server
  Const departmentInfo = getDepartmentInfo(department);

  // Display department information in the third frame (article)
  Document.getElementById('department-info').innerHTML = departmentInfo;
 }

 Function getDepartmentInfo(department) {
  // Simulated department information, replace this with actual data
  Const departmentData = {
   Department1: 'Information for Department 1',
   Department2: 'Information for Department 2',
    Department3: ‘Information for Department 3’,
    // Add more departments as needed
   };

   Return departmentData[department] || ‘Department information not available.’;
  }
 </script>

</body>
</html>
```

Q2)

```
// Product Collection
[
 { “product_id”: 101, “product_name”: “Laptop”, “price”: 1000 },
 { “product_id”: 102, “product_name”: “Smartphone”, “price”: 500 },
 { “product_id”: 103, “product_name”: “Headphones”, “price”: 100 },
 // ... (additional products)
]

// Customer Collection
[
 { “customer_id”: 201, “customer_name”: “John Doe” },
 { “customer_id”: 202, “customer_name”: “Alice Smith” },
 { “customer_id”: 203, “customer_name”: “Mr. Rajiv” },
 // ... (additional customers)
]

// Order Collection

[

  { “order_id”: 301, “customer_id”: 201, “products”: [ { “product_id”: 101, “quantity”: 2 },
{ “product_id”: 102, “quantity”: 1 } ], “order_value”: 2500, “processed”: true },

  { “order_id”: 302, “customer_id”: 202, “products”: [ { “product_id”: 103, “quantity”: 3 }
], “order_value”: 300, “processed”: false },

  // … (additional orders)

]

// Invoice Collection

[

  { “invoice_id”: 401, “order_id”: 301, “invoice_value”: 2500, “payment_status”: “Paid” },

  // … (additional invoices)

]
```

a. List all products in the inventory:

```
Db.product.find({})
```

b. List the details of orders with a value >20000:

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

```
Db.order.find({ “processed”: false })
```

d. List all the orders along with their invoice for “Mr. Rajiv”:

```
Db.order.aggregate([
  { $match: { “customer_id”: 203 } },
  { $lookup: { from: “invoice”, localField: “order_id”, foreignField: “order_id”, as: “invoice_info” } },
  { $unwind: “$invoice_info” },
  { $project: { “order_id”: 1, “order_value”: 1, “invoice_info.invoice_id”: 1, “invoice_info.invoice_value”: 1, “invoice_info.payment_status”: 1 } }
])
```

Slip 12

Q1)

```
<!DOCTYPE html>
<html lang=”en”>
<head>
  <meta charset=”UTF-8”>
  <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
  <title>Customer Registration Form</title>
  <style>
    Body {
      Font-family: Arial, sans-serif;
      Background-color: #f4f4f4;
      Margin: 0;
      Padding: 20px;
```

```
}

Form {
  Max-width: 600px;
  Margin: auto;
  Background-color: #fff;
  Padding: 20px;
  Border-radius: 8px;
  Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

Label {
  Display: block;
  Margin-bottom: 8px;
  Font-weight: bold;
}

Input, select, textarea {
  Width: 100%;
  Padding: 8px;
  Margin-bottom: 16px;
  Box-sizing: border-box;
  Border: 1px solid #ccc;
  Border-radius: 4px;
}

Textarea {
  Resize: vertical;
```

```
      Height: 100px;
    }

    Button {
      Background-color: #4caf50;
      Color: #fff;
      Padding: 10px 15px;
      Border: none;
      Border-radius: 4px;
      Cursor: pointer;
    }

    Button[type=”reset”] {
      Background-color: #f44336;
    }
  </style>
</head>
<body>

  <form id=”customerRegistrationForm”>
    <label for=”name”>Name:</label>
    <input type=”text” id=”name” name=”name” required>

    <label for=”contactNo”>Contact Number:</label>
    <input type=”tel” id=”contactNo” name=”contactNo” pattern=”[0-9]{10}” required>
    <small>Enter a 10-digit phone number.</small>

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

    <label for=”preferredDays”>Preferred Days of Purchasing:</label>
    <input type=”text” id=”preferredDays” name=”preferredDays”>

    <label for=”favoriteItem”>Favorite Item:</label>
    <select id=”favoriteItem” name=”favoriteItem”>
      <option value=”clothing”>Clothing</option>
      <option value=”electronics”>Electronics</option>
      <option value=”homeAppliances”>Home Appliances</option>
      <option value=”groceries”>Groceries</option>
    </select>

    <label for=”suggestions”>Suggestions:</label>
    <textarea id=”suggestions” name=”suggestions”></textarea>

    <button type=”submit”>Submit</button>
    <button type=”reset”>Reset</button>
  </form>

</body>
</html>
```

Q2)

```
// Movie Collection

[

  { “movie_id”: 101, “movie_name”: “Inception”, “budget”: 200000000, “release_year”: 2010, “producers”: [ “Christopher Nolan”, “Emma Thomas” ], “actors”: [ { “actor_name”: “Leonardo DiCaprio”, “role”: “Cobb” }, { “actor_name”: “Joseph Gordon-Levitt”, “role”: “Arthur” } ] },

  { “movie_id”: 102, “movie_name”: “The Dark Knight”, “budget”: 250000000, “release_year”: 2008, “producers”: [ “Christopher Nolan”, “Emma Thomas” ], “actors”: [ { “actor_name”: “Christian Bale”, “role”: “Bruce Wayne” }, { “actor_name”: “Heath Ledger”, “role”: “Joker” } ] },

  // ... (additional movies)

]


// Producer Collection

[

  { “producer_name”: “Christopher Nolan”, “produced_movies”: [ { “movie_id”: 101, “movie_name”: “Inception” }, { “movie_id”: 102, “movie_name”: “The Dark Knight” } ] },

  { “producer_name”: “Emma Thomas”, “produced_movies”: [ { “movie_id”: 101, “movie_name”: “Inception” }, { “movie_id”: 102, “movie_name”: “The Dark Knight” } ] },

  // ... (additional producers)

]


// Actor Collection

[

  { “actor_name”: “Leonardo DiCaprio”, “acted_movies”: [ { “movie_id”: 101, “movie_name”: “Inception” } ] },

  { “actor_name”: “Joseph Gordon-Levitt”, “acted_movies”: [ { “movie_id”: 101, “movie_name”: “Inception” } ] },
  { “actor_name”: “Christian Bale”, “acted_movies”: [ { “movie_id”: 102, “movie_name”: “The Dark Knight” } ] },
  // … (additional actors)
]
```

a. List the names of movies with the highest budget:

```
Db.movie.find({}, { “_id”: 0, “movie_name”: 1, “budget”: 1 }).sort({ “budget”: -1 }).limit(1)
```

b. Display the details of producers who have produced more than one movie in a year:

```
Db.producer.aggregate([
 { $unwind: “$produced_movies” },
 { $group: { “_id”: { “producer_name”: “$producer_name”, “release_year”: “$produced_movies.release_year” }, “count”: { $sum: 1 } } },
 { $match: { “count”: { $gt: 1 } } },
 { $project: { “_id”: 0, “producer_name”: “$_id.producer_name”, “release_year”: “$_id.release_year”, “count”: 1 } }
])
```

c. List the names of actors who have acted in at least one movie in which ‘Akshay’ has acted:

```
Var akshayMovies = db.actor.find({ “acted_movies.actor_name”: “Akshay” }, { “_id”: 0, “acted_movies.movie_name”: 1 }).map(function(actor) {
 Return actor.acted_movies.map(function(movie) {
  Return movie.movie_name;
 });
}).flat();
```

```
Db.actor.find({ “acted_movies.movie_name”: { $in: akshayMovies } }, { “_id”: 0,
“actor_name”: 1 })
```

d. List the names of movies produced by more than one producer:

```
Db.movie.find({ “producers”: { $size: { $gt: 1 } } }, { “_id”: 0, “movie_name”: 1 })
```

Slip 13

Q1)

```
<!DOCTYPE html>
<html lang=”en”>
<head>
  <meta charset=”UTF-8”>
  <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
  <title>Fictional Tech Product</title>
  <style>
    Body {
      Font-family: Arial, sans-serif;
      Margin: 0;
      Padding: 0;
      Background-color: #f4f4f4;
```

```
}

Header {
  Background-color: #333;
  Color: #fff;
  Text-align: center;
  Padding: 10px;
}

Nav {
  Background-color: #555;
  Color: #fff;
  Padding: 10px;
}

Nav a {
  Text-decoration: none;
  Color: #fff;
  Margin: 0 15px;
}

Section {
  Max-width: 800px;
  Margin: 20px auto;
  Padding: 20px;
  Background-color: #fff;
  Border-radius: 8px;
  Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
    }

    Aside {
     Float: right;
     Width: 30%;
     Padding: 20px;
     Background-color: #ddd;
     Border-radius: 8px;
     Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    Footer {
     Background-color: #333;
     Color: #fff;
     Text-align: center;
     Padding: 10px;
     Position: absolute;
     Bottom: 0;
     Width: 100%;
    }
  </style>
</head>
<body>

  <header>
    <h1>Fictional Tech Product</h1>
  </header>
  <nav>
    <a href=”#features”>Features</a>
    <a href=”#specs”>Specifications</a>
    <a href=”#buy”>Buy Now</a>
  </nav>

  <section id=”features”>
    <h2>Features</h2>
    <p>This fictional tech product comes with amazing features to enhance your experience.</p>
    <ul>
      <li>Wireless Connectivity</li>
      <li>Long Battery Life</li>
      <li>High-Resolution Display</li>
      <li>Advanced Security</li>
    </ul>
  </section>

  <section id=”specs”>
    <h2>Specifications</h2>
    <p>Check out the technical specifications of our product:</p>
    <ul>
      <li>Processor: Quad-core, 2.0 GHz</li>
      <li>Memory: 8 GB RAM</li>
      <li>Storage: 256 GB SSD</li>
      <li>Operating System: TechOS</li>
    </ul>
  </section>
  <aside>
   <h2>Special Offer</h2>
   <p>For a limited time, get a 20% discount on your purchase. Use code: TECH20.</p>
  </aside>

  <footer>
   <p>&copy; 2023 Fictional Tech Company | Contact us at info@fictionaltech.com</p>
  </footer>

</body>
</html>
```

Q2)

```
// Competition Collection
[
 { “competition_id”: 101, “competition_name”: “Coding Challenge”, “category”: “Programming” },
 { “competition_id”: 102, “competition_name”: “E-Rangoli”, “category”: “Arts” },
 // … (additional competitions)
]

// Student Collection
[
 { “student_id”: 201, “student_name”: “John Doe”, “class”: “FY” },
 { “student_id”: 202, “student_name”: “Alice Smith”, “class”: “SY” },
 { “student_id”: 203, “student_name”: “Bob Johnson”, “class”: “FY” },
 // … (additional students)
]

// Participation Collection
[
 { “participation_id”: 301, “student_id”: 201, “competition_id”: 101, “position”: 2 },
 { “participation_id”: 302, “student_id”: 202, “competition_id”: 102, “position”: 1 },
 // … (additional participations)
]
```

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

```
Db.participation.aggregate([
 { $group: { “_id”: “$competition_id”, “average_students”: { $avg: 1 } } },
 { $lookup: { from: “competition”, localField: “_id”, foreignField: “competition_id”, as: “competition_info” } },
 { $unwind: “$competition_info” },
 { $project: { “_id”: 0, “competition_name”: “$competition_info.competition_name”, “average_students”: 1 } }
])
```

b. Find the number of students for the programming competition:

```
Db.participation.count({ “competition_id”: 101 })
```

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

```
Db.participation.aggregate([
 { $sort: { “position”: 1 } },
 { $group: { “_id”: “$competition_id”, “winners”: { $push: { “student_id”: “$student_id”,
“position”: “$position” } } } },
 { $lookup: { from: “student”, localField: “winners.student_id”, foreignField:
“student_id”, as: “winner_info” } },
 { $unwind: “$winner_info” },
 { $project: { “_id”: 0, “competition_id”: “$_id”, “competition_name”:
“$winner_info.competition_name”, “winners”: 1 } }
])
```

d. Display students from class ‘FY’ who participated in ‘E-Rangoli’ Competition:

```
Db.participation.aggregate([
 { $match: { “competition_id”: 102 } },
 { $lookup: { from: “student”, localField: “student_id”, foreignField: “student_id”, as:
“student_info” } },
 { $unwind: “$student_info” },
 { $match: { “student_info.class”: “FY” } },
 { $project: { “_id”: 0, “student_name”: “$student_info.student_name”, “class”:
“$student_info.class” } }
])
```

Slip 14

Q1)

```
<!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 Form</title>
  <style>
    Body {
      Font-family: Arial, sans-serif;
      Margin: 20px;
    }
    Form {
      Max-width: 400px;
      Margin: auto;
    }
    Label {
      Display: block;
      Margin-bottom: 8px;
    }
    Input, select {
      Width: 100%;
      Padding: 8px;
      Margin-bottom: 12px;
      Box-sizing: border-box;
    }
    Input[type="checkbox"] {
      Width: auto;
      Margin-right: 5px;
    }
    Button {
      Background-color: #4CAF50;
      Color: white;
      Padding: 10px 15px;
      Border: none;
      Border-radius: 4px;
      Cursor: pointer;
    }
    Button[type=”reset”] {
      Background-color: #f44336;
    }
  </style>
</head>
<body>

  <form id=”travelForm”>
    <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 No.:</label>
    <input type=”tel” id=”contact” name=”contact” required>
    <label>Gender:</label>

    <input type=”radio” id=”male” name=”gender” value=”male” required>

    <label for=”male”>Male</label>

    <input type=”radio” id=”female” name=”gender” value=”female” required>

    <label for=”female”>Female</label>


    <label for=”season”>Preferred Season:</label>

    <input type=”checkbox” id=”spring” name=”season” value=”spring”>

    <label for=”spring”>Spring</label>

    <input type=”checkbox” id=”summer” name=”season” value=”summer”>

    <label for=”summer”>Summer</label>

    <input type=”checkbox” id=”autumn” name=”season” value=”autumn”>

    <label for=”autumn”>Autumn</label>

    <input type=”checkbox” id=”winter” name=”season” value=”winter”>

    <label for=”winter”>Winter</label>


    <label for=”locationType”>Location Type:</label>

    <select id=”locationType” name=”locationType” required>

      <option value=”” disabled selected>Select Location Type</option>

      <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>
```

```
  <script>
    Document.getElementById(‘travelForm’).addEventListener(‘submit’, function (e) {
      e.preventDefault(); // Prevent the default form submission
      // You can add code here to handle the form submission, e.g., sending data to a server
    });
  </script>

</body>
</html>
```

Q2)

```
// Create Scholarships
CREATE (:Scholarship {name: “Merit Scholarship”})-[:FOR_CATEGORY]->(:Category {name: “General”});
CREATE (:Scholarship {name: “OBC Scholarship”})-[:FOR_CATEGORY]->(:Category {name: “OBC”});
CREATE (:Scholarship {name: “Economically Weaker Section Scholarship”})-[:FOR_CATEGORY]->(:Category {name: “EWS”});

// Create Students
CREATE (:Student {name: “John Doe”, income: 50000})-[:APPLIED_FOR]->(:Scholarship {name: “Merit Scholarship”});
CREATE (:Student {name: “Alice Smith”, income: 70000})-[:APPLIED_FOR]->(:Scholarship {name: “OBC Scholarship”});
CREATE (:Student {name: “Bob Johnson”, income: 30000})-[:APPLIED_FOR]->(:Scholarship {name: “Economically Weaker Section Scholarship”});

// Students benefitting from scholarships
MATCH (s:Student)-[:APPLIED_FOR]->(sch:Scholarship)
WHERE s.name = “John Doe” AND sch.name = “Merit Scholarship”
CREATE (s)-[:BENEFITS]->(sch);

MATCH (s:Student)-[:APPLIED_FOR]->(sch:Scholarship)
WHERE s.name = “Alice Smith” AND sch.name = “OBC Scholarship”
CREATE (s)-[:BENEFITS]->(sch);

// Students recommending others
MATCH (s1:Student {name: “John Doe”}), (s2:Student {name: “Alice Smith”})
CREATE (s1)-[:RECOMMENDS]->(s2);
```

a. List the names of scholarships for the OBC category:

```
MATCH (sch:Scholarship)-[:FOR_CATEGORY]->(cat:Category {name: “OBC”})
RETURN sch.name;
```

b. Count the number of students benefitted by a specific scholarship in the year 2020-2021 (assumed from the question context):

```
MATCH (s:Student)-[:BENEFITS]->(sch:Scholarship {name: “Merit Scholarship”})
WHERE s.income <= sch.income_limit
RETURN COUNT(s) AS numberOfStudents;
```

c. Update the income limit for a specific scholarship:

```
MATCH (sch:Scholarship {name: “Merit Scholarship”})

SET sch.income_limit = 60000;
```

d. List the most popular scholarship (assumed based on the number of students benefitted):

```
MATCH (sch:Scholarship)

RETURN sch.name, SIZE((:Student)-[:BENEFITS]->(sch)) AS popularity

ORDER BY popularity DESC

LIMIT 1;
```

Slip 15

Q1)

```
<!DOCTYPE html>

<html lang=”en”>

<head>

  <meta charset=”UTF-8”>

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

  <title>Registration Form</title>

  <link rel=”stylesheet” href=https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css>
```

```
</head>
<body>
  <div class=”container”>
    <div class=”row”>
      <div class=”col-md-6 offset-md-3”>
        <h4>Registration Form</h4>
        <form>
          <div class=”form-group”>
            <label for=”firstname”>First Name</label>
            <input type=”text” class=”form-control” id=”firstname” required>
          </div>
          <div class=”form-group”>
            <label for=”lastname”>Last Name</label>
            <input type=”text” class=”form-control” id=”lastname” required>
          </div>
          <div class=”form-group”>
            <label for=”department”>Department / Office</label>
            <select class=”form-control” id=”department” required>
              <option>IT</option>
              <option>Sales</option>
              <option>HR</option>
              <option>Marketing</option>
            </select>
          </div>
          <div class=”form-group”>
            <label for=”username”>Username</label>
            <input type=”text” class=”form-control” id=”username” required>
          </div>
          <div class=”form-group”>
            <label for=”password”>Password</label>
            <input type=”password” class=”form-control” id=”password” required>
          </div>
          <div class=”form-group”>
            <label for=”confirm-password”>Confirm Password</label>
            <input type=”password” class=”form-control” id=”confirm-password”
required>
          </div>
          <div class=”form-group”>
            <label for=”email”>E-Mail</label>
            <input type=”email” class=”form-control” id=”email” required>
          </div>
          <div class=”form-group”>
            <label for=”contact”>Contact No.</label>
            <input type=”text” class=”form-control” id=”contact” required>
          </div>
          <button type=”submit” class=”btn btn-primary”>Submit</button>
        </form>
      </div>
    </div>
  </div>

  <script src=https://code.jquery.com/jquery-3.5.1.slim.min.js></script>
  <script
src=https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js></scr
ipt>
  <script
src=https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js></script>
```

```
</body>

</html>
```

Q2)

```
// Create Movies

CREATE (:Movie {title: “Movie1”})-[:BUSINESS]->(:Business {revenue: 1000000});

CREATE (:Movie {title: “Movie2”})-[:BUSINESS]->(:Business {revenue: 1500000});


// Create Actors

CREATE (:Actor {name: “Shahrukh Khan”});

CREATE (:Actor {name: “Amitabh Bachchan”});


// Actors acting in Movies

MATCH (a:Actor {name: “Shahrukh Khan”}), (m:Movie {title: “Movie1”})

CREATE (a)-[:ACTED_IN]->(m);


MATCH (a:Actor {name: “Amitabh Bachchan”}), (m:Movie {title: “Movie2”})

CREATE (a)-[:ACTED_IN]->(m);


// Awards Received by Movies

MATCH (m:Movie {title: “Movie1”})

CREATE (m)-[:AWARD_RECEIVED]->(:Award {category: “Best Movie”});


MATCH (m:Movie {title: “Movie2”})

CREATE (m)-[:AWARD_RECEIVED]->(:Award {category: “Best Actor”});
```

a. Find the movie which made the highest business:

```
MATCH (m:Movie)-[:BUSINESS]->(b:Business)

RETURN m.title, b.revenue

ORDER BY b.revenue DESC

LIMIT 1;
```

b. Display details of a movie along with actors:

```
MATCH (m:Movie {title: “Movie1”})<-[:ACTED_IN]-(a:Actor)

RETURN m.title, COLLECT(a.name) AS actors;
```

c. List all the movies of “Shahrukh Khan”:

```
MATCH (a:Actor {name: “Shahrukh Khan”})-[:ACTED_IN]->(m:Movie)

RETURN m.title;
```

d. Display all movies having more than 2 awards received:

```
MATCH (m:Movie)-[:AWARD_RECEIVED]->(a:Award)

WITH m, COUNT(a) AS awardCount

WHERE awardCount > 2

RETURN m.title, awardCount;
```

Slip 16

Q1)

Q2)

```
// Create Customers
CREATE (:Customer {name: “John Doe”});
CREATE (:Customer {name: “Samantha”});

// Create Orders
CREATE (:Order {orderDate: “1/1/2023”});

// Create Restaurants
CREATE (:Restaurant {name: “Restaurant1”, area: “Area1”, rating: 4.5});
CREATE (:Restaurant {name: “Restaurant2”, area: “Area2”, rating: 3.8});

// Connect Restaurants to Industries
MATCH (r:Restaurant), (i:Industry {name: “ZOMATO”})
CREATE (r)-[:CONNECTED_TO]->(i);

MATCH (r:Restaurant), (i:Industry {name: “Swiggy”})
CREATE (r)-[:CONNECTED_TO]->(i);

// Create Offers
CREATE (:Offer {discount: 10});
```

```
// Connect Customers to Orders, Restaurants, Offers

MATCH (c:Customer {name: “John Doe”}), (o:Order), (r:Restaurant {name: “Restaurant1”}), (of:Offer)

CREATE (c)-[:PLACED_ORDER]->(o)-[:ORDERED_FROM]->(r);

CREATE (c)-[:GETS_OFFER]->(of);


// Create Ratings

CREATE (:Rating {stars: 4});


// Connect Customers to Ratings

MATCH (c:Customer {name: “John Doe”}), (r:Restaurant {name: “Restaurant1”}), (ra:Rating)

CREATE (c)-[:GIVES_RATING]->(ra);

CREATE (ra)-[:FOR]->(r);


// Create Recommendations

MATCH (c1:Customer {name: “John Doe”}), (c2:Customer {name: “Samantha”})

CREATE (c1)-[:RECOMMENDS_TO]->(c2);
```

a. Count the number of customers who placed an order on “1/1/2023”:

```
MATCH (c:Customer)-[:PLACED_ORDER]->(o:Order {orderDate: “1/1/2023”})

RETURN COUNT(DISTINCT c) AS numberOfCustomers;
```

b. List the names of customers whose name starts with “S” and place orders using Swiggy:

```
MATCH (c:Customer)-[:PLACED_ORDER]->(o:Order)-[:ORDERED_FROM]->(:Restaurant)-[:CONNECTED_TO]->(:Industry {name: “Swiggy”})

WHERE c.name STARTS WITH “S”

RETURN DISTINCT c.name;
```

c. List the names of hotels with a high rating (>=4):

```
MATCH (r:Restaurant)

WHERE r.rating >= 4

RETURN r.name;
```

d. List the most recommended hotels in an area (replace “AreaX” with the specific area):

```
MATCH (c:Customer)-[:RECOMMENDS_TO]->(:Customer)-[:PLACED_ORDER]->(:Order)-[:ORDERED_FROM]->(r:Restaurant {area: “AreaX”})

RETURN r.name, COUNT(DISTINCT c) AS recommendations

ORDER BY recommendations DESC

LIMIT 1;
```

Slip 17

Q1)

```
<!DOCTYPE html>

<html>
```

```
<head>
<style>
 .box {
   Width: 300px;
   Height: 200px;
   Border: 1px solid black;
   Padding: 20px;
   Margin: 30px;
   Box-sizing: border-box;
   Background-color: orange;
 }

 .inner-box {
   Width: 100%;
   Height: 50%;
   Background-color: yellow;
 }
</style>
</head>
<body>

<div class=”box”>
 <div class=”inner-box”>
   <p>M.Sc(computer sci)</p>
   <p>Academic Year 2023-24</p>
 </div>
 <div class=”inner-box”>
   <!—You can add your content here →
 </div>
</div>

</body>
</html>
```

Q2)

```
// Create Authors
CREATE (:Author {name: “Author1”});
CREATE (:Author {name: “Author2”});

// Create Books
CREATE (:Book {title: “Comics”})-[:WROTE]->(:Author {name: “Author1”});
CREATE (:Book {title: “Mystery”})-[:WROTE]->(:Author {name: “Author2”});

// Create Publishers
CREATE (:Publisher {name: “Sage”});
CREATE (:Publisher {name: “Nova”});

// Connect Books to Publishers
MATCH (b:Book {title: “Comics”}), (p:Publisher {name: “Sage”})
CREATE (b)-[:PUBLISHED]->(p);

MATCH (b:Book {title: “Mystery”}), (p:Publisher {name: “Nova”})
CREATE (b)-[:PUBLISHED]->(p);
```

```
// Create Readers
CREATE (:Reader {name: “Reader1”});
CREATE (:Reader {name: “Reader2”});

// Connect Readers to Books
MATCH (r:Reader {name: “Reader1”}), (b:Book {title: “Comics”})
CREATE (r)-[:READ]->(b);

MATCH (r:Reader {name: “Reader2”}), (b:Book {title: “Mystery”})
CREATE (r)-[:READ]->(b);

// Recommendations and Reviews
MATCH (r:Reader {name: “Reader1”}), (b:Book {title: “Comics”})
CREATE (r)-[:RECOMMENDED]->(b);

MATCH (r:Reader {name: “Reader2”}), (b:Book {title: “Mystery”})
CREATE (r)-[:REVIEWED]->(:Review {rating: 4});
```

a. List the names of authors who wrote “Comics”:

```
MATCH (a:Author)-[:WROTE]->(b:Book {title: “Comics”})
RETURN DISTINCT a.name;
```

b. Count the number of readers of a specific book published by “Sage” (replace “BookTitle” with the specific book title):

```
MATCH (p:Publisher {name: “Sage”})<-[:PUBLISHED]-(b:Book {title: “BookTitle”})<-[:READ]-(r:Reader)

RETURN COUNT(DISTINCT r) AS numberOfReaders;
```

c. List all the publishers whose name starts with “N”:

```
MATCH (p:Publisher)

WHERE p.name STARTS WITH “N”

RETURN p.name;
```

d. List the names of people who have given a rating of (>=3) for a specific book (replace “BookTitle” with the specific book title):

```
MATCH (r:Reader)-[:REVIEWED]->(rev:Review {rating: 3})-[:OF_BOOK]->(b:Book {title: “BookTitle”})

RETURN DISTINCT r.name;
```

Slip 18

Q1)

```
<!DOCTYPE html>

<html lang=”en”>

<head>

  <meta charset=”UTF-8”>

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

  <title>2D Transformation Example</title>

  <style>
```

```
    Body {
      Display: flex;
      Align-items: center;
      Justify-content: center;
      Height: 100vh;
      Margin: 0;
    }
    Img {
      Transform-origin: center center;
      Transition: transform 0.5s ease-in-out;
    }
  </style>
</head>
<body>

  <img id=”transformImage” src=”your-image-url.jpg” alt=”Transformed Image” width=”200”>

  <script>
    Const transformImage = document.getElementById(‘transformImage’);

    // Rotate the image
    transformImage.style.transform = ‘rotate(45deg)’;

    // Scale the image
    setTimeout(() => {
      transformImage.style.transform = ‘scale(1.5)’;
    }, 1000);
    // Translate the image
    setTimeout(() => {
      transformImage.style.transform = ‘translate(50px, 50px)’;
    }, 2000);
  </script>

</body>
</html>
```

Q2)

```
// Create Doctors
CREATE (:Doctor {name: “Dr. Smith”})-[:SPECIALIZED_IN]->(:Specialization {name: “Orthopedic”});
CREATE (:Doctor {name: “Dr. Patel”})-[:SPECIALIZED_IN]->(:Specialization {name: “Pediatrics”});

// Create Hospitals
CREATE (:Hospital {name: “City Hospital”});
CREATE (:Hospital {name: “Seren Medows”});

// Connect Doctors to Hospitals
MATCH (d:Doctor {name: “Dr. Smith”}), (h:Hospital {name: “City Hospital”})
CREATE (d)-[:WORKS_IN]->(h);

MATCH (d:Doctor {name: “Dr. Patel”}), (h:Hospital {name: “Seren Medows”})
```

```
CREATE (d)-[:WORKS_IN]->(h);


// Create Reviews and Recommendations

CREATE (:Person {name: “Person1”})-[:RECOMMENDED]->(d:Doctor {name: “Dr. Smith”});

CREATE (:Person {name: “Person2”})-[:REVIEWED]->(:Review {comment: “Good experience”, rating: 4})-[:OF_DOCTOR]->(d:Doctor {name: “Dr. Patel”});
```

a. List the Orthopedic doctors in a specific area (replace “AreaX” with the specific area):

```
MATCH (d:Doctor)-[:SPECIALIZED_IN]->(:Specialization {name: “Orthopedic”})-[:WORKS_IN]->(h:Hospital {area: “AreaX”})

RETURN DISTINCT d.name;
```

b. List the doctors who specialize in a specific field (replace “SpecializationName” with the specific specialization):

```
MATCH (d:Doctor)-[:SPECIALIZED_IN]->(:Specialization {name: “SpecializationName”})

RETURN DISTINCT d.name;
```

c. List the most recommended Pediatrics in a specific hospital (replace “HospitalName” with the specific hospital):

```
MATCH (d:Doctor)-[:SPECIALIZED_IN]->(:Specialization {name: “Pediatrics”})-[:WORKS_IN]->(h:Hospital {name: “HospitalName”})

RETURN d.name, COUNT(()-[:RECOMMENDED]->(d)) AS recommendations
```

```
ORDER BY recommendations DESC

LIMIT 1;
```

d. List all doctors who visit more than 2 hospitals:

```
MATCH (d:Doctor)-[:WORKS_IN]->(h:Hospital)

WITH d, COUNT(DISTINCT h) AS hospitalCount

WHERE hospitalCount > 2

RETURN d.name;
```

Slip 19

Q1)

```
<!DOCTYPE html>

<html lang=”en”>

<head>

  <meta charset=”UTF-8”>

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

  <title>Download Page</title>

  <style>

    Body {

      Display: flex;

      Align-items: center;

      Justify-content: center;

      Height: 100vh;
      Margin: 0;
    }
    #progressBar {
      Width: 300px;
      Height: 20px;
      Background-color: #ddd;
      Border-radius: 5px;
      Overflow: hidden;
    }
    #progress {
      Height: 100%;
      Width: 0;
      Background-color: #4CAF50;
      Transition: width 0.3s ease;
    }
  </style>
</head>
<body>

  <button onclick=”startDownload()”>Start Download</button>

  <div id=”progressBar”>
    <div id=”progress”></div>
  </div>

  <script>
    Function startDownload() {
      Const progressBar = document.getElementById(‘progress’);
      Let progressValue = 0;
      Let intervalId;

      Const changeColor = () => {
        Const colors = [‘#4CAF50’, ‘#2196F3’, ‘#FF9800’];
        Const randomColor = colors[Math.floor(Math.random() * colors.length)];
        progressBar.style.backgroundColor = randomColor;
      };

      intervalId = setInterval(() => {
        progressValue += 10;
        progressBar.style.width = progressValue + ‘%’;

        if (progressValue >= 100) {
          clearInterval(intervalId);
        }

        If (progressValue % 30 === 0) {
          changeColor();
        }
      }, 1000);
    }
  </script>

</body>
</html>
```

Q2)

```
// Create Manufacturers
CREATE (:Manufacturer {name: “DELL”});
CREATE (:Manufacturer {name: “HP”});

// Create Laptops
CREATE (:Laptop {model: “Inspiron”, characteristics: “High performance, lightweight”})-
[:PRODUCES]->(:Manufacturer {name: “DELL”});
CREATE (:Laptop {model: “Spectre”, characteristics: “Ultra-thin, powerful”})-
[:PRODUCES]->(:Manufacturer {name: “HP”});

// Create Customers
CREATE (:Customer {name: “John Doe”});
CREATE (:Customer {name: “Jane Smith”});

// Customer Purchase
MATCH (c:Customer {name: “John Doe”}), (l:Laptop {model: “Inspiron”})
CREATE (c)-[:BOUGHT]->(:Purchase {purchaseDate: “26/01/2023”})-[:OF_LAPTOP]->(l);

// Recommendations and Reviews
MATCH (c:Customer {name: “Jane Smith”}), (l:Laptop {model: “Spectre”})
CREATE (c)-[:RECOMMENDS]->(:Recommendation)-[:OF_LAPTOP]->(l);

MATCH (c:Customer {name: “John Doe”}), (l:Laptop {model: “Inspiron”})
CREATE (c)-[:REVIEWED]->(:Review {comment: “Great laptop”, rating: 4})-
[:OF_LAPTOP]->(l);
```

a. List the characteristics of a specific laptop (replace “LaptopModel” with the specific laptop model):

```
MATCH (l:Laptop {model: “LaptopModel”})

RETURN l.characteristics;
```

b. List the names of customers who bought a “DELL” laptop:

```
MATCH (c:Customer)-[:BOUGHT]->(:Purchase)-[:OF_LAPTOP]->(:Laptop)-
[:PRODUCES]->(:Manufacturer {name: “DELL”})

RETURN DISTINCT c.name;
```

c. List the customers who purchased a device on a specific date (replace “26/01/2023” with the specific date):

```
MATCH (c:Customer)-[:BOUGHT]->(p:Purchase {purchaseDate: “26/01/2023”})

RETURN DISTINCT c.name;
```

d. List the most recommended device:

```
MATCH (l:Laptop)<-[:OF_LAPTOP]-(:Recommendation)<-[:RECOMMENDS]-(:Customer)

RETURN l.model, COUNT(DISTINCT (:Customer)-[:RECOMMENDS]-
>(:Recommendation)-[:OF_LAPTOP]->(l)) AS recommendations

ORDER BY recommendations DESC

LIMIT 1;
```

Slip 20

Q1)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <style>
  #progress-bar {
   Width: 100%;
   Height: 30px;
   Background-color: #f1f1f1;
   Margin-top: 20px;
   Display: none;
  }

  #progress {
   Height: 100%;
   Width: 0;
   Background-color: #4CAF50;
  }
 </style>
 <script>
  Function startDownload() {
   // Show the progress bar
   Document.getElementById("progress-bar").style.display = "block";
```

```
      // Initialize the progress bar
      Let progressBar = document.getElementById(“progress”);
      Let progressValue = 0;

      // Increase the progress by 5 every second
      Let interval = setInterval(function () {
        If (progressValue < 100) {
          progressValue += 5;
          progressBar.style.width = progressValue + “%”;
        } else {
          // Download completed, show alert
          clearInterval(interval);
          alert(“Download completed”);
        }
      }, 1000);
    }
  </script>
</head>
<body>
  <button onclick=”startDownload()”>Start Download</button>

  <div id=”progress-bar”>
    <div id=”progress”></div>
  </div>
</body>
</html>
```

Q2)

```
// Create Nursery and Items
CREATE (:Nursery {name: “GreenGarden”})-[:HAS_PLANT]->(:Plant {name: “Rose”, type: “Flowering”, quantity: 1000})
CREATE (:Nursery {name: “GreenGarden”})-[:HAS_PLANT]->(:Plant {name: “Tulip”, type: “Flowering”, quantity: 800})
CREATE (:Nursery {name: “GreenGarden”})-[:HAS_PLANT]->(:Plant {name: “Creeper”, type: “Creeping”, quantity: 600})
CREATE (:Nursery {name: “GreenGarden”})-[:HAS_FERTILIZER]->(:Fertilizer {name: “GrowthMax”})
CREATE (:Nursery {name: “GreenGarden”})-[:HAS_PRODUCT]->(:Product {name: “Gardening Gloves”})

// Create Customers
CREATE (:Customer {name: “John Doe”})-[:VISITS]->(:Nursery {name: “GreenGarden”})
CREATE (:Customer {name: “Jane Smith”})-[:VISITS]->(:Nursery {name: “GreenGarden”})

// Customer Purchases
MATCH (c:Customer {name: “John Doe”}), (p:Plant {name: “Rose”})
CREATE (c)-[:MADE_PURCHASE]->(:Purchase {purchaseDate: “2023-02-10”})-[:OF_PLANT]->(p)

MATCH (c:Customer {name: “Jane Smith”}), (p:Plant {name: “Creeper”})
CREATE (c)-[:MADE_PURCHASE]->(:Purchase {purchaseDate: “2023-02-09”})-[:OF_PLANT]->(p)

// Customer Uses App
MATCH (c:Customer {name: “John Doe”}), (a:App {name: “NurseryApp”})
CREATE (c)-[:USES_APP]->(a)
```

```
// Customer Recommendations

MATCH (c:Customer {name: “Jane Smith”}), (a:App {name: “NurseryApp”})

CREATE (c)-[:RECOMMENDS]->(:Recommendation {comment: “Great app”, rating: 5})-
[:OF_APP]->(a)
```

a. List the types of plants from your graph model:

```
MATCH (p:Plant)

RETURN DISTINCT p.type;
```

b. List the popular flowering plants:

```
MATCH (p:Plant {type: “Flowering”})

RETURN p.name, p.quantity

ORDER BY p.quantity DESC

LIMIT 5;
```

c. List the names of plants sold where qty > 500 in the last 2 days:

```
MATCH (p:Plant)-[:OF_PLANT]->(purchase:Purchase)

WHERE p.quantity > 500 AND purchase.purchaseDate >= date(“2023-02-09”)

RETURN DISTINCT p.name;
```

d. List the names of suppliers in decreasing order who supplies “Creepers”:

```
MATCH (n:Nursery)-[:HAS_PLANT]->(p:Plant {name: “Creeper”})<-[:OF_PLANT]-
(purchase:Purchase)-[:MADE_PURCHASE]->(customer:Customer)
```

```
RETURN n.name, COUNT(DISTINCT customer) AS supplierCount
ORDER BY supplierCount DESC;
```

Slip 21

Q1)

```
<!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: 0;
      Display: flex;
      Align-items: center;
      Justify-content: center;
      Height: 100vh;
    }
    Form {
      Background-color: #fff;
      Padding: 20px;
      Border-radius: 8px;
      Box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    Label {
      Display: block;
      Margin-bottom: 8px;
      Font-weight: bold;
    }
    Input {
      Width: 100%;
      Padding: 8px;
      Margin-bottom: 12px;
      Box-sizing: border-box;
    }
    Input[type=”submit”] {
      Background-color: #4CAF50;
      Color: white;
      Padding: 10px 15px;
      Border: none;
      Border-radius: 4px;
      Cursor: pointer;
    }
    Input[type=”reset”] {
      Background-color: #f44336;
      Color: white;
      Padding: 10px 15px;
      Border: none;
      Border-radius: 4px;
      Cursor: pointer;
```

```
      Margin-left: 10px;
    }
    .required {
      Color: red;
    }
    .message {
      Margin-top: 10px;
      Padding: 10px;
      Background-color: #e7f3fe;
      Border: 1px solid #4e7dcb;
      Border-radius: 4px;
      Display: none;
    }
  </style>
</head>
<body>

  <form id=”registrationForm”>
    <label for=”firstName”>First Name<span class=”required”>*</span>:</label>
    <input type=”text” id=”firstName” name=”firstName” required>

    <label for=”lastName”>Last Name<span class=”required”>*</span>:</label>
    <input type=”text” id=”lastName” name=”lastName” required>

    <label for=”email”>Email<span class=”required”>*</span>:</label>
    <input type=”email” id=”email” name=”email” required>

    <label for=”password”>Password<span class=”required”>*</span>:</label>
    <input type=”password” id=”password” name=”password” required>


    <input type=”submit” value=”Submit”>
    <input type=”reset” value=”Reset”>


    <div class=”message” id=”successMessage”>Registration Successful!</div>
    <div class=”message” id=”errorMessage”>Error submitting the form. Please try again.</div>
  </form>


  <script>
    Const registrationForm = document.getElementById(‘registrationForm’);
    Const successMessage = document.getElementById(‘successMessage’);
    Const errorMessage = document.getElementById(‘errorMessage’);


    registrationForm.addEventListener(‘submit’, function (e) {
      e.preventDefault(); // Prevent the default form submission
      // You can add code here to handle the form submission, e.g., sending data to a server


      // For demonstration purposes, show a success message
      successMessage.style.display = ‘block’;


      // Clear the form after a delay (in a real scenario, this may be replaced with appropriate logic)
      setTimeout(() => {
        registrationForm.reset();
        successMessage.style.display = ‘none’;
      }, 3000);
    });

    registrationForm.addEventListener(‘reset’, function () {
      // Reset the success message on form reset
      successMessage.style.display = ‘none’;
    });
  </script>

</body>
</html>
```

Q2)

```
// Create Brands
CREATE (:Brand {name: “Dr. Reddy”});
CREATE (:Brand {name: “Cipla”});
CREATE (:Brand {name: “SunPharma”});

// Create Medicines
CREATE (:Medicine {name: “Medicine1”})-[:MANUFACTURES]->(:Brand {name: “Dr. Reddy”})-[:USES {usePercentage: 80}]->(:State {name: “Rajasthan”})
CREATE (:Medicine {name: “Medicine2”})-[:MANUFACTURES]->(:Brand {name: “Cipla”})-[:USES {usePercentage: 95}]->(:State {name: “Rajasthan”})
CREATE (:Medicine {name: “Medicine3”})-[:MANUFACTURES]->(:Brand {name: “Cipla”})-[:USES {usePercentage: 60}]->(:State {name: “Gujarat”})

// Create ProductTypes
CREATE (:ProductType {name: “Tablet”})
CREATE (:ProductType {name: “Syrup”})

CREATE (:ProductType {name: “Powder”})


// Connect Medicines to ProductTypes

MATCH (m:Medicine {name: “Medicine1”})-[:BELONGS_TO]->(:ProductType {name: “Tablet”})

MATCH (m:Medicine {name: “Medicine2”})-[:BELONGS_TO]->(:ProductType {name: “Syrup”})

MATCH (m:Medicine {name: “Medicine3”})-[:BELONGS_TO]->(:ProductType {name: “Powder”})
```

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

```
MATCH (m:Medicine)

RETURN DISTINCT m.name;
```

b. List the medicines that are highly used in Rajasthan:

```
MATCH (m:Medicine)-[u:USES]->(s:State {name: “Rajasthan”})

WHERE u.usePercentage >= 90

RETURN m.name;
```

c. List the highly used tablets in Gujarat:

```
MATCH (m:Medicine)-[u:USES]->(s:State {name: “Gujarat”})-[:BELONGS_TO]->(:ProductType {name: “Tablet”})

WHERE u.usePercentage >= 90
```

```
RETURN m.name;
```

d. List the medicine names manufacturing “Powder”:

```
MATCH (m:Medicine)-[:BELONGS_TO]->(p:ProductType {name: “Powder”})
RETURN DISTINCT m.name;
```

Slip 22

Q1)

```
<!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 {
      Font-family: ‘Arial’, sans-serif;
      Background-color: #f0f0f0;
      Margin: 0;
      Display: flex;
      Align-items: center;
      Justify-content: center;
      Height: 100vh;
    }
```

```
.container {
   Text-align: center;
}

H1 {
   Font-size: 48px;
   Color: #333;
   Text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
   Margin-bottom: 20px;
}

P {
   Font-size: 18px;
   Line-height: 1.6;
   Color: #555;
   Margin-bottom: 30px;
}

.overflow-text {
   Overflow: hidden;
   White-space: nowrap;
   Text-overflow: ellipsis;
   Max-width: 300px;
   Margin: 0 auto;
}

.word-wrap-text {
   Word-wrap: break-word;
```

```
      Max-width: 300px;

      Margin: 0 auto;

    }

  </style>

</head>

<body>


  <div class="container">

    <h1>3D Text Effects</h1>

    <p>Explore various text effects like text shadow, text overflow, word wrap, etc.</p>


    <div class="overflow-text">

      <h2>Overflow Text: This is a long text that overflows the container and is truncated
with ellipsis.</h2>

    </div>


    <div class="word-wrap-text">

      <h2>Word Wrap Text: This is a long text that wraps onto the next line when it
reaches the container's width limit.</h2>

    </div>

  </div>


</body>

</html>
```

Q2)

```
// Create Car Showroom and Models

CREATE (:CarShowroom {name: “XYZ Showroom”})-[:HAS_MODEL]->(:CarModel {name:
“Honda City”})

CREATE (:CarShowroom {name: “XYZ Showroom”})-[:HAS_MODEL]->(:CarModel {name:
“Skoda”})

CREATE (:CarShowroom {name: “XYZ Showroom”})-[:HAS_MODEL]->(:CarModel {name:
“Creta”})

CREATE (:CarShowroom {name: “XYZ Showroom”})-[:HAS_MODEL]->(:CarModel {name:
“Swift”})

CREATE (:CarShowroom {name: “XYZ Showroom”})-[:HAS_MODEL]->(:CarModel {name:
“Ertiga”})


// Create Sales Staff and Sections

CREATE (:SalesStaff {name: “Mr. Narayan”})-[:HANDLES_SECTION]->(:Section {name:
“Honda City”})

CREATE (:SalesStaff {name: “Mr. Narayan”})-[:HANDLES_SECTION]->(:Section {name:
“Skoda”})


// Create Customers, Enquiries, and Purchases

CREATE (:Customer {name: “John Doe”})-[:ENQUIRED_ABOUT]->(:Enquiry {details:
“Interested in Honda City”})

CREATE (:Customer {name: “Jane Smith”})-[:ENQUIRED_ABOUT]->(:Enquiry {details:
“Interested in Skoda”})

CREATE (:Customer {name: “Bob”})-[:ENQUIRED_ABOUT]->(:Enquiry {details:
“Enquiring about Swift”})

CREATE (:Customer {name: “Alice”})-[:MADE_PURCHASE]->(:Purchase {details:
“Purchased Honda City”})


// Connect Customers to Sections for Purchases

MATCH (c:Customer {name: “John Doe”})-[:MADE_PURCHASE]->(p:Purchase)-
[:OF_MODEL]->(m:CarModel {name: “Honda City”})

MERGE (c)-[:ENQUIRED_ABOUT]->(:Enquiry {details: “Interested in Honda City”})
```

a. List the types of cars available in the showroom:

```
MATCH (s:CarShowroom)-[:HAS_MODEL]->(m:CarModel)

RETURN DISTINCT m.name;
```

b. List the sections handled by Mr. Narayan:

```
MATCH (s:SalesStaff {name: “Mr. Narayan”})-[:HANDLES_SECTION]->(sec:Section)

RETURN DISTINCT sec.name;
```

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

```
MATCH (c:Customer)-[:ENQUIRED_ABOUT]->(e:Enquiry)

WHERE NOT (c)-[:MADE_PURCHASE]->(:Purchase)

RETURN DISTINCT c.name;
```

d. List the highly sale car model:

```
MATCH (m:CarModel)<-[:OF_MODEL]-(p:Purchase)

RETURN m.name, COUNT(p) AS purchaseCount

ORDER BY purchaseCount DESC

LIMIT 1;
```

Slip 23

Q1)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Display with Rotation</title>
  <style>
    Body {
      Margin: 0;
      Display: flex;
      Align-items: center;
      Justify-content: center;
      Height: 100vh;
      Background-color: #f0f0f0;
    }

    #imageContainer {
      Display: flex;
      Flex-wrap: wrap;
    }

    .imageTile {
      Width: 150px;
      Height: 150px;
      Overflow: hidden;
      Border: 1px solid #ddd;
      Margin: 5px;
    }

    Img {
      Max-width: 100%;
      Max-height: 100%;
      Transform-origin: center center;
      Transition: transform 0.3s ease;
    }

    Button {
      Margin-top: 20px;
      Padding: 10px;
      Font-size: 16px;
      Cursor: pointer;
    }
  </style>
</head>
<body>

  <div id=”imageContainer”>
    <!—Replace “your-image-url.jpg” with the actual URL or path of your image →
    <div class=”imageTile”>
      <img src=”your-image-url.jpg” alt=”Image Tile 1” id=”imageTile1”>
    </div>
    <div class=”imageTile”>
      <img src=”your-image-url.jpg” alt=”Image Tile 2” id=”imageTile2”>
    </div>
    <div class=”imageTile”>
      <img src=”your-image-url.jpg” alt=”Image Tile 3” id=”imageTile3”>
    </div>
    <!—Add more image tiles as needed →
  </div>

  <button onclick=”rotateClockwise()”>Rotate Clockwise</button>
  <button onclick=”rotateAntiClockwise()”>Rotate Anti-clockwise</button>

  <script>
    Function rotateClockwise() {
      rotateImage(“imageContainer”, 90);
    }

    Function rotateAntiClockwise() {
      rotateImage(“imageContainer”, -90);
    }

    Function rotateImage(containerId, angle) {
      Const container = document.getElementById(containerId);
      Const imageTiles = container.querySelectorAll(‘.imageTile img’);

      imageTiles.forEach(img => {
        img.style.transform = ` rotate(${angle}deg)` ;
      });
    }
  </script>
```

```
</body>

</html>
```

Q2)

```
// Create Automobile Industry and Vehicle Types

CREATE (:AutomobileIndustry {name: “XYZ Automobiles”})-[:MANUFACTURES]->(:VehicleType {name: “Two-Wheeler”, characteristics: “Characteristic 1, Characteristic 2”})

CREATE (:AutomobileIndustry {name: “XYZ Automobiles”})-[:MANUFACTURES]->(:VehicleType {name: “Four-Wheeler”, characteristics: “Characteristic 3, Characteristic 4”})

CREATE (:AutomobileIndustry {name: “XYZ Automobiles”})-[:MANUFACTURES]->(:VehicleType {name: “Electric Vehicle”, characteristics: “Characteristic 5, Characteristic 6”})


// Create Customers, Purchases, and Recommendations

CREATE (:Customer {name: “John Doe”})-[:BOUGHT]->(:VehicleType {name: “Two-Wheeler”})

CREATE (:Customer {name: “Jane Smith”})-[:BOUGHT]->(:VehicleType {name: “Four-Wheeler”})

CREATE (:Customer {name: “Bob”})-[:BOUGHT]->(:VehicleType {name: “Electric Vehicle”})

CREATE (:Customer {name: “Alice”})-[:RECOMMENDS]->(:VehicleType {name: “Two-Wheeler”})

CREATE (:Customer {name: “Charlie”})-[:RECOMMENDS]->(:VehicleType {name: “Four-Wheeler”})

CREATE (:Customer {name: “David”})-[:RECOMMENDS]->(:VehicleType {name: “Electric Vehicle”})

CREATE (:Customer {name: “Eva”})-[:BOUGHT]->(:VehicleType {name: “Two-Wheeler”})
```

CREATE (:Customer {name: “Frank”})-[:BOUGHT]->(:VehicleType {name: “Four-Wheeler”})

a. List the characteristics of four-wheeler types:

MATCH (:VehicleType {name: “Four-Wheeler”}) RETURN DISTINCT characteristics;

b. List the names of customers who bought a two-wheeler vehicle:

MATCH (c:Customer)-[:BOUGHT]->(:VehicleType {name: “Two-Wheeler”}) RETURN DISTINCT c.name;

c. List the customers who bought more than one type of vehicle:

MATCH (c:Customer)-[:BOUGHT]->(vt:VehicleType)

WITH c, COLLECT(DISTINCT vt) AS vehicleTypes

WHERE SIZE(vehicleTypes) > 1

RETURN DISTINCT c.name;

d. List the most recommended vehicle type:

MATCH (vt:VehicleType)<-[:RECOMMENDS]-(c:Customer)

RETURN vt.name, COUNT(c) AS recommendationCount

ORDER BY recommendationCount DESC

LIMIT 1;

Slip 24

Q1)

```
<!DOCTYPE html>
<html>

<head>
  <title>Calendar</title>
</head>

<body>
  <h2>Input Types Examples</h2>

  <form action="">
    <label for="date">Date:</label><br>
    <input type="date" id="date" name="date"><br>

    <label for="datetime">Datetime:</label><br>
    <input type="datetime" id="datetime" name="datetime"><br>

    <label for="datetime-local">Datetime-local:</label><br>
    <input type="datetime-local" id="datetime-local" name="datetime-local"><br>

    <label for="month">Month:</label><br>
    <input type="month" id="month" name="month"><br>

    <label for="time">Time:</label><br>
    <input type="time" id="time" name="time"><br>
    <label for=”week”>Week:</label><br>
    <input type=”week” id=”week” name=”week”><br>

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

</body>

</html>
```

Q2)

```
// Create Library and Book Types
CREATE (:Library {name: “University Library”})-[:HAS_TYPE]->(:BookType {name: “Text”})
CREATE (:Library {name: “University Library”})-[:HAS_TYPE]->(:BookType {name: “Reference”})
CREATE (:Library {name: “University Library”})-[:HAS_TYPE]->(:BookType {name: “Bibliography”})

// Create Students, Purchases, and Recommendations
CREATE (:Student {name: “John Doe”})-[:BOUGHT]->(:BookType {name: “Text”})
CREATE (:Student {name: “Jane Smith”})-[:BOUGHT]->(:BookType {name: “Reference”})
CREATE (:Student {name: “Bob”})-[:BOUGHT]->(:BookType {name: “Text”})
CREATE (:Student {name: “Alice”})-[:RECOMMENDS]->(:BookType {name: “Text”})
CREATE (:Student {name: “Charlie”})-[:RECOMMENDS]->(:BookType {name: “Reference”})
CREATE (:Student {name: “David”})-[:RECOMMENDS]->(:BookType {name: “Bibliography”})
CREATE (:Student {name: “Eva”})-[:BOUGHT]->(:BookType {name: “Text”})

CREATE (:Student {name: “Frank”})-[:BOUGHT]->(:BookType {name: “Reference”})
```

a. List the books of type “text”:

```
MATCH (b:BookType {name: “Text”}) RETURN b;
```

b. List the name of the student who bought text and reference types of books:

```
MATCH (s:Student)-[:BOUGHT]->(b:BookType)

WHERE b.name IN [“Text”, “Reference”]

RETURN DISTINCT s.name;
```

c. List the most recommended book type:

```
MATCH (b:BookType)<-[:RECOMMENDS]-(s:Student)

RETURN b.name, COUNT(s) AS recommendationCount

ORDER BY recommendationCount DESC

LIMIT 1;
```

d. List the student who bought more than one type of book:

```
MATCH (s:Student)-[:BOUGHT]->(b:BookType)

WITH s, COLLECT(DISTINCT b) AS bookTypes

WHERE SIZE(bookTypes) > 1

RETURN DISTINCT s.name;
```

Slip 25

Q1)

```
<!DOCTYPE html>

<html>


<head>

  <style>

    Body {

      Font-family: Arial, sans-serif;

      Font-size: 14px;

      Color: #333;

    }


    H2 {

      Color: #4682B4;

      Font-size: 20px;

      Font-weight: bold;

    }


    .form-container {

      Background-color: #f9f9f9;

      Padding: 20px;

      Border-radius: 5px;

      Width: 500px;

      Margin: 0 auto;
```

```
}

Input[type=”text”],
Input[type=”password”] {
   Width: 100%;
   Padding: 10px;
   Margin: 5px 0 10px;
   Border: 1px solid #ddd;
   Border-radius: 3px;
}

Label {
   Display: block;
   Margin-bottom: 5px;
}

.form-container input[type=”submit”] {
   Background-color: #4CAF50;
   Color: white;
   Padding: 10px 20px;
   Margin: 10px 0;
   Border: none;
   Border-radius: 3px;
   Cursor: pointer;
   Width: 100%;
}

.form-container input[type=”submit”]:hover {
      Background-color: #45a049;
    }
  </style>
</head>

<body>
  <div class=”form-container”>
    <h2>Entry Form</h2>
    <form action=”/submit_form” method=”post”>
      <label for=”fname”>Name:</label>
      <input type=”text” id=”fname” name=”fname” required>

      <label for=”age”>Age:</label>
      <input type=”text” id=”age” name=”age” required>

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

      <label for=”sex”>Sex:</label>
      <input type=”text” id=”sex” name=”sex” required>

      <label for=”nationality”>Nationality:</label>
      <input type=”text” id=”nationality” name=”nationality” required>

      <label for=”pwd”>Password:</label>
      <input type=”password” id=”pwd” name=”pwd” required>

      <input type=”submit” value=”Submit”>
    </form>
  </div>
</body>

</html>
```

Q2)

```
// Create Departments
CREATE (:Department {name: ‘Physics’})
CREATE (:Department {name: ‘Geography’})
CREATE (:Department {name: ‘Computer’})
// Add more departments as needed

// Create Courses
CREATE (:Course {name: ‘Physics 101’})
CREATE (:Course {name: ‘Geography 202’})
CREATE (:Course {name: ‘Computer Science 301’})
// Add more courses as needed

// Create Relationships
MATCH (physicsDept:Department {name: ‘Physics’}), (physicsCourse:Course {name: ‘Physics 101’})
CREATE (physicsDept)-[:OFFERS]->(physicsCourse)

MATCH (geoDept:Department {name: ‘Geography’}), (geoCourse:Course {name: ‘Geography 202’})
```

```
CREATE (geoDept)-[:OFFERS]->(geoCourse)

// Add more relationships as needed

// Create Recommendations
CREATE (:Person {name: ‘John’})-[:RECOMMENDS]->(:Course {name: ‘Geography 202’})
CREATE (:Person {name: ‘Alice’})-[:RECOMMENDS]->(:Course {name: ‘Computer
Science 301’})
// Add more recommendations as needed
```

a. List the details of all the departments in the university.

```
MATCH (d:Department)
RETURN d;
```

b. List the names of the courses provided by the Physics department.

```
MATCH (:Department {name: ‘Physics’})-[:OFFERS]->(course:Course)
RETURN course.name;
```

c. List the most recommended course in the Geography department.

```
MATCH (:Department {name: ‘Geography’})-[:OFFERS]->(course:Course)<-
[:RECOMMENDS]-(person:Person)
RETURN course.name, COUNT(person) AS recommendations
ORDER BY recommendations DESC
LIMIT 1;
```

d. List the names of common courses across Mathematics and Computer department.

```
MATCH (mathDept:Department {name: ‘Mathematics’})-[:OFFERS]->(mathCourse:Course),
    (compDept:Department {name: ‘Computer’})-[:OFFERS]->(compCourse:Course)
WHERE mathCourse.name = compCourse.name
RETURN mathCourse.name;
```