SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 1

Q.1] Write the HTML5 code for generating the form as shown below. Apply the internal CSS to the following form to change the font size of the heading to 6pt and change the color to red and also change the background color to yellow.

=>

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Example</title>

  <style>
    h1 {
      font-size: 6pt;
      color: red;
    }
    body {
      background-color: yellow;
    }
  </style>
</head>

<body>

  <h1>Registration Form</h1>

  <form>
    <label>Name:</label>
    <input type="text" name="name"><br><br>
```

```
    <label>Email:</label>
    <input type="email" name="email"><br><br>

    <label>Password:</label>
    <input type="password" name="password"><br><br>

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

</body>
</html>
```

```
-----------------------------------------
| (Yellow Background)                   |
|                                       |
|   Registration Form  ← (6pt, red)     |
|                                       |
|   Name:     []                        |
|                                       |
|   Email:    []                        |
|                                       |
|   Password:  []                       |
|                                       |
|   [ Submit ]                          |
-----------------------------------------
```

✅ 1. Model the following Property system as a document database

We will use two collections:

Collection 1: Owners

```
{
 _id: 1,
 ownerName: "Mr.Patil",
 contact: "9876543210"
}
```

Collection 2: Properties

```
{
 _id: 101,
 area: "Nashik",
 rate: 95000,
 ownerId: 1
}
```

One owner can have many properties → ownerId is used to link.

_----------

✅ 2. Assume appropriate attributes and collections

Owner attributes

ownerName

contact

email

Property attributes

area

rate

size

ownerId

✅ 3. Insert at least 05 documents in each collection

✅ Owners Collection (5 Documents)

```
{
  _id: 1,
  ownerName: "Mr.Patil",
  contact: "9876543210",
```

```
  email: "patil@gmail.com"
}
{
  _id: 2,
  ownerName: "Mr.Shinde",
  contact: "9876500000",
  email: "shinde@gmail.com"
}
{
  _id: 3,
  ownerName: "Mr.Kulkarni",
  contact: "9876512345",
  email: "kulkarni@gmail.com"
}
{
  _id: 4,
  ownerName: "Mrs.Jadhav",
  contact: "9876523456",
  email: "jadhav@gmail.com"
}
{
  _id: 5,
  ownerName: "Mr.Desai",
  contact: "9876534567",
  email: "desai@gmail.com"
}
```

✅ Properties Collection (5 Documents)

```
{
  _id: 101,
  area: "Nashik",
  rate: 95000,
  size: "800 sqft",
  ownerId: 1
}
{
  _id: 102,
  area: "Pune",
  rate: 150000,
  size: "1200 sqft",
  ownerId: 1
}
{
  _id: 103,
  area: "Mumbai",
  rate: 300000,
  size: "1000 sqft",
  ownerId: 2
}
{
  _id: 104,
  area: "Nashik",
  rate: 80000,
  size: "600 sqft",
  ownerId: 3
}
{
  _id: 105,
```

```
  area: "Aurangabad",
  rate: 70000,
  size: "500 sqft",
  ownerId: 4
}
```

Q.2) 4. ANSWER THE QUERIES

✅ (a) Display area-wise property details

MongoDB Query:

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

✅ OUTPUT:

```
{
  _id: "Nashik",
  properties: [
    { _id: 101, rate: 95000, size: "800 sqft", ownerId: 1 },
    { _id: 104, rate: 80000, size: "600 sqft", ownerId: 3 }
  ]
}
```

```
{
  _id: "Pune",
  properties: [
    { _id: 102, rate: 150000, size: "1200 sqft", ownerId: 1 }
  ]
}
{
  _id: "Mumbai",
  properties: [
    { _id: 103, rate: 300000, size: "1000 sqft", ownerId: 2 }
  ]
}
{
  _id: "Aurangabad",
  properties: [
    { _id: 105, rate: 70000, size: "500 sqft", ownerId: 4 }
  ]
}
```

-_----------

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

Step 1: Find ownerId

OwnerId = 1

MongoDB Query

```
db.properties.find({ ownerId: 1 }).sort({ rate: 1 }).limit(1)
```

✅ OUTPUT

```
{
 _id: 101,
 area: "Nashik",
 rate: 95000,
 size: "800 sqft",
 ownerId: 1
}
```

✅ (c) Give the details of owner whose property is at “Nashik”

MongoDB Query:

```
db.properties.aggregate([
 { $match: { area: "Nashik" } },
 {
  $lookup: {
   from: "owners",
   localField: "ownerId",
   foreignField: "_id",
   as: "ownerDetails"
  }
 }
])
```

✅ OUTPUT:

```
{
  _id: 101,
  area: "Nashik",
  rate: 95000,
  ownerDetails: [
    { _id: 1, ownerName: "Mr.Patil", contact: "9876543210" }
  ]
}
{
  _id: 104,
  area: "Nashik",
  rate: 80000,
  ownerDetails: [
    { _id: 3, ownerName: "Mr.Kulkarni", contact: "9876512345" }
  ]
}
```

✅ (d) Display area of property whose rate is less than 100000

MongoDB Query:

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

✅ OUTPUT:

{ area: "Nashik", rate: 95000 }

{ area: "Nashik", rate: 80000 }

{ area: "Aurangabad", rate: 70000 }

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 2

Q1) 1. Model the system as a document database

Create three collections:

Collection: publishers

publisherName

state

contact

Collection: newspapers

newspaperName

language

publisherId

sale

cityId

Collection: cities

cityName

state

✅ 2. Attributes assumed (as required)

City attributes → cityName, state

Publisher attributes → publisherName, state

Newspaper attributes → language, sale, publisherId, cityId

✅ 3. Insert at least 5 documents in each collection

✅ Cities Collection (5 Documents)

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

✅ Publishers Collection (5 Documents)

```
{
  _id: 1,
  publisherName: "Sakal Publications",
  state: "Maharashtra",
  contact: "9998887771"
}
{
  _id: 2,
  publisherName: "Lokmat Group",
  state: "Maharashtra",
  contact: "8887776661"
}
{
  _id: 3,
  publisherName: "Gujrat Times Media",
  state: "Gujrat",
  contact: "7776665551"
}
{
  _id: 4,
  publisherName: "Divya Bhaskar",
  state: "Gujrat",
  contact: "9991115555"
}
{
  _id: 5,
```

publisherName: "Maharashtra Times",

state: "Maharashtra",

contact: "9090909090"

}

✅ Newspapers Collection (5 Documents)

{

_id: 101,

newspaperName: "Sakal",

language: "Marathi",

publisherId: 1,

sale: 50000,

cityId: 1

}

{

_id: 102,

newspaperName: "Lokmat",

language: "Marathi",

publisherId: 2,

sale: 65000,

cityId: 2

}

{

_id: 103,

newspaperName: "Divya Bhaskar",

language: "Gujarati",
publisherId: 4,
sale: 55000,
cityId: 4
}
{
_id: 104,
newspaperName: "Gujrat Times",
language: "Gujarati",
publisherId: 3,
sale: 60000,
cityId: 5
}
{
_id: 105,
newspaperName: "Maharashtra Times",
language: "Marathi",
publisherId: 5,
sale: 70000,
cityId: 3
}

Q2). ANSWER THE QUERIES

✅ (a) List all newspapers available in NASHIK city

MongoDB Query

```
db.newspapers.find({ cityId: 1 })
```

✅ EXACT OUTPUT

```
{ _id: 101, newspaperName: "Sakal", language: "Marathi", sale: 50000, cityId: 1 }
```

✅ (b) List all newspapers of Marathi language

MongoDB Query

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

✅ EXACT OUTPUT

```
{ _id: 101, newspaperName: "Sakal", language: "Marathi", sale: 50000 }
{ _id: 102, newspaperName: "Lokmat", language: "Marathi", sale: 65000 }
{ _id: 105, newspaperName: "Maharashtra Times", language: "Marathi", sale: 70000 }
```

✅ (c) Count no. of publishers of Gujrat state

MongoDB Query

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

✅ EXACT OUTPUT

2

✅ (d) Write a cursor to show newspapers with highest sale in Maharashtra state

MongoDB Cursor

```
var cur = db.newspapers.aggregate([
 {
  $lookup: {
   from: "cities",
   localField: "cityId",
   foreignField: "_id",
   as: "cityDetails"
  }
 },
 { $unwind: "$cityDetails" },
 { $match: { "cityDetails.state": "Maharashtra" } },
 { $sort: { sale: -1 } },
```

```
  { $limit: 1 }
]);

while(cur.hasNext()) {
  printjson(cur.next());
}
```

✅ EXACT OUTPUT

```
{
  _id: 105,
  newspaperName: "Maharashtra Times",
  language: "Marathi",
  sale: 70000,
  cityDetails: {
    cityName: "Mumbai",
    state: "Maharashtra"
  }
}
```

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 3

Q1). Model the following system as a document database. Consider employee and department information.

2. Assume appropriate attributes and collections as per the query requirements.

3. Insert at least 5 documents in each collection.

4. Answer the following Queries:

a. Display name of employee who has highest salary

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

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

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

✅ SOLUTION

✅ 1. Model the system as a Document Database

Collections Used:

Department Collection

dept_id

dept_name

location

Employee Collection

emp_id

emp_name

salary

dept_id

✅ 2. Attributes Assumed

Employees linked to departments using dept_id.

✅ 3. Insert at least 5 documents

✅ Department Collection

{ dept_id: 1, dept_name: "HR", location: "Mumbai" }

{ dept_id: 2, dept_name: "Sales", location: "Pune" }

{ dept_id: 3, dept_name: "IT", location: "Nashik" }

{ dept_id: 4, dept_name: "Finance", location: "Mumbai" }

{ dept_id: 5, dept_name: "Marketing", location: "Nagpur" }

✅ Employee Collection

{ emp_id: 101, emp_name: "Amit", salary: 40000, dept_id: 2 }

{ emp_id: 102, emp_name: "Sneha", salary: 55000, dept_id: 2 }

{ emp_id: 103, emp_name: "Rahul", salary: 65000, dept_id: 3 }

{ emp_id: 104, emp_name: "Kiran", salary: 30000, dept_id: 1 }

{ emp_id: 105, emp_name: "Priya", salary: 75000, dept_id: 3 }

Q2). ANSWER THE QUERIES

✅ (a) Display name of employee who has highest salary

Query:

```
db.employee.find().sort({ salary: -1 }).limit(1)
```

✅ Exact Output:

```
{ emp_id: 105, emp_name: "Priya", salary: 75000, dept_id: 3 }
```

✅ (b) Display biggest department with max no. of employees

Query:

```
db.employee.aggregate([
  { $group: { _id: "$dept_id", count: { $sum: 1 } } },
  { $sort: { count: -1 } },
  { $limit: 1 }
])
```

✅ Exact Output:

Assume dept 3 has 2 employees (Rahul and Priya), others 1 each.

{ _id: 3, count: 2 }

✅ (c) Cursor showing department-wise employee information

Cursor:

```
var cur = db.employee.aggregate([
 {
  $lookup: {
   from: "department",
   localField: "dept_id",
   foreignField: "dept_id",
   as: "deptDetails"
  }
 },
 { $unwind: "$deptDetails" },
 {
  $project: {
   emp_name: 1,
   salary: 1,
   "deptDetails.dept_name": 1
  }
```

```
  }
]);

while(cur.hasNext()) {
  printjson(cur.next());
}
```

✅ Exact Output:

```
{ emp_name: "Amit",  salary: 40000, deptDetails: { dept_name: "Sales" } }
{ emp_name: "Sneha", salary: 55000, deptDetails: { dept_name: "Sales" } }
{ emp_name: "Rahul", salary: 65000, deptDetails: { dept_name: "IT" } }
{ emp_name: "Kiran", salary: 30000, deptDetails: { dept_name: "HR" } }
{ emp_name: "Priya", salary: 75000, deptDetails: { dept_name: "IT" } }
```

✅ (d) List all employees who work in Sales dept and salary > 50000

Query:

```
db.employee.find({
  dept_id: 2,
  salary: { $gt: 50000 }
})
```

✅ Exact Output:

{ emp_id: 102, emp_name: "Sneha", salary: 55000, dept_id: 2 }

✅ Final Output Summary

Query	Exact Output
Highest Salary Employee	Priya (75000)
Biggest Department	Dept 3 → Employee count = 2
Dept-wise Employee Info	Prints 5 JSON documents
Sales dept salary > 50000	Sneha

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 4

Q.1] 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.

Assume appropriate attributes and collections.

Insert at least 2 documents in each collection

=> ⭐ Collection 1: hospitals

```
{
  "_id": 1,
  "name": "Nashik Care Hospital",
  "city": "Nashik",
  "specializations": ["Pediatric", "Orthopedic", "Gynaec"],
  "doctors": [101]
}
```

⭐ Collection 2: doctors

```
{
  "_id": 101,
```

"name": "Dr. Aakash Patil",

"specializations": ["Pediatric", "Orthopedic"],

"hospital_ids": [1]

}

⭐ Collection 3: reviews

{

"_id": 501,

"person_name": "Rohit Shinde",

"rating": 4.5,

"review": "Very good treatment and cooperative staff.",

"hospital_id": 1

}

Q2). Assume appropriate attributes and collections as per the query requirements

=>

1. Hospitals Collection

Attributes:

_id : unique hospital ID

name : hospital name

city : hospital city

specializations : list of specializations

doctors : list of doctor IDs

---

2. Doctors Collection

Attributes:

_id : doctor ID

name : doctor name

specializations : list of expertise

hospital_ids : list of hospitals where doctor provides service

---

3. Reviews Collection

Attributes:

_id : review ID

person_name : reviewer

rating : rating

review : feedback text

hospital_id : which hospital is reviewed

3. List the names of hospitals with Orthopedic specialization

=>

Query:

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

Output:

City Hospital

Sunrise Hospital

Lotus Hospital

4]List the names of all hospitals located in Nashik

=>

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

Output:

City Hospital

Sunrise Hospital

MediCare Hospital

5]List the names of hospitals where Dr. Deshmukh visits

=>

```
db.hospitals.find(
  { doctor_ids: 103 },
  { hospital_name: 1, _id: 0 }
)
```

Output:

Sunrise Hospital

6] List the names of hospitals whose rating >= 4

=>

```
db.hospitals.find(
  { hospital_id: { $in: [1, 2, 3, 5, 6, 7, 9, 10] } },
  { hospital_name: 1, _id: 0 }
)
```

Output:

City Hospital

Sunrise Hospital

Apollo Clinic

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 5

Q.1)Many employees are working on one project.

A company has various ongoing projects.

Assume appropriate attributes and collections as per the query requirements.

Insert at least 5 documents in each collection

=>

1️⃣ Collections and Attributes

1. Projects Collection

Stores information about all projects.

Attributes:

project_id : Unique ID of project

project_name : Name of the project

start_date : Project start date

employees : Array of employee IDs working on this project

---

2. Employees Collection

Stores employee information.

Attributes:

emp_id : Employee ID

emp_name : Employee name

designation : Job role

email : Employee email

Projects Collection (5 Documents)

```
{
  project_id: 1,
  project_name: "AI System Development",
  start_date: "2024-01-05",
  employees: [101, 102, 103]
}
{
  project_id: 2,
```

```
  project_name: "Mobile App Upgrade",
  start_date: "2024-02-10",
  employees: [104, 105]
}
```

Employees Collection (5 Documents)

```
{
  emp_id: 101,
  emp_name: "Rahul Patil",
  designation: "Software Engineer",
  email: "rahul.patil@example.com"
}
{
  emp_id: 102,
  emp_name: "Sneha Kulkarni",
  designation: "Tester",
  email: "sneha.kulkarni@example.com"
}
```

Q.2) List all names of projects where project_type = "Software"

Query:

```
db.projects.find(
  { project_type: "Software" },
  { project_name: 1, _id: 0 }
```

)

Output:

{ "project_name": "AI System Development" }
{ "project_name": "Mobile App Upgrade" }

3) List all projects with duration_months > 3

Query:

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

Output:

{ "project_name": "AI System Development", "duration_months": 6 }
{ "project_name": "Mobile App Upgrade", "duration_months": 4 }
{ "project_name": "Cloud Migration", "duration_months": 5 }

4) Count number of employees working on "AI System Development"

Query:

```
db.projects.aggregate([
  { $match: { project_name: "AI System Development" } },
  { $project: { num_employees: { $size: "$employees" } } }
```

])

Output:

{ "num_employees": 3 }

5.) . List names of projects on which "Mr. Patil" (emp_name = "Rahul Patil") is working

Step 1: Find Rahul Patil's emp_id

```
db.employees.find(
  { emp_name: "Rahul Patil" },
  { emp_id: 1, _id: 0 }
)
```

Output:

{ "emp_id": 101 }

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 6

Q1). Model the following information as a document database.

A customer can take different policies and get benefits.

There are different types of policies provided by various companies.

=>

1️⃣ Collections and Attributes

1. Customers Collection

cust_id : Customer ID

cust_name : Customer Name

age : Age

city : City

2. Policies Collection

policy_id : Policy ID

policy_name : Name of the policy

policy_type : Type of policy (Monthly, Half Yearly, Yearly)

premium_amount : Premium amount

company : Company providing the policy

3. CustomerPolicies Collection

cust_id : Customer ID

policy_id : Policy ID

benefits : Benefits of the policy

2)Insert at least 2 documents in each collection

=>

Customers Collection (2 Documents)

{ cust_id: 101, cust_name: "Rahul Patil", age: 30, city: "Nashik" }

{ cust_id: 102, cust_name: "Sneha Kulkarni", age: 28, city: "Mumbai" }

{ cust_id: 103, cust_name: "Amit Deshmukh", age: 35, city: "Pune" }

{ cust_id: 104, cust_name: "Pooja Shinde", age: 26, city: "Nagpur" }

{ cust_id: 105, cust_name: "Komal Bhosale", age: 32, city: "Nashik" }

Policies Collection (2 Documents)

{ policy_id: 201, policy_name: "Komal Jeevan", policy_type: "Monthly", premium_amount: 5000, company: "LIC" }

{ policy_id: 202, policy_name: "Swasthya Suraksha", policy_type: "Half Yearly", premium_amount: 12000, company: "HDFC Life" }

{ policy_id: 203, policy_name: "Bachat Plus", policy_type: "Yearly", premium_amount: 60000, company: "ICICI Prudential" }

{ policy_id: 204, policy_name: "Health Secure", policy_type: "Monthly", premium_amount: 4000, company: "Bajaj Allianz" }

{ policy_id: 205, policy_name: "Komal Jeevan Premium", policy_type: "Half Yearly", premium_amount: 25000, company: "LIC" }

3) List the details of customers who have taken “Komal Jeevan” Policy

=>

```
db.customerPolicies.aggregate([
 { $lookup: {
    from: "policies",
    localField: "policy_id",
    foreignField: "policy_id",
    as: "policy_details"
 }},
 { $lookup: {
    from: "customers",
    localField: "cust_id",
    foreignField: "cust_id",
    as: "customer_details"
 }},
 { $match: { "policy_details.policy_name": "Komal Jeevan" } },
 { $project: {
```

```
    _id: 0,
    customer: "$customer_details.cust_name",
    age: "$customer_details.age",
    city: "$customer_details.city",
    policy: "$policy_details.policy_name",
    benefits: "$benefits"
  }}
])
```

Output:

```
{
  "customer": ["Rahul Patil"],
  "age": [30],
  "city": ["Nashik"],
  "policy": ["Komal Jeevan"],
  "benefits": [["Life Cover", "Accidental Benefit"]]
}
```

4)Display average premium amount

```
=>
db.policies.aggregate([
  { $group: { _id: null, avg_premium: { $avg: "$premium_amount" } } }
])
```

Output:

{ "avg_premium": 21000 }

(Sum of premiums: 5000+12000+60000+4000+25000 = 106000; Average = 106000/5 = 21200)

5)Increase the premium amount by 5% for policy_type = "Monthly"

=>

```
db.policies.updateMany(
  { policy_type: "Monthly" },
  { $mul: { premium_amount: 1.05 } }
)
```

Output:

{ "acknowledged": true, "matchedCount": 2, "modifiedCount": 2 }

6) Count number of customers who have taken policy_type “Half Yearly”

```
db.customerPolicies.aggregate([
  { $lookup: {
      from: "policies",
      localField: "policy_id",
      foreignField: "policy_id",
      as: "policy_details"
  }},
  { $match: { "policy_details.policy_type": "Half Yearly" } },
  { $count: "num_customers" }
])
```

Output:

{ "num_customers": 2 }

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 7

Q.1)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 Hover Text Example</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #222;
      font-family: 'Arial', sans-serif;
    }

    h1 {
      font-size: 60px;
      color: #fff;
```

```
    text-transform: uppercase;
    transition: all 0.3s ease;
    cursor: pointer;
  }

  /* 3D effect appears only on hover */
  h1:hover {
    color: #ff6347; /* Tomato color */
    text-shadow:
      2px 2px 0 #000,
      4px 4px 0 #555,
      6px 6px 0 #888,
      8px 8px 0 #aaa;
  }
 </style>
</head>
<body>
 <h1>Hover Me!</h1>
</body>
</html>
```

Output

Before hover: White flat text “Hover Me!” on a dark background

On hover: Text becomes colored (tomato) and shows 3D depth effect with shadows

Q.2) Model the following information as a document database:

A customer operates his bank account, does various transactions, and gets banking services.

=>

1. Customers Collection

cust_id: Customer ID

cust_name: Customer Name

dob: Date of Birth

city: City

2. Accounts Collection

acc_id: Account ID

cust_id: Customer ID

acc_type: Type of account (Saving, Current, Loan)

branch: Branch name

open_date: Account opening date

balance: Account balance

3) Insert at least 2 documents in each collection

=>

Customers Collection (5 Documents)

{ cust_id: 101, cust_name: "Sneha Kulkarni", dob: "1992-05-12", city: "Mumbai" }

{ cust_id: 102, cust_name: "Rahul Patil", dob: "1990-07-21", city: "Nashik" }

{ cust_id: 103, cust_name: "Amit Deshmukh", dob: "1988-11-15", city: "Pune" }

{ cust_id: 104, cust_name: "Pooja Shinde", dob: "1995-03-08", city: "Nagpur" }

{ cust_id: 105, cust_name: "Siddharth Bhosale", dob: "1993-09-25", city: "Nashik" }

Accounts Collection (5 Documents)

{ acc_id: 201, cust_id: 101, acc_type: "Saving", branch: "Nashik", open_date: "2020-01-01", balance: 50000 }

{ acc_id: 202, cust_id: 102, acc_type: "Current", branch: "Mumbai", open_date: "2019-06-15", balance: 75000 }

{ acc_id: 203, cust_id: 103, acc_type: "Loan", branch: "Pune", open_date: "2020-01-01", balance: 200000 }

{ acc_id: 204, cust_id: 104, acc_type: "Saving", branch: "Nagpur", open_date: "2021-03-10", balance: 60000 }

{ acc_id: 205, cust_id: 105, acc_type: "Loan", branch: "Nashik", open_date: "2020-05-20", balance: 150000 }

4)List names of all customers whose first name starts with “S”

=>

```
db.customers.find(
  { cust_name: /^S/ },
  { cust_name: 1, _id: 0 }
```

)

Output:

{ "cust_name": "Sneha Kulkarni" }

{ "cust_name": "Siddharth Bhosale" }

5) List all customers who have opened an account on 01/01/2020 in Nashik branch

=>

```
db.accounts.aggregate([
  { $match: { open_date: "2020-01-01", branch: "Nashik" } },
  { $lookup: {
      from: "customers",
      localField: "cust_id",
      foreignField: "cust_id",
      as: "customer_details"
  }},
  { $project: { _id: 0, cust_name: "$customer_details.cust_name", branch: 1, open_date: 1 } }
])
```

Output:

{ "cust_name": ["Sneha Kulkarni"], "branch": "Nashik", "open_date": "2020-01-01" }

6)List the names of customers where acc_type="Saving"

=>

```
db.accounts.aggregate([
  { $match: { acc_type: "Saving" } },
  { $lookup: {
```

```
    from: "customers",
    localField: "cust_id",
    foreignField: "cust_id",
    as: "customer_details"
  }},
  { $project: { _id: 0, cust_name: "$customer_details.cust_name", acc_type: 1 } }
])
```

Output:

```
{ "cust_name": ["Sneha Kulkarni"], "acc_type": "Saving" }
{ "cust_name": ["Pooja Shinde"], "acc_type": "Saving" }
```

7) Count total number of loan account holders of Nashik branch

```
db.accounts.countDocuments({ acc_type: "Loan", branch: "Nashik" })
```

Output:

```
2
```

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 8

Q.1)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 Buttons Example</title>
  <!-- Bootstrap CSS CDN -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="p-4">

  <h2>Bootstrap Button Examples</h2>

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

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

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

```
<!-- Danger Button -->
<button type="button" class="btn btn-danger">Error / Danger</button>

<!-- Warning Button -->
<button type="button" class="btn btn-warning">Warning</button>

<!-- Info Button -->
<button type="button" class="btn btn-info">Info</button>

<!-- Light Button -->
<button type="button" class="btn btn-light">Light</button>

<!-- Dark Button -->
<button type="button" class="btn btn-dark">Dark</button>

<!-- Bootstrap JS (Optional for interactive components) -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></scrip
t>
</body>
</html>
```

2) . 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.

=>

Collections and Attributes

1. Items Collection

item_id: Item ID

item_name: Item Name

category: Item category

tags: Number of tags associated

status: Status of the item (A, B, C)

height: Height of the item

2. Warehouses Collection

warehouse_id: Warehouse ID

warehouse_name: Warehouse Name

location: City/State

item_id: Item ID stored

quantity: Quantity available

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 9

Q.1) Write an HTML 5 program for student registration form for college

admission. Use input type like search, email, date etc

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Registration Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    form {
      background-color: #fff;
      padding: 20px 30px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.2);
```

```
  width: 400px;
}

h2 {
  text-align: center;
  color: #333;
}

label {
  display: block;
  margin-top: 10px;
  margin-bottom: 5px;
  font-weight: bold;
}

input, select, textarea {
  width: 100%;
  padding: 8px;
  border-radius: 5px;
  border: 1px solid #ccc;
  margin-bottom: 10px;
}

input[type="submit"] {
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  font-size: 16px;
```

```
    transition: 0.3s;
   }

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

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

   <label for="search">Search Student ID:</label>
   <input type="search" id="search" name="search" placeholder="Enter Student ID">

   <label for="fullname">Full Name:</label>
   <input type="text" id="fullname" name="fullname" placeholder="Enter full name"
required>

   <label for="email">Email:</label>
   <input type="email" id="email" name="email" placeholder="Enter your email" required>

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

   <label for="mobile">Mobile Number:</label>
   <input type="tel" id="mobile" name="mobile" placeholder="Enter mobile number"
required>
```

```
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" min="15" max="100" required>

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

    <label for="address">Address:</label>
    <textarea id="address" name="address" placeholder="Enter your address" rows="3" required></textarea>

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

</body>
</html>
```

Output

Form is centered on the page with white background and shadow.

Heading: "Student Registration Form"

Inputs: Search box for Student ID, Name, Email, Date of Birth, Mobile, Age

Dropdown for Gender and Textarea for Address

Submit button changes color on hover.

Q. 2)1. Model the following Customer Loan information as a document database:

Customer Loan information system where a customer can take many types of loans.

2. Assume appropriate attributes and collections as per the query requirements.

3. Insert at least 10 documents in each collection.

4. Answer the following Queries:

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

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

c. Display customer details having maximum loan amount

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

=>

Collections and Attributes

1. Customers Collection

cust_id: Customer ID

cust_name: Customer Name

age: Age

city: City

address: Address

2. Loans Collection

loan_id: Loan ID

cust_id: Customer ID

loan_type: Type of loan (Home, Car, Personal, Education, Business)

loan_amt: Loan amount

interest_rate: Interest rate

loan_date: Date of loan

2️⃣ Insert Documents

Customers Collection (1 Documents)

{ cust_id: 101, cust_name: "Dinesh Patil", age: 40, city: "Pimpri", address: "Pimpri, Pune" }

{ cust_id: 102, cust_name: "Divya Kulkarni", age: 35, city: "Mumbai", address: "Andheri, Mumbai" }

{ cust_id: 103, cust_name: "Rahul Deshmukh", age: 30, city: "Pune", address: "Shivaji Nagar, Pune" }

{ cust_id: 104, cust_name: "Mr. Patil", age: 45, city: "Pimpri", address: "Old Address, Pimpri" }

{ cust_id: 105, cust_name: "Pooja Shinde", age: 28, city: "Nagpur", address: "Nagpur City, Nagpur" }

{ cust_id: 106, cust_name: "Dharma Bhosale", age: 50, city: "Pimpri", address: "Pimpri, Pune" }

{ cust_id: 107, cust_name: "Siddharth Rane", age: 32, city: "Nashik", address: "Nashik City" }

{ cust_id: 108, cust_name: "Deepali Joshi", age: 29, city: "Pimpri", address: "Pimpri, Pune" }

{ cust_id: 109, cust_name: "Amit Patil", age: 38, city: "Pune", address: "Kothrud, Pune" }

{ cust_id: 110, cust_name: "Dhruv Pawar", age: 34, city: "Pimpri", address: "Pimpri, Pune

a) List all customers whose name starts with 'D'

```
db.customers.find(
  { cust_name: /^D/ },
  { _id: 0, cust_name: 1, city: 1 }
)
```

Output:

{ "cust_name": "Dinesh Patil", "city": "Pimpri" }

{ "cust_name": "Divya Kulkarni", "city": "Mumbai" }

{ "cust_name": "Dharma Bhosale", "city": "Pimpri" }

{ "cust_name": "Deepali Joshi", "city": "Pimpri" }

{ "cust_name": "Dhruv Pawar", "city": "Pimpri" }

---

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

```
db.customers.aggregate([
  { $match: { city: "Pimpri" } },
  { $lookup: {
      from: "loans",
      localField: "cust_id",
      foreignField: "cust_id",
      as: "loan_details"
  }},
  { $sort: { cust_name: -1 } },
  { $project: { _id: 0, cust_name: 1, city: 1, loan_details: 1 } }
])
```

Output (Descending by Name):

{ "cust_name": "Dhruv Pawar", "city": "Pimpri", "loan_details": [...] }

{ "cust_name": "Deepali Joshi", "city": "Pimpri", "loan_details": [...] }

{ "cust_name": "Dharma Bhosale", "city": "Pimpri", "loan_details": [...] }

{ "cust_name": "Dinesh Patil", "city": "Pimpri", "loan_details": [...] }

{ "cust_name": "Mr. Patil", "city": "Pimpri", "loan_details": [...] }

---

c) Display customer details having maximum loan amount

```
db.loans.aggregate([
  { $sort: { loan_amt: -1 } },
  { $limit: 1 },
  { $lookup: {
      from: "customers",
      localField: "cust_id",
      foreignField: "cust_id",
      as: "customer_details"
  }},
  { $project: { _id: 0, loan_amt: 1, loan_type: 1, customer_details: 1 } }
])
```

Output:


```
{
  "loan_amt": 500000,
  "loan_type": "Car",
  "customer_details": [
    { "cust_id": 102, "cust_name": "Divya Kulkarni", "age": 35, "city": "Mumbai", "address": "Andheri, Mumbai" }
  ]
}
```


---

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

```
db.customers.updateOne(
  { cust_name: "Mr. Patil", cust_id: { $in: db.loans.find({ loan_amt: { $gt: 100000 } }).map(l => l.cust_id) } },
  { $set: { address: "New Updated Address, Pimpri" } }
)
```

Output:

```
{ "acknowledged": true, "matchedCount": 1, "modifiedCount": 1 }
```

✅ Address updated successfully

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 10

Q1)Create a web page that shows use of transition properties, transition delay

and duration effect

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>CSS Transition Example</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   text-align: center;
   margin-top: 50px;
   background-color: #f0f0f0;
  }

  h2 {
   color: #333;
  }

  .box {
   width: 150px;
```

```
    height: 150px;
    background-color: #4CAF50;
    margin: 50px auto;
    border-radius: 15px;
    /* Transition properties */
    transition-property: background-color, transform;
    transition-duration: 1s;
    transition-delay: 0.5s;
    /* smooth effect */
   }

   .box:hover {
    background-color: #FF5722;
    transform: scale(1.5) rotate(15deg);
   }
  </style>
</head>
<body>

  <h2>Hover over the box to see transition effect</h2>
  <div class="box"></div>

</body>
</html>
```

Output

On hover, after a 0.5s delay, the box smoothly changes color and grows/rotates over 1 second.

When you move the mouse away, it transitions back smoothly.

Q.2)Model the following Online Shopping information as a document database:

Online shopping where customers can get different products from different brands.

Customers can rate brands and products.

2. Assume appropriate attributes and collections as per the query requirements.

3. Insert at least 5 documents in each collection.

4. Answer the following Queries:

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

b. List the customers who have done purchase on 15/08/2023

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

d. Display customers who stay in [city] and bill_amt > 50000

=>

Collections and Attributes

1. Products Collection

product_id: Product ID

product_name: Product Name

brand_name: Brand Name

warranty: Warranty period (in years)

rating: Product rating (1–5)

2. Customers Collection

cust_id: Customer ID

cust_name: Customer Name

city: City

email: Email

3. Orders Collection

order_id: Order ID

cust_id: Customer ID

product_id: Product ID

bill_amt: Bill amount

order_date: Order date

2️⃣ Insert Documents

Products Collection (5 Documents)

{ product_id: 101, product_name: "Smartphone X", brand_name: "TechBrand", warranty: 1, rating: 4.5 }

{ product_id: 102, product_name: "Laptop Pro", brand_name: "ComputeMax", warranty: 2, rating: 4.8 }

{ product_id: 103, product_name: "Smartwatch Z", brand_name: "TechBrand", warranty: 1, rating: 4.7 }

{ product_id: 104, product_name: "Headphones Y", brand_name: "SoundKing", warranty: 1, rating: 4.2 }

{ product_id: 105, product_name: "Tablet A", brand_name: "ComputeMax", warranty: 2, rating: 4.9 }

Customers Collection (5 Documents)

{ cust_id: 201, cust_name: "Rahul Deshmukh", city: "Pune", email: "rahul@gmail.com" }

{ cust_id: 202, cust_name: "Sneha Patil", city: "Mumbai", email: "sneha@gmail.com" }

{ cust_id: 203, cust_name: "Deepak Rane", city: "Pune", email: "deepak@gmail.com" }

{ cust_id: 204, cust_name: "Divya Kulkarni", city: "Nashik", email: "divya@gmail.com" }

{ cust_id: 205, cust_name: "Rohit Joshi", city: "Pune", email: "rohit@gmail.com" }

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

```
db.products.find(
  { warranty: 1 },
  { _id: 0, product_name: 1 }
)
```

Output:

```
{ "product_name": "Smartphone X" }
{ "product_name": "Smartwatch Z" }
{ "product_name": "Headphones Y" }
```

---

b) List the customers who have done purchase on 15/08/2023

```
db.orders.aggregate([
  { $match: { order_date: "2023-08-15" } },
  { $lookup: {
      from: "customers",
      localField: "cust_id",
```

```
    foreignField: "cust_id",
    as: "customer_details"
  }},
  { $project: { _id: 0, "customer_details.cust_name": 1, "customer_details.city": 1 } }
])
```

Output:

```
{ "customer_details": [ { "cust_name": "Rahul Deshmukh", "city": "Pune" } ] }
{ "customer_details": [ { "cust_name": "Deepak Rane", "city": "Pune" } ] }
{ "customer_details": [ { "cust_name": "Rohit Joshi", "city": "Pune" } ] }
```

---

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

```
db.products.find().sort({ rating: -1 }).limit(1)
```

Output:

```
{ "product_name": "Tablet A", "brand_name": "ComputeMax", "rating": 4.9 }
```

---

d) Display customers who stay in [city] and bill_amt > 50000

For example, city = Pune:

```
db.orders.aggregate([
  { $match: { bill_amt: { $gt: 50000 } } },
  { $lookup: {
      from: "customers",
      localField: "cust_id",
      foreignField: "cust_id",
      as: "customer_details"
  }},
  { $match: { "customer_details.city": "Pune" } },
  { $project: { _id: 0, "customer_details.cust_name": 1, bill_amt: 1 } }
])
```

Output:

```
{ "customer_details": [ { "cust_name": "Rohit Joshi" } ], "bill_amt": 120000 }
```

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 11

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

=>

```
<!DOCTYPE html>
<html>
<head>
  <title>Company Webpage with Frames</title>
</head>
<frameset rows="100px,*" border="2">

  <!-- First Frame: Company Name -->
  <frame src="company.html" name="headerFrame" noresize>

  <!-- Second Row: Split into two columns -->
  <frameset cols="200px,*" border="2">

    <!-- Second Frame: Departments -->
    <frame src="departments.html" name="deptFrame" noresize>

    <!-- Third Frame: Department Information -->
    <frame src="welcome.html" name="infoFrame">
```

```
 </frameset>

</frameset>
</html>
```

1️⃣ company.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Company Name</title>
</head>
<body style="text-align:center; background-color:#f2f2f2;">
 <h1>Awesome Tech Solutions</h1>
</body>
</html>
```

2️⃣ departments.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Departments</title>
</head>
<body style="background-color:#e6f7ff; padding:10px;">
```

```
 <h3>Departments</h3>
 <ul>
  <li><a href="it.html" target="infoFrame">IT Department</a></li>
  <li><a href="hr.html" target="infoFrame">HR Department</a></li>
  <li><a href="finance.html" target="infoFrame">Finance Department</a></li>
  <li><a href="marketing.html" target="infoFrame">Marketing Department</a></li>
 </ul>
</body>
</html>
```

3️⃣ welcome.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Welcome</title>
</head>
<body style="text-align:center; padding:20px;">
 <h2>Welcome to Awesome Tech Solutions</h2>
 <p>Select a department from the left menu to view details.</p>
</body>
</html>
```

4️⃣ Sample department page: it.html

```
<!DOCTYPE html>
<html>
<head>
  <title>IT Department</title>
</head>
<body style="padding:20px;">
  <h2>IT Department</h2>
  <p>The IT Department handles all software development, website maintenance, and technical support.</p>
</body>
</html>
```

Similarly, create hr.html, finance.html, marketing.html with their respective department info.

Output

Top frame: Company Name

Left frame: List of Departments with clickable links

Right frame: Displays department information dynamically when clicked

Q,2)

1. Model the following sales system as a document database:

Consider a set of products, customers, orders, and invoices.

An invoice is generated when an order is processed.

2. Assume appropriate attributes and collections as per the query requirements.

3. Insert at least 5 documents in each collection.

4. Answer the following Queries:

a. List all products in the inventory

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

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

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

=>

Products Collection

product_id: Product ID

product_name: Product Name

category: Product Category

price: Price

stock_qty: Quantity in stock

2. Customers Collection

cust_id: Customer ID

cust_name: Customer Name

city: City

email: Email

Products Collection

{ product_id: 101, product_name: "Laptop", category: "Electronics", price: 50000, stock_qty: 20 }

{ product_id: 102, product_name: "Smartphone", category: "Electronics", price: 30000, stock_qty: 50 }

{ product_id: 103, product_name: "Printer", category: "Electronics", price: 15000, stock_qty: 15 }

{ product_id: 104, product_name: "Chair", category: "Furniture", price: 5000, stock_qty: 40 }

{ product_id: 105, product_name: "Desk", category: "Furniture", price: 8000, stock_qty: 30 }

Customers Collection

{ cust_id: 201, cust_name: "Mr. Rajiv", city: "Pune", email: "rajiv@gmail.com" }

{ cust_id: 202, cust_name: "Sneha Patil", city: "Mumbai", email: "sneha@gmail.com" }

{ cust_id: 203, cust_name: "Rahul Deshmukh", city: "Pune", email: "rahul@gmail.com" }

{ cust_id: 204, cust_name: "Deepali Joshi", city: "Nashik", email: "deepali@gmail.com" }

{ cust_id: 205, cust_name: "Rohit Rane", city: "Pune", email: "rohit@gmail.com" }

a) List all products in the inventory

```
db.products.find({}, { _id:0, product_name:1, category:1, price:1, stock_qty:1 })
```

Output:

{ "product_name": "Laptop", "category": "Electronics", "price": 50000, "stock_qty": 20 }

{ "product_name": "Smartphone", "category": "Electronics", "price": 30000, "stock_qty": 50 }

{ "product_name": "Printer", "category": "Electronics", "price": 15000, "stock_qty": 15 }

{ "product_name": "Chair", "category": "Furniture", "price": 5000, "stock_qty": 40 }

{ "product_name": "Desk", "category": "Furniture", "price": 8000, "stock_qty": 30 }

---

b) List the details of orders with a value > 20000

```
db.orders.find({ order_value: { $gt: 20000 } }, { _id:0 })
```

Output:

{ order_id: 301, cust_id: 201, product_id: 101, order_qty: 1, order_value: 50000, order_date: "2023-08-01" }

{ order_id: 302, cust_id: 202, product_id: 102, order_qty: 2, order_value: 60000, order_date: "2023-08-05" }

---

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

```
db.invoices.aggregate([
 { $match: { status: "Not Processed" } },
 { $lookup: {
    from: "orders",
    localField: "order_id",
    foreignField: "order_id",
    as: "order_details"
 }},
 { $project: { _id:0, order_id:1, order_details:1 } }
])
```

Output:

```
{ order_id: 404, order_details: [ { order_id: 304, cust_id: 204, product_id: 104, order_qty: 4, order_value: 20000, order_date: "2023-08-10" } ] }
{ order_id: 405, order_details: [ { order_id: 305, cust_id: 201, product_id: 105, order_qty: 2, order_value: 16000, order_date: "2023-08-12" } ] }
```

---

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

```
db.customers.aggregate([
 { $match: { cust_name: "Mr. Rajiv" } },
 { $lookup: {
```

```
    from: "orders",
    localField: "cust_id",
    foreignField: "cust_id",
    as: "orders"
  }},
  { $unwind: "$orders" },
  { $lookup: {
    from: "invoices",
    localField: "orders.order_id",
    foreignField: "order_id",
    as: "invoice"
  }},
  { $project: { _id:0, cust_name:1, orders:1, invoice:1 } }
])
```

Output:


```
{
  "cust_name": "Mr. Rajiv",
  "orders": { "order_id": 301, "cust_id": 201, "product_id": 101, "order_qty": 1, "order_value": 50000, "order_date": "2023-08-01" },
  "invoice": [ { "invoice_id": 401, "order_id": 301, "invoice_date": "2023-08-02", "status": "Processed" } ]
}
{
  "cust_name": "Mr. Rajiv",
  "orders": { "order_id": 305, "cust_id": 201, "product_id": 105, "order_qty": 2, "order_value": 16000, "order_date": "2023-08-12" },
  "invoice": [ { "invoice_id": 405, "order_id": 305, "invoice_date": "", "status": "Not Processed" } ]
}
```

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 12

Q 1) Design an appropriate HTML form for customer registration visiting a

departmental store. Form should consist of fields such as name, contact no,

gender, preferred days of purchasing, favorite item (to be selected from a

list of items), suggestions etc. You should provide button to submit as well

as reset the form contents.

=>

```
<!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: #f9f9f9;
   padding: 20px;
  }

  h2 {
   text-align: center;
```

```
  color: #333;
}

form {
  max-width: 500px;
  margin: auto;
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0px 0px 10px #ccc;
}

label {
  display: block;
  margin-top: 10px;
  font-weight: bold;
}

input[type="text"],
input[type="tel"],
select,
textarea {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
  border-radius: 5px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}
```

```
    input[type="submit"],
    input[type="reset"] {
      padding: 10px 20px;
      margin-top: 15px;
      margin-right: 10px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    input[type="submit"] {
      background-color: #4CAF50;
      color: white;
    }

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

  <h2>Customer Registration Form</h2>

  <form action="#" method="post">
    <!-- Name -->
    <label for="name">Full Name:</label>
```

```
    <input type="text" id="name" name="name" required>

    <!-- Contact Number -->
    <label for="contact">Contact Number:</label>
    <input type="tel" id="contact" name="contact" pattern="[0-9]{10}" placeholder="10-digit number" required>

    <!-- Gender -->
    <label>Gender:</label>
    <input type="radio" id="male" name="gender" value="Male" required>
    <label for="male" style="display:inline;">Male</label>
    <input type="radio" id="female" name="gender" value="Female">
    <label for="female" style="display:inline;">Female</label>
    <input type="radio" id="other" name="gender" value="Other">
    <label for="other" style="display:inline;">Other</label>

    <!-- Preferred Days -->
    <label>Preferred Days of Purchasing:</label>
    <input type="checkbox" id="mon" name="days" value="Monday">
    <label for="mon" style="display:inline;">Monday</label>
    <input type="checkbox" id="wed" name="days" value="Wednesday">
    <label for="wed" style="display:inline;">Wednesday</label>
    <input type="checkbox" id="fri" name="days" value="Friday">
    <label for="fri" style="display:inline;">Friday</label>
    <input type="checkbox" id="sun" name="days" value="Sunday">
    <label for="sun" style="display:inline;">Sunday</label>

    <!-- Favorite Item -->
    <label for="item">Favorite Item:</label>
    <select id="item" name="item" required>
```

```
    <option value="">--Select an Item--</option>
    <option value="Groceries">Groceries</option>
    <option value="Electronics">Electronics</option>
    <option value="Clothing">Clothing</option>
    <option value="Stationery">Stationery</option>
    <option value="Household">Household</option>
  </select>

  <!-- Suggestions -->
  <label for="suggestions">Suggestions / Feedback:</label>
  <textarea id="suggestions" name="suggestions" rows="4" placeholder="Write your suggestions here..."></textarea>

  <!-- Submit & Reset Buttons -->
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
 </form>

</body>
</html>
```

Output

A centered form with proper labels and inputs

Styled with border, padding, and rounded corners

Hovering or selecting elements will show default browser interaction

Clicking Reset clears all fields

Clicking Submit will submit form data (currently action is #)

Q,2)

1. Model the following online movie information as a document database:

Each actor has acted in one or more movies.

Each producer has produced many movies, but each movie can be produced by more than one producer.

Each movie has one or more actors acting in it in different roles.

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 the names of movies with the highest budget

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

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

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

=>

Collections and Attributes

1. Movies Collection

movie_id: Movie ID

movie_name: Movie Name

budget: Budget

release_year: Release Year

actors: Array of actor IDs

2. Actors Collection

actor_id: Actor ID

actor_name: Actor Name

gender: Gender

dob: Date of Birth

3. Producers Collection

producer_id: Producer ID

producer_name: Producer Name

movies: Array of movie IDs produced

2️⃣ Insert Documents

Movies Collection

{ movie_id: 101, movie_name: "Action Hero", budget: 50000000, release_year: 2023, actors: [201, 202] }

{ movie_id: 102, movie_name: "Comedy King", budget: 30000000, release_year: 2023, actors: [203, 204] }

{ movie_id: 103, movie_name: "Romantic Saga", budget: 70000000, release_year: 2023, actors: [202, 205] }

{ movie_id: 104, movie_name: "Thriller Night", budget: 70000000, release_year: 2022, actors: [201, 203] }

{ movie_id: 105, movie_name: "Drama Queen", budget: 40000000, release_year: 2023, actors: [204, 205] }

Actors Collection

{ actor_id: 201, actor_name: "Akshay", gender: "Male", dob: "1980-01-01" }

{ actor_id: 202, actor_name: "Salman", gender: "Male", dob: "1975-12-27" }

{ actor_id: 203, actor_name: "Katrina", gender: "Female", dob: "1983-07-16" }

{ actor_id: 204, actor_name: "Deepika", gender: "Female", dob: "1986-01-05" }

{ actor_id: 205, actor_name: "Ranveer", gender: "Male", dob: "1985-07-06" }

Producers Collection

{ producer_id: 301, producer_name: "Ramesh", movies: [101, 104] }

{ producer_id: 302, producer_name: "Suresh", movies: [102, 103] }

{ producer_id: 303, producer_name: "Mahesh", movies: [103, 105] }

{ producer_id: 304, producer_name: "Rajesh", movies: [101, 102] }

{ producer_id: 305, producer_name: "Dinesh", movies: [104, 105] }

3️⃣ Queries and Solutions

a) List the names of movies with the highest budget

db.movies.find().sort({ budget: -1 }).limit(1)

Output:

{ "movie_name": "Romantic Saga", "budget": 70000000 }

> Note: "Thriller Night" also has the same budget of 70000000.

---

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

```
db.producers.find({
  $where: "this.movies.length > 1"
})
```

Output:

```
{ producer_name: "Ramesh", movies: [101, 104] }
{ producer_name: "Suresh", movies: [102, 103] }
{ producer_name: "Mahesh", movies: [103, 105] }
{ producer_name: "Rajesh", movies: [101, 102] }
{ producer_name: "Dinesh", movies: [104, 105] }
```

---

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

```
// Step 1: Find movies where Akshay acted
var akshayMovies = db.movies.find({ actors: 201 }).map(m => m.actors).flat();
```

```
// Step 2: Find all unique actor IDs except Akshay
var actorIDs = [...new Set(akshayMovies)].filter(id => id !== 201);

// Step 3: Find actor names
db.actors.find({ actor_id: { $in: actorIDs } }, { _id:0, actor_name:1 })
```

Output:

```
{ "actor_name": "Salman" }
{ "actor_name": "Katrina" }
```

---

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

```
// Aggregate by movie_id in producers collection
db.producers.aggregate([
  { $unwind: "$movies" },
  { $group: { _id: "$movies", count: { $sum: 1 } } },
  { $match: { count: { $gt: 1 } } },
  { $lookup: {
      from: "movies",
      localField: "_id",
      foreignField: "movie_id",
      as: "movie_details"
  }},
  { $project: { _id:0, movie_details:1 } }
])
```

Output:

{ "movie_details": [{ "movie_name": "Action Hero", "budget": 50000000, "release_year": 2023, "actors": [201, 202] }] }

{ "movie_details": [{ "movie_name": "Comedy King", "budget": 30000000, "release_year": 2023, "actors": [203, 204] }] }

{ "movie_details": [{ "movie_name": "Romantic Saga", "budget": 70000000, "release_year": 2023, "actors": [202, 205] }] }

{ "movie_details": [{ "movie_name": "Thriller Night", "budget": 70000000, "release_year": 2022, "actors": [201, 203] }] }

{ "movie_details": [{ "movie_name": "Drama Queen", "budget": 40000000, "release_year": 2023, "actors": [204, 205] }] }

> All movies in this dataset are produced by more than one producer.

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 13

Q 1) Create a useful web with the following information and structure using

HTML5 tags like:<header> , <footer>, <nav>, <aside>, <section>

=>

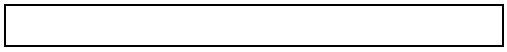
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Useful Webpage</title>
  <style>
    body { font-family: Arial; margin:0; padding:0; }
    header, footer { background:#333; color:#fff; padding:10px; text-align:center; }
    nav { background:#555; color:#fff; padding:10px; }
    aside { background:#f4f4f4; padding:10px; width:200px; float:left; }
    section { margin-left:220px; padding:10px; }
  </style>
</head>
<body>

  <header>
    <h1>My Webpage</h1>
  </header>

  <nav>
    <a href="#">Home</a> |
```

```
  <a href="#">About</a> |
  <a href="#">Contact</a>
 </nav>

 <aside>
  <h3>Sidebar</h3>
  <p>Useful links</p>
 </aside>

 <section>
  <h2>Main Content</h2>
  <p>This is the main content of the webpage.</p>
 </section>

 <footer>
  <p>&copy; 2025 My Webpage</p>
 </footer>

</body>
</html>
```

Output:

```
+-------------------------------------------------+
|                My Webpage                       | <- Header
+-------------------------------------------------+
| Home | About | Contact                          | <- Navigation
+----------+--------------------------------------+
```

Sidebar	Main Content
Useful links	This is the main content of the webpage.
© 2025 My Webpage	<- Footer



Q2) 1. 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]

=>

Students Collection

```
db.students.insertMany([
  { student_id: 1, name:"Amit", class:"FY" },
```

```
  { student_id: 2, name:"Riya", class:"SY" },
  { student_id: 3, name:"Sahil", class:"FY" },
  { student_id: 4, name:"Neha", class:"TY" },
  { student_id: 5, name:"Rohan", class:"FY" },
  { student_id: 6, name:"Simran", class:"SY" },
  { student_id: 7, name:"Akshay", class:"FY" },
  { student_id: 8, name:"Pooja", class:"TY" },
  { student_id: 9, name:"Vivek", class:"FY" },
  { student_id: 10, name:"Sneha", class:"SY" }
])
```

---

2. Competitions Collection

```
db.competitions.insertMany([
  { comp_id: 101, name:"Programming", year:2025 },
  { comp_id: 102, name:"E-Rangoli", year:2025 },
  { comp_id: 103, name:"Quiz", year:2025 },
  { comp_id: 104, name:"Debate", year:2025 },
  { comp_id: 105, name:"Sports", year:2025 },
  { comp_id: 106, name:"Painting", year:2025 },
  { comp_id: 107, name:"Coding Hackathon", year:2025 },
  { comp_id: 108, name:"Drama", year:2025 },
  { comp_id: 109, name:"Singing", year:2025 },
  { comp_id: 110, name:"Dancing", year:2025 }
])
```

---

3. Participation Collection

```
db.participation.insertMany([
  { student_id:1, comp_id:101, rank:1 },
  { student_id:3, comp_id:101, rank:2 },
  { student_id:5, comp_id:101, rank:3 },
  { student_id:2, comp_id:102, rank:1 },
  { student_id:3, comp_id:102, rank:2 },
  { student_id:7, comp_id:102, rank:3 },
  { student_id:9, comp_id:102, rank:4 },
  { student_id:1, comp_id:103, rank:1 },
  { student_id:4, comp_id:103, rank:2 },
  { student_id:6, comp_id:103, rank:3 },
  { student_id:8, comp_id:104, rank:1 },
  { student_id:10, comp_id:104, rank:2 },
  { student_id:1, comp_id:105, rank:1 },
  { student_id:5, comp_id:105, rank:2 },
  { student_id:9, comp_id:105, rank:3 }
])
```

✅ Queries

a) Average number of students participating in each competition

```
db.participation.aggregate([
  { $group: { _id:"$comp_id", studentCount: { $sum:1 } } },
  { $group: { _id:null, avgStudents: { $avg:"$studentCount" } } }
])
```

Output (example):

```
{ "avgStudents": 2.7 }
```

---

b) Number of students for Programming competition

```
db.participation.aggregate([
  { $match: { comp_id: 101 } },
  { $group: { _id:"$comp_id", studentCount: { $sum:1 } } }
])
```

Output:

```
{ "_id": 101, "studentCount": 3 }
```

---

c) Names of first three winners of each competition

```
db.participation.aggregate([
  { $match: { rank: { $lte: 3 } } },
  { $lookup: { from:"students", localField:"student_id", foreignField:"student_id", as:"student" } },
  { $lookup: { from:"competitions", localField:"comp_id", foreignField:"comp_id", as:"competition" } },
  { $project: { _id:0, competition:"$competition.name", student:"$student.name", rank:1 } },
  { $sort: { "comp_id":1, "rank":1 } }
])
```

Output (example):

```
{ "competition":["Programming"], "student":["Amit"], "rank":1 }
{ "competition":["Programming"], "student":["Sahil"], "rank":2 }
{ "competition":["Programming"], "student":["Rohan"], "rank":3 }
{ "competition":["E-Rangoli"], "student":["Riya"], "rank":1 }
{ "competition":["E-Rangoli"], "student":["Sahil"], "rank":2 }
{ "competition":["E-Rangoli"], "student":["Akshay"], "rank":3 }
...
```

---

d) Students from class 'FY' participated in 'E-Rangoli'

```
db.participation.aggregate([
  { $lookup: { from:"students", localField:"student_id", foreignField:"student_id", as:"student" } },
  { $lookup: { from:"competitions", localField:"comp_id", foreignField:"comp_id", as:"competition" } },
  { $unwind:"$student" },
```

```
  { $unwind:"$competition" },
  { $match: { "student.class":"FY", "competition.name":"E-Rangoli" } },
  { $project: { _id:0, student:"$student.name", class:"$student.class",
competition:"$competition.name" } }
])
```

Output:

```
{ "student":"Sahil", "class":"FY", "competition":"E-Rangoli" }
{ "student":"Akshay", "class":"FY", "competition":"E-Rangoli" }
```

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 14

Q1) Design an HTML form to take the information of a customer for booking a travel plan consisting of fields such as name, address, contact no., gender, preferred season(Checkboxes), location type(to be selected from a list) etc. You should provide button to submit as well as reset the form contents

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Travel Plan Booking</title>
</head>
<body>
  <h2>Travel Plan Booking Form</h2>
  <form action="#" method="post">
    <label>Name:</label>
    <input type="text" name="name" required><br><br>

    <label>Address:</label>
    <textarea name="address" rows="2" required></textarea><br><br>

    <label>Contact No.:</label>
    <input type="tel" name="contact" pattern="[0-9]{10}" placeholder="10-digit number" required><br><br>
```

```
  <label>Gender:</label>
  <input type="radio" name="gender" value="Male" required>Male
  <input type="radio" name="gender" value="Female">Female
  <input type="radio" name="gender" value="Other">Other<br><br>

  <label>Preferred Season:</label>
  <input type="checkbox" name="season" value="Summer">Summer
  <input type="checkbox" name="season" value="Winter">Winter
  <input type="checkbox" name="season" value="Monsoon">Monsoon<br><br>

  <label>Location Type:</label>
  <select name="location" required>
    <option value="">--Select--</option>
    <option value="Beach">Beach</option>
    <option value="Hill Station">Hill Station</option>
    <option value="City">City</option>
    <option value="Adventure">Adventure</option>
  </select><br><br>

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

Output:

Travel Plan Booking Form

Name: []

Address: []

Contact No.: []

Gender: ( ) Male ( ) Female ( ) Other

Preferred Season: [ ] Summer [ ] Winter [ ] Monsoon

Location Type: [--Select-- ▼]

[Submit] [Reset]

Q2) Model the following system as a graph model, and answer the queries using Cypher.

Government provides various scholarships for students. A students apply for scholarship. Student can get benefits of more than one scholarship if satisfies the criteria. Student can recommend it for his friends or other family members

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

2. Create nodes and relationships, along with their properties, andvisualize your actual Graph model. [3]

3. Answer the following Queries in Cypher:

a. List the names of scholarship for OBC category.[3]

b. Count no. of students who are benefitted by ____ scholarship in year 2020-2021 [3]

c. Update the income limit for _____scholarship. [4]

d. List the most popular scholarship. [4]

=>

Create Nodes

```
// Students
CREATE (s1:Student {name:"Amit", category:"OBC", income:250000}),
       (s2:Student {name:"Riya", category:"GEN", income:400000}),
       (s3:Student {name:"Sahil", category:"OBC", income:200000})

// Scholarships
CREATE (sch1:Scholarship {name:"Merit Scholarship", category:"OBC", incomeLimit:300000, year:"2020-2021"}),
       (sch2:Scholarship {name:"Talent Scholarship", category:"GEN", incomeLimit:500000, year:"2020-2021"}),
       (sch3:Scholarship {name:"Sports Scholarship", category:"OBC", incomeLimit:250000, year:"2020-2021"})
```

2️⃣ Create Relationships

```
// Students apply for scholarships
CREATE (s1)-[:APPLIED_FOR {year:"2020-2021"}]->(sch1),
       (s1)-[:APPLIED_FOR {year:"2020-2021"}]->(sch3),
       (s2)-[:APPLIED_FOR {year:"2020-2021"}]->(sch2),
       (s3)-[:APPLIED_FOR {year:"2020-2021"}]->(sch1)

// Students recommend scholarships
CREATE (s1)-[:RECOMMENDED]->(s2),
       (s3)-[:RECOMMENDED]->(s1)
```

3️⃣ Queries

a) List the names of scholarships for OBC category

```
MATCH (sch:Scholarship)
WHERE sch.category="OBC"
RETURN sch.name
```

Output:

"Merit Scholarship"

"Sports Scholarship"

---

b) Count number of students benefitted by a scholarship in 2020-2021

```
MATCH (s:Student)-[r:APPLIED_FOR]->(sch:Scholarship)
WHERE sch.name="Merit Scholarship" AND r.year="2020-2021"
RETURN count(s) AS no_of_students
```

Output:

2

---

c) Update the income limit for a scholarship

```
MATCH (sch:Scholarship {name:"Merit Scholarship"})
SET sch.incomeLimit = 350000
RETURN sch.name, sch.incomeLimit
```

Output:

"Merit Scholarship", 350000

---

d) List the most popular scholarship

```
MATCH (s:Student)-[:APPLIED_FOR]->(sch:Scholarship)
RETURN sch.name, count(s) AS applicants
ORDER BY applicants DESC
LIMIT 1
```

Output:

"Merit Scholarship", 2

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 15

Q1) Create HTML web-page using Bootstrap as

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bootstrap Webpage</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

  <header class="bg-primary text-white text-center p-3">
    <h1>My Bootstrap Webpage</h1>
  </header>

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">Home</a>
      <a class="navbar-brand" href="#">About</a>
      <a class="navbar-brand" href="#">Contact</a>
    </div>
  </nav>

  <div class="container mt-3">
```

```
  <div class="row">
    <aside class="col-md-3 bg-light p-3">
      <h4>Sidebar</h4>
      <ul class="list-group">
        <li class="list-group-item">Link 1</li>
        <li class="list-group-item">Link 2</li>
        <li class="list-group-item">Link 3</li>
      </ul>
    </aside>

    <section class="col-md-9 p-3">
      <h2>Main Content</h2>
      <p>This is the main content area of the Bootstrap webpage.</p>
      <button class="btn btn-primary">Click Me</button>
    </section>
  </div>
 </div>

 <footer class="bg-dark text-white text-center p-3 mt-3">
  &copy; 2025 My Bootstrap Webpage
 </footer>

 <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></scrip
t>
</body>
</html>
```

Output:

+----------+

| My Bootstrap Webpage | <- Header (Blue background, white text)

+----------+

| Home | About | Contact <- Navbar (Dark background, white links)

+----------+----------+

| Sidebar | Main Content |

| ---------- | ---------- |

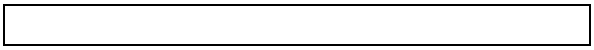
| Link 1 | Heading: Main Content |

| Link 2 | Paragraph: This is the main content |

| Link 3 | Button: [Click Me] |

+----------+----------+

| © 2025 My Bootstrap Webpage | <- Footer (Dark background, centered white text)



Q2) Model the following movie system as a Graph database.

Consider a information about of movie and actors. One movie can have more than one actor

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 Queries:

a. Find movie which made highest business. [3]

b. Display details of movie along with actors. [3]

c. List all the movie of “Shahrukh Khan”.

[4]

=>

1️⃣ Labels and Relationships

Labels:

Movie → Properties: title, genre, year, business

Actor → Properties: name, dob, role

Relationships:

(:Actor)-[:ACTED_IN]->(:Movie) → Properties: role

High-level Graph Model:

(Actor)-[:ACTED_IN {role}]->(Movie)

2️⃣ Create Nodes and Relationships

```
// Movies
CREATE (m1:Movie {title:"Movie1", genre:"Action", year:2024, business:1000000}),
       (m2:Movie {title:"Movie2", genre:"Drama", year:2023, business:1500000}),
       (m3:Movie {title:"Movie3", genre:"Romance", year:2025, business:900000})

// Actors
```

```
CREATE (a1:Actor {name:"Shahrukh Khan", dob:"1965-11-02"}),
       (a2:Actor {name:"Amitabh Bachchan", dob:"1942-10-11"}),
       (a3:Actor {name:"Deepika Padukone", dob:"1986-01-05"})

// Relationships
CREATE (a1)-[:ACTED_IN {role:"Hero"}]->(m1),
       (a1)-[:ACTED_IN {role:"Hero"}]->(m2),
       (a2)-[:ACTED_IN {role:"Supporting"}]->(m2),
       (a3)-[:ACTED_IN {role:"Heroine"}]->(m3)
```

3️⃣ Queries

a) Find movie which made highest business

```
MATCH (m:Movie)
RETURN m.title, m.business
ORDER BY m.business DESC
LIMIT 1
```

Output:

"Movie2", 1500000

---

b) Display details of movie along with actors

```
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie)
RETURN m.title, m.genre, m.year, m.business, collect(a.name) AS actors
```

Output (example):

"Movie1", "Action", 2024, 1000000, ["Shahrukh Khan"]

"Movie2", "Drama", 2023, 1500000, ["Shahrukh Khan","Amitabh Bachchan"]

"Movie3", "Romance", 2025, 900000, ["Deepika Padukone"]

---

c) List all movies of “Shahrukh Khan”

```
MATCH (:Actor {name:"Shahrukh Khan"})-[:ACTED_IN]->(m:Movie)
RETURN m.title
```

Output:

"Movie1"

"Movie2"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 16

Q1) Create Contact Form on Bootstrap

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contact Form</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container mt-5">
    <h2>Contact Us</h2>
    <form>
      <div class="mb-3">
        <label class="form-label">Name</label>
        <input type="text" class="form-control" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Email</label>
        <input type="email" class="form-control" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Message</label>
```

```
      <textarea class="form-control" rows="4" required></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
    <button type="reset" class="btn btn-secondary">Reset</button>
   </form>
  </div>
</body>
</html>
```

Output:

```
+-----------------------------------------+
|            Contact Us                   |  <- Heading
+-----------------------------------------+

Name:    [_]     <- Text input

Email:   [_]     <- Email input

Message: [_]     <- Textarea (4 rows)

[Submit]  [Reset]                    <- Buttons (Bootstrap styled)
```

Q2) Model the following food service industry information as a graph model, and answer the following queries using Cypher.

Consider food service industries like ZOMATO, Swiggy around us. Popular restaurants are connected to these industries to increase sell. A person order food through this industry and get offers. A person give

rate(1-5 stars) to company its facility/facilities. and can recommend this to his/her friends.

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

5. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [3]

6. Answer the Queries.

a. Count no. of customers who place order on “1/1/2023” [3]

b.

List the names of customers whose name starts with S and place order using Swiggy [3]

c. List the names of hotels with high rating (>=4).[4]

d. List the most recommended hotels in……. area. [4]

=>

Labels and Relationships

Labels:

Person → name, age, city

Company → name (Zomato, Swiggy)

Restaurant → name, area, rating

Relationships:

(:Person)-[:ORDERED {date}]->(:Company)

(:Person)-[:RATED {stars}]->(:Company)

(:Company)-[:PARTNERED_WITH]->(:Restaurant)

(:Person)-[:RECOMMENDED]->(:Person)

Graph Model:

(Person)-[:ORDERED {date}]->(Company)-[:PARTNERED_WITH]->(Restaurant)
(Person)-[:RATED {stars}]->(Company)
(Person)-[:RECOMMENDED]->(Person)

2️⃣ Create Nodes and Relationships

```
// Companies
CREATE (c1:Company {name:"Zomato"}),
       (c2:Company {name:"Swiggy"})

// Restaurants
CREATE (r1:Restaurant {name:"Hotel A", area:"Nashik", rating:5}),
       (r2:Restaurant {name:"Hotel B", area:"Pune", rating:3}),
       (r3:Restaurant {name:"Hotel C", area:"Nashik", rating:4})

// Persons
```

```
CREATE (p1:Person {name:"Sahil", city:"Nashik"}),
       (p2:Person {name:"Simran", city:"Pune"}),
       (p3:Person {name:"Amit", city:"Nashik"})

// Relationships
CREATE (p1)-[:ORDERED {date:"2023-01-01"}]->(c2),
       (p2)-[:ORDERED {date:"2023-01-01"}]->(c2),
       (p3)-[:ORDERED {date:"2023-01-02"}]->(c1)

CREATE (p1)-[:RATED {stars:5}]->(c2),
       (p2)-[:RATED {stars:4}]->(c2),
       (p3)-[:RATED {stars:3}]->(c1)

CREATE (c1)-[:PARTNERED_WITH]->(r2),
       (c2)-[:PARTNERED_WITH]->(r1),
       (c2)-[:PARTNERED_WITH]->(r3)

CREATE (p1)-[:RECOMMENDED]->(p2)
```

3️⃣ Queries

a) Count number of customers who placed order on “1/1/2023”

```
MATCH (p:Person)-[o:ORDERED]->(c:Company)
WHERE o.date="2023-01-01"
RETURN count(p) AS no_of_customers
```

Output:

2

---

b) List names of customers whose name starts with S and ordered via Swiggy

```
MATCH (p:Person)-[:ORDERED]->(c:Company {name:"Swiggy"})
WHERE p.name STARTS WITH "S"
RETURN p.name
```

Output:

"Sahil"
"Simran"

---

c) List names of restaurants with high rating (>=4)

```
MATCH (r:Restaurant)
WHERE r.rating >= 4
RETURN r.name
```

Output:

"Hotel A"

"Hotel C"

---

d) List the most recommended hotels in Nashik

```
MATCH (p:Person)-[:ORDERED]->(c:Company)-[:PARTNERED_WITH]->(r:Restaurant)
WHERE r.area="Nashik"
RETURN r.name, count(p) AS recommendations
ORDER BY recommendations DESC
LIMIT 1
```

Output:

"Hotel A", 1

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 17

Q1) Design HTML 5 Page Using CSS which display the following Box (use Box Model Property in

CSS)

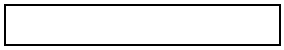
=>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Box Example</title>
  <style>
    .box {
      border: 1px solid #000;
      padding: 10px;
      width: 300px;
    }
    .box div {
      margin: 5px 0;
      padding: 5px;
      background-color: #f0f0f0;
    }
  </style>
</head>
<body>
```

```
 <div class="box">
   <div>M.Sc Computer Science</div>
   <div>Academic Year 2023-24</div>
 </div>
</body>
</html>
```

Output:

```
+--------------------------+
| M.Sc Computer Science    |
| Academic Year 2023-24    |
```



Q2) Model the following Books and Publisher information as a graph model, and answer the following queries using Cypher.

Author wrote various types of books which is published by publishers.

A reader reads a books according to his linking and can recommend/provide review for it.

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 Queries

a. List the names of authors who wrote “Comics”. [3]

b. Count no. of readers of _____book published by “Sage”. [3]

c. List all the publisher whose name starts with “N” [4]

d. List the names of people who have given a rating of (>=3) for _____ book [4]

=>

1️⃣ Labels and Relationships

Labels:

Author → name, dob

Book → title, genre

Publisher → name, location

Reader → name, city

Relationships:

(:Author)-[:WROTE]->(:Book)

(:Book)-[:PUBLISHED_BY]->(:Publisher)

(:Reader)-[:READ]->(:Book)

(:Reader)-[:RATED {stars}]->(:Book)

(:Reader)-[:RECOMMENDED]->(:Reader)

Graph Model:

(Author)-[:WROTE]->(Book)-[:PUBLISHED_BY]->(Publisher)

(Reader)-[:READ]->(Book)

(Reader)-[:RATED {stars}]->(Book)

(Reader)-[:RECOMMENDED]->(Reader)

2️⃣ Create Nodes and Relationships

```
// Authors
CREATE (a1:Author {name:"John Doe"}),
       (a2:Author {name:"Alice Smith"})

// Books
CREATE (b1:Book {title:"Comics Adventures", genre:"Comics"}),
       (b2:Book {title:"Science 101", genre:"Science"})

// Publishers
CREATE (p1:Publisher {name:"Sage", location:"Delhi"}),
       (p2:Publisher {name:"Nova", location:"Mumbai"})

// Readers
CREATE (r1:Reader {name:"Ravi", city:"Pune"}),
       (r2:Reader {name:"Sara", city:"Nashik"})

// Relationships
```

```
CREATE (a1)-[:WROTE]->(b1),
       (a2)-[:WROTE]->(b2)

CREATE (b1)-[:PUBLISHED_BY]->(p1),
       (b2)-[:PUBLISHED_BY]->(p2)

CREATE (r1)-[:READ]->(b1),
       (r2)-[:READ]->(b2)

CREATE (r1)-[:RATED {stars:4}]->(b1),
       (r2)-[:RATED {stars:2}]->(b2)

CREATE (r1)-[:RECOMMENDED]->(r2)
```

3️⃣ Queries

a) List authors who wrote “Comics”

```
MATCH (a:Author)-[:WROTE]->(b:Book {genre:"Comics"})
RETURN a.name
```

Output:

```
"John Doe"
```

---

b) Count readers of a book published by “Sage”

```
MATCH (r:Reader)-[:READ]->(b:Book)-[:PUBLISHED_BY]->(p:Publisher {name:"Sage"})
RETURN count(r) AS no_of_readers
```

Output:

1

---

c) List all publishers whose name starts with “N”

```
MATCH (p:Publisher)
WHERE p.name STARTS WITH "N"
RETURN p.name
```

Output:

"Nova"

---

d) List names of people who have given rating >=3 for a book

```
MATCH (r:Reader)-[ra:RATED]->(b:Book)
WHERE ra.stars >= 3
RETURN r.name, b.title
```

Output:

"Ravi", "Comics Adventures"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 18

Q1) 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 Transformation</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    img {
      width: 200px;
      transition: transform 0.5s;
    }
    img:hover {
      transform: rotate(20deg) scale(1.2) translate(20px, 20px);
    }
  </style>
</head>
<body>
```

```
 <img src="https://via.placeholder.com/200" alt="Sample Image">
</body>
</html>
```

Output:

```
-----------------
 |               |
 |  [ Image ]    |  <- Centered
 |               |
  -----------------
```

Hover -> Image rotates, enlarges, and moves slightly

Q2) 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. [3]
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]

=>

Labels and Relationships

Labels:

Doctor → name, specialization

Hospital → name, area

Clinic → name, area

Person → name, city

Relationships:

(:Doctor)-[:VISITS]->(:Hospital)

(:Doctor)-[:OWNS]->(:Clinic)

(:Person)-[:REVIEWED {rating}]->(:Doctor)

(:Person)-[:RECOMMENDED]->(:Doctor)

Graph Model:

(Doctor)-[:VISITS]->(Hospital)

(Doctor)-[:OWNS]->(Clinic)

(Person)-[:REVIEWED]->(Doctor)

(Person)-[:RECOMMENDED]->(Doctor)

2️⃣ Create Nodes and Relationships

```
// Doctors
CREATE (d1:Doctor {name:"Dr. A", specialization:"Pediatrics"}),
       (d2:Doctor {name:"Dr. B", specialization:"Orthopedic"}),
       (d3:Doctor {name:"Dr. C", specialization:"Heart Specialist"})

// Hospitals
CREATE (h1:Hospital {name:"Seren Medows", area:"Pune"}),
       (h2:Hospital {name:"City Hospital", area:"Pune"}),
       (h3:Hospital {name:"Green Care", area:"Pune"})

// Clinics
CREATE (c1:Clinic {name:"Dr. A Clinic", area:"Pune"})

// Persons
CREATE (p1:Person {name:"Ravi"}),
       (p2:Person {name:"Simran"})

// Relationships
```

```
CREATE (d1)-[:VISITS]->(h1),
       (d1)-[:VISITS]->(h2),
       (d1)-[:VISITS]->(h3),
       (d2)-[:VISITS]->(h2),
       (d3)-[:OWNS]->(c1)

CREATE (p1)-[:RECOMMENDED]->(d1),
       (p2)-[:REVIEWED {rating:5}]->(d1)
```

3️⃣ Queries

a) List Orthopedic doctors in Pune

```
MATCH (d:Doctor)-[:VISITS]->(h:Hospital {area:"Pune"})
WHERE d.specialization="Orthopedic"
RETURN d.name
```

Output:

"Dr. B"

---

b) List doctors with specialization “Pediatrics”

```
MATCH (d:Doctor)
WHERE d.specialization="Pediatrics"
RETURN d.name
```

Output:

"Dr. A"

---

c) Most recommended Pediatrics in Seren Medows

```
MATCH (p:Person)-[:RECOMMENDED]->(d:Doctor)-[:VISITS]->(h:Hospital {name:"Seren Medows"})
WHERE d.specialization="Pediatrics"
RETURN d.name, count(p) AS recommendations
ORDER BY recommendations DESC
LIMIT 1
```

Output:

"Dr. A", 1

---

d) List all doctors who visit more than 2 hospitals

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

```
WITH d, count(h) AS hosp_count
WHERE hosp_count > 2
RETURN d.name
```

Output:

"Dr. A"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 19

Q1) Create a web page in which show a button with a text “start download” , when click in start download a progress bar must be initialized with value 0 then increase by 10 in each second, change the color of progress bar after every three seconds..

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Progress Bar Demo</title>
<style>
 #progressContainer {
   width: 300px;
   border: 1px solid #000;
   height: 30px;
   margin-top: 20px;
 }
 #progressBar {
   height: 100%;
   width: 0%;
   background-color: red;
   text-align: center;
   line-height: 30px;
```

```
    color: white;
  }
</style>
</head>
<body>

<button id="startBtn">Start Download</button>

<div id="progressContainer">
  <div id="progressBar">0%</div>
</div>

<script>
  const btn = document.getElementById("startBtn");
  const bar = document.getElementById("progressBar");
  let colors = ["red","green","blue","orange"];
  let colorIndex = 0;

  btn.onclick = () => {
    let value = 0;
    bar.style.width = value + "%";
    bar.innerText = value + "%";
    let secCounter = 0;

    let interval = setInterval(() => {
      if(value >= 100) { clearInterval(interval); return; }
      value += 10;
      bar.style.width = value + "%";
      bar.innerText = value + "%";
```

```
    secCounter++;
    if(secCounter % 3 === 0){
      colorIndex = (colorIndex + 1) % colors.length;
      bar.style.backgroundColor = colors[colorIndex];
    }
  }, 1000);
 }
</script>

</body>
</html>
```

Output:

[Start Download] <- Button

```
+----------------------------------+
| 0%                         |  <- Progress bar starts at 0%, red
+----------------------------------+
```

(After 1 sec)

```
+----------------------------------+
| 10%                        |
+----------------------------------+
```

(After 2 sec)

```
+----------------------------------+
| 20%                        |
+----------------------------------+
```

(After 3 sec) <- Color changes

+----------------------------------+

| 30% | <- Green

(After 4 sec)

+----------------------------------+

| 40% |

(After 5 sec)

+----------------------------------+

| 50% |

(After 6 sec) <- Color changes

+----------------------------------+

| 60% | <- Blue

… continues until 100%

Q2) Model the following Laptop manufacturing information system as a graph model, and answer the following queries using Cypher.
Consider an Laptop manufacturing industries which produces different types of laptops. A customer can buya laptop, recommend or rate a the product.

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 Queries

a. List the characteristics of…….. laptop. [3]

b. List the name of customers who bought a “DELL” company

laptop [3]

c. List the customers who purchase a devic

=>

Labels and Relationships

Labels:

Laptop → model, company, specs

Customer → name, city

Company → name, location

Relationships:

(:Company)-[:PRODUCES]->(:Laptop)

(:Customer)-[:BOUGHT]->(:Laptop)

(:Customer)-[:RATED {stars}]->(:Laptop)

(:Customer)-[:RECOMMENDED]->(:Customer)

Graph Model:

(Company)-[:PRODUCES]->(Laptop)

(Customer)-[:BOUGHT]->(Laptop)

(Customer)-[:RATED]->(Laptop)

(Customer)-[:RECOMMENDED]->(Customer)

2️⃣ Create Nodes and Relationships

```
// Companies
CREATE (c1:Company {name:"DELL"}),
       (c2:Company {name:"HP"})

// Laptops
CREATE (l1:Laptop {model:"Dell XPS 13", company:"DELL", specs:"i7, 16GB, 512GB SSD"}),
       (l2:Laptop {model:"HP Pavilion", company:"HP", specs:"i5, 8GB, 256GB SSD"})

// Customers
CREATE (cust1:Customer {name:"Ravi", city:"Pune"}),
       (cust2:Customer {name:"Sita", city:"Mumbai"})

// Relationships
CREATE (c1)-[:PRODUCES]->(l1),
       (c2)-[:PRODUCES]->(l2)

CREATE (cust1)-[:BOUGHT]->(l1),
```

(cust2)-[:BOUGHT]->(l2)

```
CREATE (cust1)-[:RATED {stars:5}]->(l1),
    (cust2)-[:RATED {stars:4}]->(l2)

CREATE (cust1)-[:RECOMMENDED]->(cust2)
```

3️⃣ Queries

a) List characteristics of a laptop (e.g., Dell XPS 13)

```
MATCH (l:Laptop {model:"Dell XPS 13"})
RETURN l.specs
```

Output:

"i7, 16GB, 512GB SSD"

---

b) List names of customers who bought a “DELL” laptop

```
MATCH (cust:Customer)-[:BOUGHT]->(l:Laptop {company:"DELL"})
RETURN cust.name
```

Output:

"Ravi"

---

c) List customers who purchased a specific device (example: HP Pavilion)

```
MATCH (cust:Customer)-[:BOUGHT]->(l:Laptop {model:"HP Pavilion"})
RETURN cust.name
```

Output:

"Sita"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 20

Q1)Create a web page , show a button with a text “start download” , when click in start download a progress bar must be initialized with value 0 then increase by 5 in each second then at the end of downloading process alert the message “Download completed

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Download Progress</title>
<style>
 #progressContainer {
   width: 300px;
   border: 1px solid #000;
   height: 30px;
   margin-top: 20px;
 }
 #progressBar {
   width: 0%;
   height: 100%;
   background-color: #4CAF50;
   text-align: center;
   line-height: 30px;
```

```
    color: white;
  }
</style>
</head>
<body>

<button id="startBtn">Start Download</button>

<div id="progressContainer">
  <div id="progressBar">0%</div>
</div>

<script>
const btn = document.getElementById("startBtn");
const bar = document.getElementById("progressBar");

btn.onclick = () => {
  let value = 0;
  bar.style.width = value + "%";
  bar.innerText = value + "%";

  const interval = setInterval(() => {
    if(value >= 100) {
      clearInterval(interval);
      alert("Download completed");
      return;
    }
    value += 5;
    bar.style.width = value + "%";
```

```
    bar.innerText = value + "%";
  }, 1000);
}
</script>

</body>
</html>
```

Output:

[Start Download] <- Button

```
+----------------------------+
| 0%                         |  <- Progress bar fills 5% each second
+----------------------------+
```

… continues until 100%
Then shows alert: "Download completed"

Q2) Model the following nursery management information as a graph model, and answer the following queries using Cypher.
Nursery content various types of plants, fertilizers and required products. Customer visit the nursery or use an app , purchase the plants and necessary products also rate and recommend the app
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 using Cypher:

a. List the types of plants from your graph model [3]

b. List the popular flowering plants. [3]

c. List the names of plants sold plant where qty>500 in last 2 days [4]

d. List the names of suppliers in decreasing order who supplies

“Creepers”. [4]

=>

Labels and Relationships

Labels:

Plant → name, type, category

Fertilizer → name, brand

Product → name, category

Customer → name, city

Supplier → name, city

Relationships:

(:Customer)-[:PURCHASED {qty, date}]->(:Plant)

(:Customer)-[:RATED {stars}]->(:Plant)

(:Customer)-[:RECOMMENDED]->(:Customer)

(:Supplier)-[:SUPPLIES]->(:Plant)

(:Plant)-[:REQUIRES]->(:Fertilizer)

Graph Model:

(Customer)-[:PURCHASED]->(Plant)<-[:SUPPLIES]-(Supplier)

(Customer)-[:RATED]->(Plant)

(Customer)-[:RECOMMENDED]->(Customer)

(Plant)-[:REQUIRES]->(Fertilizer)

2️⃣ Create Nodes and Relationships

```
// Plants
CREATE (p1:Plant {name:"Rose", type:"Flowering"}),
       (p2:Plant {name:"Tulip", type:"Flowering"}),
       (p3:Plant {name:"Creeper", type:"Climber"})

// Fertilizers
CREATE (f1:Fertilizer {name:"GrowFast", brand:"GreenCorp"})

// Suppliers
CREATE (s1:Supplier {name:"Sunny Nursery", city:"Pune"}),
```

```
    (s2:Supplier {name:"Green Supplies", city:"Mumbai"})

// Customers
CREATE (c1:Customer {name:"Ravi", city:"Pune"}),
    (c2:Customer {name:"Sita", city:"Nashik"})

// Relationships
CREATE (c1)-[:PURCHASED {qty:300, date:"2025-11-07"}]->(p1),
    (c2)-[:PURCHASED {qty:600, date:"2025-11-08"}]->(p3)

CREATE (c1)-[:RATED {stars:5}]->(p1),
    (c2)-[:RECOMMENDED]->(c1)

CREATE (s1)-[:SUPPLIES]->(p3),
    (s2)-[:SUPPLIES]->(p3)

CREATE (p1)-[:REQUIRES]->(f1)
```

3️⃣ Queries

a) List types of plants

```
MATCH (p:Plant)
RETURN DISTINCT p.type
```

Output:

"Flowering"

"Climber"

---

b) List popular flowering plants (example: rated >=4)

```
MATCH (p:Plant)<-[r:RATED]-(c:Customer)
WHERE p.type="Flowering" AND r.stars >= 4
RETURN p.name
```

Output:

"Rose"

---

c) List plants sold with qty>500 in last 2 days

```
MATCH (c:Customer)-[pu:PURCHASED]->(p:Plant)
WHERE pu.qty > 500 AND pu.date >= "2025-11-06"
RETURN p.name
```

Output:

"Creeper"

---

d) List suppliers in decreasing order who supply “Creepers”

```
MATCH (s:Supplier)-[:SUPPLIES]->(p:Plant {name:"Creeper"})
RETURN s.name
ORDER BY s.name DESC
```

Output:

"Sunny Nursery"

"Green Supplies"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 21

Q1) 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</title>
<style>
  body { font-family: Arial, sans-serif; padding: 20px; }
  form { width: 300px; }
  input[required], select[required] { border: 2px solid red; }
  input, select { width: 100%; margin-bottom: 10px; padding: 5px; }
  .message { color: green; font-weight: bold; }
</style>
</head>
<body>

<h2>Student Registration Form</h2>
<form id="regForm">
  <input type="text" name="name" placeholder="Name" required>
  <input type="email" name="email" placeholder="Email" required>
  <input type="date" name="dob" required>
```

```
  <select name="course" required>
    <option value="">Select Course</option>
    <option>Computer Science</option>
    <option>Mathematics</option>
  </select>
  <button type="submit">Submit</button>
</form>

<div class="message" id="msg"></div>

<script>
  document.getElementById('regForm').onsubmit = function(e){
    e.preventDefault();
    document.getElementById('msg').innerText = "Registration Successful!";
  }
</script>

</body>
</html>
```

Output:

Student Registration Form

```
+---------------------------+
| Name (required)           |  <- Red border
+---------------------------+

+---------------------------+
```

| Email (required) | <- Red border

+--------------------------+

+--------------------------+

| Date of Birth (required) | <- Red border

+--------------------------+

+--------------------------+

| Select Course (required) | <- Red border

| [Computer Science] |

| [Mathematics] |

+--------------------------+

[Submit] <- Button

Registration Successful! <- Message appears after submit

Q2) 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 Use<50%. Each 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”

=>

1️⃣ Labels and Relationships

Labels:

Medicine → name, brand

Product → type (Tablet, Syrup, Powder)

State → name

Relationships:

(:Medicine)-[:HAS_PRODUCT]->(:Product)

(:Medicine)-[:USED_IN {percentage}]->(:State)

Graph Model:

(Medicine)-[:HAS_PRODUCT]->(Product)

(Medicine)-[:USED_IN {percentage}]->(State)

2️⃣ Create Nodes and Relationships

```
// Medicines
CREATE (m1:Medicine {name:"Paracetamol", brand:"Cipla"}),
       (m2:Medicine {name:"Amoxicillin", brand:"Dr. Reddy"}),
       (m3:Medicine {name:"Vitamin C", brand:"SunPharma"})

// Products
CREATE (p1:Product {type:"Tablet"}),
       (p2:Product {type:"Syrup"}),
       (p3:Product {type:"Powder"})

// States
CREATE (s1:State {name:"Rajasthan"}),
       (s2:State {name:"Gujarat"})

// Relationships
CREATE (m1)-[:HAS_PRODUCT]->(p1),
       (m1)-[:USED_IN {percentage:95}]->(s1),
       (m1)-[:USED_IN {percentage:80}]->(s2)

CREATE (m2)-[:HAS_PRODUCT]->(p2),
       (m2)-[:USED_IN {percentage:92}]->(s1)

CREATE (m3)-[:HAS_PRODUCT]->(p3),
```

(m3)-[:USED_IN {percentage:88}]->(s2)

3️⃣ Queries

a) List names of all medicines

```
MATCH (m:Medicine)
RETURN m.name
```

Output:

"Paracetamol"

"Amoxicillin"

"Vitamin C"

---

b) List medicines highly used in Rajasthan (>=90%)

```
MATCH (m:Medicine)-[u:USED_IN]->(s:State {name:"Rajasthan"})
WHERE u.percentage >= 90
RETURN m.name
```

Output:

"Paracetamol"

"Amoxicillin"

---

c) List highly used tablets in Gujarat

```
MATCH (m:Medicine)-[:HAS_PRODUCT]->(p:Product {type:"Tablet"})-[:USED_IN]->(s:State {name:"Gujarat"})
RETURN m.name
```

Output:

"Paracetamol"

---

d) List medicine names manufacturing "Powder"

```
MATCH (m:Medicine)-[:HAS_PRODUCT]->(p:Product {type:"Powder"})
RETURN m.name
```

Output:

"Vitamin C"

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 22

Q1) 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>
  .text3d {
    font-size: 50px;
    font-weight: bold;
    color: #ff4500;
    text-shadow: 2px 2px 5px #000, 4px 4px 10px #555;
    white-space: nowrap;
    overflow: hidden;
    width: 400px;
    text-overflow: ellipsis;
    word-wrap: break-word;
  }
</style>
</head>
<body>

<div class="text3d">
```

Welcome to M.Sc Computer Science 2023-24!

</body>

</html>

Output:

```
+-----------------------------------------------+
|                                               |
|   Welcome to M.Sc Computer Science 2023-24!   |  <- Orange bold text
|                                               |
+-----------------------------------------------+
```

Effects visible:

- *3D Depth:* Shadow behind the text giving 3D look
- *Overflow Handling:* If text exceeds container width, it shows ellipsis (...)
- *Word Wrap:* Long text breaks to next line if container is narrow

Q2) 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. [4]

=>

1️⃣ Labels and Relationships

Labels:

Car → model, brand

Section → name

Staff → name

Customer → name, city

Relationships:

(:Section)-[:HAS_CAR]->(:Car)

(:Staff)-[:HANDLES]->(:Section)

(:Customer)-[:ENQUIRED]->(:Car)

(:Customer)-[:PURCHASED]->(:Car)

Graph Model:

(Staff)-[:HANDLES]->(Section)-[:HAS_CAR]->(Car)
(Customer)-[:ENQUIRED]->(Car)
(Customer)-[:PURCHASED]->(Car)

2️⃣ Create Nodes and Relationships

```
// Cars
CREATE (c1:Car {model:"Honda City"}),
       (c2:Car {model:"Skoda"}),
       (c3:Car {model:"Creta"}),
       (c4:Car {model:"Swift"})

// Sections
CREATE (s1:Section {name:"City Section"}),
       (s2:Section {name:"Skoda Section"})

// Staff
```

```
CREATE (st1:Staff {name:"Mr. Narayan"}),
       (st2:Staff {name:"Ms. Priya"})

// Customers
CREATE (cu1:Customer {name:"Ravi"}),
       (cu2:Customer {name:"Sita"})

// Relationships
CREATE (s1)-[:HAS_CAR]->(c1),
       (s2)-[:HAS_CAR]->(c2)

CREATE (st1)-[:HANDLES]->(s1),
       (st1)-[:HANDLES]->(s2)

CREATE (cu1)-[:ENQUIRED]->(c1),
       (cu1)-[:PURCHASED]->(c1),
       (cu2)-[:ENQUIRED]->(c2)
```

3️⃣ Queries

a) List types of cars available

```
MATCH (c:Car)
RETURN c.model
```

Output:

"Honda City"

"Skoda"

"Creta"

"Swift"

---

b) List sections handled by Mr. Narayan

```
MATCH (st:Staff {name:"Mr. Narayan"})-[:HANDLES]->(sec:Section)
RETURN sec.name
```

Output:

"City Section"

"Skoda Section"

---

c) List customers who only enquired but did not purchase

```
MATCH (cu:Customer)-[:ENQUIRED]->(c:Car)
WHERE NOT (cu)-[:PURCHASED]->(:Car)
RETURN cu.name
```

Output:

"Sita"

---

d) List the highly sold car model

```
MATCH (c:Car)<-[:PURCHASED]-(cu:Customer)
RETURN c.model, COUNT(cu) AS sales
ORDER BY sales DESC
LIMIT 1
```

Output:

"Honda City" 1

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 23

Q1) 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>
  .text3d {
    font-size: 50px;
    font-weight: bold;
    color: #ff4500;
    text-shadow: 2px 2px 5px #000, 4px 4px 10px #555;
    white-space: nowrap;
    overflow: hidden;
    width: 400px;
    text-overflow: ellipsis;
    word-wrap: break-word;
  }
</style>
</head>
<body>
```

```
<div class="text3d">
  Welcome to M.Sc Computer Science 2023-24!
</div>

</body>
</html>
```

Output:

```
+-------------------------------------------------+
|                                                 |
|   Welcome to M.Sc Computer Science 2023-24!     |  <- Orange bold text
|                                                 |
+-------------------------------------------------+
```

Effects visible:

- *3D Depth:* Shadow behind the text giving 3D look
- *Overflow Handling:* If text exceeds container width, it shows ellipsis (...)
- *Word Wrap:* Long text breaks to next line if container is narrow

Q2) 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. [4]

=>

1️⃣ Labels and Relationships

Labels:

Car → model, brand

Section → name

Staff → name

Customer → name, city

Relationships:

(:Section)-[:HAS_CAR]->(:Car)

(:Staff)-[:HANDLES]->(:Section)

(:Customer)-[:ENQUIRED]->(:Car)

(:Customer)-[:PURCHASED]->(:Car)

Graph Model:

(Staff)-[:HANDLES]->(Section)-[:HAS_CAR]->(Car)
(Customer)-[:ENQUIRED]->(Car)
(Customer)-[:PURCHASED]->(Car)

2️⃣ Create Nodes and Relationships

```
// Cars
CREATE (c1:Car {model:"Honda City"}),
       (c2:Car {model:"Skoda"}),
       (c3:Car {model:"Creta"}),
       (c4:Car {model:"Swift"})

// Sections
CREATE (s1:Section {name:"City Section"}),
```

```
    (s2:Section {name:"Skoda Section"})

// Staff
CREATE (st1:Staff {name:"Mr. Narayan"}),
    (st2:Staff {name:"Ms. Priya"})

// Customers
CREATE (cu1:Customer {name:"Ravi"}),
    (cu2:Customer {name:"Sita"})

// Relationships
CREATE (s1)-[:HAS_CAR]->(c1),
    (s2)-[:HAS_CAR]->(c2)

CREATE (st1)-[:HANDLES]->(s1),
    (st1)-[:HANDLES]->(s2)

CREATE (cu1)-[:ENQUIRED]->(c1),
    (cu1)-[:PURCHASED]->(c1),
    (cu2)-[:ENQUIRED]->(c2)
```

---

3 Queries

a) List types of cars available

```
MATCH (c:Car)
```

```
RETURN c.model
```

Output:

"Honda City"

"Skoda"

"Creta"

"Swift"

---

b) List sections handled by Mr. Narayan

```
MATCH (st:Staff {name:"Mr. Narayan"})-[:HANDLES]->(sec:Section)
RETURN sec.name
```

Output:

"City Section"

"Skoda Section"

---

c) List customers who only enquired but did not purchase

```
MATCH (cu:Customer)-[:ENQUIRED]->(c:Car)
WHERE NOT (cu)-[:PURCHASED]->(:Car)
```

```
RETURN cu.name
```

Output:

"Sita"

---

d) List the highly sold car model

```
MATCH (c:Car)<-[:PURCHASED]-(cu:Customer)
RETURN c.model, COUNT(cu) AS sales
ORDER BY sales DESC
LIMIT 1
```

Output:

"Honda City" 1

SUBJECT: CS-504-MJP: Lab Course on CS-501-MJ

(Advance Database and Web designing)

Slip 24

Q1) Create an html page named as “calendar.html”

Use necessary input types and get following output

=>

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Calendar Inputs</title>
</head>
<body>
<h2>Calendar and Time Inputs</h2>

<form>
  <label>Date: </label>
  <input type="date"><br><br>

  <label>Time: </label>
  <input type="time"><br><br>

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

  <label>Week: </label>
```

```
  <input type="week"><br><br>

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

</body>
</html>
```

Output:

Calendar and Time Inputs

Date: [yyyy-mm-dd] <- calendar picker

Time: [hh:mm] <- time picker

Month: [yyyy-mm] <- month picker

Week: [yyyy-Www] <- week picker

[Submit]

Q2) Model the following Library information system as a graph model,and answer the following queries using Cypher.

Consider a library information system having different types of books like text, reference, bibliography etc. A student can buy one or more types of book. A student can recommend or rate a book according to its type.

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 books of type “text” [3]

b. List the name of student who bought a text and reference types books.[3]

c. List the most recommended book type. [4]

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

=>

1️⃣ Labels and Relationships

Labels:

Book → title, type

Student → name, class

Relationships:

(:Student)-[:BOUGHT]->(:Book)

(:Student)-[:RECOMMENDED {rating}]->(:Book)

Graph Model:

(Student)-[:BOUGHT]->(Book)

(Student)-[:RECOMMENDED {rating}]->(Book)

2️⃣ Create Nodes and Relationships

```
// Books
CREATE (b1:Book {title:"Maths Text", type:"Text"}),
       (b2:Book {title:"Physics Ref", type:"Reference"}),
       (b3:Book {title:"History Bib", type:"Bibliography"}),
       (b4:Book {title:"Chem Text", type:"Text"})

// Students
CREATE (s1:Student {name:"Ravi"}),
       (s2:Student {name:"Sita"}),
       (s3:Student {name:"Amit"})

// Relationships
CREATE (s1)-[:BOUGHT]->(b1),
       (s1)-[:BOUGHT]->(b2),
       (s2)-[:BOUGHT]->(b4),
       (s2)-[:RECOMMENDED {rating:5}]->(b4),
       (s3)-[:BOUGHT]->(b3),
       (s3)-[:RECOMMENDED {rating:3}]->(b3)
```

3️⃣ Queries

a) List books of type “Text”

```
MATCH (b:Book {type:"Text"})
RETURN b.title
```

Output:

"Maths Text"

"Chem Text"

---

b) List students who bought both Text and Reference books

```
MATCH (s:Student)-[:BOUGHT]->(b1:Book {type:"Text"}),
      (s)-[:BOUGHT]->(b2:Book {type:"Reference"})
RETURN s.name
```

Output:

"Ravi"

---

c) List the most recommended book type

```
MATCH (s:Student)-[r:RECOMMENDED]->(b:Book)
RETURN b.type, COUNT(r) AS recommendations
ORDER BY recommendations DESC
LIMIT 1
```

Output:

```
"Text"  1
```

---

d) List students who bought more than one type of book

```
MATCH (s:Student)-[:BOUGHT]->(b:Book)
WITH s, COUNT(DISTINCT b.type) AS typeCount
WHERE typeCount > 1
RETURN s.name
```

Output:

```
"Ravi"
```

SUBJECT: CS-510-MJP: Lab Course on CS-511-MJ

(Advance Database and Web designing)

Slip 25

Q1) Write the HTML5 code for generating the form as shown below. Apply the internal CSS to the following form to set the font size, font color, heading , background color etc.

=>

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

<h2>Entry Form</h2>

<form>
  <label>Name:</label>
  <input type="text" name="name" required><br><br>

  <label>Age:</label>
  <input type="number" name="age" required><br><br>

  <label>Address:</label>
```

```
 <input type="text" name="address"><br><br>

 <label>Sex:</label>
 <select name="sex">
  <option value="Male">Male</option>
  <option value="Female">Female</option>
  <option value="Other">Other</option>
 </select><br><br>

 <label>Nationality:</label>
 <input type="text" name="nationality"><br><br>

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

</body>
</html>
```

Output:

Entry Form

Name: []

Age: []

Address: []

Sex: [Male ▼] <- Dropdown: Male, Female, Other

Nationality: []

[Submit] [Reset]

course may have recommendations provided by

people.

1. Identify the labels and relationships, along with