**Slip 1**

Q1) 1)List the department and the total salary for each department, sorted by total

salary in descending order:

->CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25));

INSERT INTO Worker

(WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT) VALUES

(001, &#39;Monika&#39;, &#39;Arora&#39;, 100000, &#39;21-02-20 09.00.00&#39;, &#39;HR&#39;),

(002, &#39;Niharika&#39;, &#39;Verma&#39;, 80000, &#39;21-06-11 09.00.00&#39;, &#39;Admin&#39);

SELECT department, SUM(salary) AS total\_salary

FROM Worker

GROUP BY department

ORDER BY total\_salary DESC;

2)Find the average salary for each department and display only those

departments where the average salary is above $100,000, sorted by average

salary in descending order

=>

SELECT Department, AVG(Salary) AS avg\_salary FROM Worker GROUP BY Department HAVING AVG(Salary) > 100000 ORDER BY avg\_salary DESC;

Q.2)Model the following Property system as a document database. Consider a set of

Property, Owner. One owner can buy many properties.

1. Assume appropriate attributes and collections as per the query

requirements.

2. Insert at least 05 documents in each collection. [3]

use MScDS

db.createCollection('Property')

db. Property.insertMany).{

"\_id": ObjectId("property\_id"),

"address": "123 Main St",

"type": "Residential",

"size": "2000 sq ft",

"price": 250000,

"owner\_id": ObjectId("owner\_id")}

Owner Collection:

1. {

"\_id": ObjectId("609ed2511d5bc60e46b1dcaa"),

"name": "Mr. Patil",

"contact": "patil@example.com",

"properties\_owned": [

{

"property\_id": ObjectId("609ed20a1d5bc60e46b1dca1"),

"purchase\_date": ISODate("2024-03-29"),

"price\_paid": 250000

},

{

"property\_id": ObjectId("609ed20a1d5bc60e46b1dca4"),

"purchase\_date": ISODate("2023-06-15"),

"price\_paid": 600000

}

]

}

3. Answer the following Queries

a)Display area wise property details. [3]

db.Property.aggregate([

{

$group: {

\_id: "$area",

properties: { $push: "$$ROOT" }

}

}

])

b)Display property owned by &#39;Mr.Patil&#39; having minimum rate [3]

db.Owner.aggregate([

{

$match: { name: "Mr. Patil" }

},

{

$unwind: "$properties\_owned"

},

{

$lookup: {

from: "Property",

localField: "properties\_owned.property\_id",

foreignField: "\_id",

as: "property\_details"

}

},

{

$unwind: "$property\_details"

},

{

$sort: { "property\_details.price": 1 }

},

{

$limit: 1

}

])

c)Give the details of owner whose property is at “Nashik”. [4]

db.Property.aggregate([

{

$match: { area: "Nashik" }

},

{

$lookup: {

from: "Owner",

localField: "\_id",

foreignField: "properties\_owned.property\_id",

as: "owners"

}

},

{

$unwind: "$owners"

}

])

d)Display area of property whose rate is less than 100000. [4]

db.Property.find({ price: { $lt: 100000 } }, { area: 1 })

**Slip 2**

Q1)Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

job\_name VARCHAR(255),

manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2)

);

INSERT INTO Employee (emp\_id, emp\_name, job\_name, manager\_id, hire\_date, salary)

VALUES

(1, 'John Doe', 'Manager', NULL, '2022-01-05', 2500),

(2, 'Jane Smith', 'Engineer', 1, '2022-02-10', 2200),

(3, 'Michael Johnson', 'Engineer', 1, '2022-03-15', 2300),

(4, 'Emily Brown', 'Analyst', 2, '2022-04-20', 2100),

(5, 'David Lee', 'Analyst', 3, '2022-05-25', 2400);

1)List the average salary of each job title, sorted in descending order of

average salary:

=>

SELECT job\_name, AVG(salary) AS avg\_salary FROM Employee GROUP BY job\_name ORDER BY avg\_salary DESC

2)List the department and the maximum salary among employees who joinedafter 1995, sorted by maximum salary in descending order

SELECT department, MAX(salary) AS max\_salary

FROM employees

WHERE join\_date > '1995-01-01'

GROUP BY department

ORDER BY max\_salary DESC;

Q.2)Model the following system as a document database.Consider a database of newspaper, publisher, and city.Different publisher publishes various newspapers in different cities

2. Assume appropriate attributes and collections as per the query

requirements. [3]

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

db.Newspaper.insertMany[(

{

"\_id": ObjectId("..."),

"name": "The Times",

"language": "English",

"publisher\_id": ObjectId("..."),

"city\_id": ObjectId("...")

}])

db.Publisher.insertMany([{

"\_id": ObjectId("..."),

"name": "ABC Publications",

"state": "Maharashtra"

}])

db.City.insertMany([{

"\_id": ObjectId("..."),

"name": "Nashik",

"state": "Maharashtra"

}

4. Answer the following Queries.

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

db.Newspaper.aggregate([

{

$lookup: {

from: "City",

localField: "city\_id",

foreignField: "\_id",

as: "city"

}

},

{

$match: { "city.name": "Nashik" }

},

{

$project: { name: 1, language: 1 }

}

])

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

db.Newspaper.find({ language: "Marathi" }, { name: 1 })

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

db.Publisher.count({ state: "Gujarat" })

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

state[4]

var cursor = db.Newspaper.aggregate([

{

$lookup: {

from: "City",

localField: "city\_id",

foreignField: "\_id",

as: "city"

}

},

{

$match: { "city.state": "Maharashtra" }

},

{

$group: {

\_id: "$name",

total\_sales: { $sum: 1 }

}

},

{

$sort: { total\_sales: -1 }

}

]);

while (cursor.hasNext()) {

printjson(cursor.next());

}

**Slip 3**

Q1)Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_D

ATE,DEPARTMENT)

CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25));

INSERT INTO Worker

(WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT) VALUES

(001, &#39;Monika”&#39;Arora&#39;, 100000, &#39;21-02-20 09.00.00&#39;, &#39;HR&#39;),

(002, &#39;Niharika&#39;, &#39;Verma&#39;, 80000, &#39;21-06-11 09.00.00&#39;, &#39;Admin&#39);

Write a query to insert 5 rows in it as per the query requirements.

1)List the number of workers in each department who joined after January 1,

2021, sorted by department name:

2)Find the department with the highest number of workers earning a salary

greater than $90,000, sorted by department name:

🡺

CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker (WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT)

VALUES

(1, 'John', 'Doe', 95000, '2021-03-15', 'Finance'),

(2, 'Jane', 'Smith', 87000, '2021-05-20', 'IT'),

(3, 'Michael', 'Johnson', 105000, '2021-02-10', 'Finance'),

(4, 'Emily', 'Brown', 92000, '2021-07-08', 'Marketing'),

(5, 'David', 'Lee', 98000, '2021-04-25', 'IT');

SELECT Department, COUNT(\*) AS num\_workers FROM Worker WHERE JOINING\_DATE > '2021-01-01' GROUP BY Department ORDER BY Department;

SELECT DEPARTMENT, COUNT(\*) AS NumOfWorkers

FROM Worker

WHERE SALARY > 90000

GROUP BY DEPARTMENT

ORDER BY NumOfWorkers DESC, DEPARTMENT;

Q2) 1. Model the following system as a document database. Consider employee and

department’s information.

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

[3]

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

db.Employee.insertMany([

{

"\_id": ObjectId("..."),

"first\_name": "John",

"last\_name": "Doe",

"salary": 75000,

"department\_id": ObjectId("...")

}])

db.Department.insertMany([

{

"\_id": ObjectId("..."),

"name": "Finance",

"location": "New York"

}])

4. Answer the following Queries.

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

db.Employee.find().sort({ salary: -1 }).limit(1).project({ first\_name: 1, last\_name: 1 })

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

db.Employee.aggregate([

{

$group: {

\_id: "$department\_id",

count: { $sum: 1 }

}

},

{

$sort: { count: -1 }

},

{

$lookup: {

from: "Department",

localField: "\_id",

foreignField: "\_id",

as: "department"

}

},

{

$limit: 1

},

{

$project: {

department\_name: { $arrayElemAt: ["$department.name", 0] },

num\_employees: "$count"

}

}

])

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

var cursor = db.Department.find();

while (cursor.hasNext()) {

var department = cursor.next();

var departmentEmployees = db.Employee.find({ department\_id: department.\_id }).toArray();

print("Department: " + department.name);

departmentEmployees.forEach(function(employee) {

printjson(employee);

});

}

d. List all the employees who work in Sales dept and salary &gt; 50000 [4]

db.Employee.find({ department\_id: ObjectId("sales\_department\_id"), salary: { $gt: 50000 } })

**Slip 4**

Q1)Consider the Patient table

Patient(PatientID,Name,DateOfBirth,Gender,admit\_date,ward\_no,City)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of patients admitted to each ward, sorted by ward number:

2)Find the average age of patients admitted to each ward, and display only those

wards where the average age is below 40, sorted by average age in descending

order

=>

CREATE TABLE Patient (

PatientID INT PRIMARY KEY,

Name VARCHAR(255),

DateOfBirth DATE,

Gender VARCHAR(10),

admit\_date DATE,

ward\_no INT,

City VARCHAR(255)

);

INSERT INTO Patient (PatientID, Name, DateOfBirth, Gender, admit\_date, ward\_no, City)

VALUES

(1, 'John Doe', '1990-05-15', 'Male', '2022-03-10', 101, 'New York'),

(2, 'Jane Smith', '1985-08-20', 'Female', '2022-03-12', 102, 'Los Angeles'),

(3, 'Michael Johnson', '1982-11-30', 'Male', '2022-03-14', 103, 'Chicago'),

(4, 'Emily Brown', '1978-04-25', 'Female', '2022-03-16', 101, 'San Francisco'),

(5, 'David Lee', '1995-09-10', 'Male', '2022-03-18', 102, 'Boston');

SELECT ward\_no, COUNT(\*) AS num\_patients

FROM Patient

GROUP BY ward\_no

ORDER BY ward\_no;

SELECT ward\_no,

AVG(DATEDIFF(CURRENT\_DATE(), DateOfBirth) / 365) AS avg\_age

FROM Patient

GROUP BY ward\_no

HAVING avg\_age < 40

ORDER BY avg\_age DESC;

Q2). Model the following information system as a document database.

Consider hospitals around Nashik. Each hospital may have one or more

specializations like Pediatric, Gynaec, Orthopedic, etc. A person can

recommend/provide review for a hospital. A doctor can give service to one

or more hospitals.

2. Assume appropriate attributes and collections as per the query

requirements. [3]

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

Hospital.insertMany([{

"\_id": ObjectId("..."),

"name": "City Hospital",

"city": "Nashik",

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

"rating": 4.5

}])

Doctor.insertMany([{

"\_id": ObjectId("..."),

"name": "Dr. Deshmukh",

"hospitals": [ObjectId("..."), ObjectId("...")]

}])

4. Answer the following Queries

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

db.Hospital.find({ specializations: "Pediatric" }, { name: 1 })

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

db.Hospital.find({ city: "Nashik" }, { name: 1 })

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

db.Hospital.find({ name: { $in: db.Person.findOne({ name: "Dr. Deshmukh" }).hospitals\_visited } }, { name: 1 })

d. List the names of hospitals whose rating &gt;=4 [4]

db.Hospital.find({ rating: { $gte: 4 } }, { name: 1 })

**Slip 5**

Q1)Consider the Patient table

Patient(PatientID,Name,DateOfBirth,Gender,admit\_date,ward\_no,City)

Write a query to insert 5 rows in it that as per the query requirements.

1)List the number of male and female patients admitted to each ward, sorted by

ward number and gender:

2)Find the ward with the highest number of patients admitted, and display the

top 3 wards with the highest number of patients, sorted by the number of

patients in descending order

CREATE TABLE Patient (

PatientID INT PRIMARY KEY,

Name VARCHAR(255),

DateOfBirth DATE,

Gender VARCHAR(10),

admit\_date DATE,

ward\_no INT,

City VARCHAR(255)

);

INSERT INTO Patient (PatientID, Name, DateOfBirth, Gender, admit\_date, ward\_no, City)

VALUES

(1, 'John Doe', '1990-05-15', 'Male', '2022-03-10', 101, 'New York'),

(2, 'Jane Smith', '1985-08-20', 'Female', '2022-03-12', 102, 'Los Angeles'),

(3, 'Michael Johnson', '1982-11-30', 'Male', '2022-03-14', 103, 'Chicago'),

(4, 'Emily Brown', '1978-04-25', 'Female', '2022-03-16', 101, 'San Francisco'),

(5, 'David Lee', '1995-09-10', 'Male', '2022-03-18', 102, 'Boston');

SELECT ward\_no, Gender, COUNT(\*) AS num\_patients

FROM Patient

GROUP BY ward\_no, Gender

ORDER BY ward\_no, Gender;

SELECT ward\_no, COUNT(\*) AS num\_patients

FROM Patient

GROUP BY ward\_no

ORDER BY num\_patients DESC

LIMIT 3;

Q2). Model the following database. Many employees working on one project. A

company has various ongoing projects.

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

[3]

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

Employee Collection:

1. {

"\_id": ObjectId("..."),

"name": "John Doe",

"employee\_id": "EMP001",

"projects": ["ProjectA", "ProjectB"]

}

2. {

"\_id": ObjectId("..."),

"name": "Jane Smith",

"employee\_id": "EMP002",

"projects": ["ProjectA", "ProjectC"]

}

3. {

"\_id": ObjectId("..."),

"name": "Michael Johnson",

"employee\_id": "EMP003",

"projects": ["ProjectB"]

}

4. {

"\_id": ObjectId("..."),

"name": "Emily Brown",

"employee\_id": "EMP004",

"projects": ["ProjectC", "ProjectD"]

}

5. {

"\_id": ObjectId("..."),

"name": "David Lee",

"employee\_id": "EMP005",

"projects": ["ProjectD"]

}

Project Collection:

1. {

"\_id": ObjectId("..."),

"name": "ProjectA",

"project\_type": "TypeA",

"duration\_months": 5

}

2. {

"\_id": ObjectId("..."),

"name": "ProjectB",

"project\_type": "TypeB",

"duration\_months": 4

}

3. {

"\_id": ObjectId("..."),

"name": "ProjectC",

"project\_type": "TypeA",

"duration\_months": 6

}

4. {

"\_id": ObjectId("..."),

"name": "ProjectD",

"project\_type": "TypeC",

"duration\_months": 3

}

5. {

"\_id": ObjectId("..."),

"name": "ProjectE",

"project\_type": "TypeB",

"duration\_months": 2

}

4. Answer the following Queries

a. List all names of projects where Project\_type =….. [3]

db.Project.find({ project\_type: "TypeA" }, { name: 1 })

b. List all the projects with duration greater than 3 months [3]

db.Project.find({ duration\_months: { $gt: 3 } }, { name: 1 })

c. Count no. of employees working on ……..project [4]

db.Employee.aggregate([

{ $match: { projects: "ProjectA" } },

{ $group: { \_id: null, count: { $sum: 1 } } }

])

d. List the names of projects on which Mr. Patil is working [4]

db.Project.find({ name: { $in: db.Employee.findOne({ name: "Mr. Patil" }).projects } }, { name: 1 })

**Slip 6**

Q1)Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_D

ATE,DEPARTMENT)

Write a query to insert 5 rows in it as per the query requirements.

1)List the department and the number of workers who joined in February 2021

and have a salary greater than $80,000, sorted by department name:

2)Find the department with the highest average salary among departments with

at least 2 workers, sorted by average salary in descending order:

CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker (WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT)

VALUES

(1, 'John', 'Doe', 95000, '2021-03-15', 'Finance'),

(2, 'Jane', 'Smith', 87000, '2021-05-20', 'IT'),

(3, 'Michael', 'Johnson', 105000, '2021-02-10', 'Finance'),

(4, 'Emily', 'Brown', 92000, '2021-07-08', 'Marketing'),

(5, 'David', 'Lee', 98000, '2021-04-25', 'IT');

Q2)Model the following information as a Mongodb database in mongoshell. A customer can take different policies and get the benefit. There are different types of policies provided by various companies. Assume appropriate attributes and collections as per the query requirements. Insert at least 5 documents in each collection.

db.createCollection("customers")

db.customers.insertMany([ { name: "Amaan", policy: "Komal Jeevan", premium\_amount: 200,type: "Yearly",company:"Life Insurers Ltd" }, { name: "Vanshay", policy: "Health Guard", premium\_amount: 250,type: "Monthly",company:"Health Insurers Inc"}, { name: "Adityya", policy: "Retirement Plus", premium\_amount: 300,type: "Half Yearly",company:"Retirement Solutions" }, { name: "Rushikesh", policy: "Komal Jeevan", premium\_amount: 220,type: "Yearly",company:"Life Insurers Ltd"}, { name: "Sneha", policy: "Accident Protector", premium\_amount: 270,type: "Monthly",company:"Health Insurers Inc"} ])

List the details of customers who have taken "Komal Jeevan" Policy

db.customers.find({ policy: "Komal Jeevan" })

Display average premium amount

db.customers.aggregate([{ $group: { \_id: null, avg\_premium: { $avg: "$premium\_amount" } } }]) Increase the premium amount by 5% for policy type="Monthly"

db.policies.updateMany({ type: "Monthly" }, { $mul: { premium\_amount: 1.05 } })

Count the number of customers who have taken policy type "half yearly

db.customers.find({ "type": "Half Yearly" }).count()

**Slip 7**

Q1)Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

job\_name VARCHAR(255),

manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2)

);

INSERT INTO Employee (emp\_id, emp\_name, job\_name, manager\_id, hire\_date, salary)

VALUES

(1, 'John Doe', 'Manager', NULL, '2022-01-05', 2500),

(2, 'Jane Smith', 'Engineer', 1, '2022-02-10', 2200),

(3, 'Michael Johnson', 'Engineer', 1, '2022-03-15', 2300),

(4, 'Emily Brown', 'Analyst', 2, '2022-04-20', 2100),

(5, 'David Lee', 'Analyst', 3, '2022-05-25', 2400);

1)Find the total number of employees in each department whose salary is above

$2000, sorted by department name:

SELECT job\_name, COUNT(\*) AS num\_employees

FROM Employee

WHERE salary > 2000

GROUP BY job\_name

ORDER BY job\_name;

2)List the manager\_id and the number of employees managed by each manager

who manages more than one employee, sorted by manager\_id:

SELECT manager\_id, COUNT(emp\_id) AS num\_managed\_employees

FROM Employee

WHERE manager\_id IS NOT NULL

GROUP BY manager\_id

HAVING COUNT(emp\_id) > 1

ORDER BY manager\_id;

Q2) 1. Model the following information as a Mongodb database in mongo shell. A customer operates his bank account, does various transactions and get the banking services Assume appropriate attributes and collections as per the query requirements.

Insert at least 5 documents in each collection.

db.createCollection("customers")

db.createCollection("transactions")

db.customers.insertMany([ { firstName: "Amaan", acctype: "Saving", branch: "A" }, { firstName: "Sara", acctype: "Loan", branch: "B" }, { firstName: "Sam", acctype: "Saving", branch: "A" }, { firstName: "Denzil", acctype: "Loan", branch: "C" }, { firstName: "Merlin", acctype: "Saving", branch: "B" } ])

db.transactions.insertMany([ { customerId: 1, date: "2020-01-01" }, { customerId: 2, date: "2020-01-01" }, { customerId: 3, date: "2021-03-15" }, { customerId: 4, date: "2020-01-01" }, { customerId: 5, date: "2020-01-01" } ]) List names of all customers whose first name starts with a “S” db.customers.find({ firstName: /^S/ }, { \_id: 0, firstName: 1 }) List the names customers where acctype=”Saving” db.customers.find({ acctype: "Saving" }, { \_id: 0, firstName: 1 })

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

db.customers.find({ firstName: /^S/ }, { \_id: 0, firstName: 1 })

List the names customers where acctype=”Saving”

db.customers.find({ acctype: "Saving" }, { \_id: 0, firstName: 1 })

Count the total number of loan account holders of a specific branch

db.customers.count({ acctype: "Loan", branch: "A" })

db.customers.count({ acctype: "Loan", branch: "C" })

**Slip 8**

Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

Write a query to insert 5 rows in it that as per the query requirements.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

job\_name VARCHAR(255),

manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2)

);

INSERT INTO Employee (emp\_id, emp\_name, job\_name, manager\_id, hire\_date, salary)

VALUES

(1, 'John Doe', 'Manager', NULL, '2022-01-05', 2500),

(2, 'Jane Smith', 'Engineer', 1, '2022-02-10', 2200),

(3, 'Michael Johnson', 'Engineer', 1, '2022-03-15', 2300),

(4, 'Emily Brown', 'Analyst', 2, '2022-04-20', 2100),

(5, 'David Lee', 'Analyst', 3, '2022-05-25', 2400);

1)Find the department with the highest average salary among departments with

at least 3 employees:

SELECT job\_name AS department, AVG(salary) AS avg\_salary

FROM Employee

GROUP BY job\_name

HAVING COUNT(\*) >= 3

ORDER BY avg\_salary DESC

LIMIT 1;

2)Find the top 3 departments with the highest total salary expenditure, sorted by

total salary expenditure in descending order

SELECT job\_name AS department, SUM(salary) AS total\_salary\_expenditure

FROM Employee

GROUP BY job\_name

ORDER BY total\_salary\_expenditure DESC

LIMIT 3;

Q1)Model the following inventory information as a Mongodb database in mongoshell. 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. Assume appropriate attributes and collections as per the query requirements.

db.createCollection("items")

db.createCollection("inventories")

db.items.insertMany([ { name: "Laptop", tags: ["electronics", "laptop","Gadgets","Tech","Office","Gaming"], status: "A", height:10 }, { name: "T-shirt", tags: ["clothing", "t-shirt","Fashion","Fabric"], status: "B", height: 5 }, { name: "Notebook", tags: ["office supplies", "notebook","Stationary","School","Official"], status: "C", height: 8 }, { name: "Chair", tags: ["furniture", "chair","Home","Decor"], status: "A", height: 12 }, { name: "Basketball", tags: ["sports equipment", "basketball"], status: "B", height: 10 } ]);

db.inventories.insertMany([ { item: "Laptop", warehouse: "Warehouse A", quantity: 500 }, { item: "T-shirt", warehouse: "Warehouse B", quantity: 100 }, { item: "Notebook", warehouse: "Warehouse C", quantity: 200 }, { item: "Chair", warehouse: "Warehouse A", quantity: 50 }, { item: "Basketball", warehouse: "Warehouse B", quantity: 400 } ]);

db.inventories.insertOne({ item: "Planner", warehouse: "Warehouse B", quantity: 10 }) db.items.insertOne({ name: "Planner", tags: ["office supplies", "notebook","Stationary","School","Official"], status: "C", height: 8 })

List all the items where quantity is greater than 300

db.inventories.find({ quantity: { $gt: 300 } }, { \_id: 0, item: 1, quantity: 1 })

List all items which have less than 5 tags

db.items.find({ $expr: { $lt: [{ $size: "$tags" }, 5] } })

List all items having status equal to "B" or having quantity less than 50 and height of the product greater than 8

db.items.find({ $or: [ { status: "B" },{ $and: [{ quantity: { $lt: 50 } }, { height: { $gt: 8 } }] } ]}); Find all warehouses that keep the item "Planner" and have in-stock quantity less than 20

db.inventories.find({ item: "Planner", quantity: { $lt: 20 } }, { \_id: 0, warehouse: 1 })

**Slip 9**

Consider the Worker table

Worker(WORKER\_ID,FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_D

ATE,DEPARTMENT)

Write a query to insert 5 rows in it that as per the query requirements.

CREATE DATABASE ORG;

SHOW DATABASES;

USE ORG;

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker (WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT)

VALUES

(1, 'John', 'Doe', 95000, '2021-03-15', 'Finance'),

(2, 'Jane', 'Smith', 87000, '2021-05-20', 'IT'),

(3, 'Michael', 'Johnson', 105000, '2021-02-10', 'Finance'),

(4, 'Emily', 'Brown', 92000, '2021-07-08', 'Marketing'),

(5, 'David', 'Lee', 98000, '2021-04-25', 'IT');

1)List the departments where the total salary expenditure is less than $300,000,

sorted by total salary expenditure in ascending order

SELECT DEPARTMENT, SUM(SALARY) AS total\_salary\_expenditure

FROM Worker

GROUP BY DEPARTMENT

HAVING total\_salary\_expenditure < 300000

ORDER BY total\_salary\_expenditure ASC;

2)Find the worker with the highest salary in each department, sorted by

department name:

SELECT w1.\*FROM Worker w1 JOIN (SELECT DEPARTMENT, MAX(SALARY) AS max\_salary FROM Worker

GROUP BY DEPARTMENT) w2 ON w1.DEPARTMENT = w2.DEPARTMENT AND w1.SALARY = w2.max\_salary

ORDER BY w1.DEPARTMENT;

Model the following Customer Loan information as a document database. Consider Customer Loan information system where the customer can take many types of loans. Assume appropriate attributes and collections as per the query requirements

db.createCollection("customers");

db.createCollection("loans")

db.customers.insertMany([ { name: "David", address: "Vallab Nagar", city: "Pimpri" }, { name: "Amaan", address: "st Andrews Bandra", city: "Mumbai" }, { name: "Vanshay", address: "Chinchwad", city: "Pune" }, { name: "Dharam", address: "Rawat", city: "Pimpri" }, { name: "Mr.Patil", address: "Andheri", city: "Mumbai" }, { name: "Derek", address: "Viman Nagar", city: "Pune" }, { name: "Dinesh", address: "kasarwadi", city: "Pimpri" }, { name: "Rushikesh", address: "Churchgate", city: "Mumbai" }, { name: "Aditya", address: "Dadar", city: "Pune" }, { name: "Duncan", address: "777 Elm St", city: "Pimpri" } ])

db.loans.insertMany([ { customer\_id: ObjectId("65fc8c748b4816bd1a1d4983"), loan\_type: "Personal", loan\_amount: 10000 }, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4984"), loan\_type: "Home", loan\_amount: 200000 }, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4985"), loan\_type: "Car", loan\_amount: 50000 }, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4986"), loan\_type: "Education", loan\_amount: 30000}, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4987"), loan\_type: "Personal", loan\_amount: 150000}, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4988"), loan\_type: "Home", loan\_amount: 250000 }, { customer\_id: ObjectId("65fc8c748b4816bd1a1d4989"), loan\_type: "Car", loan\_amount: 60000 }, { customer\_id: ObjectId("65fc8c748b4816bd1a1d498a"), loan\_type: "Education", loan\_amount: 40000}, { customer\_id: ObjectId("65fc8c748b4816bd1a1d498b"), loan\_type: "Personal", loan\_amount: 200000}, { customer\_id: ObjectId("65fc8c748b4816bd1a1d498c"), loan\_type: "Home", loan\_amount: 300000 }])

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

db.customers.find({ name: /^D/ })

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

db.loans.find( { loan\_amount: { $gt: 100000 } })

Display customer details having the maximum loan amount.

db.loans.aggregate([ { $group: { \_id: "$customer\_id", maxLoanAmount: { $max: "$loan\_amount" } } }, { $lookup: { from: "customers", localField: "\_id", foreignField: "\_id", as: "customerDetails" } }, { $unwind: "$customerDetails" }, { $sort: { maxLoanAmount: -1 } }, { $limit: 1 }, { $project: { \_id: 0, "Customer Name": "$customerDetails.name", "Address": "$customerDetails.address", "City": "$customerDetails.city", "Max Loan Amount": "$maxLoanAmount" } } ])

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

db.customers.updateOne( { name: "Mr.Patil", }, { $set: { address: "Dahanu" } } )

**Slip 10**

Consider the Employee table

Employee(emp\_id,emp\_name,job\_name,manager\_id,hire\_date,salary)

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

job\_name VARCHAR(255),

manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2)

);

INSERT INTO Employee (emp\_id, emp\_name, job\_name, manager\_id, hire\_date, salary)

VALUES

(1, 'John Doe', 'Manager', NULL, '2022-01-05', 2500),

(2, 'Jane Smith', 'Engineer', 1, '2022-02-10', 2200),

(3, 'Michael Johnson', 'Engineer', 1, '2022-03-15', 2300),

(4, 'Emily Brown', 'Analyst', 2, '2022-04-20', 2100),

(5, 'David Lee', 'Analyst', 3, '2022-05-25', 2400);

1)List the manager\_id and the average salary of employees managed by each

manager who manages at least 2 employees, sorted by average salary in

descending order:

SELECT manager\_id, AVG(salary) AS avg\_salary

FROM Employee

WHERE manager\_id IS NOT NULL

GROUP BY manager\_id

HAVING COUNT(emp\_id) >= 2

ORDER BY avg\_salary DESC;

2)Find the departments where the average salary of employees is greater than

the average salary of all employees, sorted by department name

SELECT job\_name, AVG(salary) AS dept\_avg\_salary

FROM Employee

GROUP BY job\_name

HAVING AVG(salary) > (SELECT AVG(salary) FROM Employee)

ORDER BY job\_name;

Q2)Model the following Online shopping information as a document database. Consider online shopping where the customer can get different products from different brands. Customers can rate the brands and products Assume appropriate attributes and collections as per the query requirements db.createCollection("customers")

db.createCollection("products")

db.createCollection("brands")

db.customers.insertMany([ { name: "Alice", city: "New York", purchase\_date: "15/08/2023", bill\_amount: 60000 }, { name: "Bob", city: "Los Angeles", purchase\_date: "15/08/2023", bill\_amount: 70000 }, { name: "Charlie", city: "Chicago", purchase\_date: "16/08/2023", bill\_amount: 45000 }, { name: "David", city: "New York", purchase\_date: "15/08/2023", bill\_amount: 80000 }, { name: "Eve", city: "San Francisco", purchase\_date: "16/08/2023", bill\_amount: 55000 } ])

db.products.insertMany([ { name: "Laptop", brand: "Dell", warranty\_period: "1 year" }, { name: "Smartphone", brand: "Apple", warranty\_period: "2 years" }, { name: "TV", brand: "Samsung", warranty\_period: "1 year" }, { name: "Watch", brand: "Fossil", warranty\_period: "1 year" }, { name: "Headphones", brand: "Sony", warranty\_period: "2 years" } ]) db.brands.insertMany([ { name: "Dell", rating: 4.5 }, { name: "Apple", rating: 4.8 }, { name: "Samsung", rating: 4.3 }, { name: "Fossil", rating: 4.2 }, { name: "Sony", rating: 4.4 } ])

List the names of product whose warranty period is one year

db.products.find({ warranty\_period: "1 year" }, { \_id: 0, name: 1 })

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

db.customers.find({ purchase\_date: "15/08/2023" })

Display the names of products with brand which have highest rating

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

db.products.find({ brand: "Apple" }, { \_id: 0, name: 1 })

Display customers who stay in …… city and billamt >50000

db.customers.find({ city: "New York", bill\_amount: { $gt: 50000 } })