**“SLIP 1**

Consider the Worker table

1)List the department and the total salary for each department, sorted by total salary in descending order:

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

CREATE TABLE Worker ( WORKER\_ID INT PRIMARY KEY, FIRST\_NAME VARCHAR(50), LAST\_NAME VARCHAR(50), SALARY DECIMAL(10, 2), JOINING\_DATE DATE, DEPARTMENT VARCHAR(50) )

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

(1, 'John', 'Doe', 90000, '2023-01-15', 'IT’),

(2, 'Jane', 'Smith', 110000, '2022-05-20', 'Finance’),

(3, 'Michael', 'Johnson', 95000, '2023-03-10', 'IT'),

(4, 'Emily', 'Brown', 120000, '2021-11-05', 'Finance'),

(5, 'David', 'Lee', 105000, '2022-09-12', ’IT’);

SELECT DEPARTMENT, SUM(SALARY) AS TOTAL\_SALARY

FROM Worker

GROUP BY DEPARTMENT

ORDER BY TOTAL\_SALARY DESC;

SELECT DEPARTMENT, AVG(SALARY) AS AVERAGE\_SALARY

FROM Worker

GROUP BY DEPARTMENT HAVING AVG(SALARY) > 100000

ORDER BY AVERAGE\_SALARY DESC;

Q2:- Model the following Property system as a document database. Consider a set of Property, Owner. One owner can buy many properties.

Sol:-

Use property\_management

db.createCollection("property")

db.property.insertMany([

{ "\_id": 1, "area": "Mumbai", "rate": 150000, "owner\_id": 1 },

{ "\_id": 2, "area": "Pune", "rate": 90000, "owner\_id": 2 },

{ "\_id": 3, "area": "Nashik", "rate": 120000, "owner\_id": 3 },

{ "\_id": 4, "area": "Nagpur", "rate": 80000, "owner\_id": 4 },

{ "\_id": 5, "area": "Mumbai", "rate": 200000, "owner\_id": 2 } ])

db.createCollection("owner")

db.owner.insertMany([

{ "\_id": 1, "name": "Mr. Singh" },

{ "\_id": 2, "name": "Mr. Patil" },

{ "\_id": 3, "name": "Ms. Rao" },

{ "\_id": 4, "name": "Mr. Kumar" },

{ "\_id": 5, "name": "Mrs. Sharma" } ])

a)Display area wise property details. [3]

db.property.aggregate([

{ $group: { \_id: "$area", properties: { $push: { rate: "$rate", owner\_id: "$owner\_id" } } } }

]);

b)Display property owned by 'Mr.Patil' having minimum rate [3]

db.property.find({ owner\_id: 2 }).sort({ rate: 1 }).limit(1);

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

db.owner.findOne({ \_id: db.property.findOne({ area: "Nashik" }).owner\_id })

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

db.property.find({ rate: { $lt: 100000 } }, { area: 1, \_id: 0 });

**SLIP 2**

Consider the Employee table

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

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

CREATE TABLE Employee ( emp id INT PRIMARY KEY,

emp\_name VARCHAR(100), job\_name VARCHAR(100), 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 Smith', 'Manager', NULL, '1990-05-15', 75000),

(2, 'Alice Johnson', 'Developer', 1, '1998-08-20', 65000),

(3, 'Bob Williams', 'Analyst', 1, '2000-02-10', 60000),

(4, 'Emily Davis', 'Developer', 1, '1999-04-25', 70000),

(5, 'Michael Brown', 'Manager', NULL, '1993-11-30', 80000);

SELECT job name, AVG(salary) AS average salary FROM Employee

GROUP BY job\_name

ORDER BY average\_saIary DESC;

SELECT job name, MAX(salary) AS max salary FROM Employee

WHERE hire date > '1995-01-01' GROUP BY job name

ORDER BY max salary DESC;

Q2 Model the following system as a document database. Consider a database of newspaper, publisher, and city. Different publisher publishes various newspapers in different cities

Sol:- use newspaper\_management

db.createCollection("newspapers")

db.createCollection("publishers")

db.createCollection("cities")

db.newspapers.insertMany([

{ "\_id": 1, "name": "The Times", "language": "English", "city\_id": 1, "publisher\_id": 1, "sale": 5000 },

{ "\_id": 2, "name": "Maharashtra Times", "language": "Marathi", "city\_id": 2, "publisher\_id": 2, "sale": 7000 },

{ "\_id": 3, "name": "Gujarat Samachar", "language": "Gujarati", "city\_id": 3, "publisher\_id": 3, "sale": 6000 },

{ "\_id": 4, "name": "Nashik Times", "language": "Marathi", "city\_id": 4, "publisher\_id": 4, "sale": 4500 },

{ "\_id": 5, "name": "Pune Mirror", "language": "English", "city\_id": 2, "publisher\_id": 2, "sale": 5500 }

]);

// Inserting documents into Publishers collection

db.publishers.insertMany([

{ "\_id": 1, "name": "Publisher A", "state": "Maharashtra" },

{ "\_id": 2, "name": "Publisher B", "state": "Maharashtra" },

{ "\_id": 3, "name": "Publisher C", "state": "Gujarat" },

{ "\_id": 4, "name": "Publisher D", "state": "Maharashtra" },

{ "\_id": 5, "name": "Publisher E", "state": "Gujarat" }

]);

// Inserting documents into Cities collection

db.cities.insertMany([

{ "\_id": 1, "name": "Mumbai", "state": "Maharashtra" },

{ "\_id": 2, "name": "Pune", "state": "Maharashtra" },

{ "\_id": 3, "name": "Ahmedabad", "state": "Gujarat" },

{ "\_id": 4, "name": "Nashik", "state": "Maharashtra" },

{ "\_id": 5, "name": "Surat", "state": "Gujarat" }

]);

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

db.newspapers.find({ city\_id: 4 });

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

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

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

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

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

var cursor = db.newspapers.find({ city\_id: { $in: [1, 2, 4] } }).sort({ sale: -1 }).limit(1);

while (cursor.hasNext()) {

printjson(cursor.next());

}

**Slip 3**

Consider the Worker table

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 TABLE Worker ( WORKER\_ID INT PRIMARY KEY, FIRST\_NAME VARCHAR(100), LAST\_NAME VARCHAR(100), SALARY DECIMAL(10, 2), JOINING\_DATE DATE, DEPARTMENT VARCHAR(100) )

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

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

(2, 'Alice', 'Johnson', 85000, '2022-05-20', 'Finance’),

(3, 'Bob', 'Williams', 105000, '2021-07-10', 'IT’),

(4, 'Emily', 'Davis', 92000, '2020-11-05', 'Finance’),

(5, 'Michael', 'Brown', 97000, '2021-09-12', 'IT');

SELECT DEPARTMENT, COUNT(\*) AS worker\_count FROM Worker

WHERE JOINING\_DATE > '2021-01-01' GROUP BY DEPARTMENT

ORDER BY DEPARTMENT;

SELECT DEPARTMENT

FROM Worker

WHERE SALARY > 90000 GROUP BY DEPARTMENT HAVING COUNT(\*) = (

SELECT COUNT(\*)

FROM Worker WHERE SALARY > 90000 GROUP BY DEPARTMENT ORDER BY COUNT(\*) DESC LIMIT 1

**Q2 Model the following system as a document database. Consider employee and department’s information.**

Sol:-

use company\_management

db.createCollection("employees")

db.createCollection("departments")

db.employees.insertMany([

{ "\_id": 1, "name": "John Doe", "department\_id": 1, "salary": 60000 },

{ "\_id": 2, "name": "Jane Smith", "department\_id": 2, "salary": 70000 },

{ "\_id": 3, "name": "Mike Johnson", "department\_id": 2, "salary": 55000 },

{ "\_id": 4, "name": "Emily Brown", "department\_id": 1, "salary": 65000 },

{ "\_id": 5, "name": "David Lee", "department\_id": 3, "salary": 50000 }

]);

db.departments.insertMany([

{ "\_id": 1, "name": "HR" },

{ "\_id": 2, "name": "Finance" },

{ "\_id": 3, "name": "Sales" },

{ "\_id": 4, "name": "Marketing" },

{ "\_id": 5, "name": "IT" }

]);

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

db.employees.find().sort({ salary: -1 }).limit(1);

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

db.employees.aggregate([

{ $group: { \_id: "$department\_id", count: { $sum: 1 } } },

{ $sort: { count: -1 } },

{ $limit: 1 },

{ $lookup: { from: "departments", localField: "\_id", foreignField: "\_id", as: "department" } },

{ $unwind: "$department" },

{ $project: { \_id: 0, department\_name: "$department.name", count: 1 } }

]);

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

var cursor = db.departments.find();

while (cursor.hasNext()) {

var department = cursor.next();

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

var employees = db.employees.find({ department\_id: department.\_id });

while (employees.hasNext()) {

var employee = employees.next();

print("\tEmployee Name: " + employee.name + ", Salary: " + employee.salary);

}

}

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

db.employees.find({ department\_id: 3, salary: { $gt: 50000 } });

**Slip 4**

**Consider the Patient table**

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 ( PatientlD INT PRIMARY KEY, Name VARCHAR(100),

DateOfBirth DATE, Gender VARCHAR(10),

admit\_date DATE, ward no INT,

City VARCHAR(100) )

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

(1, 'John Smith', '1985-05-15', 'Male', '2023-01-10', 101, 'New York’),

(2, 'Alice Johnson', '1990-08-20', 'Female', '2023-02-05', 102, 'Los Angeles’),

(3, 'Bob Williams', '1982-04-10', 'Male', '2023-03-15', 101, 'Chicago’),

(4, 'Emily Davis', '1988-11-05', 'Female', '2023-04-20', 103, 'San Francisco'),

(5, 'Michael Brown', '1995-09-12', 'Male', '2023-05-25', 102, 'Houston’);

SELECT ward\_no, COUNT(\*) AS patient\_count FROM Patient

GROUP BY ward\_no ORDER BY ward no;

SELECT ward no, AVG(DATEDIFF(CURRENT DATE, DateOfBirth) / 365) AS average age FROM Patient

GROUP BY ward no

HAVING AVG(DATEDIFF(CURRENT DATE, DateOfBirth) / 365) < 40

ORDER BY average\_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**

Sol:-

use hospital\_management

db.createCollection("hospitals")

db.createCollection("specializations")

db.createCollection("reviews")

db.createCollection("doctors")

// Inserting documents into Hospitals collection

db.hospitals.insertMany([

{ "\_id": 1, "name": "City Hospital", "city": "Nashik", "rating": 4.5, "specializations": [1, 2] },

{ "\_id": 2, "name": "Unity Hospital", "city": "Nashik", "rating": 3.8, "specializations": [2, 3] },

{ "\_id": 3, "name": "Grace Hospital", "city": "Mumbai", "rating": 4.2, "specializations": [1, 4] },

{ "\_id": 4, "name": "Hope Hospital", "city": "Pune", "rating": 4.0, "specializations": [3, 5] },

{ "\_id": 5, "name": "Heal Hospital", "city": "Nashik", "rating": 4.8, "specializations": [1, 2, 3] }

]);

// Inserting documents into Specializations collection

db.specializations.insertMany([

{ "\_id": 1, "name": "Pediatric" },

{ "\_id": 2, "name": "Gynaecology" },

{ "\_id": 3, "name": "Orthopedic" },

{ "\_id": 4, "name": "Cardiology" },

{ "\_id": 5, "name": "Neurology" }

]);

// Inserting documents into Reviews collection

db.reviews.insertMany([

{ "\_id": 1, "hospital\_id": 1, "reviewer\_name": "John Doe", "rating": 5 },

{ "\_id": 2, "hospital\_id": 2, "reviewer\_name": "Jane Smith", "rating": 4 },

{ "\_id": 3, "hospital\_id": 3, "reviewer\_name": "Mike Johnson", "rating": 4.5 },

{ "\_id": 4, "hospital\_id": 4, "reviewer\_name": "Emily Brown", "rating": 3.5 },

{ "\_id": 5, "hospital\_id": 5, "reviewer\_name": "David Lee", "rating": 4 }

]);

// Inserting documents into Doctors collection

db.doctors.insertMany([

{ "\_id": 1, "name": "Dr. Deshmukh", "hospitals": [1, 3, 5] },

{ "\_id": 2, "name": "Dr. Gupta", "hospitals": [2, 4] },

{ "\_id": 3, "name": "Dr. Patel", "hospitals": [3, 5] },

{ "\_id": 4, "name": "Dr. Singh", "hospitals": [1, 2, 4] },

{ "\_id": 5, "name": "Dr. Sharma", "hospitals": [1, 5] }

]);

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

db.hospitals.find({ specializations: 2 }, { name: 1, \_id: 0 });

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

db.hospitals.find({ city: "Nashik" }, { name: 1, \_id: 0 });

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

db.hospitals.find({ \_id: { $in: db.doctors.findOne({ name: "Dr. Deshmukh" }).hospitals } }, { name: 1, \_id: 0 });

d)List the names of hospitals whose rating >=4 [4]

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

**Slip 5**

**Consider the Patient table**

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 ( PatientlD INT PRIMARY KEY, Name VARCHAR(100),

DateOfBirth DATE, Gender VARCHAR(10),

admit date DATE, ward no INT,

City VARCHAR(100) )

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

(1, 'John Smith', '1985-05-15', 'Male', '2023-01-10', 101, 'New York’),

(2, 'Alice Johnson', '1990-08-20', 'Female', '2023-02-05', 102, 'Los Angeles’),

(3, 'Bob Williams', '1982-04-10', 'Male', '2023-03-15', 101, 'Chicago’),

(4, 'Emily Davis', '1988-11-05', 'Female', '2023-04-20', 103, 'San Francisco'),

(5, 'Michael Brown', '1995-09-12', 'Male', '2023-05-25', 102, 'Houston’);

SELECT ward\_no, Gender, COUNT(\*) AS patient\_count FROM Patient

GROUP BY ward no, Gender ORDER BY ward no, Gender;

SELECT ward no, COUNT(\*) AS patient count FROM Patient

GROUP BY ward\_no

ORDER BY patient\_count DESC LIMIT 3;

**Q2 Model the following database. Many employees working on one project. A company has various ongoing projects.**

Sol:-

use company\_management

db.createCollection("employees")

db.createCollection("projects")

db.employees.insertMany([

{ "\_id": 1, "name": "John Doe", "project\_id": 1 },

{ "\_id": 2, "name": "Jane Smith", "project\_id": 2 },

{ "\_id": 3, "name": "Mike Johnson", "project\_id": 1 },

{ "\_id": 4, "name": "Emily Brown", "project\_id": 3 },

{ "\_id": 5, "name": "David Lee", "project\_id": 2 }

]);

db.projects.insertMany([

{ "\_id": 1, "name": "Project A", "duration": 4, "project\_type": "Type1" },

{ "\_id": 2, "name": "Project B", "duration": 5, "project\_type": "Type2" },

{ "\_id": 3, "name": "Project C", "duration": 2, "project\_type": "Type1" },

{ "\_id": 4, "name": "Project D", "duration": 6, "project\_type": "Type3" },

{ "\_id": 5, "name": "Project E", "duration": 3, "project\_type": "Type2" }

]);

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

db.projects.find({ project\_type: "Type1" }, { name: 1, \_id: 0 });

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

db.projects.find({ duration: { $gt: 3 } });

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

var projectId = db.projects.findOne({ "name": projectName }).\_id;

print("Project ID:", projectId);

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

var employeeName = "Mr. Patil"; // Assuming the employee name is "Mr. Patil"

var employee = db.employees.findOne({ name: employeeName });

if (employee) {

var projects = db.projects.find({ \_id: employee.project\_id }, { name: 1, \_id: 0 });

projects.forEach(function(project) {

print(project.name);

});

} else {

print("Employee not found!");

}

**Slip 6**

**Consider the Worker table**

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 TABLE Worker ( WORKER\_ID INT PRIMARY KEY, FIRST\_NAME VARCHAR(100), LAST\_NAME VARCHAR(100), SALARY DECIMAL(10, 2), JOINING\_DATE DATE, DEPARTMENT VARCHAR(100) )

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

VALUES

(1, 'John', 'Doe', 90000, '2021-02-10', 'IT’),

(2, 'Alice', 'Johnson', 85000, '2021-02-15', 'Finance’),

(3, 'Bob', 'Williams', 95000, '2021-02-20', 'IT’),

(4, 'Emily', 'Davis', 82000, '2021-02-25', 'Finance'),

(5, 'Michael', 'Brown', 88000, '2021-02-28', 'IT');

SELECT DEPARTMENT, COUNT(\*) AS worker\_count FROM Worker

WHERE JOINING\_DATE >= '2021-02-01' AND JOINING\_DATE <= '2021-02-28' AND SALARY > 80000

GROUP BY DEPARTMENT ORDER BY DEPARTMENT;

SELECT DEPARTMENT, AVG(SALARY) AS avg salary

FROM Worker

GROUP BY DEPARTMENT HAVING COUNT(\*) >= 2

ORDER BY avg salary DESC LIMIT 1;

**Q2 Model the following information as a document database. A customer can take different policies and get the benefit. There are different types of policies provided by various companies**

Sol:-

use insurance\_management

db.createCollection("customers")

db.createCollection("policies")

// Inserting documents into customers collection

db.customers.insertMany([

{ "\_id": 1, "name": "John Doe", "policy\_id": 1, "premium\_amount": 5000 },

{ "\_id": 2, "name": "Jane Smith", "policy\_id": 2, "premium\_amount": 7000 },

{ "\_id": 3, "name": "Mike Johnson", "policy\_id": 1, "premium\_amount": 5500 },

{ "\_id": 4, "name": "Emily Brown", "policy\_id": 3, "premium\_amount": 6000 },

{ "\_id": 5, "name": "David Lee", "policy\_id": 2, "premium\_amount": 4500 }

]);

// Inserting documents into policies collection

db.policies.insertMany([

{ "\_id": 1, "name": "Komal Jeevan", "type": "Monthly" },

{ "\_id": 2, "name": "Health Plus", "type": "Yearly" },

{ "\_id": 3, "name": "Life Secure", "type": "Half Yearly" },

{ "\_id": 4, "name": "Wealth Builder", "type": "Monthly" },

{ "\_id": 5, "name": "Retirement Plan", "type": "Monthly" }

]);

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

db.customers.find({ "policy\_id": 1 }).pretty()

b)Display average premium amount

db.customers.aggregate([

{ $group: { \_id: null, avgPremium: { $avg: "$premium\_amount" } } }

]);

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

db.customers.updateMany(

{ "policy\_id": { $in: db.policies.find({ "type": "Monthly" }).toArray().map(policy => policy.\_id) } },

{ $mul: { "premium\_amount": 1.05 } }

);

d)Count no. of customers who have taken policy type “half yearly”.

db.customers.count({ "policy\_id": { $in: db.policies.find({ "type": "Half Yearly" }).toArray().map(policy => policy.\_id) } });

**Slip 7**

**Consider the Employee table**

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

$2000, sorted by department 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:

CREATE TABLE Employee ( emp id INT PRIMARY KEY,

emp\_name VARCHAR(100), job\_name VARCHAR(100), manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2), department VARCHAR(100) )

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

(1, 'John Smith', 'Manager', 0, '2022-01-10', 3000, 'IT’),

(2, 'Alice Johnson', 'Developer', 1, '2022-02-15', 2500, 'Finance'),

(3, 'Bob Williams', 'Analyst', 1, '2022-03-20', 2200, 'IT’),

(4, 'Emily Davis', 'Developer', 1, '2022-04-25', 2800, 'Finance'),

(5, 'Michael Brown', 'Manager', 0, '2022-05-30', 3200, 'IT');

SELECT department, COUNT(\*) AS num\_empIoyees FROM Employee

WHERE salary > 2000 GROUP BY department ORDER BY department;

SELECT manager id, COUNT(\*) AS num managed employees FROM Employee

WHERE manager\_id <> 0 GROUP BY manager\_id HAVING COUNT(\*) > 1

ORDER BY manager id;

**Q2 . Model the following information as a document database. A customer operates his bank account, does various transactions and get the banking services**

Sol:-

use banking\_management

db.createCollection("customers")

db.createCollection("accounts")

db.customers.insertMany([

{ "\_id": 1, "first\_name": "John", "last\_name": "Doe", "branch": "Branch A" },

{ "\_id": 2, "first\_name": "Jane", "last\_name": "Smith", "branch": "Branch B" },

{ "\_id": 3, "first\_name": "Mike", "last\_name": "Johnson", "branch": "Branch A" },

{ "\_id": 4, "first\_name": "Sarah", "last\_name": "Brown", "branch": "Branch A" },

{ "\_id": 5, "first\_name": "Steven", "last\_name": "Lee", "branch": "Branch C" }

]);

db.accounts.insertMany([

{ "\_id": 1, "customer\_id": 1, "open\_date": ISODate("2020-01-01"), "acctype": "Saving", "branch": "Branch A" },

{ "\_id": 2, "customer\_id": 2, "open\_date": ISODate("2020-01-01"), "acctype": "Current", "branch": "Branch B" },

{ "\_id": 3, "customer\_id": 3, "open\_date": ISODate("2020-02-01"), "acctype": "Saving", "branch": "Branch A" },

{ "\_id": 4, "customer\_id": 4, "open\_date": ISODate("2020-01-01"), "acctype": "Loan", "branch": "Branch A" },

{ "\_id": 5, "customer\_id": 5, "open\_date": ISODate("2020-01-01"), "acctype": "Saving", "branch": "Branch C" }

]);

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

db.customers.find({ "first\_name": /^S/ }, { "first\_name": 1, "last\_name": 1 })

b)List all customers who has open an account on 1/1/2020 in \_\_\_branch

db.customers.find({ "branch": "Branch A" }).forEach(function(customer) {

var account = db.accounts.findOne({ "customer\_id": customer.\_id, "open\_date": ISODate("2020-01-01") });

if (account) {

print("Name: " + customer.first\_name + " " + customer.last\_name + ", Branch: " + customer.branch);

}

});

c)List the names customers where acctype=”Saving”

db.customers.aggregate([

{ $lookup: { from: "accounts", localField: "\_id", foreignField: "customer\_id", as: "account\_info" } },

{ $unwind: "$account\_info" },

{ $match: { "account\_info.acctype": "Saving" } },

{ $project: { "\_id": 0, "first\_name": 1, "last\_name": 1 } }

])

d. Count total no. of loan account holder of …….branch [4] 20Viva 5

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

**Slip 8**

**Consider the Employee table**

1)Find the department with the highest average salary among departments with at least 3 employees:

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

CREATE TABLE Employee ( emp id INT PRIMARY KEY,

emp\_name VARCHAR(100), job\_name VARCHAR(100), manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2), department VARCHAR(100) )

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

(1, 'John Doe', 'Manager', NULL, '2023-01-10', 5000, 'IT’),

(2, 'Alice Johnson', 'Developer', 1, '2023-02-15', 4000, 'Finance'),

(3, 'Bob Williams', 'Analyst', 1, '2023-03-20', 3500, 'IT’),

(4, 'Emily Davis', 'Developer', 1, '2023-04-25', 4500, 'Finance'),

(5, 'Michael Brown', 'Manager', NULL, '2023-05-30', 5200, 'IT');

SELECT department, AVG(salary) AS avg\_saIary FROM Employee

GROUP BY department HAVING COUNT(\*) >= 3

ORDER BY avg salary DESC LIMIT 1;

SELECT department, SUM(salary) AS totaI\_saIary\_expenditure FROM Employee

GROUP BY department

ORDER BY totaI\_saIary\_expenditure DESC LIMIT 3;

**Q2 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.**

Sol:-

use inventory\_management

db.createCollection("items")

db.createCollection("warehouses")

// Inserting documents into items collection

db.items.insertMany([

{ "\_id": 1, "name": "Chair", "quantity": 350, "tags": ["furniture", "office"], "status": "A", "height": 10 },

{ "\_id": 2, "name": "Desk", "quantity": 200, "tags": ["furniture", "office"], "status": "B", "height": 12 },

{ "\_id": 3, "name": "Lamp", "quantity": 100, "tags": ["lighting"], "status": "C", "height": 8 },

{ "\_id": 4, "name": "Planner", "quantity": 50, "tags": ["stationery"], "status": "B", "height": 6 },

{ "\_id": 5, "name": "Printer", "quantity": 400, "tags": ["electronics", "office"], "status": "A", "height": 15 }

]);

// Inserting documents into warehouses collection

db.warehouses.insertMany([

{ "\_id": 1, "warehouse\_name": "Warehouse A", "items": [{ "item\_id": 1, "quantity": 200 }, { "item\_id": 2, "quantity": 100 }] },

{ "\_id": 2, "warehouse\_name": "Warehouse B", "items": [{ "item\_id": 3, "quantity": 50 }, { "item\_id": 4, "quantity": 25 }] },

{ "\_id": 3, "warehouse\_name": "Warehouse C", "items": [{ "item\_id": 5, "quantity": 200 }] }

]);

a)List all the items qty is greater than 300

db.items.find({ "quantity": { $gt: 300 } })

b) List all items which have tags less than 5

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

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

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

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

db.warehouses.find({ "items": { $elemMatch: { "item\_id": 4, "quantity": { $lt: 20 } } } })

**Slip 9**

**Consider the Worker table**

1)List the departments where the total salary expenditure is less than $300,000, sorted by total salary expenditure in ascending order

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

CREATE TABLE Worker ( WORKER\_ID INT PRIMARY KEY, FIRST\_NAME VARCHAR(100), LAST\_NAME VARCHAR(100), SALARY DECIMAL(10, 2), JOINING\_DATE DATE, DEPARTMENT VARCHAR(100) )

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

VALUES

(1, 'John', 'Doe', 50000, '2023-01-10', 'IT’),

(2, 'Alice', 'Johnson', 45000, '2023-02-15', 'Finance’),

(3, 'Bob', 'Williams', 60000, '2023-03-20', 'IT’),

(4, 'Emily', 'Davis', 55000, '2023-04-25', 'Finance'),

(5, 'Michael', 'Brown', 70000, '2023-05-30', 'IT');

SELECT DEPARTMENT, SUM(SALARY) AS totaI\_saIary\_expenditure FROM Worker

GROUP BY DEPARTMENT

HAVING total salary expenditure < 300000 ORDER BY total salary expenditure ASC;

SELECT w1.DEPARTMENT, w1.FIRST\_NAME, w1.LAST\_NAME, w1.SALARY

FROM Worker w1 JOIN (SELECT DEPARTMENT, MAX(SALARY) AS max\_saIary FROM Worker GROUP BY DEPARTMENT) w2 ON w1.DEPARTMENT w2.DEPARTMENT AND w1.SALARY w2.max\_saIary ORDER BY w1.DEPARTMENT;

**Q2 Model the following Customer Loan information as a document database.**

**Consider Customer Loan information system where the customer can take**

**many types of loans.**

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 }])

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

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

B. List the names of customer in descending order who has taken a loan

from Pimpri city.

db.customers.find({ "city": "Pimpri" }).sort({ "name": -1 })

C. Display customer details having maximum loan amount.

db.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" } }

])

d)update the address of customer whose name is “Mr. Patil” and loan\_amt

is greater than 100000.

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

**Slip10**

**Consider the Employee table**

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:

2)Find the departments where the average salary of employees is greater than the average salary of all employees, sorted by department name

CREATE TABLE Employee ( emp id INT PRIMARY KEY,

emp\_name VARCHAR(100), job\_name VARCHAR(100), manager\_id INT,

hire\_date DATE,

salary DECIMAL(10, 2), department VARCHAR(100) )

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

VALUES

(1, 'John Doe', 'Manager', NULL, '2023-01-10', 5000, 'IT’),

(2, 'Alice Johnson', 'Developer', 1, '2023-02-15', 4500, 'Finance'),

(3, 'Bob Williams', 'Analyst', 1, '2023-03-20', 6000, 'IT’),

(4, 'Emily Davis', 'Developer', 1, '2023-04-25', 5500, 'Finance'),

(5, 'Michael Brown', 'Manager', NULL, '2023-05-30', 7000, 'IT'),

(6, 'Jessica Lee', 'Analyst', 5, '2023-06-15', 6000, 'IT’),

(7, 'David Smith', 'Developer', 5, '2023-07-20', 5500, 'Finance'),

(8, 'Sarah Wilson', 'Developer', 5, '2023-08-25', 5800, 'IT');

SELECT manager id, AVG(salary) AS avg salary FROM Employee

WHERE manager id IS NOT NULL GROUP BY manager id

HAVING COUNT(\*) >= 2

ORDER BY avg\_saIary DESC;

SELECT department FROM Employee GROUP BY department

HAVING AVG(salary) > (SELECT AVG(salary) FROM Employee) ORDER BY department;

**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**

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 }

])

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

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

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

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

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

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

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

D. Display customers who stay in …… city and billamt &gt;50000 .[4]

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