# **April 21, 2021 LAB\_4:**

### Problem Statement

Create a Collection called “Staff” under the database “Institute”, Add at least 10 records into the

collection. Assume the following to be the details to be incorporated in the document.

### Document Format

1. Name of the staff

2. Unique Staff ID – example – 086, (create it using \_id)

3. Department – [“Human resource”, “Development Team”, “Administration”, “Maintenance”.. ]

4. Salary – int

5. Age – int

6. Designation - string

7. Increment – Yes/No

8. Performance status = [Good, Average, Excellent, Not Satisfactory]

**use institute;**

Switched to db institute

**db.createCollection(“staff”);**

{"ok" : 1}

**db.staff.insertMany([**

**... {"\_id" : 1, name : "sachin", department : [ "human resource", "development team" ], salary : 20000, age : 21, designation : "graduation", increment : "yes", "performance" : "good" },**

**... {"\_id" : 2, "name" : "charan", "department" : [ "human resource", "learner" ], "salary" : 25000, "age" : 23, "designation" : "graduation", "increment" : "yes", "performance" : "excellent" },**

**... { "\_id" : 3, "name" : "shivam", "department" : [ "human resource", ], "salary" : 45000, "age" : 30, "designation" : "postgraduation", "increment" : "yes", "performance" : "excellent" },**

**... {"\_id" : 4, "name" : "vivek", "department" : [ "human resource", "maintenance" ], "salary" : 25000, "age" : 20, "designation" : " graduation", "increment" : "no", "performance" : "good" },**

**... { "\_id" : 5, "name" : "vivek", "department" : [ "human resource", "development team", "coder" ], "salary" : 35000, "age" : 21, "designation" : " graduation", "increment" : "yes", "performance" : "excellent" }**

**... ]);**

{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4, 5 ] }

**db.staff.insertMany([**

**{"\_id" : 6, "name" : "sachin", "department" : [ "development team", "admin" ], "salary" : 30000, "age" : 21, "designation" : " undergraduate", "increment" : "no", "performance" : "average" }, {"\_id" : 7, "name" : "rajesh", "department" : [ "development team" ], "salary" : 58000, "age" : 27, "designation" : "director", "increment" : "no", "performance" : "excellent" }, {"\_id" : 8, "name" : "ramesh", "department" : [ "admin" ], "salary" : 50000, "age" : 25, "designation" : "assistant director", "increment" : "no", "performance" : "not satisfactory" }, {"\_id" : 9, "name" : "sameer", "department" : [ "admin" ], "salary" : 40000, "age" : 45, "designation" : "board member", "increment" : "yes", "performance" : "average" }, {"\_id" : 10, "name" : "nitin", "department" : [ "human resource" ], "salary" : 30000, "age" : 29, "designation" : "hr", "increment" : "no", "performance" : "good" } ]);**

{ "acknowledged" : true, "insertedIds" : [ 6, 7, 8, 9, 10 ] }

### Execute the following queries and record the output in the observation

1. Group all the records by the department type and calculate the average sum of the salary from each department.

**db.staff.aggregate({ $group: { \_id: "$department", avgDptSal: { $avg: "$salary"}}});**

{ "\_id" : [ "human resource", "maintenance" ], "avgDptSal" : 25000 }

{ "\_id" : [ "human resource", "development team" ], "avgDptSal" : 20000 }

{ "\_id" : [ "human resource", "development team", "coder" ], "avgDptSal" : 35000 }

{ "\_id" : [ "development team", "admin" ], "avgDptSal" : 30000 }

{ "\_id" : [ "human resource" ], "avgDptSal" : 37500 }

{ "\_id" : [ "human resource", "learner" ], "avgDptSal" : 25000 }

{ "\_id" : [ "development team" ], "avgDptSal" : 58000 }

{ "\_id" : [ "admin" ], "avgDptSal" : 45000 }

1. Group all the records by the staffid, find out the average salary by the age group

**db.staff.aggregate({ $group:{ \_id: "$\_id", agegroup: { $avg: "$salary"}}});**

{ "\_id" : 10, "agegroup" : 30000 }

{ "\_id" : 1, "agegroup" : 20000 }

{ "\_id" : 4, "agegroup" : 25000 }

{ "\_id" : 2, "agegroup" : 25000 }

{ "\_id" : 3, "agegroup" : 45000 }

{ "\_id" : 6, "agegroup" : 30000 }

{ "\_id" : 9, "agegroup" : 40000 }

{ "\_id" : 7, "agegroup" : 58000 }

{ "\_id" : 8, "agegroup" : 50000 }

{ "\_id" : 5, "agegroup" : 35000 }

1. Apply the map-reduce aggregation to project the name and amount owned by each staff by doing multiple jobs in a different department.

**> var mapperFunction = function(){ emit(this.name, this.salary);};**

**> var reducerFunction = function(name, sal){ return Array.sum(sal);};**

**> db.staff.mapReduce(mapperFunction, reducerFunction,{ out: "myoutput"});**

{ "result" : "myoutput", "ok" : 1 }

**> db.myoutput.find();**

{ "\_id" : "rajesh", "value" : 58000 }

{ "\_id" : "shivam", "value" : 45000 }

{ "\_id" : "ramesh", "value" : 50000 }

{ "\_id" : "sachin", "value" : 50000 }

{ "\_id" : "charan", "value" : 25000 }

{ "\_id" : "vivek", "value" : 60000 }

{ "\_id" : "sameer", "value" : 40000 }

{ "\_id" : "nitin", "value" : 30000 }

1. Match all the records having the performance status as “Good”, group them by their Name and compute the salary for each of them.

**> db.staff.aggregate([{ $match: { "performance": { $in: ["good"]}}}, { $group: { \_id: "$name", Salary: { $sum: "$salary"}}}]);**

{ "\_id" : "sachin", "Salary" : 20000 }

{ "\_id" : "vivek", "Salary" : 25000 }

{ "\_id" : "nitin", "Salary" : 30000 }

1. Demonstrate the usage of $match, $group, aggregate pipelines. Demonstrate the usage of $min, $last, $first, $sum, $max query operators with the $group operator.

**> db.staff.aggregate([ { $match: { "performance":{ $in: ["excellent","good"]}}}, { $group: { \_id: "$department", Sal: { $sum: "$salary"}, maxSal: { $max:"$salary"}, minSal: { $min:"$salary"}, firstSal: { $first:"$salary"}, lastSal:{ $last: "$salary"}}}]);**

{ "\_id" : [ "human resource", "maintenance" ], "Sal" : 25000, "maxSal" : 25000, "minSal" : 25000, "firstSal" : 25000, "lastSal" : 25000 }

{ "\_id" : [ "human resource", "development team" ], "Sal" : 20000, "maxSal" : 20000, "minSal" : 20000, "firstSal" : 20000, "lastSal" : 20000 }

{ "\_id" : [ "human resource", "development team", "coder" ], "Sal" : 35000, "maxSal" : 35000, "minSal" : 35000, "firstSal" : 35000, "lastSal" : 35000 }

{ "\_id" : [ "human resource" ], "Sal" : 75000, "maxSal" : 45000, "minSal" : 30000, "firstSal" : 45000, "lastSal" : 30000 }

{ "\_id" : [ "human resource", "learner" ], "Sal" : 25000, "maxSal" : 25000, "minSal" : 25000, "firstSal" : 25000, "lastSal" : 25000 }

{ "\_id" : [ "development team" ], "Sal" : 58000, "maxSal" : 58000, "minSal" : 58000, "firstSal" : 58000, "lastSal" : 58000 }

1. Demonstrate the updateOne, UpdateMany, and replaceOne operations with suitable examples.

#### updateOne

**> db.staff.find({"\_id": 6});**

{ "\_id" : 6, "name" : "sachin", "department" : [ "development team", "admin" ], "salary" : 30000, "age" : 21, "designation" : " undergraduate", "increment" : "no", "performance" : "average" }

**> db.staff.updateOne({\_id:6},{$set:{"performance":"good"}});**

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

> db.staff.find({"\_id": 6});

{ "\_id" : 6, "name" : "sachin", "department" : [ "development team", "admin" ], "salary" : 30000, "age" : 21, "designation" : " undergraduate", "increment" : "no", "performance" : "good" }

#### updateMany

> db.staff.find({"performance": "good"});

{ "\_id" : 1, "name" : "sachin", "department" : [ "human resource", "development team" ], "salary" : 20000, "age" : 21, "designation" : "graduation", "increment" : "yes", "performance" : "good" }

{ "\_id" : 4, "name" : "vivek", "department" : [ "human resource", "maintenance" ], "salary" : 25000, "age" : 20, "designation" : " graduation", "increment" : "no", "performance" : "good" }

{ "\_id" : 6, "name" : "sachin", "department" : [ "development team", "admin" ], "salary" : 30000, "age" : 21, "designation" : " undergraduate", "increment" : "no", "performance" : "good" }

{ "\_id" : 10, "name" : "nitin", "department" : [ "human resource" ], "salary" : 30000, "age" : 29, "designation" : "hr", "increment" : "no", "performance" : "good" }

**>db.staff.updateMany({"performance":"good"},{$set:{"performance":"average"}});**

{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 4 }

> db.staff.find({"performance": "good"});

> db.staff.find({"performance": "average"});

{ "\_id" : 1, "name" : "sachin", "department" : [ "human resource", "development team" ], "salary" : 20000, "age" : 21, "designation" : "graduation", "increment" : "yes", "performance" : "average" }

{ "\_id" : 4, "name" : "vivek", "department" : [ "human resource", "maintenance" ], "salary" : 25000, "age" : 20, "designation" : " graduation", "increment" : "no", "performance" : "average" }

{ "\_id" : 6, "name" : "sachin", "department" : [ "development team", "admin" ], "salary" : 30000, "age" : 21, "designation" : " undergraduate", "increment" : "no", "performance" : "average" }

{ "\_id" : 9, "name" : "sameer", "department" : [ "admin" ], "salary" : 40000, "age" : 45, "designation" : "board member", "increment" : "yes", "performance" : "average" }

{ "\_id" : 10, "name" : "nitin", "department" : [ "human resource" ], "salary" : 30000, "age" : 29, "designation" : "hr", "increment" : "no", "performance" : "average" }

#### replaceOne

> db.staff.find({"age": 29});

{ "\_id" : 10, "name" : "nitin", "department" : [ "human resource" ], "salary" : 30000, "age" : 29, "designation" : "hr", "increment" : "no", "performance" : "average" }

**> db.staff.replaceOne({"age": 29}, {"salary": 32000});**

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

> db.staff.find({"\_id": 10});

{ "\_id" : 10, "salary" : 32000 }

# 