Session 4

21/04/21

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]

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

```
> db.Staff.aggregate({$group:{_id:"$dept", Avg_Sal : { $avg:
"$salary"}}}); { "_id" : "Human Resources", "Avg_Sal" : 55000 }
{ "_id" : "Administration", "Avg_Sal" : 63333.333333333336 }
{ "_id" : "Maintanence", "Avg_Sal" : 27500 }
{ "_id" : "Development Team", "Avg_Sal" : 58333.333333333336 }
```

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

```
> db.Staff.aggregate({$group:{_id:"$age", Avg_Sal : {$avg: "$salary"}}});
{ "_id" : 47, "Avg_Sal" : 80000 }
{ "_id" : 35, "Avg_Sal" : 70000 }
{ "_id" : 45, "Avg_Sal" : 80000 }
{ "_id" : 36, "Avg_Sal" : 50000 }
{ "_id" : 25, "Avg_Sal" : 25000 }
{ "_id" : 26, "Avg_Sal" : 25000 }
{ "_id" : 32, "Avg_Sal" : 45000 }
{ "_id" : 37, "Avg_Sal" : 60000 }
{ "_id" : 27, "Avg_Sal" : 50000 }
>
```

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

```
> var mapper = function() {emit(this.name,this.salary)}
> var reduce = function(Name, Sal) {return Array.sum(Sal)}
> db.Staff.mapReduce(mapper, reduce, {out: "SalaryOut"})
{ "result" : "SalaryOut", "ok" : 1 }
> db.SalaryOut.find().pretty();
{ "_id" : "Sarthak", "value" : 60000 }
{ "_id" : "Indira", "value" : 50000 }
{ "_id" : "Magnus", "value" : 30000 }
{ "_id" : "Abhiram", "value" : 25000 }
{ "_id" : "Gauri", "value" : 60000 }
{ "_id" : "Colin", "value" : 70000 }
{ "_id" : "Terrence", "value" : 80000 }
{ "_id" : "Parker", "value" : 25000 }
{ "_id" : "Cyril", "value" : 50000 }
{ "_id" : "Aditi", "value" : 80000 }
>
```

4. 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: "Good"}}, { $group: {_id: "$name", Salary:
```

{$sum: "$salary"}}}]);
{ "_id" : "Sarthak", "Salary" : 60000 }
{ "_id" : "Colin", "Salary" : 70000 }
{ "_id" : "Gauri", "Salary" : 60000 }
>

5. 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({ $group: {_id: "$dept", Salary: {$sum:
"$salary"}}}); { "_id" : "Administration", "Salary" : 190000 }
{ "_id" : "Maintanence", "Salary" : 55000 }
{ "_id" : "Development Team", "Salary" : 175000 }
{ "_id" : "Human Resources", "Salary" : 110000 }
> db.Staff.aggregate({ $group: {_id: "$dept", Salary: {$min:
"$salary"}}}); { "_id" : "Maintanence", "Salary" : 25000 }
{ "_id" : "Administration", "Salary" : 50000 }
{ "_id" : "Human Resources", "Salary" : 50000 }
{ "_id" : "Development Team", "Salary" : 25000 }
> db.Staff.aggregate({ $group: {_id: "$dept", Salary: {$max:
"$salary"}}}); { "_id" : "Maintanence", "Salary" : 30000 }
{ "_id" : "Administration", "Salary" : 80000 }
{ "_id" : "Human Resources", "Salary" : 60000 }
{ "_id" : "Development Team", "Salary" : 80000 }
> db.Staff.aggregate({ $group: {_id: "$dept", Salary: {$first:
"$salary"}}}); { "_id" : "Maintanence", "Salary" : 25000 }
{ "_id" : "Administration", "Salary" : 60000 }
{ "_id" : "Human Resources", "Salary" : 50000 }
{ "_id" : "Development Team", "Salary" : 70000 }
> db.Staff.aggregate({ $group: {_id: "$dept", Salary: {$last:
"$salary"}}}); { "_id" : "Maintanence", "Salary" : 30000 }
{ "_id" : "Administration", "Salary" : 50000 }
{ "_id" : "Human Resources", "Salary" : 60000 }
{ "_id" : "Development Team", "Salary" : 25000 }
>

6. Demonstrate the updateOne, UpdateMany, and replaceOne

operations with suitable examples

```
> db.Staff.updateOne({_id: "001"}, { $set:{salary : 50000}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Staff.find({_id:"001"});
{ "_id" : "001", "name" : "Sarthak", "salary" : 50000, "age" : 37, "designation" :
"Representative", "increment" : "Yes", "performance" : "Good", "dept" : "Administration" }
> db.Staff.find({_id:"001"}).pretty();
{
        "_id" : "001",
        "name" : "Sarthak",
        "salary" : 50000,
        "age" : 37,
        "designation" : "Representative",
        "increment" : "Yes",
        "performance" : "Good",
        "dept" : "Administration"
}
>
```

```
> db.Staff.find({dept:"Development Team"}).pretty();
{
        "_id" : "054",
        "name" : "Colin",
        "dept" : "Development Team",
        "salary" : 70000,
        "age" : 35,
        "designation" : "Executive",
        "increment" : "No",
        "performance" : "Good"
}
{
        "_id" : "114",
        "name" : "Terrence",
        "dept" : "Development Team",
        "salary" : 80000,
        "age" : 45,
        "designation" : "CEO",
        "increment" : "No",
        "performance" : "Not Satisfactory"
```

```
}
{
        "_id" : "174",
        "name" : "Parker",
        "dept" : "Development Team",
        "salary" : 25000,
        "age" : 25,
        "designation" : "Assistant",
        "increment" : "No",
        "performance" : "Average"
}
> db.Staff.updateMany({dept: "Development Team"}, { $set:{increment : "Yes"}});
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 } >
db.Staff.find({dept:"Development Team"}).pretty();
{
        "_id" : "054",
        "name" : "Colin",
        "dept" : "Development Team",
        "salary" : 70000,
        "age" : 35,
        "designation" : "Executive",
        "increment" : "Yes",
        "performance" : "Good"
}
{
        "_id" : "114",
        "name" : "Terrence",
        "dept" : "Development Team",
        "salary" : 80000,
        "age" : 45,
        "designation" : "CEO",
        "increment" : "Yes",
        "performance" : "Not Satisfactory"
}
{
        "_id" : "174",
        "name" : "Parker",
        "dept" : "Development Team",
        "salary" : 25000,
        "age" : 25,
        "designation" : "Assistant",
        "increment" : "Yes",
        "performance" : "Average"
}
>
```

```
> db.Staff.replaceOne({_id: "001"}, {salary : 45000}); {
"acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Staff.find({_id:"001"}).pretty();
{ "_id" : "001", "salary" : 45000 }
>
```