DBMS Assignment B3

1. Return Designation with Total Salary is Above 100000

```
>db.empqw.aggregate([{$group:{_id:"$Designation",total:{$sum:"$Salary"}}},{$match:{total:{$gt
:100000}}}])
{ "_id" : "Developer", "total" : 110000 }
{ "_id" : "Designer", "total" : 115000 }
```

2. Find Employee with Total Salary for Each City with Designation="DBA"

```
> db.empqw.aggregate([{$group:{_id:"$Designation", "Total_Salary":{$sum:"$Salary"}}}]);
{ "_id" : "Developer", "Total_Salary" : 110000 }
{ "_id" : "Designer", "Total_Salary" : 115000 }
{ "_id" : "Programmer", "Total_Salary" : 100000 }
{ "_id" : "Tester", "Total_Salary" : 45000 }
```

3. Find Total Salary of Employee with Designation="DBA" for Each Company

```
>db.empqw.aggregate([{$group:{_id:"$Designation",_id:"$Company_Name","Total_Salary":{$su
m:"$Salary"}}}]);
{ "_id" : "TCS", "Total_Salary" : 95000 }
{ "_id" : "Infosys", "Total_Salary" : 45000 }
{ "_id" : "GB lab", "Total_Salary" : 20000 }
{ "_id" : "capgemini", "Total_Salary" : 60000 }
{ "_id" : "IBM", "Total_Salary" : 35000 }
{ "_id" : "Nvidia", "Total_Salary" : 50000 }
{ "_id" : "VM ware", "Total_Salary" : 65000 }
```

4. Returns names and _id in upper case and in alphabetical order.

```
>
db.empqw.aggregate([{$project:{Fname:{$toUpper:"$Name.Fname"},Lname:{$toUpper:"$Name
.Lname"}}}]).pretty()
{
        "_id" : ObjectId("5fe11ede0b4d2650a1422ce9"),
        "Fname" : "SHREYAS",
        "Lname" : "CHAUDHARI"
}
{
        "_id" : ObjectId("5fe11ede0b4d2650a1422cea"),
        "Fname" : "DHRUVIL",
        "Lname" : "SHAH"
}
{
        "_id" : ObjectId("5fe11ede0b4d2650a1422ceb"),
        "Fname" : "GAURAV",
        "Lname" : "VERMA"
```

```
}
{
	"_id" : ObjectId("5fe11ede0b4d2650a1422cec"),
	"Fname" : "SUDESH",
	"Lname" : "PAWAR"
}
{
	"_id" : ObjectId("5fe11edf0b4d2650a1422ced"),
	"Fname" : "RAM",
	"Lname" : "PATEL"
}
{
	"_id" : ObjectId("5fe11edf0b4d2650a1422cee"),
	"Fname" : "POOJA",
	"Lname" : "PATEL"
}
{
	"_id" : ObjectId("5fe11edf0b4d2650a1422cef"),
	"Fname" : "VIKAS",
	"Lname" : "GUPTA"
}
{
	"_id" : ObjectId("5fe11edf0b4d2650a1422cf0"),
	"Fname" : "GITA",
	"Lname" : "RAO"
}
```


5. Count all records from collection

```
> db.empqw.find().count()
8
```

6. For each unique Designation, find avg Salary and output is sorted by AvgSal

```
>db.empqw.aggregate([{$group:{_id:"$Designation",AvgAmount:{$avg:"$Salary"}}},{$sort:{AvgAmount:1}}]).pretty()
{ "_id" : "Tester", "AvgAmount" : 22500 }
{ "_id" : "Programmer", "AvgAmount" : 50000 }
{ "_id" : "Developer", "AvgAmount" : 55000 }
{ "_id" : "Designer", "AvgAmount" : 57500 }
```

7. Return separates value in the Expertise array where Name of Employee="Swapnil"

```
> db.empqw.find({"Name.Fname":"Gaurav"},{Expertise:1,_id:0}).pretty()
{ "Expertise" : [ "Mysql", "R language", "UI/UX" ] }
```

8. Return separates value in the Expertise array and return sum of each element of array
>db.empqw.aggregate([{$match:{Expertise:{$not:{$size:0}}}},{$unwind:"$Expertise"},{$group:{_id:"$Expertise",count:{$sum:1}}}])
{ "_id" : "YACC", "count" : 1 }
{ "_id" : "Cloud", "count" : 1 }
{ "_id" : "Java", "count" : 1 }
{ "_id" : "mongoDB", "count" : 4 }
{ "_id" : "Cpp", "count" : 2 }
{ "_id" : "python", "count" : 1 }
{ "_id" : "MySQL", "count" : 1 }
{ "_id" : "Sqlite", "count" : 1 }
{ "_id" : "Mysql", "count" : 3 }
{ "_id" : "R language", "count" : 2 }
{ "_id" : "DSA", "count" : 1 }
{ "_id" : "Cassandra", "count" : 1 }
{ "_id" : "JavaScript", "count" : 2 }
{ "_id" : "Html", "count" : 1 }
{ "_id" : "scala", "count" : 1 }
{ "_id" : "UI/UX", "count" : 1 }

9. Return Array for Designation whose address is "Pune"
> db.empqw.distinct("Designation",{"Address.city":"pune"})
["Developer"]

10. Return Max and Min Salary for each company.
> db.empqw.aggregate([{$group:{_id:"$Company_Name",minSalary:{$min:"$Salary"}}}]).pretty()
{ "_id" : "TCS", "minSalary" : 25000 }
{ "_id" : "GB lab", "minSalary" : 20000 }
{ "_id" : "Infosys", "minSalary" : 45000 }
{ "_id" : "capgemini", "minSalary" : 60000 }
{ "_id" : "IBM", "minSalary" : 35000 }
{ "_id" : "VM ware", "minSalary" : 65000 }
{ "_id" : "Nvidia", "minSalary" : 50000 }

B. Use Employee database created in Assignment 01 and perform following indexing operation

1. To Create Single Field Indexes on Designation

> db.Employee.createIndex({Designation:1})

{

"createdCollectionAutomatically" : true,

"numIndexesBefore" : 1,

```
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

2. To Create Compound Indexes on Name: 1, Age: -1

```
> db.Employee.createIndex({Name:1,Age:-1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 2,
    "numIndexesAfter" : 3,
    "ok" : 1
}
```

3. To Create Multikey Indexes on Expertise array

```
> db.Employee.createIndex({Expertise:1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 3,
    "numIndexesAfter" : 4,
    "ok" : 1
}
```

4. Return a List of All Indexes on Collection

```
> db.Employee.getIndexes()
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_"
    },
```

```
        {
            "v" : 2,
            "key" : {
                "Designation" : 1
            },
            "name" : "Designation_1"
        },
        {
            "v" : 2,
            "key" : {
                "Name" : 1,
                "Age" : -1
            },
            "name" : "Name_1_Age_-1"
        },
        {
            "v" : 2,
            "key" : {
                "Expertise" : 1
            },
            "name" : "Expertise_1"
        }
    ]
```

5. Rebuild Indexes

```
    > db.Employee.reIndex()
    {
        "nIndexesWas" : 4,
        "nIndexes" : 4,
        "indexes" : [
```

```
{
	"v" : 2,
	"key" : {
		"_id" : 1
	},
	"name" : "_id_"
},
{
	"v" : 2,
	"key" : {
		"Designation" : 1
	},
	"name" : "Designation_1"
},
{
	"v" : 2,
	"key" : {
		"Name" : 1,
		"Age" : -1
	},
	"name" : "Name_1_Age_-1"
},
{
	"v" : 2,
	"key" : {
		"Expertise" : 1
	},
	"name" : "Expertise_1"
}
```

```
        ],
        "ok" : 1
}
```

6. Drop Index on Remove Specific Index

```
> db.Employee.dropIndex({Salary:1})
{ "nIndexesWas" : 5, "ok" : 1 }
```

7. Remove All Indexes except for the _id index from a collection

```
> db.Employee.dropIndexes()
{
        "nIndexesWas" : 4,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
```

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student: Dhruvil Shah	**Class: TE Comp 1**
Semester/Year: 5th/2020	**Roll No: 047**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part B-03**

GROUP: B ASSIGNMENT NO: 03

AIM: Implement aggregation and indexing with suitable example using MongoDB.

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To develop basic Database administration skill.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Front End: Java/PHP/Python
- Backend: MongoDB

IMPLEMENTATION:

A. Use Employee database created in Assignment 01 and perform following aggregation operation
   1. Return Designation with Total Salary is Above 200000
   2. Find Employee with Total Salary for Each City with Designation="DBA"
   3. Find Total Salary of Employee with Designation="DBA" for Each Company
   4. Returns names and _id in upper case and in alphabetical order.
   5. Count all records from collection
   6. For each unique Designation, find avg Salary and output is sorted by AvgSal
   7. Return separates value in the Expertise array where Name of Employee="Swapnil"
   8. Return separates value in the Expertise array and return sum of each element of array
   9. Return Array for Designation whose address is "Pune"
   10. Return Max and Min Salary for each company.

B. Use Employee database created in Assignment 01 and perform following indexing operation

1. To Create Single Field Indexes on Designation
2. To Create Compound Indexes on Name: 1, Age: -1
3. To Create Multikey Indexes on Expertise array
4. Return a List of All Indexes on Collection
5. Rebuild Indexes
6. Drop Index on Remove Specific Index
7. Remove All Indexes except for the _id index from a collection

CONCLUSION:

QUESTIONS:

1. What is MongoDB Aggregation? Explain different type of aggregation methods.
2. Enlist different Pipeline Operators, Expression Operators, and Comparison Operators.
3. Describe SQL to aggregation Mapping Chart.
4. Explain Indexing Methods in the mongo Shell.
5. What is different option for indexing?
6. What is use of Drop Duplicates option in Indexing?
7. Write method to return a list of all indexes on a collection and databases.
8. Explain different Single Purpose Aggregation Operations.

Q1 What is Mongo DB Aggregation ? Explain different types of aggregation methods.

Ans Aggregation operations process data records and return computed results. Aggregation operations group values from multiple documents together and can perform a variety of operations on the grouped data to return a single result.

Types of aggregation methods :

1] Aggregation Pipeline:

MongoDB aggregation framework is modeled on the concept of data processing pipelines

2] Map Reduce:

MongoDB also provides map-reduce operations to perform aggregation. Map reduces uses custom javascript functions to perform the map and reduce operations as well the optional finalize operation.

3] Single Purpose Aggregation Operations

db.collection.estimatedDocumentCount() , db.collection.count() db.collection.distinct() these operations aggregate documents from a single collection.

Q2 Enlist different pipeline operators, expression operators and comparison operators.

Ans Pipeline Operators :

1] $ project
2] $ match
3] $ limit

4] $ sort
5] $ group

Expression Operators:
1] $ first
2] $ last
3] $ min
4] $ max
5] $ avg

Comparison Operators
1] $ eq
2] $ gt
3] $ in
4] $ gte
5] $ ne

Q3 Describe SQL to aggregation Mapping Chart

Ans The aggregation pipeline allows mongo DB to provide native aggregation capabilities that correspondo to many common data aggregation operations in SQL

SQL Terms and Concepts	Mongo DB Aggregation Operators
Where	$ match
Group By	$ group
Having	$ match
Select	$ project
Order By	$ sort

Limit	$limit
sum ()	$ sum
count ()	$sort By Count
join	$ Lookup
merge into table	$merge

Q4 Explain indexing methods in the Mongo Shell.

Name	Description
db.collection.createIndex()	Build an index on a collection
db.collection.dropIndex()	Remove a specified index on a collection
db.collection.dropIndexes()	Removes all indexes on a collection
db.collection.reIndex()	Rebuilds all existing indexes on a collection.
cursor.explain ()	Reports on the query execution plan for a cursor

Q5 What is different option for indexing?

Ans MongoDB provides a number of different index types to support specific types of data and queries.

1) Single Field Index
2) Compound Field Index
3) Test Index
4) Hashed Index
5) Multi-key Index
6) Geospatial Index

Q6 What is the use of Drop Duplicates option in indexing?

Ans The use of drop duplicates in indexing is to achieve uniqueness to your index.

Q7 Write a method to return a list of all indicies on a collection of databases.

Ans ~~db.collection~~

```
db.getCollectionNames().forEach(function(collection) {
    indexes = db[collection].getIndexes();
    print("Indexes for " + collection + ":");
    printjson(indexes);
});
```

Q8 Explain different single purpose aggregation operations

Ans Count -

MongoDB can return a count of the numbers of documents that match a query. The count command as well as count() methods provide access to count in mongo shell.

Distinct -

The distinct operations takes a number of documents that match a query and return all of the unique values for a field in their matching documents.

Group

The group operation takes a number of documents that match a query and then collects groups of documents based on the value of a field or fields.