Name: Tejas Rajesh Machkar
Roll No: 23
Class: TE2 Comp
PRN: F18112025

------------------------------------A PART--
COLLECTION SAME AS FIRST ASSIGNMENT

1)Return Designation with Total Salary is Above 200000

```
db.employee.find({salary_no:{$gt:20000}},{'des':true,'_id':false}).pretty();
{ "des" : "tester" }
{ "des" : "HR" }
{ "des" : "Tester" }
{ "des" : "Developer" }
{ "des" : "Developer" }
{ "des" : "Senior Developer" }
{ "des" : "tester" }
{ "des" : "Developer" }
{ "des" : "Senior Developer" }
```

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

```
db.employee.aggregate([{$match:{des:"tester"}},{$group:{_id:"$address.padd.city",total:{$sum:"$salary_no"}}}]);
{ "_id" : [ "pune" ], "total" : 42999 }
{ "_id" : [ "Dream City" ], "total" : 51000 }
```

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

```
db.employee.aggregate([{$match:{des:"tester"}},{$group:{_id:"$company_name",total:{$sum:"$salary_no"}}}]);
{ "_id" : "serum", "total" : 11999 }
{ "_id" : "INFOSYS", "total" : 31000 }
{ "_id" : "Infosys", "total" : 51000 }
```

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

```
db.employee.aggregate([{$project: {fname: {$toUpper: "$name.fname"}}},{$sort:{fname:1}}]);
{ "_id" : ObjectId("5fdddf42fd9f4759b65a8fb2"), "fname" : "ANKITA" }
{ "_id" : ObjectId("5fdddf6afd9f4759b65a8fb5"), "fname" : "ATULYA" }
{ "_id" : ObjectId("5fdddf0ffd9f4759b65a8faf"), "fname" : "HARSHIKA" }
{ "_id" : ObjectId("5fdde002fd9f4759b65a8fb6"), "fname" : "KAUSHIK" }
{ "_id" : ObjectId("5fddded9fd9f4759b65a8fae"), "fname" : "MACHO" }
{ "_id" : ObjectId("5fdde00efd9f4759b65a8fb7"), "fname" : "SANYUKTA" }
{ "_id" : ObjectId("5fdddf60fd9f4759b65a8fb4"), "fname" : "SAYALI" }
{ "_id" : ObjectId("5fdddf2efd9f4759b65a8fb1"), "fname" : "SHABBIR" }
{ "_id" : ObjectId("5fdddf55fd9f4759b65a8fb3"), "fname" : "SHREYA" }
{ "_id" : ObjectId("5fdddf20fd9f4759b65a8fb0"), "fname" : "SUMEDH" }
```

5)Count all records from collection

```
 db.employee.count();
10
> db.employee.aggregate([{$group:{_id:null,count:{$sum:1}}}]);
{ "_id" : null, "count" : 10 }
```

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

```
 db.employee.aggregate([{$group:{_id:{$toUpper: "$des"},avgsal:{$avg:"$salary_no"}}},{$sort:{avgsal:1}}]);
```

```
{ "_id" : "TESTER", "avgsal" : 28749.75 }
{ "_id" : "SENIOR DEVELOPER", "avgsal" : 35500 }
{ "_id" : "DEVELOPER", "avgsal" : 94000 }
{ "_id" : "HR", "avgsal" : 310000 }
```

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

```
db.employee.find({"name.fname":"Sumedh"},{expertise:true}).pretty();
{ "_id" : ObjectId("5fdddf20fd9f4759b65a8fb0"), "expertise" : "web dev" }
```

8)Return separates value in the Expertise array and return sum of each element of array

```
db.employee.find({},{expertise:true}).pretty();
{ "_id" : ObjectId("5fdded9fd9f4759b65a8fae"), "expertise" : "web dev" }
{ "_id" : ObjectId("5fdddf0ffd9f4759b65a8faf"), "expertise" :
"Recruitment" }
{ "_id" : ObjectId("5fdddf20fd9f4759b65a8fb0"), "expertise" : "web dev" }
{
        "_id" : ObjectId("5fdddf2efd9f4759b65a8fb1"),
        "expertise" : "managing ppl"
}
{
        "_id" : ObjectId("5fdddf42fd9f4759b65a8fb2"),
        "expertise" : "managing ppl"
}
{ "_id" : ObjectId("5fdddf55fd9f4759b65a8fb3"), "expertise" :
"completion" }
{ "_id" : ObjectId("5fdddf60fd9f4759b65a8fb4"), "expertise" : "Talking" }
{ "_id" : ObjectId("5fdddf6afd9f4759b65a8fb5"), "expertise" : "web dev" }
{ "_id" : ObjectId("5fdde002fd9f4759b65a8fb6"), "expertise" : "android
dev" }
{ "_id" : ObjectId("5fdde00efd9f4759b65a8fb7"), "expertise" : "app dev" }
```

9)Return Array for Designation whose address is "Pune"

```
 db.employee.find({"address.padd.city":"pune"},{des:true}).pretty();
{ "_id" : ObjectId("5fdded9fd9f4759b65a8fae"), "des" : "tester" }
{ "_id" : ObjectId("5fdde00efd9f4759b65a8fb7"), "des" : "tester" }
```

10)Return Max and Min Salary for each company

```
db.employee.aggregate([{$group:{_id:{$toUpper:"$company_name"},max:{$max:
"$salary_no"},min:{$min:"$salary_no"}}}]);
{ "_id" : "", "max" : 51000, "min" : 51000 }
{ "_id" : "TCS", "max" : 210000, "min" : 21000 }
{ "_id" : "SERUM", "max" : 11999, "min" : 11999 }
{ "_id" : "AMAZON", "max" : 21000, "min" : 21000 }
{ "_id" : "INFOSYS", "max" : 310000, "min" : 21000 }
```

-----------------------------------B PART--

1)To Create Single Field Indexes on Designation

```
 db.employee.createIndex({des:1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

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

```
db.employee.createIndex({age:-1,name:1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

3)To Create Multikey Indexes on Expertise array

```
db.employee.createIndex({address:1});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 4,
        "numIndexesAfter" : 5,
        "ok" : 1
}
```

4). Return a List of All Indexes on Collection

```
> db.employee.getIndexes();
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "des" : 1
                },
                "name" : "des_1"
        },
        {
                "v" : 2,
                "key" : {
                        "age" : -1,
                        "name" : 1
                },
                "name" : "age_-1_name_1"
        },
        {
                "v" : 2,
                "key" : {
                        "name" : 1
                },
                "name" : "name_1"
        },
        {
                "v" : 2,
                "key" : {
                        "address" : 1
                },
                "name" : "address_1"
        }
]
```

5)Rebuild Indexes

```
 db.employee.reIndex();
{
        "nIndexesWas" : 5,
        "nIndexes" : 5,
        "indexes" : [
                {
                        "v" : 2,
                        "key" : {
                                "_id" : 1
                        },
                        "name" : "_id_"
                },
                {
                        "v" : 2,
                        "key" : {
                                "des" : 1
                        },
                        "name" : "des_1"
                },
                {
                        "v" : 2,
                        "key" : {
                                "age" : -1,
                                "name" : 1
                        },
                        "name" : "age_-1_name_1"
                },
                {
                        "v" : 2,
                        "key" : {
                                "name" : 1
                        },
                        "name" : "name_1"
                },
                {
                        "v" : 2,
                        "key" : {
                                "address" : 1
                        },
                        "name" : "address_1"
                }
        ],
        "ok" : 1
}
```

6)Drop Index on Remove Specific Index

```
 db.employee.dropIndex("address_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
}
```

Name: Tejas. Rajesh. Machkar
RollNo: 28
Class: TE2 COMP
PRN: F18112025

DBMSL- GroupB
Assignment-3

- Questions:

Q1) What's MongoDB aggregation? Explain different types of aggregation method.

A.1) Aggregation operation process data records and return computed results. Aggregation operations group values from multiple documents together and can perform a variety of ~~program~~ operations on the group data to return a single result.

- MongoDB provides 3 way to perform aggregation:

i) Aggregation pipeline: MongoDB's aggregation framework is modeled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transforms documents into an aggregated result.

ii) Map reduce: Map reduce operations have 2-phase - a map stage that processes each document and exits one or more object for ~~an~~ each input document; & a reduced phase that contains the output of map aggregate.

iii) Single purpose aggregation: MongoDB also provides db.collection.count() and db.collection.distinct(). All these aggregations are performed on docs of some collection.

Q2) Enlist difference pipeline operations, expression operation and comparison operators.

A2) Pipeline operators:

- $project: reshapes a document stream.
- $ match: filters a document stream.

- $redact: Restricts the content of a document on a ~~per-fitted~~ per-field level.
- $limit: Restricts number of documents.
- $skip: skip number of documents from pipeline.
- $unwind: Takes an array of documents and returns them as a stream.
- $group: Group documents to calculate different aggregate values.
- $sort: take all documents and return a stream of sorted ones.
- $geoNear: Returns an ordered stream of documents based on a proximity to geospatial point.

2) Expression operators:

• $ addtoSet	• $ max
• $ first	• $ avg
• $ last	• $ push
• $ min	• $ sum

3) Compaoeison operators:

• $eg	• $in	• $ne
• $get	• $it	• $min
• $gte	• $lte	

Q3) Describe SQL to aggregation mapping chart.

A.3)

SQL terms, functions ; concepts	MongoDB aggregation operators
WHERE	$ match
GROUP BY	$ group
HAVING	$ project
SELECT	$ sort
ORDER BY	$ limit
LIMIT	$ sum
SUM ()	$ surt By count
COUNT()	$ lookup
JOIN	

Q4) Explain indexing methods in MongoShell.

A.4) 1) db.collection.createIndex() : Builds an index on collection.

2) db.collection.dropIndex() : Removes all indexes.

3) db.collection.getIndexes(): Returns an array of docs that describes the existing indices on a collection.

4) db.collection.dropIndexes(): Removes all indexes.

5) db.collection.reIndex(): Rebuilds all existing indices.

6) db.collection.totalIndexSize(): Reports the total size used by the indic on a collection.

7) cursor.explain() : Reports on query execution plan for a cursor.

8) cursor.hint(): Forces MongoDB to use a specific index for a query.

9) cursor.max(): Specifies an exclusive upper index bound for a cursor for use with cursor.hint().

10) cursor.min(): Specifies an inclusive lower index bound for a cursor.

Q5) What are different options for indexing?

A-5) 1) Single field index.

2) Compound index.

3) Multikey index.

4) Geospatial index.

5) Test index

6) Hashed index.

Q6) What is the use of drop duplicates options in indexing?

A.6) The use of drop duplicates in indexing is to achieve uniqueness to your index.

Q7) Write a method to return a list of all indices, on a collection and db

A.7) Collection: db.collection.getIndexes();

Database: db.getCollectionNames().forEach (function collection)

indexes = db.[collection].getIndexes();

```
print ("Indexes for" + collection + ":");
print json (indexes); });
```

Q8) Explain different single puvepose aggregation opeerations.

A.8 1) Count: Retuoens a count of documents that match a query. The count command as well as the count() and cursor.count() methods provide access to count in the mongo shell.

2) Distinct: Take a number of documents that match a query and returns all of the uniaue values for a field in their matching documents. The distinct command and db.collections.distinct() method provide this opecration in the mongo shell.

3) Group: Takes a namuer of docs that match a query and then collects group of docs based on the value of a field(s). It returns an array of docs with computed results for each group of docs To access the group functionality via group command or db.collection.group() method in mongo shell.