Assignment No: 10

- Date of completion :
  Date of submission :

- Title :- MongoDB Aggregation and Indexing.

- Problem Statement:- Implement aggregation and indexing with suitable example using MongoDB.

- Learning Objectives:-
  1. To understand indexing in MongoDB.
  2. To understand aggregation concept in MongoDB.

- Learning Outcomes:-
  1. To implement indexing and aggregation in MongoDB.

- Software and Hardware Requriements:- PC with configuration as latest version of 64 bit OS, open source Fedora 2GHz, 8GB RAM, 500 GB HDD, 15" color mointor, keyboard, mouse, MongoDB.

- Theory :-
  - Aggregation - Aggregation operations process data records & 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. In SQL count(*) & with group by is an equivalent of Mongo Aggregation.

Aggregate() method -
For aggregation on MongoDB, you should use aggregate() method.

Syntax:

```
db.collection_name.aggregate(AGGREGATE_OPERATION);
```

e.g.

```
db.mycol.aggregate([{$group : {_id : "$by_user",
            num_tutorial : {$sum : 1}}}]);
```

- ~~Aggre~~
- Pipeline Concept - In UNIX command, shell pipeline means the possibly to execute an operation on some input & use the output as input for next command & so on. MongoDB also supports same concept in aggregation framework. There is a set of possible stages & each one of those is taken as a set of documents or an input & produces resulting set of documents. This can be used for next stage & so on.

Following are the possible stages in aggregation framework:

1) $projeet - used to select some specific fields from a collection.
2) $match - This is a filtering operation & thus this can reduce the amount of documents that are given as input for result stage.

3) $group - This does the actual aggregation.

4) $sort - sorts the documents

5) $skip - with this, it is possible to skip forward in the list of documents for a given amount of documents.

6) $limit - This limits the amounts of documents to work at, by the given number, starting from the current position.

7) $unwind - This is used to unwind documents that are using arrays. When using an array, the data is kind of ~~pre-wind~~ pre-joined & this operation will be undone with this to have individual documents again.

- Indexing:- Indexes support the efficient resolution of queries. without indexes, MongoDB must scan every document of a collection to selection those which match the query statement. This scan is highly inefficient & require MongoDB to process a large volume of data.

  Indexes are special structures that stores a small portion of the data set in ~~att~~ an easy-to-~~struture~~ traverse form. The index stores the value of a specified field or set of fields, ordered by the value of the field as specified in the index.

1. Default-id Index - MongoDB creates a unique index on the id field during the creation of a collection & we cannot drop this index. The _id index prevents clients from inserting two documents with the same value for _id field.

2. Single field - It is used to create user-defined ascending / descending indexes on a single fields of a document.

```
db. collection_name. createIndex({fieldname:-1})
```

We can view the index names using:

```
db. collection_name. getIndex()
```

3. Compound Index -

```
db. students. createIndex({roll:1, name:1})
```

4. Unique Index -

```
db. students. createIndex({roll:1}, {unique:true})
```

* Test cases :-

Description	Expected o/p & Actual o/p	Result.
db.purchase_orders.find({product:"tooth_brush"}).count();	3	Pass
db.purchase_orders.distinct("product");	['guitar','milk','pizza','tooth_brush']	Pass

db.purchase_orders.aggregate([{$match:{product:{$in:['toothbrush','pizza']}}},{$group:{_id:$product,earnings:{$sum:"$total"}}}]);	{"_id":"pizza","earnings":13.25}
	{"_id":"toothbrush","earning":14.25}
		Pass.

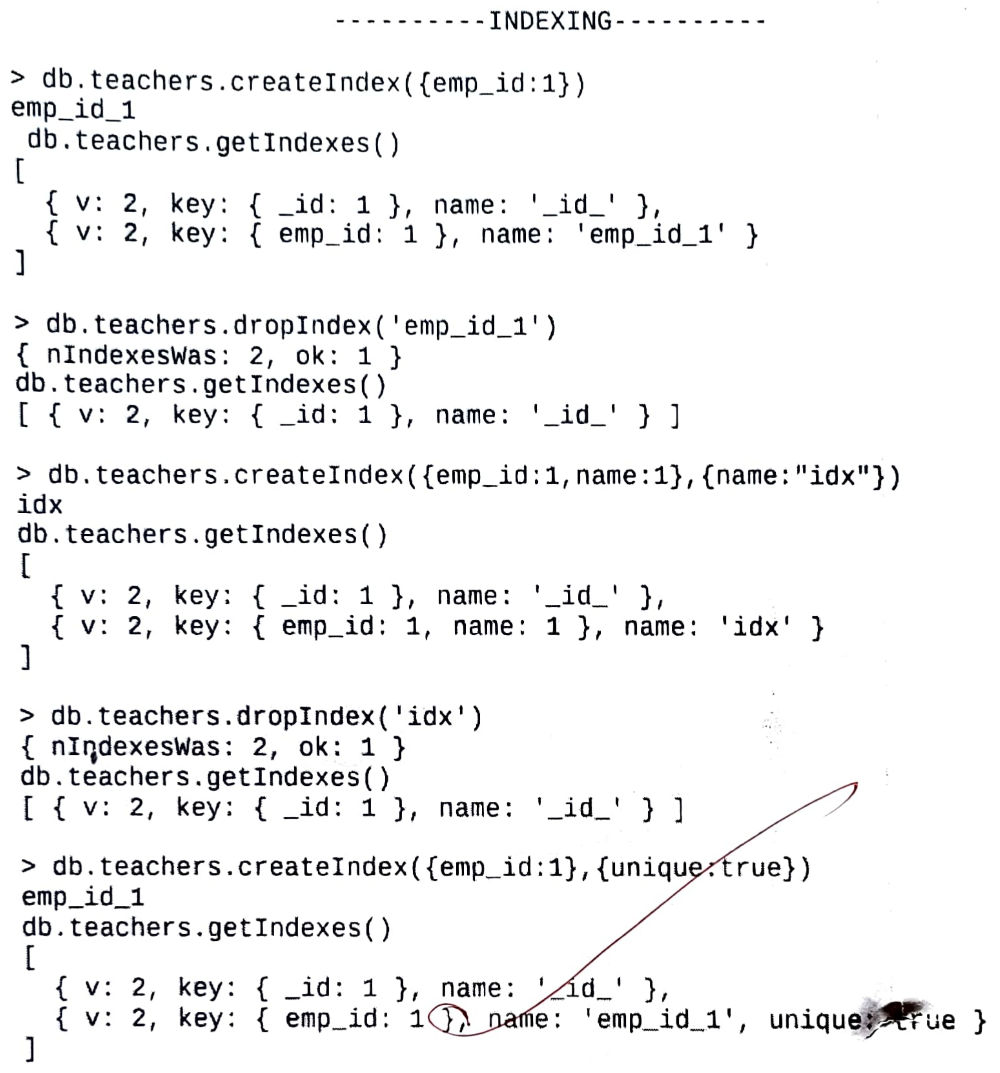
* Conclusion :- Hence, we have implemented the concept of aggregation and indexing in MongoDB.

```
******Aggregate********
> db.purchase_orders.insertMany([
...         {product: "toothbrush", total: 4.75, customer: "Mike"},
...         {product: "guitar", total: 199.99, customer: "Tom"},
...         {product: "milk", total: 11.33, customer: "Mike"},
...         {product: "pizza", total: 8.50, customer: "Karen"},
...         {product: "toothbrush", total: 4.75, customer: "Karen"},
...         {product: "pizza", total: 4.75, customer: "Dave"},
...         {product: "toothbrush", total: 4.75, customer: "Mike"},
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6360a1aa87c8e546d5bcf064"),
                ObjectId("6360a1aa87c8e546d5bcf065"),
                ObjectId("6360a1aa87c8e546d5bcf066"),
                ObjectId("6360a1aa87c8e546d5bcf067"),
                ObjectId("6360a1aa87c8e546d5bcf068"),
                ObjectId("6360a1aa87c8e546d5bcf069"),
                ObjectId("6360a1aa87c8e546d5bcf06a")
        ]
}
> db.purchase_orders.aggregate(
... [
...     {$match:{product:{$in:['toothbrush','pizza']}}},
...     {$group:{_id:'$product', earnings:{$sum:"$total"}}}
... ]
... )
{ "_id" : "pizza", "earnings" : 13.25 }
{ "_id" : "toothbrush", "earnings" : 14.25 }
```

```
> db.purchase_orders.find({product:"toothbrush"}).count()
3
> db.purchase_orders.distinct("product")
[ "guitar", "milk", "pizza", "toothbrush" ]
> db.purchase_orders.distinct("product")
[ "guitar", "milk", "pizza", "toothbrush" ]
> db.purchase_orders.aggregate(
... [
...     {
...         $group : {
...             _id:"$customer",
...             moneyspent:{$sum: "$total"}
...         }
...     }
... ]
... )
{ "_id" : "Tom", "moneyspent" : 199.99 }
{ "_id" : "Karen", "moneyspent" : 13.25 }
{ "_id" : "Dave", "moneyspent" : 4.75 }
{ "_id" : "Mike", "moneyspent" : 20.83 }
> db.purchase_orders.aggregate(
...     [
...         {$project: {_id: 0,"product":1}},
...         {$sort: {total: 1}},
...         {$limit: 1}
...     ]
... )
{ "product" : "toothbrush" }
> db.purchase_orders.aggregate(
... [
...     {$group: {_id: "$product", totalMoneySpent: {$sum: "$total"}}},
...     {$sort: {"totalMoneySpent": 1}}
... ]
... )
{ "_id" : "milk", "totalMoneySpent" : 11.33 }
{ "_id" : "pizza", "totalMoneySpent" : 13.25 }
{ "_id" : "toothbrush", "totalMoneySpent" : 14.25 }
{ "_id" : "guitar", "totalMoneySpent" : 199.99 }
> db.purchase_orders.aggregate(
... [
...     {$match:{product: {$in: ["pizza", "toothbrush"]}}},
...     {$group:{_id:"$customer", totalMoneySpentOnTnP: {$sum:"$total"}}}
... ]
... )
{ "_id" : "Karen", "totalMoneySpentOnTnP" : 13.25 }
{ "_id" : "Dave", "totalMoneySpentOnTnP" : 4.75 }
{ "_id" : "Mike", "totalMoneySpentOnTnP" : 9.5 }
> db.purchase_orders.aggregate(
... [
...     {$match:{product:"toothbrush"}},
...     {$group:{_id:"$customer", countOfToothbrush:{$count:{}}}},
...     {$sort:{countOfToothbrush: -1}},
...     {$limit: 1}
... ]
... )
[ { _id: 'Mike', countOfToothbrush: 2 } ]
```

*******Indexing********

```
> db.teachers.save(
...
...   {
...     _id: ObjectId("634d7be6429b791350d5bb42"),
...     name: 'Anushka Wable',
...     qualificaton: 'PhD',
...     deptno: 1,
...     deptname: 'Comp',
...     experience: 11,
...     designation: 'Professor',
...     salary: { basic: 200000, ta: 20000, da: 40000, hra: 50000 },
...     date_of_joining: '20-10-2011',
...     area_of_expertise: 'Cyber Security',
...     empId:101
...   }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.teachers.save(
...   {
...     _id: ObjectId("634d7be6429b791350d5bb43"),
...     name: 'Prajwal Kakade',
...     qualificaton: 'PhD in Data Science',
...     deptno: 2,
...     deptname: 'IT',
...     experience: 9,
...     designation: 'Professor',
...     salary: { basic: 100000 },
...     date_of_joining: '01-11-2013',
...     area_of_expertise: 'Data science',
...     empId:102
...   }
... )
```



----------INDEXING----------

```
> db.teachers.createIndex({emp_id:1})
emp_id_1
 db.teachers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { emp_id: 1 }, name: 'emp_id_1' }
]

> db.teachers.dropIndex('emp_id_1')
{ nIndexesWas: 2, ok: 1 }
db.teachers.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]

> db.teachers.createIndex({emp_id:1,name:1},{name:"idx"})
idx
db.teachers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { emp_id: 1, name: 1 }, name: 'idx' }
]

> db.teachers.dropIndex('idx')
{ nIndexesWas: 2, ok: 1 }
db.teachers.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]

> db.teachers.createIndex({emp_id:1},{unique:true})
emp_id_1
db.teachers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { emp_id: 1 }, name: 'emp_id_1', unique: true }
]
```