	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**		
Name: Siddhesh Dilip Khairnar			
Class: SY	Division: B		Roll No: 272028
Semester: III		Academic Year: 2022-2023	
Subject Name & Code: Database Management System: ADUA21204			
Title of Assignment: Implement aggregation and indexing with suitable example using Mongo DB.			

Assignment No.- 08

PAGE NO. :

DATE : / /

Assignment no. 8

Name : Siddhesh Dilip Khairnar

PRN no : 22110398 Roll no : 272028

1. Brief about Aggreation operation in Mongo DB with syntax of aggreate() method.

Ans Aggregation operation process multiple document and return computed result. you can use aggregation operation to:

- Group values from multiple document together.
- Perform operation on the grouped data to return a single result.
- Analyze data change over time.

To perform aggregation operation, you can use :

- Aggregation pipelines, which are the prefered method for performing aggregation.
- single purpose aggragation method, which are simple but lack the capabilities of an aggregation pipeline.

2. Brief about Index in Mongo DB with syntax to create an index, syntax of drop index.

Ans Indexes support the efficient execution of queries in MongoDB. without indexes, Mongo DB must perform a collection scan that scan every document in a collection, to select those document that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limited the number of document it must inspect.

- syntax to create an index :

To create an index in the Mongo shell, use db. collection. createIndex()

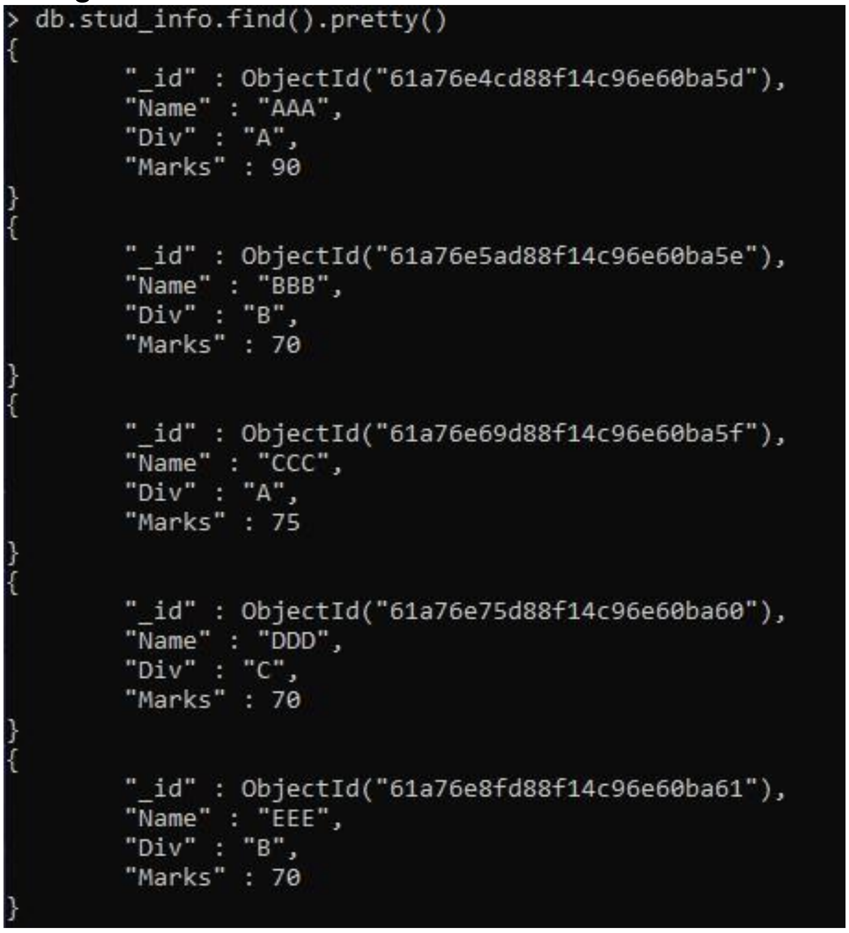
→ db. collection. create Index (<key and index type specification>, <option>)

- Syntax to drop index :

→ db. collection_Name. dropIndex (index : <document / string>)

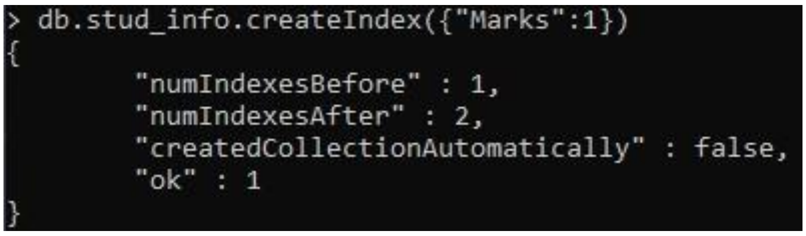
Implementation:

Using the below collection



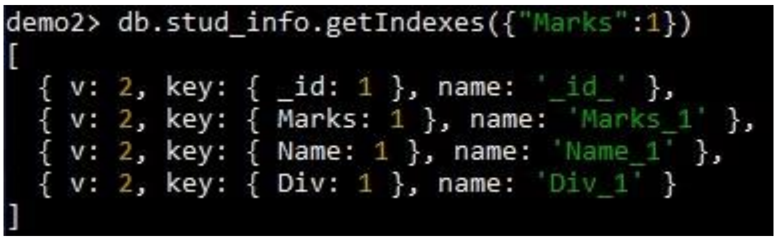
```
> db.stud_info.find().pretty()
{
        "_id" : ObjectId("61a76e4cd88f14c96e60ba5d"),
        "Name" : "AAA",
        "Div" : "A",
        "Marks" : 90
}
{
        "_id" : ObjectId("61a76e5ad88f14c96e60ba5e"),
        "Name" : "BBB",
        "Div" : "B",
        "Marks" : 70
}
{
        "_id" : ObjectId("61a76e69d88f14c96e60ba5f"),
        "Name" : "CCC",
        "Div" : "A",
        "Marks" : 75
}
{
        "_id" : ObjectId("61a76e75d88f14c96e60ba60"),
        "Name" : "DDD",
        "Div" : "C",
        "Marks" : 70
}
{
        "_id" : ObjectId("61a76e8fd88f14c96e60ba61"),
        "Name" : "EEE",
        "Div" : "B",
        "Marks" : 70
}
```

1. Creates index



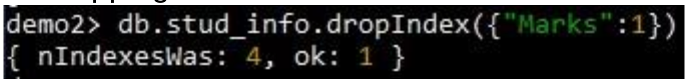
```
> db.stud_info.createIndex({"Marks":1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

2. Displays the index.



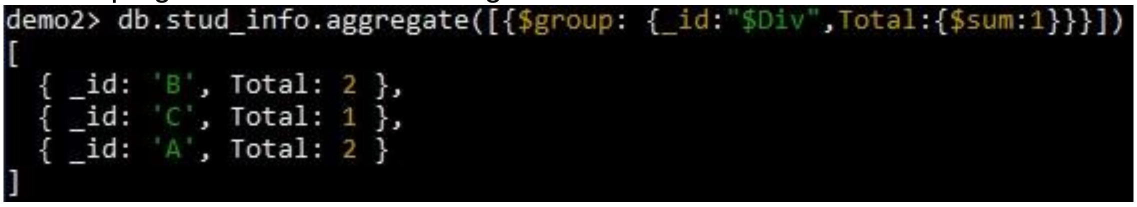
```
demo2> db.stud_info.getIndexes({"Marks":1})
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { Marks: 1 }, name: 'Marks_1' },
  { v: 2, key: { Name: 1 }, name: 'Name_1' },
  { v: 2, key: { Div: 1 }, name: 'Div_1' }
]
```

3. Dropping the index



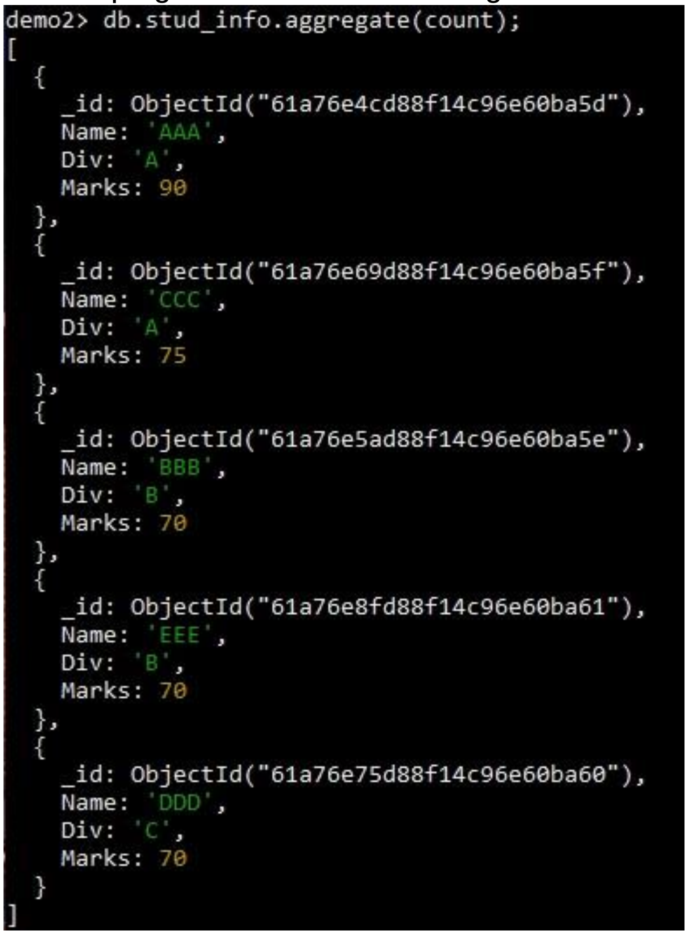
```
demo2> db.stud_info.dropIndex({"Marks":1})
{ nIndexesWas: 4, ok: 1 }
```

4. Grouping the students according to their division



```
demo2> db.stud_info.aggregate([{$group: {_id:"$Div",Total:{$sum:1}}}])
[
  { _id: 'B', Total: 2 },
  { _id: 'C', Total: 1 },
  { _id: 'A', Total: 2 }
]
```

5. Grouping the students according to their divisions.



```
demo2> db.stud_info.aggregate(count);
[
  {
    _id: ObjectId("61a76e4cd88f14c96e60ba5d"),
    Name: 'AAA',
    Div: 'A',
    Marks: 90
  },
  {
    _id: ObjectId("61a76e69d88f14c96e60ba5f"),
    Name: 'CCC',
    Div: 'A',
    Marks: 75
  },
  {
    _id: ObjectId("61a76e5ad88f14c96e60ba5e"),
    Name: 'BBB',
    Div: 'B',
    Marks: 70
  },
  {
    _id: ObjectId("61a76e8fd88f14c96e60ba61"),
    Name: 'EEE',
    Div: 'B',
    Marks: 70
  },
  {
    _id: ObjectId("61a76e75d88f14c96e60ba60"),
    Name: 'DDD',
    Div: 'C',
    Marks: 70
  }
]
```