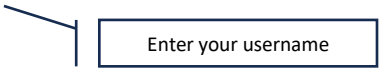
DBMS LAB ASSIGNMENT

Q10.Practicing on Mongo DB basics: Download any dataset from the given repositories and import the dataset into Mongo DB database and practice on insert commands, selection, projection and selection with conditions and aggregate functions.

DATASET: https://data.world/alexandra/music-composers

TASK 1: Connecting to mongodb server using cmd.

Command : mongosh "mongodb+srv://cluster0.ezjbfwh.mongodb.net/" --apiVersion 1 --username praketpatitiwarimath20



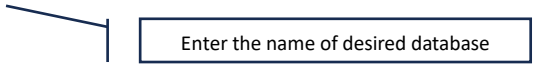

```
Microsoft Windows [Version 10.0.22631.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tiwar>mongosh "mongodb+srv://cluster0.ezjbfwh.mongodb.net/" --apiVersion 1 --username praketpatitiwarimath20
Enter password: *********
Current Mongosh Log ID: 65a02dcf0895578a4afd61b9
Connecting to:          mongodb+srv://<credentials>@cluster0.ezjbfwh.mongodb.net/?appName=mongosh+2.0.0
Using MongoDB:          6.0.12 (API Version 1)
Using Mongosh:          2.0.0
mongosh 2.1.1 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/
```

TASK 2: Switching to the required database.

Command : use Lab.



```
Atlas atlas-sjgjhs-shard-0 [primary] test> use Lab
switched to db Lab
```

TASK 3: Loading JSON file, containing data , into the database

Command line1: const data = EJSON.parse(fs.readFileSync("C:\\Users\\Tiwar\\OneDrive\\Desktop\\composers.json"))

Path to file

Command line2: db.composers.insertMany(data)

Collection name

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> const data = EJSON.parse(fs.readFileSync("C:\\Users\\Tiwar\\OneDrive\\Desktop\\composers.json"))

Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.insertMany(data)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("65a02dfe0895578a4afd61ba"),
    '1': ObjectId("65a02dfe0895578a4afd61bb"),
    '2': ObjectId("65a02dfe0895578a4afd61bc"),
    '3': ObjectId("65a02dfe0895578a4afd61bd"),
    '4': ObjectId("65a02dfe0895578a4afd61be"),
    '5': ObjectId("65a02dfe0895578a4afd61bf"),
    '6': ObjectId("65a02dfe0895578a4afd61c0"),
    '7': ObjectId("65a02dfe0895578a4afd61c1"),
    '8': ObjectId("65a02dfe0895578a4afd61c2"),
    '9': ObjectId("65a02dfe0895578a4afd61c3"),
    '10': ObjectId("65a02dfe0895578a4afd61c4"),
    '11': ObjectId("65a02dfe0895578a4afd61c5"),
    '12': ObjectId("65a02dfe0895578a4afd61c6"),
    '13': ObjectId("65a02dfe0895578a4afd61c7"),
    '14': ObjectId("65a02dfe0895578a4afd61c8"),
    '15': ObjectId("65a02dfe0895578a4afd61c9"),
    '16': ObjectId("65a02dfe0895578a4afd61ca"),
    '17': ObjectId("65a02dfe0895578a4afd61cb"),
    '18': ObjectId("65a02dfe0895578a4afd61cc"),
    '19': ObjectId("65a02dfe0895578a4afd61cd"),
    '20': ObjectId("65a02dfe0895578a4afd61ce"),
    '21': ObjectId("65a02dfe0895578a4afd61cf"),
    '22': ObjectId("65a02dfe0895578a4afd61d0"),
    '23': ObjectId("65a02dfe0895578a4afd61d1"),
    '24': ObjectId("65a02dfe0895578a4afd61d2"),
    '25': ObjectId("65a02dfe0895578a4afd61d3"),
    '26': ObjectId("65a02dfe0895578a4afd61d4"),
    '27': ObjectId("65a02dfe0895578a4afd61d5"),
    '28': ObjectId("65a02dfe0895578a4afd61d6"),
    '29': ObjectId("65a02dfe0895578a4afd61d7"),
    '30': ObjectId("65a02dfe0895578a4afd61d8"),
    '31': ObjectId("65a02dfe0895578a4afd61d9"),
    '32': ObjectId("65a02dfe0895578a4afd61da"),
    '33': ObjectId("65a02dfe0895578a4afd61db"),
```

```
    '4654': ObjectId("65a02dfe0895578a4afd73e8"),
    '4655': ObjectId("65a02dfe0895578a4afd73e9"),
    '4656': ObjectId("65a02dfe0895578a4afd73ea"),
    '4657': ObjectId("65a02dfe0895578a4afd73eb")
  }
}
```

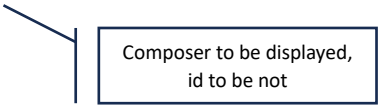
TASK 4: Displaying the dataset

Command : db.composers.find()

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find()
[
  {
    _id: ObjectId("65a02dfe0895578a4afd61ba"),
    Composer: 'Mary Anne à Beckett',
    DOB: '1817'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd61bb"),
    Composer: 'Michel van der Aa',
    DOB: '1970'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd61bc"),
    Composer: 'Thorvald Aagaard',
    DOB: '1877'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd61bd"),
    Composer: 'Truid Aagesen',
    DOB: '1593'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd61be"),
    Composer: 'Heikki Aaltoila',
    DOB: '1905'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd61bf"),
    Composer: 'Juhan Aavik',
    DOB: '1884'
  },
  {
```

TASK 5: Projecting specific column.

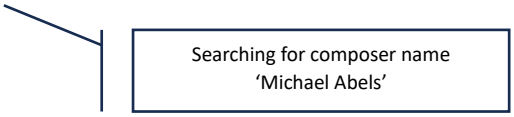
Command : db.composers.find({},{Composer:1,_id:0})




```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({},{Composer:1,_
id:0})
[
  { Composer: 'Mary Anne à Beckett' },
  { Composer: 'Michel van der Aa' },
  { Composer: 'Thorvald Aagaard' },
  { Composer: 'Truid Aagesen' },
  { Composer: 'Heikki Aaltoila' },
  { Composer: 'Juhan Aavik' },
  { Composer: "Evaristo Felice Dall'Abaco" },
  { Composer: 'Joseph Abaco' },
  { Composer: 'Antonio Maria Abbatini' },
  { Composer: 'Gamal Abdel-Rahim' },
  { Composer: 'Mohamed Abdelwahab Abdelfattah' },
  { Composer: 'Keiko Abe' },
  { Composer: 'Rosalina Abejo' },
  { Composer: 'Carl Friedrich Abel' },
  { Composer: 'Clamor Heinrich Abel' },
  { Composer: 'Ludwig Abel' },
  { Composer: 'Mark Abel' },
  { Composer: 'Michael Abels' },
  { Composer: 'Nicanor Abelardo' },
  { Composer: 'David Abell' }
]
Type "it" for more
```

TASK 6: Searching for specific data entry with single constraint.

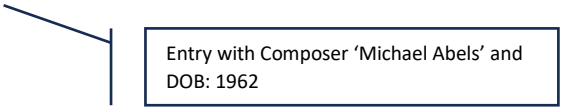
Command : db.composers.find({Composer:'Michael Abels'})




```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({Composer:'Micha
el Abels'})
[
  {
    _id: ObjectId("65a02dfe0895578a4afd61cb"),
    Composer: 'Michael Abels',
    DOB: '1962'
  }
]
```

TASK 7: Searching for specific data entry with 2 constraints.

Command : db.composers.find({Composer:'Michael Abels',DOB:"1962"})




```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({Composer:'Micha
el Abels',DOB:"1962"})
[
  {
    _id: ObjectId("65a02dfe0895578a4afd61cb"),
    Composer: 'Michael Abels',
    DOB: '1962'
  }
]
```

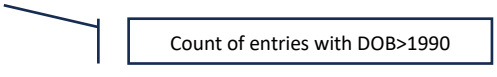
TASK 8: Getting number of entries in dataset.

Command : db.composers.find().count()

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find().count()
4658
```

TASK 9: Getting number of specific occurences in dataset.

Command : db.composers.find({DOB: {$gt:'1990'}}).count()




```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({DOB: {$gt:'1990
'}}).count()
6
```

TASK 10: Inserting 1 Entry in Dataset.

Command : db.composers.insertOne({Composer:"Sachin",DOB:"2001"})

Inserting data with Composer: Sachin and DOB:2001

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.insertOne({Composer:"
Sachin",DOB:"2001"})
{
  acknowledged: true,
  insertedId: ObjectId("65a044e70895578a4afd73ec")
}
```

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({Composer:"Sachi
n"})
[
  {
    _id: ObjectId("65a044e70895578a4afd73ec"),
    Composer: 'Sachin',
    DOB: '2001'
  }
]
```

TASK 11: Inserting more than 1 Entry in Dataset.

Command :
db.composers.insertMany([{Composer:"Sourabh",DOB:"2002"},{Composer:"Praket",DOB:"2003"}])

Inserting 2 Entries.

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.insertMany([{Composer
:"Sourabh",DOB:"2002"},{Composer:"Praket",DOB:"2003"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("65a048090895578a4afd73ed"),
    '1': ObjectId("65a048090895578a4afd73ee")
  }
}
```

TASK 12: Selecting data with DOB > 1990.

Command : db.composers.find({ DOB: {$gt: '1990'}})

All entries with DOB > 1990. 'gt' means greater than

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({ DOB: {$gt: '1990'}})
[
  {
    _id: ObjectId("65a02dfe0895578a4afd69f3"),
    Composer: 'Jay Greenberg',
    DOB: '1991'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd6b4d"),
    Composer: 'David Earle Johnson',
    DOB: '1998'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd70a9"),
    Composer: 'Matt Savage',
    DOB: '1992'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd720c"),
    Composer: 'Tan Yan Zhang',
    DOB: '1998'
  },
  {
    _id: ObjectId("65a02dfe0895578a4afd72a5"),
    Composer: 'Jacobus Vaet',
    DOB: ' 1529"'
  },
  {
    _id: ObjectId("65a044e70895578a4afd73ec"),
    Composer: 'Sachin',
    DOB: '2001'
  },
  {
    _id: ObjectId("65a048090895578a4afd73ed"),
    Composer: 'Sourabh',
    DOB: '2002'
  },
  {
```

TASK 13: Selecting data with DOB < 1820 .

Command : db.composers.find({DOB: {$lt:'1820'}})

All entries with DOB < 1820. 'lt' means less than

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({DOB: {$lt:'1820
'}})
[
  {
    _id: ObjectId("65a3795496b6f054122dc17c"),
    Composer: 'Mary Anne à Beckett',
    DOB: '1817'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc17f"),
    Composer: 'Truid Aagesen',
    DOB: '1593'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc182"),
    Composer: "Evaristo Felice Dall'Abaco",
    DOB: '1675'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc183"),
    Composer: 'Joseph Abaco',
    DOB: '1710'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc184"),
    Composer: 'Antonio Maria Abbatini',
    DOB: '1595'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc189"),
    Composer: 'Carl Friedrich Abel',
    DOB: '1723'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc18a"),
    Composer: 'Clamor Heinrich Abel',
    DOB: '1634'
```

TASK 14: Selecting data with DOB < 1820 and displaying only first 3 matching results.

Command : db.composers.find({DOB: {$lt:'1820'}}).limit(3)

limit() command limits the number of results displayed

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({DOB: {$lt:'1820
'}}).limit(3)
[
  {
    _id: ObjectId("65a3795496b6f054122dc17c"),
    Composer: 'Mary Anne à Beckett',
    DOB: '1817'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc17f"),
    Composer: 'Truid Aagesen',
    DOB: '1593'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc182"),
    Composer: "Evaristo Felice Dall'Abaco",
    DOB: '1675'
  }
]
```

TASK 15: Selecting data with DOB <= 1820 .

Command : db.composers.find({DOB: {$lte:'1820'}})

'lte' means less than equal to

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({DOB: {$lte:'182
0'}})
[
  {
    _id: ObjectId("65a3795496b6f054122dc17c"),
    Composer: 'Mary Anne à Beckett',
    DOB: '1817'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc17f"),
    Composer: 'Truid Aagesen',
    DOB: '1593'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc182"),
    Composer: "Evaristo Felice Dall'Abaco",
    DOB: '1675'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc183"),
    Composer: 'Joseph Abaco',
    DOB: '1710'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc184"),
    Composer: 'Antonio Maria Abbatini',
    DOB: '1595'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc189"),
    Composer: 'Carl Friedrich Abel',
    DOB: '1723'
  },
  {
    _id: ObjectId("65a3795496b6f054122dc18a"),
    Composer: 'Clamor Heinrich Abel',
    DOB: '1634'
  },
```

TASK 16: Deleting 1 Entry from Dataset.

Command : db.composers.deleteOne({Composer:'Andrew of Crete'})

deleteOne() deletes only 1 entry

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.deleteOne({Composer:'
Andrew of Crete'})
{ acknowledged: true, deletedCount: 1 }
```

TASK 17: Deleting more than 1 Entry from Dataset.

Command : db.composers.deleteMany({DOB: {$gt:'1990'}})

deleteMany() deletes many entries at a time

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.deleteMany({DOB: {$gt
:'1990'}})
{ acknowledged: true, deletedCount: 6 }
```

TASK 18: Selecting all the entries with names starting with M and getting its count.

Command : db.composers.find({Composer:{$regex:"^M"}}).count()

```
Atlas atlas-sjgjhs-shard-0 [primary] Lab> db.composers.find({Composer:{$rege
x:"^M"}}).count()
310
```