Msc(Data Sci and AI)

Practical-4: INDEXING USING Mangodb

Name: Krishna Shantaram Pinvankar **Roll No. – L022**

Write-up: -

- **Indexing in mongodb**
- **Types of indexing**
- **Document document-oriented NoSQL**

1. Mongo DB indexing

a. Create index in Mongo DB

b. Finding the indexes in a collection

c. Drop indexes in a collection

d. Drop all the indexes

```
use students
db.createCollection("studentgrades")
db.studentgrades.insertMany(
 [
 {name: "Aditys", subject: "Maths", score: 92},
 {name: "Krish", subject: "Physics", score: 87},
 {name: "Krishna", subject: "Maths", score: 99, notes: "Exceptional Performance"},
 {name: "Prag", subject: "Literature", score: 78},
 {name: "Tom", subject: "History", score: 65, notes: "Adequate"}
 ]
)db
```

db.studentgrades.find({},{_id:0})

db.studentgrades.find().pretty()

db.studentgrades.createIndex({name: 1}, {name: "student name index"})

Finding indexes You can find all the available indexes in a MongoDB collection by using the getIndexes method. This will return all the indexes in a specific collection. db..getIndexes() Let's view all the indexes in the studentgrades collection using the following command:

db.studentgrades.getIndexes()

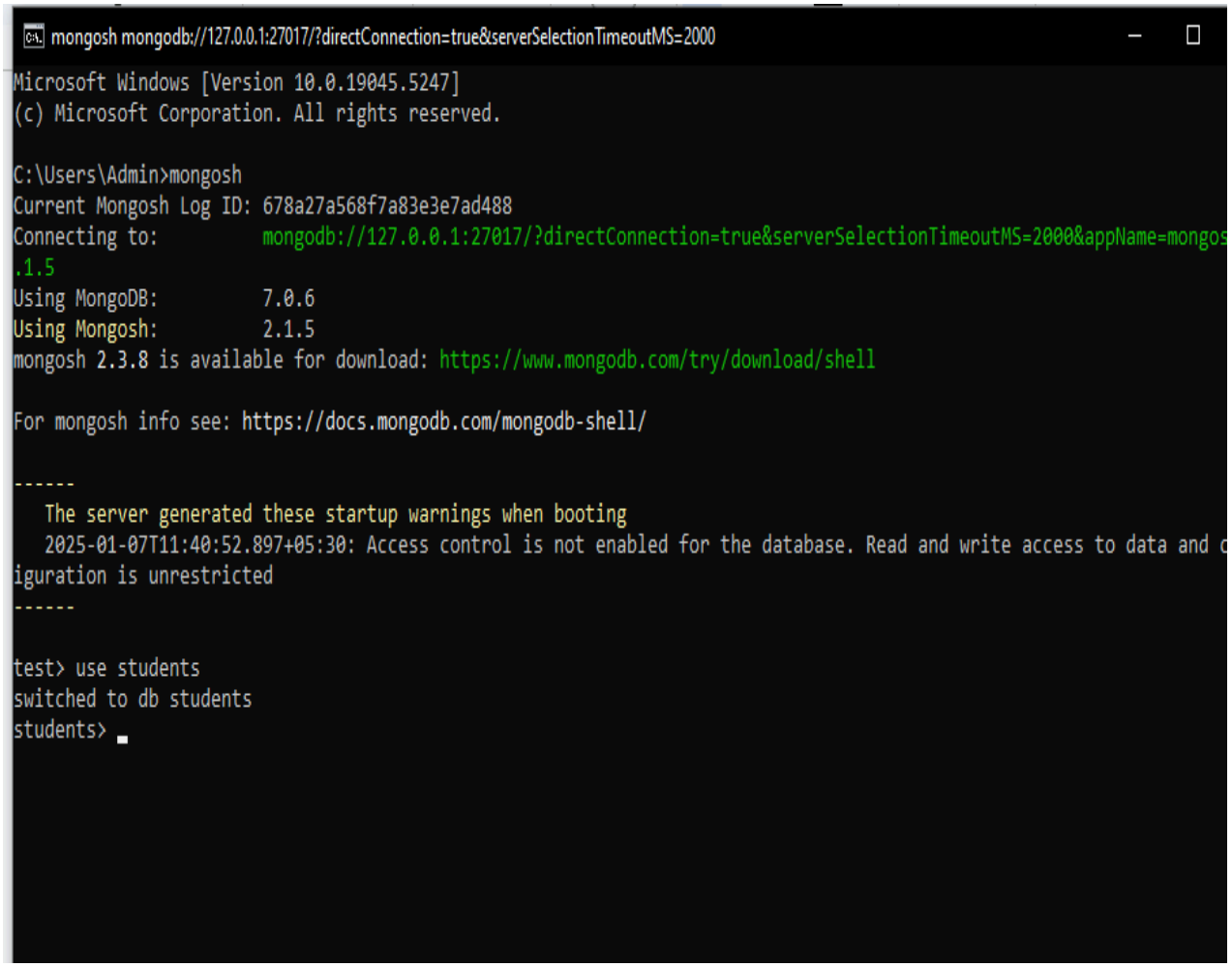
Dropping indexes To delete an index from a collection, use the dropIndex method while specifying the index name to be dropped. db..dropIndex() Let's remove the user-created index with the index name student name index, as shown below. db.studentgrades.dropIndex("student name index")

You can also use the index field value for removing an index without a defined name: db.studentgrades.dropIndex({name:1})

The dropIndexes command can also drop all the indexes excluding the default _id index. db.studentgrades.dropIndexes()

2. Create all the types of indexes (discussed in class) which will help in finding certain words in a document by using AIRPORT (dataset).

****


mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin>mongosh
Current Mongosh Log ID: 678a27a568f7a83e3e7ad488
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongos
.1.5
Using MongoDB: 7.0.6
Using Mongosh: 2.1.5
mongosh 2.3.8 is available for download: https://www.mongodb.com/try/download/shell
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
The server generated these startup warnings when booting
2025-01-07T11:40:52.897+05:30: Access control is not enabled for the database. Read and write access to data and c
iguration is unrestricted
test> use students
switched to db students
students>

Cmd

Mongosh

```
C:\Users\Admin>mongosh
Current Mongosh Log ID: 678a27a568f7a83e3e7ad488
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS
Using MongoDB:          7.0.6
Using Mongosh:          2.1.5
mongosh 2.3.8 is available for download: https://www.mongodb.com/try/download/shell

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

------
   The server generated these startup warnings when booting
   2025-01-07T11:40:52.897+05:30: Access control is not enabled for the database. Read and write
------

test> use students
switched to db students
students> db.createCollection("studentgrades")
{ ok: 1 }
students> db.studentgrades.insertMany(
...  [
...  {name: "Aditys", subject: "Maths", score: 92},
...  {name: "Krish", subject: "Physics", score: 87},
...  {name: "Krishna", subject: "Maths", score: 99, notes: "Exceptional Performance"},
...  {name: "Prag", subject: "Literature", score: 78},
...  {name: "Tom", subject: "History", score: 65, notes: "Adequate"}
...  ]
... )
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('678a299568f7a83e3e7ad489'),
    '1': ObjectId('678a299568f7a83e3e7ad48a'),
    '2': ObjectId('678a299568f7a83e3e7ad48b'),
    '3': ObjectId('678a299568f7a83e3e7ad48c'),
    '4': ObjectId('678a299568f7a83e3e7ad48d')
  }
}
```

```
students> db.studentgrades.find({},{_id:0})
[
  { name: 'Aditys', subject: 'Maths', score: 92 },
  { name: 'Krish', subject: 'Physics', score: 87 },
  {
    name: 'Krishna',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  { name: 'Prag', subject: 'Literature', score: 78 },
  { name: 'Tom', subject: 'History', score: 65, notes: 'Adequate' }
]
```

```
students> db.studentgrades.find().pretty()
[
  {
    _id: ObjectId('678a299568f7a83e3e7ad489'),
    name: 'Aditys',
    subject: 'Maths',
    score: 92
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48a'),
    name: 'Krish',
    subject: 'Physics',
    score: 87
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48b'),
    name: 'Krishna',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48c'),
    name: 'Prag',
    subject: 'Literature',
    score: 78
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48d'),
    name: 'Tom',
    subject: 'History',
    score: 65,
    notes: 'Adequate'
  }
]
```

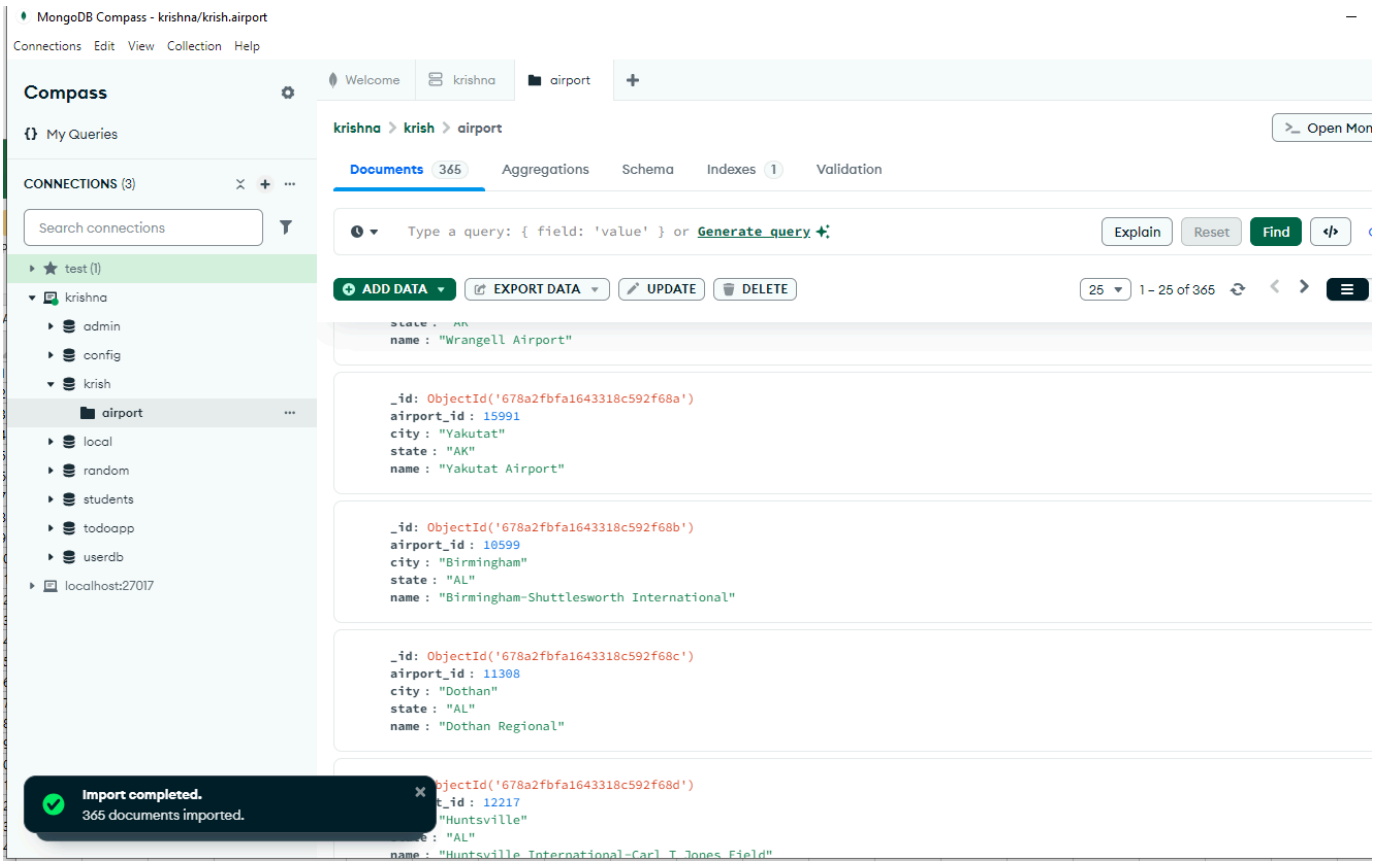
```
students> db.studentgrades.find().pretty()
[tudent name index
  {
    _id: ObjectId('678a299568f7a83e3e7ad489'),
    name: 'Aditys',
    subject: 'Maths',
    score: 92
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48a'),
    name: 'Krish',
    subject: 'Physics',
    score: 87
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48b'),
    name: 'Krishna',
    subject: 'Maths',
    score: 99,
    notes: 'Exceptional Performance'
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48c'),
    name: 'Prag',
    subject: 'Literature',
    score: 78
  },
  {
    _id: ObjectId('678a299568f7a83e3e7ad48d'),
    name: 'Tom',
    subject: 'History',
    score: 65,
    notes: 'Adequate'
  }
]
```

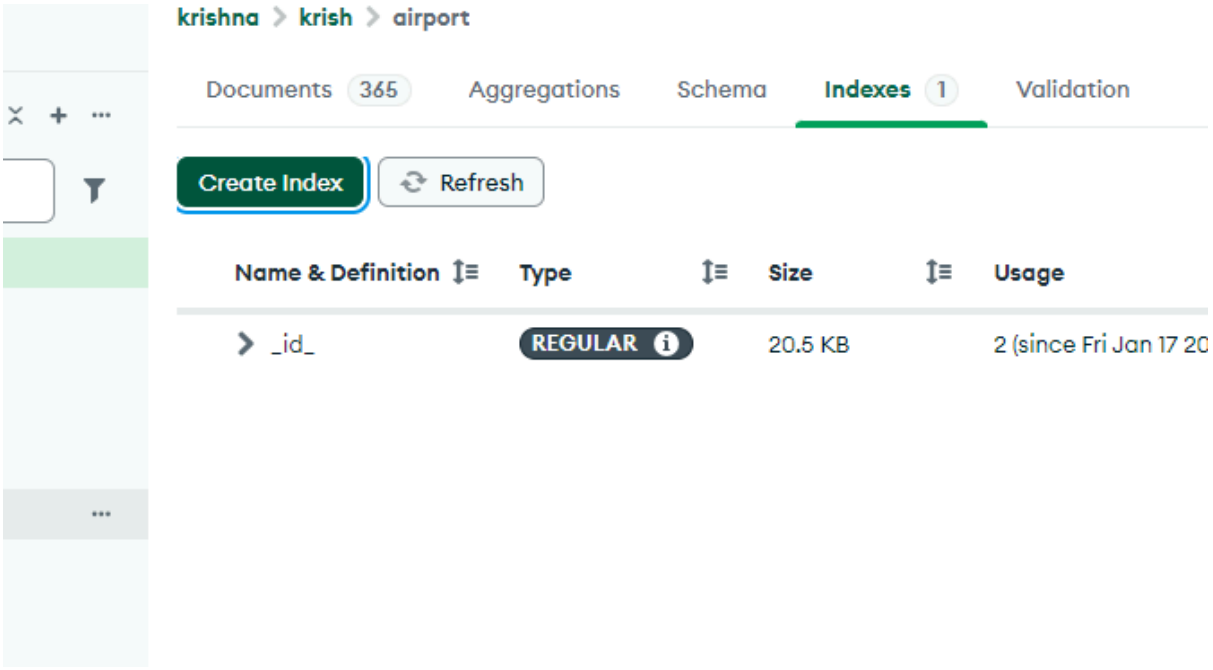
```
students> db.studentgrades.getIndexes()
[ { v: 2, key: { name: 1 }, name: 'student name index' }
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'student name index' }
]
students> db.studentgrades.createIndex( {name: 1}, {name: "student name index"} )
student name index
students> db.studentgrades.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'student name index' }
]
students>
{ nIndexesWas: 2, ok: 1 }
students> db.studentgrades.dropIndex("student name index")
MongoshInternalError[IndexNotFound]: index not found with name [student name index]
students> db.studentgrades.dropIndex({name:1})
MongoshInternalError[IndexNotFound]: can't find index with key: { name: 1 }
students> index. db.studentgrades.dropIndexes()
ReferenceError: index is not defined
students>
```

Q.2

Download the file airport data from kaggle

Open compass


MongoDB Compass - krishna/krish.airport
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (3)
Search connections
test (1)
krishna
admin
config
krish
airport
local
random
students
todoapp
userdb
localhost:27017
Welcome
krishna
airport
krishna > krish > airport
Open Mon
Documents 365
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 25 of 365
name : "Wrangell Airport"
_id: ObjectId('678a2fbfa1643318c592f68a')
airport_id : 15991
city : "Yakutat"
state : "AK"
name : "Yakutat Airport"
_id: ObjectId('678a2fbfa1643318c592f68b')
airport_id : 10599
city : "Birmingham"
state : "AL"
name : "Birmingham-Shuttlesworth International"
_id: ObjectId('678a2fbfa1643318c592f68c')
airport_id : 11308
city : "Dothan"
state : "AL"
name : "Dothan Regional"
Import completed.
365 documents imported.


krishna > krish > airport
Documents 365
Aggregations
Schema
Indexes 1
Validation
Create Index
Refresh
Name & Definition
Type
Size
Usage
_id_
REGULAR
20.5 KB
2 (since Fri Jan 17 20

Name & Definition	Type	Size	Usage	Properties	Status
_id_ _id ↑	REGULAR	20.5 KB	7 (since Fri Jan 17 2025)	UNIQUE	READY
airport_id_-1 airport_id ↓	REGULAR	20.5 KB	0 (since Fri Jan 17 2025)		READY
city_1 city ↑	REGULAR	24.6 KB	0 (since Fri Jan 17 2025)	HIDDEN	READY
name_text _fts (text) _ftsx ↑	TEXT	41.0 KB	0 (since Fri Jan 17 2025)		READY