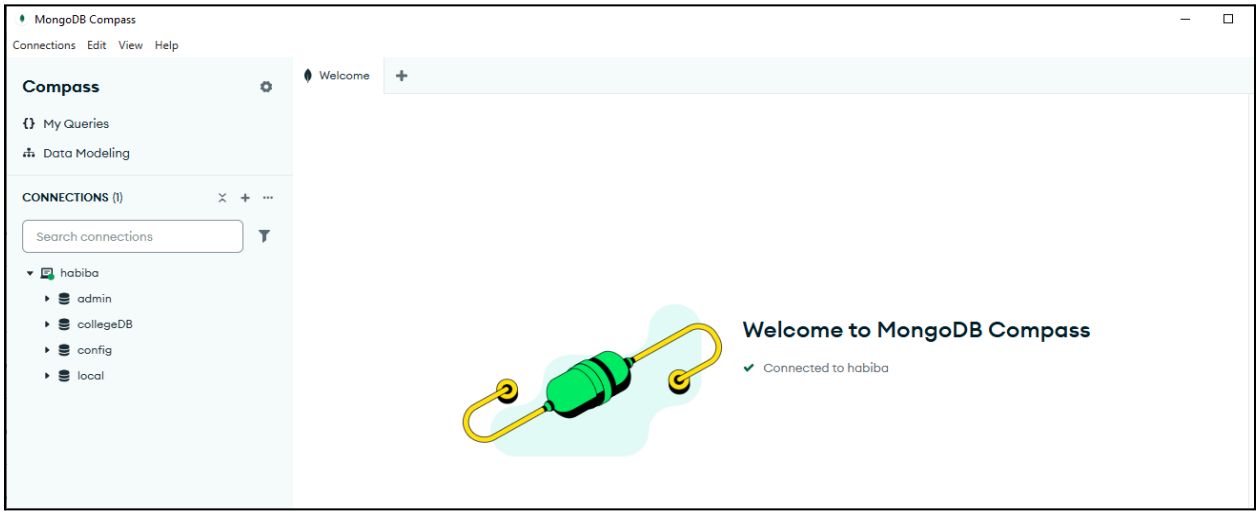
Practical No:- 14

Aim 14(a):- Write a program to demonstrate Mongo shell operations and MongoDB Compass usage.

Part A: Connection and Environment Initialization
Establish Connection: The shell connects to the local instance at mongodb://127.0.0.1:27017

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Anish>mongosh
Current Mongosh Log ID: 69637688dddbd7010b1e2620
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.10
Using MongoDB:          8.2.3
Using Mongosh:          2.5.10

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


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2026-01-11T14:56:07.113+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------
```

Verify with Compass: Open the MongoDB Compass application to view the active connection to localhost:27017



Part B: Database and Collection Setup
List Existing Databases: Use show databases to view default databases like admin, config, and local.

```
test> show databases
admin       40.00 KiB
collegeDB   72.00 KiB
config      96.00 KiB
local       40.00 KiB
test>
```

Create or Switch Database: Execute use college to enter the specific database context for this practical.

```
test> use college
switched to db college
college>
```

Manage Existing Collections: use db.student.drop() to remove old collections, which returns true upon success.

```
college> db.student.drop()
true
college>
```

Aim 14(b):- Write a program to demonstrate collections and documents in MongoDB.

Part C: Working with Documents (Insertion)

Insert Single Document: Add a record for "habiba" using insertOne with fields for roll, name, course, and age.

```
> db.students.insertOne({roll:40, name: "Habiba", course:"Computer Science", age:18})
< {
    acknowledged: true,
    insertedId: ObjectId('69637cc628e35c68291ba186')
  }
```

Insert Multiple Documents: Add "Mahek" and "Zaberiya" simultaneously using insertMany within an array [].

```
college> db.students.insertMany([
... {roll:35, name: "Mahek", course: "Computer Science",age:19},
... {roll:38, name: "Zaberiya",course: "Computer Science",age:18}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69637bf3dddbd7010b1e2622'),
    '1': ObjectId('69637bf3dddbd7010b1e2623')
  }
}
college>
```

```
> db.students.find()
< {
    _id: ObjectId('69637cc628e35c68291ba186'),
    roll: 40,
    name: 'Habiba',
    course: 'Computer Science',
    age: 18
  }
  {
    _id: ObjectId('69637dae28e35c68291ba187'),
    roll: 35,
    name: 'Mahek',
    course: 'Computer Science',
    age: 19
  }
  {
    _id: ObjectId('69637dae28e35c68291ba188'),
    roll: 38,
    name: 'Zaberiya',
    course: 'Computer Science',
    age: 18
  }
```

Part D: CRUD Operations (Read, Update, Delete)

Query (Read): Use db.students.find() to retrieve all data or a filter like { **course:** "Computer Science" } to find specific records.

```
> db.students.find({ course: "Computer Science"})
< {
    _id: ObjectId('69637cc628e35c68291ba186'),
    roll: 40,
    name: 'Habiba',
    course: 'Computer Science',
    age: 18
  }
  {
    _id: ObjectId('69637dae28e35c68291ba187'),
    roll: 35,
    name: 'Mahek',
    course: 'Computer Science',
    age: 19
  }
  {
    _id: ObjectId('69637dae28e35c68291ba188'),
    roll: 38,
    name: 'Zaberiya',
    course: 'Computer Science',
    age: 18
  }
```

Modify (Update): Use updateOne with the $set operator to change specific field values, such as updating an age.

```
> db.students.updateOne({roll:40},{$set:{age:19}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.students.find()
< {
    _id: ObjectId('69637cc628e35c68291ba186'),
    roll: 40,
    name: 'Habiba',
    course: 'Computer Science',
    age: 19
  }
  {
    _id: ObjectId('69637dae28e35c68291ba187'),
    roll: 35,
    name: 'Mahek',
    course: 'Computer Science',
    age: 19
  }
  {
    _id: ObjectId('69637dae28e35c68291ba188'),
    roll: 38,
    name: 'Zaberiya',
    course: 'Computer Science',
    age: 18
```

Remove (Delete): Use deleteOne to remove a specific record based on a field like roll or the unique _id.

```
> db.students.deleteOne({ roll: 38})
< {
    acknowledged: true,
    deletedCount: 1
  }
```

```
> db.students.find()
< {
    _id: ObjectId('69637cc628e35c68291ba186'),
    roll: 40,
    name: 'Habiba',
    course: 'Computer Science',
    age: 19
  }
  {
    _id: ObjectId('69637dae28e35c68291ba187'),
    roll: 35,
    name: 'Mahek',
    course: 'Computer Science',
    age: 19
  }
```