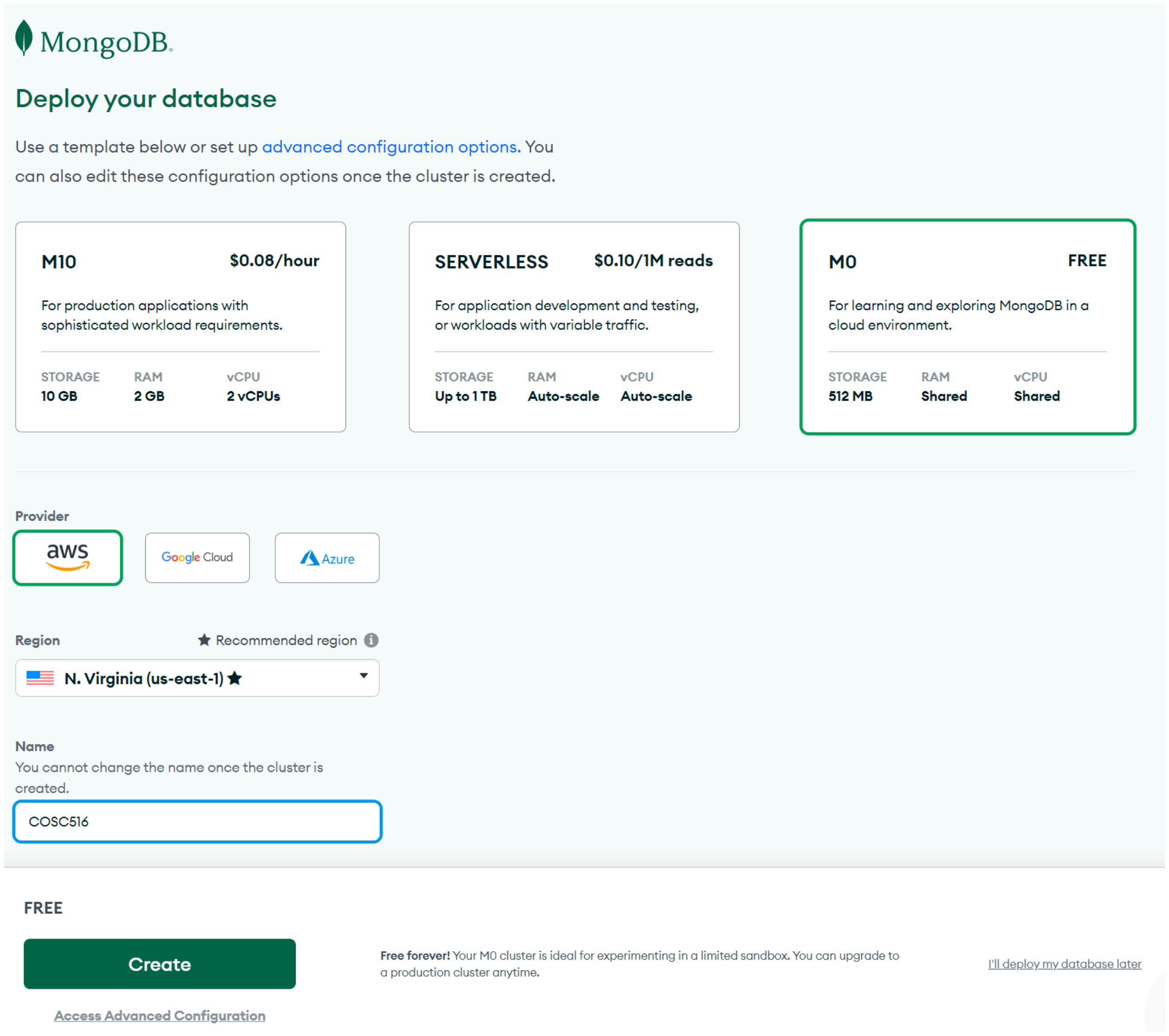
Assignment-7

Redis: 45 points

> **Submit a PDF with code listing, and screenshots showing outputs of insert(), delete(), and the queries. Screenshots should be uniquely distinguishable for each submission. Be careful of plagiarism from online sources/peers.**

MongoDB

Login to MongoDB Atlas and create a M0 cluster:



MongoDB

Deploy your database

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10 — **$0.08/hour**
For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

SERVERLESS — **$0.10/1M reads**
For application development and testing, or workloads with variable traffic.

STORAGE	RAM	vCPU
Up to 1 TB	Auto-scale	Auto-scale

M0 — **FREE**
For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Provider: aws [✓] / Google Cloud / Azure

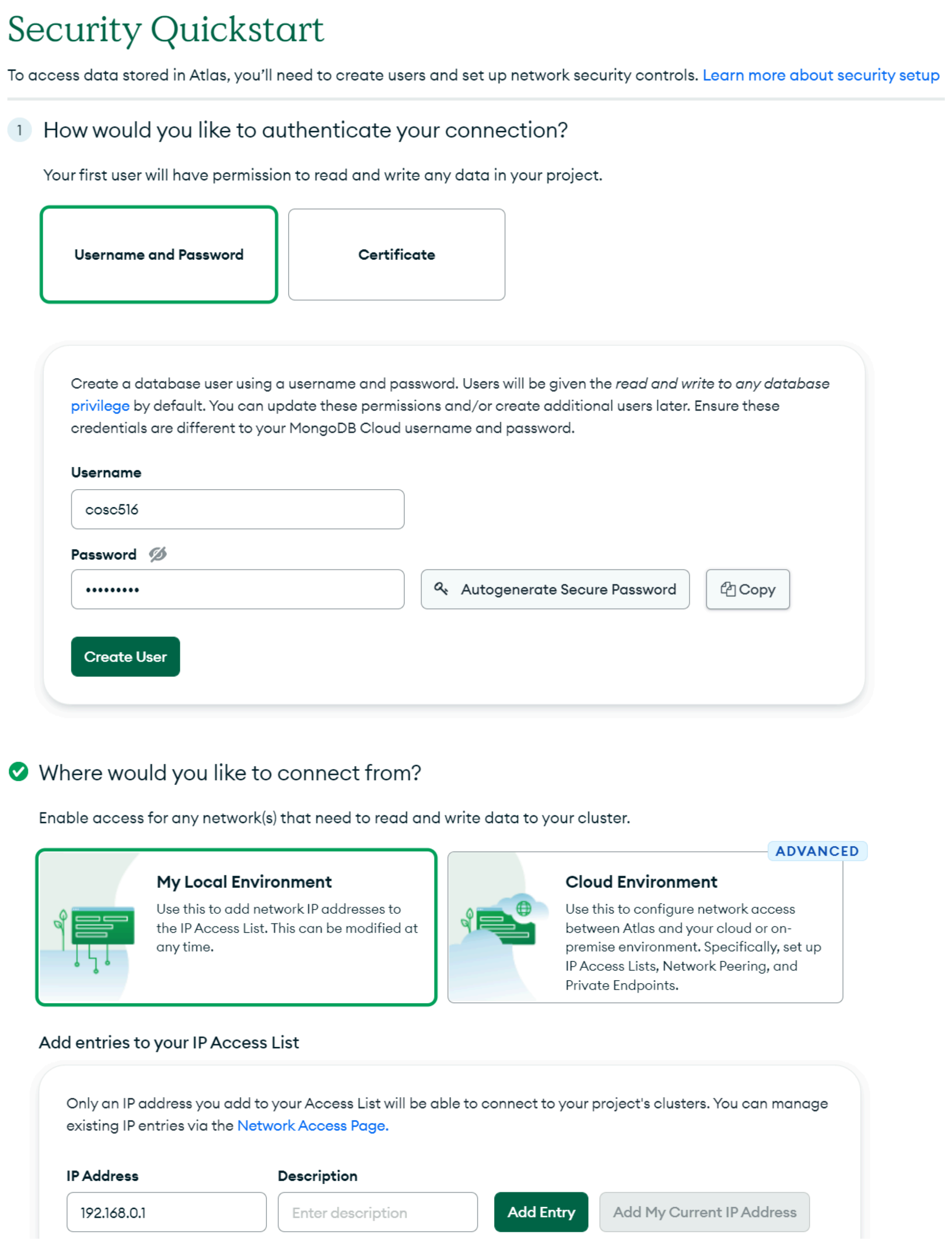
Region — ★ Recommended region
N. Virginia (us-east-1) ★

Name
You cannot change the name once the cluster is created.
COSC516

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Create a Database User account and connect wit your own IP to allow access from your local machine



Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. Learn more about security setup

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username: cosc516

Password: •••••••••

✓ Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Cloud Environment ADVANCED
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

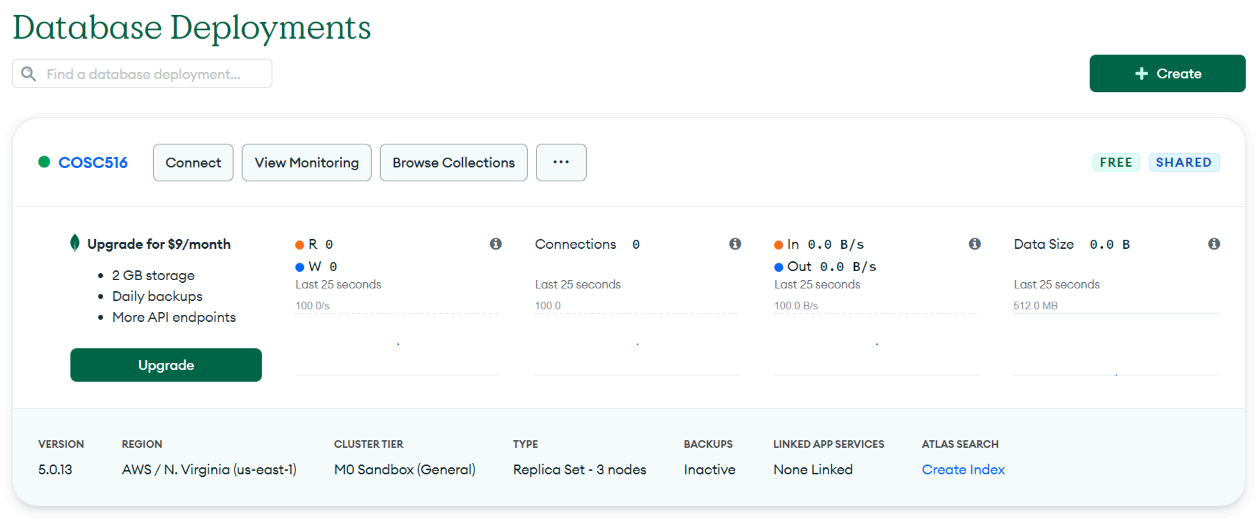
Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.

IP Address: 192.168.0.1

Description: Enter description

Get the connection information from the dashboard:



Database Deployments

Find a database deployment...

+ Create

COSC516 Connect View Monitoring Browse Collections ··· FREE SHARED

Upgrade for $9/month
- 2 GB storage
- Daily backups
- More API endpoints

Upgrade

R 0
W 0
Last 25 seconds
100.0/s

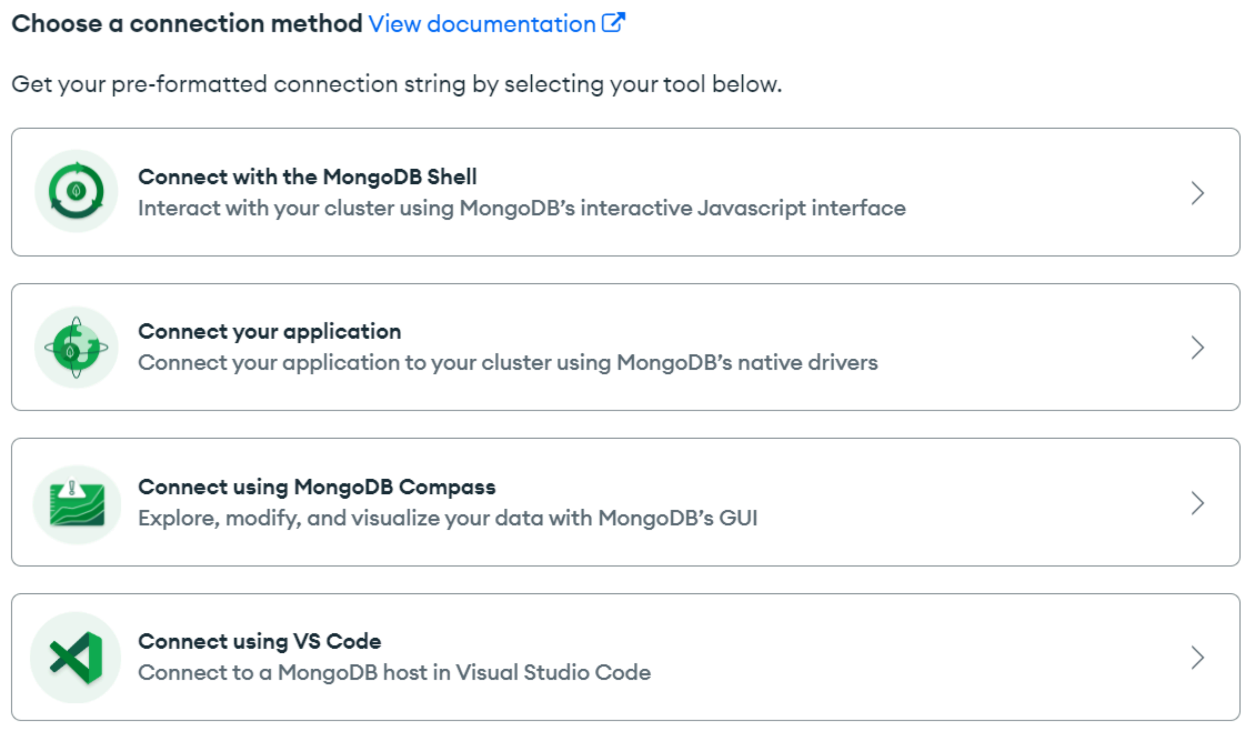
Connections 0
Last 25 seconds
100.0

In 0.0 B/s
Out 0.0 B/s
Last 25 seconds
100.0 B/s

Data Size 0.0 B
Last 25 seconds
512.0 MB

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SEARCH
5.0.13	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

Connecting to cluster. You can use MongoDB Compass application or using a query language



Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

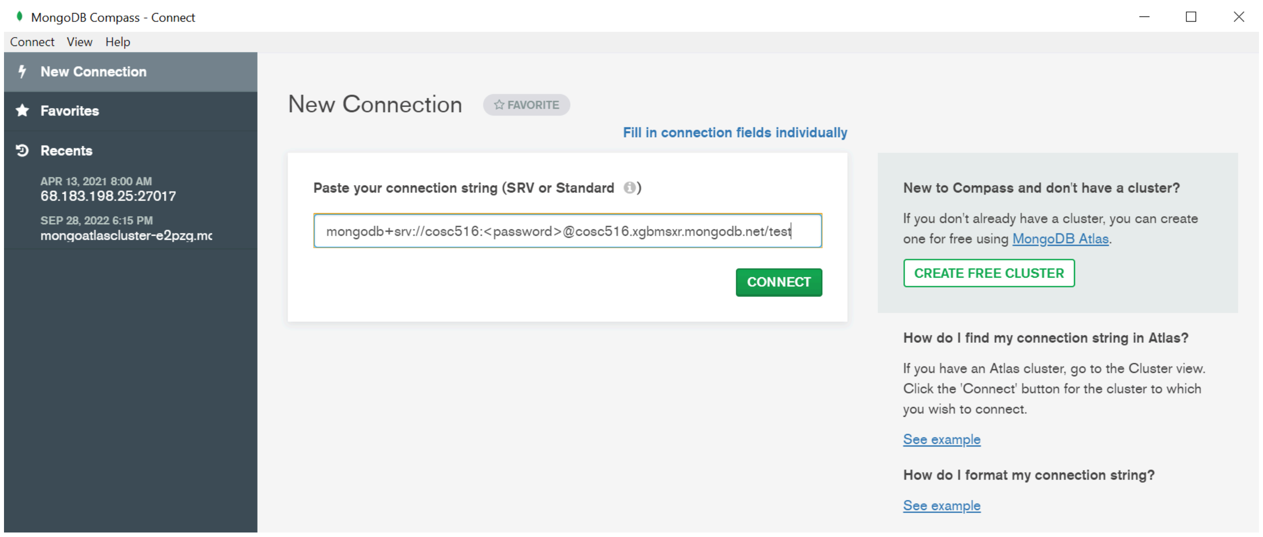
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Connect using VS Code
Connect to a MongoDB host in Visual Studio Code

Connecting using MongoDB Compass



Paste your connection string (SRV or Standard)

```
mongodb+srv://cosc516:<password>@cosc516.xgbmsxr.mongodb.net/test
```

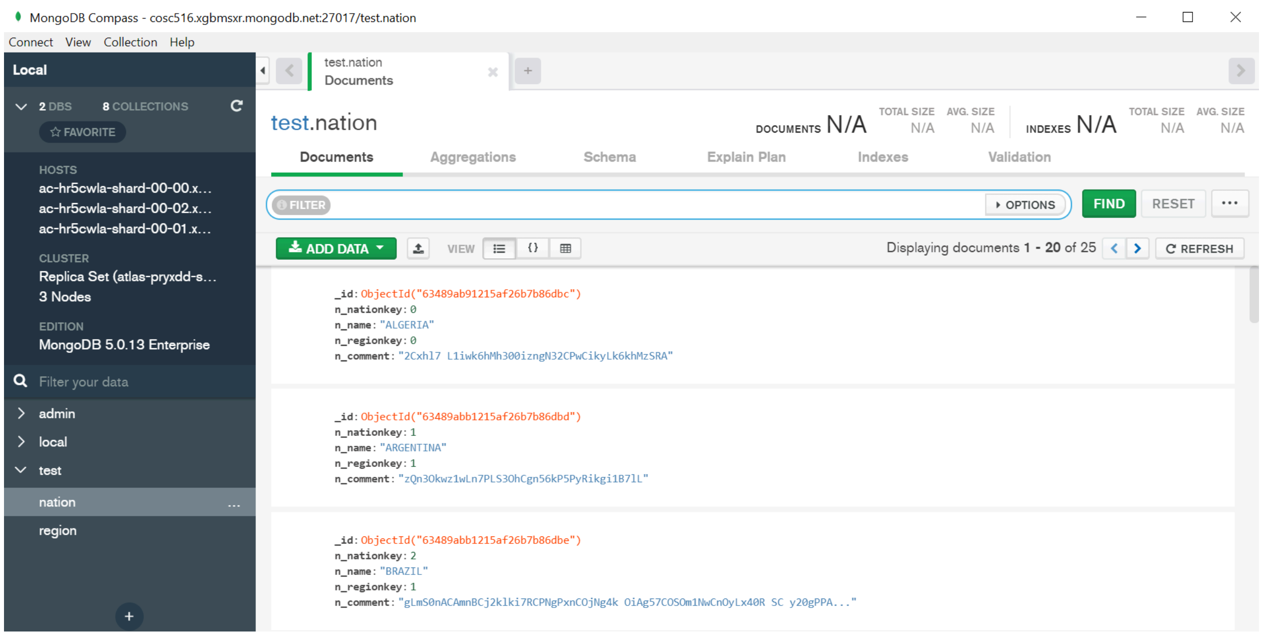
New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas.

How do I find my connection string in Atlas?

If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.

How do I format my connection string?



```
_id: ObjectId("63489ab91215af26b7b86dbc")
n_nationkey: 0
n_name: "ALGERIA"
n_regionkey: 0
n_comment: "2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6khMzSRA"

_id: ObjectId("63489abb1215af26b7b86dbd")
n_nationkey: 1
n_name: "ARGENTINA"
n_regionkey: 1
n_comment: "zQn3Okwz1wLn7PLS3OhCgn56kP5PyRikgi1B7lL"

_id: ObjectId("63489abb1215af26b7b86dbe")
n_nationkey: 2
n_name: "BRAZIL"
n_regionkey: 1
n_comment: "gLmS0nACAmnBCj2klki7RCPNgPxnCOjNg4k OiAg57COSOm1NwCnOyLx40R SC y20gPPA..."
```

A few References:

1. MongoDB Introduction
2. Creating a Collection
3. Modifying Documents
4. Querying using find()
5. Mongo Java Driver API

Now complete the tasks as follows: (See an example java program included also) [5x9=45]

1. Write the method load() to load the TPC-H customer and orders data into separate collections (like how it would be stored in a relational model). The data files are in the data folder.

2. Write the method loadNest() to load the TPC-H customer and order data into a nested collection called custorders where each document contains the customer information and all orders for that customer.

3. Write the method query1() that returns the customer name given a customer id using the customer collection.

4. Write the method query2() that returns the order date for a given order id using the orders collection.

5. Write the method query2Nest() that returns order date for a given order id using the custorders collection.

6. Write the method query3() that returns the total number of orders using the orders collection.

7. Write the method query3Nest() that returns the total number of orders using the custorders collection.

8. Write the method query4() that returns the top 5 customers based on total order amount using the customer and orders collections.

9. Write the method query4Nest() that returns the top 5 customers based on total order amount using the custorders collection.

Starter Code:

```
import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Projections.*;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.mongodb.client.model.Projections.fields;

import org.bson.Document;
import org.bson.conversions.Bson;

import com.mongodb.BasicDBList;
import com.mongodb.BasicDBObject;
import com.mongodb.client.AggregateIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

/**
 * Program to create a collection, insert JSON objects, and perform simple
 * queries on MongoDB.
 */
public class MongoDB {
	/**
	 * MongoDB database name
	 */
	public static final String DATABASE_NAME = "mydb";

	/**
	 * MongoDB collection name
	 */
	public static final String COLLECTION_NAME = "data";

	/**
	 * Mongo client connection to server
	 */
	public MongoClient mongoClient;

	/**
	 * Mongo database
	 */
	public MongoDatabase db;

	/**
	 * Main method
	 *
	 * @param args
	 *            no arguments required
	 */
	public static void main(String[] args) throws Exception {
		MongoDB qmongo = new MongoDB();
		qmongo.connect();
		qmongo.load();
		qmongo.loadNest();
		System.out.println(qmongo.query1(1000));
		System.out.println(qmongo.query2(32));
		System.out.println(qmongo.query2Nest(32));
		System.out.println(qmongo.query3());
		System.out.println(qmongo.query3Nest());
		System.out.println(MongoDB.toString(qmongo.query4()));
		System.out.println(MongoDB.toString(qmongo.query4Nest()));
	}

	/**
	 * Connects to Mongo database and returns database object to manipulate for
	 * connection.
	 *
	 * @return
	 *         Mongo database
	 */
	public MongoDatabase connect() {
		try {
			// Provide connection information to MongoDB server
			// TODO: Replace with your cluster info
			String url = "";
			mongoClient = MongoClients.create(url);
		} catch (Exception ex) {
			System.out.println("Exception: " + ex);
			ex.printStackTrace();
		}

		// Provide database information to connect to
		// Note: If the database does not already exist, it will be created
		// automatically.
		db = mongoClient.getDatabase(DATABASE_NAME);
		return db;
	}

	/**
	 * Loads TPC-H data into MongoDB.
	 *
	 * @throws Exception
	 *                  if a file I/O or database error occurs
	 */
	public void load() throws Exception {
		// TODO: Load customer and orders data
	}

	/**
	 * Loads customer and orders TPC-H data into a single collection.
	 *
 * @throws Exception
 *                  if a file I/O or database error occurs
 */
public void loadNest() throws Exception {
          // TODO: Load customer and orders data into single collection called custorders
          // TODO: Consider using insertMany() for bulk insert for faster performance
}

/**
 * Performs a MongoDB query that prints out all data (except for the _id).
 */
public String query1(int custkey) {
          System.out.println("\nExecuting query 1:");
          // TODO: Write query
          MongoCollection<Document> col = db.getCollection("customer");

          // See: https://docs.mongodb.com/drivers/java/sync/current/usage-examples/find/

          return null;
}

/**
 * Performs a MongoDB query that returns order date for a given order id using
 * the orders collection.
 */
public String query2(int orderId) {
          // TODO: Write a MongoDB query
          System.out.println("\nExecuting query 2:");

          return null;
}

/**
 * Performs a MongoDB query that returns order date for a given order id using
 * the custorders collection.
 */
public String query2Nest(int orderId) {
          // TODO: Write a MongoDB query
          System.out.println("\nExecuting query 2 nested:");

          MongoCollection<Document> col = db.getCollection("custorders");

          return null;
}
```

```
/**
 * Performs a MongoDB query that returns the total number of orders using the
 * orders collection.
 */
public long query3() {
        // TODO: Write a MongoDB query
        System.out.println("\nExecuting query 3:");

        MongoCollection<Document> col = db.getCollection("orders");
        return 0;
}

/**
 * Performs a MongoDB query that returns the total number of orders using the
 * custorders collection.
 */
public long query3Nest() {
        // TODO: Write a MongoDB query
        System.out.println("\nExecuting query 3 nested:");
        MongoCollection<Document> col = db.getCollection("custorders");

        return 0;
}

/**
 * Performs a MongoDB query that returns the top 5 customers based on total
 * order amount using the customer and orders collections.
 */
public MongoCursor<Document> query4() {
        // TODO: Write a MongoDB query. Note: Return an iterator.
        System.out.println("\nExecuting query 4:");

        return null;
}

/**
 * Performs a MongoDB query that returns the top 5 customers based on total
 * order amount using the custorders collection.
 */
public MongoCursor<Document> query4Nest() {
        // TODO: Write a MongoDB query. Note: Return an iterator.
        System.out.println("\nExecuting query 4 nested:");

        MongoCollection<Document> col = db.getCollection("custorders");
```

```
		return null;
	}

	/**
	 * Returns the Mongo database being used.
	 *
	 * @return
	 *        Mongo database
	 */
	public MongoDatabase getDb() {
		return db;
	}

	/**
	 * Outputs a cursor of MongoDB results in string form.
	 *
	 * @param cursor
	 *             Mongo cursor
	 * @return
	 *        results as a string
	 */
	public static String toString(MongoCursor<Document> cursor) {
		StringBuilder buf = new StringBuilder();
		int count = 0;
		buf.append("Rows:\n");
		if (cursor != null) {
			while (cursor.hasNext()) {
				Document obj = cursor.next();
				buf.append(obj.toJson());
				buf.append("\n");
				count++;
			}
			cursor.close();
		}
		buf.append("Number of rows: " + count);
		return buf.toString();
	}
}
```