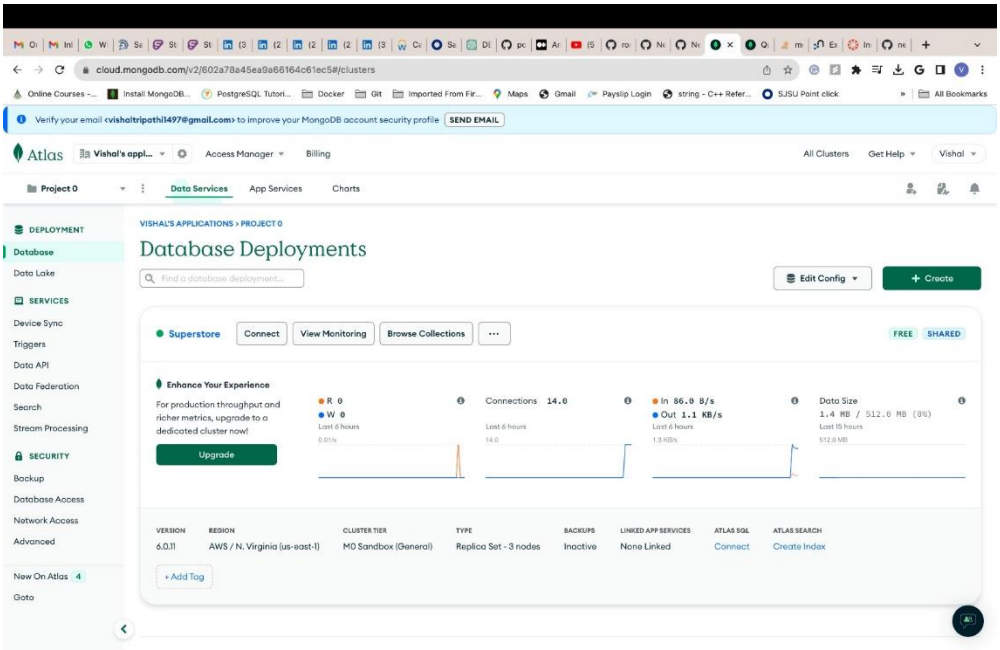
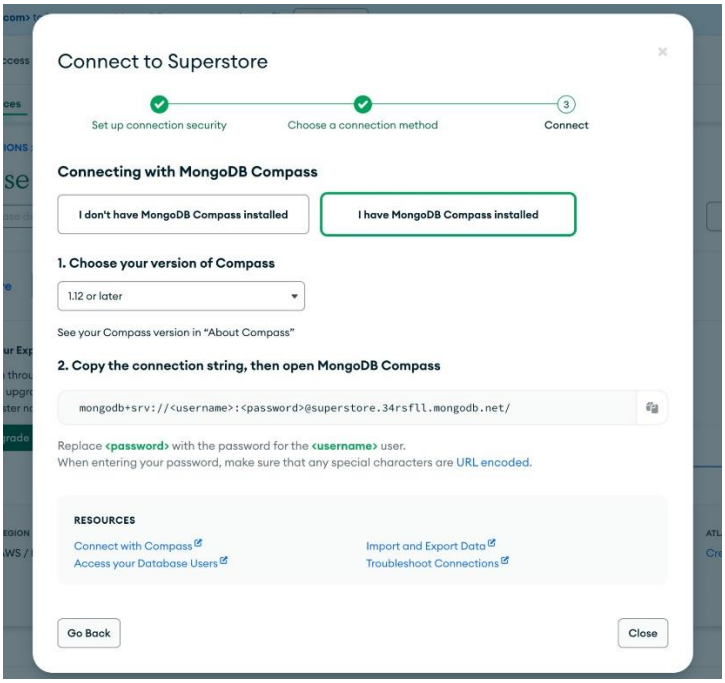
MONGODB EXPLAINATIONS AND CODES

1. Establishing Connection:

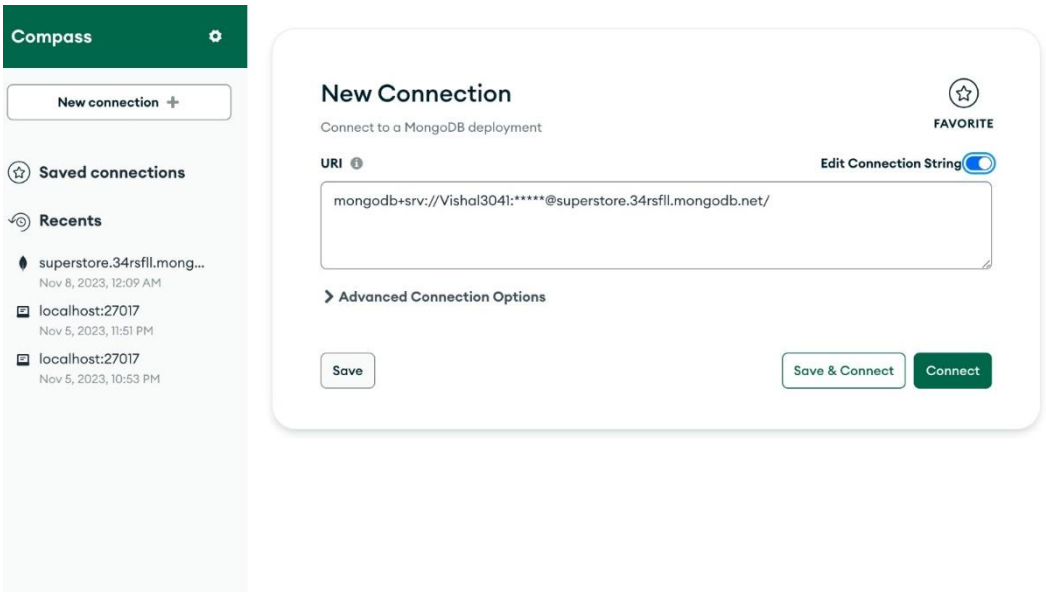
STEP-1: First go on atlas website and open the page. There click on “create” and connection page will open up.



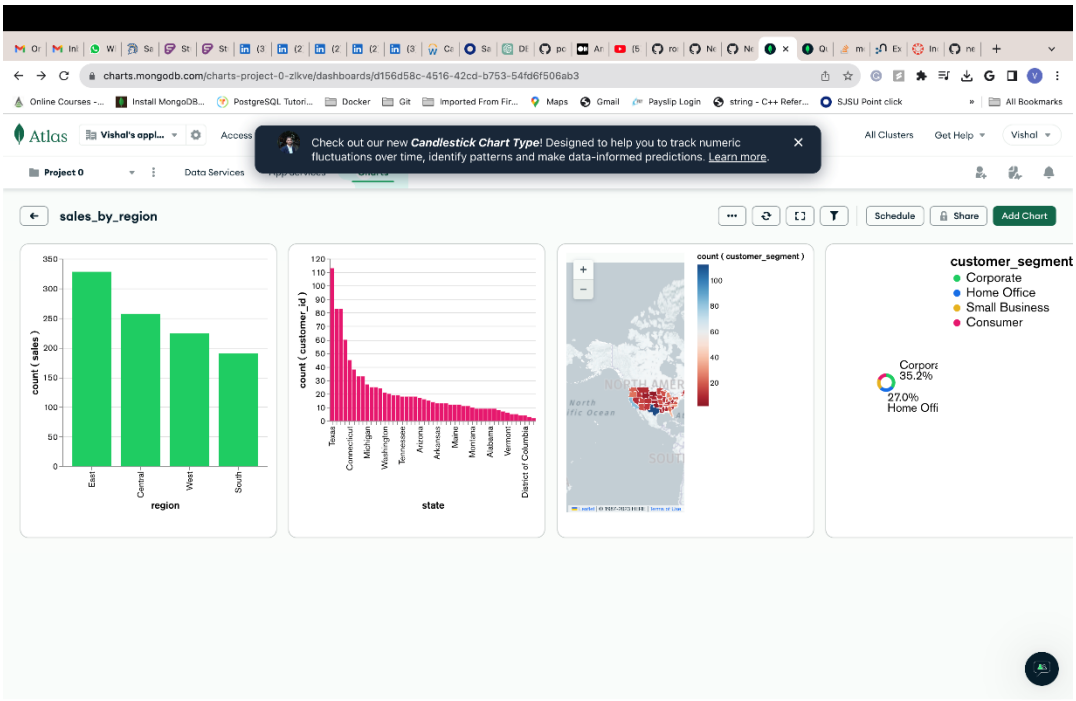
STEP-2: Now on the pop up, if you have mongoDB compass, select the option saying “I have mongoDB compass installed” and choose the version that you have you will get a url, copy that.



STEP-3: Now open mongoDB compass and in the new connection, it will be asking for a url or a string to connect to. There paste the link you just copied on the portal and hit connect. The connection is established.



2. Plot Dashboard:

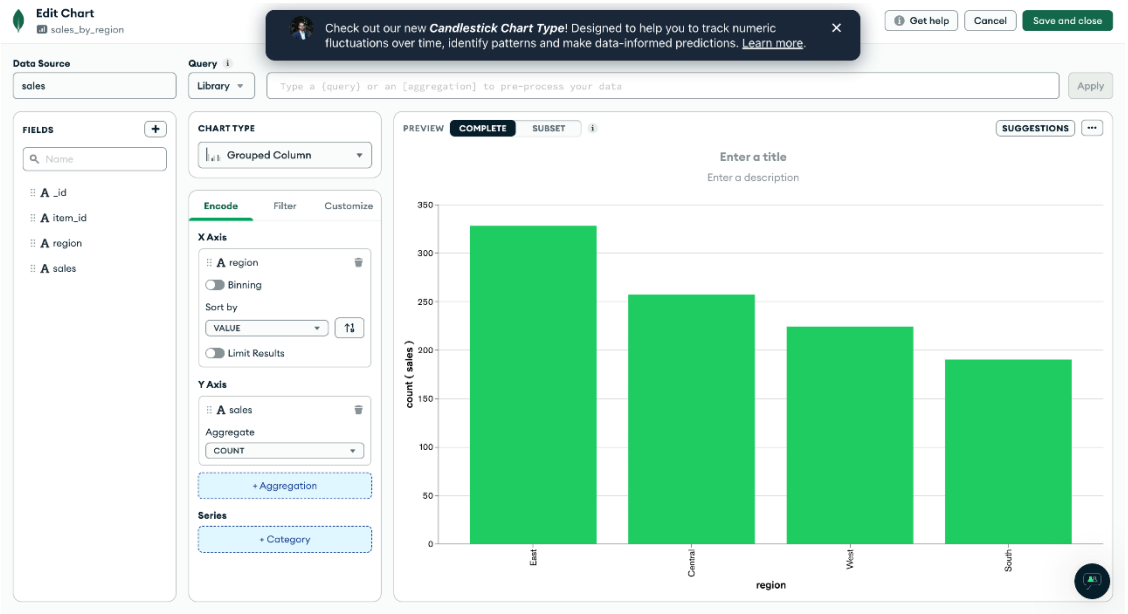


This is the dashboard created on mongoDB charts shows 4 different plots made for the sales data and they will be explained below.

3. Plots:

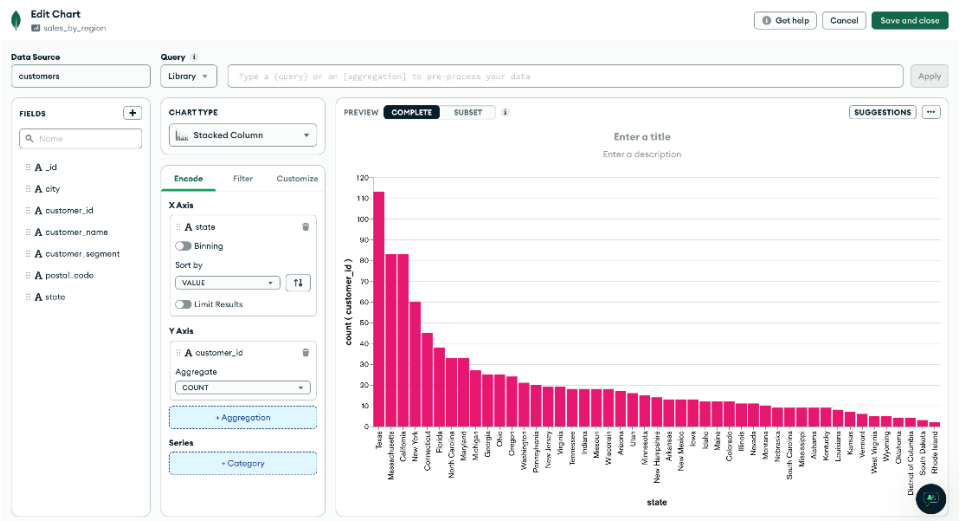
a. Sales table

From the sales table in the database, we have created this plot that shows regions on the x axis and the count of customers on the y axis giving us a detailed analysis of customer behavior telling the company which region uses their products the most.



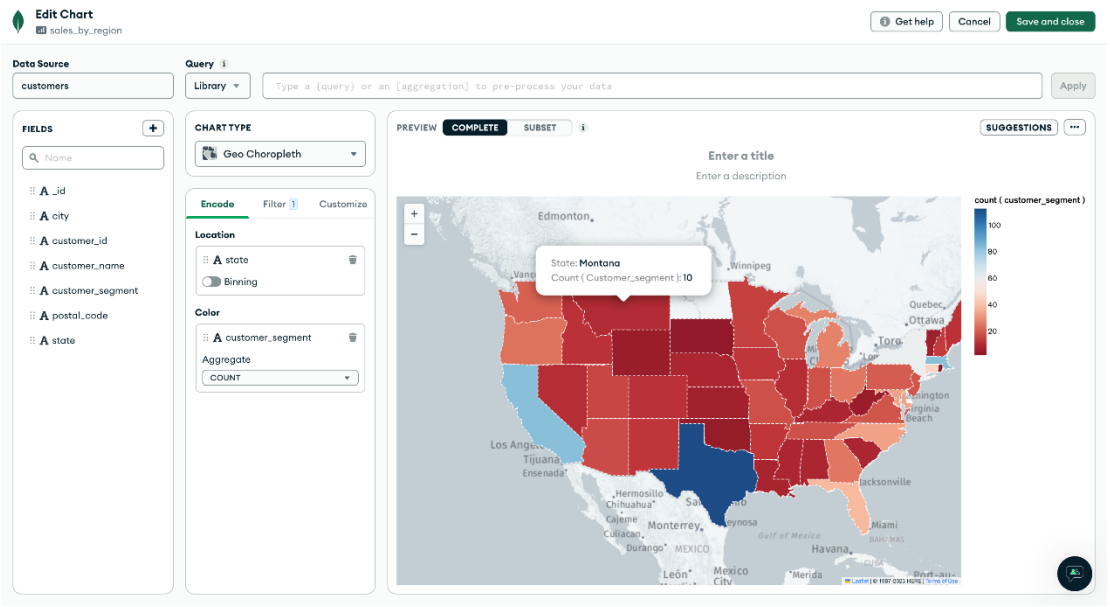
b. Customer table

This time we have chosen the customer table and on the plot, we have state where the customer belongs to on the x axis and on the y axis we have count of customers and we can see the aggregates on the encode part left of the plot. Again, this plot gives the company an analysis and inference which shows which state has most of their customers and on the basis of that the company can make some business decisions in order to maintain a well-balanced supply and demand process making the system more efficient than the existing one.



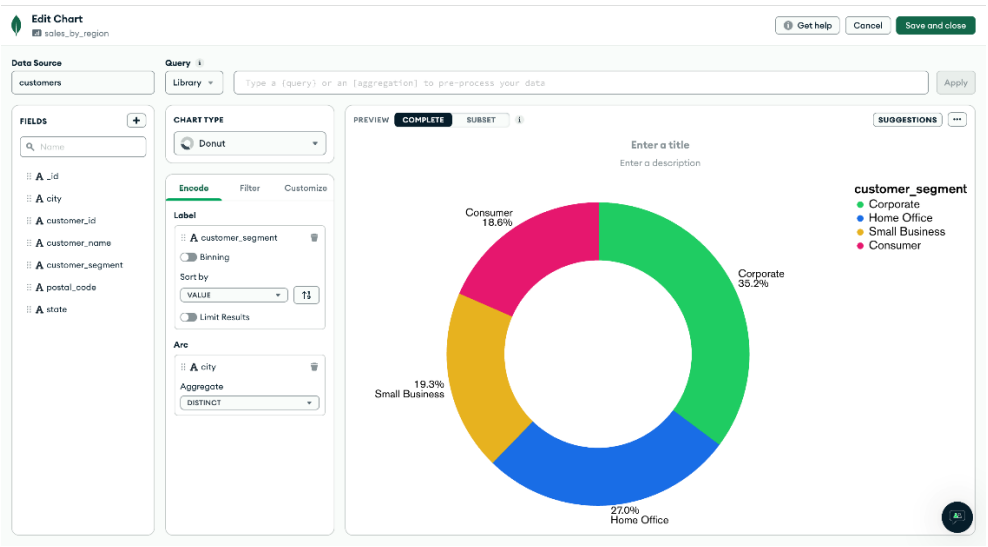
c. Customer table

Here again in the customer table this time we are analyzing the customer segments (office, home office, business, home) count with the respective states so that we can analyze what states order most of what type of segment products and later the company can deploy those amounts of segments to their popularity in each state accordingly again solving the above issue. Here we have a choropleth plot where the color shows the count of segment, and the geolocations here have states in the USA giving us an easier to understand data pictorially.

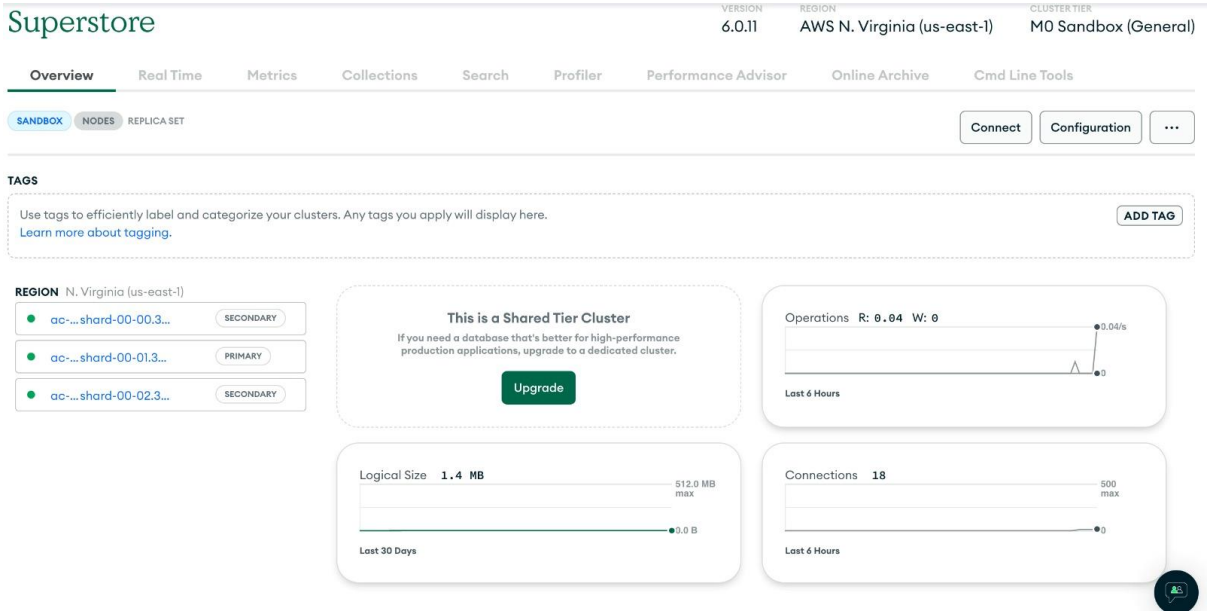


d. Customer table

For this plot, we have created a doughnut plot now providing an overall detail and count in percentages of what type of customer segments are in the dataset so far and their popularity for that specific company. Each color here represents a type of segment and their area covered is their weightage.



4. Sharding:



MongoDB creates shards and by default it creates 3 shards where we can see that 1 is primary and 2 are secondary shards.

5. Aggregate and CRUD operations:

a. Python command for connection and import csv data to mongoDB

```
[47]: import csv
from pymongo import MongoClient

# Establish a connection to MongoDB
client = MongoClient("mongodb+srv://[username]:[password]@superstore.34rsfll.mongodb.net/")
db = client["Superstore_database"]

# Load data from CSV into MongoDB collections
def import_data_into_collection(csv_filename, collection_name, relevant_columns):
    collection = db[collection_name]
    with open(csv_filename, 'r') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            document = {key: row[key] for key in relevant_columns}
            collection.insert_one(document)

# Define relevant columns for each collection
customer_columns = ['customer_id', 'customer_name', 'customer_segment', 'city', 'state', 'postal_code']
order_columns = ['order_id', 'customer_id', 'item_id', 'order_date', 'order_quantity', 'shipping_cost',
product_columns = ['item_id', 'item', 'category', 'department', 'unit_price']
sales_columns = ['item_id', 'region', 'sales']

# Import data into relevant collections
import_data_into_collection('superstore.csv', 'customers', customer_columns)
import_data_into_collection('superstore.csv', 'orders', order_columns)
import_data_into_collection('superstore.csv', 'products', product_columns)
import_data_into_collection('superstore.csv', 'sales', sales_columns)

# # Close the MongoDB connection
# client.close()
```

Here first we are connecting python with mongoDB with the help of pymongo by providing the database name and the url that we created earlier. Then we are reading the cleaned and refined csv file for the super store where with the help of for loops we are reading the elements in the file. And lastly, we are importing

the csv file in the database and creating 4 different tables namely customers, orders, products, sales and allotting the column names from our csv file to input in the database for entries and creating the document.

b. Update command

```
[57]: # Update a single document
      query = {"customer_id": "3035"}
      new_values = {"$set": {"customer_name": "Harshal Shah"}}
      db.customers.update_one(query, new_values)
```

```
[57]: UpdateResult({'n': 1, 'electionId': ObjectId('7fffffff0000000000000f1'), 'opTime': {'ts': Timestamp(16
      99434567, 3), 't': 241}, 'nModified': 1, 'ok': 1.0, '$clusterTime': {'clusterTime': Timestamp(169943456
      7, 3), 'signature': {'hash': b'QF\x11\xc0\x84\xb5:\x02T\x8e9\x98\xf3\x1f\xd1;g\xabsZ', 'keyId': 7233456
      183300849671}}, 'operationTime': Timestamp(1699434567, 3), 'updatedExisting': True}, acknowledged=True)
```

In the command here, we are updating a row where the customer_id is 3035 and changing the name from old to new (harshal shah).

c. Finding all the rows

```
[61]: ans = db.customers.find({"customer_id": "3035"})
```

```
[63]: for x in ans :
          print(x)
```

```
{'_id': ObjectId('654b274335f7479ef5356032'), 'customer_id': '3035', 'customer_name': 'Harshal Shah',
'customer_segment': 'Home Office', 'city': 'Lombard', 'state': 'Illinois', 'postal_code': '60148'}
{'_id': ObjectId('654b274435f7479ef5356033'), 'customer_id': '3035', 'customer_name': 'Mark Bailey', 'c
ustomer_segment': 'Home Office', 'city': 'Lombard', 'state': 'Illinois', 'postal_code': '60148'}
```

In this code/ command, we are finding all the rows where customer_id is 3035 giving us all the orders by that customer and as an output we can see that there are 2 rows returned.

d. Deleting command

```
[66]: db.customers.delete_one({"customer_id": "3035"})
```

```
[66]: DeleteResult({'n': 1, 'electionId': ObjectId('7fffffff0000000000000f1'), 'opTime': {'ts': Timestamp(16
      99434746, 36), 't': 241}, 'ok': 1.0, '$clusterTime': {'clusterTime': Timestamp(1699434746, 36), 'signat
      ure': {'hash': b'\xac\xe6jS8\xf5F\xaf"5#S\xbf\xe9\x88\x16\\\xb0(\x02', 'keyId': 723456183300849671}},
      'operationTime': Timestamp(1699434746, 36)}, acknowledged=True)
```

In the command here, we are deleting the row where the customer_id is 3035 and removing it from the database completely as a CRUD operation example.