**MSIS 685 – Introduction to Big Data**

**Big data technology - MongoDB**

**Title – Integrative data solutions in MongoDB - CRUD, aggregation & visualization of data**

Submitted by,

**Group 11**

**Saipriya Ramanan**

*[saipriya.ramanan001@umb.edu]*

**Dimple Asha Gurajala**

*[GurajalaRajendraN001@umb.edu]*

**Nikita D Navale**

*[Nikita.Navale001@umb.edu]*

Tutorial Index

[**Introduction:** 3](#_Toc164613659)

[**Use-case overview:** 3](#_Toc164613660)

[**Technologies used:** 4](#_Toc164613661)

[**Project Scope:** 4](#_Toc164613662)

[**Pros & Cons of MongoDB:** 4](#_Toc164613663)

[**Comparison with Alternatives:** 5](#_Toc164613664)

[**Software requirements:** 6](#_Toc164613665)

[**Sample datasets:** 6](#_Toc164613666)

[**Estimated time & cost :** 6](#_Toc164613667)

[**Steps to follow to gain Hands-On with MongoDB:** 6](#_Toc164613668)

[**Step 1: Creating an account and a cluster in MongoDB Atlas** 7](#_Toc164613669)

[**Step 2: Connecting MongoDB Atlas to MongoDB Compass through connection link** 8](#_Toc164613670)

[**Step 3: CRUD (Create, Read, Update, Delete) and aggregation operations in Compass** 9](#_Toc164613671)

[**Step 4: Visualization of data using charts in MongoDB Atlas** 12](#_Toc164613672)

[**Step 5: CRUD operations and Data visualization in Visual studio code using Python** 14](#_Toc164613673)

# **Introduction:**

**What is MongoDB?**

MongoDB is a type of database software that differs from traditional database systems. Unlike the classic relational databases which store data in tables (rows and columns), MongoDB stores data in a more flexible format called documents. These documents are grouped into collections, and each document can have a different structure. This format is based on JSON (JavaScript Object Notation), a popular format for exchanging data on the web, making MongoDB particularly well-suited for modern web applications.

**Where is it used?**

MongoDB is used in a wide variety of applications and industries due to its flexibility, scalability, and ease of use. Some of the examples are,

* Web applications
* Mobile applications
* Internet of Things (IoT)
* E-Commerce platform
* Gaming, etc

# **Use-case overview:**

This project focuses on demonstrating the essential CRUD (Create, Read, Update, Delete) operations using MongoDB, a leading NoSQL database, with Python as the programming language. The operations are implemented within Visual Studio Code, a popular development environment that supports an array of programming tools and extensions, including those for Python and MongoDB. This setup provides a practical scenario for developing and testing database-driven applications.

The primary objective of this project is to create a simple yet effective demonstration of how MongoDB can be interfaced with Python to perform database operations. This involves setting up a MongoDB database, connecting to it through Python using the **pymongo** library, and executing typical database operations that any data-driven application would require.

Understanding CRUD operations is fundamental for any software developer working with databases, as these operations form the backbone of data manipulation and retrieval. MongoDB, being a highly scalable and flexible NoSQL database, offers a unique approach to managing data that can be particularly advantageous in scenarios involving large, unstructured, or semi-structured data. This project not only aims to teach these basic operations but also to provide insight into the practical application of MongoDB in modern web and application development.

# **Technologies used:**

* ***MongoDB****:* A document-oriented NoSQL database used for high volume data storage.
* ***Python****:* A versatile programming language favoured for its readability and broad range of applications in web development, data analysis, artificial intelligence, scientific computing, and more.
* ***PyMongo:*** The official MongoDB driver for Python, used to connect to and interact with the database.
* ***Visual Studio Code:*** An extensible code editor that supports a wide array of programming languages and tools, ideal for developing and debugging Python applications.

# **Project Scope:**

The scope of this project includes,

* Setting up a MongoDB database and configuring it to run with Python.
* Developing Python scripts to perform CRUD operations, thereby manipulating data stored in MongoDB.
* Demonstrating the use of pymongo for database connections and operations.
* Highlighting the real-time impact of these operations through simple, interactive Python scripts.
* Performing aggregation operations in MongoDB compass
* And visualizing the trends using MongoDB charts

# **Pros & Cons of MongoDB:**

|  |
| --- |
| **Advantages** |
| **High Flexibility**  MongoDB allows storing of varied data types without a fixed schema, which is a significant advantage in agile development environments. |
| **Scalability**  Efficiently handles large data sets and traffic through sharding and replication. |
| **Rich Query Language**  Supports complex queries and aggregation operations, making it powerful for data analysis. |
| **User Friendly**  MongoDB's document model is considered more intuitive for developers, especially those with a background in JavaScript. |

|  |
| --- |
| **Disadvantages** |
| **Data Redundancy**  Without transactional integrity checks, MongoDB can lead to data redundancy and consistency issues in complex transaction scenarios |
| **Memory Usage**  MongoDB stores key names for each document, which can lead to higher memory consumption compared to column-based databases |
| **Join Operations**  While MongoDB supports join operations, they are typically more resource-intensive and slower compared to relational databases |

# **Comparison with Alternatives:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Feature*** | ***MongoDB*** | ***MySQL (SQL Databases)*** | ***Cassandra (NoSQL)*** | ***Redis (NoSQL)*** |
| ***Data Model*** | Document-based, stores data in BSON (binary JSON). | Relational, stores data in tables with rows and columns. | Wide-column store, stores data in rows with a dynamic column model. | Key-value store, manages data as a collection of key-value pairs. |
| ***Schema Flexibility*** | Schema-less, allows dynamic schema. | Requires predefined schema, less flexibility. | Schema-flexible, but typically designed around a fixed schema. | Schema-less, simple data model focused on keys and values. |
| ***Scaling*** | Horizontally scalable via sharding. | Vertically scalable, often requires scaling hardware. | Horizontally scalable, designed to run across multiple nodes. | Horizontally scalable, but primarily in-memory. |
| ***Query Language*** | Uses MongoDB Query Language, JSON-style documents. | Uses SQL, powerful for complex queries and joins. | Uses CQL (Cassandra Query Language), like SQL. | No query language, operations are based on direct key access. |
| ***Performance*** | Fast for simple queries, slower for complex joins. | Optimal for complex joins and transactions. | High write and read throughput, scalable performance. | Extremely fast due to in-memory data access. |
| ***Consistency and Availability*** | Configurable consistency models (eventual consistency available). | Strong consistency with ACID compliance. | Tuneable consistency can favour availability over consistency. | Configurable but favours availability. |
| ***Use Cases*** | Ideal for dynamic, unstructured data, content management, real-time analytics. | Best for structured data with complex relationships, banking systems. | Suitable for very large data sets, real-time analytics - write performance is critical | Perfect for caching, real-time analytics, session storage. |

# **Software requirements:**

1. **Visual Studio Code**

Available for free download at <https://code.visualstudio.com/>

1. **MongoDB Compass**

Participants can choose to install MongoDB compass locally from

<https://www.mongodb.com/products/tools/compass>

1. **MongoDB Atlas**

Available for free download at <https://www.mongodb.com/cloud/atlas/register>

1. **Python**

Python 3.8 or newer installed, which can be downloaded from <https://www.python.org/downloads/>

1. **Python libraries**

This can be installed via pip with the command pip install pymongo. Also install matplotlib and pandas

# **Sample datasets:**

****

# **Estimated time & cost :**

About 1.5 hours to complete the entire tutorial, including installation and configuration of all necessary software, and hands-on implementation of CRUD operations. No additional cost if MongoDB and all software are installed locally.

Note - If using AWS (Amazon Web Services) to host MongoDB, the cost will depend on the chosen configuration and usage time. A basic setup on AWS Document DB or MongoDB Atlas could cost around $0.10 to $0.25 per hour. Participants should terminate all instances after completing the tutorial to avoid extra charges.

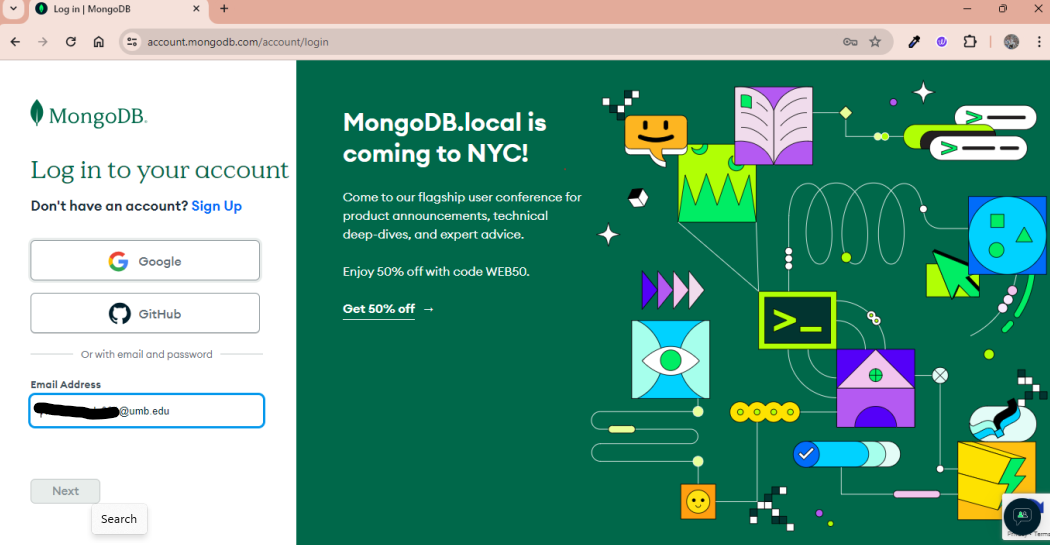
# **Steps to follow to gain Hands-On with MongoDB:**

Below are the steps to follow to perform CRUD operations in MongoDB compass without any knowledge of querying, create a pipeline to filter out data and to visualize the same in MongoDB Atlas

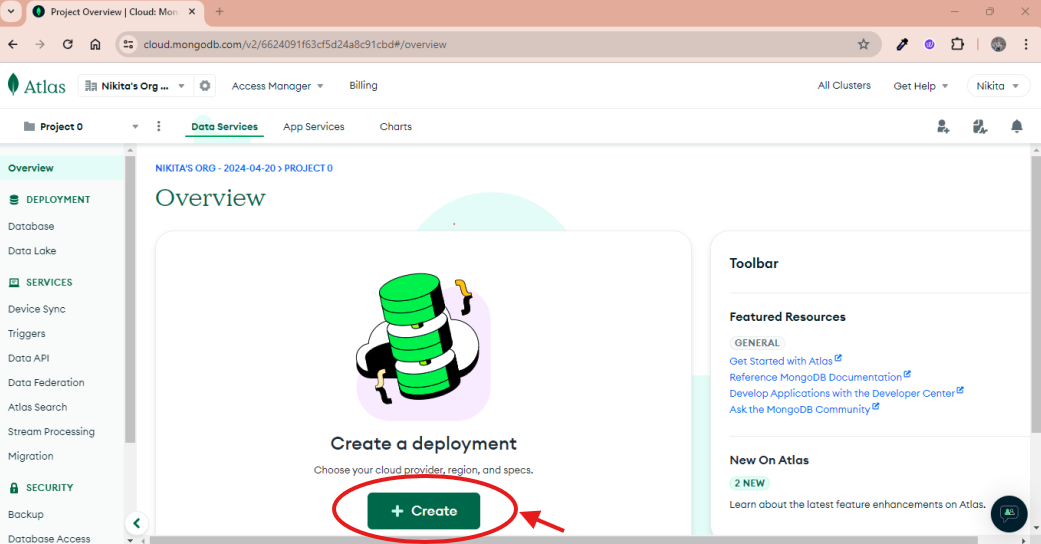
*Pre-requisite: Install all the Software’s, plug-ins and libraries which are mentioned under Software requirements*

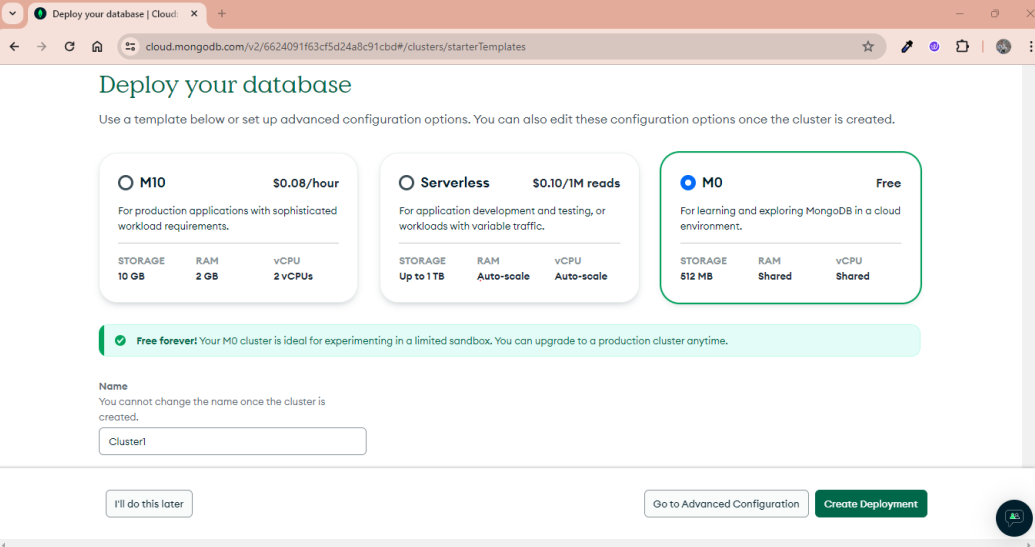
## **Step 1: Creating an account and a cluster in MongoDB Atlas**

* Open MongoDB Atlas cloud in your browser and create a new account in [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/register) using a username and password

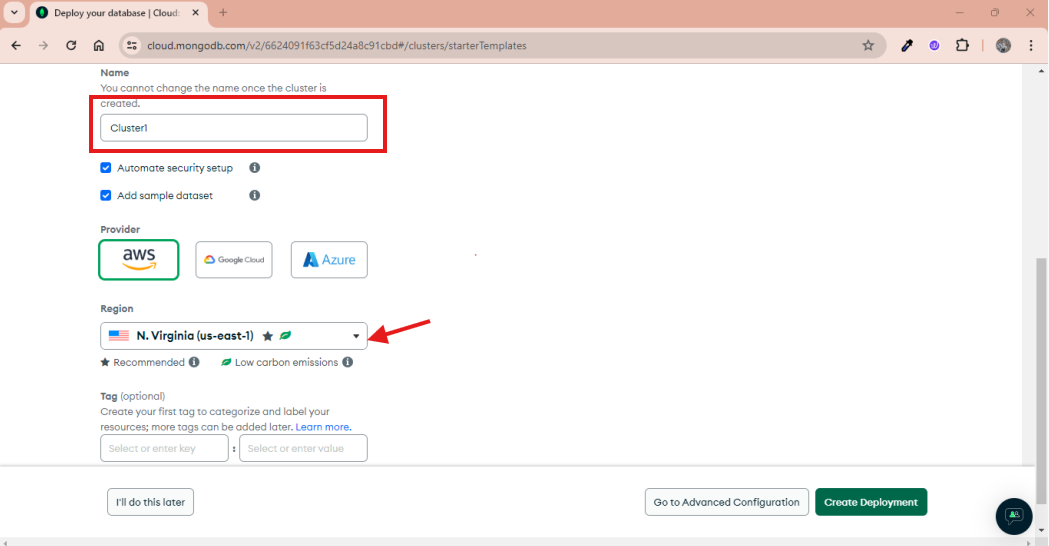


* Once after the login, user will land in Atlas overview page, now using the “+ Create” button, create a new cluster (free/unpaid version is available).
* Select the ‘M0’ free version, provide the cluster name while creating the cluster, as it cannot be renamed once after the cluster creation.
* Now select “Create deployment”



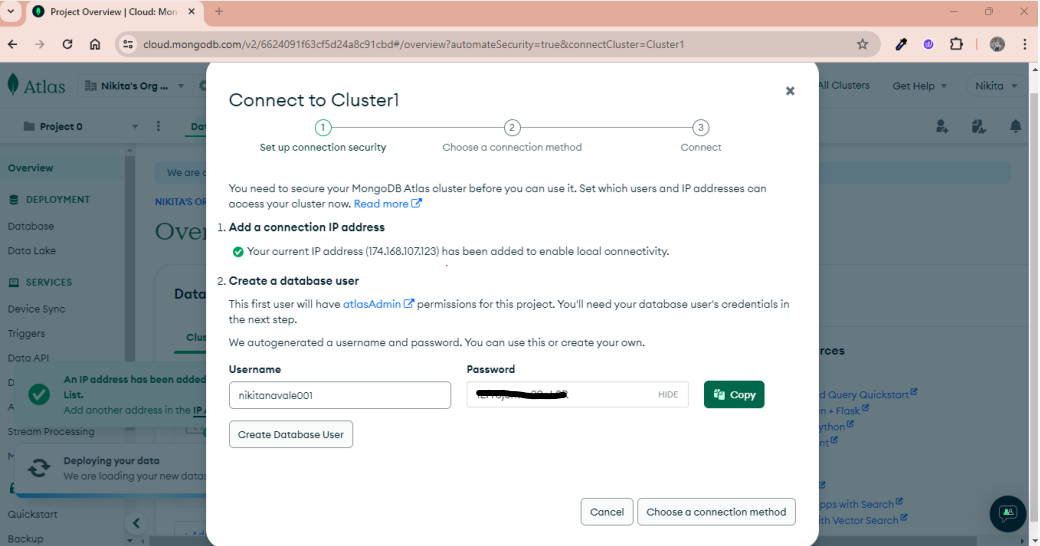


* Select Provider as ‘AWS’ and appropriate region. For e.g. N. Virginia (us-east-1).
* Finally select ‘Create Deployment’, this will create a new cluster.

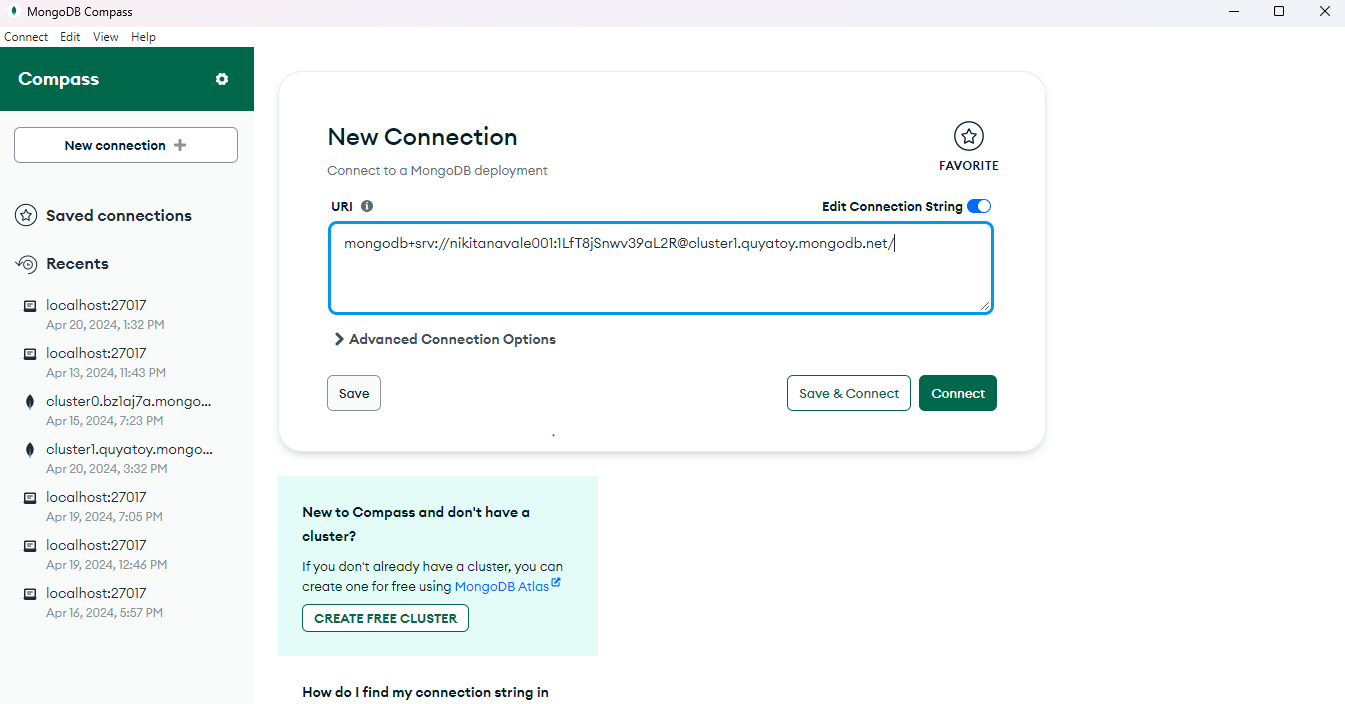


## **Step 2: Connecting MongoDB Atlas to MongoDB Compass through connection link**

* User lands in the “Connect to cluster” pop up once after the deployment is created. Username and Password will be autogenerated in the same window. **Save the credentials as this will be used in the later part of the connection.** Now Click on Create Database User and then click on choose a connection method.

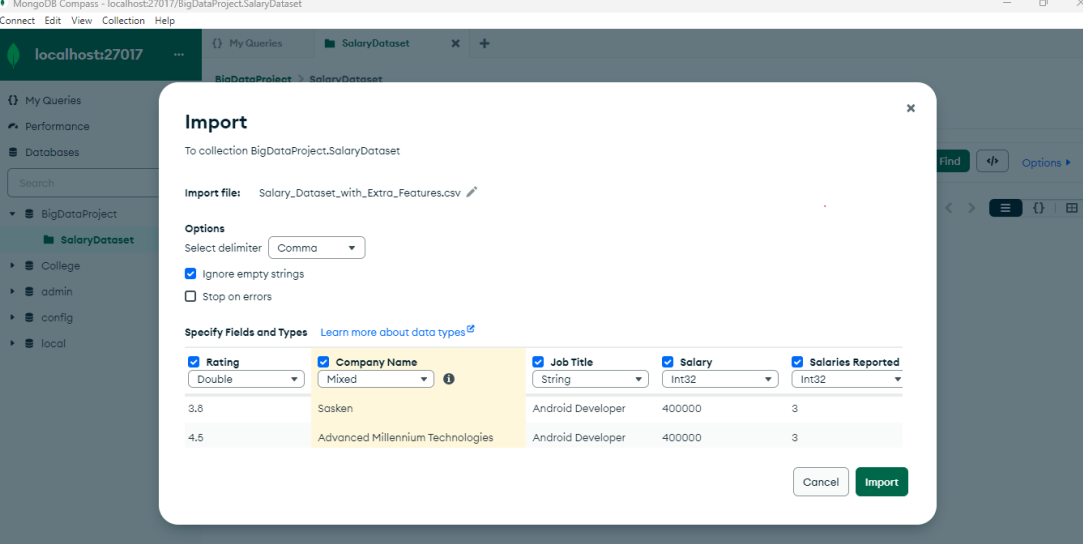


* Follow the steps and click “Compass” option under “Access your data through tools”. Now select the option that says, “I have MongoDB Compass Installed”. Copy the connection string from the same page and save it as you will need this link to connect to compass.
* Open Compass, click “New connection”, paste the connection string in the URL textbox, and replace the “<password>” with the password that was auto generated in previous step. Click connect

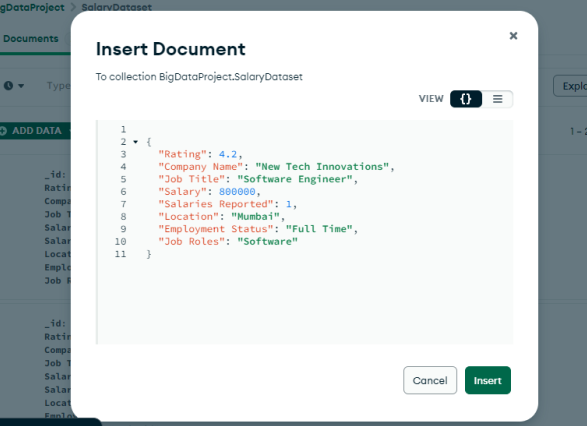


## **Step 3: CRUD (Create, Read, Update, Delete) and aggregation operations in Compass**

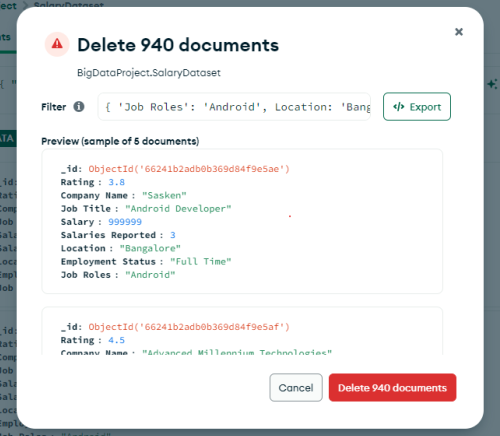
* Click the “+ADD DATA” icon next to Database from the left panel. Provide Db name, collection name, and “Create database” button.
* Database and collection will be created successfully. Now import the provided sample dataset by clicking “Add data +” button to perform CRUD and aggregation operation.



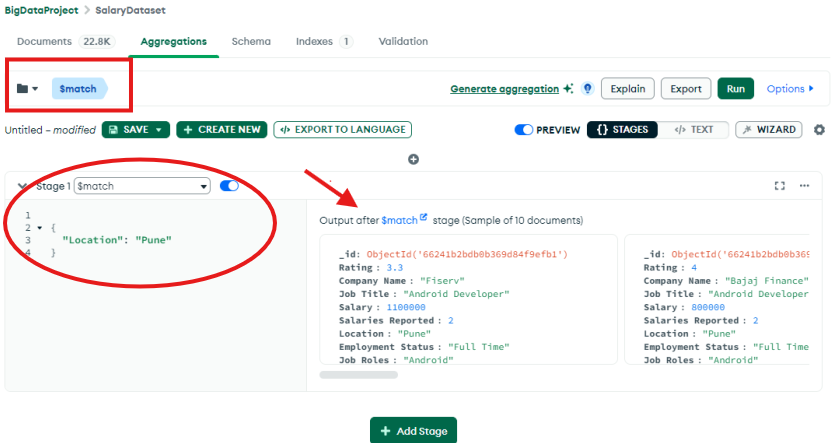
* Click “Add data” and select “Insert document” for **Create** operation. Write your query in JSON format to create a document.
* Provide a query to **read** or retrieve any value from the dataset, ***e.g. - {“Job Roles”: “Android”, “Location”: “Bangalore”}***
* Select the “**Update**” button to modify any data
* Select the “Delete” button to delete single or multiple data from the dataset.

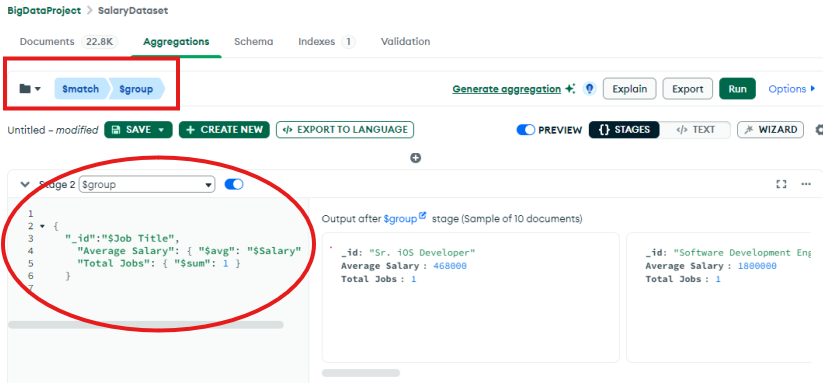


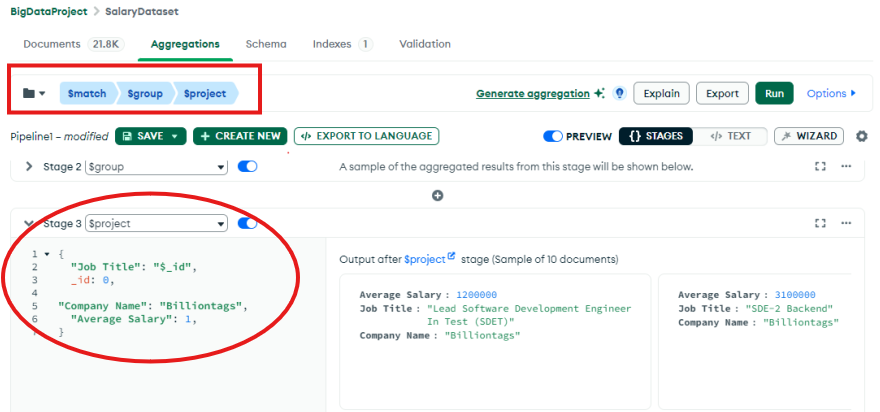




* Select the “Aggregation” tab in compass, click “Add stage” to and select the preferred operations, e.g., select **“$match”, “$Group” and “$Project”**, where match is used to filter the dataset based on the query provided. Group is used for “Group by” operations and “project” is used to select fields that you want to be displayed in the document. After creating this aggregation pipeline. You will be able to see only those records which satisfy the above condition.

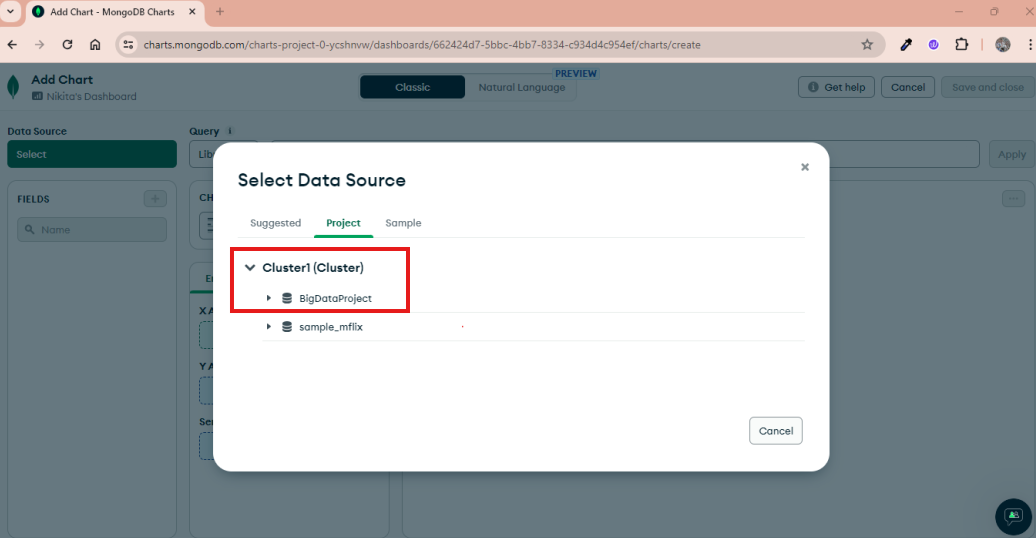






## **Step 4: Visualization of data using charts in MongoDB Atlas**

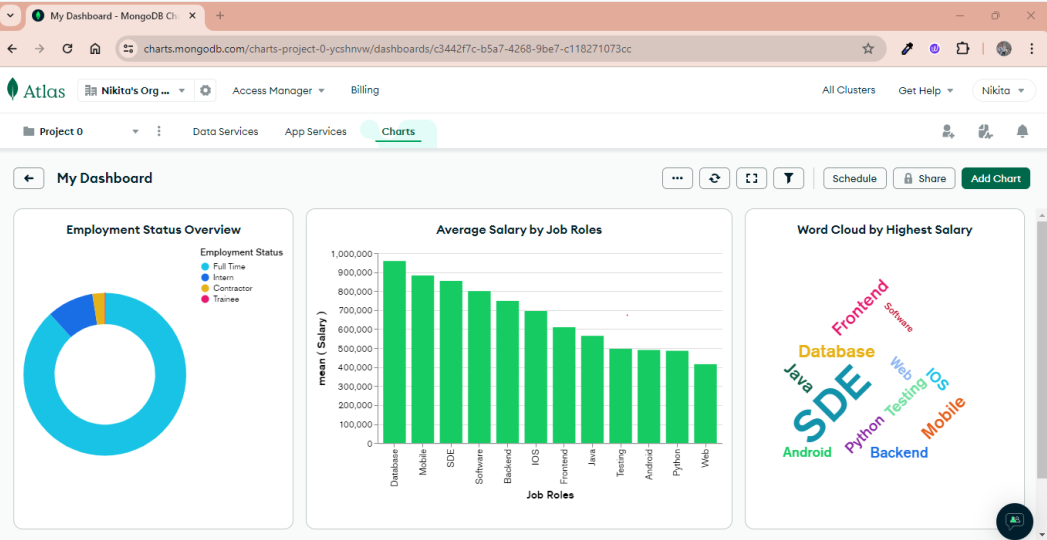
* Sign-in to MongoDB atlas using the account created in Step1. Select the “Charts” tab and click “Add Charts”. In the landing page, click “select” under data source, and choose the cluster created in MongoDB compass (BigDataProject)



* A chart will be displayed based on the dataset value, choose x-axis, y-axis, colour, or line based on your choice what you want to visualize. E.g. select Job roles as x-axis, Salary as y-axis, and in y-axis, select the mean to get the average salary for the given job roles
* Use different chart types available in the drop down and try to visualize the dataset values. The aggregation operation performed in compass can also be linked to atlas by providing the query in the search box available in the text box on top of “Edit chart” page.



* Collection of charts can be viewed under the dashboard



## **Step 5: CRUD operations and Data visualization in Visual studio code using Python**

Alternative to MongoDB Compass: Using Visual Studio Code and Python for MongoDB CRUD Operations

Visual Studio Code paired with Python offers a powerful alternative to MongoDB Compass for those who prefer or require a programmatic approach to database management. This method is particularly useful for developers who need to integrate database operations within a larger application or automate interactions with MongoDB. Here’s how you can set this up:

• Begin by setting up your environment in Visual Studio Code by installing Python, the Python extension for VS Code, and setting up a virtual environment.

• Install the MongoDB Python driver (PyMongo).

• Create a new Python file in VS Code for your MongoDB operations.

• Connect to your MongoDB database using PyMongo, and write your script to perform CRUD operations—creating, reading, updating, and deleting data.

• For data visualization, install libraries like matplotlib and pandas, then use them to create visual representations of your data directly from MongoDB.

• Finally, leverage VS Code's features to run, test, and debug your script, refining it as needed.

***Create – using INSERT operation***

# Insert Operation

from pymongo import MongoClient

# Configuration

mongo\_connection\_string = 'mongodb://localhost:27017/'

database\_name = 'BigData'

collection\_name = 'Group11'

# Connect to MongoDB

client = MongoClient(mongo\_connection\_string)

db = client[database\_name]

collection = db[collection\_name]

try:

# Perform CRUD operations here

# Example: Insert a document

document = {'name': 'John', 'age': 30}

result = collection.insert\_one(document)

print('Inserted document ID:', result.inserted\_id)

# Other CRUD operations...

# Close the connection

client.close()

**Output:**



***Read – using printing the entire data***

# Read operation

from pymongo import MongoClient

# Configuration

mongo\_connection\_string = 'mongodb://localhost:27017/'

database\_name = 'BigData'

collection\_name = 'Group11'

# Connect to MongoDB

client = MongoClient(mongo\_connection\_string)

db = client[database\_name]

collection = db[collection\_name]

try:

# Find all documents in the collection

all\_documents = collection.find()

# Print all documents

print("All Documents:")

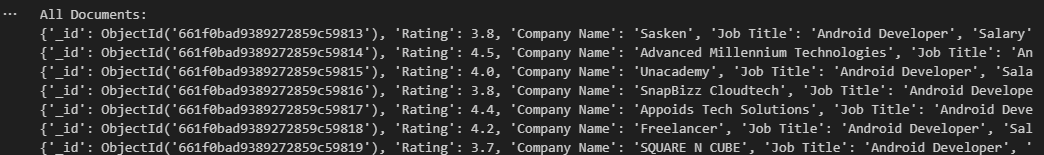
for doc in all\_documents:

print(doc)

# Close the connection

client.close()

**Output:**



***Update – using UPDATE operation***

#Update and read the updated record

from pymongo import MongoClient

# Configuration

mongo\_connection\_string = 'mongodb://localhost:27017/'

database\_name = 'BigData'

collection\_name = 'Group11'

# Connect to MongoDB

client = MongoClient(mongo\_connection\_string)

db = client[database\_name]

collection = db[collection\_name]

try:

# Perform update operation - Update a single document

# Define the filter criteria to identify the document to update

filter\_criteria = {'name': 'John'}

# Define the update operation

update\_data = {'$set': {'age': 35}} # Set the new age value

# Perform the update operation

result = collection.update\_one(filter\_criteria, update\_data)

# Print the number of documents matched and modified

print('Matched count:', result.matched\_count)

print('Modified count:', result.modified\_count)

# Retrieve and print the updated document

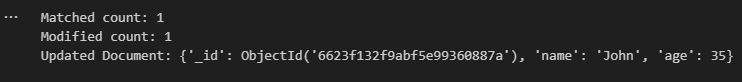
updated\_document = collection.find\_one(filter\_criteria)

print('Updated Document:', updated\_document)

# Close the connection

client.close()

**Output**

****

***Delete – using DELETE operation***

# Delete Operation

from pymongo import MongoClient

# Configuration

mongo\_connection\_string = 'mongodb://localhost:27017/'

database\_name = 'BigData'

collection\_name = 'Group11'

# Connect to MongoDB

client = MongoClient(mongo\_connection\_string)

db = client[database\_name]

collection = db[collection\_name]

try:

# Perform delete operation - Delete a single document

# Define the filter criteria to identify the document to delete

delete\_criteria = {'name': 'John'}

# Perform the delete operation

result = collection.delete\_one(delete\_criteria)

# Print the number of documents deleted

print('Deleted count:', result.deleted\_count)

# Close the connection

client.close()

**Output**

****

**Types of visualizations in python:**

**Count vs location**

#Location based Bar Graph

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from CSV file

data = pd.read\_csv(r'C:\Users\reshm\OneDrive\Desktop\BigData\Project File.csv')

# Plot bar graph for location distribution

plt.figure(figsize=(12, 6))

location\_counts = data['Location'].value\_counts()

location\_counts.plot(kind='bar', color='salmon')

plt.title('Location Distribution')

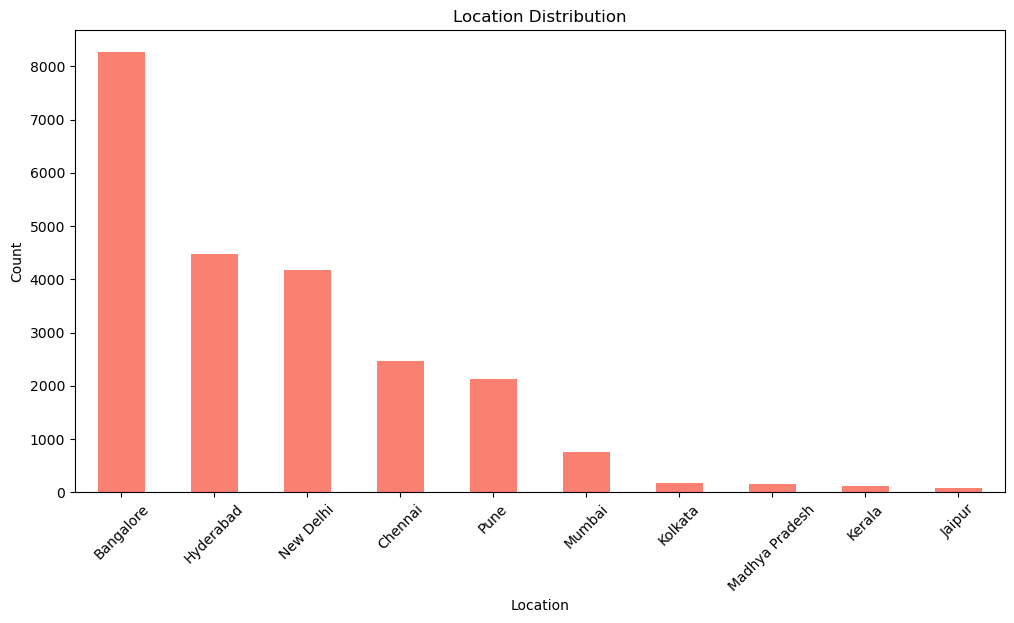
plt.xlabel('Location')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**Output:**

****

**Company distribution rate:**

# Comapny Pie Chart

import pandas as pd

import matplotlib.pyplot as plt

# Load the data from CSV file

data = pd.read\_csv(r'C:\Users\reshm\OneDrive\Desktop\BigData\Project File.csv')

# Count occurrences of each company

company\_counts = data['Company Name'].value\_counts()

# Plot pie chart

plt.figure(figsize=(8, 8))

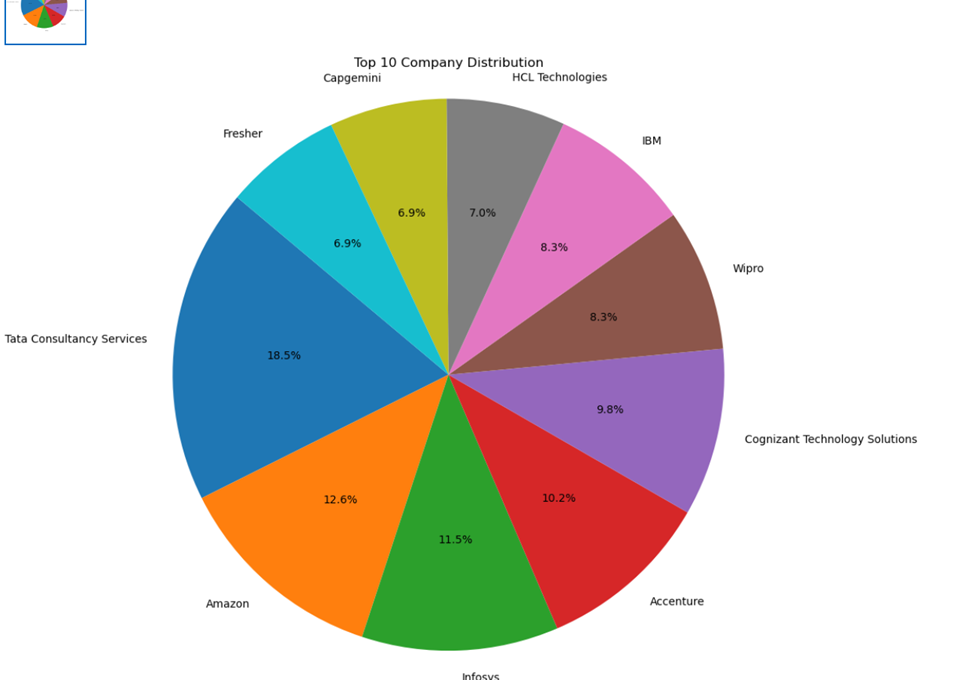
plt.pie(company\_counts, labels=company\_counts.index, autopct='%1.1f%%', startangle=140)

plt.title('Company Distribution')

plt.axis('equal')

plt.show()

**Output:**



**\*\*\*End of tutorial\*\*\***