Mongo Database

Motivation:

Nowadays software is indeed a distinguishing factor for most of the organization across the world and database is an important part of the business but most of these databases are relational database, so rather than rewriting the application and waiting for days for inclusion of simple development like adding column in table based on Agile transformation. It is suitable to switch to document database like Mongo DB. While working on traditional waterfall and agile framework, lots and lots of tables are built release by release but by using Mongo DB, one can reduce the database objects and parse through its memory rather than working on several joins. MongoDB has capability to eliminate majority of job of writing a lengthy line of code, as it offers flexible schema and allows a rapid development cycle leading to organization to reduce their timeline from conceptualization to production in days.

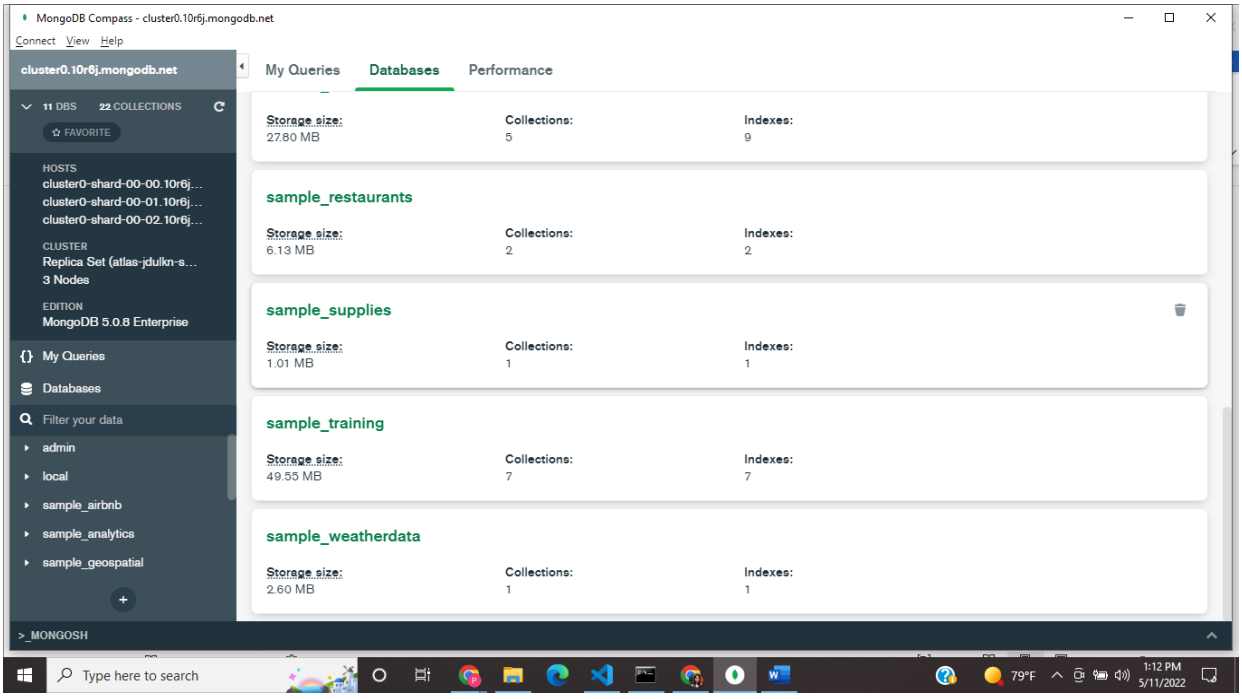
Project description: Using aggregations operation and understanding how the values from various documents are collected and grouped to return a single output.

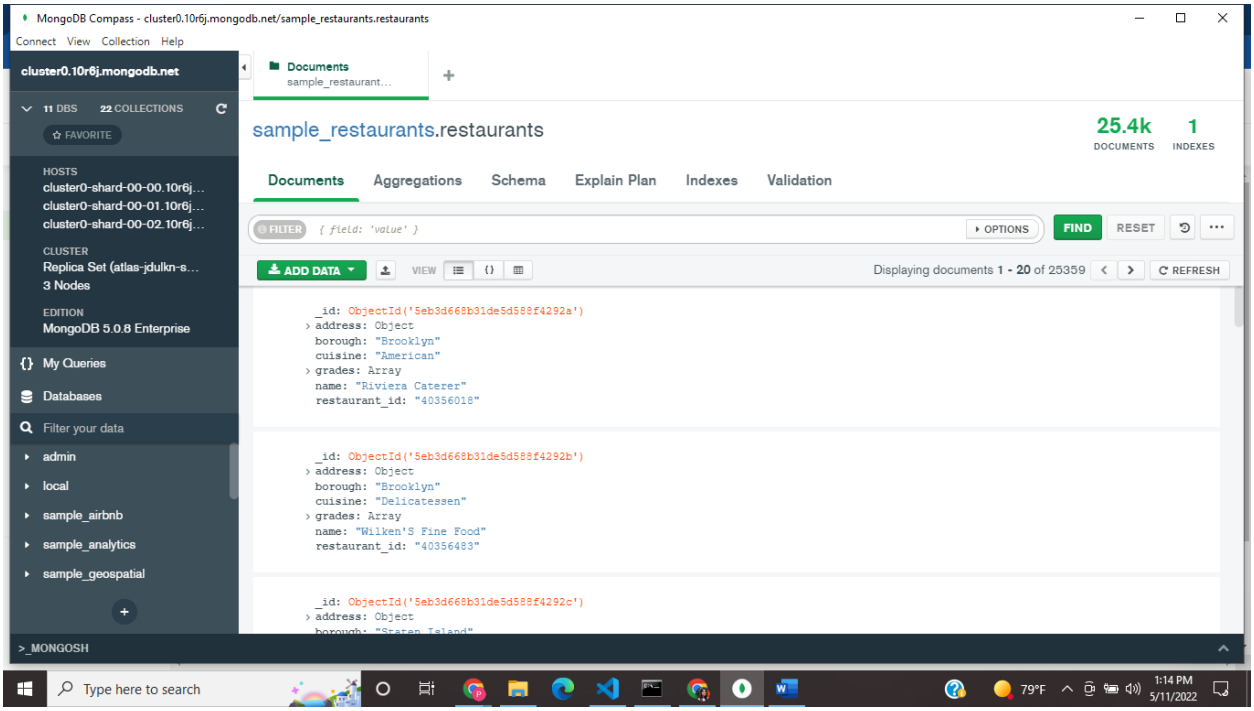
Tools and Techniques: Mongo DB Compass, Node.js, MongoDB shell, MongoAtlas

Project Approach: Idea was to use the aggregation commands through Mongodb Compass and node.js

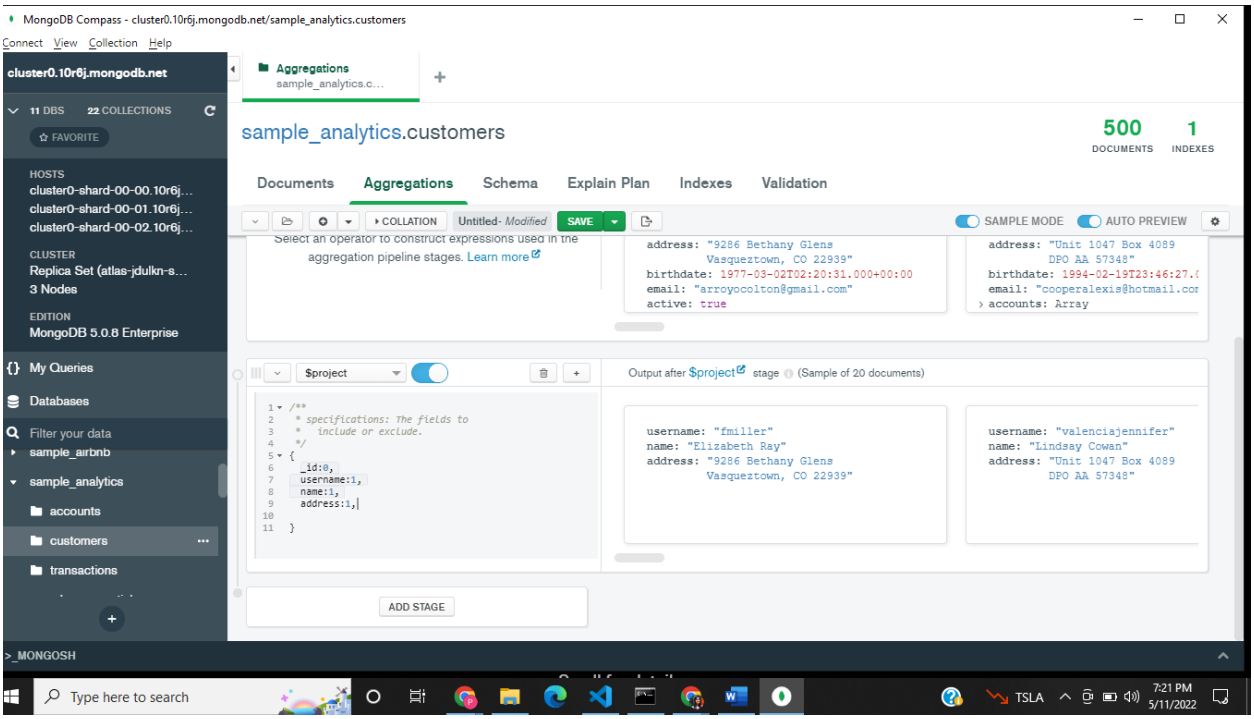
Create a MongoDB database by creating a cluster at MongoDB Atlas, load a sample and then connect to a database by using the user-id password created while creating a cluster. So for the project I wanted to explore connecting database and using query filter through MongoDB compass as well as using node.js application.

Approach 1: MongoDB Compass Below are the screen capture that shows the established connection using MongoDB compass

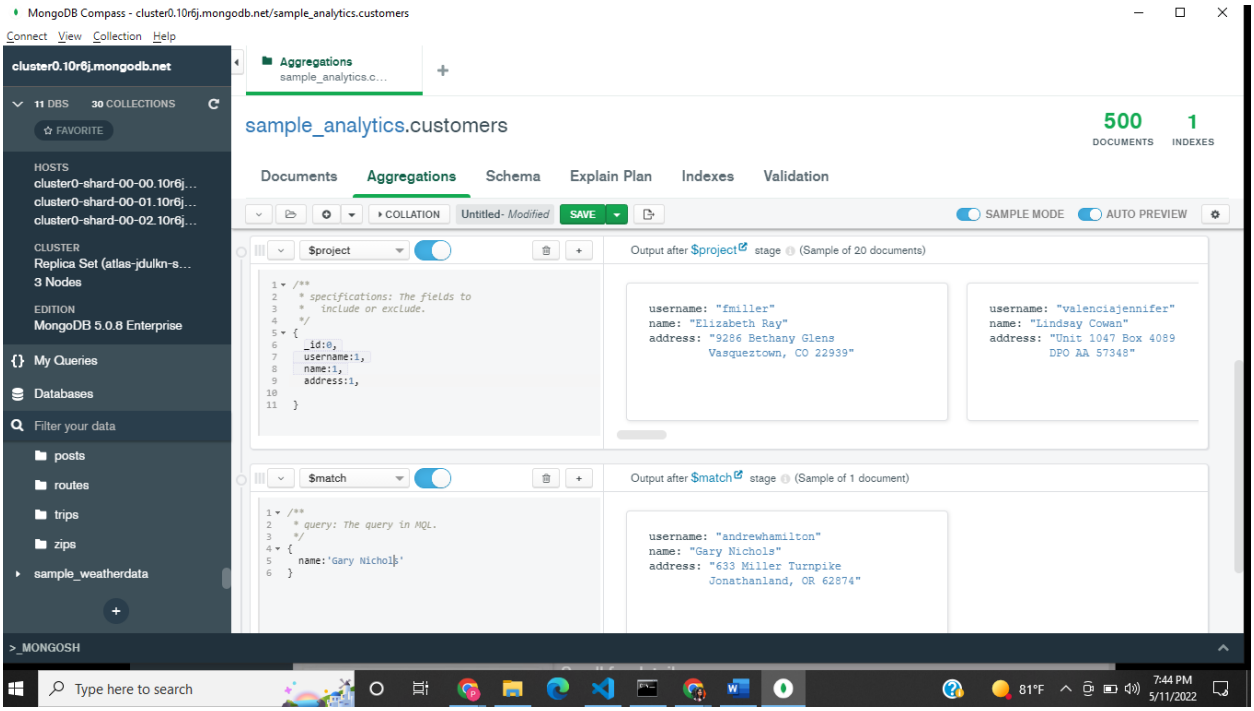




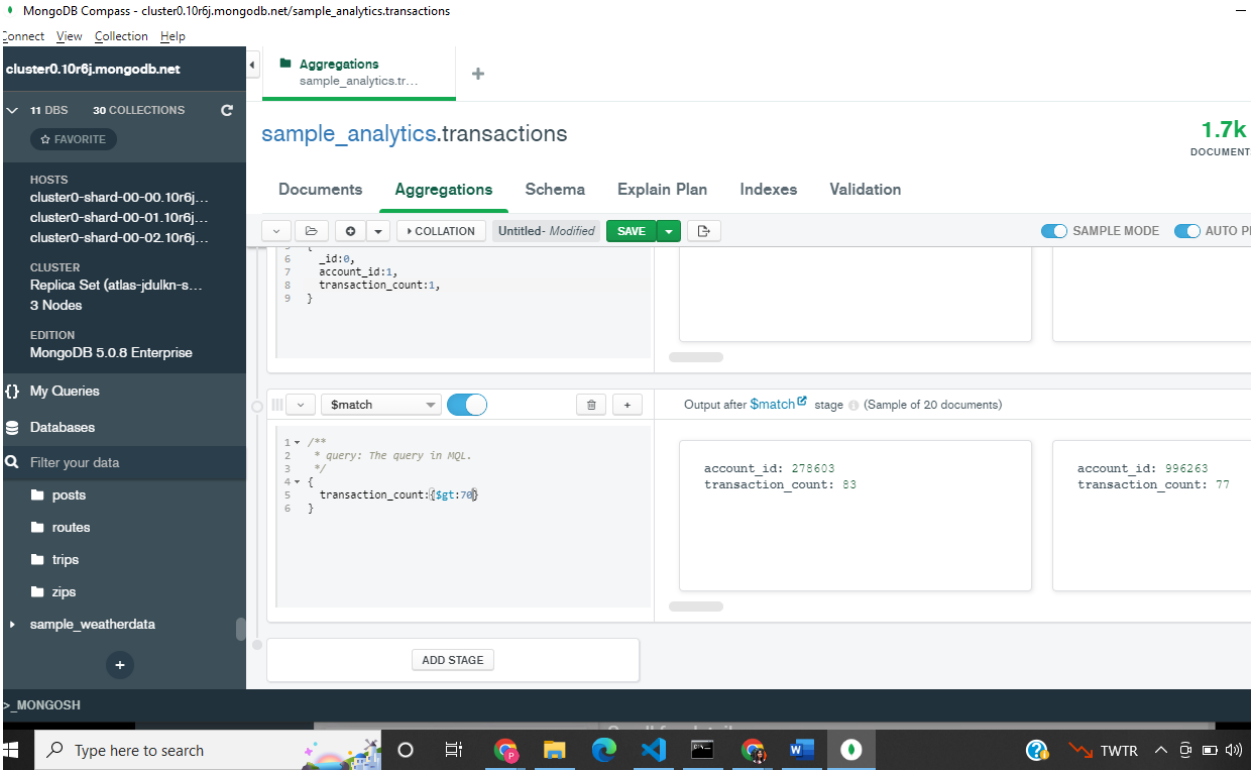
The database used is sample_analytics-customer database. To use aggregations, there are several collections of documents having multi-staged pipeline and each stage here can take a document as input and based on the operation we get output as one or multiple documents. As shown below in the pipeline stage, I used **$project** operator and specified field that I like to include i.e. id, username, name and address and we get an output of collection of documents.



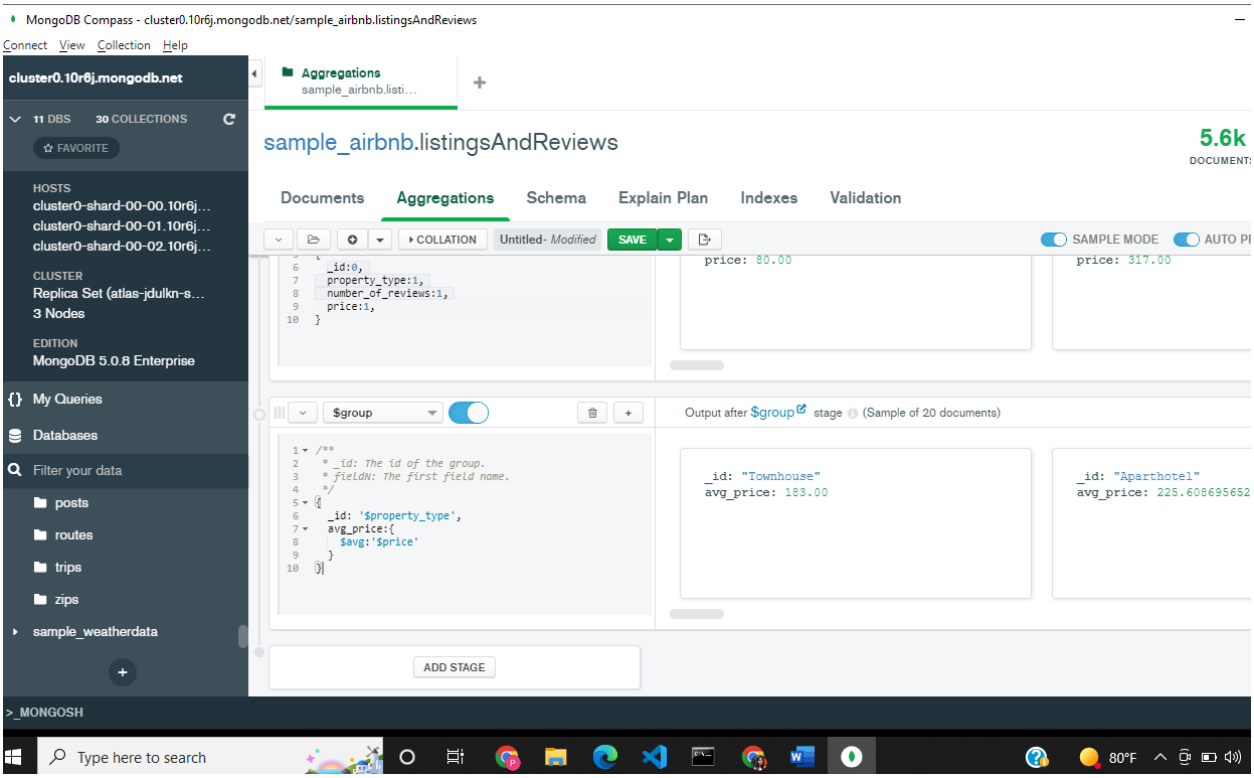
Further , **a $match** is added to next stage to query the collection of documents, an input in the second stage is the output from the previous stage. $match filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage. Hence the filter query was made to filter user ‘name’ -Gary Nichols, and the below output shows only one document that ‘name’ gary nichols



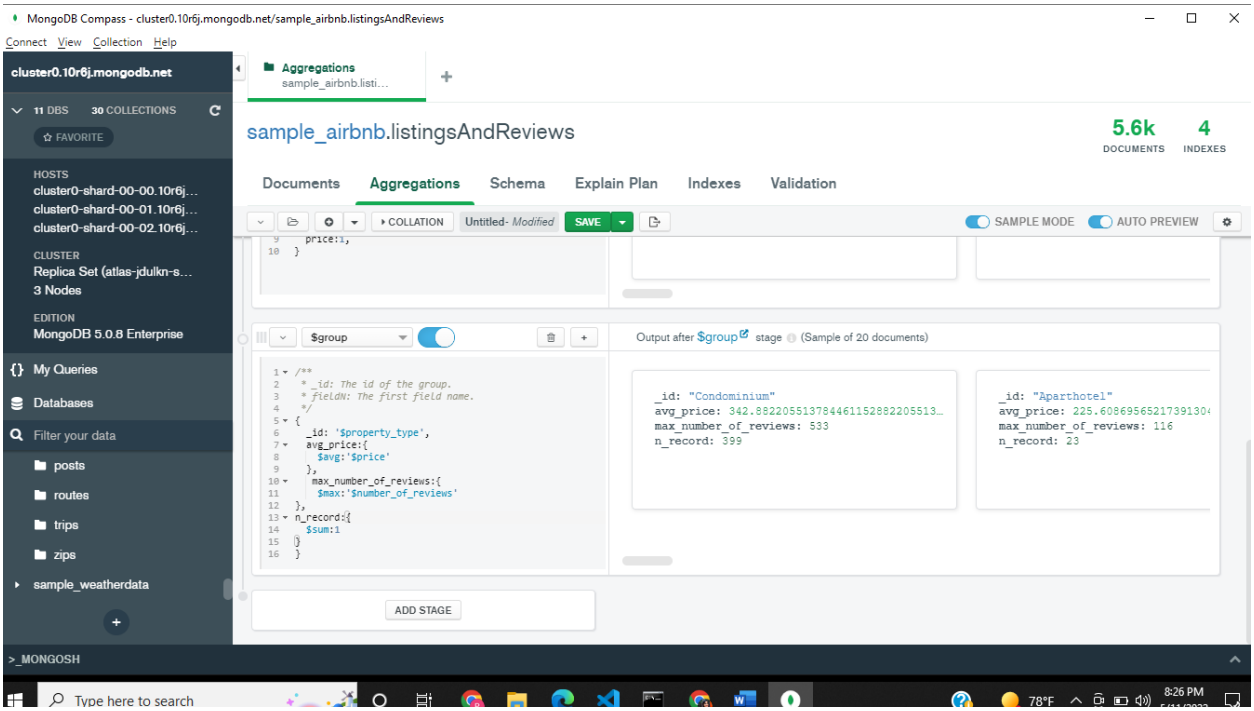
Query to filter transaction_count>70



$group to specify _id expression i.e. property type for **Airbnb database** and for each distinct grouping output a document with average of price.



Query filter on documents to find property type with maximum reviews and its average price for stay on Airbnb database.



Approach 2: Node.js application

Check is node is installed using node-v

```
PS C:\Users\priya> node -v
v16.15.0
```

Check if MongoDB node.js driver is installed using npm install mongodb and finally check the version

Node driver will allow us to easily interact with MongoDB database from node.js application. Driver will help us connect with the database and execute the query.

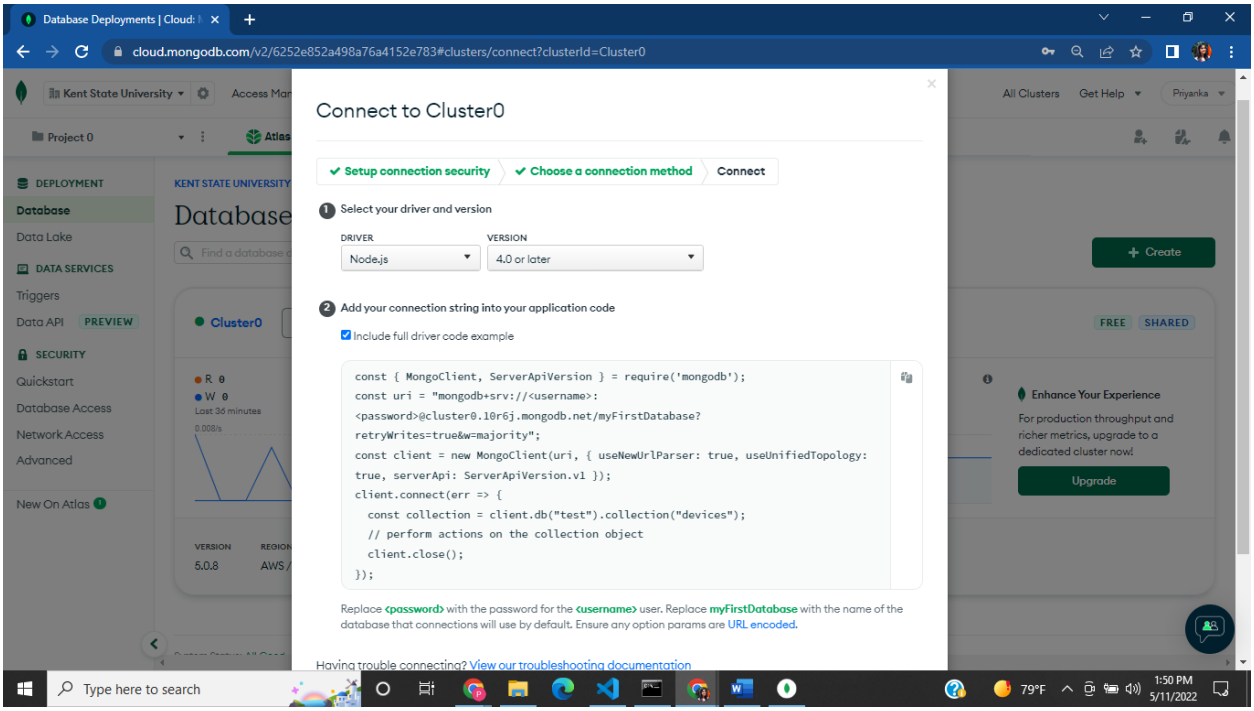
```
PS C:\Users\priya> npm install mongodb

up to date, audited 29 packages in 5s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\priya> npm list mongodb
priya@ C:\Users\priya
├── mongodb@4.5.0
└─┬ mongoose@6.3.3
  └── mongodb@4.5.0 deduped
```

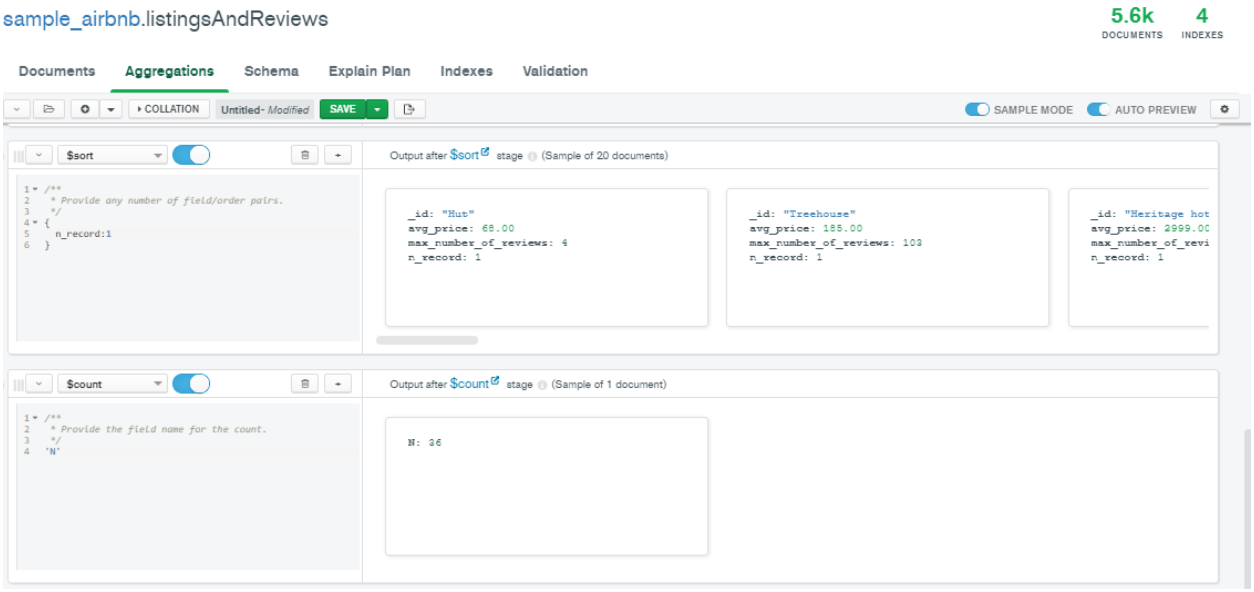
Using Node.js application



Connected to database using the above url and by defining the credential through node.js application and performed few queries on sample database.

Conclusion

Mongodb Compass has interactive interface and is an effective tool, that can be used for querying, analyzing, and optimizing the data in Mongo sample_database. The intuitive GUI is easy to use and through aggregation operation I learned how the input from collection of documents is used to fetch a single output. The query filters the collection of documents as per the specified condition and provides a single output. While querying the Airbnb database which has total 5000+ plus collection of documents in record the aggregation operations reduce it to 36 documents and saved time while querying. This approach does not require writing a lengthy query to perform aggregations. It is simple, easy, user friendly and helps break complex queries into stages. It is also interesting as it allows to verify the whether the query is functioning properly at every stage by examining its input and output simultaneously. Also, the tool does limit the number of stages used in query and how many do I combine them.



Similar, to this approach the Mongodb shell allows to create the aggregate pipeline stages using db.collection.aggregate() command. Data analyzing and abstraction is quick and simplified using these tools leading to better query performance and reduction in execution time to minutes rather than days.

Lastly, while experiencing the querying through Node.js application it can be concluded that node.js also ease the work and offers fast execution speed once the connection to database is established. Many giant organizations like Linkedin, Netflix or PayPal have transformed their products using Node.js. Also, ebay's product 'talk' used Node.js and mongodb to handle its heavy input output operations. JSON data can be handled easily on front end and server end level using MongoDB and Node.js together.

This could help organization save the overall cost and built a system that is economical in terms of operation and maintenance and that can handle so many attributions from one-one to one-many inside a single document. Mongo Db allows to define the ownership and accountability and can manage document-oriented information, store it, and retrieve it. This helps in load balancing, appropriate indexing, extract real-time statistics, split, and replicate datasets for scalability and stability leading to reduction in deployment time.