LAB 2 - SJSU MOVIE DATABASE - REPORT
DATA 225 – DB SYSTEMS FOR ANALYTICS

Submitted by : Gayathri Sundareshwar, Keerthana Gopikrishnan and Deepasha Jenamani

3rd May 2022

SJSU MOVIE DATABASE (SMD) – MONGO DB

PROBLEM STATEMENT

On its online platform, a well-known streaming business seeks to deliver a better experience for their customers. They currently deal with tens of thousands of films and clients. They wish to undertake a study to find areas for development and acquire an analysis based on demographics, traffic, consumer interest, and other aspects that can aid their business before making the platform adjustments. However, this process is difficult because it requires documenting all the customer's viewing histories and recommending movies based on comparable tastes.

The management would like to provide options for customers such as movie recommendations. Because it will allow for improved client connection, providing them with a unique experience that will set it apart from other similar platforms. It's also vital to keep information up to date and efficient when a company wishes to grow into new markets or partner with other industries. A system for recording and storing resources is required to achieve this goal. As a result, to construct this, a database management system solution would be incredibly beneficial.

BUSINESS REQUIREMENT

In order to construct an effective model, the business requirements must be identified. The requirements of the company are as follows:

1. Keeping track of each movie and its corresponding information, tagging it to a unique ID field.
2. Keeping track of every movie's title, genre, description, and other pertinent information.
3. Preserving customer details.
4. Keeping an automated monthly billing system in place for all the valid customers.
5. Tracking and handling customer complaints.
6. Maintaining a record of each user's activities such as their watch history.
7. Visualizing data based on factors such as user activities, demographics, and customer complaints.
8. Analyzing and comprehending profit factors.
9. Generating monthly subscription data and movies watched.

SPECIFICATIONS

Software, Packages and Libraries used in this implementation are:

1. Mongo DB Atlas connected with Mongo Shell and Mongo Compass,
2. Database - Mongo DB Atlas,
3. Mongo Charts, Seaborn and Matplotlib for visualization.
4. Jupyter Notebook
5. Pymongo, pymongo[srv], pixiedust, pixiedust_node installations in Jupyter

SOLUTION REQUIREMENT

There are some rules which were established in the business requirement, and there are others which were not defined explicitly. Some of those are:

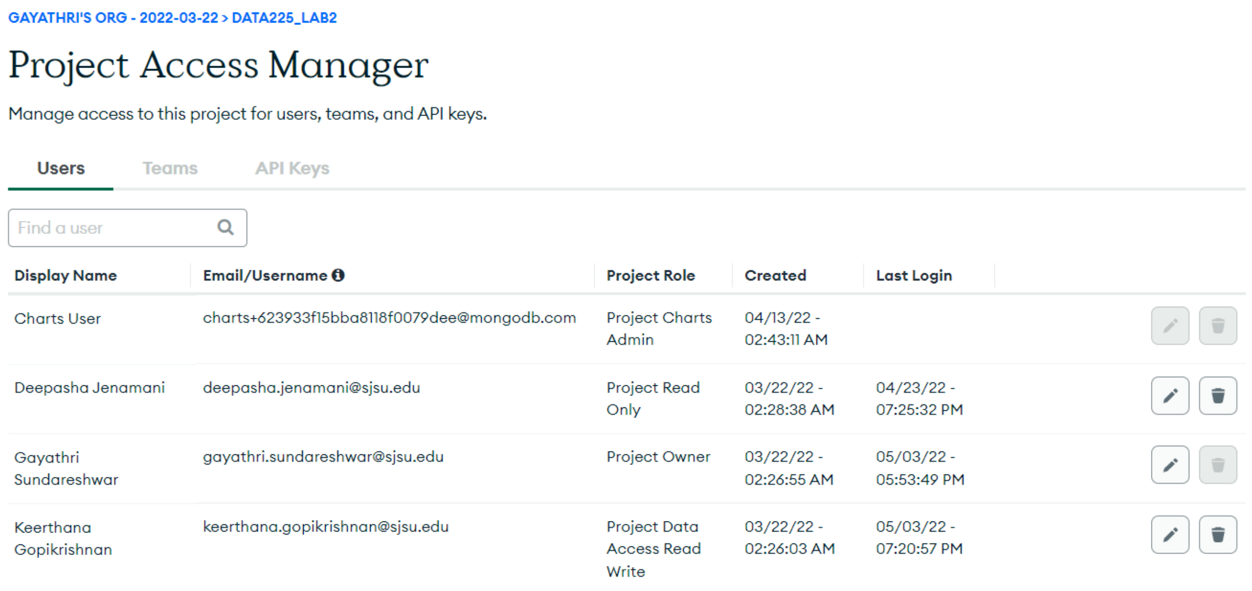
- The subscription plan can only be shared among five family members, according to the business rules.
- The subscription plan fee should be a fixed amount of $19.95.
- Any future date should be rejected by the user's watch history.
- If there is an open complaint, the estimated resolution date should be in the future. And if, on the other hand, the ticket is resolved, the estimated resolution date and close date should not be in the future.

While loading the data we must make sure all these conditions are met. Along with this we also must make sure the data that we load is denormalized and doesn't contain any relationship requirement.

The table 1 and figure 1 below categorizes the on the different types of users who are involved in this project.

Username	Email ID	Project Role
Gayathri	gayathri.sundareshwar@sjsu.edu	Project Owner
Keerthana	keerthana.gopikrishnan@sjsu.edu	Project Data Access Read Write
Deepasha	deepasha.jenamani@sjsu.edu	Project Read Only

Table 1. Project user access


GAYATHRI'S ORG - 2022-03-22 > DATA225_LAB2
Project Access Manager
Manage access to this project for users, teams, and API keys.
Users
Teams
API Keys
Find a user

Display Name	Email/Username	Project Role	Created	Last Login
Charts User	charts+623933f15bba8118f0079dee@mongodb.com	Project Charts Admin	04/13/22 - 02:43:11 AM	
Deepasha Jenamani	deepasha.jenamani@sjsu.edu	Project Read Only	03/22/22 - 02:28:38 AM	04/23/22 - 07:25:32 PM
Gayathri Sundareshwar	gayathri.sundareshwar@sjsu.edu	Project Owner	03/22/22 - 02:26:55 AM	05/03/22 - 05:53:49 PM
Keerthana Gopikrishnan	keerthana.gopikrishnan@sjsu.edu	Project Data Access Read Write	03/22/22 - 02:26:03 AM	05/03/22 - 07:20:57 PM

Figure 1. Project Access Manager

Certain business requirements must be followed while establishing this model, such as just one customer having an account that may be shared by a maximum of five family members. Customers must join up for a $19.95 per month membership plan. Only one individual may be the owner of a subscription.

DATABASE ANALYSIS AND DESIGN

Conceptual Model

Figure 2 depicts the conceptual model of the SJSU movie model that we implemented in My SQL. The conceptual model depicts the entities that have been identified as well as the relationships that exist between them.

The data is denormalized in Mongo DB, and all the different tables have now to be combined into a single collection containing all the relevant fields. The fields will be discussed in detail in the logical model section of this document.

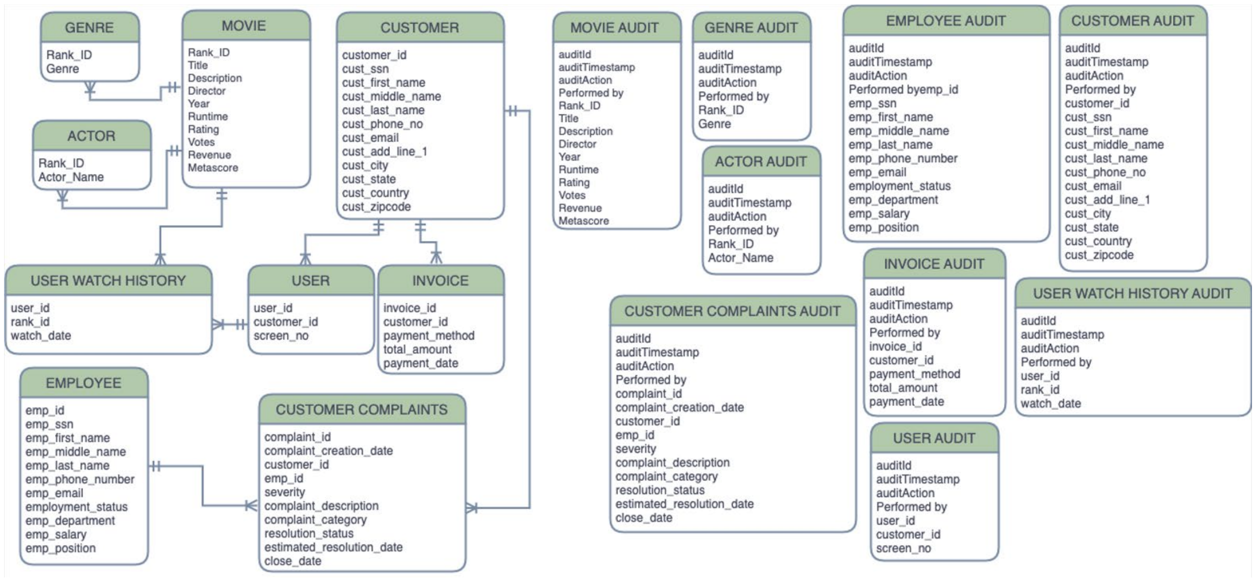


Figure 2. Conceptual Model used in My SQL

Since Mongo DB doesn’t have relations, all the tables and relationships seen above will be denormalized into a single collection. Some of the fields are explained below,

Fields related to Customer details such as customer id, snn, address etc., will store customer information such as name, SSN, and address. This will be followed by the fields related to complaints such as compliant id ,creation date, resolution status. Etc., which will in turn store any data related to complaints raised by a customer, such as account, video, or audio quality issues. A customer may lodge one or more complaints. A new document is created to keep track of each of those complaints. It also keeps track of each ticket's severity level, estimated close date, and actual close date if the ticket has been resolved.

The following up fields will be related to employee’s details such employee id, SSN, phone number which will contain information about employees. Designations such as trainee, junior or senior support associate, tech support lead, or manager will be taken into consideration while creating this model. Each employee can work on a single or multiple open complaints. The collection will also have fields such as rank id, title, rating, etc., which are information about the movie.

The collection will also have fields related to billing and payment. Each customer's payment information is stored in the fields related to invoice such as invoice id, payment method, total amount, etc. The payment method, payment date, total amount, and corresponding customer ID are all included in it.

As per the business requirement, each customer will be granted access to five screens. Any information related to the user's and their screens will also be included in this collection. One customer may have multiple users, but the number should be limited to five to meet business requirements. Movies watched by each of these users will be logged in the fields related to watch history such as watch date. In the fields related to user watch history information regarding each user and movie watched will get collected which can have one or more instances.

Most of these fields are allowed to have multiple instances of the same data. Hence there will be new document with an index Id created whenever a new instance is fed into the model. There is also a possibility of some of the fields being empty, a model should be able to support that as well.

Logical Model

The points made in the conceptual model were taken into consideration with higher degree of importance while initiating the logical model design stage. In the logical model phase, we will now work on deciding the actual fields that will be added when creating the physical model for this project.

Some of fields that are finalized are listed below in figure 3,

Fields to be included in the "SMD" collection					
Customer related fields	**User related fields**	**Movie related fields**	**Payment related fields**	**Employee related fields**	**Complaint related fields**
customer_id	user_id	rank_id	invoice_id	emp_id	complaint_id
cust_ssn	screen_no	Title	payment_method	emp_ssn	complaint_creation_date
cust_first_name	watched_date	Genre	total_amount	emp_first_name	severity
cust_middle_name		Description	payment_date	emp_middle_name	complaint_description
cust_last_name		Director		emp_last_name	complaint_category
cust_phone_no		Actors		emp_phone_number	resolution_status
cust_email		Year		emp_email	estimated_resolution_date
cust_add_line_1		Runtime (Minutes)		employment_satus	close_date
cust_city		Rating		emp_department	
cust_state		Votes		emp_salary	
cust_country		Revenue (Millions)		emp_position	
cust_zipcode		Metascore			

Figure 3.. List of fields in 'SMD' collection

All the above listed fields will be added to the SMD collection that we create. The reason is because since Mongo DB doesn't support relations. All the data will be denormalized down to single collection with no relationships.

Denormalization

Previously we have tired creating a database for similar scenario in My SQL. During that period, we had to have multiple tables with so many different constraints on each of those tables. The data cleansing and ETL occupied a major portion of the time allocated for this task. While querying as well, complex joins have to be performed.

Whereas, in Mongo DB, the relationships are non-existent. Hence , the need for complex joins are eliminated and the time taken for ETL and cleaning the data is practically reduced

All the fields which were previously in separate tables in My SQL, are now combined into a single collection while loading in Mongo DB. In layman terms, this can be explained as, all the customer details, employee details, movie details will now be present in a single collection. The pivoting point of the collection will be customer id. Each unique customer, and that customer's respective payments, users, watch history will be stored as a new document with a brand-new index id, every time a new input has been made.

Physical Model

The logical model shown above serves as the foundation for the physical model. Using Mongo DB cloud deployment, a database called data225_lab2 is created and deployed in the cloud. Figures 4 illustrate the connection that has been created in Mongo DB Atlas.

Figure 5 depicts the collection that has been created in the database. The database and collection were accessed through compass.

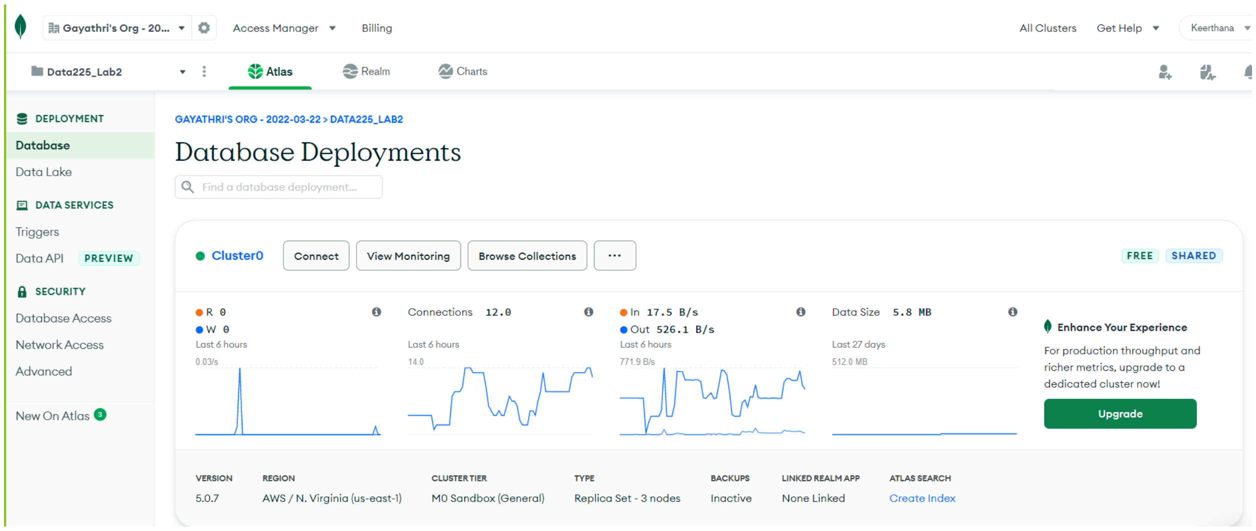
Figure 6 depicts the same collection but accessed through Mongo Shell.

The idea discussed in the logical and conceptual model will be brought to life in the physical model. The screenshots provided will give a better understanding on the structure and design of how the data was denormalized.

The executions will be performed in different platforms listed below.

- Mongo Cloud Deployment (create cluster and connect to shell and compass)
- MongoDB Compass (Importing the csv file into created collections and running aggregation stages)
- Mongo DB shell (performing execution of scenarios)
- Mongo Charts (Visualization of the queried scenarios)
- Jupyter Notebook (to connect mongo db with python)
- Matplotlib and seaborn (for visualizations in python)

Mongo Cloud Deployment



R 0 W 0 Last 6 hours 0.03/s

Connections 12.0 Last 6 hours 14.0

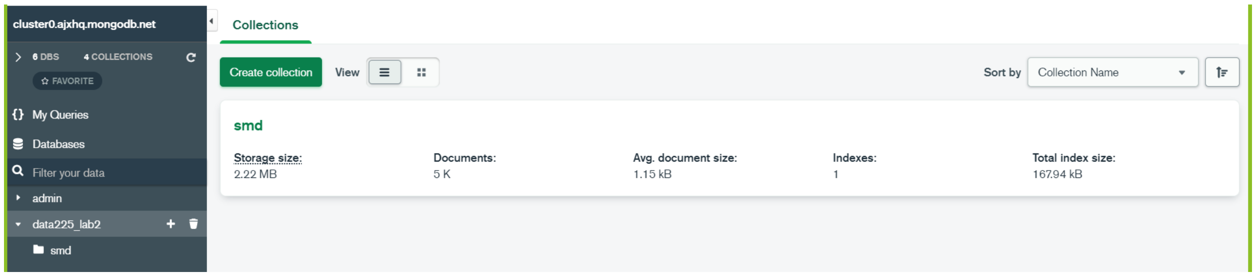
In 17.5 B/s Out 526.1 B/s Last 6 hours 771.9 B/s

Data Size 5.8 MB Last 27 days 512.0 MB

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

Figure 4. Mongo DB Atlas connection

Mongo DB Compass

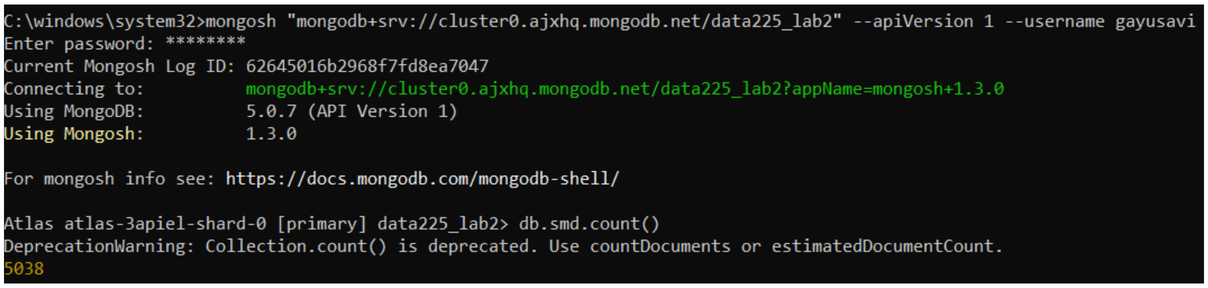


smd

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
2.22 MB	5 K	1.15 kB	1	167.94 kB

Figure 5. Mongo DB compass connection

Mongo Shell



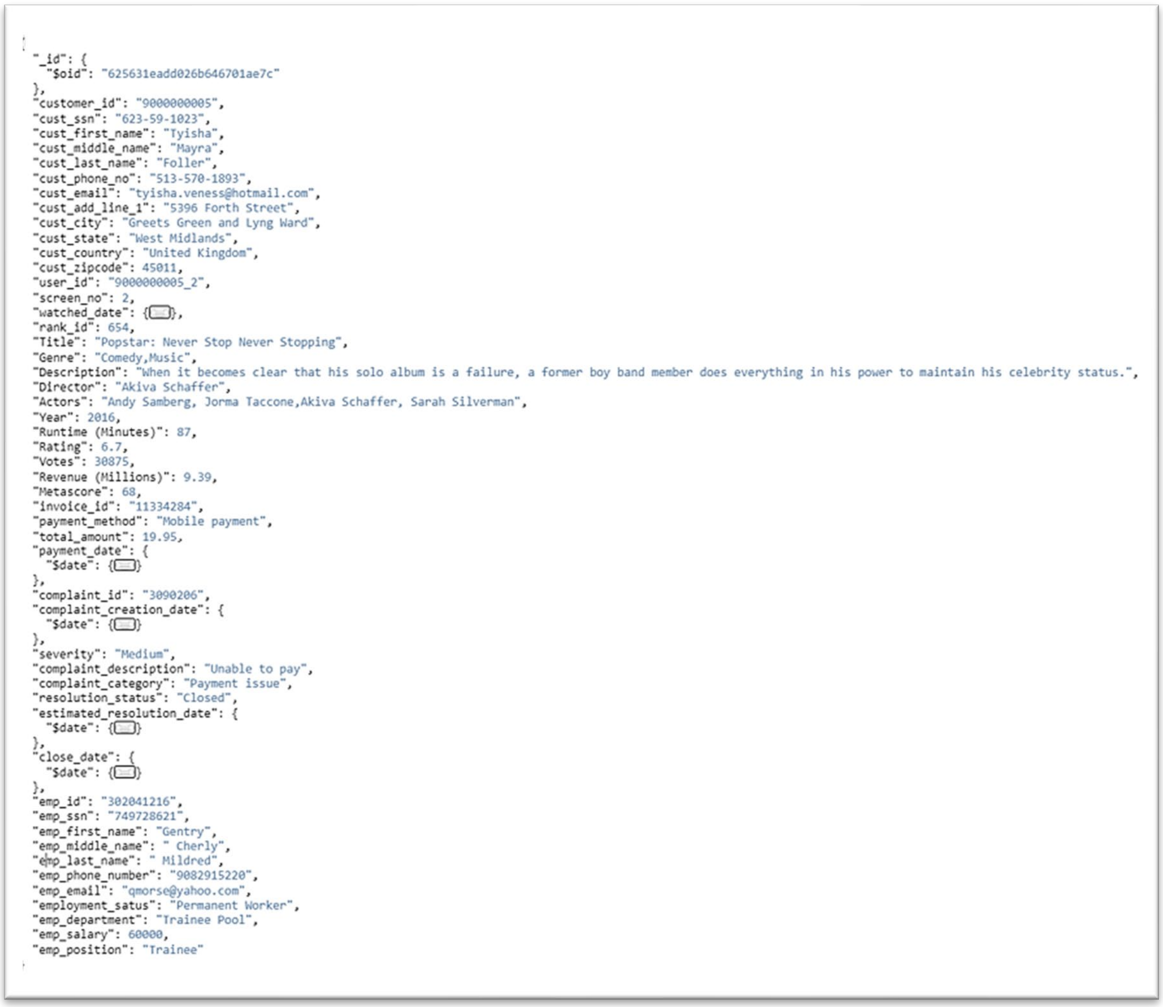
```
C:\windows\system32>mongosh "mongodb+srv://cluster0.ajxhq.mongodb.net/data225_lab2" --apiVersion 1 --username gayusavi
Enter password: ********
Current Mongosh Log ID: 62645016b2968f7fd8ea7047
Connecting to:          mongodb+srv://cluster0.ajxhq.mongodb.net/data225_lab2?appName=mongosh+1.3.0
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.0

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

Atlas atlas-3apiel-shard-0 [primary] data225_lab2> db.smd.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
5038
```

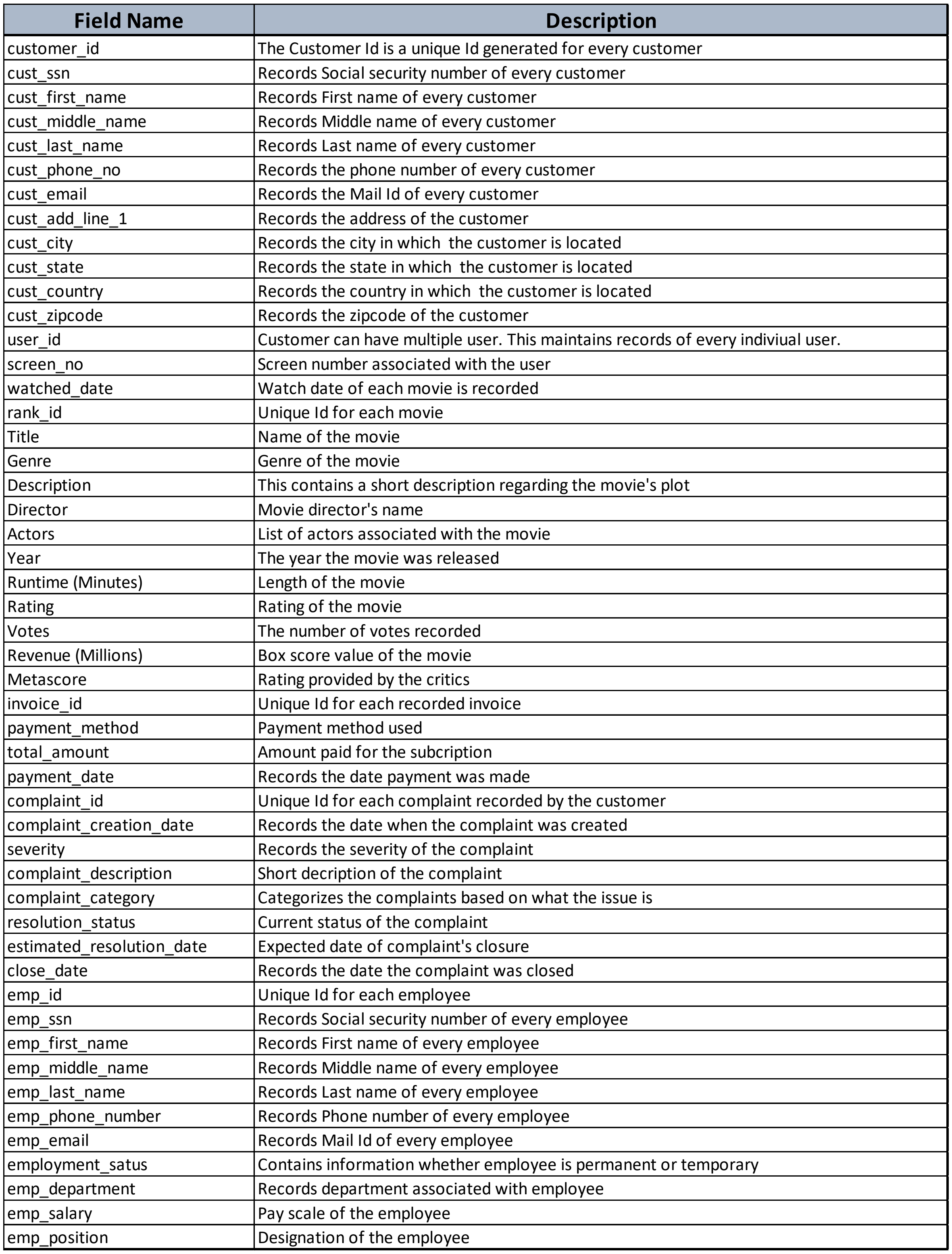
Figure 6. Mongo DB Shell connection

Since the connection, database and collection are created. The next step would be to create the fields to be added to the collection. Figure 7 depicts a sample document created using the columns depicted in figure 8 which contains all the fields along with a short description. Every document will contain fields from the list depicted in figure 8.



```
{
  "_id": {
    "$oid": "625631eadd026b646701ae7c"
  },
  "customer_id": "9000000005",
  "cust_ssn": "623-59-1023",
  "cust_first_name": "Tyisha",
  "cust_middle_name": "Mayra",
  "cust_last_name": "Foller",
  "cust_phone_no": "513-570-1893",
  "cust_email": "tyisha.veness@hotmail.com",
  "cust_add_line_1": "5396 Forth Street",
  "cust_city": "Greets Green and Lyng Ward",
  "cust_state": "West Midlands",
  "cust_country": "United Kingdom",
  "cust_zipcode": 45011,
  "user_id": "9000000005_2",
  "screen_no": 2,
  "watched_date": {…},
  "rank_id": 654,
  "Title": "Popstar: Never Stop Never Stopping",
  "Genre": "Comedy,Music",
  "Description": "When it becomes clear that his solo album is a failure, a former boy band member does everything in his power to maintain his celebrity status.",
  "Director": "Akiva Schaffer",
  "Actors": "Andy Samberg, Jorma Taccone,Akiva Schaffer, Sarah Silverman",
  "Year": 2016,
  "Runtime (Minutes)": 87,
  "Rating": 6.7,
  "Votes": 30875,
  "Revenue (Millions)": 9.39,
  "Metascore": 68,
  "invoice_id": "11334284",
  "payment_method": "Mobile payment",
  "total_amount": 19.95,
  "payment_date": {
    "$date": {…}
  },
  "complaint_id": "3090206",
  "complaint_creation_date": {
    "$date": {…}
  },
  "severity": "Medium",
  "complaint_description": "Unable to pay",
  "complaint_category": "Payment issue",
  "resolution_status": "Closed",
  "estimated_resolution_date": {
    "$date": {…}
  },
  "close_date": {
    "$date": {…}
  },
  "emp_id": "302041216",
  "emp_ssn": "749728621",
  "emp_first_name": "Gentry",
  "emp_middle_name": " Cherly",
  "emp_last_name": " Mildred",
  "emp_phone_number": "9082915220",
  "emp_email": "qmorse@yahoo.com",
  "employment_satus": "Permanent Worker",
  "emp_department": "Trainee Pool",
  "emp_salary": 60000,
  "emp_position": "Trainee"
}
```

Figure 7. Sample records in the collection 'SMD'

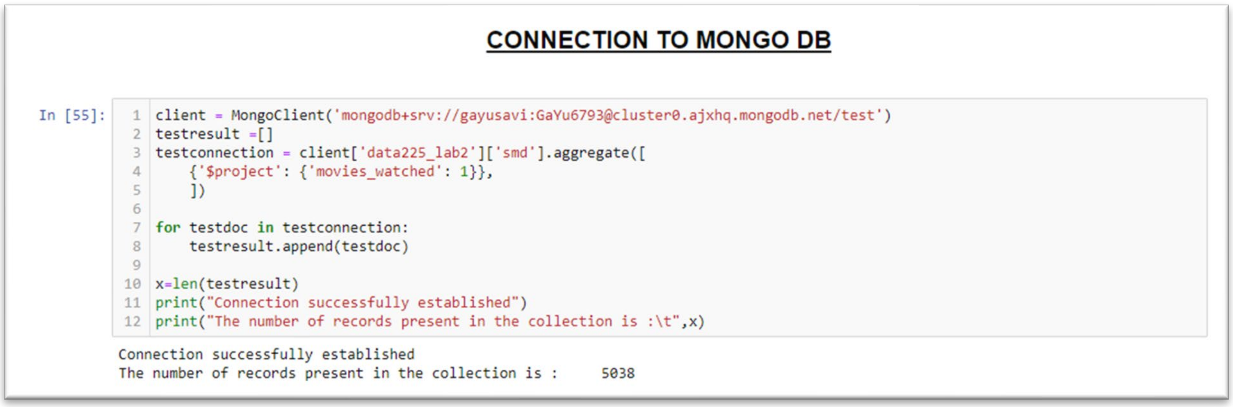


Field Name	Description
customer_id	The Customer Id is a unique Id generated for every customer
cust_ssn	Records Social security number of every customer
cust_first_name	Records First name of every customer
cust_middle_name	Records Middle name of every customer
cust_last_name	Records Last name of every customer
cust_phone_no	Records the phone number of every customer
cust_email	Records the Mail Id of every customer
cust_add_line_1	Records the address of the customer
cust_city	Records the city in which the customer is located
cust_state	Records the state in which the customer is located
cust_country	Records the country in which the customer is located
cust_zipcode	Records the zipcode of the customer
user_id	Customer can have multiple user. This maintains records of every indiviual user.
screen_no	Screen number associated with the user
watched_date	Watch date of each movie is recorded
rank_id	Unique Id for each movie
Title	Name of the movie
Genre	Genre of the movie
Description	This contains a short description regarding the movie's plot
Director	Movie director's name
Actors	List of actors associated with the movie
Year	The year the movie was released
Runtime (Minutes)	Length of the movie
Rating	Rating of the movie
Votes	The number of votes recorded
Revenue (Millions)	Box score value of the movie
Metascore	Rating provided by the critics
invoice_id	Unique Id for each recorded invoice
payment_method	Payment method used
total_amount	Amount paid for the subcription
payment_date	Records the date payment was made
complaint_id	Unique Id for each complaint recorded by the customer
complaint_creation_date	Records the date when the complaint was created
severity	Records the severity of the complaint
complaint_description	Short decription of the complaint
complaint_category	Categorizes the complaints based on what the issue is
resolution_status	Current status of the complaint
estimated_resolution_date	Expected date of complaint's closure
close_date	Records the date the complaint was closed
emp_id	Unique Id for each employee
emp_ssn	Records Social security number of every employee
emp_first_name	Records First name of every employee
emp_middle_name	Records Middle name of every employee
emp_last_name	Records Last name of every employee
emp_phone_number	Records Phone number of every employee
emp_email	Records Mail Id of every employee
employment_satus	Contains information whether employee is permanent or temporary
emp_department	Records department associated with employee
emp_salary	Pay scale of the employee
emp_position	Designation of the employee

Figure 8. . List of fields in the SJSU Database

Python Connection

Figure 9 depicts the connection of python with mondo db in jupyter notebook. Total count of documents present in the collection is displayed as the result.


CONNECTION TO MONGO DB

In [55]:
```python
client = MongoClient('mongodb+srv://gayusavi:GaYu6793@cluster0.ajxhq.mongodb.net/test')
testresult =[]
testconnection = client['data225_lab2']['smd'].aggregate([
    {'$project': {'movies_watched': 1}},
    ])

for testdoc in testconnection:
    testresult.append(testdoc)

x=len(testresult)
print("Connection successfully established")
print("The number of records present in the collection is :\t",x)
```

```
Connection successfully established
The number of records present in the collection is :	5038
```

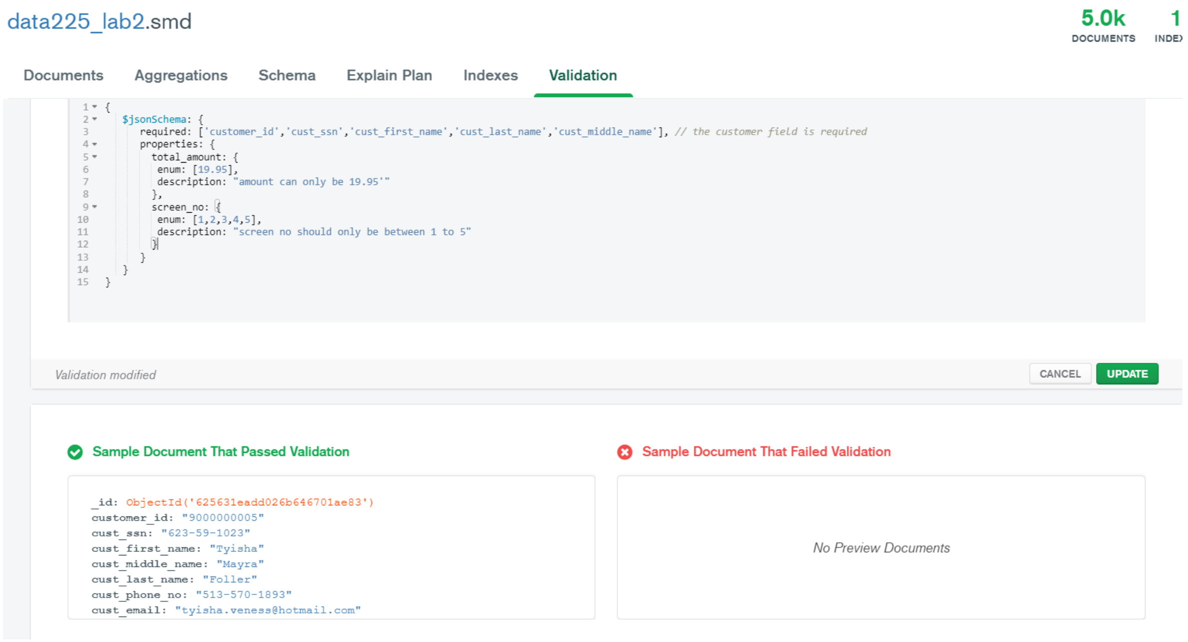
Figure 9. Connecting MongoDB with python in Jupyter notebook

VALIDATION OF BUSINESS RULES AND FEATURES

There are some rules which were established in the business requirement, and there are others which were not defined explicitly. Some of those are:

- The subscription plan can only be shared among five family members, according to the business rules.
- The subscription plan fee should be a fixed amount of $19.95
- Customer id, SSN, first name and last name shouldn't be empty.

These business rules are respected by validating the data using validation rules. Figure 10 depicts the validation rules and it also shows that all the documents that we have inserted into the collection adheres to the business requirements.



```
{
  $jsonSchema: {
    required: ['customer_id','cust_ssn','cust_first_name','cust_last_name','cust_middle_name'], // the customer field is required
    properties: {
      total_amount: {
       enum: [19.95],
       description: "amount can only be 19.95'"
      },
      screen_no: {
       enum: [1,2,3,4,5],
       description: "screen no should only be between 1 to 5"
      }
    }
  }
}
```

Sample Document That Passed Validation

```
_id: ObjectId('625631eadd026b646701ae83')
customer_id: "9000000005"
cust_ssn: "623-59-1023"
cust_first_name: "Tyisha"
cust_middle_name: "Mayra"
cust_last_name: "Foller"
cust_phone_no: "513-570-1893"
cust_email: "tyisha.veness@hotmail.com"
```

Sample Document That Failed Validation

No Preview Documents

Figure 10. Validation rules performed in Mongo DB Compass

IMPLEMENTATION

After importing the data into the created collection, the integrity of the data is tested by querying different scenarios using the data. Visualization of the scenarios were also implemented using Mongo DB Charts. Variety of charts were plotted including bar, line, pie, and geospatial charts. As we execute these scenarios using both Mongo charts visualization and shell queries, we notice that the result returned by both are the same.

In addition to the scenarios tested, more exploratory analysis can be performed if necessary. The execution time of the queries are also measured using the "execution stats".

Queries Performed

The table 2 depicts the list of scenarios implemented, the kind of chart plotted and execution log as well.

These executions are performed in

- Mongo Shell,
- Jupyter Notebook (Connecting Mongo DB to Python)

The visualizations were done using

- Mongo Charts and
- Jupyter notebook visualizations (Seaborn and Matplotlib).

Details logs can be found in the "MongoDB_Query_And_Visualization_Logs.pdf" file.

Sr.No	DESCRIPTION	Plot Type	Execution Logs
1	Number of movies watched per country	Geospatial	MongoDB_Query_And_Visualization_Logs.p
2	Number of movies released per year	Bar	
3	Top 5 employee who have responded to most complaints	Query Result	
4	Number of customers per country	Bar	
5	Number of complaints recorded per complaint category	Bar	
6	Complaints closed per month	Pie	
7	Total complaints per month	Pie	
8	Payment made per month	Line	
9	Top 10 customer based on movies watched	Query Result	
10	Top 10 directors based on user watch history	Bar	
11	Complaints based on Severity	Pie	
12	Average salary based on designation	Bar	
13	Employees per designation based on employment status	Bar	
14	5 latest closed complaints	Query Result	
15	Most preferred screens by customer	Line	
16	20 Highest grossing movies	Bar	
17	Top 5 most rated movies	Bar	

Table 2. Queries performed in the Mongo DB.

PERFORMANCE MEASUREMENT

Monitoring the performance of our database will help us to preemptively handle the possible problems and bugs that could occur in our application before the release. Aside from helping in fixing the issues, it can also be helpful in finding out which is the optimum way to arrive at a solution. A query can be dealt with in multiple ways, but it is always advisable to take the route which is more efficient and less resource consuming.

We have performed performance monitoring for few sample scenarios between My SQL and Mongo DB. The insight gained from those sample scenarios has been helpful to understand Mongo DB's ability to handle large amount of data and also in lesser span of time. Since Mongo DB is denormalized the complexity of joins are basically extinct. This can also be a major factor playing a role in performance measurement between My SQL and Mongo DB.

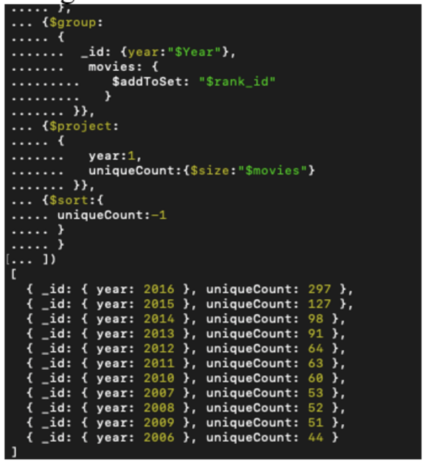
The following scenarios performed both in My SQL and Mongo DB portrays the difference in performance between both. Let us now take a look at them.

Scenario 1

No Of Movies Released Per Year

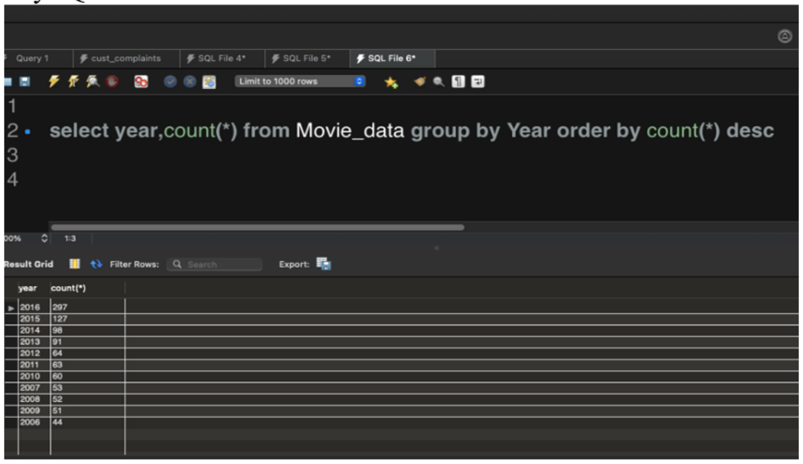
A query has been run on both My SQL and Mongo DB to display the total no of movies released per year. Figure 11 depicts the query ran in both Mongo DB and My SQL workbench.

Mongo DB:



```
..... },
... {$group:
..... {
.......   _id: {year:"$Year"},
.......    movies: {
.........     $addToSet: "$rank_id"
.........    }
....... }},
... {$project:
..... {
.......    year:1,
.......    uniqueCount:{$size:"$movies"}
....... }},
... {$sort:{
..... uniqueCount:-1
..... }
..... }
[... ])
[
  { _id: { year: 2016 }, uniqueCount: 297 },
  { _id: { year: 2015 }, uniqueCount: 127 },
  { _id: { year: 2014 }, uniqueCount: 98 },
  { _id: { year: 2013 }, uniqueCount: 91 },
  { _id: { year: 2012 }, uniqueCount: 64 },
  { _id: { year: 2011 }, uniqueCount: 63 },
  { _id: { year: 2010 }, uniqueCount: 60 },
  { _id: { year: 2007 }, uniqueCount: 53 },
  { _id: { year: 2008 }, uniqueCount: 52 },
  { _id: { year: 2009 }, uniqueCount: 51 },
  { _id: { year: 2006 }, uniqueCount: 44 }
]
```

MySQL workbench:

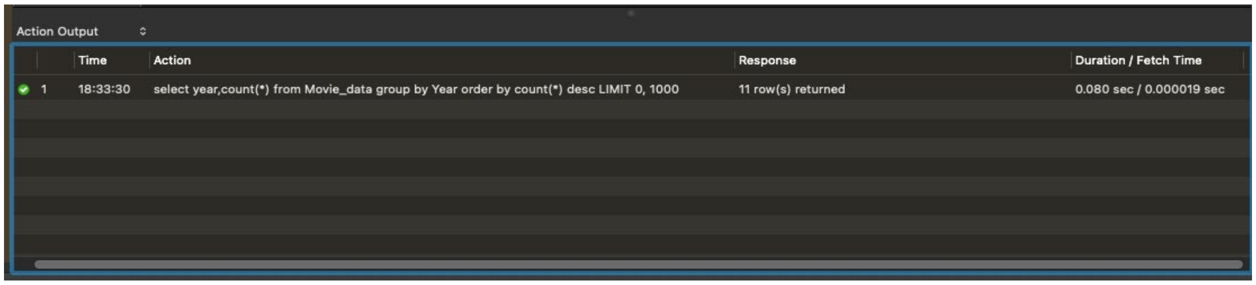


```
select year,count(*) from Movie_data group by Year order by count(*) desc
```

year	count(*)
2016	297
2015	127
2014	98
2013	91
2012	64
2011	63
2010	60
2007	53
2008	52
2009	51
2006	44

Figure 11. Queries in Mongo DB and My SQL for scenario 1

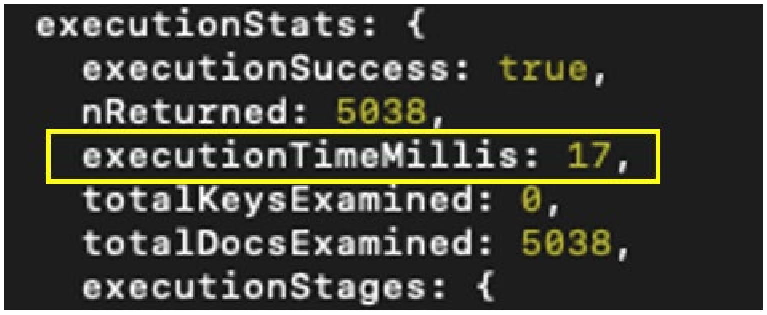
Figure 12 and 13 depicts the time take for the above-mentioned queries to run in both Mongo DB and My SQL. We can observe that MongoDB is faster when compared to My SQL.



Action Output

	Time	Action	Response	Duration / Fetch Time
1	18:33:30	select year,count(*) from Movie_data group by Year order by count(*) desc LIMIT 0, 1000	11 row(s) returned	0.080 sec / 0.000019 sec

Figure 12. Time taken to run in My SQL



```
executionStats: {
  executionSuccess: true,
  nReturned: 5038,
  executionTimeMillis: 17,
  totalKeysExamined: 0,
  totalDocsExamined: 5038,
  executionStages: {
```

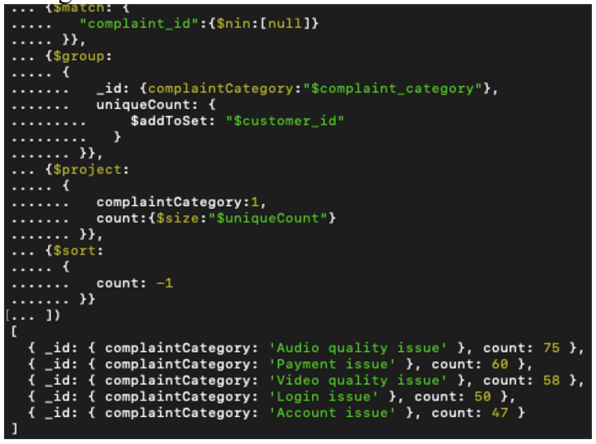
Figure 13. Time taken to run in Mongo DB

Scenario 2

No of complaints recorded per complaint category

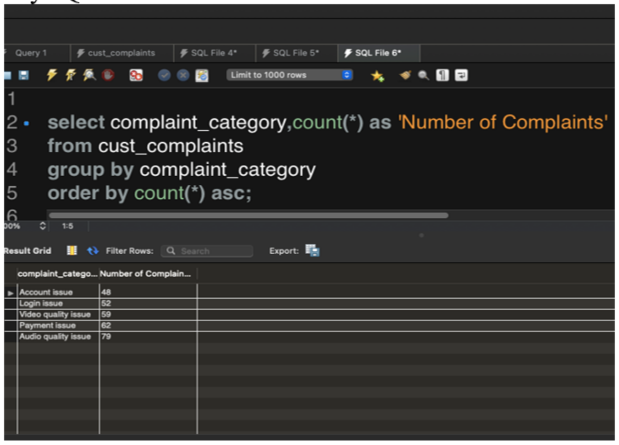
A query has been run on both My SQL and Mongo DB to display the no of complaints recorded per complaint category. Figure 14 depicts the query ran in both Mongo DB and My SQL workbench.

Mongo DB:



```
... {$match: {
.....    "complaint_id":{$nin:[null]}
..... }},
... {$group:
..... {
.......    _id: {complaintCategory:"$complaint_category"},
.......    uniqueCount: {
.........       $addToSet: "$customer_id"
.........     }
....... }},
... {$project:
..... {
.......    complaintCategory:1,
.......    count:{$size:"$uniqueCount"}
....... }},
... {$sort:
..... {
.......    count: -1
....... }}
[... ])
[
  { _id: { complaintCategory: 'Audio quality issue' }, count: 75 },
  { _id: { complaintCategory: 'Payment issue' }, count: 60 },
  { _id: { complaintCategory: 'Video quality issue' }, count: 58 },
  { _id: { complaintCategory: 'Login issue' }, count: 50 },
  { _id: { complaintCategory: 'Account issue' }, count: 47 }
]
```

MySQL Workbench:

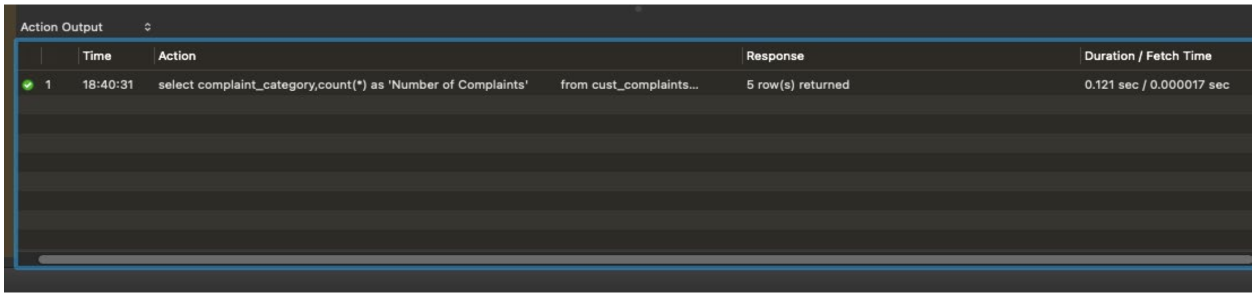


```
select complaint_category,count(*) as 'Number of Complaints'
from cust_complaints
group by complaint_category
order by count(*) asc;
```

complaint_catego...	Number of Complain...
Account issue	48
Login issue	52
Video quality issue	59
Payment issue	62
Audio quality issue	79

Figure 14. Queries in Mongo DB and My SQL for scenario 2

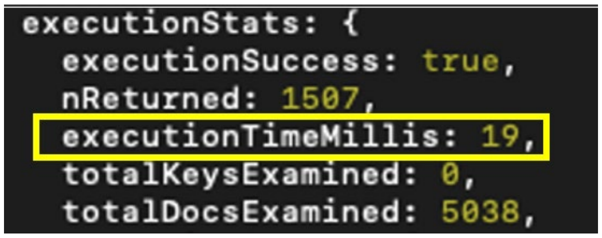
Figure 15 and 16 depicts the time take for the above-mentioned queries to run in both Mongo DB and My SQL. We can observe that MongoDB is faster when compared to My SQL.



Action Output

	Time	Action	Response	Duration / Fetch Time
1	18:40:31	select complaint_category,count(*) as 'Number of Complaints' from cust_complaints...	5 row(s) returned	0.121 sec / 0.000017 sec

Figure 15. Time taken to run in My SQL



```
executionStats: {
  executionSuccess: true,
  nReturned: 1507,
  executionTimeMillis: 19,
  totalKeysExamined: 0,
  totalDocsExamined: 5038,
```

Figure 16. Time taken to run in Mongo DB

From the above scenarios, we shall see that the performance of Mongo DB is better in comparison with My SQL workbench.

FUNCTIONAL ANALYSIS

The application was designed not only to serve as an archive for the movies, the customer , their watch history, and payment methods. But instead, this can help in deriving the knowledge patterns which in turn will provide insights for the business management team to make purchase / management decisions.

Tracking the user watch history can help us in finding out the wavering interest of people in different time periods, for example, listing the top 10 most watched movies can help in finding the most loved genre during that time period. Tracking this over the years can help the management to decide on future copyright purchases for upcoming movies.

The pictorial representation of data aids in better data analysis and faster decision making. This goal is accomplished using data visualization techniques. Plotting the data derived from the query results as visualizations makes it easier for the management team to understand.

Unlike MySQL, Mongo DB doesn't have tables and relations. All the data is denormalized and has been loaded into a single collection as documents. This reduces the time and effort taken to perform complex join queries, and as we have seen in performance measurement, we shall see that the execution time taken by Mongo DB is comparatively less than MySQL. Also, unlike MySQL, Mongo DB also is a better choice when dealing with data with higher scalability

The collection that we have created in Mongo DB consist of multiple fields which are helpful in attaining different insights . For example , Monitoring the complaint creation date, estimated closure and closure date will aid in finding out the category of complaints which take more time to resolve than the estimated date. Knowledge attained from this data can be helpful in determining the vertical where employee needs more training on.

LIMITATION

Even though the model is effective in handling the current business scenario and adequately captures all requirements, it does have a few limitations, which are listed below.

1. It does not encrypt a customer's payment information.
2. Any customer's or employee's Social Security Number (SSN), which is sensitive information, is not encrypted and thus not protected from cyberattacks.
3. Very limited payment methods are available currently, more will be added in future as required when new partnership between platforms emerge.
4. The dataset provided for this project has data only from 2006 – 2016, new data can be added.
5. Email notifications for ticket status & payment are not enabled yet in this version.

SCOPE

The plan is to cover the limitations as parts of future scope in each release.

1. Important information such as Employee and Customer SSN, Address and Phone number can be encrypted using field level AES or DES encryptions.
2. The application created now is accustomed to only movies , this can be further enhanced with series, documentaries, short films, reality shows etc.
3. Email notifications can be implemented for the complaints , payments. Also, audit tables can be set up in a way that the admin can receive timely audit table updates.
4. Customer input regarding a movie they watched can be recorded as user rating, and recommendations can be formulated using this data.
5. Subscription plans can be enhanced with different tiers each having its own set of functionalities . ex, tier 1 with 1080p video and 5 screens and tier 2 with 4k videos and 7 screens.

REFERENCES

Dataset - https://www.kaggle.com/PromptCloudHQ/imdb-data

PROJECT TEAM

Gayathri Sundareshwar , Keerthana Gopikrishnan, Deepasha Jenamani