CS 779 Term Project Fall 2020

Abstract

This purpose of this project is to explore an advanced database concept and build upon it with our current knowledge as well as knew skills learned through the process of exploration. The concept for this project is MongoDB, a type of NoSQL database. The concept will be built on with prior skills of using the object-oriented programming language Python as well as the new skills of exploring data storage and data analysis. The data being worked with is from the Million Songs Dataset project and the Taste Profile dataset by The Echo Nest.

Christopher Pellegrino

cp1992@bu.edu

Table of Contents

[Introduction 1](#_Toc58759668)

[Area of Focus 1](#_Toc58759669)

[Figure 1: Mapping RDBMS to MongoDB 1](#_Toc58759670)

[Figure 2: Sharded Cluster 2](#_Toc58759671)

[Figure 3: Differences Between RDBMS and MongoDB 2](#_Toc58759672)

[Proof of Concept 3](#_Toc58759673)

[Planned Goals to Learn 3](#_Toc58759674)

[Skills Brought from Other Courses 3](#_Toc58759675)

[Data Used Specifically 3](#_Toc58759676)

[Work Done 5](#_Toc58759677)

[Pre-Requisite Work 5](#_Toc58759678)

[Querying the Data 7](#_Toc58759679)

[Results & Discussion 12](#_Toc58759680)

[Challenges & Overcoming Them 12](#_Toc58759681)

[Real World Application 13](#_Toc58759682)

[Conclusion 14](#_Toc58759683)

[Appendices 15](#_Toc58759684)

[Appendix A 15](#_Toc58759685)

[Bibliography 17](#_Toc58759686)

## Introduction

### Area of Focus

The area of focus for my project is non-relational databases, also known as NoSQL databases. A NoSQL database is a type of database that stores data in a format other than relational tables. They store relationships differently than relational databases, as related data does not need to be split between tables. They instead nest data within a single data structure. The cost of storage has decreased over time (See [Appendix A1](#_A1)_Rise_of)), and data would come in semi-structured, unstructured, and polymorphic types that relational databases were not ideal for. NoSQL gave developers the flexibility needed to better work with these data structures, as the schema did not need to be designed ahead of time to the degree relational databases do.



### Figure : Mapping RDBMS to MongoDB

Relational Database Management Systems (RDBMS) and the NoSQL database MongoDB have similarities in terms of their architecture. As seen in Figure 1, the Tables are equivalent to Collections on MongoDB. The rows of an RDBMS contain the actual records, which is what Documents in MongoDB does. The columns in an RDBMS are equivalent to the Fields in MongoDB. What it does not show is that the Collections are stored in a Cluster, which is a sharded cluster in which servers carry fragments of the data to achieve better performance and handle large data sets (See [Appendix A2](#_A2)_Clusters_and)). See Figure 2 for a visual representation of sharding.



### Figure 2: Sharded Cluster

The MongoDB database can contain multiple collections, and collections can have different types of documents (objects). The collections within can have key-value pair lists or arrays, or nested documents. This shows how MongoDB is flexible by being able to use different datasets that may not share the same schema. Sharding reduces the amount of work each shard handles as well as the amount of data each server needs to store. This allows for faster processing of queries.

|  |  |  |
| --- | --- | --- |
| Figure 3: Differences Between RDBMS and MongoDB | | |
| Difference Between MongoDB & RDBMS | RDBMS | MongoDB |
| Schema | Fixed Schema | Schema-less |
| Transactions | Supports transactions | Compromises on transactions, by giving high availability and partitioning |
| Sharding | No | Yes |
| Query Caching | No query caching | Query caching happens here, which leads to faster access of data |

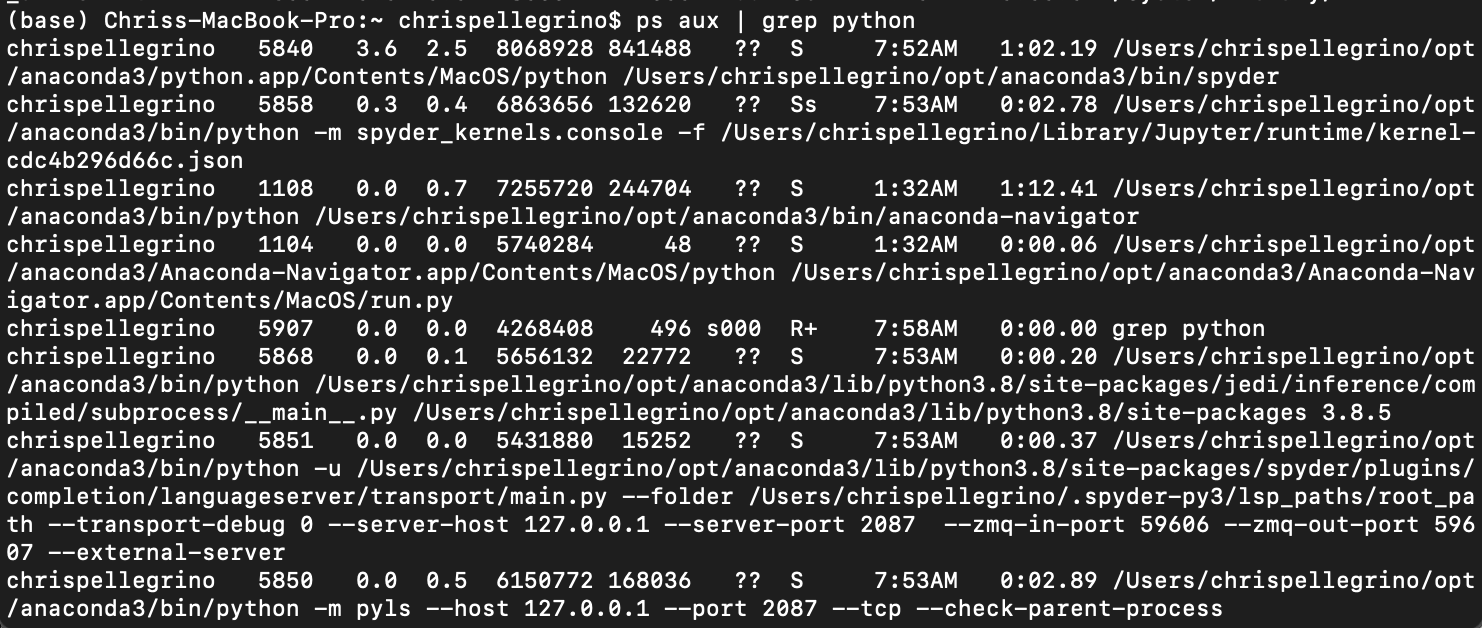
Adapted from: [Schaefer, L. (n.d.).](#_Bibliography)

### Proof of Concept

Proof of concept for this project is to verify the feasibility of using MongoDB to perform data analysis on datasets that do not share the same schema, thus conveying the NoSQL database’s flexibility. This will be done using comma separated value (CSV) files named songs\_data.csv and train\_triplets.txt. General queries, aggregate functions (e.g. sum, average, count, etc.), and transforming the train\_triplets.txt file to a CSV file so that its data can be stored in the Google Cloud. If I had had more time I would’ve learned more about Google’s data warehouse Google Cloud (not the same as Google Drive).

### Planned Goals to Learn

My learning goals for this term project are to learn how to use MongoDB via Mongo Atlas and Mongo Compass. I also planned to learn how to write some BASH script for Terminal (Mac OS command line) which uses UNIX.



### Skills Brought from Other Courses

Since MongoDB provides support for Python, my preferred programming language, I will be using my knowledge from CIT-12:5 Introduction to Python Programming (taken at Bunker Hill Community College) and MET CS 521: Information Structures with Python. I will be building on this skill by using the PyMongo distrubtion with Mongo Atlas.

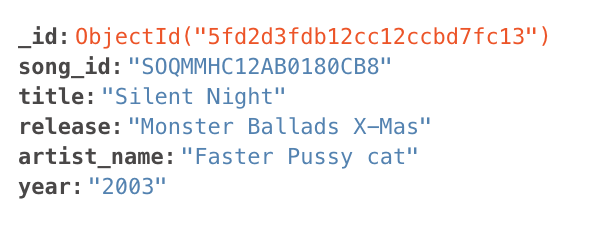
In terms of environment, in CS 521 I used Anaconda distribution for Python that has an application called Spyder, which will be my Integrated Development Environment (IDE) for this assignment.

### Data Used Specifically

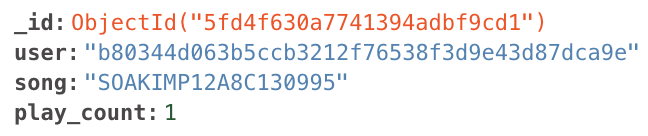
I specifically chose a data set that related to music, not only because of my personal interest in music and the music industry, but also how technology seems to be changing the music industry. A prime example of this is Spotify which uses Cassandra, which is another NoSQL database. Spotify changed the music industry such that now streams are considered more important than actual sales and radio airplay.



My specific datasets are from Million Song Dataset Subset (



File Name: song\_data.csv

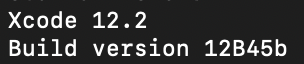


## Work Done

### Pre-Requisite Work

Since I had never used MongoDB (or any NoSQL database) before, there was a significant amount of software that needed to be installed, often through Terminal: Mac OS version of Windows Command Line. The specific tools I needed were:

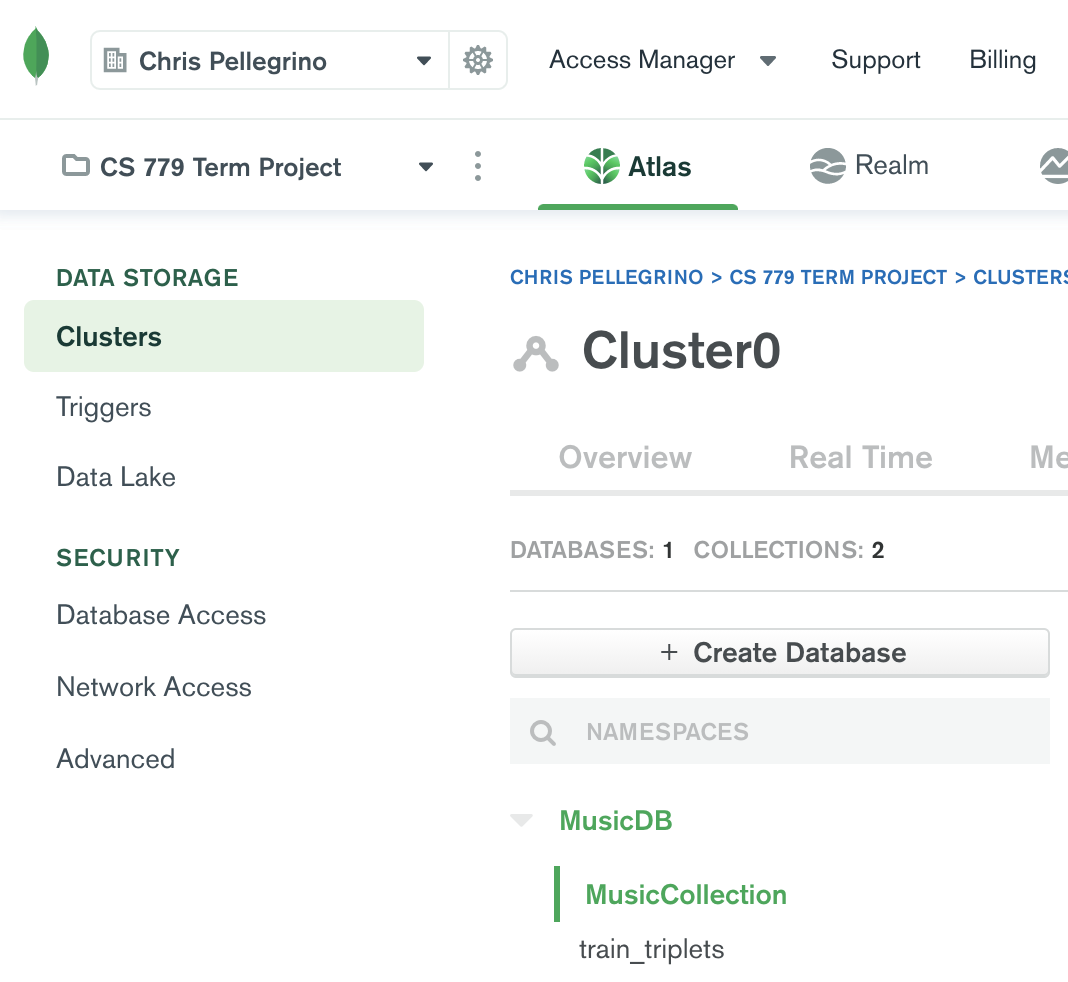
* **XCode**: Apple’s Integrated Development Environment (IDE) for Mac computers



* **Homebrew**: a package manager for macOS and Linux



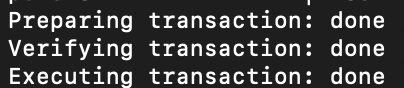
* A **MongoDB Atlas** account (see [Appendix A3](#_A3)_MongoDB_Atlas))



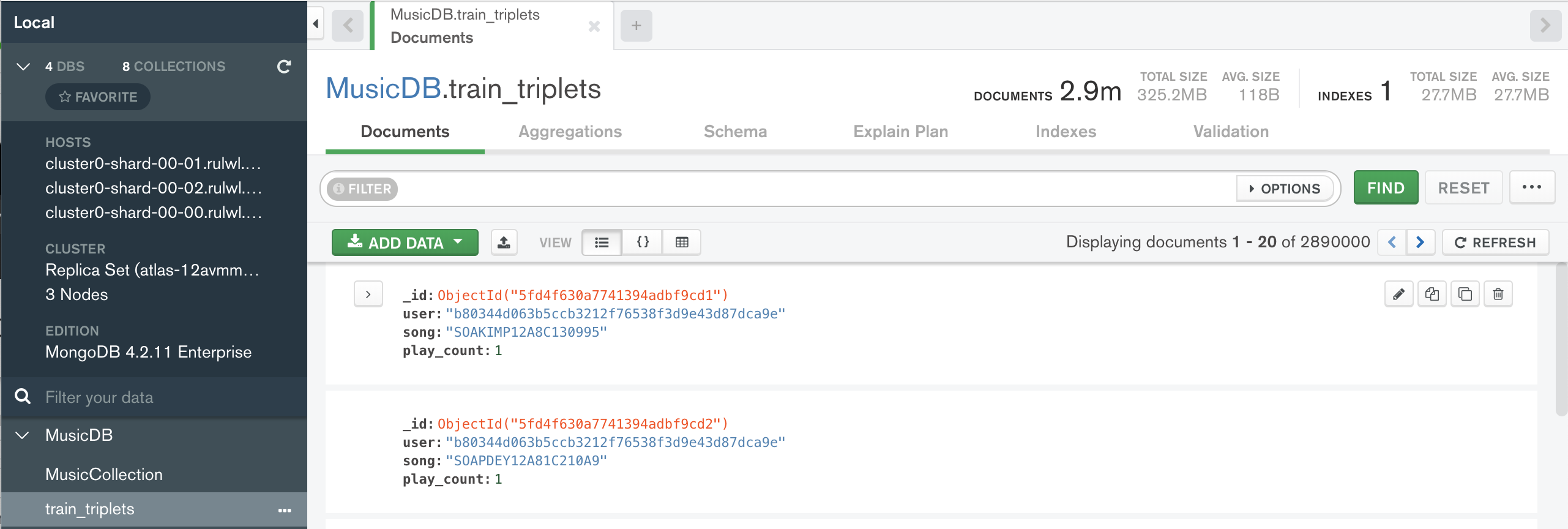
* Updating my **Anaconda/Spyder IDE**

> conda update --all

> conda update spyder



* **MongoDB Tools** (See [The MongoDB Database Tools Documentation. (n.d.)](#_Bibliography))
* **Mongo Compass** which allows for easy import of CSV and JSON files



**Obtaining and Prepping the Data to Import**

The information gathered in the two CSV files were songs that had the formatsong\_data.csv and train\_triplets.txt.

Since Mongo Compass does not accept .txt files. I had to programmatically convert the .txt file to a .csv file using the python extension Pandas:

﻿read\_file = pd.read\_csv (r'/Users/chrispellegrino/Desktop/train\_triplets.txt', delimiter = "\t", names = ['user', 'song', 'play\_count']) #use pandas to read txt

read\_file.to\_csv (r'/Users/chrispellegrino/Desktop/train\_triplets.csv', index = None) #coverting to csv

csv\_file = pd.read\_csv (r'/Users/chrispellegrino/Desktop/train\_triplets.csv')

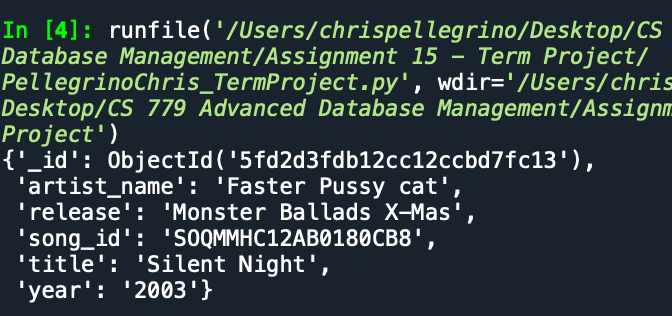
print(csv\_file.head())

### Querying the Data

#### 1) Simple Query

pprint.pprint(song\_data.find\_one({"title": "Silent Night"}))

This code prints the song\_data dataset (MusicCollection) to find one song record that has “Silent Night” as its title. There are many versions of the song within the collection, so this is a way to quickly and simply test it, similar to using SELECT in SQL and then having the WHERE clause be title = “Silent Night” and the ROWNUM = 1.



#### 2) More Complex Query that takes from both datasets

result = song\_data.aggregate([

{

"$match": {

"artist\_name": "Boyz II Men",

"title": "Silent Night"

}

},

{

"$lookup": {

"from": "train\_triplets",

"localField": "song\_id",

"foreignField": "song",

"as": "song\_objects"

}

}

])

print(list(result))

The above PyMongo Code shows user who played the song “Silent Night” by Boyz II Men.



**Aggregate Functions**

artist = input("Enter artist name: ") # asks for input from user

result = list(song\_data.aggregate([

{

"$match": {

"artist\_name": artist #takes input from user and matches it with artist in artist\_name field

}

},

{

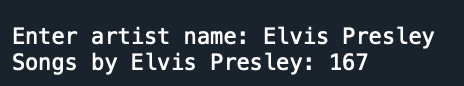
"$count": "field\_name" # counts the number of songs

}

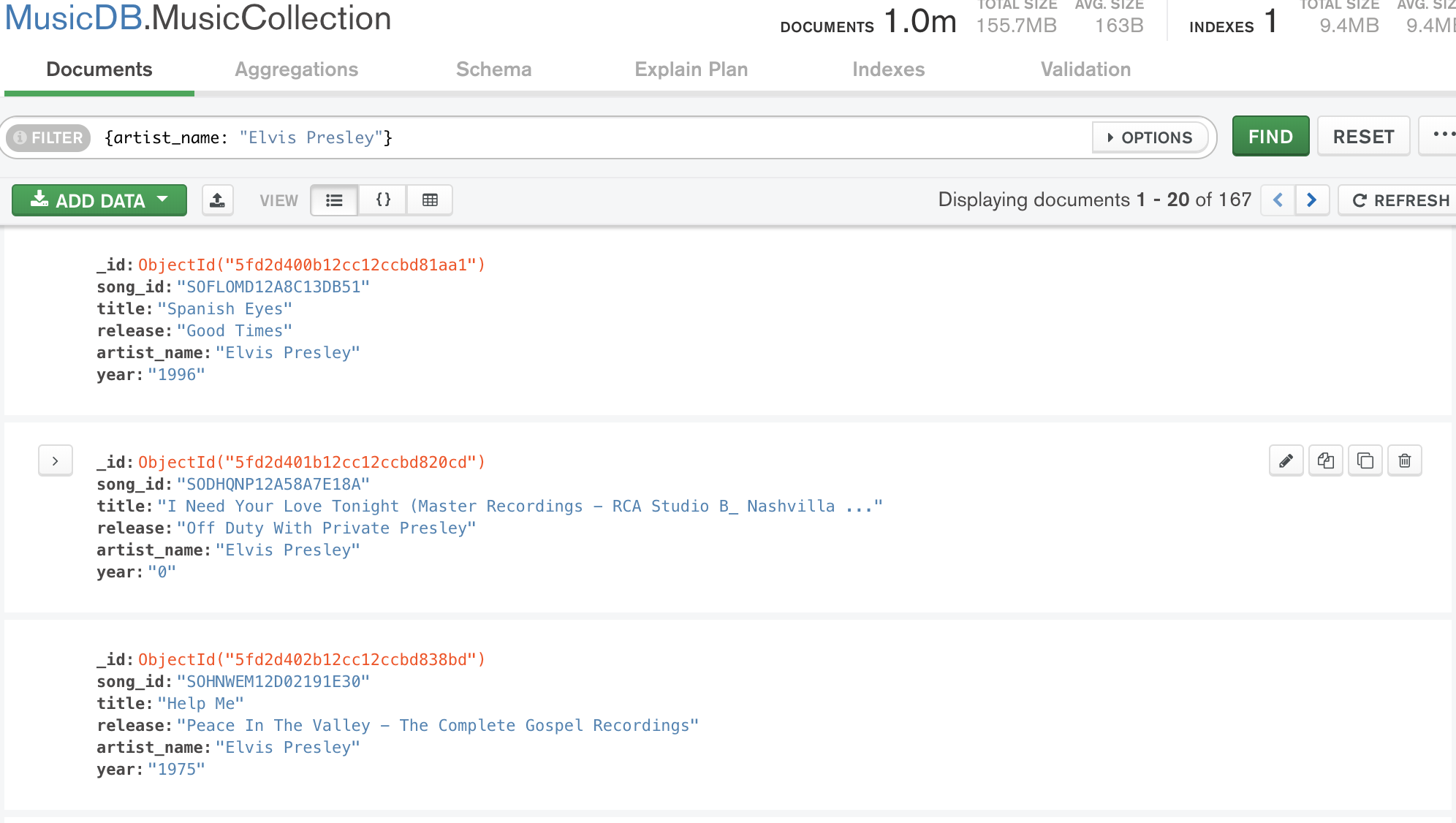
]))

print("Songs by " + artist + ": " + str(result[0]["field\_name"])) #prints it out

The above code takes input from the user of an artist’s name and aggregates the data to count amount of songs for a specific artist.



Use Mongo Compass to see if an artist is in the database



Two ways of doing the same task. One non-aggregate, one aggregate.

Non-Aggregate

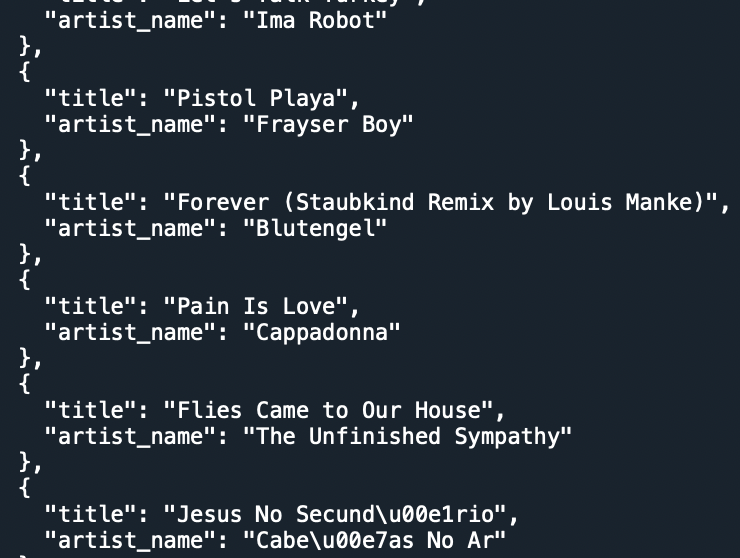
Note: 0 means to exclude the field, 1 means to include it. Everything is 0 by default

Selects songs from 2003 while only outputting the artist name and song name, while purposely omitting the \_id

result = song\_data.find({ "year": "2003" }, { "\_id": 0, "artist\_name": 1, "title": 1 })

items = list(result)

print(json.dumps(items, indent=2))



Aggregate

﻿result = song\_data.aggregate([

{

"$match": {

"year": "2003"

}

},

{

"$project": {

"\_id": 0,

"artist\_name": 1,

"title": 1

}

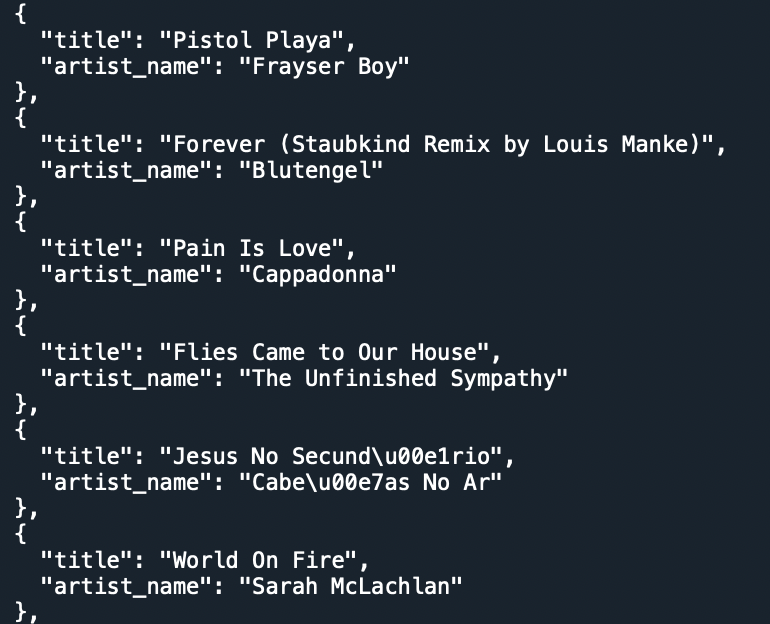
}

])

items = list(result)

print(json.dumps(items, indent=2))

Here, $match works like the find function, as by matching the year to 2003, the results are filtered. The $project part of this code is like the second part of the find function, where certain data can be marked as 0 to be excluded or 1 to be included.



## Results & Discussion

### Challenges & Overcoming Them

There were a significant amount of challenges when building and attempting to do this project.

Simply finding out that I needed XCode to download Homebrew to download MongoDB Tools to use them with PyMongo (which also needed to be installed).

One was I wanted to use mapReduce on the data, however after seeing that it is not available on the free version of MongoDB, this was no longer an option.

var mapFunction1 = function() {

emit(artist\_name, title);

};

var reduceFunction1 = function(artist\_name, title) {

return Array.length(title);

};

db.MusicDB.MusicCollection.mapReduce(

mapFunction1,

reduceFunction1,

{

query: { year: "2003"},

out: "Songs Published In Year"

}

)



Later on when I tried it in PyMongo I realized my datasets (specifically the train\_triplets one from taste\_profile) would cause the execution to time out.

As far as the results of my queries, I was satisfied with the outcomes, although I do feel they took more effort to make than in a traditional relational/SQL database.

### Real World Application

The real world applications for MongoDB/NoSQL in general is that they offer more flexibility to developers using non-structured data. MongoDB works well with PyMongo to help with querying data and data analysis. Once databases become so large (e.g. millions of users on Amazon), having a relational database can be difficult to maintain.

An example would be Shutterfly, an internet-based photo sharing and publishing site that has over 6 billion images. Having a non-relational database improved their programmers’ productivity, performance and scalability of their database to suit their large data set ([Gopinath, 2019](#_Bibliography)).

## Conclusion

This assignment was quite difficult for me as I had no idea about how NoSQL databases worked, let alone how to operate one. There was a very steep learning curve that I struggled with, particularly with just setting everything up in order to even use MongoDB. I would even say that more hours were spent on just preparing to do the project than doing the actual project itself.

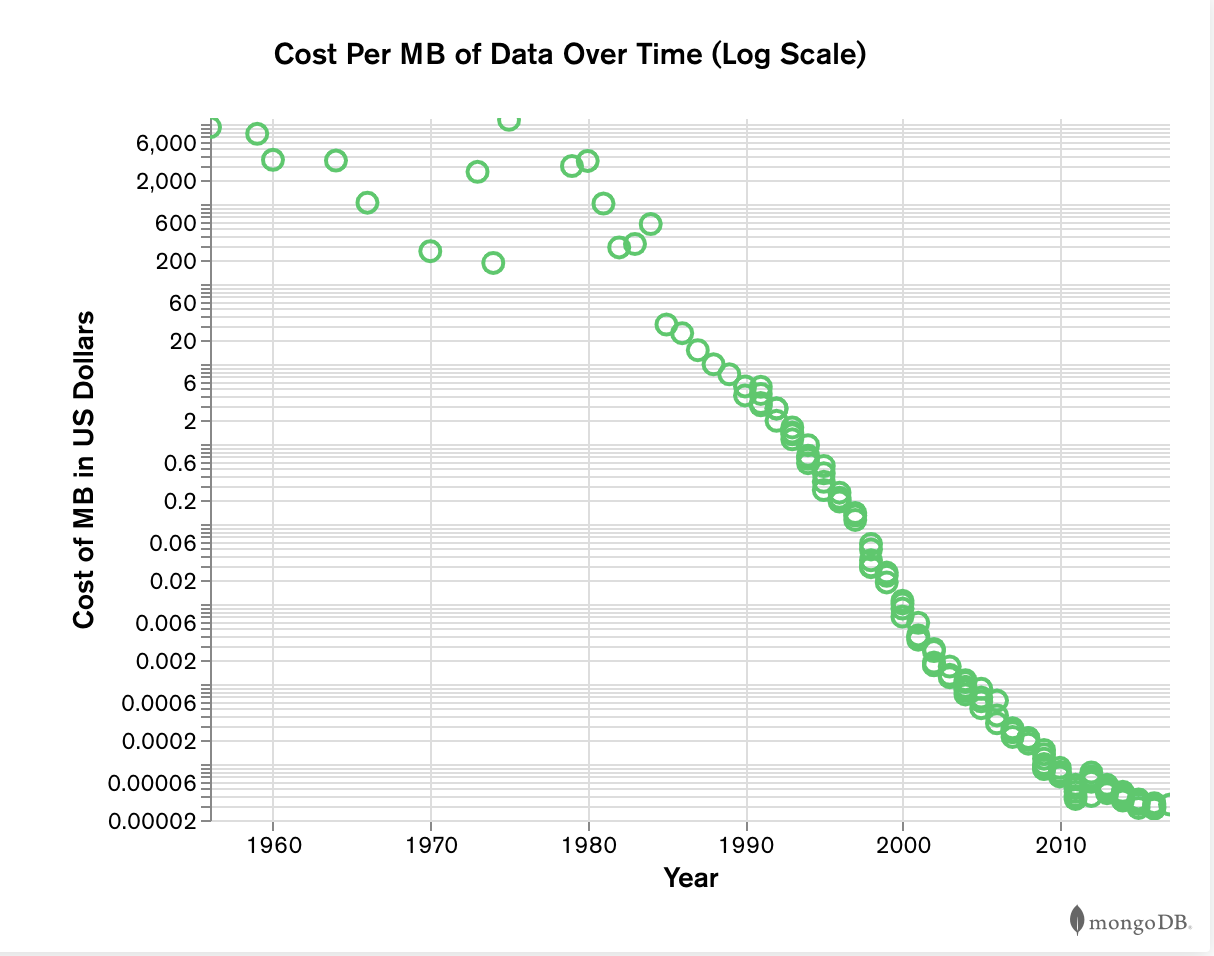
A huge learning curve that was not directly related to the project was having to learn BASH/Unix for Terminal. I followed tutorials and often kept getting errors. I had to install, update, and uninstalled, and reinstall multiple applications. It was extremely frustrating and far beyond anything I had ever learned in terms of working with Mac computers. I do feel that learning command line is important though, as in my career searches on indeed.com,

## Appendices

### Appendix A

#### A1) Rise of NoSQL

NoSQL databases gained popularity as the cost of storage became cheaper and the cost of developers increased.



See ([Schaefer, L., n.d.](#_Bibliography)) for further informatin.

#### A2) Clusters and Replica Sets

A replica set is when several servers carry the same data to prevent failure. Sharded cluster is when multiple servers carry only parts of the data to achieve higher performance. However the word cluster has now come to mean both replica sets and sharded clusters. This is particularly true for MongoDB Atlas. See ([Spiegal, 2019](#_Bibliography)) for more information.

#### A3) MongoDB Atlas

MongoDB Atlas is a cloud database for applications. Its features include full-text search capabilities called Atlas Search, a serverless, Atlas Data Lake: a scalable data lake that removes the need for datamovement, operational overhead, and only charges users for queries run, and a Realm Database: a lightweight database embedded on the client. By storing data on the device itself, users can access it offline. For more information, see

#### A3) Sharding in MongoDB



See [Sharding Introduction. (n.d.)](#_Bibliography) for more information.

#### A4) Million Songs Dataset

From the website:

The **Million Song Dataset** is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

Its purposes are:

* To encourage research on algorithms that scale to commercial sizes
* To provide a reference dataset for evaluating research
* As a shortcut alternative to creating a large dataset with APIs (e.g. The Echo Nest's)
* To help new researchers get started in the MIR field

The core of the dataset is the feature analysis and metadata for one million songs, provided by The Echo Nest. The dataset does not include any audio, only the derived features.

See

## Bibliography

Banerjee, A. (2018, February 22). Million Song Data Set Subset. Kaggle. https://www.kaggle.com/anuragbanerjee/million-song-data-set-subset

Brody, A. (2017, March 9). SQL Vs NoSQL: The Differences Explained. panoply blog. https://blog.panoply.io/sql-or-nosql-that-is-the-question#:~:text=Reasons%20to%20Use%20a%20NoSQL,1.

Choudhury, S. (2019, June 6). How Data Sharding Works in a Distributed SQL Database. yugabyteDB. https://blog.yugabyte.com/how-data-sharding-works-in-a-distributed-sql-database/

MongoDB Cloud. (n.d.). mongodb. https://www.mongodb.com/cloud

The MongoDB Database Tools Documentation. (n.d.). https://docs.mongodb.com/database-tools/

Gopinath, S. (2019, May 22). Real World Use Cases of MongoDB. Edureka!. https://www.edureka.co/blog/real-world-use-cases-of-mongodb/

Schaefer, L. (n.d.). What is NoSQL?. mongodb. https://www.mongodb.com/nosql-explained#:~:text=NoSQL%20databases%20(aka%20%22not%20only,wide%2Dcolumn%2C%20and%20graph.

Sharding Introduction. (n.d.). https://docs.mongodb.com/v3.0/core/sharding-introduction/

Spiegel, A. (2019, September 5). What is a cluster MongoDB. Quora. <https://www.quora.com/What-is-a-cluster-MongoDb>

Theirry, B. et al. (2011). Proceedings of the 12 International Conference on Music Information Retrieval (ISMIR 2011). International Conference on Music Information Retrieval.

Thierry, B. et al., (2011, February 8). Getting the Dataset. Million Song Dataset. http://millionsongdataset.com/pages/getting-dataset/#subset

Thierry, B. et al., (2011, February 8). An Example Track Description. Million Song Dataset. http://millionsongdataset.com/pages/example-track-description/

Thierry, B. et al. (2011, February 8). The Echo Nest Taste Profile Subset. Million Song Dataset. http://millionsongdataset.com/tasteprofile/

Unsupported Commands in M0/M2/M5 Clusters. (n.d.). mongodb. https://docs.atlas.mongodb.com/reference/unsupported-commands

What Is MongoDB?. (n.d.). https://www.educba.com/what-is-mongodb/