Final project report, CS327E

Abel Morales am73272

Uday Dutt ud435

1 Introduction

1.1 Goal

This project aims to implement a database that contains information on different forms of media from a variety of sources. We will explore the data through psql queries to find interesting trends.

1.2 Data Sets

The main dataset used for this class was the IMDB dataset. Along with the IMDB dataset, the movielens dataset and the the-numbers dataset were also explored. For the final milestone, milestone 4, a different dataset called cinemalytics was analyzed. For each of these datasets, a smaller CSV file was downloaded so that we could get a sense of what the data actually looked like. Upon investigation, the IMDB dataset was made up of 10 files, and the Movielens dataset came with 6 files, of which we used 4. The the-numbers dataset came with 36 files, and the cinemalytics dataset which came with 5 files.

1.3 Tools

Instead of using the old-fashioned Microsoft Outlook to run queries and do data analysis, this class uses a vast variety of tools in order to get the student prepared for whatever they will be using once they get a job in the databases field. Among the tools used, Athena on Amazon Web Services was initially used to query the IMDB data.

By the end of the semester, an API known was pyspark was used to do database management. This utilizes the python language to do database work. Amazon Quicksight was used to create visualizations, and Github was used to push/pull files between members. Stache entries were used to turn the projects in.

2 Analysis

2.1 Setup

To start up, the first thing our group did was create a github repository.. For labs one and two, Athena was the sql program that was utilized for the most part. In order to set up Athena, we had to manually load the files into the Athena server using load statements. For projects later on in the semester, psql was the program that was used for the majority of the class. For the milestones, we had to setup an EMR cluster to work on databases using pySpark. All of the required setups for the labs were done either during class time or were done later while following the instructions from the class github snippets page.Other accounts, like lucidchart, were created, but there was not much of a setup process that needed to be done.

One big setup that needed to be targeted was the teamwork aspect of the project. There were some times that one partner would do more work than the other, and vice-versa. For most of the semester though, Uday was the one who wrote the queries and did other small steps like the lucidchart, while Abel did the visualizations and the data loading, since he had it all setup on his desktop back at his home.

2.2 Database Design

Our database ended up being comprised of fifteen different relations, the three main tables beings title_basics, person_basics, and songs. The title_basics table contains the basic information for every imdb title. This includes an identifying id, the type of title it is (such as movie or television episode), the primary and original titles, a boolean identifier to note if it the title is considered adult media, the year it came out, the year it ended (in the case of television shows), and the runtime in minutes. The title_episodes, title_genres, title_ratings, title_tags, and title_financials tables all contain foreign keys which references the title_id primary key in title_basics.

The person_basics table contains basic information on each person, such as their primary name, birth year, death year (if it applies), and gender. There are four intermediate tables between title_basics and person_basics which link the two based on how a person is connected to a title. These four tables are directors, principals, stars, and writers. Lastly, the person_professions table also has a foreign key which references the person_basics table.

The songs table contains an id to identify every song, along with the title of the song, and the duration in minutes. There is an intermediate table between songs and title_basics called title_songs which connects a song to any movie it has been in. Additionally there is a second intermediate table betweens songs and person_basics called singer_songs which connects a song to its artist.

2.3 Data Load

Uploading the data proved to be the most challenging part of every project and milestone. Each project was an exercise in understanding the intricacies of uploading data to a relational database.

Lab two was the first project which involved actually uploading data to the database. The main method we used for loading in lab 2 was the psql /copy command. It should have been as simple as downloading the csv files from the s3 bucket and writing a /copy command with the proper directory, but that was not necessarily the case. The smaller tables like person_basics and title_basics copied without error, but the principals, writers, directors, and stars files would cause the copy command to run for hours before timing out. To be able to upload the data we had to split up the csv files into smaller files so that the copy command could execute without timing out.

Milestone one was the first time we had to use emr clusters and pyspark to upload data. The pyspark program was given to us, so the only problems we ran into involved the amazon emr service. For instance, the cluster would fail to initialize because our default location was set to Ohio and some of the options we needed were only available in N. Virginia. Once these small details were fixed, running the pyspark program became trivial.

The errors that came up in milestones two and three resulted from writing incorrect or improper pyspark code. In order to get our code to work we would have to upload our pyspark script to the master node, attempt to run it, check the errors it gave after crashing, update the script to try to fix the errors, and repeating the process. This method was to ensure that we had the correct file locally so that we could turn it in. One programming error which was hard to fix did not result in an error. The pyspark script would run, but after completion the new table would only have about six entries. The program would execute but it was not executing correctly. The problem, as it turned out, was improper indentation in the loop that executed the update psql command.

It was during milestone four that we ran into a major problem while trying to upload the cinemalytics data. The emr service would successfully copy a previous cluster which worked properly, however, we were unable to connect to the cluster. Some of the data could be uploaded with a simple /copy command straight from a csv file. The challenging part was uploading the data which needed to be processed in some way. The new person_basics data, for example, required processing the date of birth to extract the year from the date and converting it from a string to integer. In order to accomplish this we ended up creating two temporary tables. The data was uploaded from the csv file to the first temporary table as is. Any null values in the date of birth column was replaced with a zero to facilitate the next step. We queried the first temporary table, trimmed the date of birth so that only the last four characters remained, converted it into an integer, and inserted the results into the second temporary table. From the second temporary table we could query and insert the necessary data into person_basics. The remaining two tables, singer_songs and title_songs, were uploaded by copying the related csv files to temporary tables and doing a query/insert into, much like updating person_basics.

2.4 Data Analysis (Queries)

Many queries were created over the course of the semester, and some were better than others. The queries were primarily made in a way that would fit the requirements that were listed on the rubric for that specific lab (some labs state something like ‘must have 2 queries using an outer join’). Second to meeting the rubric’s requirements, the queries covered topics that were interesting in some way.

The first query that was written in this lab was: “find Tom Cruise’s movies, sorted by rating.” At first, even such basic queries were a little bit difficult to do. Our lab partnership had to first understand the relational schema between all the tables in the database, and only then did things begin to make sense. Difficulties that arose during lab 1 were mainly figuring out what should join with what and based on what primary key relationship. After this was identified, then all 20 queries in lab 1 became a breeze.

Later on, in lab 3, we were assigned to do aggregate queries. For the most part, this assignment was about as easy as lab 1. However, it proved to be more difficult at times. For example, sometimes the query would not work because we were missing a group by clause. After understanding what everything meant, there was still one more difficulty that arose during this lab: outer joins. Up to lab 3, we had been using only inner joins. On lab 3, we were forced to use outer joins. This was difficult because sometimes it can be hard to conceptualize in our heads the difference between a table that results from a left outer join and a right outer join. A left join basically is an outer join where the items from the left table is kept, but if there is not a column to go with it from the right table, then it will return a null value, and thus it will be excluded from the query. From this lab, we made a really boring query, but it’s actually a very funny query when visualized: “Find all dead people that were Stars, while alive, in 1 title, and the number of titles they worked in.” Obviously, when we do the actual visualization of this query, we will get a barchart where every bar had length one, because everything else has been filtered out of the query. Although the visualization might not be that interesting, the query itself is still very interesting because it shows all of the people who were stars in one title during their lifetime.

Another query from lab 3 that was interesting was “List the number of principals each profession had in the year 2015.” This query was particularly interesting, because we got to see which role in imdb gets to be the principal the most. We thought that the order would be actor, actress and then director. Surprisingly, the order was actually actor, director, and then actress. This is interesting because it shows that even when selecting the principal of an imdb show, there is still some segregation based on gender, since actors and directors are both male. In fact, in order to support this query, we rand another query that saw the number of principals in the year 2015 based on gender. The results of the query showed that a significantly larger amount of men were made into the principal of the show than women were.

For milestones 1 and 2, we used the movielens dataset to create a query that finds how a movie was rated each year, and comparing the maximum rating received and compare that against the minimum rating received for that year.

For milestone 3, we ended up making a query that used code which was not covered in class at all. Using the the-numbers dataset, a table was created that showed the net earnings based on the type of show it was (Movie, episode series,video game, etc). We wished to find a way to segment the data into the earners that earned a negative amount , a positive amount between 0 and 1 million, between 1 and 2 million, between 2 and 3 million, and then above 3 million. To calculate the earnings, we used a simple formula based off the data that we were given: earning=box office-budget. However, we did not know how to make one column into 4 different columns on SQL. This brought up a very interesting idea into our heads: what if we can use a conditional statement in the select clause? Sure enough, after some more research into the idea, we found out that this was indeed possible. Thus, we were able to segment one column of data into 4 columns by using “SELECT count(case when ___ then 1 end).” This query was probably our most interesting query so far.

2.5 Visualization

Compared to programs meant for data manipulation and visualization like R or MatLab, the Amazon Quicksight was not too friendly in that it did not allow its users to change some aspects of the visualization output. Thus, for lab 3 when we first used the Amazon Quicksight software, we were a bit disappointed with how the visualizations turned out to look like.

There were not many problems that were encountered while using the Amazon Quicksight software. However, it was still annoying to use. For example, when trying to place certain variables on x axis and other variables on the y axis in order to make the visualization look the way you want it to look, sometimes achieving this was difficult. We found out that in order to get a certain variable on the axis you want it to be on, you have to put the variables in a specific order when you input them. Weird, built in features like this made Quicksight a real pain to work with.

Some queries were visualized very well, while many others came out very poorly. Our favorite example of a query with a terrible visualization is “find all dead people that were Stars, while alive, in 1 title, and the number of titles they worked in.” The visualization was just a bunch data on a bar graph with length of one.

One good query was the one over the movie earnings, where we segmented the data into 4 columns using a conditional count statement. The graph looked very good, and was informative. Another graph was a visualization of the query: “Find all IMDB genres with average ratings above 5”. The visualization was mediocre, not good and not bad. The funny thing about this query was that out of the 28 genres, all 28 genres had an average rating of above 5. Thus, the query we used basically had an unnecessary statement in there that did not filter the data at all.

3 Conclusion

In conclusion, the original aim, which was to implement a database that contains information of different forms of media from a variety of sources, was met. Overall, we learned a ton about database architecture through the lectures and labs in this class.

As a group, we progressively got better and better with communication and with getting the work done. We realized that later on in the semester, these databases projects will be too hard to do in one or two nights. We got to a point where we began our labs on Monday, as soon as Dr. Cohen releases the assignment. One struggle our group had to face was that Uday is taking 21 hours this semester, so other classes sometimes interfered with the amount of work he could put into this class. It was even worse for him because we were not given any time over the weekend to work on a lab. Thus, he had to learn how to manage this class's workload during his weekdays.

Additionally, our SQL codes, Quicksight usage, and other databases programming, got much better as the semester progressed. By the end of the semester, in milestone 4, we came up with one aggregate query that used all of the required tables and also showed an interesting insight about song length based on gender. Our Quicksight issues that were encountered earlier in the semester were gone, and the result was a good looking visualization. Through the course of the lectures and labs, we have implemented all of our knowledge to create a big database that can efficiently parse through data, and have successfully created numerous queries and visual insights to go along with them.