Project Deliverable 2: **Data Understanding** a) Exploratory Data Analysis b) Dashboard And **Data Preparation**

Group 4

Members - Dhruv Jani, Dhruvil Patel, Karan Issrani, Mohit Gupta, Yaw Frempong

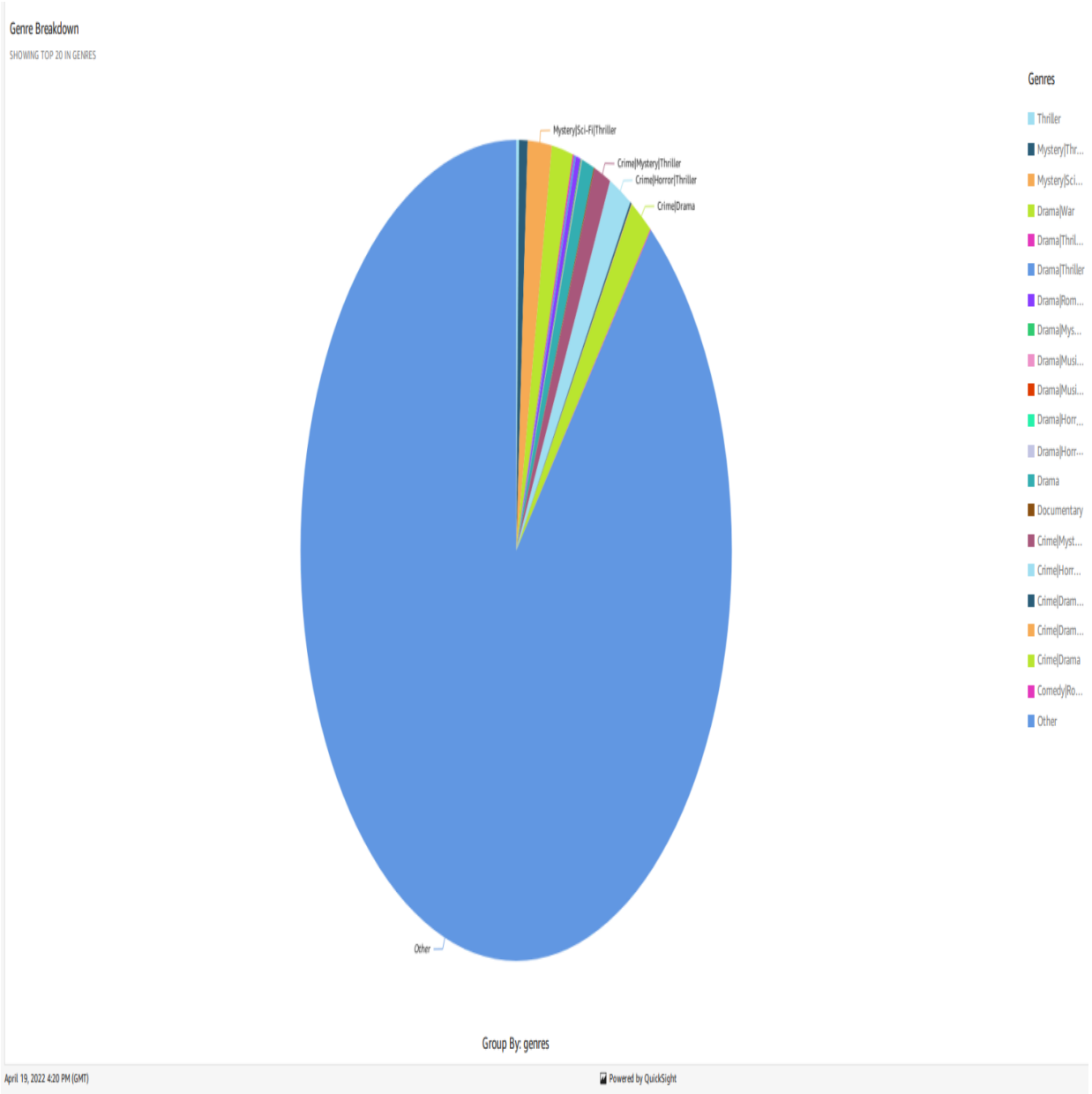
Data Understanding

The Data we have was divided in 4 different files. The link of the data we used is https://files.grouplens.org/datasets/movielens/ml-latest-small.zip
The data is divided into links.csv, movies.csv, tags.csv and ratings.csv

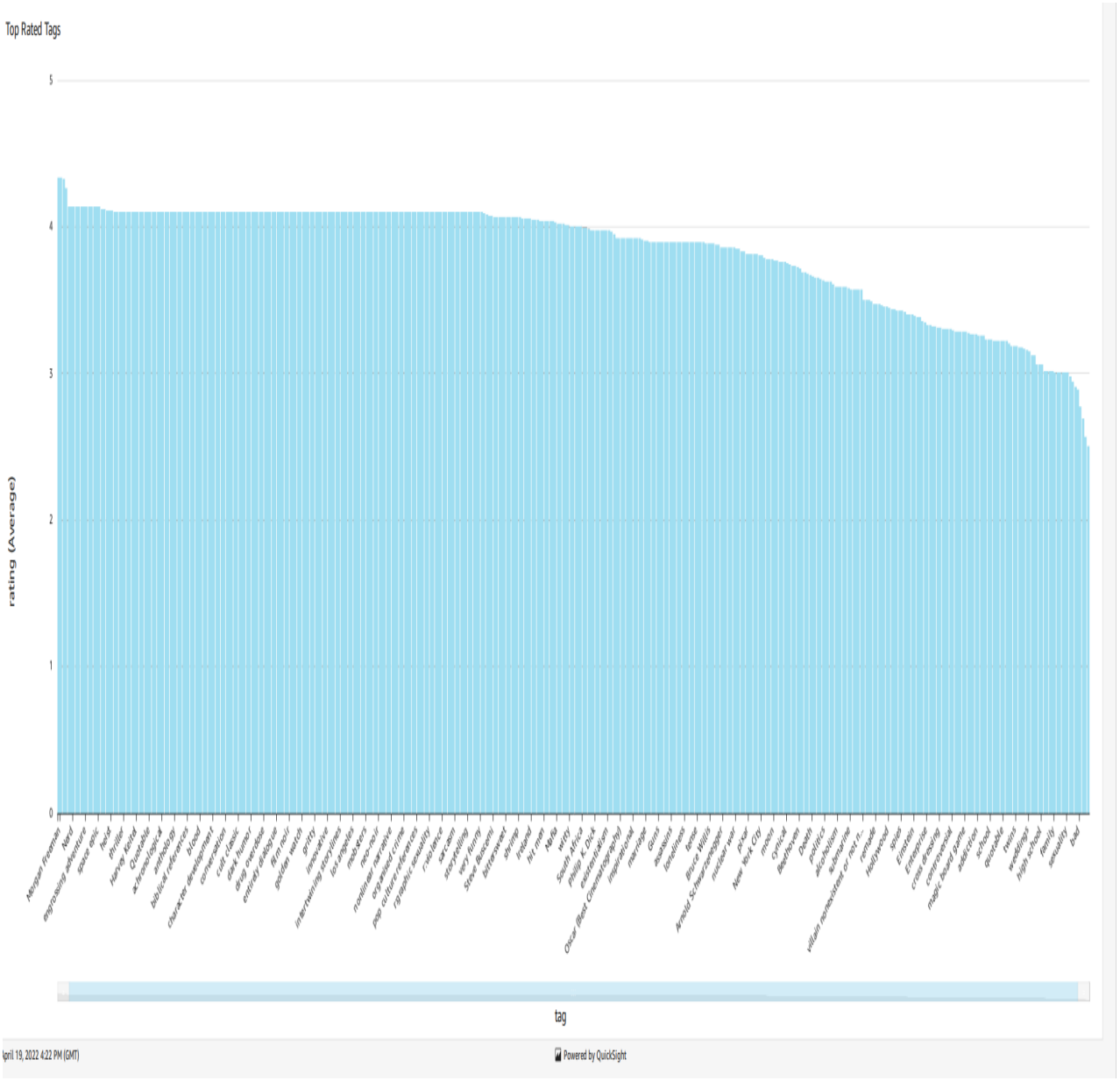
We used the primary key Movie ID to merge these data files and form our data set. Our data set consists of 100000 entries. The dataset merged_data.csv has all the data in it. Our data has movieId,imdbId,tmdbId,title,genres,rating,userId,tag,timestamp,userId column names respectively.

Now we do exploratory data Analysis on our dataset.

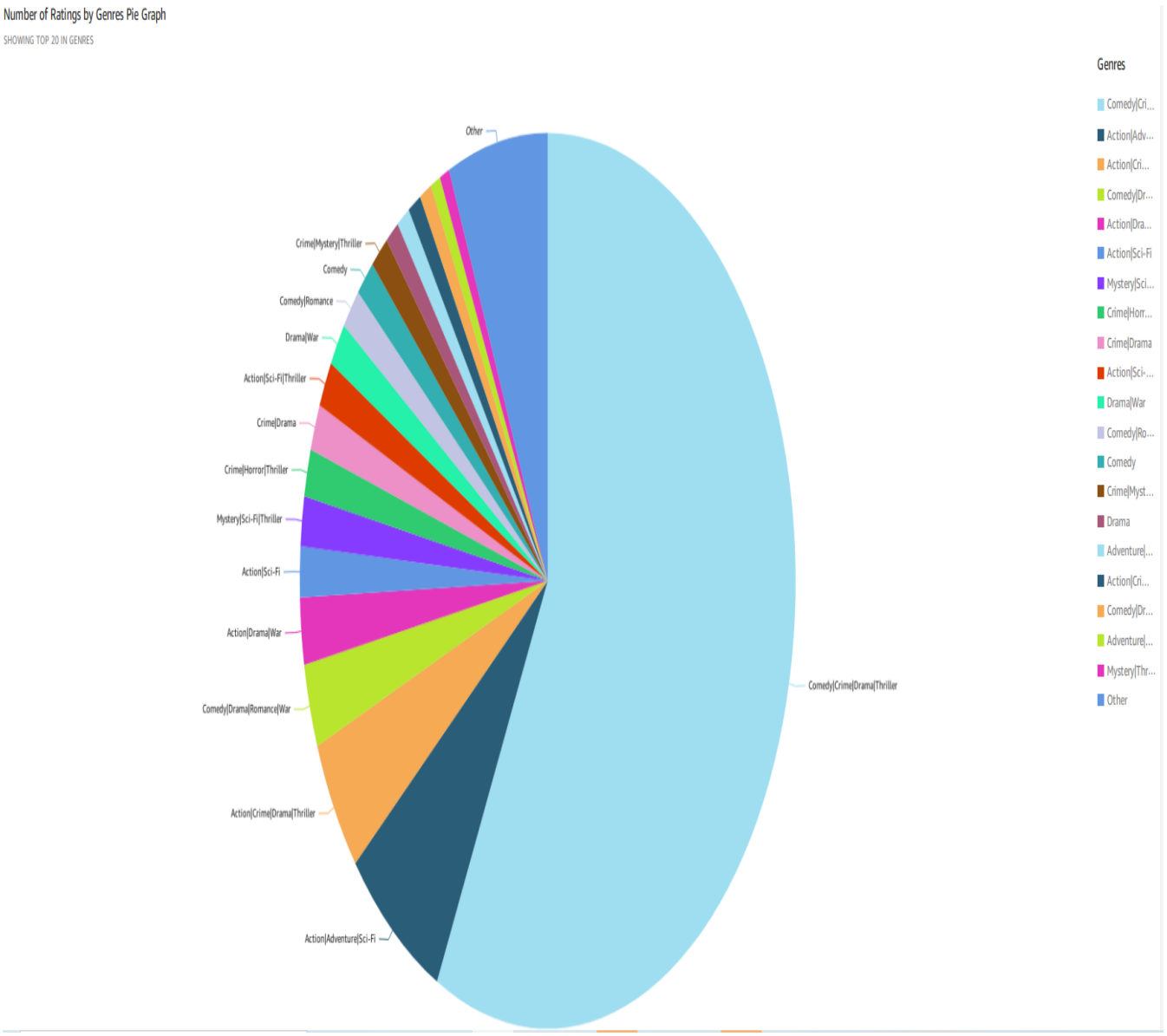
a) Exploratory Data Analysis
We use AWS Quick sight for our exploratory data analysis.



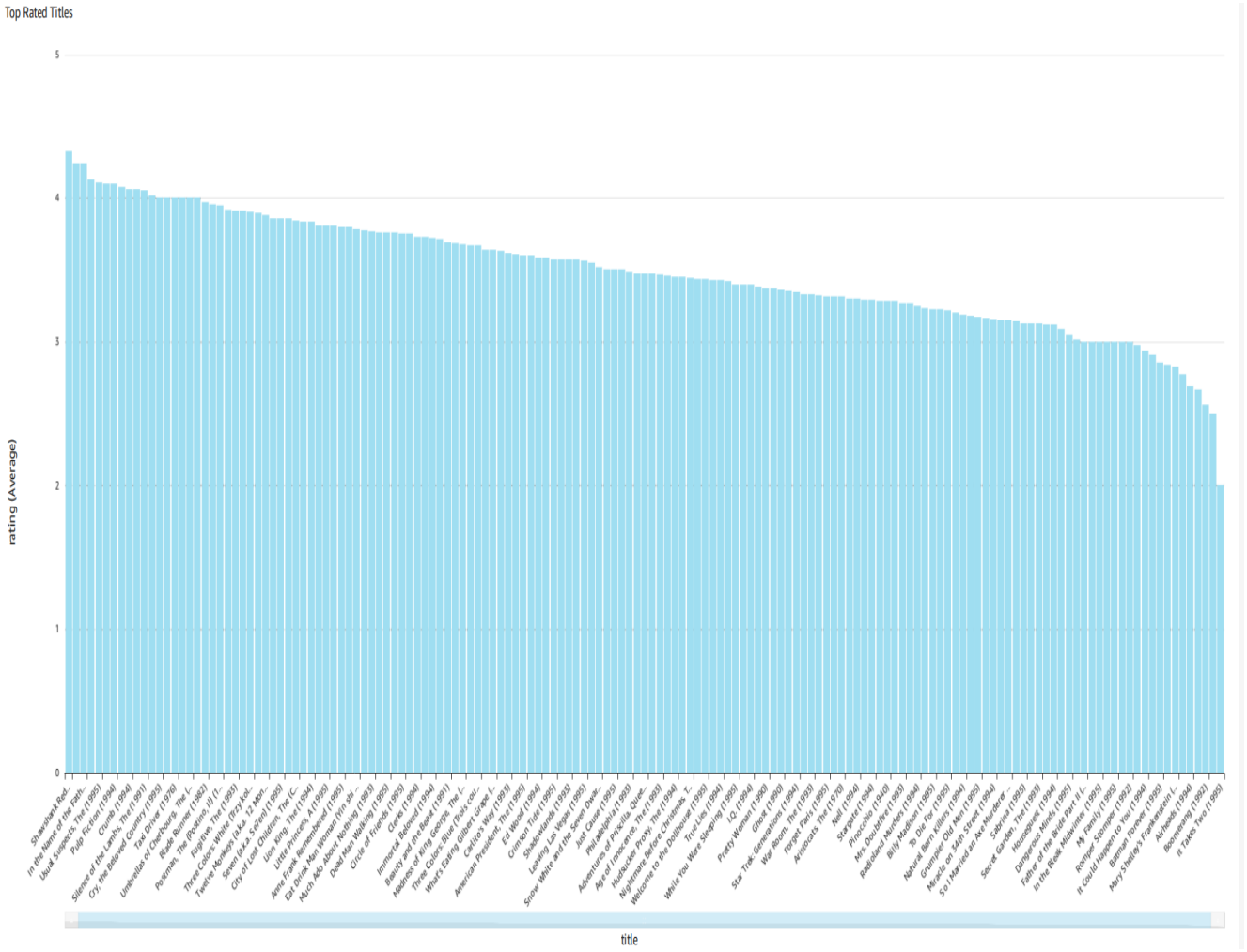
The graph above is the pie chart for representing the movies by genres. Here we can see that a large number of movies dont have one unique genre. But lie in the others category. The others category here represents that a large proportion of movies has more than one genre.



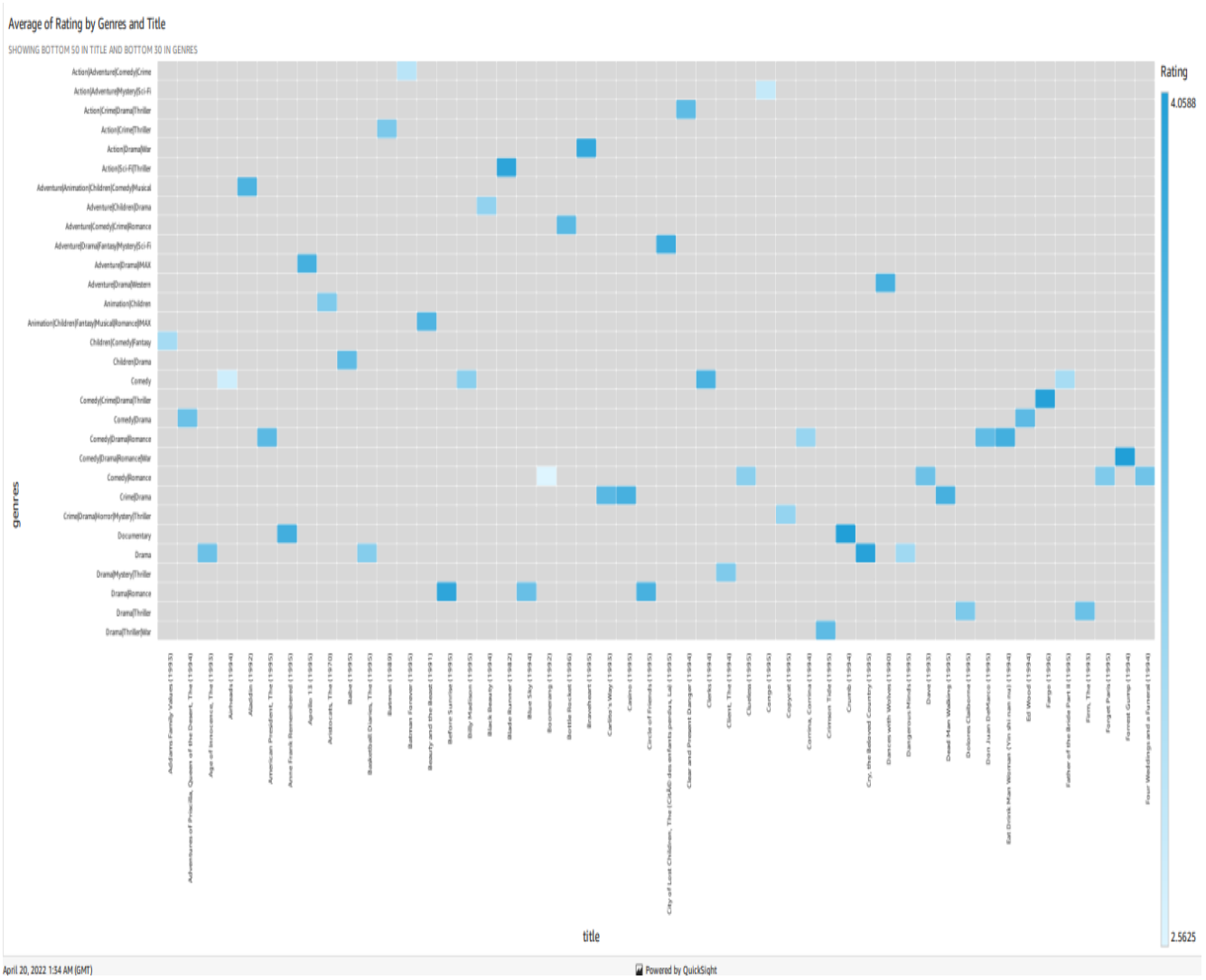
Here we see the graph of ratings vs tags. The artists who have the highest to lowest ratings are given below. This gives us an idea of the peoples choice. We can see that in our data Morgan Freeman has the highest number of ratings. This also gives us an idea that the graph is skewed. Tags can’t be considered as a feature in our model.



Here we plot the pie diagram to represent the number of ratings by genres. This is an interesting plot which shows us that which genres united together gives the best combination and has the highest rating. Our data says that the movies having genres like Comedy, Criminal, Drama and Thriller are the highest rated movies among users. The movies with Action genre precede these choices. The least preferred genre would be a combination of Criminal, Mystery, Thriller and Adventure movies. These features give the recommendation system a clear idea of what the preferences are in similar audience.



The most important relation of all is the title and ratings analysis. This graph shows us the average ratings for a movie out of 5. The favourites and relevant favourites can be mapped through this. Our main aim is to take the users input and based on the input give similar movies that might be the users favourite or are rated similar by the other users.



Average of Ratings by Genre and Title this tells us the value of these 3 features in our model. This clearly shows that our data is ready to undergo further process. Each movie can be distinguished as to which movie belongs to which genre and what rating it scores in its genre.

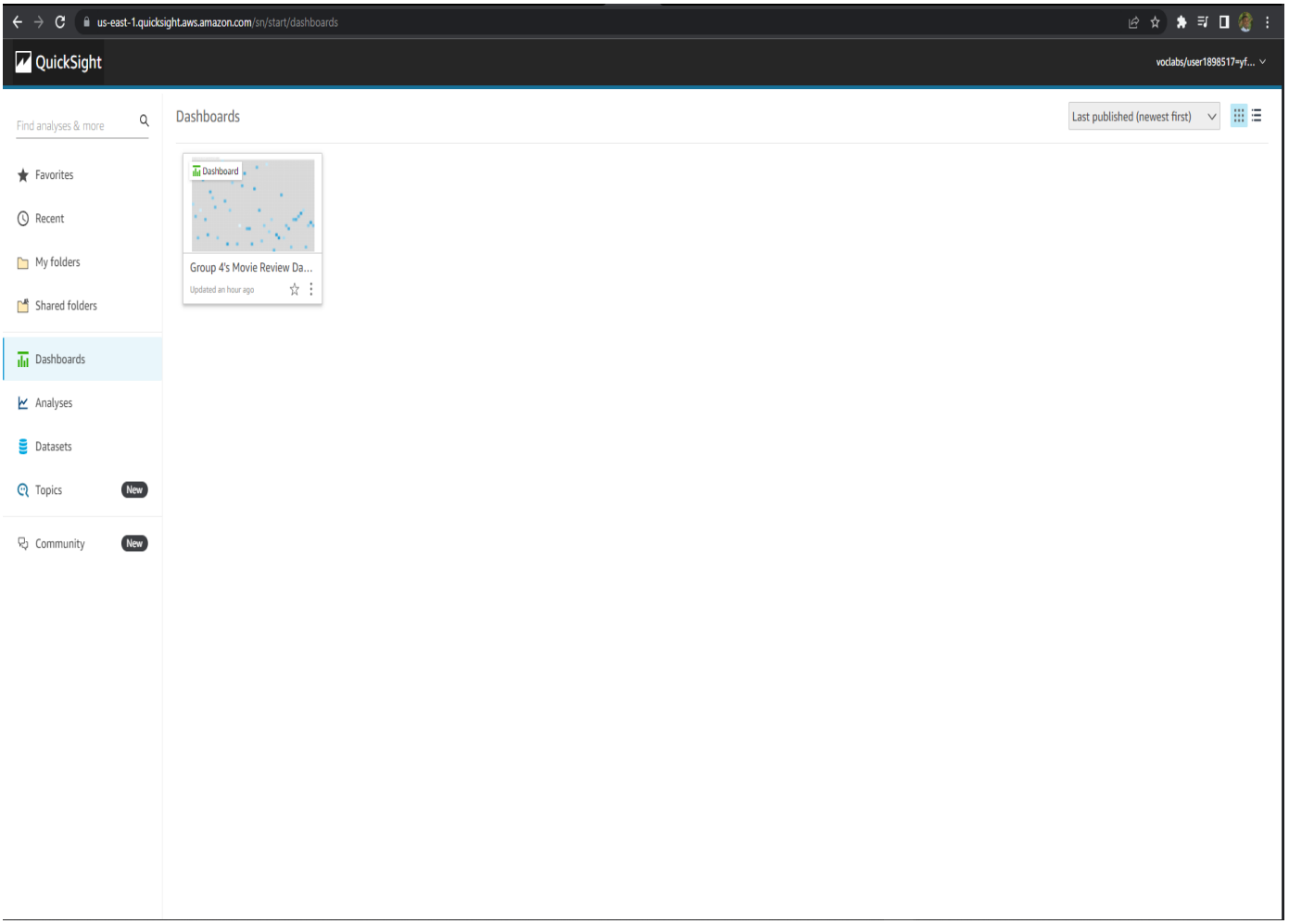
b) Dashboard

The link to the dashboard is

https://us-east-1.quicksight.aws.amazon.com/sn/accounts/530834130991/dashboards/1b211208-2faf-4afd-844a-6dc13bcfe64f?directory_alias=yawfrempong

Since it is a private instance. It cant be accessed by others.

The screenshot below will show how the dashboard in Quicksight Looks.


us-east-1.quicksight.aws.amazon.com/sn/start/dashboards
QuickSight
voclabs/user1898517=yf...
Find analyses & more
Favorites
Recent
My folders
Shared folders
Dashboards
Analyses
Datasets
Topics
New
Community
New
Dashboards
Last published (newest first)
Dashboard
Group 4's Movie Review Da...
Updated an hour ago

Data Preparation

Step 1

We use Sage maker for the data preparation process and data cleaning process. We take Imports that are required for the process.

```
#imports
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import boto3
import os
import pandas as pd
!pip install wordcloud
from wordcloud import WordCloud, STOPWORDS
from collections import Counter
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
import random
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.1-cp36-cp36m-manylinux1_x86_64.whl (366 kB)
     |████████████████████████████████| 366 kB 24.6 MB/s
Requirement already satisfied: pillow in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from wordcloud) (8.
4.0)
Requirement already satisfied: numpy>=1.6.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from wordclou
d) (1.19.5)
Requirement already satisfied: matplotlib in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from wordcloud)
(3.3.4)
Requirement already satisfied: python-dateutil>=2.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from
matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from mat
plotlib->wordcloud) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/
```

Step 2

We need to Create an S3 bucket where all our csv files will be stored. We write simple python code to get the csv files. Convert them to dataframes and merge these data frames to get the required data set.

```
In [2]: bucketname = 'recommendation-storage'
my_region = boto3.session.Session().region_name
print(my_region)

s3 = boto3.resource('s3')
try:
    if my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucketname)
    print("S3 bucket created sucessfully")
except Exception as e:
    print("S3 error: " + str(e))

prefix = 'kmeans-clustering'
output_path = 's3://{}/{}/output'.format(bucketname,prefix)
print(output_path)
```

```
us-east-1
S3 bucket created sucessfully
s3://recommendation-storage/kmeans-clustering/output
```

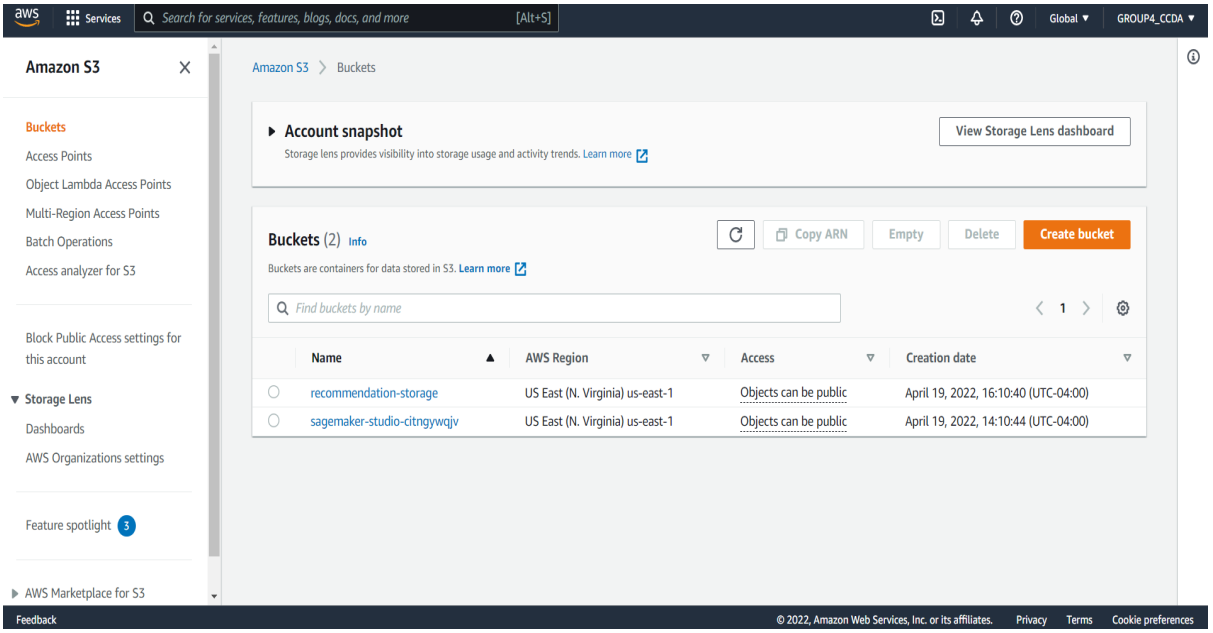
In this code we check if the region is us-east-1 so that our account location matches the S3 bucket location.

```
In [7]: data_key = 'ratings.csv'
        data_location = 's3://{}/{}'.format(bucketname, data_key)
        title_ratings = pd.read_csv(data_location,sep=',')

        data_key = 'movies.csv'
        data_location = 's3://{}/{}'.format(bucketname, data_key)
        title_movies = pd.read_csv(data_location,sep=',')
        data_key = 'links.csv'
        data_location = 's3://{}/{}'.format(bucketname, data_key)
        title_links = pd.read_csv(data_location,sep=',')
        data_key = 'tags.csv'
        data_location = 's3://{}/{}'.format(bucketname, data_key)
        title_tags = pd.read_csv(data_location,sep=',')


        model_data = pd.merge(title_ratings,title_movies, on = 'movieId')


        model_data.to_csv('merged_data.csv')

        boto3.Session().resource('s3').Bucket(bucketname).Object(os.path.join(prefix, 'merged_data.csv')).upload_file('merged_data.csv')
```

We get the file merged_data after merging the files we need for our analysis. Here MovieId is the primary key for movies.csv and a foreign key in all other csvs.

Step 3

Check our S3 folder has been created. This folder will have our S3 repository and all other data. The folder now also has the newly merged data file.The recommendation-storage is the bucket we need.



Step 4

We can write a json code to take our data

```
data_dir = "data"
!mkdir $data_dir

!cd $data_dir && wget http://files.grouplens.org/datasets/movielens/ml-latest-small.zip
!cd $data_dir && unzip ml-latest-small.zip
dataset_dir = data_dir + "/ml-latest-small/"
!ls $dataset_dir
```

```
--2022-04-19 20:51:57--  http://files.grouplens.org/datasets/movielens/ml-latest-small.zip
Resolving files.grouplens.org (files.grouplens.org)... 128.101.65.152
Connecting to files.grouplens.org (files.grouplens.org)|128.101.65.152|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 978202 (955K) [application/zip]
Saving to: ‘ml-latest-small.zip’

ml-latest-small.zip 100%[===================>] 955.28K  4.75MB/s    in 0.2s

2022-04-19 20:51:58 (4.75 MB/s) - ‘ml-latest-small.zip’ saved [978202/978202]

Archive:  ml-latest-small.zip
   creating: ml-latest-small/
  inflating: ml-latest-small/links.csv
  inflating: ml-latest-small/tags.csv
  inflating: ml-latest-small/ratings.csv
  inflating: ml-latest-small/README.txt
  inflating: ml-latest-small/movies.csv
links.csv  movies.csv  ratings.csv  README.txt  tags.csv
```

Step 5

Now we read the newly merged csv file to perform further cleaning of data according to our needs.

In [15]:
```
project_data = pd.read_csv(dataset_dir + '/result_final_1 (1).csv')
print(project_data.info())
project_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98386 entries, 0 to 98385
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   movieId    98386 non-null  int64
 1   imdbId     98386 non-null  int64
 2   tmdbId     98386 non-null  int64
 3   title      98386 non-null  object
 4   genres     98386 non-null  object
 5   rating     98386 non-null  float64
 6   userId     98386 non-null  int64
 7   tag        98386 non-null  object
 8   timestamp  98386 non-null  int64
 9   userId.1   98386 non-null  int64
dtypes: float64(1), int64(6), object(3)
memory usage: 7.5+ MB
None
```

Out[15]:

	movieId	imdbId	tmdbId	title	genres	rating	userId	tag	timestamp	userId.1
0	1	114709	862	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	4.0	1	pixar	1139045764	336
1	1	114709	862	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	4.0	1	pixar	1137206825	474
2	1	114709	862	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	4.0	1	fun	1525286013	567
3	3	113228	15602	Grumpier Old Men (1995)	Comedy|Romance	4.0	1	moldy	1143424860	289
4	3	113228	15602	Grumpier Old Men (1995)	Comedy|Romance	4.0	1	old	1143424860	289

Step 6

In [22]:
```
watched_df = project_data.copy()
watched_df = watched_df[watched_df['rating'] > 3]
watched_df = watched_df[['userId', 'movieId', 'timestamp','genres','title']]
watched_df['EVENT_TYPE']='watch'

clicked_df = project_data.copy()
clicked_df = clicked_df[clicked_df['rating'] > 1]
clicked_df = clicked_df[['userId', 'movieId', 'timestamp', 'genres','title']]
clicked_df['EVENT_TYPE']='click'

interactions_df = clicked_df.copy()
interactions_df = interactions_df.append(watched_df)
interactions_df.sort_values("timestamp", axis = 0, ascending = True,
                 inplace = True, na_position ='last')
```

In [25]:
```
interactions_df.rename(columns = {'userId':'USER_ID', 'movieId':'ITEM_ID',
                                  'timestamp':'TIMESTAMP','genres':'GENRES'}, inplace = True)
print(interactions_df.head())
interactions_filename = "interactions.csv"
interactions_df.to_csv((data_dir+"/"+interactions_filename), index=False, float_format='%.0f')
```

```
       USER_ID  ITEM_ID   TIMESTAMP         GENRES              title  \
43159      262       28  1137180037  Drama|Romance  Persuasion (1995)
43159      262       28  1137180037  Drama|Romance  Persuasion (1995)
5658        33       28  1137180037  Drama|Romance  Persuasion (1995)
5658        33       28  1137180037  Drama|Romance  Persuasion (1995)
78591      474       28  1137180037  Drama|Romance  Persuasion (1995)

      EVENT_TYPE
43159      click
43159      watch
5658       watch
5658       click
78591      click
```

In this step, we define two variables in the dataset to filter out unliked movies and better simulate data gathered by a video-on-demand (VOD) platform.

Since this is an explicit feedback movie rating dataset, it includes movies rated from 1 to 5. For this tutorial, we want to include only moves that were "liked" by the users, and simulate a implicit dataset that is similar to data that is gathered by a video-on-demain (VOD) platform. For that, you will next filter out all interactions below 2 out of 5, and create two EVENT_TYPE variables: *click* and *watch*. Any movies rated 2 and above are assigned as *click*, and any movies rated 4 and above are assigned as *click* and *watch*.

This is the final step for our data cleaning. We get the data like we want from here.
Further, We will choose a ML algorithm to do further processes.

The AWS features we used are:-
AWS S3
AWS SAGE MAKER
AWS QuickSight

Sources
https://grouplens.org/datasets/movielens/
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
https://numpy.org/