CAPSTONE PROJECT REPORT – NETFLIX

Executive Post Graduate Certification in Data Science & Artificial Intelligence

Project Report

December 14, 2024

Project By: Bhupinder Singh

Contents

1.0 About

1.1 Introduction

This document explains the details of capstone project of Netflix. The dataset used for the project consists of tv shows and movies on Netflix. The dataset was provided by Intellipaat team to work on this capstone project. The dataset is used in conjunction with IMDB ratings dataset which is also provided by the Intellipaat team.

1.2 Project Goal

The project consists of analyzing the dataset of the tv shows and movies and recommending movies based on the interest and ratings. The objective I set for myself is to build an effective recommendation system for Netflix that utilizes the **surprise** library from **sciket** package.

2.0 Problem Statement

For this Project, we will be creating one such Recommendation Engine from the ground-up, where every single user, based on their area of interest and ratings, would be recommended a list of movies that are best suited for them.

3.0 Proposed Solution

3.1 Technology

This analysis of the dataset uses Python as a main technology to work on this project. I have used Google Colab as an IDE. This IDE is fully loaded with most of the ML libraries to be used in the analysis of the dataset.

3.2 Machine Learning Libraries Used

We have used following main libraries the purpose of this project:

1) Pandas
2) Numpy
3) Matplotlib
4) Seaborn
5) Scikit-surprise

Multiple methods were used from these libraries.

3.3 Description of Dataset

There are two datasets used for the analysis. The details of those datasets are as follows:

3.3.1 Combined_Data1

The dataset was provided in the form of a text file (.TXT). This dataset consists of movies ratings given by a user on a specific date.

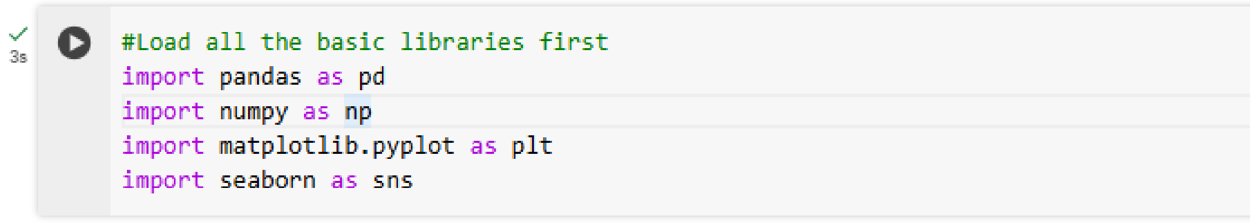
3.3.2 Movie_Titles

This dataset consists of tv shows and movies list and the year in which they were released.

3.4 EDA (Part 1 – Combined Dataset)

The output consists of many sections. Each section is explained in detail along with a screenshot of the solution.

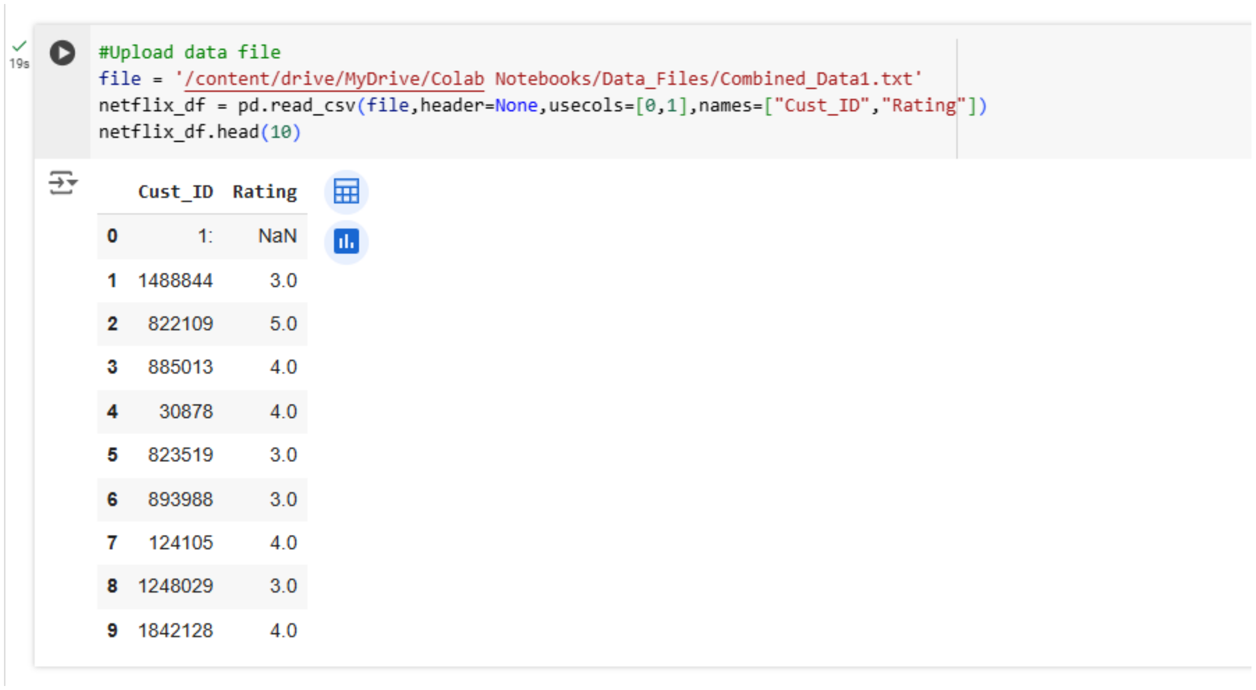
3.4.1 Load Basic Libraries



```
#Load all the basic libraries first
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

3.4.2 Data Load (Customers & Moving Ratings)

We are loading data from "Combined_Data1.txt" data file. This is a text file, and data consists of in different formats. The names of the columns are not named; so, will name the columns as Cust_ID & Rating. The value in first column "Cust_ID" is having a value with colon (:) with Rating value as NaN which means this is a Movie ID. We will separate this value and store this value on a new column "Movie_ID".

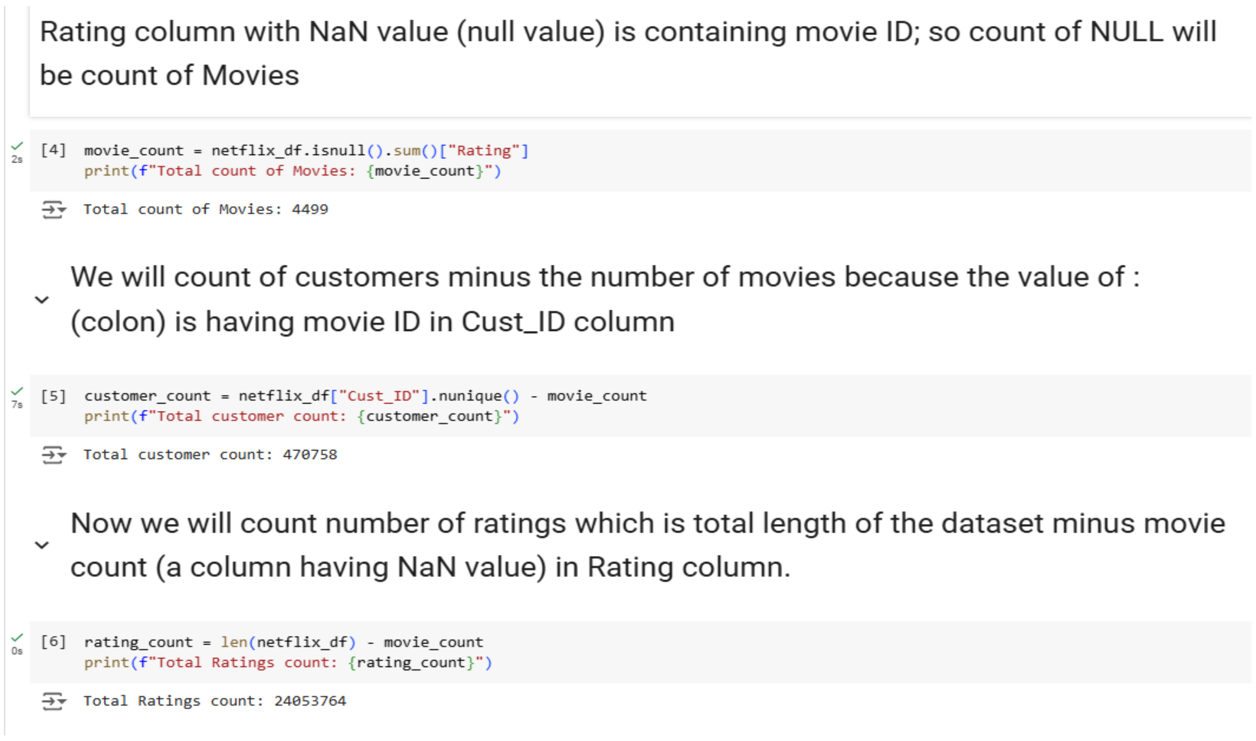


```
#Upload data file
file = '/content/drive/MyDrive/Colab Notebooks/Data_Files/Combined_Data1.txt'
netflix_df = pd.read_csv(file,header=None,usecols=[0,1],names=["Cust_ID","Rating"])
netflix_df.head(10)
```

```
   Cust_ID  Rating
0       1:     NaN
1  1488844     3.0
2   822109     5.0
3   885013     4.0
4    30878     4.0
5   823519     3.0
6   893988     3.0
7   124105     4.0
8  1248029     3.0
9  1842128     4.0
```

3.4.3 Handling Missing Values

I have worked on handling missing values or correcting data which is not properly defined in the dataset e.g. Movie ID is given at the start of the customer ID and Ratings. In other words, we can say data is defined as group above format report.



Rating column with NaN value (null value) is containing movie ID; so count of NULL will be count of Movies

```
[4] movie_count = netflix_df.isnull().sum()["Rating"]
    print(f"Total count of Movies: {movie_count}")
```

```
Total count of Movies: 4499
```

We will count of customers minus the number of movies because the value of : (colon) is having movie ID in Cust_ID column

```
[5] customer_count = netflix_df["Cust_ID"].nunique() - movie_count
    print(f"Total customer count: {customer_count}")
```

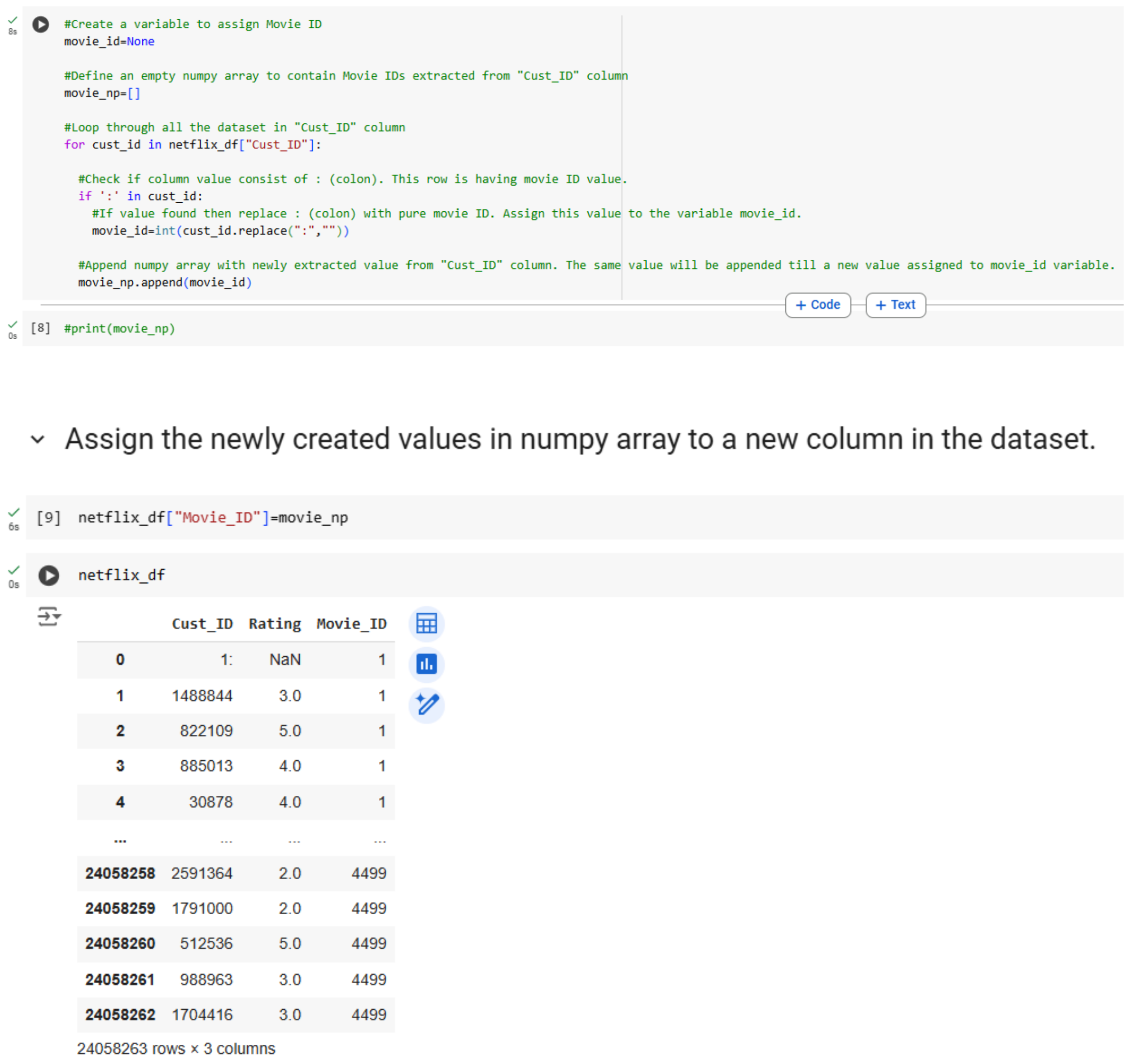
```
Total customer count: 470758
```

Now we will count number of ratings which is total length of the dataset minus movie count (a column having NaN value) in Rating column.

```
[6] rating_count = len(netflix_df) - movie_count
    print(f"Total Ratings count: {rating_count}")
```

```
Total Ratings count: 24053764
```

3.4.4 Extract Movie ID from the data



```
#Create a variable to assign Movie ID
movie_id=None

#Define an empty numpy array to contain Movie IDs extracted from "Cust_ID" column
movie_np=[]

#Loop through all the dataset in "Cust_ID" column
for cust_id in netflix_df["Cust_ID"]:

  #Check if column value consist of : (colon). This row is having movie ID value.
  if ':' in cust_id:
    #If value found then replace : (colon) with pure movie ID. Assign this value to the variable movie_id.
    movie_id=int(cust_id.replace(":",""))

  #Append numpy array with newly extracted value from "Cust_ID" column. The same value will be appended till a new value assigned to movie_id variable.
  movie_np.append(movie_id)
```

```
[8] #print(movie_np)
```

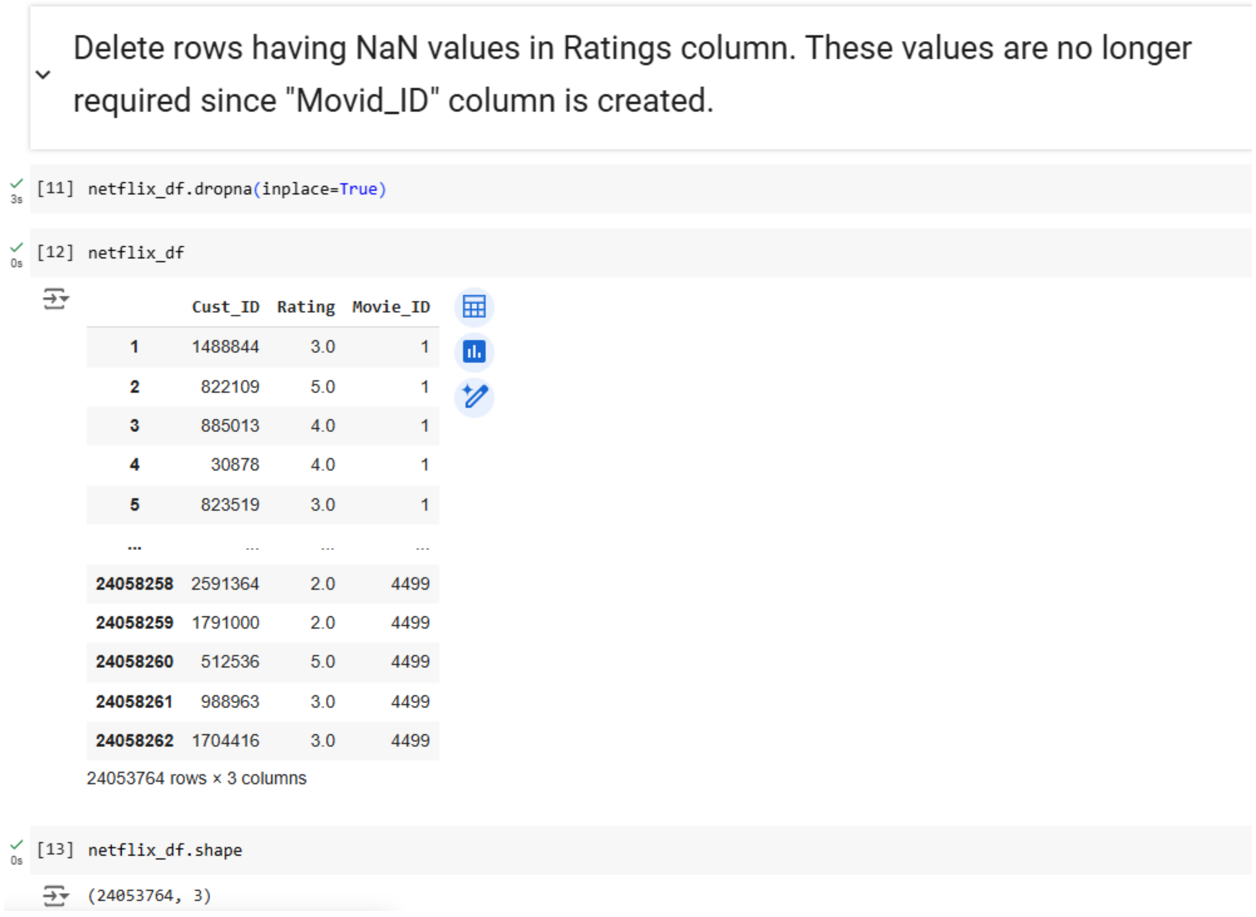
Assign the newly created values in numpy array to a new column in the dataset.

```
[9] netflix_df["Movie_ID"]=movie_np
```

```
netflix_df
```

```
          Cust_ID  Rating  Movie_ID
0              1:     NaN         1
1         1488844     3.0         1
2          822109     5.0         1
3          885013     4.0         1
4           30878     4.0         1
...           ...     ...       ...
24058258  2591364     2.0      4499
24058259  1791000     2.0      4499
24058260   512536     5.0      4499
24058261   988963     3.0      4499
24058262  1704416     3.0      4499
24058263 rows × 3 columns
```

3.4.5 Delete NaN (NULL) values



Delete rows having NaN values in Ratings column. These values are no longer required since "Movid_ID" column is created.

```
[11] netflix_df.dropna(inplace=True)
```

```
[12] netflix_df
```

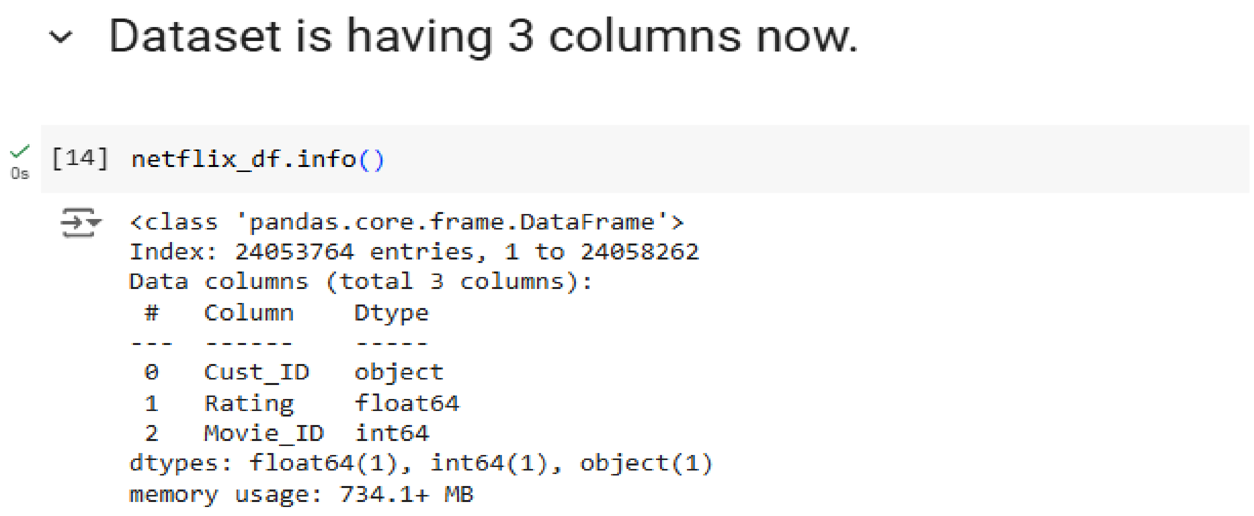
	Cust_ID	Rating	Movie_ID
1	1488844	3.0	1
2	822109	5.0	1
3	885013	4.0	1
4	30878	4.0	1
5	823519	3.0	1
...	...	...	...
24058258	2591364	2.0	4499
24058259	1791000	2.0	4499
24058260	512536	5.0	4499
24058261	988963	3.0	4499
24058262	1704416	3.0	4499

24053764 rows × 3 columns

```
[13] netflix_df.shape
```

```
(24053764, 3)
```

3.4.6 Summary of Dataset



Dataset is having 3 columns now.

```
[14] netflix_df.info()
```

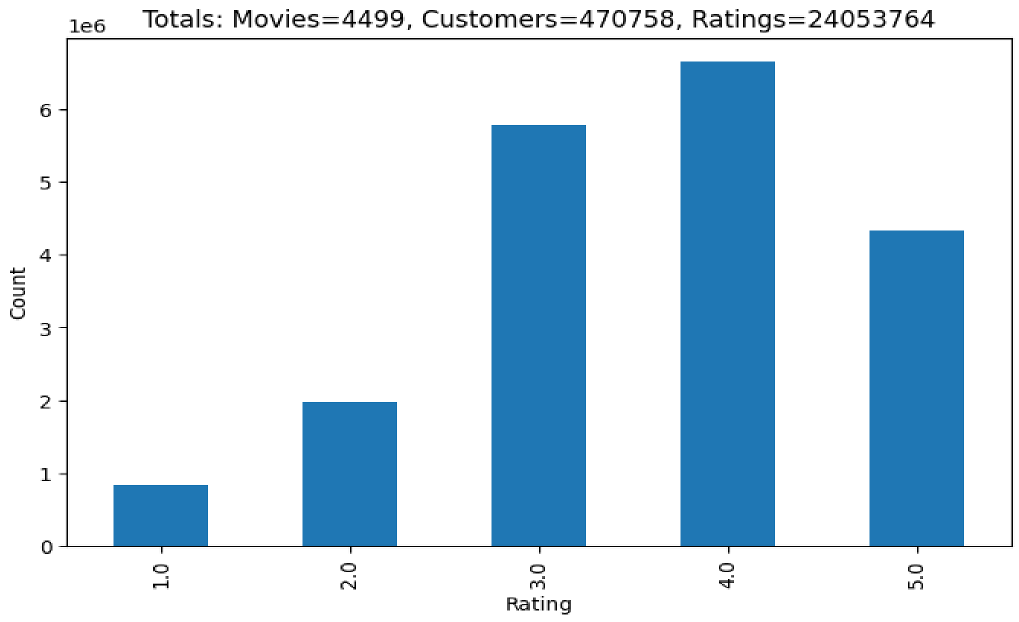
```
<class 'pandas.core.frame.DataFrame'>
Index: 24053764 entries, 1 to 24058262
Data columns (total 3 columns):
 #   Column    Dtype  
---  ------    -----  
 0   Cust_ID   object 
 1   Rating    float64
 2   Movie_ID  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 734.1+ MB
```

3.4.7 Create Data Plot

```
ax = data_for_plot.plot(figsize=(8,5),legend = False, kind = 'bar')
plt.title(f"Totals: Movies={movie_count}, Customers={customer_count}, Ratings={rating_count}")

plt.xlabel("Rating")
plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```



3.4.8 List of total Movie Ratings

Make a list of total ratings of all the movies. This will show how many total ratings a movie has got.

This will be our filter 1

```
[17] movie_rating_summary = netflix_df.groupby("Movie_ID")["Rating"].count()
```

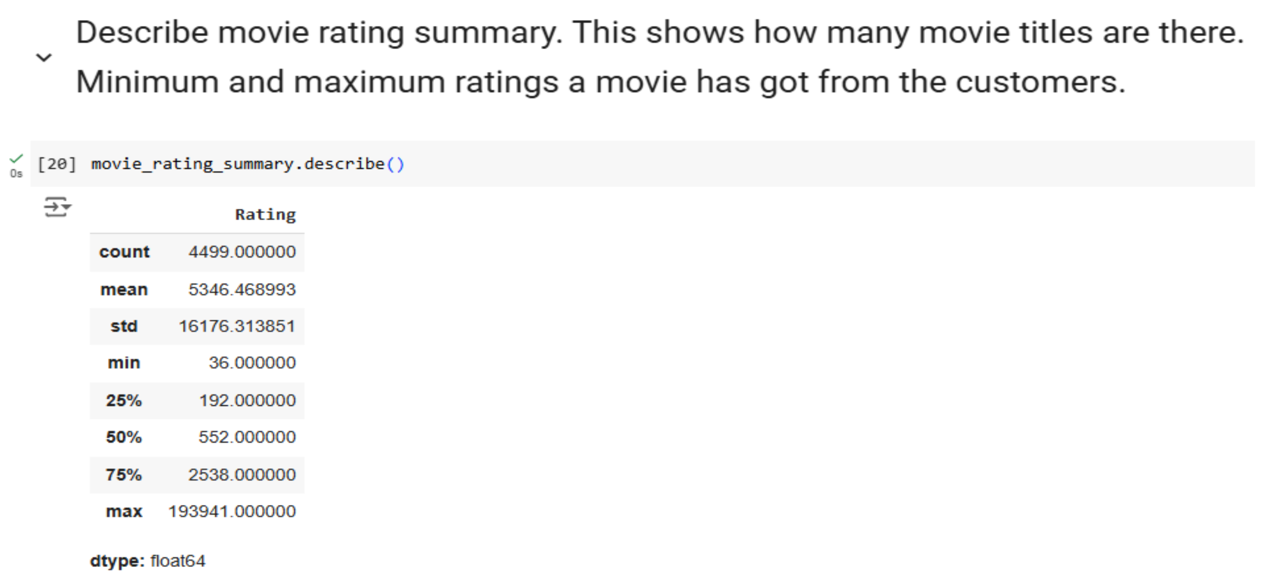
```
[18] movie_rating_summary
```

	Rating
Movie_ID	
1	547
2	145
3	2012
4	142
5	1140
...	...
4495	614
4496	9519
4497	714
4498	269
4499	428

4499 rows × 1 columns

dtype: int64

3.4.9 Movie Ratings Summary



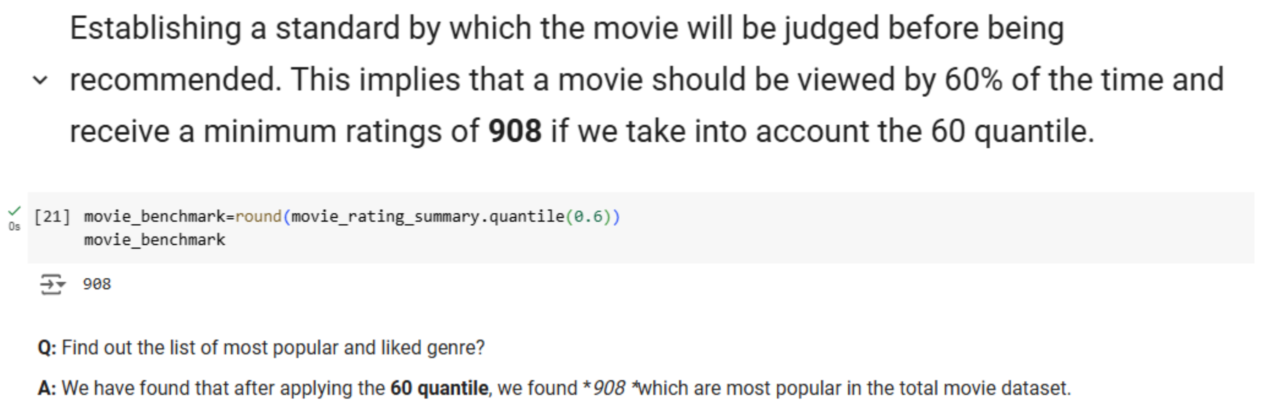
Describe movie rating summary. This shows how many movie titles are there. Minimum and maximum ratings a movie has got from the customers.

```
[20] movie_rating_summary.describe()
```

	Rating
count	4499.000000
mean	5346.468993
std	16176.313851
min	36.000000
25%	192.000000
50%	552.000000
75%	2538.000000
max	193941.000000

dtype: float64

3.4.10 Movie Benchmark



Establishing a standard by which the movie will be judged before being recommended. This implies that a movie should be viewed by 60% of the time and receive a minimum ratings of **908** if we take into account the 60 quantile.

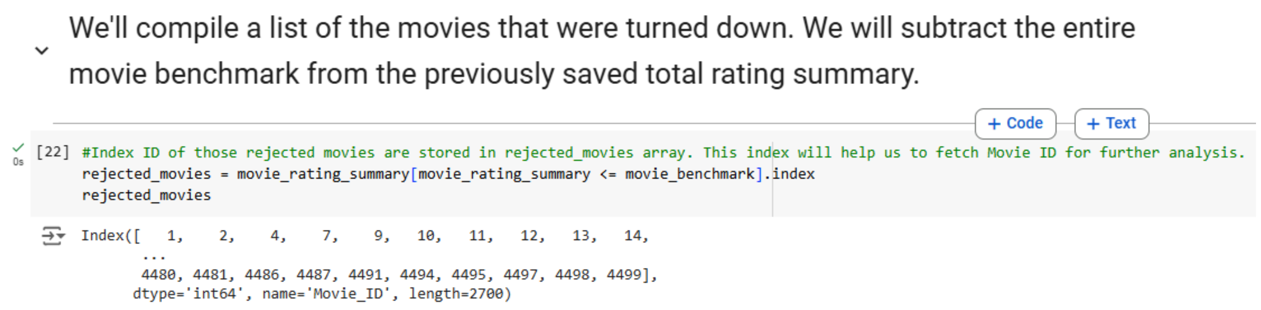
```
[21] movie_benchmark=round(movie_rating_summary.quantile(0.6))
     movie_benchmark
```

```
908
```

Q: Find out the list of most popular and liked genre?

A: We have found that after applying the **60 quantile**, we found *908 *which are most popular in the total movie dataset.

3.4.11 Rejected Movies with Index value

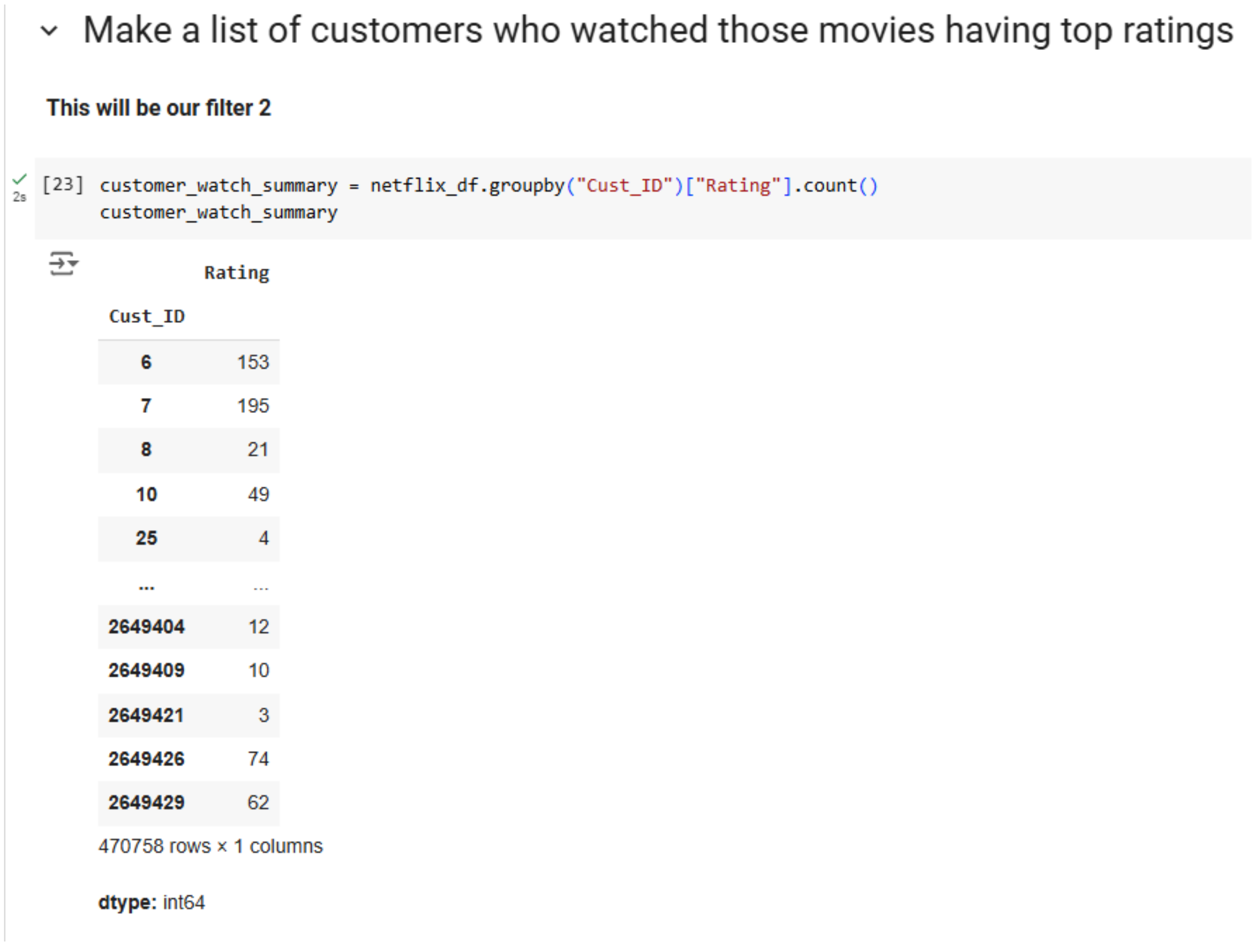


We'll compile a list of the movies that were turned down. We will subtract the entire movie benchmark from the previously saved total rating summary.

```
[22] #Index ID of those rejected movies are stored in rejected_movies array. This index will help us to fetch Movie ID for further analysis.
     rejected_movies = movie_rating_summary[movie_rating_summary <= movie_benchmark].index
     rejected_movies
```

```
Index([   1,    2,    4,    7,    9,   10,   11,   12,   13,   14,
       ...
       4480, 4481, 4486, 4487, 4491, 4494, 4495, 4497, 4498, 4499],
      dtype='int64', name='Movie_ID', length=2700)
```

3.4.12 List of Customers watched movies



Make a list of customers who watched those movies having top ratings

This will be our filter 2

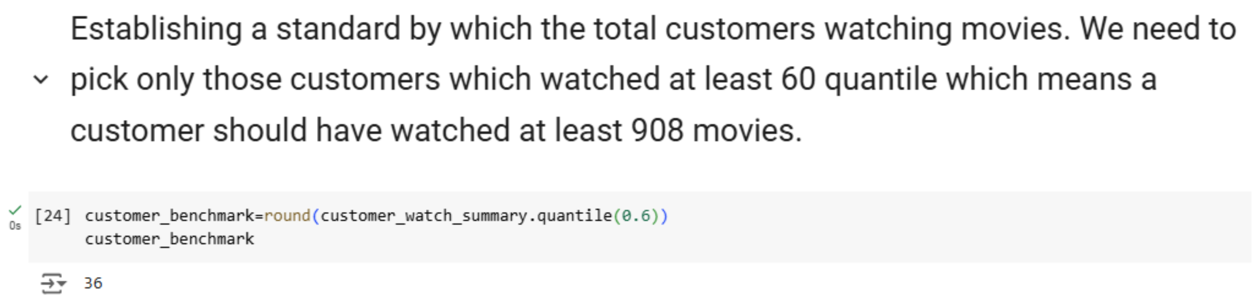
```
[23] customer_watch_summary = netflix_df.groupby("Cust_ID")["Rating"].count()
     customer_watch_summary
```

	Rating
Cust_ID	
6	153
7	195
8	21
10	49
25	4
...	...
2649404	12
2649409	10
2649421	3
2649426	74
2649429	62

470758 rows × 1 columns

dtype: int64

3.4.13 List of Customers watched top rated movies

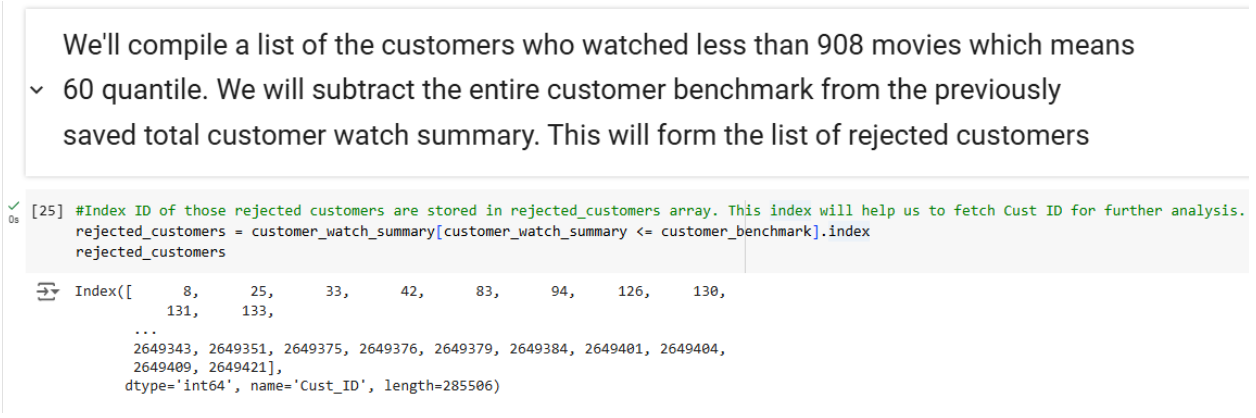


Establishing a standard by which the total customers watching movies. We need to pick only those customers which watched at least 60 quantile which means a customer should have watched at least 908 movies.

```
[24] customer_benchmark=round(customer_watch_summary.quantile(0.6))
     customer_benchmark
```

```
36
```

3.4.14 List of Rejected Customers

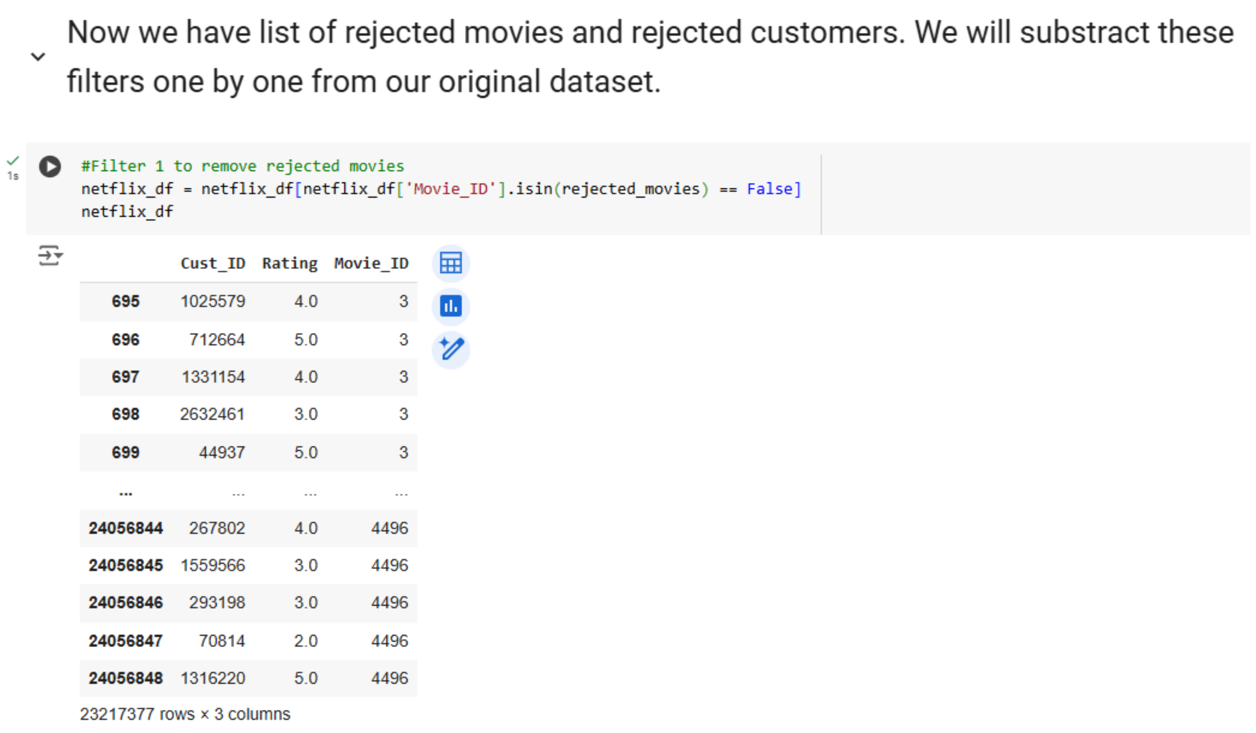


We'll compile a list of the customers who watched less than 908 movies which means 60 quantile. We will subtract the entire customer benchmark from the previously saved total customer watch summary. This will form the list of rejected customers

```
[25] #Index ID of those rejected customers are stored in rejected_customers array. This index will help us to fetch Cust ID for further analysis.
     rejected_customers = customer_watch_summary[customer_watch_summary <= customer_benchmark].index
     rejected_customers
```

```
Index([      8,      25,      33,      42,      83,      94,     126,     130,
           131,     133,
       ...
       2649343, 2649351, 2649375, 2649376, 2649379, 2649384, 2649401, 2649404,
       2649409, 2649421],
      dtype='int64', name='Cust_ID', length=285506)
```

3.4.15 Final Dataset after filter of customers and movies

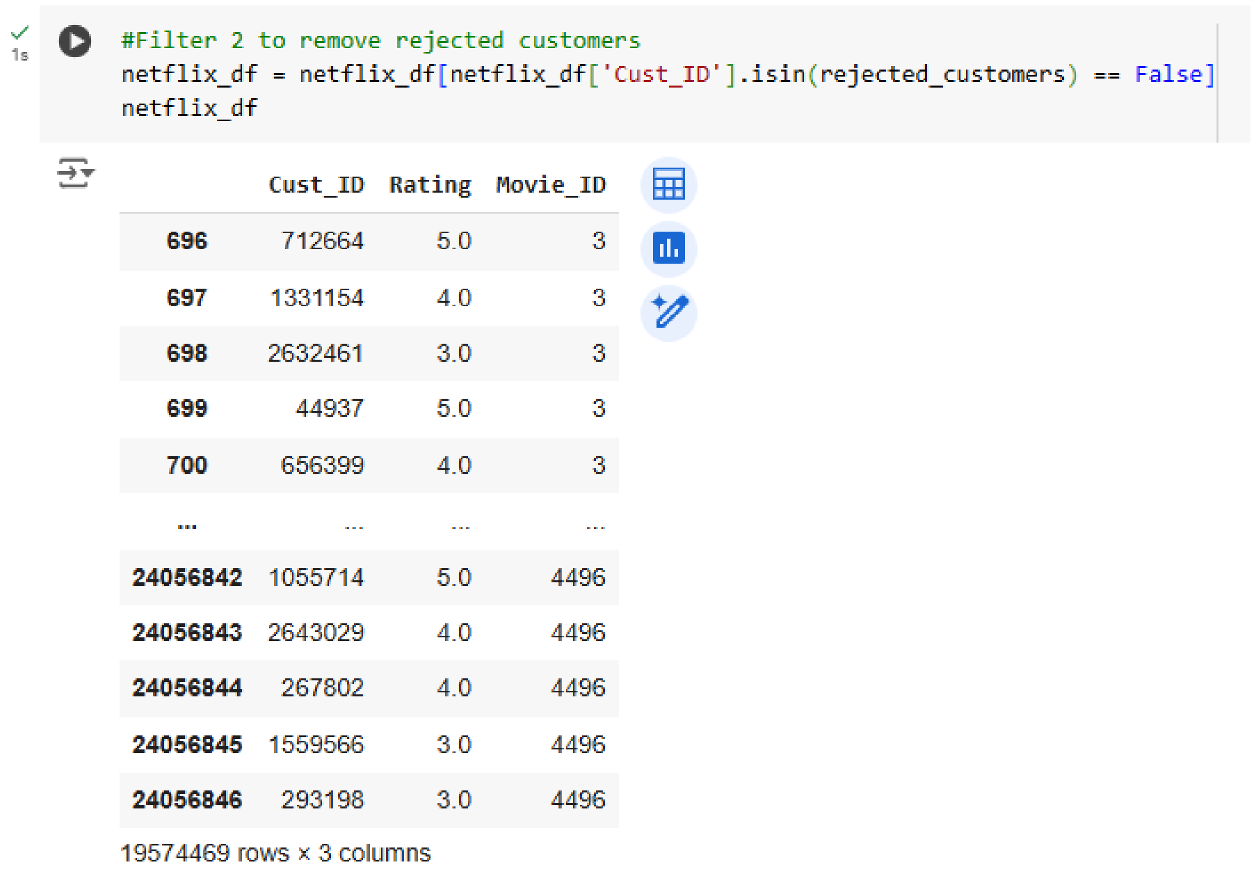


Now we have list of rejected movies and rejected customers. We will substract these filters one by one from our original dataset.

```
#Filter 1 to remove rejected movies
netflix_df = netflix_df[netflix_df['Movie_ID'].isin(rejected_movies) == False]
netflix_df
```

	Cust_ID	Rating	Movie_ID
695	1025579	4.0	3
696	712664	5.0	3
697	1331154	4.0	3
698	2632461	3.0	3
699	44937	5.0	3
...	...	...	...
24056844	267802	4.0	4496
24056845	1559566	3.0	4496
24056846	293198	3.0	4496
24056847	70814	2.0	4496
24056848	1316220	5.0	4496

23217377 rows × 3 columns



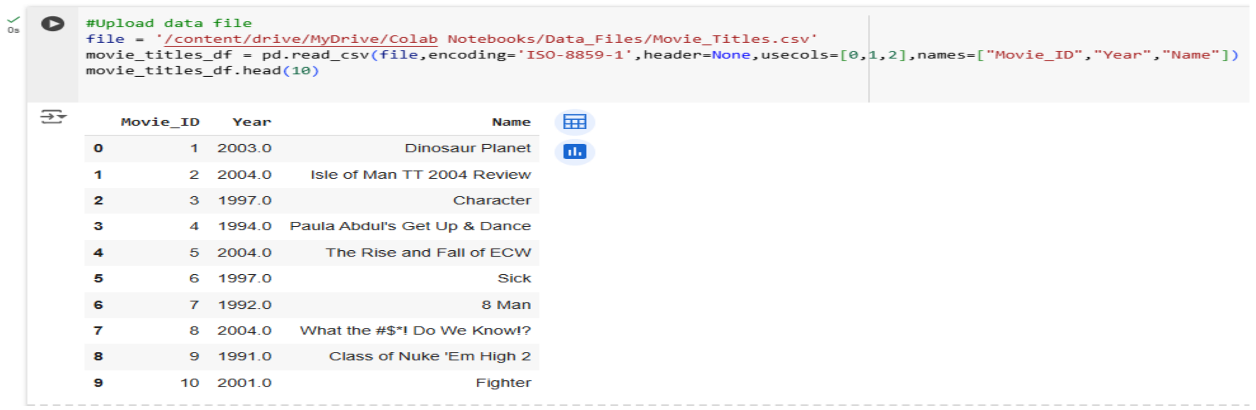
```
#Filter 2 to remove rejected customers
netflix_df = netflix_df[netflix_df['Cust_ID'].isin(rejected_customers) == False]
netflix_df
```

```
          Cust_ID  Rating  Movie_ID
696        712664     5.0         3
697       1331154     4.0         3
698       2632461     3.0         3
699         44937     5.0         3
700        656399     4.0         3
...           ...     ...       ...
24056842  1055714     5.0      4496
24056843  2643029     4.0      4496
24056844   267802     4.0      4496
24056845  1559566     3.0      4496
24056846   293198     3.0      4496
19574469 rows × 3 columns
```

3.5 EDA (Part 2 – Movie Listing)

In this EDA part we are loading another data from "**Movie_Titles.csv**" data file. This is a csv file. The names of the columns are not named; so will name the columns as *Movie_ID*, *Year* & *Name*.

3.5.1 Load Data (Movie_Titles.csv)



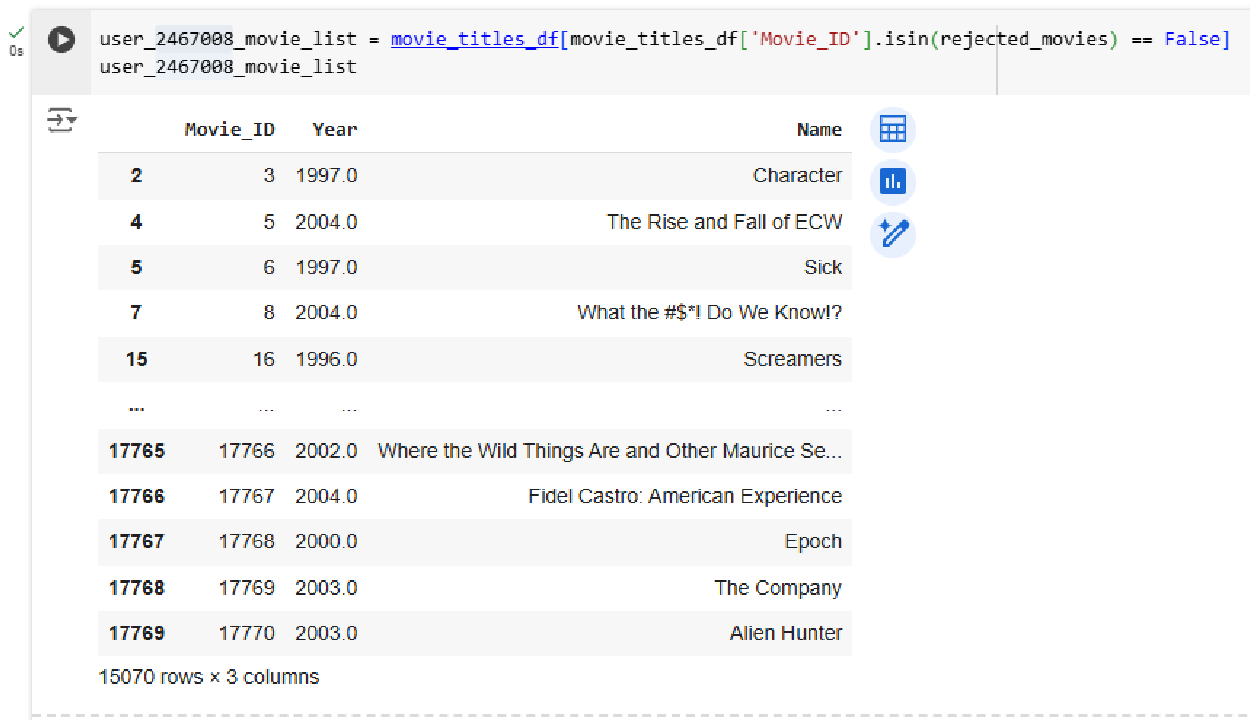
```
#Upload data file
file = '/content/drive/MyDrive/Colab Notebooks/Data_Files/Movie_Titles.csv'
movie_titles_df = pd.read_csv(file,encoding='ISO-8859-1',header=None,usecols=[0,1,2],names=["Movie_ID","Year","Name"])
movie_titles_df.head(10)
```

```
   Movie_ID    Year                          Name
0         1  2003.0               Dinosaur Planet
1         2  2004.0    Isle of Man TT 2004 Review
2         3  1997.0                     Character
3         4  1994.0  Paula Abdul's Get Up & Dance
4         5  2004.0      The Rise and Fall of ECW
5         6  1997.0                          Sick
6         7  1992.0                         8 Man
7         8  2004.0      What the #$*! Do We Know!?
8         9  1991.0        Class of Nuke 'Em High 2
9        10  2001.0                       Fighter
```

3.5.2 Remove Movie Title which are Rejected

The dataset is named with **user ID 2467008** because we will be considering this user for recommendations.

Remove movie titles which are rejected as part of our analysis



```
user_2467008_movie_list = movie_titles_df[movie_titles_df['Movie_ID'].isin(rejected_movies) == False]
user_2467008_movie_list
```

	Movie_ID	Year	Name
2	3	1997.0	Character
4	5	2004.0	The Rise and Fall of ECW
5	6	1997.0	Sick
7	8	2004.0	What the #$*! Do We Know!?
15	16	1996.0	Screamers
...	...	...	...
17765	17766	2002.0	Where the Wild Things Are and Other Maurice Se...
17766	17767	2004.0	Fidel Castro: American Experience
17767	17768	2000.0	Epoch
17768	17769	2003.0	The Company
17769	17770	2003.0	Alien Hunter

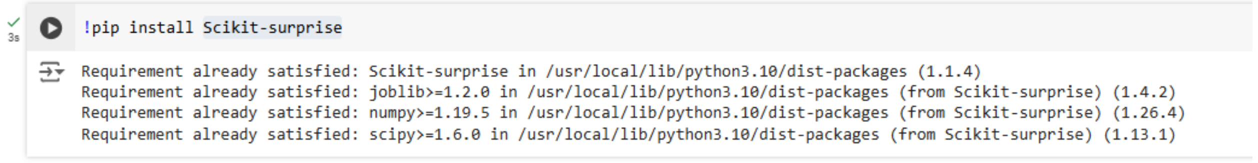
15070 rows × 3 columns

4.0 Training the Model

This section explores the model building process. There are various steps to build a model. Those are explained as follows:

4.1 Install Scikit-Surprise Package

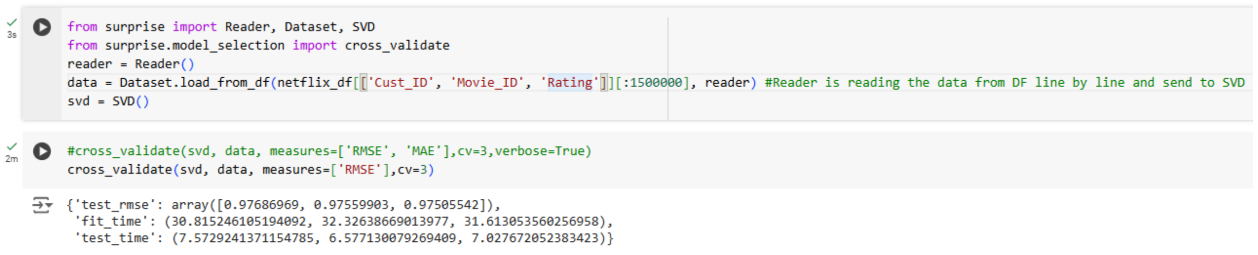
Surprise is a Python scikit for building and analyzing recommender systems that deal with explicit rating data. Surprise was designed with the following purposes in mind: Give users perfect control over their experiments.



```
!pip install Scikit-surprise
```

```
Requirement already satisfied: Scikit-surprise in /usr/local/lib/python3.10/dist-packages (1.1.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from Scikit-surprise) (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from Scikit-surprise) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from Scikit-surprise) (1.13.1)
```

4.2 Model Building

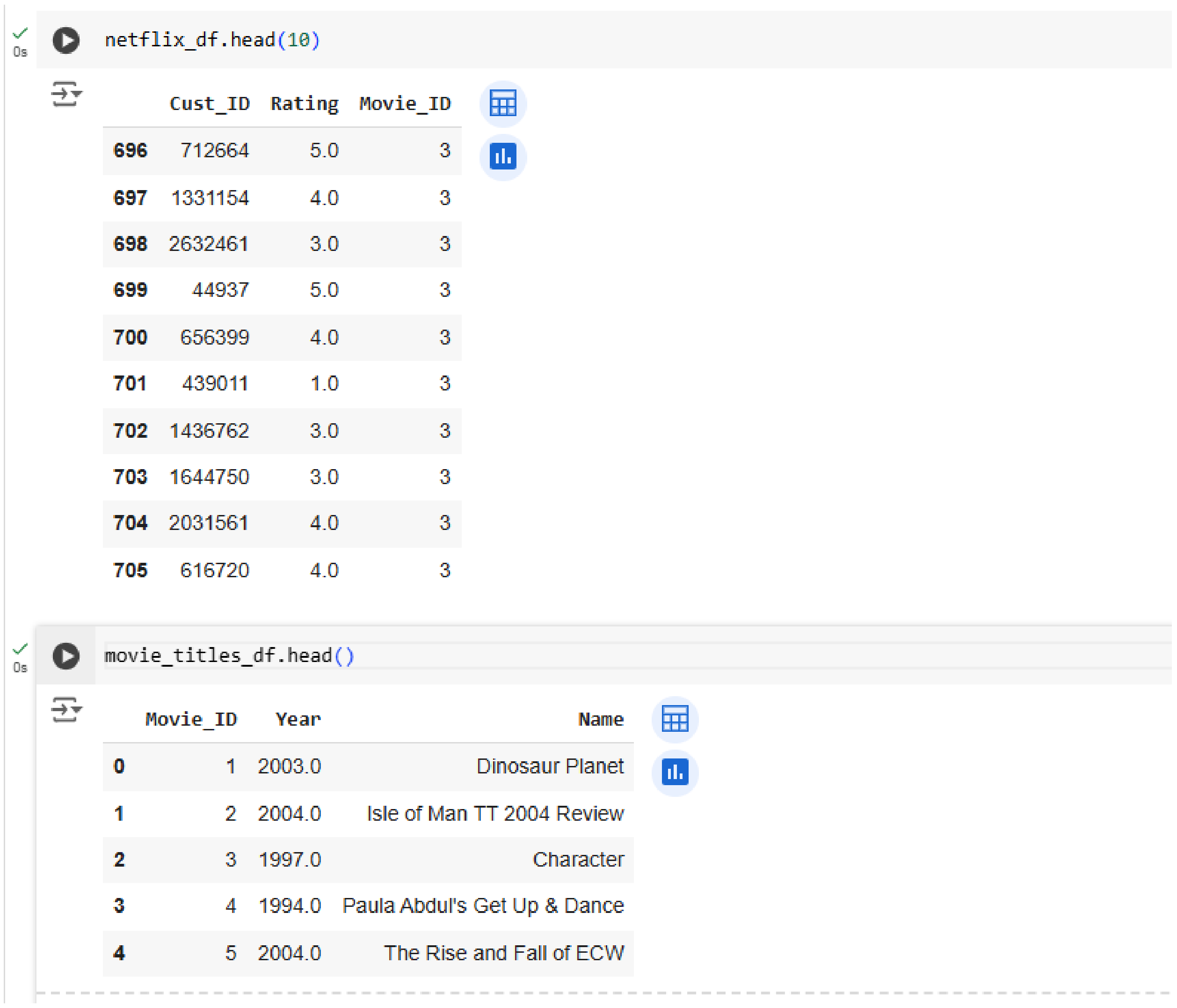


```
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
reader = Reader()
data = Dataset.load_from_df(netflix_df[['Cust_ID', 'Movie_ID', 'Rating']][:1500000], reader) #Reader is reading the data from DF line by line and send to SVD
svd = SVD()
```

```
#cross_validate(svd, data, measures=['RMSE', 'MAE'],cv=3,verbose=True)
cross_validate(svd, data, measures=['RMSE'],cv=3)
```

```
{'test_rmse': array([0.97686969, 0.97559903, 0.97505542]),
 'fit_time': (30.815246105194092, 32.32638669013977, 31.613053560256958),
 'test_time': (7.5729241371154785, 6.577130079269409, 7.027672052383423)}
```

4.3 Recommendation



```
netflix_df.head(10)
```

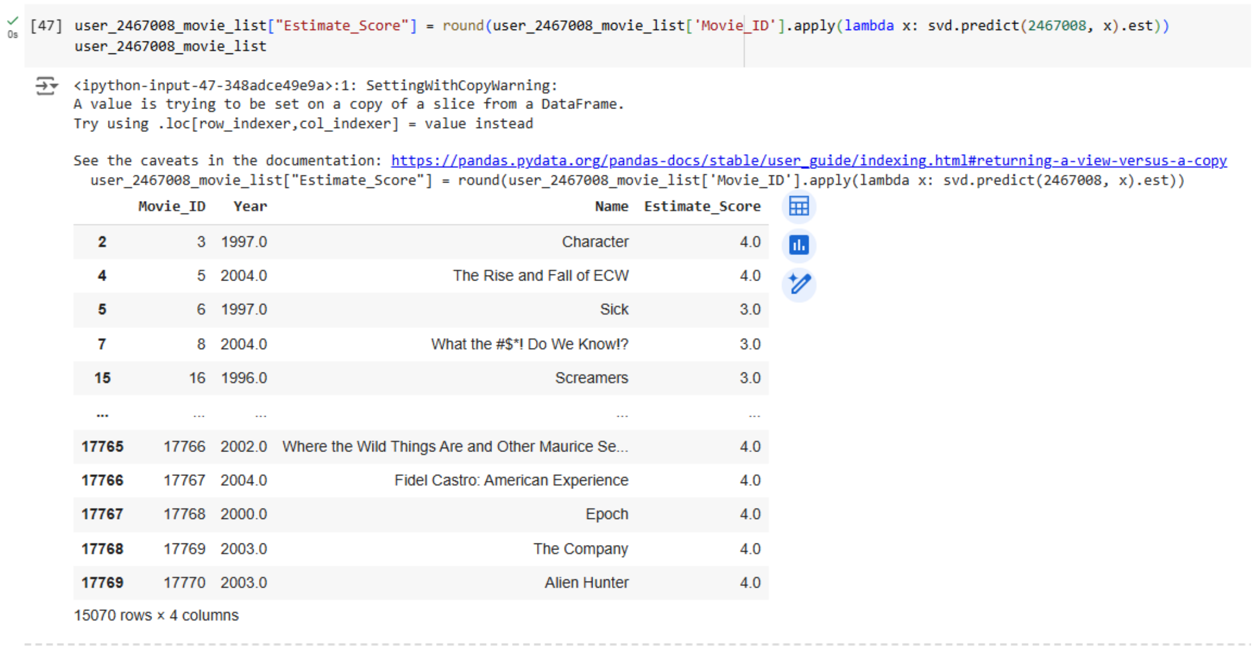
	Cust_ID	Rating	Movie_ID
696	712664	5.0	3
697	1331154	4.0	3
698	2632461	3.0	3
699	44937	5.0	3
700	656399	4.0	3
701	439011	1.0	3
702	1436762	3.0	3
703	1644750	3.0	3
704	2031561	4.0	3
705	616720	4.0	3

```
movie_titles_df.head()
```

	Movie_ID	Year	Name
0	1	2003.0	Dinosaur Planet
1	2	2004.0	Isle of Man TT 2004 Review
2	3	1997.0	Character
3	4	1994.0	Paula Abdul's Get Up & Dance
4	5	2004.0	The Rise and Fall of ECW

4.4 Estimation Score

Estimation score to see how much rating a customer to give when he/she watches a given movie.



```
[47] user_2467008_movie_list["Estimate_Score"] = round(user_2467008_movie_list['Movie_ID'].apply(lambda x: svd.predict(2467008, x).est))
     user_2467008_movie_list
```

```
<ipython-input-47-348adce49e9a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  user_2467008_movie_list["Estimate_Score"] = round(user_2467008_movie_list['Movie_ID'].apply(lambda x: svd.predict(2467008, x).est))
```

	Movie_ID	Year	Name	Estimate_Score
2	3	1997.0	Character	4.0
4	5	2004.0	The Rise and Fall of ECW	4.0
5	6	1997.0	Sick	3.0
7	8	2004.0	What the #$*! Do We Know!?	3.0
15	16	1996.0	Screamers	3.0
...	...	...	...	...
17765	17766	2002.0	Where the Wild Things Are and Other Maurice Se...	4.0
17766	17767	2004.0	Fidel Castro: American Experience	4.0
17767	17768	2000.0	Epoch	4.0
17768	17769	2003.0	The Company	4.0
17769	17770	2003.0	Alien Hunter	4.0

15070 rows × 4 columns

4.5 Customer Recommendations of top best and worst movies

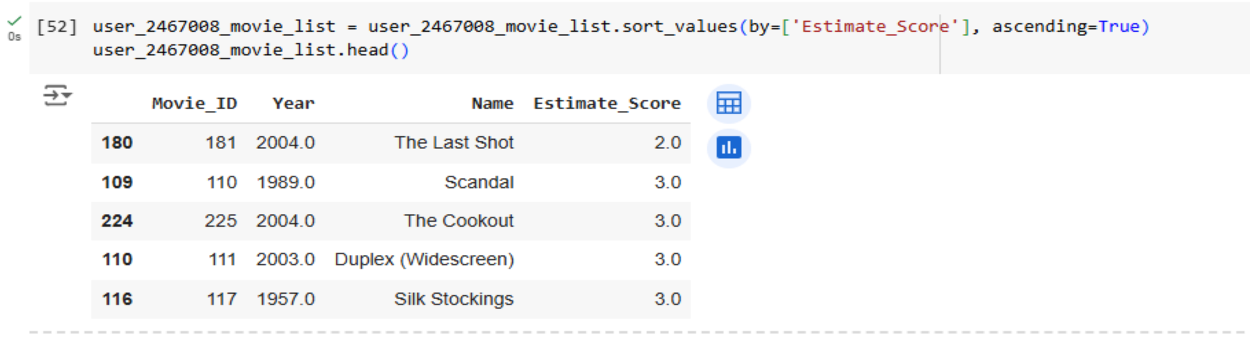
Sort values of the movies rated by 2467008 user

Create Model that finds the best suited Movie for one user in every genre.

Q: Find what Genre Movies have received the best and worst ratings based on User Rating.

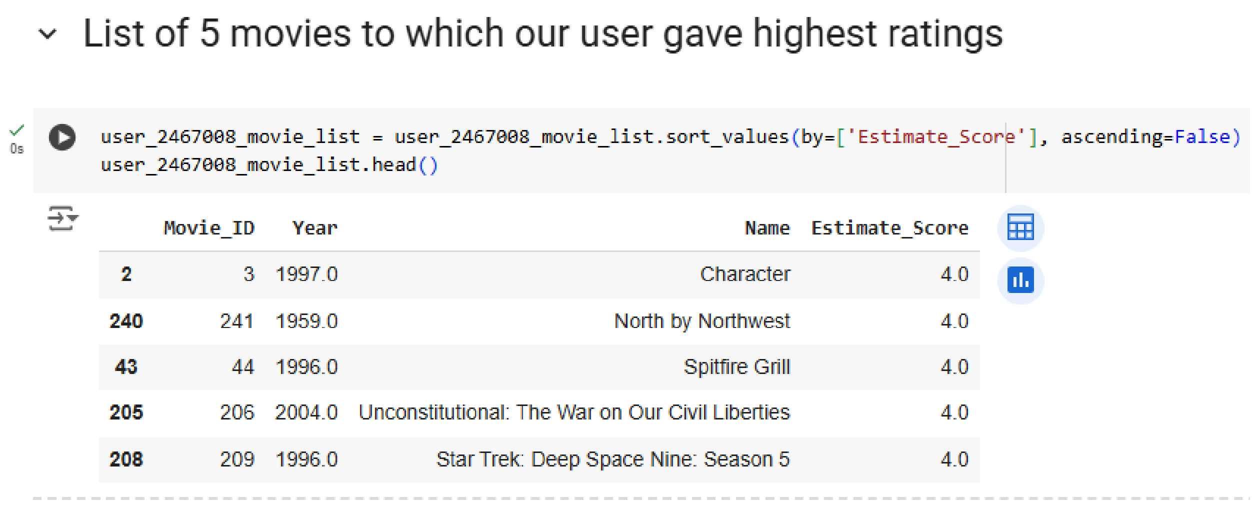
A: Below output provide this list

List of 5 movies to which our user gave worst ratings



```
[52] user_2467008_movie_list = user_2467008_movie_list.sort_values(by=['Estimate_Score'], ascending=True)
     user_2467008_movie_list.head()
```

	Movie_ID	Year	Name	Estimate_Score
180	181	2004.0	The Last Shot	2.0
109	110	1989.0	Scandal	3.0
224	225	2004.0	The Cookout	3.0
110	111	2003.0	Duplex (Widescreen)	3.0
116	117	1957.0	Silk Stockings	3.0



List of 5 movies to which our user gave highest ratings

```
user_2467008_movie_list = user_2467008_movie_list.sort_values(by=['Estimate_Score'], ascending=False)
user_2467008_movie_list.head()
```

```
     Movie_ID    Year                                      Name  Estimate_Score
2           3  1997.0                                 Character             4.0
240       241  1959.0                        North by Northwest             4.0
43         44  1996.0                              Spitfire Grill             4.0
205       206  2004.0  Unconstitutional: The War on Our Civil Liberties     4.0
208       209  1996.0              Star Trek: Deep Space Nine: Season 5     4.0
```

5.0 Conclusion

The purpose of this Capstone Project – Netflix (tv shows and movies) is the recommendation of movies and shows by a user. Based on a user's previous ratings, the model predicts how that user will recommend top or worst movies.

6.0 Acronyms

Acronym	Description
EDA	Exploratory Data Analysis
IDE	Integrated Development Environment
DF	Pandas DataFrame
Colab	Google Colaboratory IDE
IMDB	Internet Movie Database