Data Analysis of Netflix Dataset with MySQL and Pandas in Jupyter Notebook 📊 Exploring Cinematic Trends with Python and SQL.

Module 1 using pandas

Task 1: Decoding Netflix's Cinematic Data. Before we start our analysis, first, we need to view the dataset. It is essential to view the data and check the columns.

In [1]:
```
# prompt: import pandas as pd
# df=pd.read_csv('./NetflixOriginals.csv',encoding='latin-1')
import numpy as np
```

In [5]:
```
import pandas as pd
```

In [6]:
```
df = pd.read_csv(r'E:\python project\netflix data anlysis folder\NetflixOriginals raw data.csv',encoding='ISO-8859
```

Task 2: Unveiling the Datatypes in Netflix Originals Dataset.

In [7]:
```
df.dtypes
```

Out[7]:
```
Title          object
Genre          object
Premiere       object
Runtime       float64
IMDB Score    float64
Language       object
dtype: object
```

Task 3: Lowercasing Netflix Originals. We've found out the data types of the dataset. Now, we need to convert all the names into lowercase. This will be helpful when extracting column names in the upcoming tasks.

In [8]:
```
df.columns.str.lower()
```

Out[8]:
```
Index(['title', 'genre', 'premiere', 'runtime', 'imdb score', 'language'], dtype='object')
```

Task 4: Unveiling Null Values in Netflix Originals Dataset. Now, we need to ensure that the dataset does not contain any null values. So,check for null values in our dataset.

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Title         0
Genre         4
Premiere      0
Runtime       6
IMDB Score    3
Language      0
dtype: int64
```

Task 5: We've identified some null values in our data. We need to remove those to obtain cleaned data. Let's proceed by removing rows with missing entries.

In [10]:
```
df.dropna().inplace = True
```

Task 6: Tackling Duplicates in Netflix Checking for duplicate rows is crucial for maintaining data accuracy.

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
np.int64(23)
```

Task 7: Eliminating Duplicates for Netflix Insights. We've identified some duplicated values in our dataset. Let's proceed to remove these duplicate rows.

In [12]:
```
df.drop_duplicates(inplace = True)
```

Converting Netflix Premiere object to Datetime

In [13]:
```
df['Premiere'] = pd.to_datetime(df['Premiere'],errors='coerce')
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_4244\922186834.py:1: UserWarning: Could not infer format, so each el
ement will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, ple
ase specify a format.
  df['Premiere'] = pd.to_datetime(df['Premiere'],errors='coerce')
```

Task 9: Unveiling the Cinematic Epochs with Year Extraction..

In [14]:
```
df['year'] = df['Premiere'].dt.year
```

Task 10: Standardizing IMDb Score Column in Netflix data

In [15]:
```
df.rename(columns={'IMDB Score':'imdb_score'},inplace=True)
```

Task 11: Archiving Refined Netflix Originals Datase We have successfully cleaned the dataset. Now, we are ready to export it for further SQL analysis in the next module. Let's proceed with exporting the cleaned dataset now.

In [16]:
```
df.to_csv('./netflix.csv', index=False)
```

Module 2 connected mysql : Counting Cinematic Uniqueness: Distinct Titles in the Netflix Collection.

We've successfully connected the database to the Jupyter notebook, allowing us to run SQL queries directly within the notebook. Now, let's check the count of different titles from the dataset. This will provide us with the number of unique content entries in the dataset.

In [ ]:
```
pip install mysqlclient
```

In [ ]:
```
pip install ipython-sql pymysql sqlalchemy
```

In [ ]:
```
%load_ext sql
```

In [ ]:
```
%sql mysql://root:1234@127.0.0.1/movies
```

In [ ]:
```
%sql SELECT * FROM netflix;
```

Task 2: Counting Cinematic Uniqueness: Distinct Titles in the Netflix Collection. We've successfully connected the database to the Jupyter notebook, allowing us to run SQL queries directly within the notebook. Now, let's check the count of different titles from the dataset. This will provide us with the number of unique content entries in the dataset.

In [ ]:
```
df= %sql SELECT COUNT(distinct Title) AS No_of_content FROM netflix;
print(df)
```

Task 4: Counting Netflix Movies by Language. Thanks for finding the best movies and shows for me. Now, I'm interested in knowing the number of movies in each language. This task allows us to understand the distribution of movies across different languages in the dataset.

In [ ]:
```
%sql SELECT Language, COUNT(*) AS Movie_Count FROM netflix GrouP BY Language order by Movie_count desc limit 10;
```

Task 5: Unveiling the Average IMDb Scores in Netflix Originals.

In [ ]:
```
%sql Select Genre,avg(imdb_score) as Avg_Score FROM netflix group by Genre
```

Task 6: Netflix Genres with the Highest Average IMDb Scores.

In [ ]:
```
%sql select Genre,avg(imdb_score) as Avg_Score from netflix group by Genre order by Avg_Score desc limit 5;
```

Task 7: Netflix Movies Surpassing Genre IMDb Score Averages.

In [ ]:
```
%sql select * from netflix M1 join(select Genre,Avg(imdb_Score) as Avg_Score from netflix group by Genre) M2 ON M1
```

Task 8: Counting Netflix Movies by Genre Before 2020. We successfully identified the best content in their genre groups. Now, let's find the number of contents in each genre that premiered before 2020. This task allows us to analyze the distribution of movies across different genres before the year 2020.

In [ ]:
```
%sql select Genre,count(Genre)as Movie_Count from netflix where year < 2020 group by Genre;
```

Task 9: The Highest Rated Netflix Movie. We have obtained the number of movies in each genre before 2020. Now, let's find out the best content in the entire dataset. This task will help identify the top-rated show or movie. Le

In [ ]:
```
%sql select * from netflix order by imdb_score desc limit 1;
```

Task 10: Netflix Movies with Similar Premieres. let's find movies with similar genres released within a week of each other. This task helps identify movies with related genres released in close timeframe.

In [ ]:
```
%sql select A.Title as Movie1,B.Title as Movie2,A.Genre from netflix A join(select * from netflix ) B ON A.Genre =
```