Comprehensive Analysis of Netflix Movies and TV Shows Using SQL

Milestone: Implementation in MySQL

Group 08

Sree Ramya Pasala
Sri Lakshmi Swetha Jalluri

857-339-9096 (Tel of Student 1)

857-379-0666 (Tel of Student 2)

Pasala.sr@northeastern.edu

jalluri.s@northeastern.edu

Percentage of Effort Contributed by Student1: 50

Percentage of Effort Contributed by Student2: 50

Signature of Student 1:

Signature of Student 2:

Submission Date: 10-11-2024

In [2]:
```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in c:\users\91799\anaconda3\anaconda5\lib\site-packages
(9.1.0)
Note: you may need to restart the kernel to use updated packages.
WARNING: There was an error checking the latest version of pip.
```

In [4]:
```
import mysql.connector
from mysql.connector import Error

# Function to connect to MySQL
def create_connection():
    try:
        connection = mysql.connector.connect(
            host='127.0.0.1',  # Use 127.0.0.1 instead of 'localhost'
            database='analysis_of_netflix3',
            user='root',
            password='Sree2002@',
            auth_plugin='mysql_native_password'
        )
        if connection.is_connected():
            print("Connected to MySQL database")
        return connection
    except Error as e:
        print("Error while connecting to MySQL:", e)
        return None

# Store the connection object globally
connection = create_connection()
```

```
Connected to MySQL database
```

In [5]:
```
# Function to execute a SQL query
def execute_query(query):
    try:
        if connection is not None and connection.is_connected():
            cursor = connection.cursor()
            cursor.execute(query)
            results = cursor.fetchall()
            for row in results:
                print(row)
        else:
            print("Connection is not established. Please check.")
    except Error as e:
        print("Error executing query:", e)
```

In [6]:
```
execute_query("SELECT * FROM shows;")
```

```
(1, 'MOVIE', 'DICK', 'KIRSTEN', 'JOHN', 'US', datetime.date(2021, 9, 25), 2020, Decimal('4.5'), '120MINS', 'DOCUMENT
ATION')
(2, 'TV SHOW', 'SPECTRUM', 'BROOK', 'JACK', 'US', datetime.date(2014, 11, 15), 2014, Decimal('4.0'), '200MINS', 'DRA
MA')
(3, 'TV SHOW', 'CHOTA BHEEM', 'VISHAL', 'RAJIV', 'INDIA', datetime.date(2012, 6, 12), 2009, Decimal('4.8'), '200MINS
', 'DRAMA')
(4, 'MOVIE', 'JAWS', 'STEVEN', 'ROY', 'US', datetime.date(2016, 11, 15), 2016, Decimal('4.0'), '150MINS', 'ACTION')
(5, 'TV SHOW', 'HE MAN', 'ROBERT', 'MARY', 'UK', datetime.date(2017, 9, 17), 2015, Decimal('4.0'), '200MINS', 'MYSTR
Y')
(6, 'TV SHOW', 'JACK WHITE', 'NICK', 'NEEL', 'US', datetime.date(2019, 10, 12), 2019, Decimal('4.9'), '160MINS', 'CO
MEDY')
(7, 'MOVIE', 'SCHOOL', 'KONG', 'THINY', 'JAPAN', datetime.date(2018, 1, 18), 2008, Decimal('4.9'), '250MINS', 'DOCUM
ENTATION')
(8, 'TV SHOW', 'TONY', 'JACK', 'ROBERT', 'US', datetime.date(2021, 10, 19), 2021, Decimal('4.0'), '120MINS', 'SCIENT
IFIC')
(9, 'TV SHOW', 'KOTA FACTORY', 'MAYUR', 'HARRI', 'INDIA', datetime.date(2020, 11, 9), 2020, Decimal('4.6'), '1SEASON
', 'DRAMA')
(10, 'TV SHOW', 'MIDNIGHT MASS', 'MIKE', 'KATE', 'JAPAN', datetime.date(2019, 4, 8), 2019, Decimal('4.8'), '120MINS'
, 'ACTION')
(11, 'MOVIE', 'GO GO', 'ALEX', 'MAISIE', 'US', datetime.date(2021, 6, 19), 2021, Decimal('4.2'), '180MINS', 'MYSTR
Y')
(12, 'MOVIE', 'JEANS', 'SHANKAR', 'PRASH', 'INDIA', datetime.date(2010, 4, 10), 2010, Decimal('4.9'), '240MINS', 'CO
MEDY')
(13, 'TV SHOW', 'SQUID', 'ALEE', 'MAX', 'JAPAN', datetime.date(2015, 2, 14), 2015, Decimal('4.9'), '180MINS', 'TRILL
ER')
(14, 'TV SHOW', 'ANGRY BIRD', 'ANTTI', 'FINDLAND', 'UK', datetime.date(2018, 12, 17), 2012, Decimal('4.8'), '180MINS
', 'TRILLER')
(15, 'MOVIE', 'RRR', 'RAJAMOLI', 'NTR', 'INDIA', datetime.date(2020, 1, 18), 2020, Decimal('4.5'), '180MINS', 'ACTIO
N')
```

In [7]:
```
execute_query("SELECT CAST_NAME FROM shows;")
```

```
('JOHN',)
('JACK',)
('RAJIV',)
('ROY',)
('MARY',)
('NEEL',)
('THINY',)
('ROBERT',)
('HARRI',)
('KATE',)
('MAISIE',)
('PRASH',)
('MAX',)
('FINDLAND',)
('NTR',)
```

In [8]:
```
execute_query("SELECT * FROM shows WHERE RATING >= 4.5 ORDER BY RATING DESC;")
```

```
(6, 'TV SHOW', 'JACK WHITE', 'NICK', 'NEEL', 'US', datetime.date(2019, 10, 12), 2019, Decimal('4.9'), '160MINS', 'CO
MEDY')
(7, 'MOVIE', 'SCHOOL', 'KONG', 'THINY', 'JAPAN', datetime.date(2018, 1, 18), 2008, Decimal('4.9'), '250MINS', 'DOCUM
ENTATION')
(12, 'MOVIE', 'JEANS', 'SHANKAR', 'PRASH', 'INDIA', datetime.date(2010, 4, 10), 2010, Decimal('4.9'), '240MINS', 'CO
MEDY')
(13, 'TV SHOW', 'SQUID', 'ALEE', 'MAX', 'JAPAN', datetime.date(2015, 2, 14), 2015, Decimal('4.9'), '180MINS', 'TRILL
ER')
(3, 'TV SHOW', 'CHOTA BHEEM', 'VISHAL', 'RAJIV', 'INDIA', datetime.date(2012, 6, 12), 2009, Decimal('4.8'), '200MINS
', 'DRAMA')
(10, 'TV SHOW', 'MIDNIGHT MASS', 'MIKE', 'KATE', 'JAPAN', datetime.date(2019, 4, 8), 2019, Decimal('4.8'), '120MINS'
, 'ACTION')
(14, 'TV SHOW', 'ANGRY BIRD', 'ANTTI', 'FINDLAND', 'UK', datetime.date(2018, 12, 17), 2012, Decimal('4.8'), '180MINS
', 'TRILLER')
(9, 'TV SHOW', 'KOTA FACTORY', 'MAYUR', 'HARRI', 'INDIA', datetime.date(2020, 11, 9), 2020, Decimal('4.6'), '1SEASON
', 'DRAMA')
(1, 'MOVIE', 'DICK', 'KIRSTEN', 'JOHN', 'US', datetime.date(2021, 9, 25), 2020, Decimal('4.5'), '120MINS', 'DOCUMENT
ATION')
(15, 'MOVIE', 'RRR', 'RAJAMOLI', 'NTR', 'INDIA', datetime.date(2020, 1, 18), 2020, Decimal('4.5'), '180MINS', 'ACTIO
N')
```

In [9]:
```
execute_query("SELECT * FROM shows WHERE TYPE = 'MOVIE' AND DIRECTOR_NAME ='STEVEN';")
```

```
(4, 'MOVIE', 'JAWS', 'STEVEN', 'ROY', 'US', datetime.date(2016, 11, 15), 2016, Decimal('4.0'), '150MINS', 'ACTION')
```

In [10]:
```
execute_query("SELECT * FROM shows WHERE DATE_ADDED > '2020-01-01';")
```

```
(1, 'MOVIE', 'DICK', 'KIRSTEN', 'JOHN', 'US', datetime.date(2021, 9, 25), 2020, Decimal('4.5'), '120MINS', 'DOCUMENT
ATION')
(8, 'TV SHOW', 'TONY', 'JACK', 'ROBERT', 'US', datetime.date(2021, 10, 19), 2021, Decimal('4.0'), '120MINS', 'SCIENT
IFIC')
(9, 'TV SHOW', 'KOTA FACTORY', 'MAYUR', 'HARRI', 'INDIA', datetime.date(2020, 11, 9), 2020, Decimal('4.6'), '1SEASON
', 'DRAMA')
(11, 'MOVIE', 'GO GO', 'ALEX', 'MAISIE', 'US', datetime.date(2021, 6, 19), 2021, Decimal('4.2'), '180MINS', 'MYSTR
Y')
(15, 'MOVIE', 'RRR', 'RAJAMOLI', 'NTR', 'INDIA', datetime.date(2020, 1, 18), 2020, Decimal('4.5'), '180MINS', 'ACTIO
N')
```

In [11]:
```
execute_query("SELECT * FROM shows WHERE DURATION >= '180MINS';")
```

```
(2, 'TV SHOW', 'SPECTRUM', 'BROOK', 'JACK', 'US', datetime.date(2014, 11, 15), 2014, Decimal('4.0'), '200MINS', 'DRA
MA')
(3, 'TV SHOW', 'CHOTA BHEEM', 'VISHAL', 'RAJIV', 'INDIA', datetime.date(2012, 6, 12), 2009, Decimal('4.8'), '200MINS
', 'DRAMA')
(5, 'TV SHOW', 'HE MAN', 'ROBERT', 'MARY', 'UK', datetime.date(2017, 9, 17), 2015, Decimal('4.0'), '200MINS', 'MYSTR
Y')
(7, 'MOVIE', 'SCHOOL', 'KONG', 'THINY', 'JAPAN', datetime.date(2018, 1, 18), 2008, Decimal('4.9'), '250MINS', 'DOCUM
ENTATION')
(9, 'TV SHOW', 'KOTA FACTORY', 'MAYUR', 'HARRI', 'INDIA', datetime.date(2020, 11, 9), 2020, Decimal('4.6'), '1SEASON
', 'DRAMA')
(11, 'MOVIE', 'GO GO', 'ALEX', 'MAISIE', 'US', datetime.date(2021, 6, 19), 2021, Decimal('4.2'), '180MINS', 'MYSTR
Y')
(12, 'MOVIE', 'JEANS', 'SHANKAR', 'PRASH', 'INDIA', datetime.date(2010, 4, 10), 2010, Decimal('4.9'), '240MINS', 'CO
MEDY')
(13, 'TV SHOW', 'SQUID', 'ALEE', 'MAX', 'JAPAN', datetime.date(2015, 2, 14), 2015, Decimal('4.9'), '180MINS', 'TRILL
ER')
(14, 'TV SHOW', 'ANGRY BIRD', 'ANTTI', 'FINDLAND', 'UK', datetime.date(2018, 12, 17), 2012, Decimal('4.8'), '180MINS
', 'TRILLER')
(15, 'MOVIE', 'RRR', 'RAJAMOLI', 'NTR', 'INDIA', datetime.date(2020, 1, 18), 2020, Decimal('4.5'), '180MINS', 'ACTIO
N')
```

In [12]:
```
execute_query("SELECT AVG(RATING) AS avg_rating FROM shows WHERE LISTED_IN = 'ACTION';")
```

```
(Decimal('4.43333'),)
```

In [13]:
```
execute_query("SELECT MIN(RELEASE_YEAR) AS oldest_release, MAX(RELEASE_YEAR) AS newest_release FROM shows;")
```

```
(2008, 2021)
```

In [16]:
```
pip install pandas matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:\users\91799\appdata\roaming\python\python311\site-packages (2.0.3)
Requirement already satisfied: matplotlib in c:\users\91799\anaconda3\anaconda5\lib\site-packages (3.7.2)
Requirement already satisfied: seaborn in c:\users\91799\anaconda3\anaconda5\lib\site-packages (0.12.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\91799\appdata\roaming\python\python311\site-packag
es (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\91799\appdata\roaming\python\python311\site-packages (from p
andas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\91799\appdata\roaming\python\python311\site-packages (from
pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\91799\appdata\roaming\python\python311\site-packages (from
pandas) (1.25.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from matplo
tlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from matplotli
b) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from matpl
otlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from matpl
otlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\91799\appdata\roaming\python\python311\site-packages (fro
m matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from matplotli
b) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\91799\anaconda3\anaconda5\lib\site-packages (from m
atplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\users\91799\appdata\roaming\python\python311\site-packages (from pytho
n-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.

WARNING: There was an error checking the latest version of pip.
```

In [17]:
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Function to fetch data as DataFrame
def fetch_data(query):
    try:
        if connection is not None and connection.is_connected():
            return pd.read_sql(query, connection)
        else:
            print("Connection is not established.")
            return None
    except Exception as e:
        print("Error fetching data:", e)
        return None
```

In [18]:
```
query = """
SELECT LISTED_IN AS Genre, AVG(RATING) AS Avg_Rating
FROM shows
GROUP BY LISTED_IN
ORDER BY Avg_Rating DESC;
"""
df = fetch_data(query)

# Plot the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='Genre', y='Avg_Rating', palette='viridis')
plt.title("Average Rating by Genre")
plt.xticks(rotation=45, ha='right')
plt.ylabel("Average Rating")
plt.xlabel("Genre")
plt.show()
```

```
C:\Users\91799\AppData\Local\Temp\ipykernel_20784\2276930851.py:10: UserWarning: pandas only supports SQLAlchemy con
nectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not teste
d. Please consider using SQLAlchemy.
  return pd.read_sql(query, connection)
```

Average Rating by Genre
Average Rating
0
1
2
3
4
5
COMEDY
TRILLER
DOCUMENTATION
DRAMA
ACTION
MYSTRY
SCIENTIFIC
Genre

In [19]:
```
query = """
SELECT RELEASE_YEAR, COUNT(*) AS Num_Shows
FROM shows
GROUP BY RELEASE_YEAR
ORDER BY RELEASE_YEAR;
"""
df = fetch_data(query)

# Plot the line graph
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x='RELEASE_YEAR', y='Num_Shows', marker='o')
plt.title("Number of Shows Released Over Years")
plt.xlabel("Release Year")
plt.ylabel("Number of Shows")
plt.grid()
plt.show()
```

```
C:\Users\91799\AppData\Local\Temp\ipykernel_20784\2276930851.py:10: UserWarning: pandas only supports SQLAlchemy con
nectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not teste
d. Please consider using SQLAlchemy.
  return pd.read_sql(query, connection)
```

Number of Shows Released Over Years
Number of Shows
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
2008
2010
2012
2014
2016
2018
2020
Release Year

In [22]:
```
query = """
SELECT TITLE, RATING, RELEASE_YEAR
FROM shows
WHERE RATING IS NOT NULL
ORDER BY RELEASE_YEAR, RATING;
"""
df = fetch_data(query)  # Use the `fetch_data` function defined earlier
plt.figure(figsize=(10, 6))
plt.scatter(df['RELEASE_YEAR'], df['RATING'], color='blue', alpha=0.6)

# Add titles and labels
plt.title("Ratings by Release Year", fontsize=16)
plt.xlabel("Release Year", fontsize=12)
plt.ylabel("Rating", fontsize=12)

# Add grid for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```

```
C:\Users\91799\AppData\Local\Temp\ipykernel_20784\2276930851.py:10: UserWarning: pandas only supports SQLAlchemy con
nectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not teste
d. Please consider using SQLAlchemy.
  return pd.read_sql(query, connection)
```

In [23]:
```
if connection is not None and connection.is_connected():
    connection.close()
    print("MySQL connection closed.")
```

```
MySQL connection closed.
```

In [ ]: