🎬 Project 2: Streaming Service Content Performance Analysis

Scenario:
A streaming service wants to analyze the performance of its content across different regions and genres. The goal is to understand viewer preferences and content popularity over time.

Datasets:

1. User Viewing Data: User ID, content ID, watch time (in minutes), and device type (some rows may have missing device types or incorrect watch times).
2. Content Library: Content ID, title, genre, release date, and region availability (some duplicate entries and inconsistent region tags).
3. Subscription Data: User ID, subscription type, start date, renewal status (some missing renewal statuses and irregular date formats).

Tasks:

- Python: Clean the datasets by fixing incorrect timestamps and removing duplicates.
- SQL: Set up tables, import the cleaned data, and write queries to analyze genre popularity, top-watched shows per region, and subscriber retention rates.
- Power BI: Create dashboards to visualize user engagement, genre popularity by region, and subscription trends over time.

Phases & Steps:

Phase 1: Data Cleaning with Python

1. Import datasets using Pandas.

```python
# a. import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
# b. loading the dataset into pandas
library=pd.read_csv('streaming_content_library.csv')
subscription=pd.read_csv('streaming_subscription_data.csv')
viewing=pd.read_csv('streaming_user_viewing_data.csv')
```

1.User Viewing Data: User ID, content ID, watch time (in minutes), and device type (some rows may have missing device types or incorrect watch times).

1. inspecting the data

```
# a.first dataset on viewing
viewing=pd.read_csv('streaming_user_viewing_data.csv')
```

```
# inspecting the data shape
viewing.shape
```

```
(3060, 5)
```

```
# Checking the structure of the viewing dataset
print(viewing.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3060 entries, 0 to 3059
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Viewing_ID          2908 non-null   float64
 1   User_ID             2906 non-null   float64
 2   Content_ID          2908 non-null   float64
 3   Watch_Time_Minutes  2907 non-null   float64
 4   Device_Type         2905 non-null   object 
dtypes: float64(4), object(1)
memory usage: 119.7+ KB
None
```

```
# Checking the first 10 rows of the viewing dataset
print(viewing.head())
```

```
   Viewing_ID  User_ID  Content_ID  Watch_Time_Minutes Device_Type
0         1.0    159.0      1565.0               185.0          TV
1         2.0    710.0      1636.0                94.0          PC
2         3.0   2700.0      1012.0               125.0      Tablet
3         NaN   1594.0      1851.0               128.0          PC
4         5.0   2153.0      1999.0               300.0      Mobile
```

2. Identify and correct invalid timestamps.

a. Correcting the timestamp

```
# 2.    Identify and correct invalid timestamps.
```

```
clean_viewing1 = pd.DataFrame(clean_viewing1)
```

```
clean_viewing1["Watch_Time_Minutes"] = pd.to_numeric(clean_viewing1["Watch_Time_Minutes"], errors="coerce")
```

```
clean_viewing1["Watch_Time_Hours"] = clean_viewing1["Watch_Time_Minutes"] / 60
```

```
clean_viewing1 = clean_viewing1.drop(columns=["Watch_Time_Minutes"])
```

```
print(clean_viewing1.iloc[ :4])
```

```
   Viewing_ID  User_ID  Content_ID Device_Type  Watch_Time_Hours
0         1.0    159.0      1565.0          TV          3.083333
1         2.0    710.0      1636.0          PC          1.566667
2         3.0   2700.0      1012.0      Tablet          2.083333
4         5.0   2153.0      1999.0      Mobile          5.000000
```

identifying missing values in the viewing dataset

```
# count of missing values in the viewing dataset
viewing.isna().sum()
```

```
Viewing_ID           152
User_ID              154
Content_ID           152
Watch_Time_Minutes   153
Device_Type          155
dtype: int64
```

```
# droping all the null values from the viewing_ID
clean_viewing1= viewing.dropna(subset=['Viewing_ID','User_ID','Content_ID', 'Watch_Time_Minutes','Device_Type'])
```

```
clean_viewing1.isna().sum()
```

```
Viewing_ID           0
User_ID              0
Content_ID           0
Watch_Time_Minutes   0
Device_Type          0
dtype: int64
```

c. Remove duplicate content entries.

```
# 3.    Remove duplicate content entries.
# a. checking for duplicated from the viewing data set
clean_viewing1.duplicated().sum()
```

```
46
```

```
# b. removing duplicate values from the dataset
clean_viewing1.drop_duplicates(inplace=True)
```

```
# c. verifying whether we duplicates values are removed from the viewing dataset
clean_viewing1.duplicated().sum()
```

```
0
```

- All the duplicate values have been removed from the viewing

d. Save cleaned data to new CSV files.

```
# 5.    Save cleaned data to new CSV files.
```

```
clean_viewing1.to_csv("update2_cleaned01_viewing.csv", index=False)
```

- First dataset for viewing fully cleaned

2. Content Library: Content ID, title, genre, release date, and region availability (some duplicate entries and inconsistent region tags).\

```
library=pd.read_csv('streaming_content_library.csv')
```

```
# inspecting the first 10 row in the dataset
library.head()
```

	Content_ID	Title	Genre	Release_Date	Region_Availability
0	1000.0	Experience.	Drama	2021-08-24	Canada
1	1001.0	Child manager article.	Comedy	2021-06-28	Australia
2	1002.0	Parent keep.	Drama	2016-08-20	Canada
3	1003.0	Cold case science.	Comedy	2015-08-09	UK
4	1004.0	Up either cover question.	Comedy	2024-08-26	Australia

```
# inspecting the data shape
library.shape
```

```
(3060, 5)
```

```
# Checking the structure of the library dataset
library.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3060 entries, 0 to 3059
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Content_ID           2909 non-null   float64
 1   Title                2905 non-null   object 
 2   Genre                2909 non-null   object 
 3   Release_Date         2904 non-null   object 
 4   Region_Availability  2906 non-null   object 
dtypes: float64(1), object(4)
memory usage: 119.7+ KB
```

1. Identifying missing values and removing them

```
# all the columns have missing values
library.isna().any()
```

```
Content_ID            True
Title                 True
Genre                 True
Release_Date          True
Region_Availability   True
dtype: bool
```

```
# count of missing values in the viewing dataset
library.isna().sum()
```

```
Content_ID            151
Title                 155
Genre                 151
Release_Date          156
Region_Availability   154
dtype: int64
```

```
# droping the missing values
library1 = library.dropna(subset=['Content_ID','Title','Genre','Release_Date','Region_Availability'])
```

```
library1.isnull().sum()
```

```
Content_ID            0
Title                 0
Genre                 0
Release_Date          0
Region_Availability   0
dtype: int64
```

```
library1.shape
```

```
(2364, 5)
```

```
library1.head(5)
```

	Content_ID	Title	Genre	Release_Date	Region_Availability
0	1000.0	Experience.	Drama	2021-08-24	Canada
1	1001.0	Child manager article.	Comedy	2021-06-28	Australia
2	1002.0	Parent keep.	Drama	2016-08-20	Canada
3	1003.0	Cold case science.	Comedy	2015-08-09	UK

2. Remove duplicate content entries.

```
# 3.    Remove duplicate content entries.
# a. checking for duplicated from the library data set
library.duplicated().sum()
```

```
60
```

```
library1.drop_duplicates(inplace=True)
```

```
library1.duplicated().sum()
```

```
0
```

3. Normalize inconsistent genre labels and region tags.

```
# 4. Normalize inconsistent genre labels and region tags.
```

```
library1['Region_Availability'].unique()
```

```
array(['Canada', 'Australia', 'UK', 'US', 'Global'], dtype=object)
```

```
library1 = pd.DataFrame(library1)
```

```
# normalizing the Genre,Removing space and standardize capitalization
library1["Genre"] = library1["Genre"].str.strip().str.title()
```

```
# Normalizing Region availability tag?
region = {"UK": "United Kingdom", "CANADA": "Canada","canada": "Canada", "australia": "Australia",'US': 'United States'}
```

```
library1["Region_Availability"] = library1["Region_Availability"].replace(region)
```

4. Save cleaned data to new CSV files.

```
library1.to_csv("update1_Library1_clean", index=False)
```

3. Subscription Data: User ID, subscription type, start date, renewal status (some missing renewal statuses and irregular date formats).

```
subscription=pd.read_csv('streaming_subscription_data.csv')
```

```
# inspecting the first 10 row in the dataset
print(subscription.head())
```

```
   Subscription_ID  User_ID Subscription_Type  Start_Date Renewal_Status
0              1.0   1275.0               NaN  2023-06-03      Cancelled
1              2.0   1737.0           Premium  2022-08-19         Active
2              3.0   2645.0           Premium  2024-08-25      Cancelled
3              4.0    551.0           Premium  2022-04-28         Active
4              5.0   1284.0          Standard  2023-10-11      Cancelled
```

```
# inspecting the data shape
subscription.shape
```

```
(3060, 5)
```

```
# count of missing values in the viewing dataset
subscription.isna().sum()
```

```
Subscription_ID      155
User_ID              151
Subscription_Type    152
Start_Date           154
Renewal_Status       150
dtype: int64
```

```
# Creating subscrition DataFrame
subscription1 = pd.DataFrame(subscription)
```

```
print(subscription1[ : 5] )
```

```
   Subscription_ID  User_ID Subscription_Type  Start_Date Renewal_Status
0              1.0   1275.0               NaN  2023-06-03      Cancelled
1              2.0   1737.0           Premium  2022-08-19         Active
2              3.0   2645.0           Premium  2024-08-25      Cancelled
3              4.0    551.0           Premium  2022-04-28         Active
4              5.0   1284.0          Standard  2023-10-11      Cancelled
```

```python
 # a. Handle Missing Values
# i) missing values in user_Id replaced by -1
subscription1.fillna({"subscription_ID": "-1"}, inplace=True)
```

```python
# ii)  missing values in subscription_type replaced with 'Unknown'
subscription1.fillna({"Subscription_Type": "Unknown"}, inplace=True)
```

```python
print(subscription1[ : 5] )
```

```
   Subscription_ID  User_ID Subscription_Type  Start_Date Renewal_Status
0              1.0   1275.0           Unknown  2023-06-03      Cancelled
1              2.0   1737.0           Premium  2022-08-19         Active
2              3.0   2645.0           Premium  2024-08-25      Cancelled
3              4.0    551.0           Premium  2022-04-28         Active
4              5.0   1284.0          Standard  2023-10-11      Cancelled
```

```python
# b. Convert Start_Date to Datetime Format
subscription1["Start_Date"] = pd.to_datetime(subscription1["Start_Date"], errors="coerce")
```

```python
# b. Convert Start_Date to Datetime Format
subscription1["Start_Date"] = pd.to_datetime(subscription1["Start_Date"], errors="coerce")
```

```python
# c. Validate Subscription Type
valid_subscription_types = ["Basic", "Standard", "Premium", "Unknown"]
```

```python
# Fixing invalid subscription types
subscription1.loc[~subscription1["Subscription_Type"].isin(valid_subscription_types), "Subscription_Type"] = "Unknown"
```

```python
# d. Validating the Renewal Status in the subscription
valid_renewal_statuses = ["Active", "Cancelled", "Expired"]
```

```python
# fixing the invalid statuses
subscription1.loc[~subscription1["Renewal_Status"].isin(valid_renewal_statuses), "Renewal_Status"] = "Unknown"  # Fix invalid statuses
```

```python
# e. Removing Duplicates from the subscription dataset**
subscription1.drop_duplicates(inplace=True)
```

```python
# f. Saving the subscription clean dataset
subscription1.to_csv("updated2_subscription1_data.csv", index=False)
```

```python
# **7. Display Cleaned Data**
print(subscription1.head(50))
```

Note

- A connection between the mysql database and note book was create .
- The data was then imported to mysql using sqlachemy.

Phase 2: SQL Database Integration

Phase 2: SQL Database Integration

1. Set up a database.

```sql
/* create database for the streaming service */
create database media;

/*Select the databse to use*/
use media;
```

2. Create tables for User Viewing Data, Content Library, and Subscription Data.

```sql
/*2.    Create tables for User Viewing Data, Content Library, and Subscription Data.*/
/*creating tables for the database*/

create table library(
Content_ID bigint primary key,
Title varchar(50) NOT NULL,
Genre varchar(50) NOT NULL,
Release_Date date NOT NULL,
Region_availability varchar(50) NOT NULL
);

create table subscription(
Subscription_ID int NOT NULL,
User_ID bigint NOT NULL,
Subscription_Type varchar(50)NOT NULL,
Start_Date Date ,
Renewal_Status varchar(50) NOT NULL

);
create table viewing(
Viewing_ID int NOT NULL,
User_ID bigint NOT NULL,
Content_ID bigint NOT NULL,
Watch_time timestamp NULL NULL,
Dervice_type varchar(50) NOT NULL
);
```

3. Import cleaned data into the database.

4. Run SQL queries:
    - Top-watched genres by region.

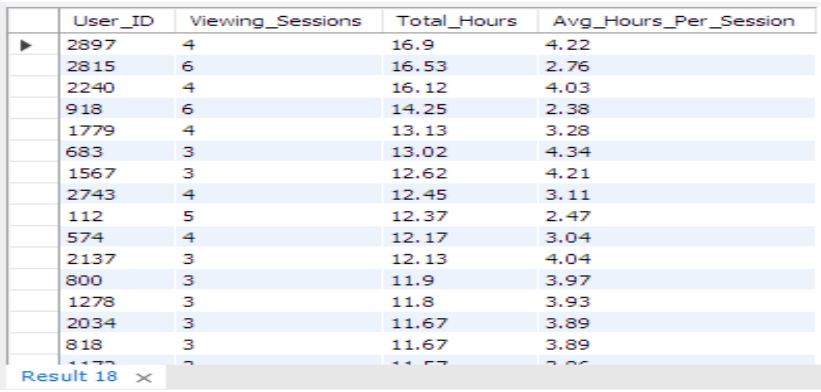
```sql
SELECT Genre, Region_Availability AS region, COUNT(*) AS views_count,
RANK() OVER (PARTITION BY Region_Availability ORDER BY COUNT(*) DESC) AS genre_rank
FROM library
WHERE Genre IS NOT NULL
AND Region_Availability IS NOT NULL
GROUP BY Region_Availability, Genre
ORDER BY genre_rank DESC limit 10;
```

Genre	region	views_count	genre_rank
Sci-Fi	Australia	61	6
Comedy	Canada	68	6
Comedy	Global	63	6
Action	United Kingdom	74	6
Sci-Fi	United States	61	6
Drama	Australia	65	5
Thriller	Canada	74	5
Action	Global	68	5
Documentary	United Kingdom	75	5
Comedy	United States	73	5

    - Viewer retention analysis.

```sql
SELECT User_ID,COUNT(*) AS Viewing_Sessions,Round(SUM(Hours),2) AS
Total_Hours,Round(AVG(Hours),2) AS Avg_Hours_Per_Session
FROM viewing
WHERE User_ID IS NOT NULL
GROUP BY User_ID
```

ORDER BY Total_Hours DESC;



User_ID	Viewing_Sessions	Total_Hours	Avg_Hours_Per_Session
2897	4	16.9	4.22
2815	6	16.53	2.76
2240	4	16.12	4.03
918	6	14.25	2.38
1779	4	13.13	3.28
683	3	13.02	4.34
1567	3	12.62	4.21
2743	4	12.45	3.11
112	5	12.37	2.47
574	4	12.17	3.04
2137	3	12.13	4.04
800	3	11.9	3.97
1278	3	11.8	3.93
2034	3	11.67	3.89
818	3	11.67	3.89

Result 18

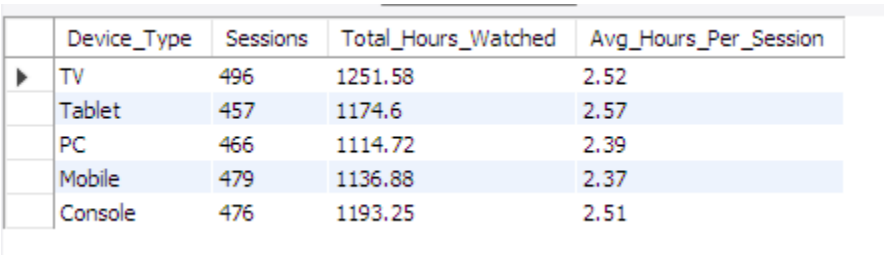
Device Type Usage Summary

SELECT Device_Type, COUNT(*) AS Sessions,Round(SUM(Hours),2) AS Total_Hours_Watched,Round(AVG(Hours),2) AS Avg_Hours_Per_Session

FROM Viewing

GROUP BY Device_Type

ORDER BY Device_Type DESC LIMIT 1000;



Device_Type	Sessions	Total_Hours_Watched	Avg_Hours_Per_Session
TV	496	1251.58	2.52
Tablet	457	1174.6	2.57
PC	466	1114.72	2.39
Mobile	479	1136.88	2.37
Console	476	1193.25	2.51

- Monthly subscriber growth.

SELECT DATE_FORMAT(Start_Date, '%Y-%m') AS Month, COUNT(*) AS New_Subscriptions

FROM Subscription

GROUP BY Month

ORDER BY Month;

Month	New_Subscriptions
2022-02	20
2022-03	78
2022-04	69
2022-05	68
2022-06	79
2022-07	70
2022-08	72
2022-09	67
2022-10	77
2022-11	66
2022-12	65

Result 25

Phase 3: Data Visualization with Power BI

1. Connect Power BI to the database.
2. Visualize:
    - Viewer engagement across devices.
    - Regional genre popularity.
    - Subscription trends and churn rates.