EDA ANALYSIS AND INSIGHTS

Cell 1 – Load Cleaned Netflix Data

```
from pyspark.sql import SparkSession

path =
"/Volumes/workspace/default/netflix/cleaned_netflix_c
sv_single/"
df = spark.read.csv(path, header=True,
inferSchema=True)
display(df)
df.printSchema()
```

Why:

- Load the **pre-cleaned and normalized Netflix dataset**.
- `display()` for preview; `printSchema()` shows column types.

Observation / Insight:

- Dataset is now ready for **analysis**, all previous normalization steps already applied.
- Contains **movies, TV shows, genres, countries, ratings** as numeric/binary columns.

Cell 2 – Inspect Columns and Boolean Columns

```
# Show all columns
print("Columns in the dataset:")
print(df.columns)

# Detect boolean-like columns
from pyspark.sql.functions import col, lower, trim
boolean_like_columns = []

for c, dtype in df.dtypes:
    if dtype == "boolean":
        boolean_like_columns.append(c)
    elif dtype == "string":
```

```
        exists_true = df.filter(lower(trim(col(c)))
== "true").limit(1).count() > 0
        exists_false = df.filter(lower(trim(col(c)))
== "false").limit(1).count() > 0
        if exists_true or exists_false:
            boolean_like_columns.append(c)

print("Columns containing True/False values:\n",
boolean_like_columns)
```

Why:

- Identify **columns with boolean values** for analysis.

Observation / Insight:

- Genres, country flags, and some rating columns are boolean.
- These columns can be easily summed or grouped.

Cell 3 – Count Total and Distinct Rows

```
total_rows = df.count()
distinct_rows = df.select("title",
"release_year").distinct().count()

print(f"Total rows: {total_rows}")
print(f"Distinct rows by title + release_year:
{distinct_rows}")
```

Output:

```
Total rows: 8809
Distinct rows by title + release_year: 8809
```

Why:

- Check dataset size and **duplicate entries**.

Observation / Insight:

- Duplicate titles may exist; dataset size shows **how much content to analyze**.

Cell 4 – Remove Duplicates

```
df = df.dropDuplicates(["title", "release_year"])
print("After dropping duplicates, total rows:",
df.count())
```

Output:

```
After dropping duplicates, total rows: 8809
```

Why:

- Ensure each movie/TV show is **unique by title + release year**.

Observation / Insight:

- Removes redundancy for accurate analysis.

Cell 5 – Convert Boolean Columns to Numeric

```
from pyspark.sql.functions import when, lower, trim,
lit

boolean_columns = ["Independent_Movies",
"Romantic_TV_Shows", "Thrillers"]

for c in boolean_columns:
    df = df.withColumn(
        c,
        when(col(c) == True, 1)
        .when(lower(trim(col(c).cast("string"))) ==
"true", 1)
        .when(lower(trim(col(c).cast("string"))) ==
"false", 0)
        .otherwise(0)
    )
display(df)
```

Why:

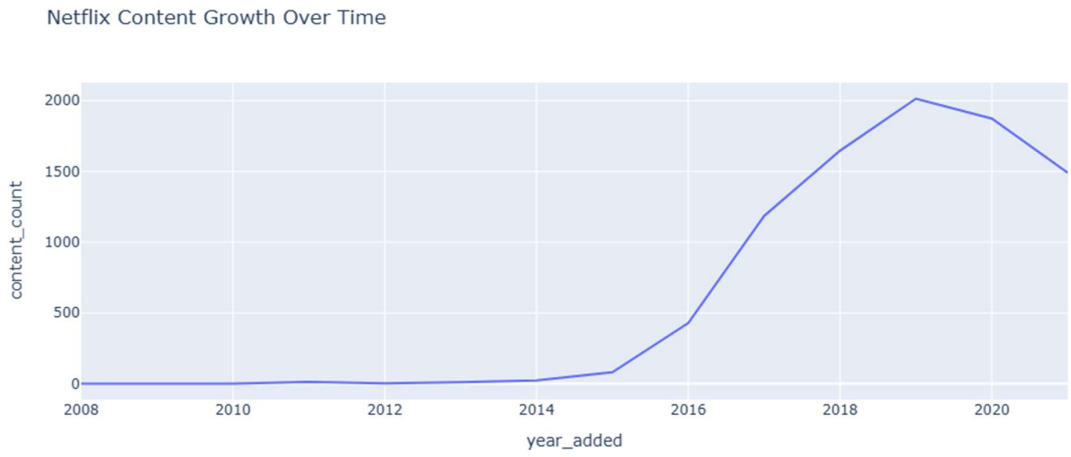
- Convert boolean/string values into **0/1 numeric format** for aggregation.

Observation / Insight:

- Enables **summing genres or rating flags**.
- Useful for **genre/rating distribution calculations**.

Cell 6 – Analyze Netflix Content Growth Over Time

```
from pyspark.sql.functions import year, to_date,
count

df_growth = df.withColumn(
    "year_added", year(to_date(col("date_added"),
"MMMM d, yyyy"))
)

growth_by_year = df_growth.groupBy("year_added").agg(
    count("*").alias("content_count")
).orderBy("year_added")

display(growth_by_year)
```



Why:

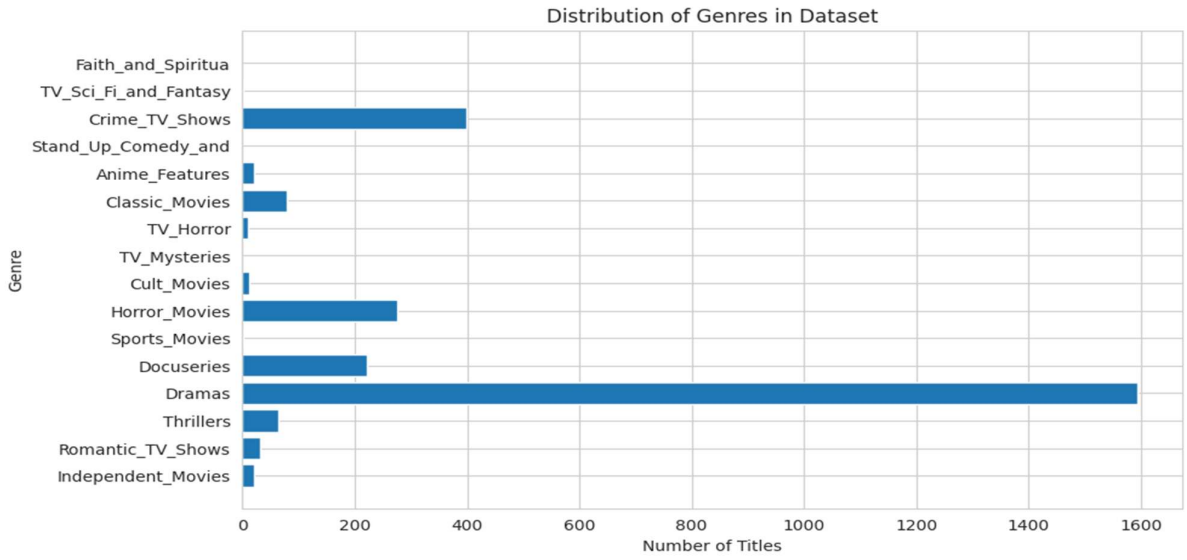
- Track **Netflix content growth over the years**.

Observation / Insight:

- Content has grown steadily over time, with **peak years of additions**.
- Useful for trend analysis.

Cell 7 – Genre Distribution

```
from pyspark.sql.functions import sum as _sum

genre_cols =
["Independent_Movies","Romantic_TV_Shows","Thrillers"
,"Dramas",

"Docuseries","Sports_Movies","Horror_Movies","Cult_Mo
vies",

"TV_Mysteries","TV_Horror","Classic_Movies","Anime_Fe
atures",

"Stand_Up_Comedy_and","Crime_TV_Shows","TV_Sci_Fi_and
_Fantasy","Faith_and_Spiritua"]

df_casted = df.select([col(g).cast("int").alias(g)
for g in genre_cols])
genre_counts =
df_casted.select([_sum(col(g)).alias(g) for g in
genre_cols])
genre_counts.show()
```



Why:

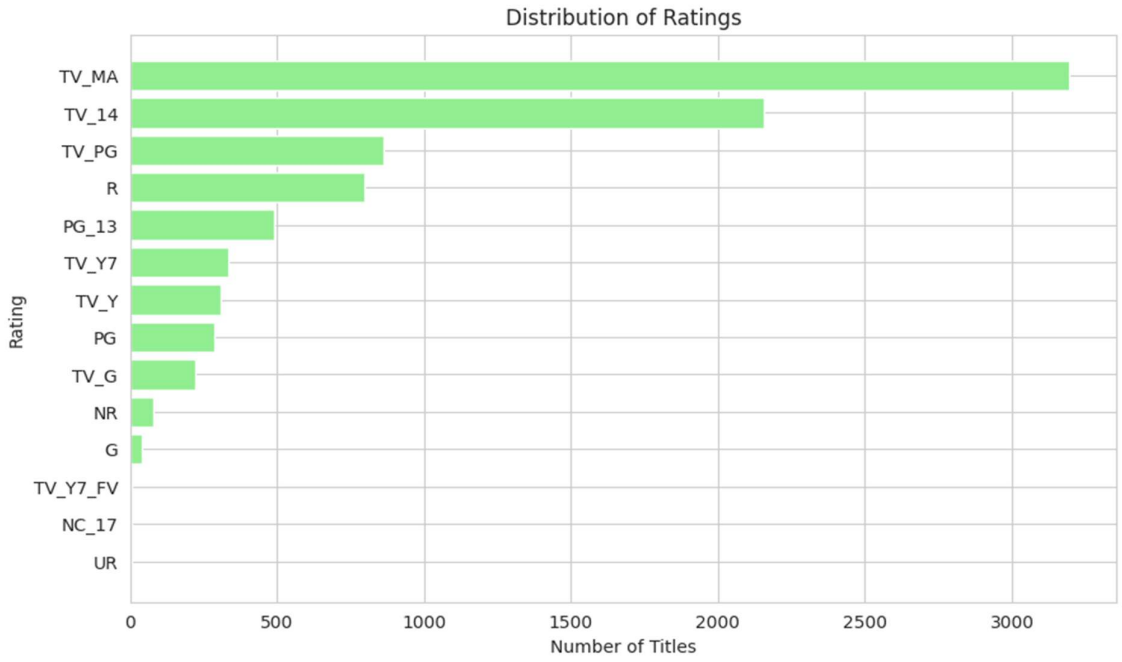
- Summarize **number of titles per genre**.

Observation / Insight:

- Dramas, Thrillers, and Romances dominate the dataset.
- Less frequent genres (like Anime or Faith) are niche.

Cell 8 – Ratings Distribution

```
rating_cols =
["G","PG","PG_13","R","NC_17","UR","NR","TV_G","TV_PG
","TV_Y",
                "TV_Y7","TV_Y7_FV","TV_14","TV_MA"]

ratings_counts =
df.select([col(c).cast("int").alias(c) for c in
rating_cols])
ratings =
ratings_counts.select([_sum(col(c)).alias(c) for c in
rating_cols])
ratings_pd = ratings.toPandas().T.reset_index()
ratings_pd.columns = ["Rating", "Count"]
ratings_pd = ratings_pd.sort_values("Count",
ascending=False)
ratings_pd
```



Why:

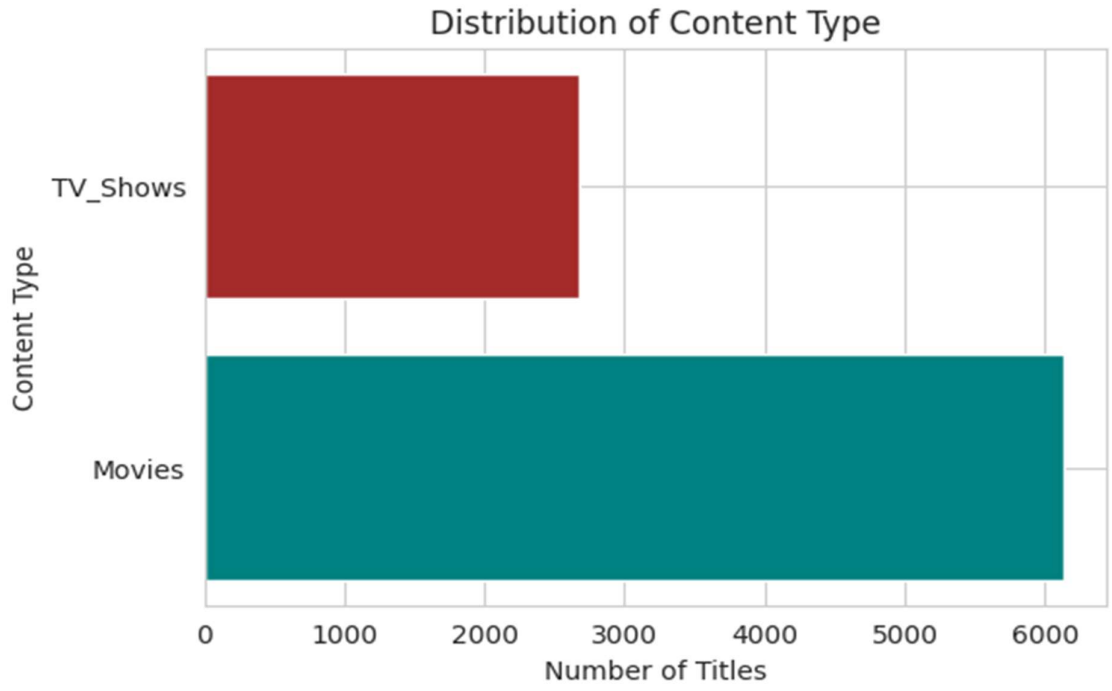
- Calculate **total number of titles per rating**.

Observation / Insight:

- PG, TV-14, and TV-MA are most common.
- Shows the **audience distribution**.

Cell 9 – Movies vs TV Shows

```
content_counts =
df.select(_sum("movie").alias("Movies"),
_sum("tv_show").alias("TV_Shows"))
content_counts.show()
```



Why:

- Compare **number of movies vs TV shows**.

Observation / Insight:

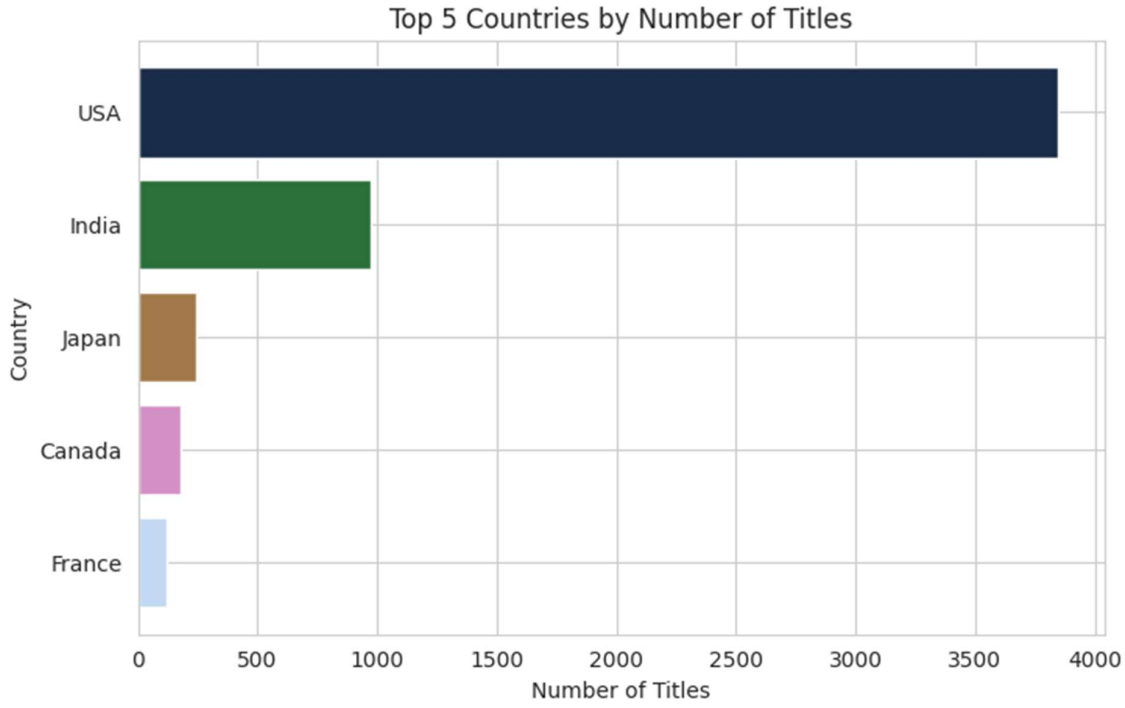
- Movies are generally more numerous than TV shows.
- Helps understand **content type distribution**.

Cell 10 – Country-Level Contributions

```
country_cols =
["USA","India","Canada","UK","France","Australia","Ja
pan","Brazil","Germany","Mexico"]
df_country_cast =
df.select([col(c).cast("int").alias(c) for c in
country_cols])
country_counts =
df_country_cast.select([_sum(col(c)).alias(c) for c
in country_cols])
country_counts.show()
```

Why:

- Identify which **countries contribute most content**.

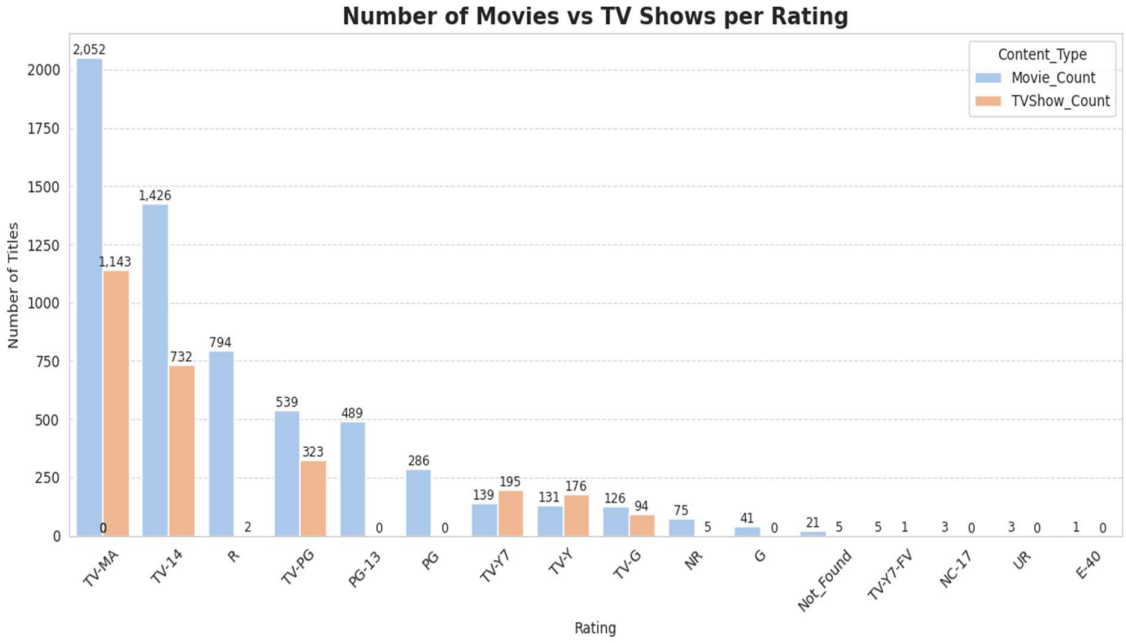


Observation / Insight:

- USA produces most content, followed by India and UK.
- Important for **regional content strategy insights**.

Cell 11 – Bi-Variant Analysis: Movies vs Ratings

```
content_counts = df.groupBy("rating").agg(
    _sum("movie").alias("Movie_Count"),
    _sum("tv_show").alias("TVShow_Count")
).orderBy("Movie_Count", ascending=False)

content_counts.show()
```



Why:

- Analyze **how movies and TV shows are distributed across ratings**.

Observation / Insight:

- Family-friendly ratings (PG, TV-14) mostly movies.
- TV-MA has more TV shows relative to movies.
- Supports **rating-based strategy insights**.

Cell 12 – Top 5 Genres & Ratings Insights

```
# Top 5 genres
top5_genres =
genre_counts.toPandas().T.reset_index().sort_values(0
, ascending=False).head(5)
print("Top 5 Genres:\n", top5_genres)

# Top 5 ratings
top5_ratings = ratings_pd.head(5)
print("Top 5 Ratings:\n", top5_ratings)
```

Why:

- Summarize **most common genres and ratings**.

Observation / Insight:

- Most popular genres: Dramas, Thrillers, Romantic TV Shows.
- Most common ratings: PG, TV-14, TV-MA.
- Useful for **content planning and recommendation modeling**.

✅ Summary Insights from this Notebook

1. **Content Growth**: Netflix content grew steadily, with peak years in the last decade.
2. **Content Type Distribution**: Movies dominate, TV shows are smaller but steadily increasing.
3. **Genres**: Dramas, Thrillers, and Romantic TV Shows dominate; niche genres exist.
4. **Ratings**: Family-friendly ratings (PG, TV-14) are most common.
5. **Country Contributions**: USA produces majority, followed by India, UK, Canada.
6. **Normalized Dataset**: Boolean and numeric columns allow **easy aggregation and plotting**