Netflix Movies and TV Shows Analysis

By: Jessie Estrada

Data Cleaning & Transformation

To prepare the Netflix dataset for analysis, I performed SQL based normalization using MySQL Workbench. The original data was highly denormalized, with multiple values stored in separate columns for categories, countries, directors, and cast members. I converted these multicolumn formats into single column, multi row structures for more effective filtering, joining, and visual reporting in Power BI.

Summary of Normalized Tables

Table Name	Description
netflix_listed_in	Flattened 3 listed_in columns into a single category column
countries_released	Unpivoted 12 country columns into one country column
netflix_directors	Combined 13 director columns into one director column
netflix_cast	Merged 50 separate cast columns into a single cast column

SQL Transformations

netflix_listed_in

This query consolidates category information from the listed in table. The original data contained up to three category columns per show. This transformation unifies them into a single column called category.

```
CREATE TABLE netflix_data.netflix_listed_in AS
(
  SELECT *
  FROM (
    SELECT show_id, listed_in_1 AS category FROM netflix_data.`listed in`
    UNION
    SELECT show_id, listed_in_2 AS category FROM netflix_data.`listed in`
    UNION
    SELECT show_id, listed_in_3 AS category FROM netflix_data.`listed in`
  ) a
  WHERE category IS NOT NULL
);
```

Countries_released

This query normalizes country data from the countries table. The original table contained up to 12 Country_n columns. This transformation produces one row per country per show.

```
CREATE TABLE netflix_data.countries_released AS

(
  SELECT *
  FROM (
    SELECT show_id, Country_1 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_2 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_3 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_4 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_5 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_6 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_7 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_8 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_9 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_10 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_11 AS country FROM netflix_data.countries
    UNION
    SELECT show_id, Country_12 AS country FROM netflix_data.countries
  ) a
  WHERE country IS NOT NULL
);
```

netflix_directors

This query restructures director data from the directors table. The original table stored up to 13 separate columns for directors. The resulting table links each show_id to a single director per row.

```
CREATE TABLE netflix_data.netflix_directors AS
(
  SELECT *
  FROM (
    SELECT show_id, director_1 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_2 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_3 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_4 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_5 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_6 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_7 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_8 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_9 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_10 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_11 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_12 AS director FROM netflix_data.directors
    UNION
    SELECT show_id, director_13 AS director FROM netflix_data.directors
  ) a
  WHERE director IS NOT NULL
);
```

netflix_cast

This query flattens cast information from the cast table. The original data included up to **50 cast columns** per show. The result is a normalized structure where each cast member is listed in a separate row alongside their corresponding show_id.

```
CREATE TABLE netflix_data.netflix_cast AS

(
  SELECT *
  FROM (
    SELECT show_id, cast_1 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_2 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_3 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_4 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_5 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_6 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_7 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_8 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_9 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_10 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_11 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_12 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_13 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_14 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_15 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_16 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_17 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_18 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_19 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_20 AS cast FROM netflix_data.cast
    UNION
```

```
SELECT show_id, cast_21 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_22 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_23 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_24 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_25 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_26 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_27 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_28 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_29 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_30 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_31 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_32 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_33 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_34 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_35 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_36 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_37 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_38 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_39 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_40 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_41 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_42 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_43 AS cast FROM netflix_data.cast
UNION
SELECT show_id, cast_44 AS cast FROM netflix_data.cast
UNION
```

```
    SELECT show_id, cast_45 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_46 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_47 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_48 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_49 AS cast FROM netflix_data.cast
    UNION
    SELECT show_id, cast_50 AS cast FROM netflix_data.cast
  ) a
  WHERE cast IS NOT NULL
);
```

Data Structuring in Excel

After performing SQL based normalization, I exported the data to Excel for additional preparation and organization. I split the dataset into six logically separated CSV files based on entity type. Each file contains a subset of information related to Netflix titles, with show_id acting as the unique identifier across all files to maintain referential integrity.

Structured Files

File Name	Description
netflix_titles.csv	Main fact table containing title metadata (type, title, release year, etc.)
cast.csv	One row per cast member per show (show_id, cast)
descriptions.csv	One row per show_id containing the description text
countries.csv	One row per show country combination (show_id, country)
listed_in.csv	One row per show category combination (show_id, category)
directors.csv	One row per director per show (show_id, director)

Key Relationships

All files use show_id as the **foreign key**, allowing them to be rejoined during analysis or dashboarding in Power BI. This structure supports:

- Efficient joins
- Cleaner visual modeling

This Excel restructuring step ensured the dataset was ready for importing into Power BI, where relationships could be explicitly defined and maintained in the model view.

Power BI Dashboards

After cleaning and structuring the dataset, I imported the finalized data from MySQL into Power BI. I then created interactive dashboards to highlight key insights across genre, time, rating, and global distribution.

Dashboard 1: Overview of Netflix Content

This dashboard features **4 visualizations** that give an insightful view of the Netflix catalog:

1. Top 10 Genres (Bar Chart)

- Visualizes the most common content categories (listed_in)
- Ranked by total number of shows/movies in each genre
- Useful for identifying dominant content themes on Netflix

2. Productions Over Time (Line Chart)

- Displays content additions by year from 2014 to 2020
- Two separate lines: one for **Movies**, one for **TV Shows**
- Helps visualize content growth trends and compare formats over time

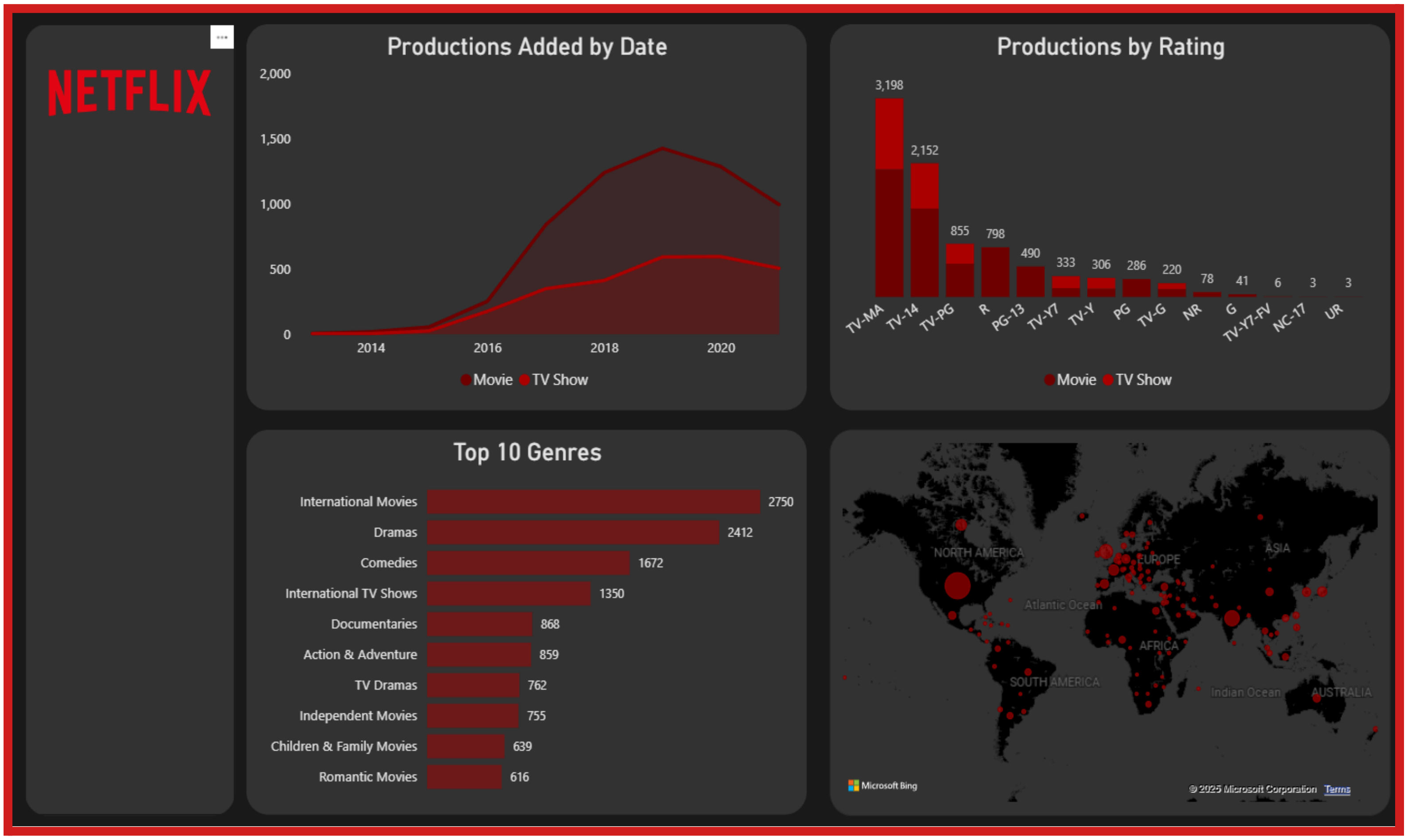
3. Content Ratings Breakdown (Stacked Bar Chart)

- Shows the number of productions for each **rating category** (e.g., TV MA, PG, R)
- Stacked by **type** (Movie vs TV Show)
- Useful for understanding the maturity level of the platform’s catalog

4. Global Production Map (Filled Circle Map)

- Highlights all countries where Netflix content was produced

- Each circle's size is proportional to the number of shows/movies produced in that country
- Offers a clear picture of Netflix's global production footprint



Dashboard 2: Title Detail Viewer

This Power BI report emulates a **Netflix style detail page**, providing a comprehensive overview of any selected movie or TV show. It is designed to be modular and dynamically updates based on the selected title.

Information Displayed

Dynamic Content Selector

- A dropdown enables users to choose a specific **movie or TV show** from the catalog.
- All visual and textual fields update automatically based on the selection.

Title Information

- **Name:** Title of the movie or show
- **Release Year:** Year the production was first released
- **Rating:** Official content rating (e.g., PG, R, TV-MA)

Descriptive Fields

- **Synopsis:** A short plot summary pulled from the dataset
- **Genres:** Content classification, such as “Dramas & Family Movies”

Production Credits

- **Director(s):** Displays associated director(s)
- **Cast:** Main cast members featured in the title

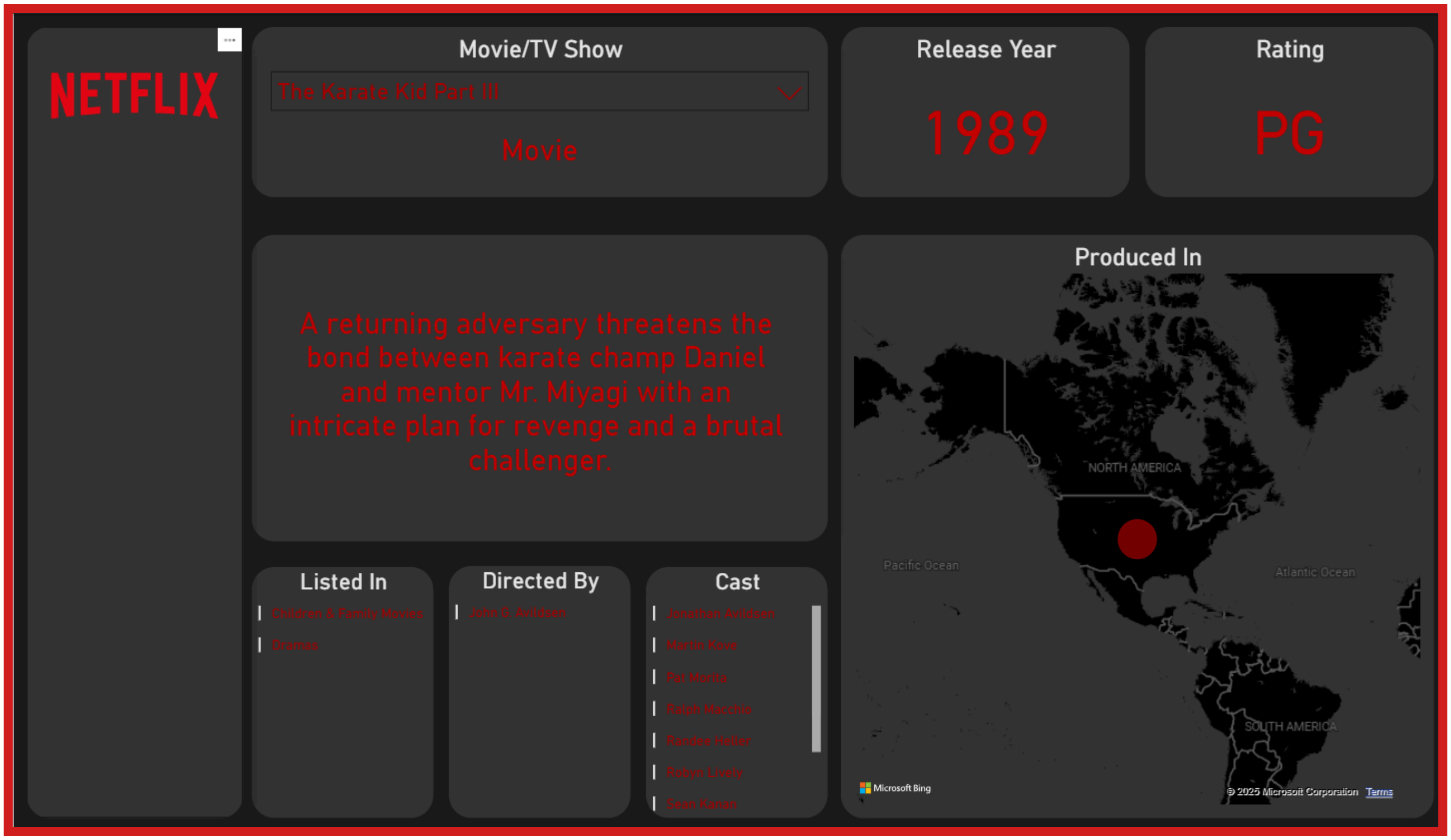
Geographic Visualization

- An interactive **map visualization** showing the main country associated with the production
- Useful for understanding where the content was produced or primarily set

Purpose & Use Case

This dashboard offers an **at a glance media profile** interface, ideal for building user facing catalog experiences or internal content review tools. Its modular design allows for:

- Scalable media libraries
- Quick switching between titles
- Contextual information aggregation (cast, rating, synopsis, location)



NETFLIX

Movie/TV Show

The Karate Kid Part III

Movie

Release Year

1989

Rating

PG

A returning adversary threatens the bond between karate champ Daniel and mentor Mr. Miyagi with an intricate plan for revenge and a brutal challenger.

Listed In

- Children & Family Movies
- Dramas

Directed By

- John G. Avildsen

Cast

- Jonathan Avildsen
- Martin Kove
- Pat Morita
- Ralph Macchio
- Randee Heller
- Robyn Lively
- Sean Kanan

Produced In