DATA CLEANING

I created a staging table so as to avoid tampering with the original dataset. This is a good practice because it keeps the original dataset intact while duplicating a new table to be used for analysis.

```
21
22    ## Create a staging table to use for analysis  ##
23
24 •  CREATE TABLE netflix_staging
25    LIKE netflix_titles;
26
27 •  INSERT netflix_staging
28    SELECT *
29    FROM netflix_titles;
30
31 •  SELECT *
32    FROM netflix_staging;
33
```

The second step was checking for duplicates. I used the **ROW_NUMBER** window function to get unique numbers for each row. Then I used a CTE to check if there was any row which appeared more than once, which would have meant that there was a duplicate. Upon checking, I found out that there were zero duplicates.

```
35
36    -- 1. Checking for duplicates and removing the duplicates if they are there
37
38    SELECT *,
39    ROW_NUMBER() OVER(PARTITION BY show_id, `type`, title, director, country, date_added, release_year, rating, duration) AS row_num
40    FROM netflix_staging;
41
42    WITH duplicates_cte AS
43    (
44    SELECT *,
45    ROW_NUMBER() OVER(PARTITION BY
46    show_id, `type`, title, director, country, date_added, release_year, rating) AS row_num
47    FROM netflix_staging
48    )
49    SELECT *
50    FROM duplicates_cte
51    WHERE row_num > 1;
52
53    ## after checking, it is confirmed that this dataset does not contain any duplicates
54
```

The third step involved standardizing the data. The only thing that needed standardizing was the "date_added" column. There were two formats being used there and I needed to use one format, which was not part of the two which were used. Therefore, using **CASE** statements, I changed the formats then changed the data type to **DATETIME.**

```
-- 2. Standardize the data

SELECT *
FROM netflix_staging;

SELECT date_added,
    CASE
        WHEN date_added LIKE '%-%-%' THEN
                STR_TO_DATE(date_added, '%d-%b-%y')
        ELSE    STR_TO_DATE(date_added, '%M%d,%Y')
    END AS converted_date
FROM netflix_staging;



UPDATE netflix_staging
SET date_added = CASE
                    WHEN date_added LIKE '%-%-%' THEN
                            STR_TO_DATE(date_added, '%d-%b-%y')
                    ELSE    STR_TO_DATE(date_added, '%M%d,%Y')
                    END;


ALTER TABLE netflix_staging
MODIFY COLUMN date_added DATE;

## changed the formatting of the date_added which was in 2 different formats and changed them to the standard format
## updated the changes on the table and altered the table from text to datetime data type
```

Then I handled all the **NULL** values, which was basically deleting them because there was no information that could have otherwise been used to populate them.

```
-- 3. Null or Blank values

SELECT *
FROM netflix_staging;

SELECT *
FROM netflix_staging
WHERE director IS NULL;

DELETE
FROM netflix_staging
WHERE director IS NULL;

## deleted all the nulls that existed in director and country because there is no way of populating them
```

This was followed by **DELETING** any unnecessary columns in the table.

```
105
106     -- 4. Remove any unnecessary data
107
108 •   SELECT *
109     FROM netflix_staging;
110
111 •   ALTER TABLE netflix_staging
112     DROP COLUMN cast;
113
114 •   ALTER TABLE netflix_staging
115     DROP COLUMN duration;
116
117 •   ALTER TABLE netflix_staging
118     DROP COLUMN listed_in;
119
120 •   ALTER TABLE netflix_staging
121     DROP COLUMN `description`;
122
123     ## removed all the unnecessary columns that will not be used
124
```

DATA EXPLORATION

The exploratory data analysis involved checking for the number of movies and television shows added on **Netflix** based on director, country, year, and rating. This was followed by a little time series to determine the monthly rolling total of movies and tv shows added on **Netflix**.

```
8  •  SELECT MIN(date_added), MAX(date_added)
9     FROM netflix_staging;
10
11
12 •  SELECT director, COUNT(title) AS movies_directed
13    FROM netflix_staging
14    GROUP BY director
15    ORDER BY 2 DESC;
16
17 •  SELECT country, COUNT(title) AS movies_per_country
18    FROM netflix_staging
19    GROUP BY country
20    ORDER BY 2 DESC;
21
22 •  SELECT YEAR(date_added), COUNT(title) AS movies_per_year
23    FROM netflix_staging
24    GROUP BY YEAR(date_added)
25    ORDER BY 2 DESC;
26
27 •  SELECT rating, COUNT(title) AS movies_per_rating
28    FROM netflix_staging
29    GROUP BY rating
30    ORDER BY 2 DESC;
```

```sql

SELECT *
FROM netflix_staging;


WITH rolling_cte AS
(
SELECT SUBSTRING(date_added, 1, 7) AS `MONTH`, COUNT(title) AS monthly_addition
FROM netflix_staging
GROUP BY `MONTH`
ORDER BY 1 ASC
)
SELECT `MONTH`, monthly_addition,
SUM(monthly_addition) OVER (ORDER BY `MONTH`) AS rolling_total
FROM rolling_cte;
```

VISUALISATION

I did the visualization of the analysis using **Microsoft Power BI,** where I used a slicer to create an interactive visualization which shows the top 10 countries and directors in terms of either a movie or a show on **Netflix.**