Using NETFLIX data taken from Kaggle, I am attempting to derive insights from the shows added on Netflix between the period 2008 and 2022.

Initially, imported the data into SQL server using flat file import.

Check the general outline of the file containing 12 columns x 8,807 rows.

```
SELECT TOP 5 *
FROM [dbo].[netflix_raw]
```

Checking for duplicates in the show_id which looks like the primary key for the table.

```
SELECT show_id
FROM [dbo].[netflix_raw]
UNION
SELECT show_id
FROM [dbo].[netflix_raw]; -- 8807 records as output.
```

Since the number of records in the Union's table is same as the number of records in original file, it means that all are unique show_ids.

Now, to check status of nulls in the table.

```
SELECT
        sum(CASE WHEN show_id IS NULL THEN 1 ELSE 0 END) AS show_id_nulls,
        sum(CASE WHEN type IS NULL THEN 1 ELSE 0 END) AS type_nulls,
        sum(CASE WHEN title IS NULL THEN 1 ELSE 0 END) AS title_nulls,
        sum(CASE WHEN director IS NULL THEN 1 ELSE 0 END) AS director_nulls, --2634 NULL
        sum(CASE WHEN cast IS NULL THEN 1 ELSE 0 END) AS cast_nulls, --825 NULL
        sum(CASE WHEN country IS NULL THEN 1 ELSE 0 END) AS country_nulls, -- 831 NULL
        sum(CASE WHEN date_added IS NULL THEN 1 ELSE 0 END) AS date_added_nulls, -- 98 NULL
        sum(CASE WHEN release_year IS NULL THEN 1 ELSE 0 END) AS release_year_nulls,
        sum(CASE WHEN rating IS NULL THEN 1 ELSE 0 END) AS rating_nulls, --4 NULL
        sum(CASE WHEN duration IS NULL THEN 1 ELSE 0 END) AS duration_nulls, --3 NULL
        sum(CASE WHEN listed_in IS NULL THEN 1 ELSE 0 END) AS listed_in_nulls,
        sum(CASE WHEN description IS NULL THEN 1 ELSE 0 END) AS description_nulls
FROM [dbo].[netflix_raw];
```

Nulls are present in 6 columns which needs to be checked.

Before proceeding forward, store data into a temporary table. This will allow us to retain the netflix_raw table at a later point in case any changes need to be made. And, we can ultimately change the temporary table into final Cleaned_netflix file ready for analysis.

```
SELECT * INTO #netflix
FROM [dbo].[netflix_raw]-- created a temporary table with 8,807 rows
```

Checking for director_nulls which has ~2600 NULLs.
Currently checking to see if there is a relation between the director and cast.

```
        WITH directornull  AS (
        SELECT concat(director,'---',cast) AS director_cast
```

```
        FROM #netflix)
        SELECT director_cast, count(*) AS count
        FROM directornull
        GROUP BY director_cast
        HAVING count(*) > 1
        ORDER BY count(*) DESC;-- ~30 values may be connected.
```

Assumptions for filling certain rows with data. Major contributors in this list are just 3.

```
        SELECT *
        FROM #netflix
        WHERE cast = 'David Attenborough';-- Director is common for most of David
Attenborough's movie/show.

        UPDATE #netflix
        SET director = 'Alistair Fothergill'
        WHERE cast = 'David Attenborough' AND director IS NULL;-- 15 rows affected.

        UPDATE #netflix
        SET director = 'Rajiv Chilaka'
        WHERE cast LIKE '%Julie Tejwani%' AND director IS NULL;-- 5 rows affected

        SELECT *
        FROM #netflix
        WHERE cast LIKE 'Michela Luci%'

        UPDATE #netflix
        SET director ='Mark Thronton, Todd Kauffman'
        WHERE cast LIKE 'Michela Luci%'-- 14 rows affected
```

For the remaining we mark the director as unknown.

```
        UPDATE #netflix
        SET director = 'Unknown'
        WHERE director IS NULL;--2602 rows affected
```

To populate the missing countries using the director field.

Ennumerating number of rows where this is possible.

```
        SELECT *
        FROM #netflix
        WHERE country IS NULL AND director IS NOT NULL-- 831 rows.
```

Using a SELF JOIN in order to fill up missing data where possible.

```
        UPDATE #netflix
        SET #netflix.country = A.country
                FROM #netflix INNER JOIN #netflix A ON #netflix.director = A.director
                WHERE #netflix.country IS NULL
                        AND A.country IS NOT NULL -- 557 rows affected.
```

Now checking for country Nulls column and update as Not available.

```
        SELECT *
        FROM #netflix
        WHERE country IS NULL -- 274 rows are NULL.

        UPDATE #netflix
        SET country = 'Not available'
        WHERE country IS NULL;
```

Next, checking for listed_in (genre) column.

```
        SELECT listed_in, count(listed_in)
        FROM #netflix
        WHERE cast is NULL
        GROUP BY listed_in
        ORDER BY count(listed_in) DESC; --Seen that cast is empty usually for Docuseries or
Documentaries or Reality TV
```

Major categories are found to be with tag '%Docu%', '%Reality%, '%Kids%'' which will be set as 'Not applicable'. Remaining shall be set as 'Not available'

```
        UPDATE #netflix
        SET cast = 'Not applicable'
        WHERE cast IS NULL AND listed_in LIKE 'Docu%'-- 522 rows affected.

        UPDATE #netflix
        SET cast = 'Not applicable'
        WHERE cast IS NULL
                AND (listed_in LIKE '%Docu%'
                        OR listed_in LIKE 'Real%'
                        OR listed_in LIKE '%Real%') -- 183 rows affected.
                -- Remaining cast nulls are 120nos. Out of 120nos, there are some titles

        SELECT listed_in,count(*)
        FROM #netflix
        WHERE cast IS NULL
        GROUP BY listed_in
        ORDER BY count(*) DESC
        -- Next big category is Kids' TV (23 + 16 + 10 nos.) which can also be replace with
Not applicable.

        SELECT count(listed_in)
        FROM #netflix
        WHERE cast IS NULL and listed_in LIKE '%Kids%';-- 42 nos.

        UPDATE #netflix
        SET cast = 'Not applicable'
        WHERE cast IS NULL AND listed_in LIKE '%Kids%'-- 42 rows affected

        SELECT *
        FROM #netflix
        WHERE cast IS NULL -- 78 rows still have NULLs in cast.

        UPDATE #netflix
        SET cast = 'Not applicable'
        WHERE cast IS NULL AND listed_in LIKE '%Children%' -- Accounting for children shows
(26 rows)

        SELECT count(*)
```

```
FROM #netflix
WHERE cast IS NULL -- 52 are NULLS which are replaced to N.A. in next code.

UPDATE #netflix
SET cast = 'Not available'
WHERE cast IS NULL;-- 52 rows affected.
```

Now, we check for date_added NULLs.

```
SELECT *
FROM #netflix
WHERE date_added IS NULL;
```

98 nos. are NULLs. Since there is no identifiable way to fill it up, we shall leave it as is. About 1% of data and shouldn't affect the whole analysis by much.

Next check for rating_nulls.

```
SELECT *
FROM #netflix
WHERE rating IS NULL;

UPDATE #netflix
SET rating = 'Not available'
WHERE rating IS NULL;-- 4 rows updated.
```

4 rows with no rating available which are upadted to N.A.

Checking for duration NULLs.

```
SELECT *
FROM #netflix
WHERE duration IS NULL;
-- Here 3 rows data is interchanged between duration and rating.
UPDATE #netflix
SET duration = rating
WHERE show_id IN ('s5542','s5795','s5814');

UPDATE #netflix
SET rating = 'Not available'
WHERE show_id IN ('s5542','s5795','s5814');
```

After completing all columns we recheck as below.

```
SELECT
        sum(CASE WHEN show_id IS NULL THEN 1 ELSE 0 END) AS show_id_nulls, -- 0
        sum(CASE WHEN type IS NULL THEN 1 ELSE 0 END) AS type_nulls, -- 0
        sum(CASE WHEN title IS NULL THEN 1 ELSE 0 END) AS title_nulls, -- 0
        sum(CASE WHEN director IS NULL THEN 1 ELSE 0 END) AS director_nulls, --0
        sum(CASE WHEN cast IS NULL THEN 1 ELSE 0 END) AS cast_nulls, -- 0
        sum(CASE WHEN country IS NULL THEN 1 ELSE 0 END) AS country_nulls, -- 0
        sum(CASE WHEN date_added IS NULL THEN 1 ELSE 0 END) AS date_added_nulls, -- 98
        sum(CASE WHEN release_year IS NULL THEN 1 ELSE 0 END) AS release_year_nulls, --0
        sum(CASE WHEN rating IS NULL THEN 1 ELSE 0 END) AS rating_nulls, --0
        sum(CASE WHEN duration IS NULL THEN 1 ELSE 0 END) AS duration_nulls, --0
```

```
        sum(CASE WHEN listed_in IS NULL THEN 1 ELSE 0 END) AS listed_in_nulls, -- 0
        sum(CASE WHEN description IS NULL THEN 1 ELSE 0 END) AS description_nulls --0
FROM #netflix;
```

Satisfied with the result, we go ahead with checking country column since it has delimiter separated multiple countries.

```
SELECT DISTINCT country,len(country)
FROM #netflix
ORDER BY len(country) DESC -- 674 entries out of 749 entries have more than one country in
their statement.
```

I assume the left most country is the actual country of origin for the particular show.

```
SELECT
        CASE WHEN country LIKE '%, %' THEN LEFT(substring(country,1,CHARINDEX(',
',country)),len(substring(country,1,CHARINDEX(', ',country)))-1)
        ELSE country END
FROM #netflix;
```

Add a new column country_m in order to add new countries.

```
ALTER TABLE #netflix
ADD Country_m varchar(100);

UPDATE #netflix
SET Country_m =
        CASE WHEN country LIKE '%, %' THEN LEFT(substring(country,1,CHARINDEX(',
',country)),len(substring(country,1,CHARINDEX(', ',country)))-1)
        ELSE country END-- 8807 records affected.
```

Dropping original country column to avoid any confusion

```
ALTER TABLE #netflix
DROP COLUMN country;
```

Now, save the temporary table into a cleaned netflix table.

```
SELECT * INTO [dbo].[Cleaned_netflix]
FROM #netflix;

DROP TABLE IF EXISTS #netflix
```

Checking column listed_in for peculiarities, it is seen that there are more than one genre attached to each show based on the listed_in column id.

```
-- Checking listed_in column for peculiarities
SELECT DISTINCT listed_in,count(listed_in)
FROM [dbo].[Cleaned_netflix]
GROUP BY listed_in
ORDER BY count(listed_in) DESC;
```

```
-- Listed_in has maximum of 3 categories- checked by below code.
SELECT
        listed_in
        , replace(listed_in,', ','') AS replaced_string
        , (len(listed_in) - len(REPLACE(listed_in,', ','')))/2 + 1 AS number_of_genre
FROM Cleaned_netflix
ORDER BY number_of_genre DESC;
```

To isolate the individual genres and their frequency, we create a separate table. The first step would be to separate the listed_in table into 3 separate columns- genre1, genre2, genre3.

```
ALTER TABLE [dbo].[Cleaned_netflix]
ADD
        genre1 varchar(100)
        , genre2 varchar(100)
        , genre3 varchar(100)

UPDATE [dbo].[Cleaned_netflix]
SET genre1 = TRIM(REVERSE(PARSENAME(REPLACE(REVERSE(listed_in), ',', '.'), 1)));

UPDATE [dbo].[Cleaned_netflix]
SET genre2 = TRIM(REVERSE(PARSENAME(REPLACE(REVERSE(listed_in),',','.'), 2)));

UPDATE [dbo].[Cleaned_netflix]
SET genre3 = TRIM(REVERSE(PARSENAME(REPLACE(REVERSE(listed_in),',','.'), 3)));
```

Now we construct a table using the three separated columns and adding it to a new column genre, which will essentially convert wide data into long. This data shall be stored in a separate genre table.

```
-- Making a unioned table for genre alone. This table named [dbo].[genre]  will have each
show with three rows. (expecting 8807 x 3 = 26,421 rows)

ALTER TABLE [dbo].[Cleaned_netflix]
ADD genre varchar(100)

SELECT * INTO [dbo].[genre]-- note that any aggregation in this table needs to be
mean/average.
FROM
        (SELECT show_id , type , title , rating , Country_m , genre = genre1
        FROM [dbo].[Cleaned_netflix]
        UNION ALL
        SELECT show_id , type , title , rating , Country_m , genre = genre2
        FROM [dbo].[Cleaned_netflix]
        UNION ALL
        SELECT show_id , type , title , rating , Country_m , genre = genre3
        FROM [dbo].[Cleaned_netflix])gen;-- 26,421 rows.
```

Leaving the above table separately for analysis.

Checking the range of years that is covered in the data.

```
-- Add columns year, month, date, day of week for date_added
SELECT max(year(date_added)), min(year(date_added))
FROM [dbo].[Cleaned_netflix]-- year taken out ranging between 2008 and 2021
```

Also see the difference between release year and date_added year.

```
SELECT
        max(year(date_added)-release_year) AS [Max Year gap]
        , min(year(date_added)-release_year) AS [Min Year gap]
        , avg(Year(date_added)-release_year) AS [Avg Year gap]
FROM [dbo].[Cleaned_netflix]
-- Maximum year gap of release date and date added to Netflix is 93 but the average year gap
is just 4 years.
```

Oddly there is a min year gap which is negative, which suggests that some shows were added to netflix before year of release which will be investigated in following steps.

Now adding columns to check year, month, weekday, releasetoadd into the Cleaned_netflix table to get mroe insights.

```
ALTER TABLE [dbo].[Cleaned_netflix]
ADD date_added_y INT, date_added_m varchar(100), date_added_w varchar(100), releasetoadd INT

UPDATE [dbo].[Cleaned_netflix]
SET date_added_y = year(date_added);

UPDATE [dbo].[Cleaned_netflix]
SET date_added_m = datename(month,date_added);

UPDATE [dbo].[Cleaned_netflix]
SET date_added_w = datename(w,date_added);

UPDATE [dbo].[Cleaned_netflix]
SET releasetoadd = year(date_added)-release_year;
```

Updated the above values into the new columns.

```
SELECT *
FROM [dbo].[Cleaned_netflix]
WHERE year(date_added)-release_year <0;
```

There are 14 rows where the release year is greater than date_added which seems to be odd. Currently ignoring this to see if there is any change required during visualization.

We now proceed with analysis phase.

Most frequent genre overall

```
SELECT genre, count(genre) AS [Show count]-- Total of 42 genre
FROM [dbo].[genre]
GROUP BY genre
ORDER BY count(genre) DESC;
```

Most frequent genre arranged by the year

```
SELECT
        A.date_added_y
        , G.genre
```

```
        , count(G.genre) AS [Shows with Genre count]
FROM [dbo].[genre] G INNER JOIN [dbo].[Cleaned_netflix] A
        ON A.show_id = G.show_id
WHERE date_added_y IS NOT NULL
GROUP BY A.date_added_y, G.genre
HAVING count(G.genre)>0
ORDER BY A.date_added_y ASC, [Shows with Genre count] DESC;
```

Top 10 genre based on show type

```
WITH genre_ranked_type AS
( SELECT
        type
        , genre
        , count(genre)/3 AS [Shows with genre Count]
        , DENSE_RANK() OVER (PARTITION BY type ORDER BY count(genre) DESC) AS genre_rank
FROM [dbo].[genre]
GROUP BY type,genre)
SELECT *
FROM genre_ranked_type
```

Most frequent genre based on rating

```
SELECT
        rating
        , genre
        , count(genre)/3 AS [Genre Count ]
FROM [dbo].[genre]
GROUP BY rating, genre
ORDER BY rating, [Genre Count] DESC;
```

Most frequent genre combination overall

```
SELECT
        date_added_y
        , listed_in
        , count(listed_in) AS [Show count]
FROM [dbo].[Cleaned_netflix]
GROUP BY date_added_y, listed_in
HAVING date_added_y IS NOT NULL
ORDER BY date_added_y ASC, count(listed_in) DESC;
```

Number of shows added over the years.

```
SELECT date_added_y,type, count(*) AS [Shows added]
FROM [dbo].[Cleaned_netflix]
WHERE date_added_y IS NOT NULL
GROUP BY date_added_y,type
ORDER BY date_added_y;
```

Which day of week is popular for release of movie / tv show?

```
SELECT date_added_w,type, count(*) AS [Shows added]
FROM [dbo].[Cleaned_netflix]
```

```
WHERE date_added_w IS NOT NULL
GROUP BY date_added_w,type
ORDER BY count(*) DESC ;
```

Checking duration specifics for TV shows

```
SELECT
        duration
        , count(duration) AS [Number of shows]
FROM [dbo].[Cleaned_netflix]
WHERE duration LIKE '%Season%'
GROUP BY duration
ORDER BY [Number of shows] DESC
```

Checking duration specifics for movies

```
SELECT
        cast(replace(duration,' min','') AS int) [Movie duration]
        , count(duration) AS [Number of shows]
FROM [dbo].[Cleaned_netflix]
WHERE duration NOT LIKE '%Season%'
GROUP BY cast(replace(duration,' min','') AS int)
ORDER BY [Movie duration] DESC;
```

Check for fall and rise of ratings

```
SELECT date_added_y, rating, count(rating)
FROM [dbo].[Cleaned_netflix]
GROUP BY date_added_y, rating
```

Country prominence

One particular value was found in country_m column where (blank space) was added as a category. This will be converted to not available.

```
SELECT country_m
FROM [dbo].[Cleaned_netflix]
WHERE Country_m = ' ';

UPDATE [dbo].Cleaned_netflix
SET Country_m = 'Not available'
WHERE Country_m = ' ';
```

Check which country has most shows linked.

```
SELECT country_m, count(*) AS [Number of shows]
FROM [dbo].[Cleaned_netflix]
GROUP BY Country_m
ORDER BY [Number of shows] DESC
```

Countries with rise and fall of rating by shows?

```
SELECT country_m, date_added_y, rating, count(*) AS [Number of shows]
FROM [dbo].[Cleaned_netflix]
GROUP BY Country_m, date_added_y, rating
ORDER BY Country_m;
```

-- FINISH --