Business Problem

Analyze the data and generate insights that could help Netflix in deciding which type of shows/movies to produce and how they can grow the business in different countries.

Hints

1. The exploration should have a goal. As you explore the data, keep in mind that you want to answer which type of shows to produce and how to grow the business.
2. Ensure each recommendation is backed by data. The company is looking for data-driven insights, not personal opinions or anecdotes.
3. Assume that you are presenting your findings to business executives who have only a basic understanding of data science. Avoid unnecessary technical jargon.
4. Start by exploring a few questions: What type of content is available in different countries?

    a. How has the number of movies released per year changed over the last 20-30 years?

    b. Comparison of tv shows vs. movies.

    c. What is the best time to launch a TV show?

    d. Analysis of actors/directors of different types of shows/movies.

    e. Does Netflix has more focus on TV Shows than movies in recent years.

    f. Understanding what content is available in different countries.

Column Description

1. show_id : Unique ID for every Movie / TV Show
2. type : Identifier - A Movie or TV Show
3. title : Title of the Movie / Tv Show
4. director : Director of the Movie
5. cast : Actors involved in the movie/show
6. country : Country where the movie/show was produced
7. date_added : Date it was added on Netflix
8. release_year : Actual Release year of the movie/show
9. rating : TV Rating of the movie/show
10. duration : Total Duration - in minutes or number of seasons

11. listed_in : Genre
12. description : The summary description

Evaluation Criteria (100 Points):

1. Defining Problem Statement and Analysing basic metrics. (10 Points)
2. Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary. (10 Points)
3. Non-Graphical Analysis: Value counts and unique attributes. (10 Points)
4. Visual Analysis - Univariate, Bivariate after pre-processing of the data. Note: Pre-processing involves unnesting of the data in columns like Actor, Director, Country.
    a. For continuous variable(s): Distplot, countplot, histogram for univariate analysis. (10 Points)
    b. For categorical variable(s): Boxplot. (10 Points)
    c. For correlation: Heatmaps, Pairplots. (10 Points)
5. Missing Value & Outlier check (Treatment optional). (10 Points)
6. Insights based on Non-Graphical and Visual Analysis. (10 Points)
    a. Comments on the range of attributes.
    b. Comments on the distribution of the variables and relationship between them.
    c. Comments for each univariate and bivariate plot.
7. Business Insights (10 Points) - Should include patterns observed in the data along with what you can infer from it.
8. Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

Submission Process:

1. Type your insights and recommendations in the rich-text editor.
2. Convert your jupyter notebook into PDF (Save as PDF using Chrome browser’s Print command), upload it in your Google Drive (set the permission to allow public access), and paste that link in the text editor.
3. Alternatively, you can directly submit your PDF on the portal.
4. Optionally, you may add images/graphs in the text editor by taking screenshots or saving matplotlib graphs using plt.savefig(...).
5. After submitting, you will not be allowed to edit your submission.

Let's Start!

In [1]:
```
# importing the required libraries
import re, numpy as np, pandas as pd, plotly.express as px, plotly.grapl

# reading the data from the `data.csv` saved in the same folder as this
df = pd.read_csv('data.csv')
```

The First Step towards solving any Business Problem through Data is Exploratory Data Analysis (EDA) and the First Step towards EDA is basis analysis of the data (number of records, number of features and their corresponding data types), locating and eliminating Missing Values, and transforming features into something which is explorable and meaningfull.

We will now work on getting answers of all the points we just mentioned and these answers will help us give answer to Q2 and Q5 of the Evaluation Criteria.

Q2: Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary. (10 Points)
Q5: Missing Value & Outlier check (Treatment optional). (10 Points)

In [2]:
```
# printing sample data
df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021

In [3]:

```
# printing the shape of the data
df.shape
```

Out[3]: (8807, 12)

In [4]:

```
# checking columns' datatype
df.dtypes
```

Out[4]:
```
show_id          object
type             object
title            object
director         object
cast             object
country          object
date_added       object
release_year      int64
rating           object
duration         object
listed_in        object
description      object
dtype: object
```

Data Cleaning and Preprocessing

In [5]:
```
# dropping `show_id` column because it won't help us in our analysis as
df.drop('show_id', axis=1, inplace=True)

# converting `release_year` to `object` because we will be using it as 
df['release_year'] = df['release_year'].astype(object)

# converting `date_added` to `datetime` because we will be performing ye
df['date_added'] = pd.to_datetime(df['date_added'], infer_datetime_forma
```

In [6]:
```
# since now we have all our columns as categorical, `df.describe` would
df.describe(datetime_is_numeric=False)
```

Out[6]:

	type	title	director	cast	country	date_added	release_year	rati
count	8807	8807	6173	7982	7976	8797	8807	88
unique	2	8807	4528	7692	748	1714	74	
top	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	2020-01-01	2018	T M
freq	6131	1	19	19	2818	110	1147	32

In [7]:
```
# checking for missing values
df.isna().sum()
```

Out[7]:
```
type                0
title               0
director         2634
cast              825
country           831
date_added         10
release_year        0
rating              4
duration            3
listed_in           0
description         0
dtype: int64
```

In [8]:
```
# working on `duration` column and checking NaNs values
df[df['duration'].isna()]
```

Out[8]:

	type	title	director	cast	country	date_added	release_year	rating	dura
5541	Movie	Louis C.K. 2017	Louis C.K.	Louis C.K.	United States	2017-04-04	2017	74 min	I
5794	Movie	Louis C.K.: Hilarious	Louis C.K.	Louis C.K.	United States	2016-09-16	2010	84 min	I
5813	Movie	Louis C.K.: Live at the Comedy Store	Louis C.K.	Louis C.K.	United States	2016-08-15	2015	66 min	I

In [9]:
```
# we saw that `duration` is filled in `rating` column. hence correcting
for idx in df[df['duration'].isna()].index:
    df.loc[idx, 'duration'] = df.loc[idx, 'rating']
    df.loc[idx, 'rating'] = np.nan

# again checking for NaNs values and this time it should be an empty da
assert df[df['duration'].isna()].shape[0] == 0
```

Transforming duration Column

In [10]:

```
# we will first check if all the corresponding to `type=Movie` and `typ
# from `df['type'].unique()`) ends in `min` and `Season` or `Seasons` r

# filtering `Movie` and checking some conditions to successfully remove
ser = df[df['type'] == 'Movie']['duration']
ser = ser[ser.notna()].str.split()
assert set(ser.apply(lambda x: len(x)).to_list()) == {2,}
assert set(ser.apply(lambda x: x[1]).to_list()) == {'min',}

# filtering `TV Show` and checking some conditions to successfully remo
ser = df[df['type'] == 'TV Show']['duration']
ser = ser[ser.notna()].str.split()
assert set(ser.apply(lambda x: len(x)).to_list()) == {2,}
assert set(ser.apply(lambda x: x[1]).to_list()) == {'Season', 'Seasons'

# stripping off 'min', 'Season', and 'Seasons' from `duration` column
df['duration'] = df['duration'].str.split().apply(lambda x: x[0]).astype

df.head()
```

Out[10]:

	type	title	director	cast	country	date_added	release_year	rating	du
0	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	2021-09-25	2020	PG-13	
1	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA	
2	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	2021-09-24	2021	TV-MA	
3	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	2021-09-24	2021	TV-MA	
4	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2021	TV-MA	

Transforming dataframe basis the description Column

In [11]:

```
# in `df.describe` we saw that in the column `description` top entry oc
print(df[df['description'].duplicated()].shape)
df[df['description'].duplicated()].head()
```

(32, 11)

Out[11]:

	type	title	director	cast	country	date_added	release_year
79	Movie	Tughlaq Durbar (Telugu)	Delhiprasad Deenadayalan	Vijay Sethupathi, Parthiban, Raashi Khanna	NaN	2021-09-11	2021
237	Movie	Boomika (Hindi)	Rathindran R Prasad	Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madh...	NaN	2021-08-23	2021
238	Movie	Boomika (Malayalam)	Rathindran R Prasad	Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madh...	NaN	2021-08-23	2021
239	Movie	Boomika (Telugu)	Rathindran R Prasad	Aishwarya Rajesh, Vidhu, Surya Ganapathy, Madh...	NaN	2021-08-23	2021
851	Movie	99 Songs (Tamil)	NaN	NaN	NaN	2021-05-21	2021

In [12]:

```
# on perusing the above dataframe, one can observe that these are rows 
# also, for the same movie (based on description) there are NaNs presen

all_duplicated_description = df[df['description'].duplicated()]['descri
all_duplicated_description = {desc: all_duplicated_description.count(de

replace_desc_dict, custom_replace_desc_dict = dict(), dict()

for idx, desc in enumerate(all_duplicated_description.keys()):
    temp_df = df[df['description'] == desc]

    # checking if for same `description` we have same `type`, `director
    # if all these are same or missing, we are treating different title
    # otherwise treating them as different and updating them in `custom
    if (temp_df['type'].nunique() < 2) and (temp_df['director'].nunique
        replace_desc_dict[desc] = all_duplicated_description[desc]
    else:
        custom_replace_desc_dict[desc] = all_duplicated_description[des
len(replace_desc_dict), len(custom_replace_desc_dict)
```

Out[12]: (25, 2)

In [13]:

```
# forming a common strategy to fill NaNs in rows corresponding to 25 de
for desc in list(replace_desc_dict.keys()):
    temp_df = df[df['description'] == desc]
    for column_name in df.columns:
        col_filler = df.loc[temp_df.index][column_name]
        col_filler_value = col_filler[col_filler.notna()].mode()
        if col_filler_value.to_list() != []:
            df.loc[temp_df.index, column_name] = col_filler_value[0]
```

In [14]:

```
# analyzing rows corresponding to first `description` in `custom_replac
desc = list(custom_replace_desc_dict.keys())[0]

temp_df = df[df['description'] == desc]
# we saw that there are a total of 3 rows corresponding to this `descri
# hence filtering only for `type=Movie` and applying the same strategy 

temp_df = temp_df[temp_df['type'] == 'Movie']
for column_name in df.columns:
    col_filler = df.loc[temp_df.index][column_name]
    col_filler_value = col_filler[col_filler.notna()].mode()
    if col_filler_value.to_list() != []:
        df.loc[temp_df.index, column_name] = col_filler_value[0]
```

In [15]:

```
# analyzing rows corresponding to second `description` in `custom_repla
desc = list(custom_replace_desc_dict.keys())[1]
temp_df = df[df['description'] == desc]
temp_df

# nothing we can do here because there are a total of 2 rows correspond
# hence leaving as it is.
```

Out[15]:

	type	title	director	cast	country	date_added	release_year	ratin
2336	Movie	Thackeray (Hindi)	Abhijit Panse	Nawazuddin Siddiqui, Amrita Rao, Rajeev Panday...	India	2020-06-26	2019	TV-1
8173	TV Show	Thackeray	NaN	NaN	India	2019-05-25	2019	TV M

In [16]:

```
# checking shape of dataframe before and after dropping duplicates
print(df.shape, ' - Shape of DataFrame with duplicates.')
df.drop_duplicates(inplace=True)
print(df.shape, ' - Shape of DataFrame without duplicates.')
```

```
(8807, 11)  - Shape of DataFrame with duplicates.
(8777, 11)  - Shape of DataFrame without duplicates.
```

In [17]:

```
# checking `df.describe()` for further steps
df.describe(include=object)
```

Out[17]:

	type	title	director	cast	country	date_added	release_year	rati
count	8777	8777	6150	7956	7963	8767	8777	87
unique	2	8777	4528	7687	748	1714	74	
top	Movie	Dick Johnson Is Dead	Rajiv Chilaka	David Attenborough	United States	2020-01-01	2018	T N
freq	6105	1	19	19	2816	110	1140	32

Filling NaNs in director, cast, country and rating Columns

In [18]:

```
# let's fill NaNs in `director` and `cast` as 'Anonymous' and in `count
df['director'].fillna('Anonymous', inplace=True)
df['cast'].fillna('Anonymous', inplace=True)
df['country'].fillna('Not Available', inplace=True)
df['rating'].fillna('Not Available', inplace=True)
# now only 2 more columns contain NaNs and these are `date_added` and `
```

Filling NaTs in the date_added Column

In [19]:

```
df[df['date_added'].isna()]
```

Out[19]:

	type	title	director	cast	country	date_added	release_year	ra
6066	TV Show	A Young Doctor's Notebook and Other Stories	Anonymous	Daniel Radcliffe, Jon Hamm, Adam Godley, Chris...	United Kingdom	NaT	2013	
6174	TV Show	Anthony Bourdain: Parts Unknown	Anonymous	Anthony Bourdain	United States	NaT	2018	
6795	TV Show	Frasier	Anonymous	Kelsey Grammer, Jane Leeves, David Hyde Pierce...	United States	NaT	2003	
6806	TV Show	Friends	Anonymous	Jennifer Aniston, Courteney Cox, Lisa Kudrow, ...	United States	NaT	2003	T
6901	TV Show	Gunslinger Girl	Anonymous	Yuuka Nanri, Kanako Mitsuhashi, Eri Sendai, Am...	Japan	NaT	2008	T
7196	TV Show	Kikoriki	Anonymous	Igor Dmitriev	Not Available	NaT	2010	
7254	TV Show	La Familia P. Luche	Anonymous	Eugenio Derbez, Consuelo Duval, Luis Manuel Áv...	United States	NaT	2012	T

7406	TV Show	Maron	Anonymous	Marc Maron, Judd Hirsch, Josh Brener, Nora Zeh...	United States	NaT	2016	
7847	TV Show	Red vs. Blue	Anonymous	Burnie Burns, Jason Saldaña, Gustavo Sorola, G...	United States	NaT	2015	
8182	TV Show	The Adventures of Figaro Pho	Anonymous	Luke Jurevicius, Craig Behenna, Charlotte Haml...	Australia	NaT	2015	T

In [20]:

```
# what we have in above dataframe is that for all the records missing `
# what we can do here is that fill NaNs with release_year-12-31, i.e., 

indices_to_modify = df[df['date_added'].isna()].index

df.loc[indices_to_modify, 'date_added'] = df.loc[indices_to_modify]['re
```

Transforming director, cast, listed_in , and country Columns

In [21]:

```
for column_name in ['director', 'cast', 'listed_in', 'country']:
    df[column_name] = df[column_name].apply(lambda x: x.split(','))

df = df.explode('director').explode('cast').explode('listed_in').explod

for col in df.select_dtypes(object):
    df[col] = df[col].astype(str).str.strip()

df.drop_duplicates(inplace=True)
df.reset_index(drop=True, inplace=True)
df.shape
```

Out[21]: (201316, 11)

In [22]:

```
df.head()
```

Out[22]:

	type	title	director	cast	country	date_added	release_year	rating
0	Movie	Dick Johnson Is Dead	Kirsten Johnson	Anonymous	United States	2021-09-25	2020	PG-13
1	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
2	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
3	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
4	TV Show	Blood & Water	Anonymous	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA

In [23]:

```
df.describe(include=object)
```

Out[23]:

	type	title	director	cast	country	date_added	release_year	
count	201316	201316	201316	201316	201316	201316	201316	2
unique	2	8777	4994	36433	124	1719	74	
top	Movie	Kahlil Gibran's The Prophet	Anonymous	Anonymous	United States	2020-01-01	2018	-
freq	145305	700	50497	2139	59214	3748	24198	

Okay, so now we are ready for some EDA because we have cleansed the data as much as it was possible. The next step would be observing interesting patterns within our data and perform Univariate, Bivariate, and/or Multivariate analysis.

Let's start exploring how are the categories distributed within our object type columns. All the columns makes sense to explore category-balance using value_counts() method except description column because it would be similar to the title column.

Data Analysis

In [24]:

```
df['type'].value_counts()
```

Out[24]:

```
Movie      145305
TV Show     56011
Name: type, dtype: int64
```

Inference A: There are almost 2.5 times Movie titles in our dataset than TV shows. Audience tend to like movies over tv shows.

In [25]:

```
df['title'].value_counts()
```

Out[25]:

```
Kahlil Gibran's The Prophet                          700
Holidays                                             504
Movie 43                                             468
The Eddy                                             416
Narcos                                               378
                                                    ...
Marc Maron: Thinky Pain                                1
Lo and Behold: Reveries of the Connected World         1
Miniforce: Super Dino Power                            1
Edmilson Filho: Notas, Comedy about Relationships      1
Dick Johnson Is Dead                                   1
Name: title, Length: 8777, dtype: int64
```

Inference B: On exploding our data, a row in the original dataframe expanded to as high as 700 rows for title Kahlil Gibran's The Prophet while few rows didn't experienced any change.

In [26]:

```
df['director'].value_counts()
```

Out[26]:
```
Anonymous                   50497
Martin Scorsese               419
Youssef Chahine               409
Cathy Garcia-Molina           356
Steven Spielberg              355
                            ...  
Charlie Siskel                  1
Jonathan Ignatius Green         1
Brendon Marotta                 1
Sharon Grimberg                 1
Kirsten Johnson                 1
Name: director, Length: 4994, dtype: int64
```

Inference C: Martin Scorsese occurs the most - 419 - number of times in our dataset; closely followed by Youssef Chahine - 409 times. These directors have worked in movies or tc shows that star many cast members and belong to many genres.

In [27]:
```
df['cast'].value_counts()
```

Out[27]:
```
Anonymous          2139
Liam Neeson         161
Alfred Molina       160
John Krasinski      139
Salma Hayek         130
                   ... 
Elise Loehnen         1
Wil Willis            1
Samantha Bee          1
Eleanor Rocha         1
Zbyněk Vičar          1
Name: cast, Length: 36433, dtype: int64
```

Inference D: Liam Neeson occurs the most - 161 - number of times in our dataset; closely followed by Alfred Molina - 160 times. These actors have worked in movies or tv shows that are directed by many directors and have many cast members and belong to many genres.

In [28]:
```
df['country'].value_counts()
```

Out[28]:
```
United States     59214
India             22733
United Kingdom    12965
Not Available     11620
Japan              8595
                  ...  
Panama                2
Mongolia              2
Kazakhstan            1
Nicaragua             1
Uganda                1
Name: country, Length: 124, dtype: int64
```

Inference E: Highest number of entries - 59214 - in our dataset corresponds to the US followed by India at 22733. These countries have multi-director, multi-starrer, multi-genre content.

In [29]:
```
df['date_added'].value_counts().head(50)
```

Out[29]:
```
2020-01-01    3748
2019-11-01    2258
2021-07-01    2219
2017-10-01    1899
2021-09-01    1756
2018-03-01    1752
2019-12-31    1695
2019-10-01    1563
2018-10-01    1419
2021-06-02    1260
2021-08-01    1248
2021-01-01    1216
2017-09-01    1210
2018-01-01    1141
2018-11-01    1128
2021-07-06    1061
2019-01-01    1059
2017-08-01    1008
2020-04-01    1000
2019-09-01     970
2020-11-01     896
2017-07-01     888
2020-06-01     879
2020-10-01     867
2019-07-01     866
2018-04-01     847
2021-05-01     825
2017-05-01     796
2019-03-01     792
2021-04-01     792
2020-07-05     790
2019-02-01     787
2019-11-20     761
2020-09-01     753
2018-08-01     753
2018-07-01     747
2016-01-01     726
2018-12-01     715
2017-12-01     713
2021-08-27     702
2019-08-01     696
2021-06-19     685
2019-12-01     680
2017-11-01     664
2019-12-15     640
2020-10-19     628
2020-12-01     615
2021-04-16     610
2017-06-01     597
2017-03-10     593
Name: date_added, dtype: int64
```

Inference F: Most of the movies are added on the platform within first week of a month.

In [30]:
```
df['release_year'].value_counts()
```

Out[30]:
```
2018    24198
2019    21756
2017    20516
2020    19652
2016    18465
        ...  
1947        8
1946        6
1942        6
1943        5
1925        1
Name: release_year, Length: 74, dtype: int64
```

Inference G: Audience prefer newer content.

In [31]:
```
df['rating'].value_counts()
```

Out[31]:
```
TV-MA            73675
TV-14            43597
R                25817
PG-13            16222
TV-PG            14893
PG               10903
TV-Y7             6304
TV-Y              3662
TV-G              2749
NR                1573
G                 1530
NC-17              149
TV-Y7-FV            86
UR                  86
Not Available       70
Name: rating, dtype: int64
```

Inference H: Highest number of entries - 73695 - in our dataset corresponds to the TV-MA followed by TV-14 at 43597. These are two most preferred rating types among audience.

In [32]:
```
df['listed_in'].value_counts()
```

Out[32]:
```
Dramas                          29713
International Movies            28084
Comedies                        20758
International TV Shows          12797
Action & Adventure              12167
Independent Movies               9810
Children & Family Movies         9755
TV Dramas                        8894
Thrillers                        7036
Romantic Movies                  6411
TV Comedies                      4963
Crime TV Shows                   4733
Kids' TV                         4565
Horror Movies                    4538
Sci-Fi & Fantasy                 4029
Romantic TV Shows                3049
Music & Musicals                 3028
Documentaries                    2409
Anime Series                     2313
TV Action & Adventure            2288
Spanish-Language TV Shows        2088
British TV Shows                 1808
Sports Movies                    1531
Classic Movies                   1443
TV Mysteries                     1281
Korean TV Shows                  1122
Cult Movies                      1067
TV Sci-Fi & Fantasy              1045
Anime Features                   1017
TV Horror                         941
Docuseries                        845
LGBTQ Movies                      838
TV Thrillers                      768
Teen TV Shows                     742
Reality TV                        735
Faith & Spirituality              719
Stand-Up Comedy                   540
Movies                            412
TV Shows                          337
Classic & Cult TV                 272
Stand-Up Comedy & Talk Shows      268
Science & Nature TV               157
Name: listed_in, dtype: int64
```

Inference I: Highest number of entries - 29713 - in our dataset corresponds to the Dramas followed by Comedy at 20758.

Data Vizualization

Let's start building some intuitive plots to get to know the data better.

In [33]:
```
df.head()
```

Out[33]:

	type	title	director	cast	country	date_added	release_year	rating
0	Movie	Dick Johnson Is Dead	Kirsten Johnson	Anonymous	United States	2021-09-25	2020	PG-13
1	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
2	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
3	TV Show	Blood & Water	Anonymous	Ama Qamata	South Africa	2021-09-24	2021	TV-MA
4	TV Show	Blood & Water	Anonymous	Khosi Ngema	South Africa	2021-09-24	2021	TV-MA

In [34]:

```
# defining some functions for plotting graphs with minimal code repetit
def update_fig_layout_wo_legend(fig, title, x_title, y_title):
    fig.update_layout(
        xaxis={'visible': True, 'showticklabels': True},
        font_family='monospace',
        font_color='black',
        title_x=0.50,
        title_y=0.95,
        title_text=title,
        xaxis_title=x_title,
        yaxis_title=y_title,
        plot_bgcolor='rgba(255,255,255,255)',
        paper_bgcolor='rgba(255,255,255,255)',
        showlegend=False,
        # hovermode=False
        )
    fig.show()
    return None

def update_fig_layout_w_legends(fig, title, x_title, y_title):
    fig.update_layout(
        xaxis={'visible': True, 'showticklabels': True},
```

```
        font_family='monospace',
        font_color='black',
        title_x=0.50,
        title_y=0.95,
        title_text=title,
        xaxis_title=x_title,
        yaxis_title=y_title,
        plot_bgcolor='rgba(255,255,255,255)',
        paper_bgcolor='rgba(255,255,255,255)',
        showlegend=True,
        # hovermode=False
        )
    fig.show()
    return None

def plot_bar(x, y, title, x_title, y_title):
    fig = px.bar(x=x, y=y, width=1250)
    update_fig_layout_wo_legend(fig, title, x_title, y_title)
    return None

def plot_line(x, y, title, x_title, y_title):
    fig = px.line(x=x, y=y, width=1250)
    update_fig_layout_wo_legend(fig, title, x_title, y_title)
    return None

def plot_histogram(x, title, x_title, y_title):
    fig = px.histogram(x=x)
    update_fig_layout_wo_legend(fig, title, x_title, y_title)
    return None

def plot_box(x, y, title, x_title, y_title):
    fig = px.box(x=x, y=y)
    update_fig_layout_wo_legend(fig, title, x_title, y_title)
    return None

def plot_box_one_variable(x, title, y_title):
    fig = px.box(x)
    update_fig_layout_wo_legend(fig, title, '', y_title)
    return None

def plot_dodge_bar(data, x, y, color, title, x_title, y_title):
    fig = px.bar(data_frame=data,x=x, y=y, color=color, barmode='group'
    update_fig_layout_w_legends(fig, title, x_title, y_title)
    return None

def plot_bar_with_running_total(x, y, title, x_title, y_title):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=x, y=y.cumsum()))
    fig.add_trace(go.Bar(x=x, y=y))
    update_fig_layout_wo_legend(fig, title, x_title, y_title)
    return None

def plot_heatmap(data, fig_num):
    fig = px.imshow(data.corr(), labels=dict(color='Correlation'),
    x=['Movie Released Year', 'Movie Added Year', 'Duration'], y=['Movi
    update_fig_layout_w_legends(fig, f'{fig_num} Correlation Matrix', '
    return None
```

```
def plot_heatmap_1(data, fig_num):
    fig = px.imshow(data.corr(), labels=dict(color='Correlation'),
    x=['Movie Released Year', 'Number of Directors', 'Number of Actors'
    update_fig_layout_w_legends(fig, f'{fig_num} Correlation Matrix', '
    return None
```

We have a total of 10817 unique (title, country) pairs for 124 countries. Let's plot a barplot for top 20 contributing countries. We would have to remove "Not Available" entry from the dataset to truly capture top 20's list because "Not Available" sits at position 3. Can be verified using `.index` attribute

We will also perform similar analysis for `director`, `cast`, `rating`, and `listed_in` columns and try to figure out who are busiest directors and actors and what rating and genres are popular among the audience

In [35]:

```
itr_list = [
    ('country', 'Not Available', 'Number of Movies + TV Shows by Country
    ('director', 'Anonymous', 'Number of Movies + TV Shows by Director'
    ('cast', 'Anonymous', 'Number of Movies + TV Shows by Actor', 'Acto
    ('rating', 'Not Available', 'Number of Movies + TV Shows by Rating'
    ('listed_in', None, 'Number of Movies + TV Shows by Genre', 'Genre'
]

for idx, (col, search_phrase, title, x_title, y_title, top_n) in enumera
    temp = df[['title', col]].drop_duplicates()[col].value_counts()
    x, y = np.array(temp.index), np.array(temp.values)

    if search_phrase:
        delete_at = np.where(x==search_phrase)[0][0]
        x, y = np.delete(x, delete_at), np.delete(y, delete_at)

    x, y = x[:top_n], y[:top_n]

    plot_bar_with_running_total(x, y, f'Figure {idx + 1}: ' + title, x_
```

Inferences From Figure 1:

Inference 1: United States, India, United Kingdom, Canada, and France are the top 5 content contibuting countries.

Inference 2: More than 85% of the content comes from only 20 out of 124 countries.

Inference 3: Only 17 countries contribute 100 or more Movies / TV Shows.

Inferences From Figure 2:

Inference 4: Rajiv Chilaka has directed most number of content for Netflix and is closely followed by Jan Suter.

Inference 5: In contrast to Figure 1, we see that our data is not skewed on Directors. We see that top 20 directors out of a total of 4994 has directed only 2.5% of all the content present in our dataset.

Inferences From Figure 3:

Inference 6: Anupam Kher has acted in the most number of content for Netflix and is closely followed by Shah Rukh Khan.

Inference 7: Inline with Figure 2, we see that our data is also not skewed on Cast. We see that top 20 actors out of more than 36000 has acted only 0.8% of all the content present in our dataset.

Inferences From Figure 4:

Inference 8: TV-MA is the most preferred rating among the audience followed by TV-14.

Inference 9: Out of content available of 15 rating types, top 3 capture almost 71% of the content.

Inferences From Figure 5:

Inference 10: International Movies and International TV Shows are preferred all across the globe.

Inference 11: Drama, Comedy, and Documentary are the top 3 preferred Genres by the audience, in the order as is.

Next, let's get some insights on duration column

In [36]:

```
temp_df = df.drop_duplicates(subset=['title', 'duration'])[['type', 'du
movie_dist = temp_df[temp_df['type'] == 'Movie']['duration']
show_dist = temp_df[temp_df['type'] == 'TV Show']['duration']
```

In [37]:

```
plot_histogram(show_dist, 'Figure 6: Distribution of Number of Seasons'
```

Inferences From Figure 6:

Inference 12: 2413 TV Shows out of 2672 have at a maximum of 3 Seasons. With this we could assume that audience like TV Shows with lesser seasons.

In [38]:

```
plot_histogram(movie_dist, 'Figure 7: Distribution of Duration (mins)',
```

Inferences From Figure 7:

Inference 13: We see a near perfect Normal Distribution (if we ignore observations on higher-end band of duration). The peak of the distribution is around 96 minutes with most of the movies having duration between 86 to 106 minutes.

In [39]:

```
temp_df = df.drop_duplicates(subset=['title', 'duration'])[['type', 'du
movie_dist = temp_df[temp_df['type'] == 'Movie'][['listed_in', 'duratio
show_dist = temp_df[temp_df['type'] == 'TV Show'][['listed_in', 'durati
```

In [40]:

```
plot_box(show_dist['listed_in'], show_dist['duration'], 'Figure 8: Genr
```

Inferences From Figure 8:

Inference 14: TV Shows belonging to Classic & Cult tend to be longer with half of them having atleast 4 seasons.

Inference 15: Drama, Comedy, Action & Adventure are the next three genres who tend to have longer TV Shows with half of shows of each of these categories having atleast 2 seasons.

In [41]:
```
plot_box(movie_dist['listed_in'], movie_dist['duration'], 'Figure 9: Ge
```

Inferences From Figure 9:

Inference 16: Movies belonging to Romantic, LGBTQ, and Sports tend to have a very consolidated range of their duration.

Inference 17: Drama, Comedy, Action & Adventure, and Classical are the four genres who tend to have varied length of movies with duration as low as 5 minutes to as high as 250 minutes.

Next, Let's examine the number of Movies and TV Shows added vs released each year.

In [42]:
```
temp_df = df.drop_duplicates(subset=['title', 'date_added', 'release_yea
added_year = temp_df['date_added'].apply(lambda x: x[:4]).value_counts(
release_year = temp_df['release_year'].value_counts().reset_index()
temp_df = pd.merge(left=added_year, right=release_year, left_on='index'
temp_df = temp_df.fillna(0).rename({'index': 'Year', 'date_added': 'Add
temp_df = temp_df.melt(id_vars=['Year'], value_vars=['Added', 'Released
plot_dodge_bar(temp_df, 'Year', 'Count', 'Content', 'Figure 10: Year vs
```

Inferences From Figure 10:

Inference 18: With platform coming into existence in 2003, added its first 2 movies the very same year. Over the span of 13 years, 2003-2015, there was a constant-to-linear growth in number of content added to the platform while the growth of content released was exponential.

Inference 19: Over the next 6 years, growth of both content added and released was exponential with more content being added to the platform than released. This can be accounted for the fact that Netflix started adding Movies and TV Shows that were released in the past.

Inference 20: We saw a dip in both the number of content added and released in the recent years post 2018. While Covid can be held accountable for this decrease in the years of 2020 and 2021; we do not know why less contents were released in 2019 as compared to 2018.

Let's examine the above graph separately for Movies and TV Shows.

In [43]:
```
# for movies

temp_df = df.drop_duplicates(subset=['title', 'type', 'date_added', 're
temp_df = temp_df[temp_df['type'] == 'Movie']
added_year = temp_df['date_added'].apply(lambda x: x[:4]).value_counts(
release_year = temp_df['release_year'].value_counts().reset_index()
temp_df = pd.merge(left=added_year, right=release_year, left_on='index'
temp_df = temp_df.fillna(0).rename({'index': 'Year', 'date_added': 'Add
temp_df = temp_df.melt(id_vars=['Year'], value_vars=['Added', 'Released
plot_dodge_bar(temp_df, 'Year', 'Count', 'Movies', 'Figure 11: Year vs 

# for tv shows

temp_df = df.drop_duplicates(subset=['title', 'type', 'date_added', 're
temp_df = temp_df[temp_df['type'] == 'TV Show']
added_year = temp_df['date_added'].apply(lambda x: x[:4]).value_counts(
release_year = temp_df['release_year'].value_counts().reset_index()
temp_df = pd.merge(left=added_year, right=release_year, left_on='index'
temp_df = temp_df.fillna(0).rename({'index': 'Year', 'date_added': 'Add
temp_df = temp_df.melt(id_vars=['Year'], value_vars=['Added', 'Released
plot_dodge_bar(temp_df, 'Year', 'Count', 'TV Show', 'Figure 12: Year vs
```

Inferences From Figure 11:

Inference 21: Trend is similar to what we saw in Figure 10. Similar inferences can be drawn for Movies from this figure as for Movies + TV Shows from the previous figure.

Inferences From Figure 12:

Inference 22: Trend in this figure deviate from what we saw in Figure 10 and Figure 11. Here we see a continuous rise in TV Shows released until 2020 in contrast to 2018 in the previous two figures. We can assume that this could happen either because TV Shows are longer than movies and would take more time in editting and post-shoot stuff or because TV Shows grew more popular in recent years and audience preferred them over movies.

Next, let's plot a heatmap of correlation between content release year, content added year and, duration.

In [44]:
```
temp_df = df.drop_duplicates(subset=['title', 'type', 'date_added', 're
temp_df = temp_df[temp_df['type'] == 'Movie'][['release_year', 'date_ad
temp_df['date_added'] = pd.to_datetime(temp_df['date_added']).dt.year
temp_df = temp_df.astype(int)
plot_heatmap(temp_df, 'Figure 13: Movies -')

temp_df = df.drop_duplicates(subset=['title', 'type', 'date_added', 're
temp_df = temp_df[temp_df['type'] == 'TV Show'][['release_year', 'date_
temp_df['date_added'] = pd.to_datetime(temp_df['date_added']).dt.year
temp_df = temp_df.astype(int)
plot_heatmap(temp_df, 'Figure 14: TV Shows -')
```

Inferences From Figure 13:

Inference 23: Duration of movies decreased over time.

Inferences From Figure 14:

Inference 24: TV Shows with more seasons were added over time.

Inference 25: More TV Shows were released and added over time as can be verified from Figure 12.

In [45]:
```
temp_df = df.drop_duplicates(subset=['title', 'director', 'cast', 'relea
plot_heatmap_1(temp_df, 'Figure 15: ')
```

Inferences From Figure 15:

Inference 26: Good correlation observed observed between year and number of movies and/or tv shows released.

Inference 27: More and more directors and actors debutted over time in more and more movies and tv shows.

Outlier Analysis

In [46]:

```
m_temp = df[df['type'] == 'Movie'].drop_duplicates(subset=['title', 'du
s_temp = df[df['type'] == 'TV Show'].drop_duplicates(subset=['title', '

plot_box_one_variable(m_temp, 'Figure 16: Outlier Detection in duration
plot_box_one_variable(s_temp, 'Figure 17: Outlier Detection in # Season
```

Inferences From Figure 16:

Inference 28: The outlier range for movies is above 154 and below 47; and there are clearly many outliers present in the dataset having duration in these outlier range.

Inferences From Figure 17:

Inference 29: The outlier range for tv shows is above 3; and there are clearly many tv shows present in the dataset having duration in this outlier range.

Let's look at if any titles were added to the platform before their release date or not? ideally this should not happen but let's just check.

In [47]:

```
temp_df = df.drop_duplicates(subset=['title'])[['title', 'release_year'
temp_df['date_added'] = pd.to_datetime(temp_df['date_added']).dt.year.a
temp_df['release_year'] = temp_df['release_year'].astype(int)
print(temp_df[temp_df['release_year'] > temp_df['date_added']].sort_val
```

```
                         title  release_year  date_added
0                  Jack Taylor          2016        2013
1                  Tokyo Trial          2017        2016
2                       Sense8          2018        2016
3   Hans Teeuwen: Real Rancour          2018        2017
4         Arrested Development          2019        2018
5                     Incoming          2019        2018
6     Unbreakable Kimmy Schmidt          2019        2018
7             BoJack Horseman          2020        2019
8                 Fuller House          2020        2019
9            Maradona in Mexico          2020        2019
10            The Hook Up Plan          2020        2019
11                       Hilda          2021        2020
12               Love Is Blind          2021        2020
13                Polly Pocket          2021        2020
```

Inference 30: In contrast to what we believed, we saw that 14 titles were infact added on the platform before there release date. This could be a result of wrong data entry or if the trailers were made available on Netflix in name of the movie.

Conclusion

---

1. Defining Problem Statement and Analysing basic metrics. (10 Points)

Analyze the data and generate insights that could help Netflix in deciding which type of TV Show/Movie to produce and how to grow the business in different countries.

Method 1: Exploratory Data Analysis - Knowing the data; filling in missing data; transforming the data;

Method 2: Uni-Variate Analysis - Histograms; Count Plots

Method 3: Multi-Variate Analysis - Bar Plots; Line Plots; Box Plots; Pair Plots

Method 4: Correlation - Heatmaps; Correlation Matrix

2: Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary. (10 Points)

Statistical Summary of the Raw Data:

type - 2 unique values - 'Movie' and 'TV Show' with frequency of 6131 and 2676 respectively.

title - 8807 unique values.

director - 4528 unique values with 'Rajiv Chilaka' occurring 19 times. 2634 NULL values.

cast - 7692 unique values with David Attenborough occurring 19 times. 825 NULL values.

country - 748 unique values with 'United States' occurring 2818 times. 831 NULL values.

date_added - 1714 unique values with '2020-01-01' occurring 110 times. 10 NULL values.

release_year - 74 unique values with '2018' occurring 1147 times.

rating - 17 unique values with 'TV-MA' occurring 3207 times. 4 NULL values.

duration - 220 unique values with '1 Season' occurring 1793 times. 3 NULL values.

listed_in - 514 unique values with 'Dramas, International Movies' occurring 362 times.

description - 8775 unique values with 'Paranormal activity at a lush, abandoned prope...' occurring 4 times.

Statistical Summary of the Processed Data:

type - 2 unique values - 'Movie' and 'TV Show' with frequency of 145305 and 56011 respectively.

title - 8777 unique values with 'Kahlil Gibran's The Prophet' occurring 700 times.

director - 4994 unique values with 'Anonymous' occurring 50497 times.

cast - 36433 unique values with 'Anonymous' occurring 2139 times.

country - 124 unique values with 'United States' occurring 59214 times.

date_added - 1719 unique values with '2020-01-01' occurring 3748 times.

release_year - 74 unique values with '2018' occurring 24198 times.

rating - 15 unique values with 'TV-MA' occurring 73675 times.

duration - Max value = 312; Min Value = 1; Mean Value = 77.58; Median Value = 95

listed_in - 42 unique values with 'Dramas' occurring 29713 times.

description - 8775 unique values with 'A troubled young girl and her mother find sola...' occurring 700 times.

Property of the Data	Raw Data Statistics	Processed Data Statistics
Shape of the Data	(8807, 12)	(201364, 11)
Data types of columns	int - release_year object - all other columns	int - duration object - all other columns
Conversion of data types	release_year: int duration: object	release_year: object duration: int
Missing Values	Present in director, cast, country, date_added, rating, duration	None Present

---

3. Non-Graphical Analysis: Value counts and unique attributes. (10 Points)

a. There are almost 2.5 times Movie titles in our dataset than TV shows.

b. On exploding our data, a row in the original dataframe expanded to

700 for title Kahlil Gibran's The Prophet while few remained they were

c. Martin Scorsese occurs the most - 419 - number of times in our dataset; closely followed by Youssef Chahine - 409 times.

d. Liam Neeson occurs the most - 161 - number of times in our dataset; closely followed by Alfred Molina - 160 times.

e. Highest number of entries - 59214 - in our dataset corresponds to the US followed by India at 22733.

f. Most of the movies are added on the platform on 1st of the months.

g. More movies from recent years are present in our dataset.

h. Highest number of entries - 73695 - in our dataset corresponds to the TV-MA followed by TV-14 at 43597.

i. Highest number of entries - 29713 - in our dataset corresponds to the Dramas followed by International Movies at 28084.

j. Note: Number of unique values in all these columns are mentioned in the Statistical Summary in the asnwer of Q2.

---

4. Visual Analysis - Univariate, Bivariate after pre-processing of the data. Note: Pre-processing involves unnesting of the data in columns like Actor, Director, Country.

Refer to Figure 1 through 15.

---

5. Missing Value & Outlier check (Treatment optional). (10 Points)

Missing values were observed in columns director, cast, country, date_added, rating, duration columns and were treated as follows:

- For columns director and cast, NaNs were replaced with "Anonymous"

- For columns country and rating, NaNs were replaced with "Not Available"

- For column date_added, NaTs were replaced with 31st December of the content released year.

- For column duration, NaNs were a result of wrong data entry; these values were present in the corresponding rows of rating column.

Outliers were observed in columns duration and date_added & release_year.

- Refer to Figure 16 and 17 and Inference 28 to 30.

---

6. Insights based on Non-Graphical and Visual Analysis. (10 Points)

Refer to Inference A to I and 1 to 30.

Comments on range of columns - date_added, release_year, and duration.

- Netflix added its first content on Dec 31st, 2003. The latest content was uploaded on the platform on Sept 25th, 2021

- Netflix added its first content on Dec 31st, 2003. The latest content was uploaded on the platform on Sept 25th, 2021

- Duration of a movie on Netflix ranges from 3 to 312 minutes.

- TV Shows on Netflix have a mximum of 17 seasons while minimum being 1.

---

7. Business Insights (10 Points) - Should include patterns observed in the data along with what you can infer from it.

Please refer to Infrence 1 to 30 above for detailed answer of this question. Some of the most valuable insights are written below.

- International Movies and TV Shows are preferred by audience all across the globe.

- TV-MA, TV-14, and R are the most common ratings among the customer-base of Netflix.

- Movies and TV Shows from US, India, and UK are the most popular ones among the people.

- Dramas, Comedies, and Documentaries are the most common genres for TV Shows.

- Dramas, Comedies, Documentaries, and Classical are the most common genres for Movies.

- Audience love watching TV Shows with less then 2 seasons and Movies about 100 minutes long.

---

8. Recommendations (10 Points) - Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

- Netflix can introduce regional content in most popular genres (Dramas, Comedies, and Documentaries) and ratings (TV-MA, TV-14, and R) to increase its customer base in less popular regions for example Europe and Africa.
- Netflix can produce longer movies and shows in the popular genres - Dramas, Comedies, and Documentaries - because these genres are loved by the audience and new shows will not find it hard to find people's love. This would account for recurring payments for Netflix because longer shows will span over months or years and people would have to pay to

watch what they love.

- Since most of the movies and shows on Netflix are added in the first week of the month, the Engineering Team should maintain and scale servers accordingly for this week and also for the rest of the month.
- Since the database does not have many old movies or shows, it is worth the shot to add these contents on Netflix to see if they get any traction from the audience.
- Add new movies and shows belonging to less popular ratings and genres to give audience more options in these unppopular areas to watch from and who know they may love the new content.
- Netflix can introduce a weekly subscription plan that is valid only for the first week of a month. Since most of the new movies and shows are added in the first week, people signing up for this subscription will try to consume as much content as possible and this could form a habit and thus produce a loyal customer of Netflix.
- Netflix can bring together most popular cast members and directors under one umbrella to produce shows and movies that would be popular and accepted by a very large base of audience.