The Walt Disney Company

Bruno Athayde e Silva

Table of Contents

Introduction

Dataset Description

The description below was taken from the Wikipedia website dedicated to The Walt Disney Company.

"The Walt Disney Company, commonly known as Disney is an American multinational mass media and entertainment conglomerate headquartered at the Walt Disney Studios complex in Burbank, California.

Disney was originally founded on October 16, 1923, by brothers Walt and Roy O. Disney as the Disney Brothers Cartoon Studio; it also operated under the names the Walt Disney Studio and Walt Disney Productions before changing its name to the Walt Disney Company in 1986. The company established itself as a leader in the American animation industry before diversifying into live-action film production, television, and theme parks.

Since the 1980s, Disney has created and acquired corporate divisions in order to market more mature content than is typically associated with its flagship family-oriented brands. The company is known for its film studio division, Walt Disney Studios, which includes Walt Disney Pictures, Walt Disney Animation Studios, Pixar, Marvel Studios, Lucasfilm, 20th Century Studios, 20th Century Animation, and Searchlight Pictures."

I will use the `disney-characters.csv` and `disney_movies_total_gross.csv` for my analysis.

- **disney-characters**
  - This table contains information about the movies and each characters (hero or villain type) in it.
- **disney_movies_total_gross**
  - This table contains information about the Disney movies box office.

Question of Interest

This analysis will investigate the relationship between box office gross and Disney characters of children's movies. This will also investigate the relationship between box office gross and movies' genre.

I am interested in finding out if exists any correlation between those variables. For example, one would think that movies with strong heroes like Moana, Alladin, or Elsa would be more popular and, consequently, bring more revenue to the company. Or one would also think that movies' genres are related to a movie success or not.

Methodology

This project aims to assess the data provided, capturing some insights present in this dataset.

The dataset provided will be analyzed in a way we can get some trends out of it, showing how powerful characters may impact the final movie revenue and/or how movie's genres can impact its revenue.

In [1]:
```
# import libraries needed for the analysis

import numpy as np
import pandas as pd
import altair as alt
```

In [2]:
```
# load dataset from CSV files

character_df = pd.read_csv('disney-characters.csv', delimiter = ',')
gross_df = pd.read_csv('disney_movies_total_gross.csv', delimiter = ',')
```

In [3]:
```
# check the data stored in the character dataframe
character_df.head()
```

Out[3]:

	movie_title	release_date	hero	villian	song
0	\nSnow White and the Seven Dwarfs	December 21, 1937	Snow White	Evil Queen	Some Day My Prince Will Come
1	\nPinocchio	February 7, 1940	Pinocchio	Stromboli	When You Wish upon a Star
2	\nFantasia	November 13, 1940	NaN	Chernabog	NaN
3	Dumbo	October 23, 1941	Dumbo	Ringmaster	Baby Mine
4	\nBambi	August 13, 1942	Bambi	Hunter	Love Is a Song

In [4]:
```
# check the data stored in the dataframe
# check data types and missing values
character_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   movie_title   56 non-null     object
 1   release_date  56 non-null     object
 2   hero          52 non-null     object
 3   villian       46 non-null     object
 4   song          47 non-null     object
dtypes: object(5)
memory usage: 2.3+ KB
```

The dataframe above (characters_df) has 56 rows and 5 columns.

The character_df presents a *movie_title*, a *release_date*, a *hero*, a *villain* and a *song* columns. All columns present an object data type.

The dataframe also presents some null values that will be addressed later on.

In [5]:
```
# check the data stored in the gross_df dataframe
gross_df.head()
```

Out[5]:

	movie_title	release_date	genre	MPAA_rating	total_gross	inflation_adjusted_gross
0	Snow White and the Seven Dwarfs	Dec 21, 1937	Musical	G	$184,925,485	$5,228,953,251
1	Pinocchio	Feb 9, 1940	Adventure	G	$84,300,000	$2,188,229,052
2	Fantasia	Nov 13, 1940	Musical	G	$83,320,000	$2,187,090,808
3	Song of the South	Nov 12, 1946	Adventure	G	$65,000,000	$1,078,510,579
4	Cinderella	Feb 15, 1950	Drama	G	$85,000,000	$920,608,730

In [6]:
```
# check the data stored in the dataframe
# check data types and missing values
gross_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579 entries, 0 to 578
Data columns (total 6 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   movie_title               579 non-null    object
 1   release_date              579 non-null    object
 2   genre                     562 non-null    object
 3   MPAA_rating               523 non-null    object
 4   total_gross               579 non-null    object
 5   inflation_adjusted_gross  579 non-null    object
dtypes: object(6)
memory usage: 27.3+ KB
```

The dataframe above (gross_df) has 579 rows and 6 columns.

The gross_df presents a *movie_title*, a *release_date*, a *genre*, a *MPAA_rating*, a *total_gross* and an *inflation_adjusted_gross*. All columns present an object data type.

The dataframe also presents some null values that will be addressed later on.

As said above, both dataframes contain only object data types observations, which may represent a problem in the future when assessing the data and should be treated accordingly.

That is the next step I will take.

In [7]:
```
# change data type of the release_date for both dataframes

character_df['release_date'] = pd.to_datetime(character_df['release_date'])
gross_df['release_date'] = pd.to_datetime(gross_df['release_date'])
```

In [8]:
```
# check if the data types have changed

print(character_df.info())
print(gross_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   movie_title   56 non-null     object
 1   release_date  56 non-null     datetime64[ns]
 2   hero          52 non-null     object
 3   villian       46 non-null     object
 4   song          47 non-null     object
dtypes: datetime64[ns](1), object(4)
memory usage: 2.3+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579 entries, 0 to 578
Data columns (total 6 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   movie_title               579 non-null    object
 1   release_date              579 non-null    datetime64[ns]
 2   genre                     562 non-null    object
 3   MPAA_rating               523 non-null    object
 4   total_gross               579 non-null    object
 5   inflation_adjusted_gross  579 non-null    object
dtypes: datetime64[ns](1), object(5)
```

```
memory usage: 27.3+ KB
None
```

The **character_df** seems to have the correct data type for its columns now that the *release_date* had been changed; however, I need to eliminate the extra character (\n) from *movie_title* column. On the other hand, the **gross_df** is still presenting data type *object* where it should have floats for example. It also should be converted to an appropriate data type.

In [9]:

```
# eliminate "\n" from the movie_titles column on the character_df

character_df['movie_title'] = character_df['movie_title'].str.replace('\n', '', regex = True)
character_df.head()
```

Out[9]:

	movie_title	release_date	hero	villian	song
0	Snow White and the Seven Dwarfs	1937-12-21	Snow White	Evil Queen	Some Day My Prince Will Come
1	Pinocchio	1940-02-07	Pinocchio	Stromboli	When You Wish upon a Star
2	Fantasia	1940-11-13	NaN	Chernabog	NaN
3	Dumbo	1941-10-23	Dumbo	Ringmaster	Baby Mine
4	Bambi	1942-08-13	Bambi	Hunter	Love Is a Song

In [10]:

```
def convert_type(dataframe):
    '''
    Given a dataframe, remove special characters and convert string to float

    Parameters
    ----------
    dataframe: pandas.core.frame.DataFrame
        The dataframe to work with.

    Returns
    -------
    dataframe: pandas.core.frame.DataFrame
        The new dataframe

    Examples
    --------
    >>> convert_type(gross_df)
```

```
        movie_title release_date genre    MPAA_rating  total_gross  inflation_adjusted_gross
    1   Pinocchio   1940-02-09   Adventure G           84300000.0   2.188229e+09
    '''


    if isinstance (dataframe, pd.DataFrame) == False:
        raise TypeError("This is not a Dataframe!")

    # remove the $ sign

    dataframe['total_gross'] = dataframe['total_gross'].str.replace('[\$]', '', regex = True)
    dataframe['inflation_adjusted_gross'] = dataframe['inflation_adjusted_gross'].str.replace('[\$]', '', regex


    # remove the comma

    dataframe['total_gross'] = dataframe['total_gross'].str.replace(',', '', regex = True)
    dataframe['inflation_adjusted_gross'] = dataframe['inflation_adjusted_gross'].str.replace(',', '', regex =


    # convert total_gross and inflation_adjusted_gross from the gross_df to float

    dataframe['total_gross'] = dataframe['total_gross'].astype('float')
    dataframe['inflation_adjusted_gross'] = dataframe['inflation_adjusted_gross'].astype('float')


    return dataframe
```

In [11]:

```
# use function to convert data type of gross_df

convert_type(gross_df)
gross_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579 entries, 0 to 578
Data columns (total 6 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   movie_title               579 non-null    object
 1   release_date              579 non-null    datetime64[ns]
 2   genre                     562 non-null    object
 3   MPAA_rating               523 non-null    object
 4   total_gross               579 non-null    float64
 5   inflation_adjusted_gross  579 non-null    float64
```

```
dtypes: datetime64[ns](1), float64(2), object(3)
memory usage: 27.3+ KB
```

In [12]:

```
#  merge the 2 dataframes on movie_title

combined_df = pd.merge(character_df, gross_df, on = ['movie_title'], how = 'inner')

combined_df.head()
```

Out[12]:

	movie_title	release_date_x	hero	villian	song	release_date_y	genre	MPAA_rating	total_gross	inflation_adjuste
0	Snow White and the Seven Dwarfs	1937-12-21	Snow White	Evil Queen	Some Day My Prince Will Come	1937-12-21	Musical	G	184925485.0	5.228
1	Pinocchio	1940-02-07	Pinocchio	Stromboli	When You Wish upon a Star	1940-02-09	Adventure	G	84300000.0	2.188
2	Fantasia	1940-11-13	NaN	Chernabog	NaN	1940-11-13	Musical	G	83320000.0	2.187
3	Cinderella	1950-02-15	Cinderella	Lady Tremaine	Bibbidi-Bobbidi-Boo	1950-02-15	Drama	G	85000000.0	9.206
4	Cinderella	1950-02-15	Cinderella	Lady Tremaine	Bibbidi-Bobbidi-Boo	2015-03-13	Drama	PG	201151353.0	2.011

In [13]:

```
# rename the release_date column and drop the duplicated one

combined_df = combined_df.drop(columns = ['release_date_x'])
combined_df = combined_df.rename(columns = {'release_date_y': 'release_date'})
```

In [14]:

```
# check for duplicate rows

combined_df.duplicated().sum()
```

Out[14]:

```
0
```

In [15]:
```
# check the null values in the dataframe

count_nan = combined_df.isnull().sum()
count_nan
```

Out[15]:
```
movie_title                 0
hero                        1
villian                     5
song                        5
release_date                0
genre                       1
MPAA_rating                 7
total_gross                 0
inflation_adjusted_gross    0
dtype: int64
```

In [16]:
```
# drop null values

combined_df = combined_df.dropna(axis = 0).reset_index(drop = True)
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   movie_title               32 non-null     object
 1   hero                      32 non-null     object
 2   villian                   32 non-null     object
 3   song                      32 non-null     object
 4   release_date              32 non-null     datetime64[ns]
 5   genre                     32 non-null     object
 6   MPAA_rating               32 non-null     object
 7   total_gross               32 non-null     float64
 8   inflation_adjusted_gross  32 non-null     float64
dtypes: datetime64[ns](1), float64(2), object(6)
memory usage: 2.4+ KB
```

Even though I can identify some null values in the *combined_df*, those will not affect my analysis at this point. So, I decided to keep those rows for now. I will reassess that later on.

In [17]:

```
# get the top 15 children's movies by inflation_adjusted_gross

combined_df_top = combined_df.nlargest(15, 'inflation_adjusted_gross').reset_index(drop = True)
combined_df_top.head()
```

Out[17]:

	movie_title	hero	villian	song	release_date	genre	MPAA_rating	total_gross	inflation_adjusted_gross
0	Snow White and the Seven Dwarfs	Snow White	Evil Queen	Some Day My Prince Will Come	1937-12-21	Musical	G	184925485.0	5.228953e+09
1	Pinocchio	Pinocchio	Stromboli	When You Wish upon a Star	1940-02-09	Adventure	G	84300000.0	2.188229e+09
2	Lady and the Tramp	Lady and Tramp	Si and Am	Bella Notte	1955-06-22	Drama	G	93600000.0	1.236036e+09
3	Cinderella	Cinderella	Lady Tremaine	Bibbidi-Bobbidi-Boo	1950-02-15	Drama	G	85000000.0	9.206087e+08
4	The Jungle Book	Mowgli	Kaa and Shere Khan	The Bare Necessities\n	1967-10-18	Musical	Not Rated	141843000.0	7.896123e+08

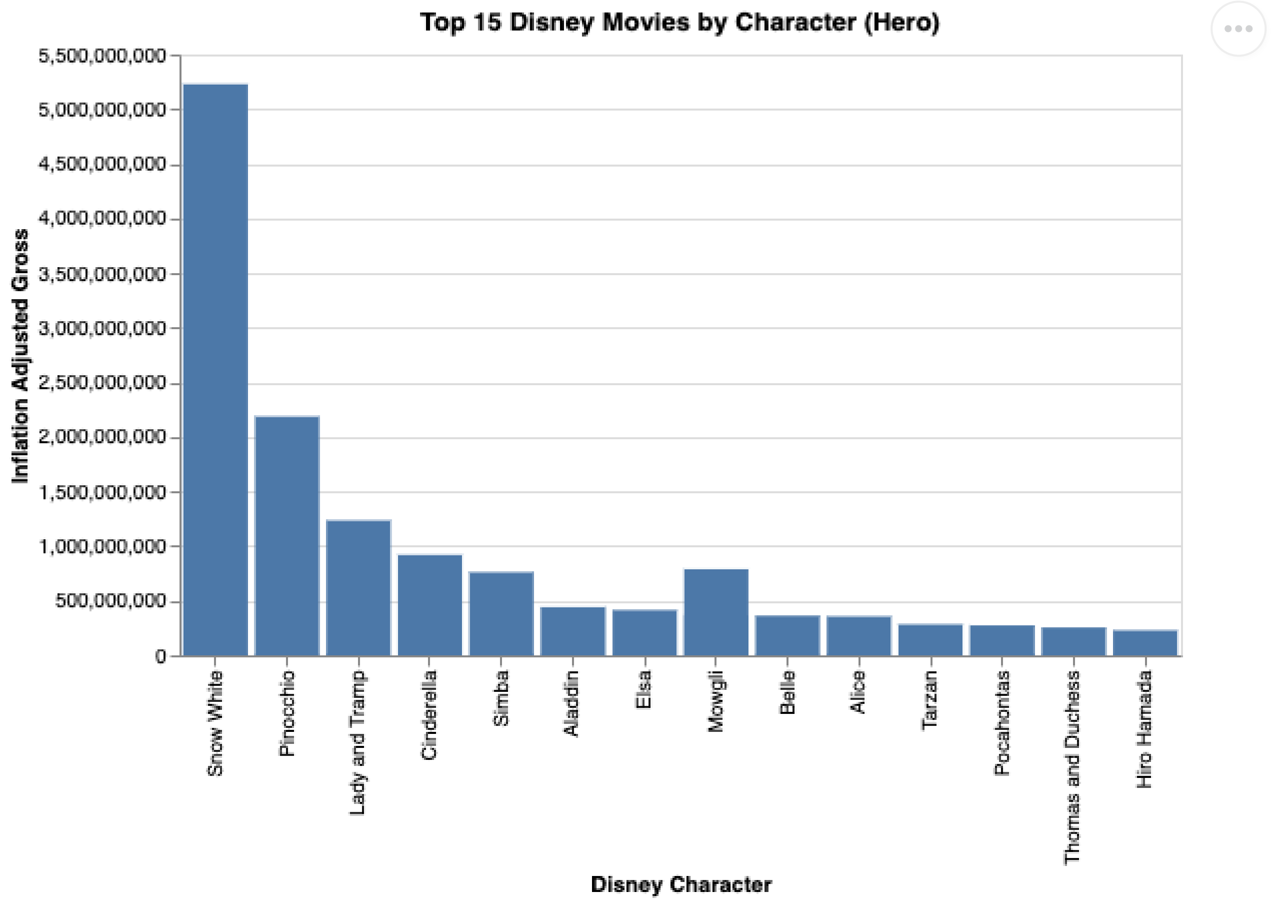
In [18]:

```
# use altair to generate a chart

top_15_plot = (
    alt.Chart(combined_df_top, width = 500, height = 300).
    mark_bar().
    encode(
        x = alt.X("hero:N", title = "Disney Character", sort = '-y'),
        y = alt.Y("inflation_adjusted_gross:Q", title = "Inflation Adjusted Gross"),
    ).properties (title = "Top 15 Disney Movies by Character (Hero)" )
)

top_15_plot
```

Out[18]:



From the above plot, we can identify that movies with Snow White, Pinocchio and Lady and Tramp represented the top 3 movies with the highest total gross revenue.

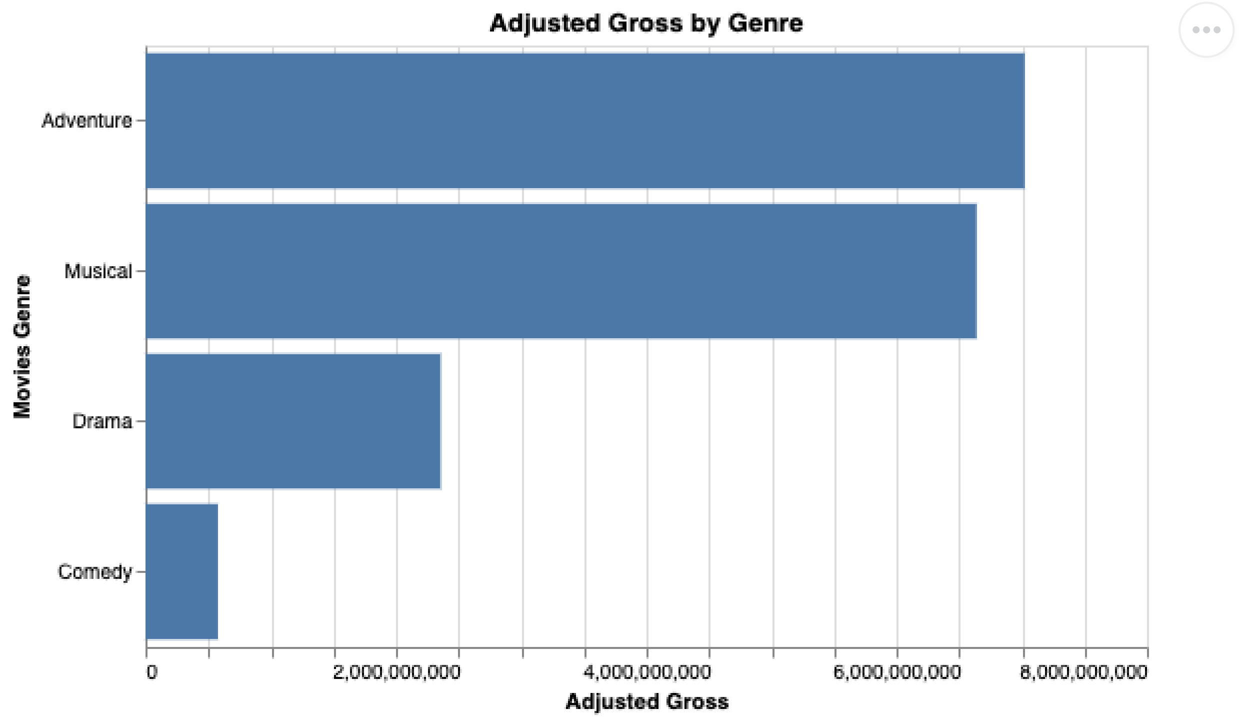
Surprisingly, Simba, Aladdin, Elsa and Tarzan, for instance, did not make it to the top.

In [19]:
```
# use altair to plot a bar chart

genre_chart = (
    alt.Chart(combined_df, width = 500, height = 300).
    mark_bar().
    encode(
        y = alt.Y("genre:N", title = "Movies Genre", sort = '-x'),
        x = alt.X("sum(inflation_adjusted_gross):Q", title = "Adjusted Gross"),
    ).properties(title = "Adjusted Gross by Genre")
)

genre_chart
```

Out[19]:



From the chart above, we can clearly see that adventure and musical movies represent the most significant revenue in children's movies.

This is somewhat expected, since those genres of movies also represent a big portion of the movies made by The Walt Disney Company.

Discussions

In this project, I analyzed The Walt Disney Company dataset focusing on children's movies, trying to find a relationship between the inflation-adjusted gross revenue, movies' genres and movies' hero characters. This project aimed to analyze data to predict the success of box office movies based on a solid hero character and its genres.

When comparing the revenue against the movie's characters, a few insights were not what I expected. For example, even though the analysis shows that Snow White, a strong female character, represented the highest adjusted gross revenue, it is surprising that Elsa, Aladdin or Pocahontas did not make the Top 5 movies with the highest adjusted gross.

Diving deeper into the genre of the movies, I can also get other important and surprising insights from the data. For example, not by surprise, adventure movies represent the highest adjusted gross; however, musical movies come in second place, pretty close to the adventure movies.

Another question that could be looked at given this dataset is the impact the director of a movie has on its success. Of course, a director's job is very subject; however, one thing that is undeniable and is more relevant than ever is the importance of having an identifiable and unique point of view.

Reference

- Data Source
    - This Disney database was obtained was curated by **Kelly Garret**.
- Question of Interest
    - This question of interest was inspired by **Kelly Garret** and **Linchen Zhen**
- The Walt Disney Company
    - The Wikipedia website regarding The Walt Disney Company