final_project

May 31, 2024

1 Final Project

Completed By : Vicky (Marie) Howe

1.1 Introduction

1.1.1 Exploration Question(s)

1. What genre of movie generated the most revenue over time?
2. What genre did a movie's director have the most sucess?
3. How did the yearly revenue generated perform by revenue source

1.1.2 Reasoning

I chose to analyse the financial performance of not only the types of movies, what directors are top performers but looking into how the different revenue streams performed over time to identify opportunities for increased revenue streams.

1.1.3 Dataset Description

The data sets used for this analysis were obtained from data.world which follows a Creative Common Attribution 4.0 International License.

There were 5 datasets available to use: disney-characters.csv disney-director.csv disney-voice-actors.csv disney_revenue_1991-2016.csv disney_movies_total_gross.csv Each file is a .csv containing information regarding Disney's revenue, actors, directors, box office success that include movie titles, revenue,characters, songs, directors ect. I will be using the `disney-director`, `disney_revenue_1991-2016` and `disney_movies_total_gross` tables in the manner described below:

- **disney-director.csv**
  - This file contains information on directors and movie titles they are associated with.
- **disney_revenue_1991-2016.csv**
  - This file includes information such as the pertenant year, revenue categories and the total of the categories.
- **disney_movies_total_gross.csv**
  - This file includes information such as the movie titles, both gross revenue and inflation adjusted gross value, genre, MPAA rating year, release date categories.

1.2 Methods & Results:

Since I am interested in the financial performance of movie genres, directors responsible and revenue streams, I will need to use the tables that containt the most data pertaining to this implying I will need to use the **disney-director.csv**, **disney_revenue_1991-2016.csv** and **disney_movies_total_gross.csv** tables.

Let's first import the tables.

[1]:
```
# Import Required Libraries
import altair as alt
import pandas as pd
import numpy as np

# Import Files Required
movies_df = pd.read_csv("data/disney_movies_total_gross.csv")
directors_df = pd.read_csv("data/disney-director.csv")
revenue_df = pd.read_csv("data/disney_revenue_1991-2016.csv")
```

Lets explore the movies table.

[2]:
```
movies_df.head(10)
```

[2]:

	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
5	20,000 Leagues Under the Sea	Dec 23, 1954	Adventure	NaN	$28,200,000	$528,279,994
6	Lady and the Tramp	Jun 22, 1955	Drama	G	$93,600,000	$1,236,035,515
7	Sleeping Beauty	Jan 29, 1959	Drama	NaN	$9,464,608	$21,505,832
8	101 Dalmatians	Jan 25, 1961	Comedy	G	$153,000,000	$1,362,870,985
9	The Absent Minded Professor	Mar 16, 1961	Comedy	NaN	$25,381,407	$310,094,574

[3]:
```
movies_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 movies table has 579 rows and 6 columns.
- Each movie_title has a release_date, genre, MPAA_rating, total_gross and an inflation_adjusted_gross revenue all type objects.

Now lets reorganize and cleanup our movies table

[4]:
```
import final_proj_clean_func as ps

movies_cleaned = ps.clean_movie_data(movies_df)

movies_cleaned.head()
```

[4]:

	movie_title	genre	gross_revenue
0	Snow White and the Seven Dwarfs	Musical	5228953251
1	Pinocchio	Adventure	2188229052
2	Fantasia	Musical	2187090808
3	Song of the South	Adventure	1078510579
4	Cinderella	Drama	920608730

Now lets revisit the first question, What genre of movie generated the most revenue over time?

Lets graph the data to see this relationship using altair.

[5]:
```
# Altair chart with genre_grouped data
genre_grouped_chart = alt.Chart(movies_cleaned, width=500, height=300).
 ↪mark_bar().encode(
                    x=alt.X('genre', sort = 'y', title='Movie Genre'),
                    y=alt.Y('gross_revenue', title='Gross Revenue ($)'),
).properties(title='Distribution of Gross Revenue by Movie Genre')

# Show the chart
genre_grouped_chart
```

[5]:
```
alt.Chart(…)
```

From the graph it appears the 'Musical' genre has a very large variance from all the others. Let's take look at this data a little closer by grouping the movies created per genre.

[6]:
```
# Group the data by genre
genre_grouped = movies_cleaned.groupby('genre').agg({'movie_title': 'count',␣
↪'gross_revenue': 'sum'}).reset_index()

# show the data
genre_grouped.head(12)
```

[6]:

	genre	movie_title	gross_revenue
0	Action	36	5349644857
1	Adventure	119	23464429224
2	Black Comedy	3	156730475
3	Comedy	162	14144323456
4	Concert/Performance	2	114821678
5	Documentary	16	203488418
6	Drama	103	7967675338
7	Horror	5	125346327
8	Musical	15	9566260328
9	Romantic Comedy	22	1720691633
10	Thriller/Suspense	23	2125628766
11	Western	7	516709946

Let's now graph this using Altair

[7]:
```
# Altair chart with updated data from genre_grouped
genre_grouped_chart = alt.Chart(genre_grouped, width=500, height=300).
↪mark_bar().encode(
    x=alt.X('genre', sort = 'y', title= 'Movies Grouped By Genre'),
    y=alt.Y('gross_revenue', title='Gross Revenue ($)'),
).properties(title='Distribution of Gross Revenue by Movies Grouped by Genre')

# Show the chart
genre_grouped_chart
```

[7]:
```
alt.Chart(…)
```

I notice there are alot more movies created in the adventure genre than any other thus netting the most revenue overall for Disney than any other genre.

Now lets have a look at the directors data table.

[8]:
```
directors_df.head(10)
```

[8]:

	name	director
0	Snow White and the Seven Dwarfs	David Hand
1	Pinocchio	Ben Sharpsteen
2	Fantasia	full credits

```
3                           Dumbo   Ben Sharpsteen
4                           Bambi       David Hand
5                 Saludos Amigos      Jack Kinney
6           The Three Caballeros  Norman Ferguson
7                Make Mine Music      Jack Kinney
8              Fun and Fancy Free      Jack Kinney
9                     Melody Time   Clyde Geronimi
```

Now lets gather specific information regarding the directors table.

[9]:
```
directors_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      56 non-null     object
 1   director  56 non-null     object
dtypes: object(2)
memory usage: 1.0+ KB
```

The directors table contains 56 rows and 2 columns. We notice that the column 'name' contains the title of the movie directed. We can merge this with our other dataset.

[10]:
```
# Merge directors df with cleaned_movies df using 'name' and 'movie_title' as␣
 ↪the common key
merged_df = pd.merge(directors_df, movies_cleaned, left_on='name',␣
 ↪right_on='movie_title', how='right').loc[:,['movie_title', 'director',␣
 ↪'genre','gross_revenue']]

# Display the merged DataFrame
merged_df.head(10)
```

[10]:

	movie_title	director	genre	gross_revenue
0	Snow White and the Seven Dwarfs	David Hand	Musical	5228953251
1	Pinocchio	Ben Sharpsteen	Adventure	
2	Fantasia	full credits	Musical	
3	Song of the South	NaN	Adventure	
4	Cinderella	Wilfred Jackson	Drama	
5	Lady and the Tramp	Hamilton Luske	Drama	
6	101 Dalmatians	Wolfgang Reitherman	Comedy	
7	Babes in Toyland	NaN	Musical	
8	Bon Voyage!	NaN	Comedy	
9	The Jungle Book	Wolfgang Reitherman	Musical	

```
1     2188229052
2     2187090808
3     1078510579
4      920608730
5     1236035515
6     1362870985
7      124841160
8      109581646
9      789612346
```

Looks like our data is still not tidy, lets remove the NaN values and the entry "full credits" from the director column

[11]:
```
# Remove NaN values and 'full credits' from the 'director' column
tidy_merged_df = merged_df[(merged_df['director'].notna()) &␣
↪(merged_df['director'] != 'full credits')]

# Display the cleaned DataFrame
tidy_merged_df.head(10)
```

[11]:
```
                        movie_title             director      genre  \
0   Snow White and the Seven Dwarfs           David Hand    Musical
1                         Pinocchio       Ben Sharpsteen  Adventure
4                         Cinderella      Wilfred Jackson      Drama
5                 Lady and the Tramp       Hamilton Luske      Drama
6                     101 Dalmatians  Wolfgang Reitherman     Comedy
9                   The Jungle Book  Wolfgang Reitherman    Musical
10                  The Aristocats  Wolfgang Reitherman    Musical
24                Oliver & Company      George Scribner  Adventure
33               The Little Mermaid         Ron Clements  Adventure
46          The Rescuers Down Under         Mike Gabriel  Adventure

    gross_revenue
0      5228953251
1      2188229052
4       920608730
5      1236035515
6      1362870985
9       789612346
10      255161499
24      102254492
33      223726012
46       55796728
```

[12]:
```
tidy_merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41 entries, 0 to 511
```

```
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   movie_title    41 non-null     object
 1   director       41 non-null     object
 2   genre          41 non-null     object
 3   gross_revenue  41 non-null     int64
dtypes: int64(1), object(3)
memory usage: 1.6+ KB
```

Now that we have clean data lets graph it using Altair

[13]:
```
# Count the occurrences of each director-genre pair
count_df = tidy_merged_df.groupby(['director', 'genre']).size().
 ↪reset_index(name='count')

# Plotting with Altair
tidy_merged_chart = alt.Chart(count_df, width=800, height=400,).mark_circle().
 ↪encode(
                x='director:N',
                y='genre:N',
             size='count:Q',
            color='genre:N',
).properties(title='Comparison of Directors to Genres')

tidy_merged_chart
```

[13]:
```
alt.Chart(…)
```

This show us that Some directors made multiple genre movies, but overall the majority of directors stuck to a single type of genre movie. I aslo noticed the majority of genres for movies was 'Adventure' which corresponds with the previous discovery of most gross_revenue being from this genre.

Now lets investigate the revenue generated by director

[14]:
```
# Plotting with Altair
chart = alt.Chart(tidy_merged_df, width=800,height=400,).mark_bar().encode(
    x=alt.X('director:N', sort='y'),
    y='gross_revenue:Q',
).properties(title='Comparison of Directors to Movies')


chart
```

[14]:
```
alt.Chart(…)
```

I noticed David Hand significantly surpassed all other directors in revenue generation for the movie(s) he directed. This could be due to the fact we are using inflation corrected gross_revenue or that the movie directed by David hand was the most sucessful movie of all time for Disney.

Lets investigate the revenue data set

[15]:
```
# Print the DataFrame info
revenue_df.info()
revenue_df.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 7 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Year                              26 non-null     int64
 1   Studio Entertainment[NI 1]        25 non-null     float64
 2   Disney Consumer Products[NI 2]    24 non-null     float64
 3   Disney Interactive[NI 3][Rev 1]   12 non-null     float64
 4   Walt Disney Parks and Resorts     26 non-null     float64
 5   Disney Media Networks             23 non-null     object
 6   Total                             26 non-null     int64
dtypes: float64(4), int64(2), object(1)
memory usage: 1.5+ KB
```

[15]:

	Year	Studio Entertainment[NI 1]	Disney Consumer Products[NI 2]	Disney Interactive[NI 3][Rev 1]	Walt Disney Parks and Resorts	Disney Media Networks	Total
0	1991	2593.0	724.0	NaN	2794.0	NaN	6111
1	1992	3115.0	1081.0	NaN	3306.0	NaN	7502
2	1993	3673.4	1415.1	NaN	3440.7		
3	1994	4793.0	1798.2	NaN	3463.6		
4	1995	6001.5	2150.0	NaN	3959.8		
5	1996	NaN	NaN	NaN	4502.0		
6	1997	6981.0	3782.0	174.0	5014.0		
7	1998	6849.0	3193.0	260.0	5532.0		
8	1999	6548.0	3030.0	206.0	6106.0		
9	2000	5994.0	2602.0	368.0	6803.0		

```
2                    NaN   8529
3                    359  10414
4                    414  12525
5                  4,142  18739
6                   6522  22473
7                   7142  22976
8                   7512  23402
9                   9615  25402
```

The revenue table contains 26 rows and 7 columns.

Lets cleanup this data by making the column names consistent

[16]:
```
# Remove numbers and symbols from column names
revenue_df_cleaned = revenue_df.copy()
revenue_df_cleaned.columns = revenue_df_cleaned.columns.str.replace(r'\[NI␣
↪1\]', '', regex=True)\
                                                        .str.replace(r'\[NI␣
↪2\]', '', regex=True)\
                                                        .str.replace(r'\[NI␣
↪3\]', '', regex=True)\
                                                        .str.replace(r'\[Rev␣
↪1\]','', regex=True)
```

This data is still not tidy. Lets fill the Nan values with zero and remove the commas from the Disney Media Networks column. We also need to convert the Disney Media Networks column to a type float from an object.

[17]:
```
# Lets make the column headers consistent namestyles and datatypes
revenue_df_cleaned['Disney Media Networks'] = revenue_df_cleaned['Disney Media␣
↪Networks'].str.replace(',', '').astype(float)

# Lets fill the empty cells with 0's
revenue_df_cleaned = revenue_df_cleaned.fillna(0)

# View the data
revenue_df_cleaned.head()
```

[17]:
```
   Year  Studio Entertainment  Disney Consumer Products  Disney Interactive  \
0  1991                2593.0                     724.0                 0.0
1  1992                3115.0                    1081.0                 0.0
2  1993                3673.4                    1415.1                 0.0
3  1994                4793.0                    1798.2                 0.0
4  1995                6001.5                    2150.0                 0.0

   Walt Disney Parks and Resorts  Disney Media Networks  Total
0                         2794.0                    0.0   6111
1                         3306.0                    0.0   7502
```

```
2                          3440.7                    0.0   8529
3                          3463.6                  359.0  10414
4                          3959.8                  414.0  12525
```

[18]:
```
# Check the dataframe
revenue_df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 7 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Year                            26 non-null     int64
 1   Studio Entertainment            26 non-null     float64
 2   Disney Consumer Products        26 non-null     float64
 3   Disney Interactive              26 non-null     float64
 4   Walt Disney Parks and Resorts   26 non-null     float64
 5   Disney Media Networks           26 non-null     float64
 6   Total                           26 non-null     int64
dtypes: float64(5), int64(2)
memory usage: 1.5 KB
```

I think a stacked bar chart would display this relationship the best. Lets get the data ready to plot an Altair bar chart.

[19]:
```
# Reshape the data for Altair
tidy_df = revenue_df_cleaned.melt(id_vars=['Year'], var_name='Category',␣
 ↪value_name='Value')

# Filter out 'Total' row
tidy_df = tidy_df[tidy_df['Category'] != 'Total']

# Plotting with Altair
revenue_stream_chart = alt.Chart(tidy_df, width=800, height=400,).mark_bar().
 ↪encode(
    x=alt.X('Year:N', sort='ascending'),
    y=alt.Y('Value:Q',title= 'Revenue($)'),
    color='Category:N',
).properties(title='Revenue Contribution of Categories Over Years').
 ↪configure_legend(
    title=None,
    orient='right'
)

revenue_stream_chart
```

[19]:
```
alt.Chart(…)
```

I notice that the categories studio Entertainment and Disney consumer product have seen the least growth so lets have a closer look by plotting a line graph for these categories. The year 1996 looks to be missing data so lets remove this from our analysis.

[26]:
```
# Filter the DataFrame for 'Disney Consumer Products' and 'Studio␣
↪Entertainment', excluding 1996
filtered_df = revenue_df_cleaned[revenue_df_cleaned['Year'] != 1996][['Year',␣
↪'Disney Consumer Products', 'Studio Entertainment']]

# Reshape the data for Altair
tidy_df = filtered_df.melt(id_vars=['Year'], var_name='Category',␣
↪value_name='Value')

# Plot line with Altair
line_chart = alt.Chart(tidy_df, width=600,height=400,).mark_line().encode(
    x='Year:N',
    y=alt.Y('Value:Q',title= 'Revenue($)'),
    color='Category:N',
).properties(title='Revenue Contribution of Disney Consumer Products and Studio␣
↪Entertainment')

# Add a trendline for analysis
trendline = line_chart.transform_regression('Year', 'Value',␣
↪groupby=['Category']).mark_line(strokeDash=[5, 5])
trend_chart = (line_chart + trendline)

trend_chart
```

[26]:
```
alt.LayerChart(…)
```

The trend of both these revenue streams are positive, however they are very inconsist with some years trending down.

1.3 Discussion:

In summary to answer our original questions;

1. What genre of movie generated the most revenue over time?

I was not expecting to see such a large variance from the 'Musical' genre to the others. As there was only one really sucessful movie, grouping the movies created per genre showed me that adventure genre movies are more often made thus netting the most revenue overall for Disney than any other genre.

2. What genre did a movie's director have the most sucess?

As most of the directors stuck to a single type of genre for the movies they made and since the majority of genres for movies were 'Adventure' the most gross_revenue was derived from being from this genre. however one very sucessful director, David Hand significantly surpassed all other

directors in revenue generation for the movie(s) in the 'Musical' genre. We can see that David hand was the most sucessful director of all time for Disney.

3. How did the yearly revenue generated perform by revenue source

We can see that overall the revenue generated by Disney as trended upwards over the timefrom 1991 - 2016. There are a few revenue streams I identified having the least amount of growth, Studio Entertainment and Disney Consumer Products. The trend of both these revenue streams are positive, however they are very inconsist with some years trending down. Identifying the reasons behind this is beyond the scope of the datasets studied here. I also noticed that in the year 2016 the Disney Consumet Product went to zero. This could be bad data, missing data or Disney decided to drop this revenue stream.

These findings can be used to influence genres of movies to produce that are more profitable than others as well as which revenue stream.

Other questions this data could look at are: - What move titles were the most sucessful? - What impact does inflation have on the gross revenue? - Does the gross_revenue of movies made from 1991 - 2016 influence revenue stream sucess.

1.4 References

- project layout influenced by: the project_sample
- questions influences by https://studentwork.prattsi.org/infovis/visualization/disney-shows-and-movies-data-visualization/

[21]:
```
!black final_proj_clean_func.py
```

```
reformatted final_proj_clean_func.py
All done!
1 file reformatted.
```

[22]:
```
!flake8 final_proj_clean_func.py
```

```
final_proj_clean_func.py:37:80: E501 line too long (83 > 79 characters)
final_proj_clean_func.py:49:80: E501 line too long (84 > 79 characters)
final_proj_clean_func.py:58:24: W605 invalid escape sequence '\$'
```

[23]:
```
!black test_final_proj.py
```

```
reformatted test_final_proj.py
All done!
1 file reformatted.
```

[24]:
```
!flake8 test_final_proj.py
```

[25]:
```
!black final_project.ipynb
```

```
reformatted final_project.ipynb
All done!
1 file reformatted.
```