final_project

November 7, 2023

1 Programming in Python for Data Science

2 Final Project

##Submitted by : Aditi Jaswal

3 Foreword

##This notebook will be showing some exploratory data analysis for the Disney dataset located https://data.world/kgarrett/disney-character-success-00-16. We will explore about the data and then find important questions. Later in the notebook, we will try to answer those questions using descriptive analysis

4 About the dataset

##There are 5 datasets related to disney, i.e, movies, characters, revenue, directors and voice-actors. My main analysis is for disney movie dataset. Walt Disney Studios serves as the cornerstone upon which The Walt Disney Company was established. Over the years, this division has been responsible for the creation of a vast catalog of over 600 films, with its inaugural release, "Snow White and the Seven Dwarfs," dating back to 1937. While a significant number of these cinematic endeavors achieved great success, a few encountered less favorable outcomes. This notebook embarks on an exploration of a dataset encompassing Disney movies, aiming to delve into the factors that underlie the success of these cinematic productions.website

I am going to use 3 datasets, i.e, * **disney_movies_total_gross.csv** * **disney-director.csv** * **disney-voice-actor.csv**

In **disney_movies_total_gross.csv** , there are 6 columns, i.e, movie_title(title of the movie), release_date(date of release for the movie), genre(drama, adventure, musical, comedy, action) MPAA_rating(PG, PG-13, R, G) total_gross(total box office collection), inflation_adjusted_gross(Actual profit made by the movie)

In **disney-director.csv**, 2 columns,i.e Name(Name of the movie) and director(director of the movie) are there.

In **disney-voice-actor.csv**, 3 columns, character(character name whose voice is given), voice actor(name of the voice actor) and movie(Name of the movie)are there.

5 Questions of Interest

In this notebook, I will be finding top 10 movies which did best business on box office, best movie from each genre. Also, I will be exploring if MPAA_rating impacted the popularity of movies or not.

Also, I am going to check the popularity trends for all 5 genre in disney movies.

For, director and voice datasets, I am going to merge them in one dataset and explore which director used maximum number of voice artists, and the movie which used maximum number of voice artists

6 Import libraries needed for this lab

import pandas as pd import random import test_assignment6 as t import altair as alt import string import inspect from hashlib import sha1

[1]:
```
# Import libraries needed for this lab
import pandas as pd
import random
import altair as alt
import string
import inspect
from hashlib import sha1
import matplotlib.pyplot as plt
```

[2]:
```
movies = pd.read_csv("data/disney_movies_total_gross.csv")
```

[3]:
```
movies.head()
```

[3]:
```
                       movie_title  release_date      genre MPAA_rating  \
0  Snow White and the Seven Dwarfs  Dec 21, 1937    Musical           G
1                        Pinocchio   Feb 9, 1940  Adventure           G
2                         Fantasia  Nov 13, 1940    Musical           G
3                Song of the South  Nov 12, 1946  Adventure           G
4                       Cinderella  Feb 15, 1950      Drama           G

    total_gross inflation_adjusted_gross
0  $184,925,485           $5,228,953,251
1   $84,300,000           $2,188,229,052
2   $83,320,000           $2,187,090,808
3   $65,000,000           $1,078,510,579
4   $85,000,000             $920,608,730
```

[4]:
```
movies.describe()
```

```
[4]:            movie_title  release_date   genre MPAA_rating total_gross  \
count                   579           579     562         523         579
unique                  573           553      12           5         576
top          The Jungle Book  Dec 25, 1997  Comedy          PG          $0
freq                      3             3     182         187           4

       inflation_adjusted_gross
count                       579
unique                      576
top                          $0
freq                          4
```

```
[5]: movies.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
```

6.1 Here we see that datatype of all the column is object. Lets convert them to correct datatype so we can perform analysis

```
[6]: # Remove dollar signs and commas from the "inflation_adjusted_gross" column
movies['inflation_adjusted_gross'] = movies['inflation_adjusted_gross'].str.
 ↪replace('$', '').str.replace(',', '')
```

```
<ipython-input-6-8c8daa339bdb>:2: FutureWarning: The default value of regex will
change from True to False in a future version. In addition, single character
regular expressions will*not* be treated as literal strings when regex=True.
  movies['inflation_adjusted_gross'] =
movies['inflation_adjusted_gross'].str.replace('$', '').str.replace(',', '')
```

```
[7]: #Convert the "inflation_adjusted_gross" column to integer
movies['inflation_adjusted_gross'] = movies['inflation_adjusted_gross'].
 ↪astype(int)
```

```
[8]: movies['total_gross'] = movies['total_gross'].str.replace('$', '').str.
 ↪replace(',', '')
```

```
<ipython-input-8-46c0396df8fd>:1: FutureWarning: The default value of regex will
change from True to False in a future version. In addition, single character
regular expressions will*not* be treated as literal strings when regex=True.
  movies['total_gross'] = movies['total_gross'].str.replace('$',
'').str.replace(',', '')
```

[9]:
```
#Convert the "total_gross" column to integer
movies['total_gross'] = movies['total_gross'].astype(int)
```

[10]:
```
movies['release_date'] = pd.to_datetime(movies['release_date'])
```

6.2 lets find out top 10 movies which did the best business in the market:

[11]:
```
top_movies = movies.sort_values('inflation_adjusted_gross', ascending=False)

# Display the top 10 movies
top_movies.head(10)
```

[11]:
```
                              movie_title release_date      genre MPAA_rating  \
0         Snow White and the Seven Dwarfs   1937-12-21    Musical           G
1                               Pinocchio   1940-02-09  Adventure           G
2                                Fantasia   1940-11-13    Musical           G
8                          101 Dalmatians   1961-01-25     Comedy           G
6                      Lady and the Tramp   1955-06-22      Drama           G
3                      Song of the South   1946-11-12  Adventure           G
564  Star Wars Ep. VII: The Force Awakens   2015-12-18  Adventure       PG-13
4                              Cinderella   1950-02-15      Drama           G
13                        The Jungle Book   1967-10-18    Musical   Not Rated
179                         The Lion King   1994-06-15  Adventure           G

     total_gross  inflation_adjusted_gross
0      184925485                5228953251
1       84300000                2188229052
2       83320000                2187090808
8      153000000                1362870985
6       93600000                1236035515
3       65000000                1078510579
564    936662225                 936662225
4       85000000                 920608730
13     141843000                 789612346
179    422780140                 761640898
```

6.3 now lets see the best movie from each genre

[12]:
```
genre_wise_best= movies.groupby("genre")['movie_title',␣
↪'inflation_adjusted_gross'].max()
genre_wise_best
```

```
<ipython-input-12-0a5cbac29cb1>:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  genre_wise_best= movies.groupby("genre")['movie_title',
'inflation_adjusted_gross'].max()
```

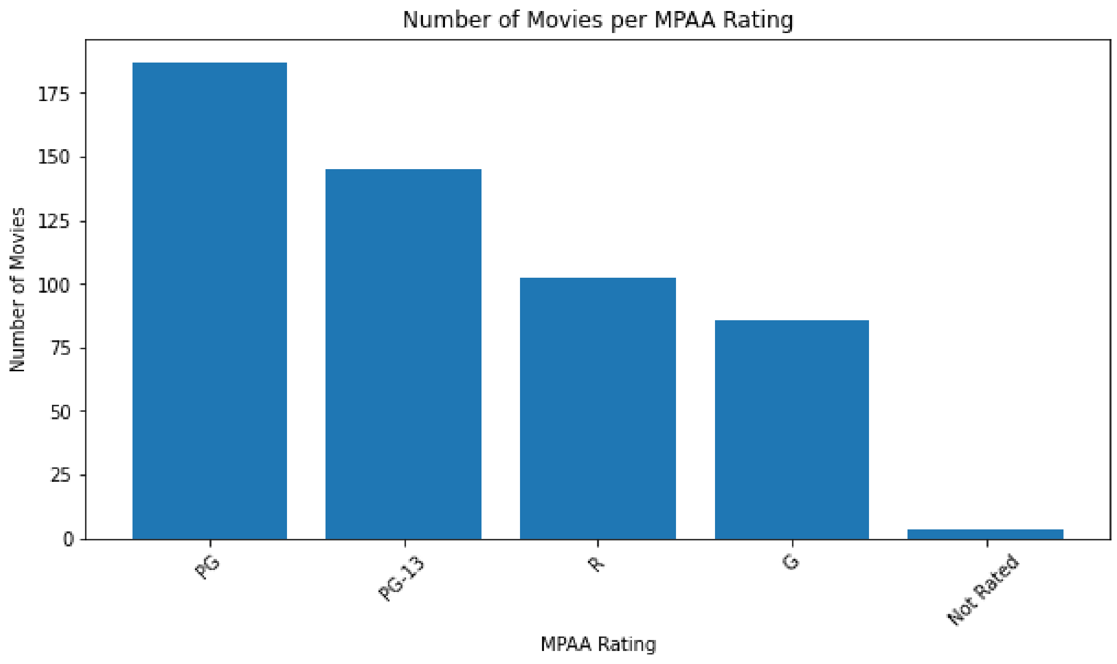
[12]:
```
                                                movie_title  \
genre
Action                                                 Tron
Adventure                                          Zootopia
Black Comedy                           The Royal Tenenbaums
Comedy                                            You Again
Concert/Performance  Jonas Brothers: The 3D Concert Experi…
Documentary                          X Games 3D: The Movie
Drama                                      crazy/beautiful
Horror                                    The Puppet Masters
Musical              Tim Burton's The Nightmare Before Chr…
Romantic Comedy                      While You Were Sleeping
Thriller/Suspense                               Unbreakable
Western                                           Tombstone

                     inflation_adjusted_gross
genre
Action                              660081224
Adventure                          2188229052
Black Comedy                         76758193
Comedy                             1362870985
Concert/Performance                  76646993
Documentary                          35981010
Drama                              1236035515
Horror                               48546161
Musical                            5228953251
Romantic Comedy                     356389765
Thriller/Suspense                   485424724
Western                             115781734
```

##lets explore MPAA_rating column. From describe(), we can see that there are 5 unique entries for this column, i.e, PG, G, PG-13, R and Not rated. From https://en.wikipedia.org/wiki/Motion_Picture_Association_film_rating_system , PG = Parentl Guidance, G - General Audience, PG-13 = Parents Strongly Cautioned and R = Restricted.

```
[13]: rating_counts = movies['MPAA_rating'].value_counts()
      rating_counts
```

```
[13]: PG           187
      PG-13        145
      R            102
      G             86
      Not Rated      3
      Name: MPAA_rating, dtype: int64
```

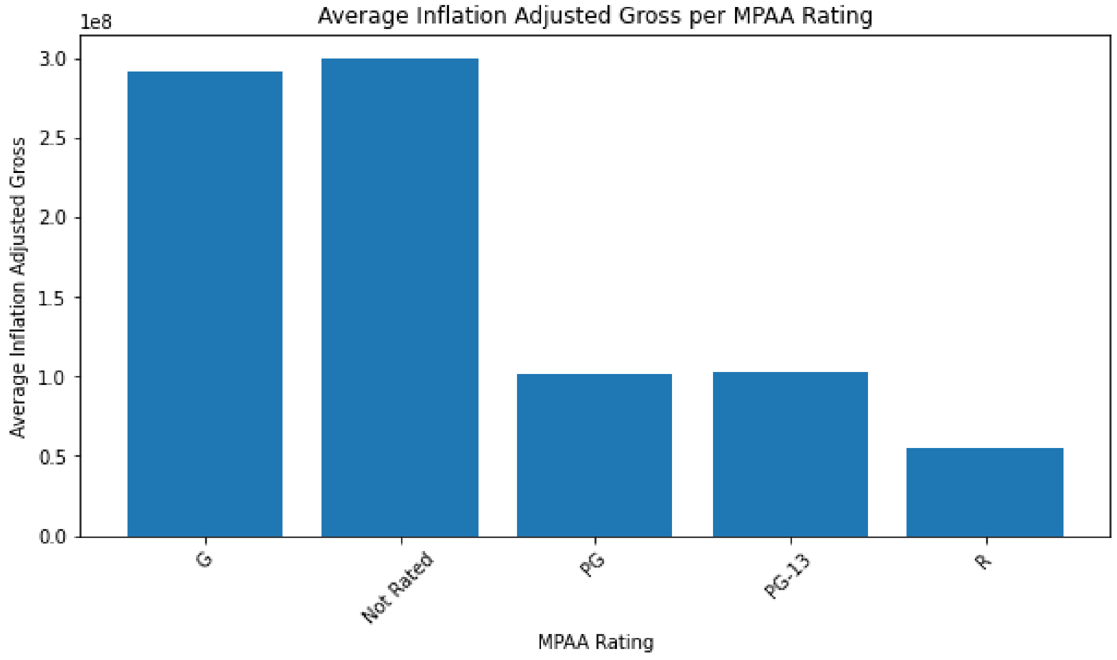
```
[14]: plt.figure(figsize=(10, 5))
      plt.bar(rating_counts.index, rating_counts)
      plt.title('Number of Movies per MPAA Rating')
      plt.xlabel('MPAA Rating')
      plt.ylabel('Number of Movies')
      plt.xticks(rotation=45)
      plt.show()
```



6.4 Lets find out if there is any trend between the ,MP rating and business done by disney movies:

```
[15]: average_gross = movies.groupby('MPAA_rating')['inflation_adjusted_gross'].mean()
```

```
[16]: plt.figure(figsize=(10, 5))
      plt.bar(average_gross.index, average_gross)
      plt.title('Average Inflation Adjusted Gross per MPAA Rating')
      plt.xlabel('MPAA Rating')
      plt.ylabel('Average Inflation Adjusted Gross')
      plt.xticks(rotation=45)
      plt.show()
```



##here we found out that although least number of movies were 'not rated' but those movies did maximum of the business.However, movies made for general audiences also did good business.

6.5 From the above stats, it seems that some genre are more polpular than others. lets find out movie genre trend

[17]:
```
## finding release year from release date column
movies['release_year'] = movies['release_date'].dt.year
```

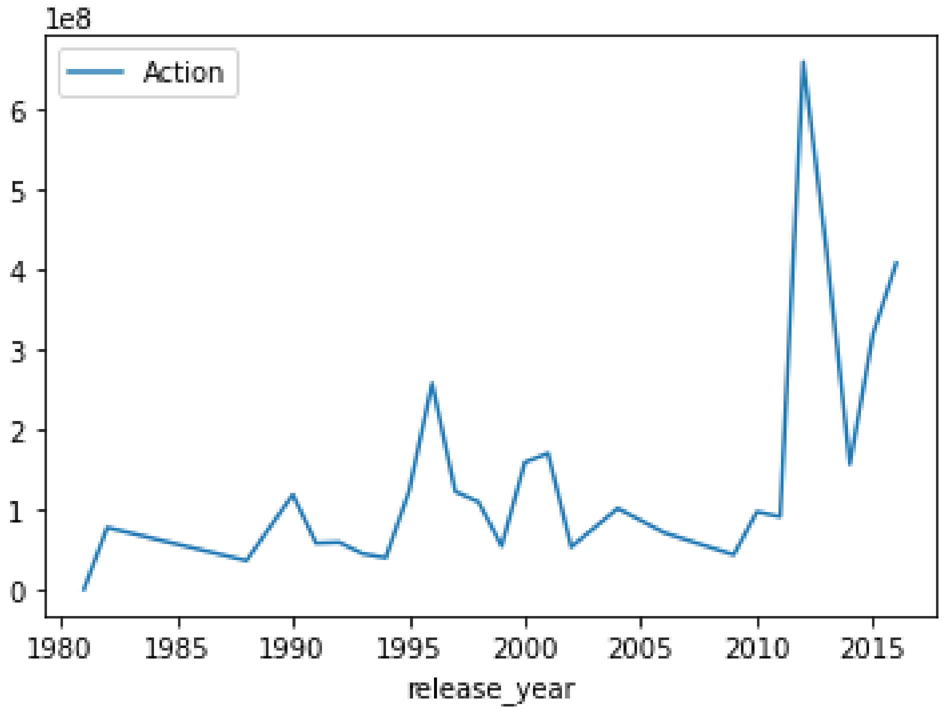
[18]:
```
# Group the movies data by 'genre' and 'release_year', and compute the mean␣
↪'adjusted_gross' for each group
grouped_movies = movies.groupby(['genre',␣
↪'release_year'])['inflation_adjusted_gross'].mean().reset_index()

# Display the first 10 rows of the grouped_movies DataFrame
print(grouped_movies.head(10))
```

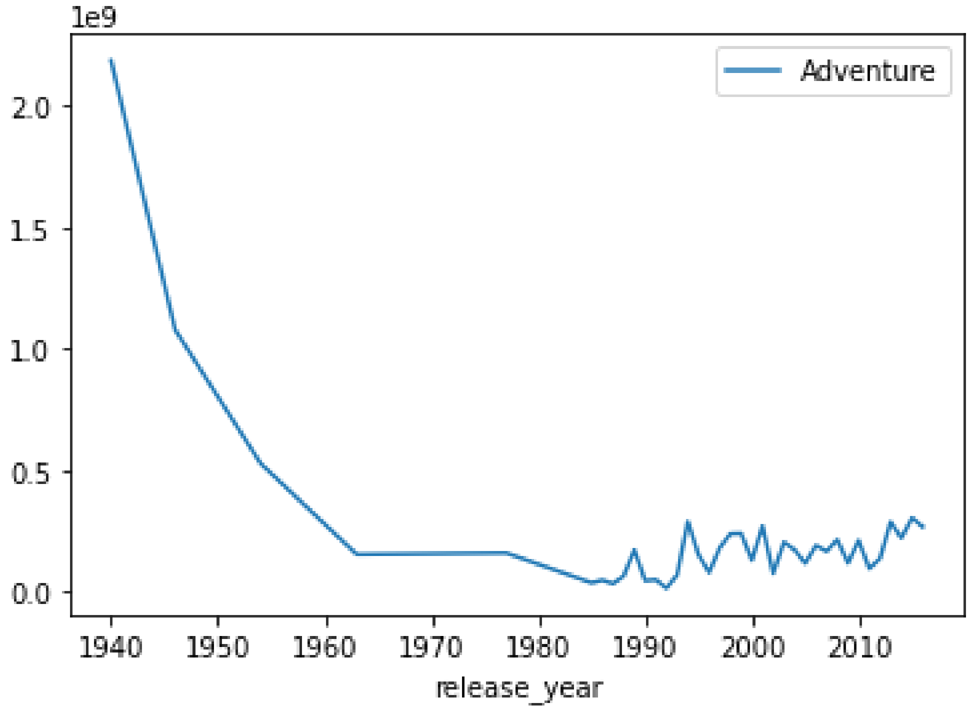
```
    genre  release_year  inflation_adjusted_gross
0  Action          1981                       0.0
1  Action          1982                77184895.0
```

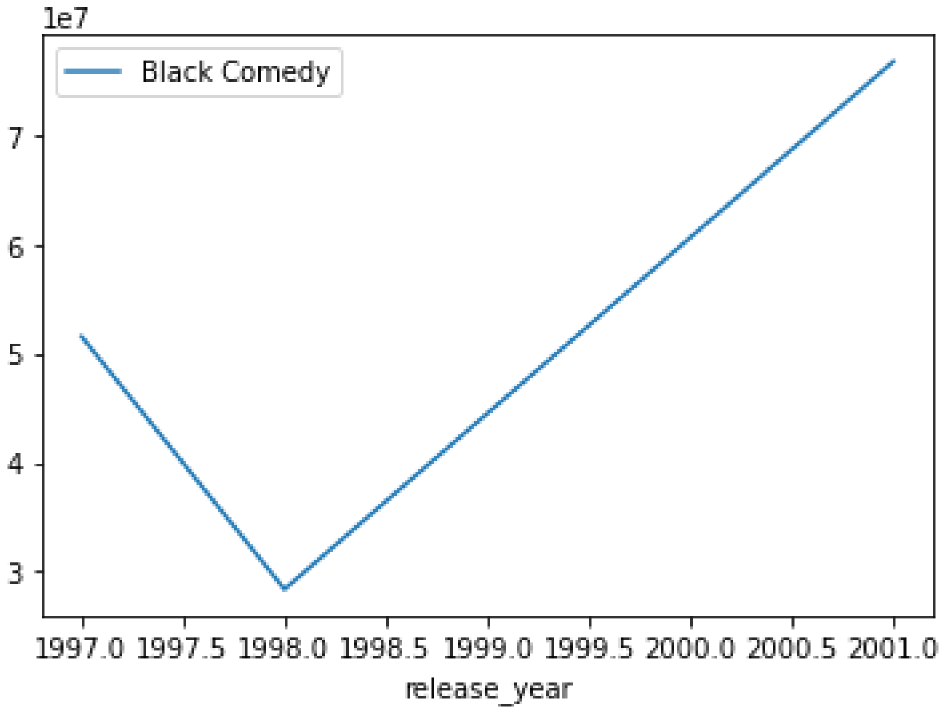
```
2   Action          1988                 36053517.0
3   Action          1990                118358772.0
4   Action          1991                 57918572.5
5   Action          1992                 58965304.0
6   Action          1993                 44682157.0
7   Action          1994                 39545796.0
8   Action          1995                122162426.5
9   Action          1996                257755262.5
```

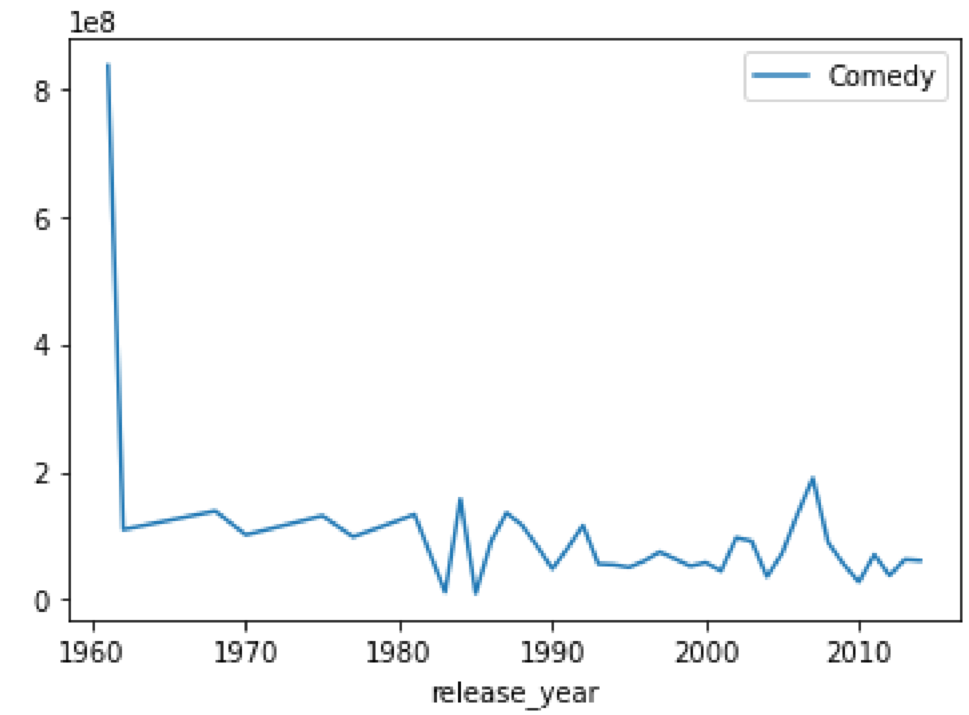
[19]:
```python
# Create a plot for the genre with the highest mean adjusted gross in each year
plt.figure(figsize=(12, 6))
for genre, df in grouped_movies.groupby('genre'):
    df.plot(x='release_year', y='inflation_adjusted_gross', label=genre)
plt.title('Box Office Revenues of Movies Grouped by Genre and Release Year')
plt.xlabel('Release Year')
plt.ylabel('Adjusted Gross (in millions)')
plt.legend()
plt.show()
```

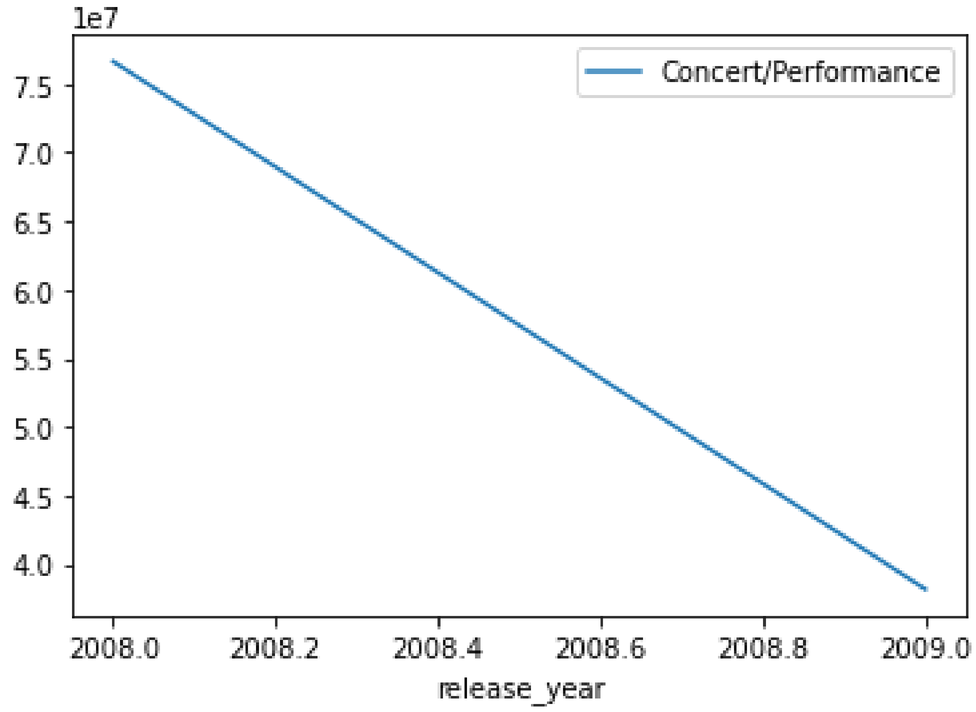
```
<Figure size 864x432 with 0 Axes>
```

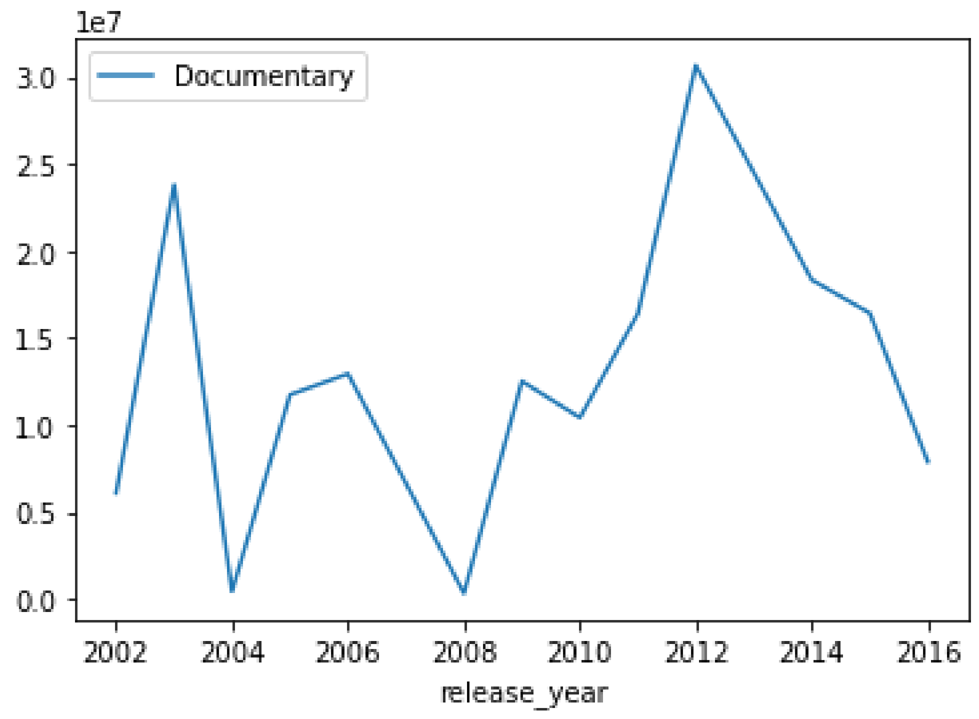


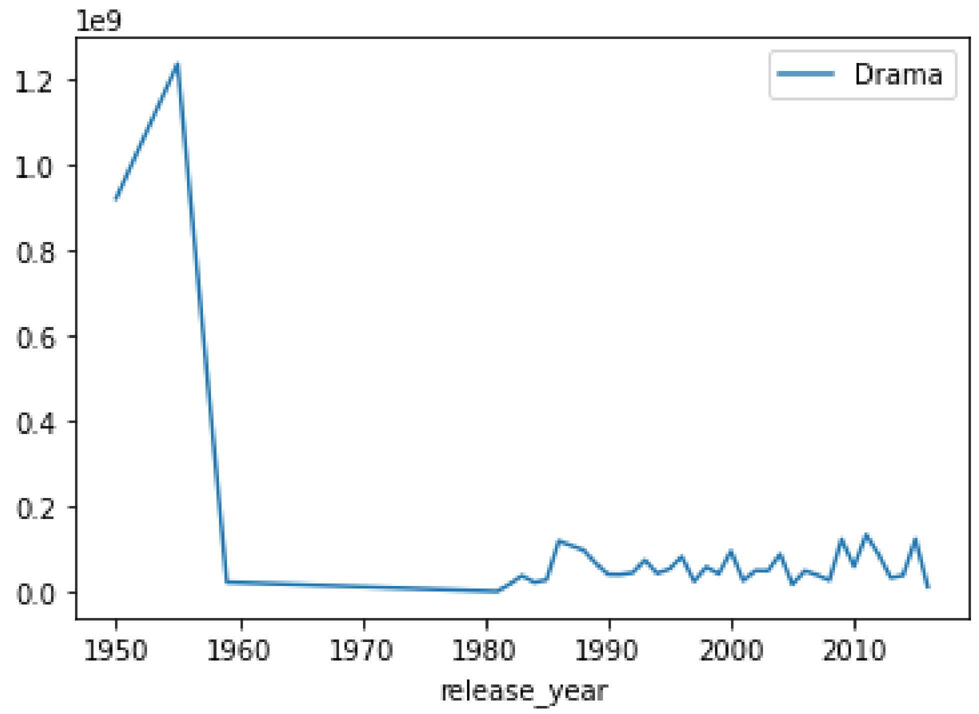

1e9
Adventure
2.0
1.5
1.0
0.5
0.0
1940
1950
1960
1970
1980
1990
2000
2010
release_year

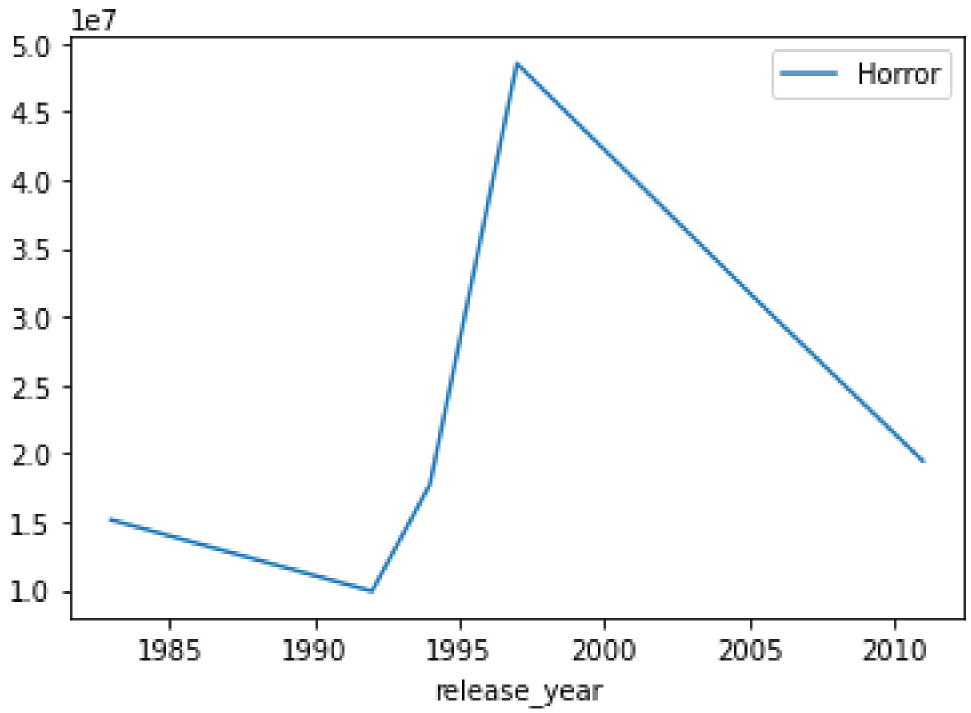

1e7
Black Comedy
7
6
5
4
3
1997.0
1997.5
1998.0
1998.5
1999.0
1999.5
2000.0
2000.5
2001.0
release_year

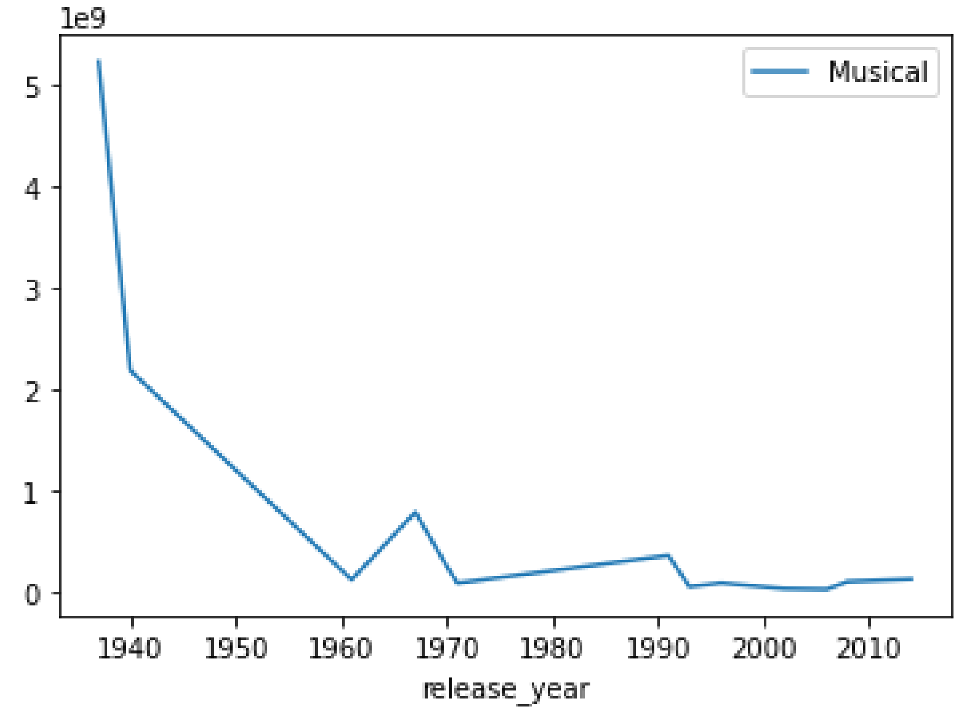

1e8
8
6
4
2
0
1960
1970
1980
1990
2000
2010
release_year
Comedy

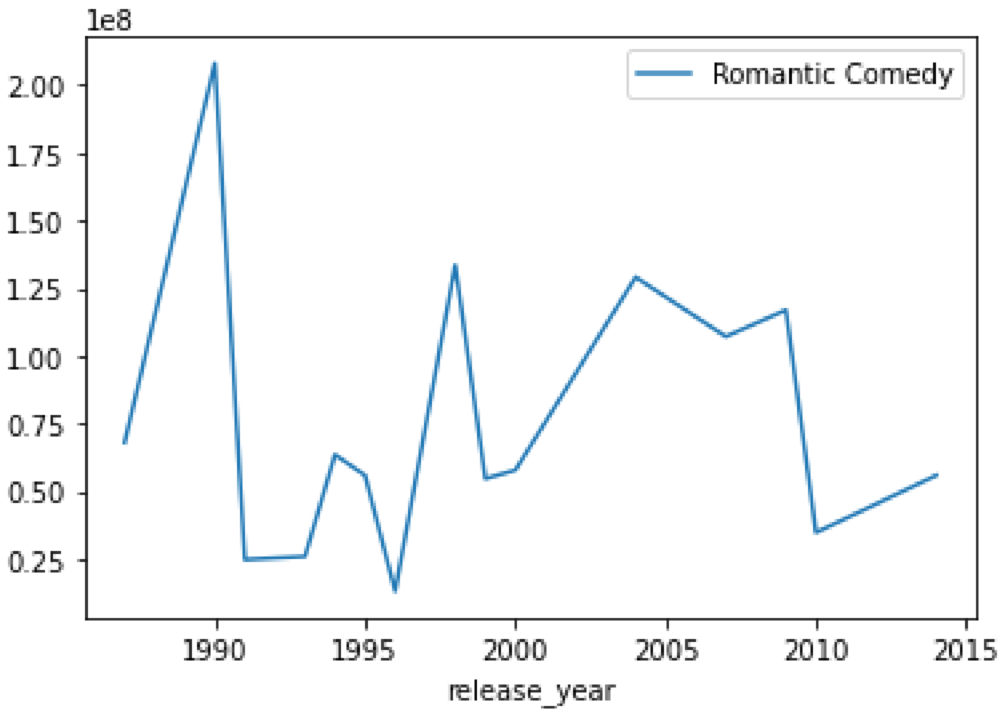

1e7
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
2008.0
2008.2
2008.4
2008.6
2008.8
2009.0
release_year
Concert/Performance

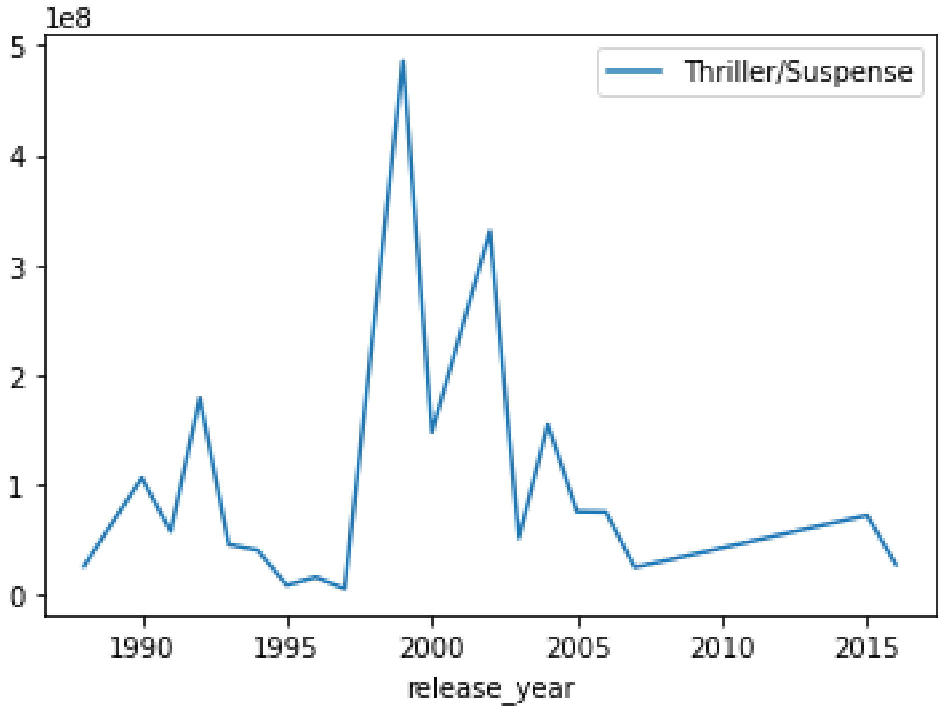

1e7
Documentary
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2002
2004
2006
2008
2010
2012
2014
2016
release_year

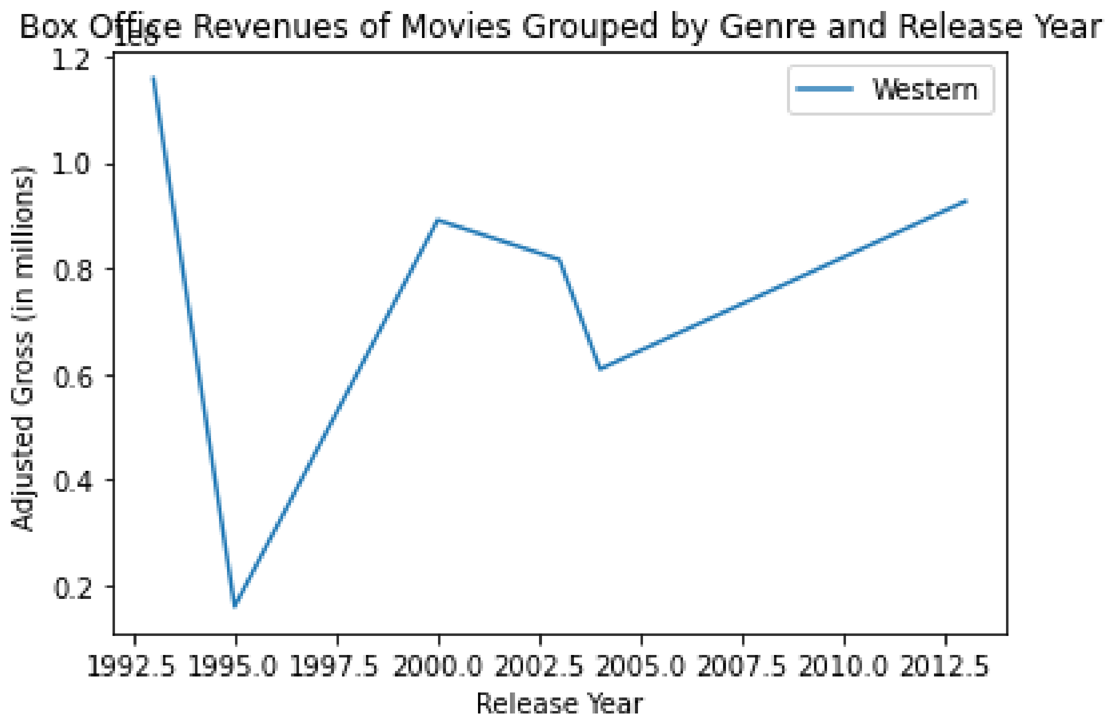

1e9
Drama
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1950
1960
1970
1980
1990
2000
2010
release_year


1e7
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Horror
1985
1990
1995
2000
2005
2010
release_year

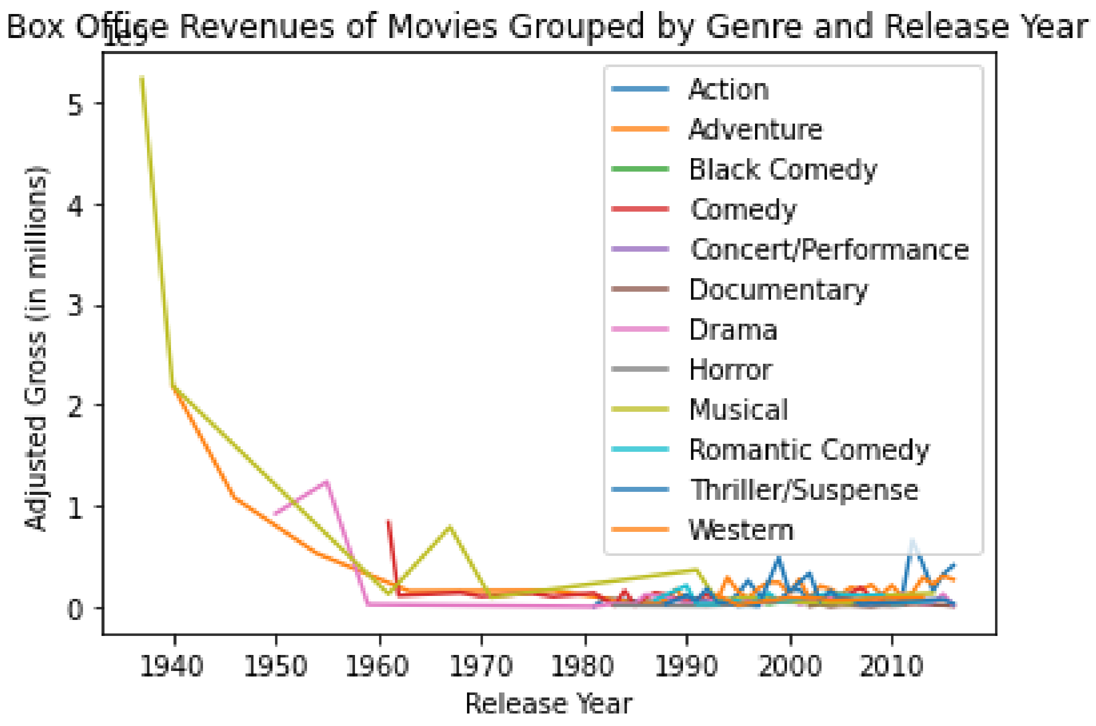

1e9
5
4
3
2
1
0
Musical
1940
1950
1960
1970
1980
1990
2000
2010
release_year


1e8
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
1990
1995
2000
2005
2010
2015
release_year
Romantic Comedy


1e8
5
4
3
2
1
0
1990
1995
2000
2005
2010
2015
release_year
Thriller/Suspense



[20]:
```
grouped_data = grouped_movies.groupby('genre')

# Plot the data
for genre, genre_data in grouped_data:
    plt.plot(genre_data['release_year'],␣
 ↪genre_data['inflation_adjusted_gross'], label=genre)

# Set up the plot
plt.title('Box Office Revenues of Movies Grouped by Genre and Release Year')
plt.xlabel('Release Year')
plt.ylabel('Adjusted Gross (in millions)')
plt.legend()
plt.show()
```



##From the above line plot, we can see that action genre was growing the fastest. However musical dropped significantly. Thus, disney's movie genre were changing in terms of popularity over the period of time

6.6 lets explore other 2 datasets now

[21]:
```
director = pd.read_csv("data/disney-director.csv")
```

[22]:
```
director
```

[22]:
```
                                      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
10  The Adventures of Ichabod and Mr. Toad        Jack Kinney
11                              Cinderella    Wilfred Jackson
12                       Alice in Wonderland     Clyde Geronimi
```

```
13                              Peter Pan       Hamilton Luske
14                     Lady and the Tramp       Hamilton Luske
15                        Sleeping Beauty       Clyde Geronimi
16                         101 Dalmatians  Wolfgang Reitherman
17                  The Sword in the Stone  Wolfgang Reitherman
18                         The Jungle Book  Wolfgang Reitherman
19                         The Aristocats  Wolfgang Reitherman
20                              Robin Hood  Wolfgang Reitherman
21  The Many Adventures of Winnie the Pooh  Wolfgang Reitherman
22                            The Rescuers  Wolfgang Reitherman
23                   The Fox and the Hound          Art Stevens
24                      The Black Cauldron           Ted Berman
25               The Great Mouse Detective         Ron Clements
26                        Oliver & Company      George Scribner
27                       The Little Mermaid         Ron Clements
28                  The Rescuers Down Under         Mike Gabriel
29                    Beauty and the Beast       Gary Trousdale
30                                 Aladdin         Ron Clements
31                           The Lion King         Roger Allers
32                              Pocahontas         Mike Gabriel
33             The Hunchback of Notre Dame       Gary Trousdale
34                                Hercules         Ron Clements
35                                   Mulan           Barry Cook
36                                  Tarzan           Chris Buck
37                           Fantasia 2000         full credits
38                                Dinosaur         Ralph Zondag
39                 The Emperor's New Groove          Mark Dindal
40               Atlantis: The Lost Empire       Gary Trousdale
41                            Lilo & Stitch       Chris Sanders
42                         Treasure Planet         Ron Clements
43                             Brother Bear       Robert Walker
44                        Home on the Range            Will Finn
45                          Chicken Little          Mark Dindal
46                      Meet the Robinsons  Stephen J. Anderson
47                                    Bolt       Chris Williams
48               The Princess and the Frog         Ron Clements
49                                 Tangled        Nathan Greno
50                        Winnie the Pooh  Stephen J. Anderson
51                          Wreck-It Ralph           Rich Moore
52                                  Frozen           Chris Buck
53                              Big Hero 6             Don Hall
54                                Zootopia         Byron Howard
55                                   Moana         Ron Clements
```

[23]:
```
voice = pd.read_csv("data/disney-voice-actors.csv")
```

[24]:
```
voice
```

[24]:
```
             character        voice-actor                            movie
0         Abby Mallard        Joan Cusack                   Chicken Little
1       Abigail Gabble       Monica Evans                   The Aristocats
2             Abis Mal    Jason Alexander              The Return of Jafar
3                  Abu       Frank Welker                          Aladdin
4             Achilles               None      The Hunchback of Notre Dame
..                 …                  …                                …
930               Zeus           Rip Torn                         Hercules
931  Ziggy the Vulture        Digby Wolfe                  The Jungle Book
932               Zini        Max Casella                         Dinosaur
933             Zipper       Corey Burton      Chip 'n Dale Rescue Rangers
934               Zira  Suzanne Pleshette  The Lion King II: Simba's Pride

[935 rows x 3 columns]
```

6.7 lets create a function named "merge_dataframes_inner" to merge 2 datsets and merge datasets director and voice in one dataset named voice_director

[25]:
```python
def merge_dataframes_inner(df1, df2, common_column):
    """
    Merge two DataFrames using an inner join on a common column.

    Parameters
    ----------
    df1: DataFrame
      The first DataFrame to be merged.
    df2: DataFrame
      The second DataFrame to be merged.
    common_column: str
      The name of the common column to perform the inner join.

    Returns
    ----------
    merged_df: DataFrame
      The resulting DataFrame after merging the two input DataFrames using an␣
 ↪inner join on the common column.

    Examples
    ----------

    df1 = pd.DataFrame({
    'ID': [1, 2, 3, 4],
    'Name': ['Alice', 'Bob', 'Charlie', 'David']
})

    df2 = pd.DataFrame({
    'ID': [3, 4, 5, 6],
    'Age': [25, 30, 22, 28]
})
    """
    merged_df = pd.merge(df1, df2, on=common_column, how='inner')
    return merged_df
```

[26]:
```
# Rename the 'name' column in the 'director' dataframe to 'movie' for a␣
↪consistent column name
director = director.rename(columns={'name': 'movie'})

#Calling the function to merge the datasets
voice_director = merge_dataframes_inner(director, voice, 'movie')
voice_director
```

[26]:
```
                                movie      director       character  \
0    Snow White and the Seven Dwarfs    David Hand         Bashful
1    Snow White and the Seven Dwarfs    David Hand             Doc
2    Snow White and the Seven Dwarfs    David Hand           Dopey
3    Snow White and the Seven Dwarfs    David Hand          Grumpy
4    Snow White and the Seven Dwarfs    David Hand           Happy
..                                ...           ...             ...
637                             Moana  Ron Clements            Maui
638                             Moana  Ron Clements  Moana Waialiki
639                             Moana  Ron Clements             Pua
640                             Moana  Ron Clements         Tamatoa
641                             Moana  Ron Clements    Tui Waialiki

          voice-actor
0      Scotty Mattraw
1          Roy Atwell
2        Eddie Collins
3         Pinto Colvig
4          Otis Harlan
..                ...
637    Dwayne Johnson
638   Auli'i Cravalho
639              None
640   Jemaine Clement
641  Temuera Morrison

[642 rows x 4 columns]
```

6.8 unit testing

[27]:
```
# Helper DataFrames for unit tests
df1 = pd.DataFrame({
    'ID': [1, 2, 3, 4],
    'Name': ['Alice', 'Bob', 'Charlie', 'David']
})

df2 = pd.DataFrame({
    'ID': [3, 4, 5, 6],
    'Age': [25, 30, 22, 28]
})

# Unit test 1: Test merging on 'ID'
merged_result = merge_dataframes_inner(df1, df2, 'ID')
expected_result = pd.DataFrame({
    'ID': [3, 4],
    'Name': ['Charlie', 'David'],
    'Age': [22, 28]
})
merged_result.shape == expected_result.shape
```

[27]:
```
True
```

[28]:
```
voice_director.describe()
```

[28]:
```
           movie      director    character voice-actor
count        642           642          642         642
unique        54            29          635         499
top     Zootopia  Ron Clements  Arthur/Wart        None
freq          22           106            3          37
```

[29]:
```
voice_director.isnull()
```

[29]:
```
     movie  director  character  voice-actor
0    False     False      False        False
1    False     False      False        False
2    False     False      False        False
3    False     False      False        False
4    False     False      False        False
..     …         …          …           …
637  False     False      False        False
638  False     False      False        False
639  False     False      False        False
640  False     False      False        False
641  False     False      False        False
```

```
[642 rows x 4 columns]
```

[30]:
```
voice_director.head()
```

[30]:
```
                           movie    director character     voice-actor
0  Snow White and the Seven Dwarfs  David Hand   Bashful  Scotty Mattraw
1  Snow White and the Seven Dwarfs  David Hand       Doc      Roy Atwell
2  Snow White and the Seven Dwarfs  David Hand     Dopey   Eddie Collins
3  Snow White and the Seven Dwarfs  David Hand    Grumpy    Pinto Colvig
4  Snow White and the Seven Dwarfs  David Hand     Happy     Otis Harlan
```

6.9 Lets find out which director wortked with maximum number of voice-actors in their movie

[31]:
```
## using chaining for group by director and then get the unique values for␣
↪voice-actors who worked with the director
director_voice_actor_count = voice_director.groupby('director')['voice-actor'].
↪nunique().reset_index(name='voice_actor_count')
director_voice_actor_count = director_voice_actor_count.
↪sort_values(by='voice_actor_count', ascending=False)

print(director_voice_actor_count)
```

```
               director  voice_actor_count
22         Ron Clements                 92
27  Wolfgang Reitherman                 71
10        Gary Trousdale                 39
8             David Hand                 24
15          Mike Gabriel                 24
7        Clyde Geronimi                 23
12        Hamilton Luske                 22
14          Mark Dindal                 21
3          Byron Howard                 20
2       Ben Sharpsteen                 19
4            Chris Buck                 18
19           Rich Moore                 17
23  Stephen J. Anderson                 16
26            Will Finn                 16
1            Barry Cook                 16
25      Wilfred Jackson                 15
11      George Scribner                 14
21         Roger Allers                 14
6        Chris Williams                 13
24           Ted Berman                 12
18         Ralph Zondag                 11
5        Chris Sanders                 11
0           Art Stevens                 10
```

```
13          Jack Kinney                   10
20        Robert Walker                    7
16         Nathan Greno                    7
28         full credits                    7
17     Norman Ferguson                     2
9              Don Hall                    1
```

6.10 Lets find out the movie which used max numbers of voice artists

[32]:
```
movie_voice_actor_count = voice.groupby('movie')['voice-actor'].nunique().
 ↪reset_index(name='voice_actor_count')
movie_voice_actor_count = movie_voice_actor_count.
 ↪sort_values(by='voice_actor_count', ascending=False)

print(movie_voice_actor_count)
```

```
                                     movie  voice_actor_count
43                                Hercules                 22
138                               Zootopia                 20
34                               DuckTales                 20
93                          The Aristocats                 20
14                    Beauty and the Beast                 19
..                                     …                 …
86                        Steamboat Willie                  1
111   The Little Mermaid: Ariel's Beginning                  1
112                       The Pirate Fairy                  1
113                           The Plow Boy                  1
69                          Pete's Dragon                  1

[139 rows x 2 columns]
```

##Who directed the movie with maximum number of voice artists in it? What was the name of the movie and what was the count of voice actor?

6.11 Ques: Find out the name of movie in which maximum number of voice-actors were used? Also find out the count of voice actors and the director who directed that movie?

[33]:
```
# Create a new column to count the number of voice actors per movie
voice_director['voice_actor_count'] = voice_director.
 ↪groupby('movie')['voice-actor'].transform('count')

# Filter the dataset to include only the rows of the movie with the maximum␣
 ↪number of voice actors
max_voice_actors_movie = voice_director[voice_director['voice_actor_count'] ==␣
 ↪voice_director['voice_actor_count'].max()]

# Print the movie details
```

```
print(max_voice_actors_movie[['movie', 'director', 'voice_actor_count']].
 ↪drop_duplicates())
```

```
        movie      director  voice_actor_count
382  Hercules  Ron Clements                 22
613  Zootopia  Byron Howard                 22
```

#lets find out top 10 movies

6.12 using black for improving function

[34]:
```
!black my_functions.py
```

```
All done!
1 file left unchanged.
```

[35]:
```
!black test_my_function.py
```

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

6.13 using pytest

[36]:
```
import pytest
```

[37]:
```
pytest
```

[37]:
```
<module 'pytest' from '/opt/conda/lib/python3.8/site-
packages/pytest/__init__.py'>
```

[38]:
```
!pytest test_my_function.py
```

```
============================= test session starts

==============================
platform linux -- Python 3.8.5, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /home/jupyter/prog-python-ds-students/release/final_project
plugins: anyio-3.2.1, dash-1.20.0
collected 2 items


test_my_function.py ..

[100%]

============================== 2 passed in 0.79s

===============================
```

6.13.1 Thank you :)

```
[ ]:
```