In [10]:
```
# Import libraries
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt

# Set plot space as inline for inline plots and qt for external plots
%matplotlib inline

import seaborn as sns
```

In [8]:
```
#Get current working directory
os.getcwd()
```

Out[8]:
```
'C:\\Users\\user\\Documents\\Phase_1'
```

In [56]:
```
#import datasets
Moviegross = pd.read_csv('C:\\Users\\user\\Documents\\Phase_1\\bom.movie_g
Moviegross
```

Out[56]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

In [31]:
```
type(Moviegross)
```

Out[31]:
```
pandas.core.frame.DataFrame
```

In [33]:
```
#First 5 rows of Moviegross
Moviegross.head(5)
```

Out[33]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010

In [34]:
```
# last 3 rows of Moviesgross
Moviegross.tail(3)
```

Out[34]:

	title	studio	domestic_gross	foreign_gross	year
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

In [35]:
```
# Summary of Moviesgross DataFrame
Moviegross.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

In [37]:
```
#Checking Moviegross duplicates
# Moviegross.puplicate(). value_counts()
Moviegross.shape
```

Out[37]:
```
(3387, 5)
```

In [48]:
```
Moviegross.isnull().sum()
```

Out[48]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [49]:
```
#Finding total sum number of null entries
Moviegross.isnull().sum().sum()
```

Out[49]:
```
1383
```

In [51]:
```
# Filling Null
Moviegross_1=Moviegross.fillna(value = 0)
Moviegross_1
```

Out[51]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	0	2018
3383	Edward II (2018 re-release)	FM	4800.0	0	2018
3384	El Pacto	Sony	2500.0	0	2018
3385	The Swan	Synergetic	2400.0	0	2018
3386	An Actor Prepares	Grav.	1700.0	0	2018

3387 rows × 5 columns

In [235]:
```
#Fist year of movies production check
Moviegross_1["year"].min()
```

Out[235]:
```
2010
```

In [236]:
```
# Checking latest production year
Moviegross_1["year"].max()
```

Out[236]:
```
2018
```

In [253]:
```
Moviegross_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   title           3387 non-null   object
 1   studio          3387 non-null   object
 2   domestic_gross  3387 non-null   int32 
 3   foreign_gross   3387 non-null   object
 4   year            3387 non-null   int64 
dtypes: int32(1), int64(1), object(3)
memory usage: 119.2+ KB
```

In [255]:
```
#change datatype of the domestic gross columns to int
Moviegross_1["domestic_gross"]=Moviegross_1["domestic_gross"].astype(np.ir
Moviegross_1["domestic_gross"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3387 entries, 0 to 3386
Series name: domestic_gross
Non-Null Count  Dtype
--------------  -----
3387 non-null   int64
dtypes: int64(1)
memory usage: 26.6 KB
```

In [256]:
```
#change datatype of the foreign gross columns to int
Moviegross_1["foreign_gross"].info
```

Out[256]:
```
<bound method Series.info of 0       652000000
1       691300000
2       664300000
3       535700000
4       513900000
          ...    
3382            0
3383            0
3384            0
3385            0
3386            0
Name: foreign_gross, Length: 3387, dtype: object>
```

In [257]:
```
# Convert foreign_gross to float then integers
Moviegross_1["foreign_gross"]=Moviegross_1["foreign_gross"].astype(str)
Moviegross_1["foreign_gross"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3387 entries, 0 to 3386
Series name: foreign_gross
Non-Null Count  Dtype
--------------  -----
3387 non-null   object
dtypes: object(1)
memory usage: 26.6+ KB
```

In [259]:
```
# Converting Moviegross_1, foreign_gross column data type to float
Moviegross_1["foreign_gross"]=Moviegross_1["foreign_gross"].str.replace(',
Moviegross_1["foreign_gross"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3387 entries, 0 to 3386
Series name: foreign_gross
Non-Null Count  Dtype
--------------  -----
3387 non-null   float64
dtypes: float64(1)
memory usage: 26.6 KB
```

In [260]:
```
# Converting Moviegross_1, foreign_gross column data type from float to ir
Moviegross_1["foreign_gross"]=Moviegross_1["foreign_gross"].astype(np.int
Moviegross_1["foreign_gross"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3387 entries, 0 to 3386
Series name: foreign_gross
Non-Null Count  Dtype
--------------  -----
3387 non-null   int64
dtypes: int64(1)
memory usage: 26.6 KB
```

In [261]:
```
#create total column adding domestic and foreign gross
Moviegross_1['studio_grosssum'] \
        = Moviegross_1['domestic_gross']\
        + Moviegross_1['foreign_gross']
```

In [262]:
```
# checking if studio_grosssum column is created
Moviegross_1.head(5)
```

Out[262]:

	title	studio	domestic_gross	foreign_gross	year	studio_grosssum
0	Toy Story 3	BV	415000000	652000000	2010	1067000000
1	Alice in Wonderland (2010)	BV	334200000	691300000	2010	1025500000
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000	664300000	2010	960300000
3	Inception	WB	292600000	535700000	2010	828300000
4	Shrek Forever After	P/DW	238700000	513900000	2010	752600000

In [272]:
```
# Count for each studio
Moviegross_1['studio'].value_counts()
```

Out[272]:
```
studio
IFC           166
Uni.          147
WB            140
Fox           136
Magn.         136
             ...
E1              1
PI              1
ELS             1
PalT            1
Synergetic      1
Name: count, Length: 258, dtype: int64
```

In [276]:
```
#set the columns as studios and value counts in a new dataframe to pull tc
prod_studios = Moviegross_1['studio'].value_counts().reset_index()
prod_studios.columns = ['studio', 'count']
prod_studios.head(5)
```

Out[276]:

	studio	count
0	IFC	166
1	Uni.	147
2	WB	140
3	Fox	136
4	Magn.	136

In [278]:
```
prod_studios = prod_studios[prod_studios['count'] >= 100]
prod_studios
```

Out[278]:

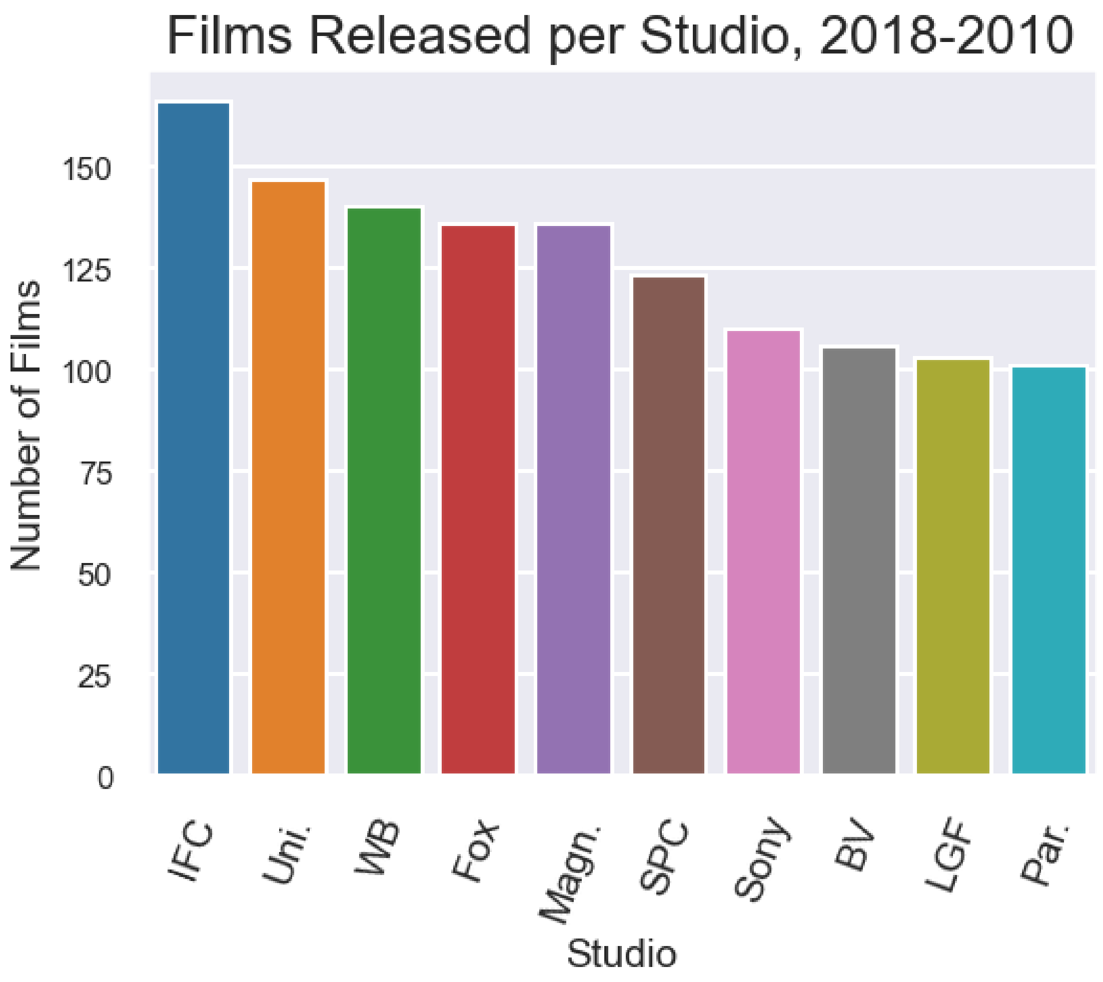
	studio	count
0	IFC	166
1	Uni.	147
2	WB	140
3	Fox	136
4	Magn.	136
5	SPC	123
6	Sony	110
7	BV	106
8	LGF	103
9	Par.	101

In [286]:

```
# Creating a bar graph using Seaborn
sns.barplot(x='studio', y='count', data=prod_studios)
plt.xlabel('Studio', fontsize = 15)
plt.ylabel('Number of Films', fontsize = 15)
plt.title('Films Released per Studio, 2018-2010', fontsize = 20)
sns.set_style('darkgrid')
sns.set_context('talk')
plt.xticks(rotation=70, fontsize=15)
plt.yticks(fontsize=12)
```

Out[286]:

```
(array([  0.,  25.,  50.,  75., 100., 125., 150., 175.]),
 [Text(0, 0.0, '0'),
  Text(0, 25.0, '25'),
  Text(0, 50.0, '50'),
  Text(0, 75.0, '75'),
  Text(0, 100.0, '100'),
  Text(0, 125.0, '125'),
  Text(0, 150.0, '150'),
  Text(0, 175.0, '175')])
```



In [305]:
```
# Creat an SQL querry to columns to plot/vizualize
sql_query = "SELECT studio, studio_grosssum FROM Moviegross_1"
Moviegross_2 = (ps.sqldf(sql_query, locals()))
print(Moviegross_2)
```

```
          studio  studio_grosssum
0             BV       1067000000
1             BV       1025500000
2             WB        960300000
3             WB        828300000
4           P/DW        752600000
...          ...              ...
3382       Magn.             6200
3383          FM             4800
3384        Sony             2500
3385  Synergetic             2400
3386       Grav.             1700

[3387 rows x 2 columns]
```

In [302]:
```
# Moviegross_2.head(10)
```

In [308]:
```
Moviegross_2.groupby('studio').mean()
```

Out[308]:

	studio_grosssum
studio	
0	2.655492e+07
3D	1.600000e+07
A23	8.210000e+04
A24	1.148278e+07
ADC	1.241000e+05
...	...
XL	2.290000e+05
YFG	1.100000e+06
Yash	2.174689e+07
Zee	1.671000e+06
Zeit.	1.622719e+06

258 rows × 1 columns

In [315]:
```
#run an pandaSQL query for studio_grosssum profit by studio
query = """SELECT avg(studio_grosssum), studio
           FROM Moviegross_2
           GROUP BY studio
           ORDER BY avg(studio_grosssum) ASC;
        """
result = sqldf(query, locals())
print(result)
```

```
     avg(studio_grosssum)      studio
0            2.400000e+03  Synergetic
1            2.800000e+03         ALP
2            3.600000e+03       Hiber
3            4.600000e+03        TAFC
4            5.100000e+03         BSM
..                    ...         ...
253          2.296600e+08     WB (NL)
254          2.542000e+08    GrtIndia
255          4.171027e+08          BV
256          5.076500e+08        P/DW
257          8.703000e+08          HC

[258 rows x 2 columns]
```

In [317]:
```
result.info
```

Out[317]:
```
<bound method DataFrame.info of      avg(studio_grosssum)      studio
0            2.400000e+03  Synergetic
1            2.800000e+03         ALP
2            3.600000e+03       Hiber
3            4.600000e+03        TAFC
4            5.100000e+03         BSM
..                    ...         ...
253          2.296600e+08     WB (NL)
254          2.542000e+08    GrtIndia
255          4.171027e+08          BV
256          5.076500e+08        P/DW
257          8.703000e+08          HC

[258 rows x 2 columns]>
```

In [320]:
```
result = pd.DataFrame(result)
result
```

Out[320]:

	avg(studio_grosssum)	studio
0	2.400000e+03	Synergetic
1	2.800000e+03	ALP
2	3.600000e+03	Hiber
3	4.600000e+03	TAFC
4	5.100000e+03	BSM
...	...	...
253	2.296600e+08	WB (NL)
254	2.542000e+08	GrtIndia
255	4.171027e+08	BV
256	5.076500e+08	P/DW
257	8.703000e+08	HC

258 rows × 2 columns

In [336]:
```
result.info
```

Out[336]:
```
<bound method DataFrame.info of      avg(studio_grosssum)      studio
0              2.400000e+03  Synergetic
1              2.800000e+03         ALP
2              3.600000e+03       Hiber
3              4.600000e+03        TAFC
4              5.100000e+03         BSM
..                      ...         ...
253            2.296600e+08     WB (NL)
254            2.542000e+08    GrtIndia
255            4.171027e+08          BV
256            5.076500e+08        P/DW
257            8.703000e+08          HC

[258 rows x 2 columns]>
```

In [342]:
```
top10gross = result.head(10)
top10gross
```

Out[342]:

	avg(studio_grosssum)	studio
0	2400.0	Synergetic
1	2800.0	ALP
2	3600.0	Hiber
3	4600.0	TAFC
4	5100.0	BSM
5	5900.0	KS
6	10200.0	FOR
7	10800.0	Indic.
8	11500.0	CARUSEL
9	11700.0	PDF

In [348]:
```
# change column name
top10grosss = top10gross.rename(columns={'avg(studio_grosssum)': 'avarage_
top10grosss
```

Out[348]:

	avarage_gross	studio
0	2400.0	Synergetic
1	2800.0	ALP
2	3600.0	Hiber
3	4600.0	TAFC
4	5100.0	BSM
5	5900.0	KS
6	10200.0	FOR
7	10800.0	Indic.
8	11500.0	CARUSEL
9	11700.0	PDF

In [349]:
```
# Select top 10 studios average gross
top10grosss.head(10)
```

Out[349]:

	avarage_gross	studio
0	2400.0	Synergetic
1	2800.0	ALP
2	3600.0	Hiber
3	4600.0	TAFC
4	5100.0	BSM
5	5900.0	KS
6	10200.0	FOR
7	10800.0	Indic.
8	11500.0	CARUSEL
9	11700.0	PDF

In [352]:
```
# seleting topaverage gross
result1 = top10grosss.nlargest(10, 'avarage_gross')
result1
```

Out[352]:

	avarage_gross	studio
9	11700.0	PDF
8	11500.0	CARUSEL
7	10800.0	Indic.
6	10200.0	FOR
5	5900.0	KS
4	5100.0	BSM
3	4600.0	TAFC
2	3600.0	Hiber
1	2800.0	ALP
0	2400.0	Synergetic

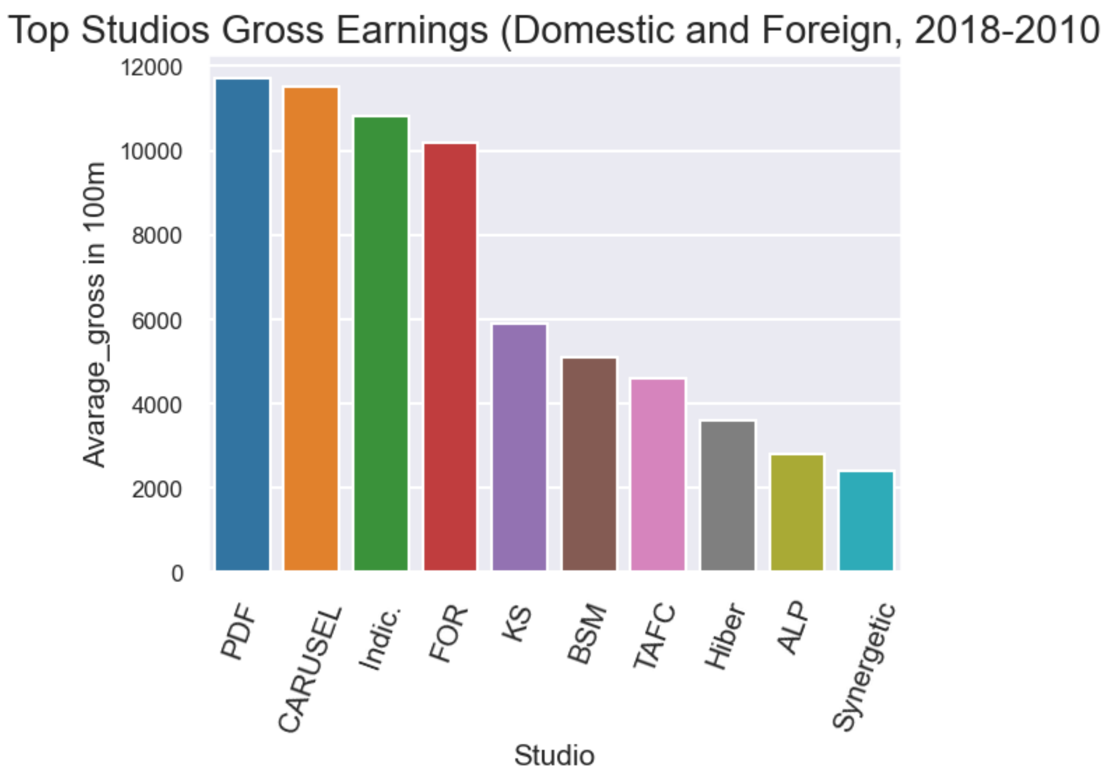
In [353]:
```
# Making the salection a panda dataframe
result1 = pd.DataFrame(result1)
result1
```

Out[353]:

	avarage_gross	studio
9	11700.0	PDF
8	11500.0	CARUSEL
7	10800.0	Indic.
6	10200.0	FOR
5	5900.0	KS
4	5100.0	BSM
3	4600.0	TAFC
2	3600.0	Hiber
1	2800.0	ALP
0	2400.0	Synergetic

In [354]:
```
# Creating a bar graph using Seaborn
sns.barplot(x='studio', y='avarage_gross', data=result1)
plt.xlabel('Studio', fontsize = 15)
plt.ylabel('Avarage_gross in 100m', fontsize = 15)
plt.title('Top Studios Gross Earnings (Domestic and Foreign, 2018-2010', f
sns.set_style('darkgrid')
sns.set_context('talk')
plt.xticks(rotation=70, fontsize=15)
plt.yticks(fontsize=12)
```

Out[354]:
```
(array([    0.,  2000.,  4000.,  6000.,  8000., 10000., 12000., 1400
0.]),
 [Text(0, 0.0, '0'),
  Text(0, 2000.0, '2000'),
  Text(0, 4000.0, '4000'),
  Text(0, 6000.0, '6000'),
  Text(0, 8000.0, '8000'),
  Text(0, 10000.0, '10000'),
  Text(0, 12000.0, '12000'),
  Text(0, 14000.0, '14000')])
```



In [57]:
```
#Importing TittleBasics dataset
TittleBasics = pd.read_csv("C:\\Users\\user\\Documents\\Phase_1\\title.bas
TittleBasics
```

Out[57]:

	tconst	primary_title	original_title	start_year	runtime_minutes	g
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,[
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,[
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	[
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,[
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fa
...	...	...	...	...	...	
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	[
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Docume
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Co
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Docume

146144 rows × 6 columns

In [55]:
```
#checking if TittleBasics
type(TittleBasics)
```

Out[55]:
```
pandas.core.frame.DataFrame
```

In [42]:
```
TittleBasics.shape
```

Out[42]:
```
(146144, 6)
```

In [43]:
```
#First 5 rows of TittleBasics Dataset
TittleBasics.head(5)
```

Out[43]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [44]:
```
#Last 3 rows of TittleBasics DataFrame
TittleBasics.tail(3)
```

Out[44]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

In [45]:
```
# Summary of TittleBasics DataFrame
TittleBasics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146143 non-null  object 
 2   original_title   146122 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [46]:
```
# Checking duplicates in TittleBasics DataFrame
TittleBasics.puplicate(). value_counts()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call la
st)
~\AppData\Local\Temp\ipykernel_16888\3927300561.py in ?()
      1 # Checking duplicates in TittleBasics DataFrame
----> 2 TittleBasics.puplicate(). value_counts()

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_
name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'puplicate'
```

In [52]:
```
# Checking Null values
TittleBasics.isnull().sum()
```

Out[52]:
```
tconst                 0
primary_title          1
original_title        22
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
```

In [53]:
```
# Sum of null entries
TittleBasics.isnull().sum().sum()
```

Out[53]:
```
37170
```

In [54]:
```
# Filling Null with 0
TittleBasics_1 = TittleBasics.fillna(value = 0)
TittleBasics_1
```

Out[54]:

	tconst	primary_title	original_title	start_year	runtime_minutes	g
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,D
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,D
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	D
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	0.0	Comedy,D
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fa
...	...	...	...	...	...	
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	D
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	0.0	Docume
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	0.0	Co
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	0.0	Docume

146144 rows × 6 columns

In [ ]:
```
# Performing an inner join on tittlebasics and tmd_movies dataframes the '
joined_df = pd.merge(, df2, on='Key', how='inner')
```

In [30]:
```
#Importing TittleRatings dataset
TittleRatings = pd.read_csv("C:\\Users\\user\\Documents\\Phase_1\\title.ra
TittleRatings
```

Out[30]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21
...	...	...	...
73851	tt9805820	8.1	25
73852	tt9844256	7.5	24
73853	tt9851050	4.7	14
73854	tt9886934	7.0	5
73855	tt9894098	6.3	128

73856 rows × 3 columns

In [58]:
```
# DataFrame shape
TittleRatings.shape
```

Out[58]:
```
(73856, 3)
```

In [59]:
```
#First 5 rows of TittleRatings DataFrame
TittleRatings.head(5)
```

Out[59]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

In [60]:
```
#Last 5 rows of TittleRatings DataFrame
TittleRatings.tail(5)
```

Out[60]:

	tconst	averagerating	numvotes
73851	tt9805820	8.1	25
73852	tt9844256	7.5	24
73853	tt9851050	4.7	14
73854	tt9886934	7.0	5
73855	tt9894098	6.3	128

In [61]:
```
# Summary of TittleRatings DataFrame
TittleRatings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   tconst         73856 non-null  object 
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

In [62]:
```
# Checking duplicates in TittleRatings DataFrame
TittleRatings.puplicate(). value_counts()
```

```
------------------------------------------------------------------------
---
AttributeError                            Traceback (most recent call la
st)
~\AppData\Local\Temp\ipykernel_16888\2671966041.py in ?()
      1 # Checking duplicates in TittleRatings DataFrame
----> 2 TittleRatings.puplicate(). value_counts()

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_
name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'puplicate'
```

In [63]:
```
# Checking Null values
TittleRatings.isnull().sum()
```

Out[63]:
```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

In [64]:
```
# Sum of null entries
TittleRatings.isnull().sum().sum()
```

Out[64]:
```
0
```

In [375]:
```
tmd_movies = pd.read_csv('C:\\Users\\user\\Documents\\Phase_1\\tmdb.movies
tmd_movies
```

Out[375]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	relea
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	11/
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	3/
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	5
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	11/
4	4	[28, 878, 12]	27205	en	Inception	27.920	7/
...	...	...	...	...	...	...	
26511	26512	[27, 18]	488143	en	Laboratory Conditions	0.600	10/
26512	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	5
26513	26514	[14, 28, 12]	381231	en	The Last One	0.600	1(
26514	26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	6/
26515	26516	[53, 27]	309885	en	The Church	0.600	1(

26516 rows × 10 columns

In [376]:
```
# tmd_movies shape
tmd_movies.shape
```

Out[376]:
```
(26516, 10)
```

In [377]:
```
# Summary of tmd_movies DataFrame
tmd_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26516 entries, 0 to 26515
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         26516 non-null  int64
 1   genre_ids          26516 non-null  object
 2   id                 26516 non-null  int64
 3   original_language  26516 non-null  object
 4   original_title     26516 non-null  object
 5   popularity         26516 non-null  float64
 6   release_date       26516 non-null  object
 7   title              26516 non-null  object
 8   vote_average       26516 non-null  float64
 9   vote_count         26516 non-null  int64
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

In [378]:
```
# Head
tmd_movies.head()
```

Out[378]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	11/19/2010
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	3/26/2010
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	5/7/2010
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	11/22/1995
4	4	[28, 878, 12]	27205	en	Inception	27.920	7/16/2010

In [379]:
```
# Tail
tmd_movies.tail()
```

Out[379]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	relea
26511	26512	[27, 18]	488143	en	Laboratory Conditions	0.6	10/
26512	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.6	5
26513	26514	[14, 28, 12]	381231	en	The Last One	0.6	1(
26514	26515	[10751, 12, 28]	366854	en	Trailer Made	0.6	6/
26515	26516	[53, 27]	309885	en	The Church	0.6	1(

In [380]:
```
# Checking duplicates in tmd_movies DataFrame
tmd_movies.puplicate(). value_counts()
```

```
--------------------------------------------------------------------------
---
AttributeError                            Traceback (most recent call la
st)
~\AppData\Local\Temp\ipykernel_16888\3147665519.py in ?()
      1 # Checking duplicates in tmd_movies DataFrame
----> 2 tmd_movies.puplicate(). value_counts()

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_
name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'puplicate'
```

In [382]:
```
# Checking Null values
tmd_movies.isnull().sum()
```

Out[382]:
```
Unnamed: 0           0
genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
vote_average         0
vote_count           0
dtype: int64
```

In [385]:
```
# Performing an inner join on TittleBasics and tmd_movies dataframes the '
joined_df = pd.merge(TittleBasics, tmd_movies, on='original_title', how='i
joined_df
```

Out[385]:

runtime_minutes	genres	Unnamed: 0	genre_ids	id	original_language
122.0	Drama	24185	[35, 18]	299782	en
NaN	Horror,Thriller	5872	[27, 878, 12]	117856	en
NaN	Action,Horror	5872	[27, 878, 12]	117856	en
86.0	Animation,Family	5872	[27, 878, 12]	117856	en
91.0	Action,Animation,Comedy	8456	[16, 28, 35, 10751]	116977	en
...	...	...	...	...	...
81.0	Drama	24305	[28, 878]	373449	ta
NaN	Comedy	24840	[35]	557606	en
63.0	Sport	23218	[]	469698	en
60.0	Documentary	25626	[27]	536235	en
NaN	Action	13030	[16]	544776	en

In [388]:
```
# Understanding the dataframe
joined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21091 entries, 0 to 21090
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   tconst             21091 non-null  object 
 1   primary_title      21091 non-null  object 
 2   original_title     21091 non-null  object 
 3   start_year         21091 non-null  int64  
 4   runtime_minutes    19465 non-null  float64
 5   genres             20792 non-null  object 
 6   Unnamed: 0         21091 non-null  int64  
 7   genre_ids          21091 non-null  object 
 8   id                 21091 non-null  int64  
 9   original_language  21091 non-null  object 
 10  popularity         21091 non-null  float64
 11  release_date       21091 non-null  object 
 12  title              21091 non-null  object 
 13  vote_average       21091 non-null  float64
 14  vote_count         21091 non-null  int64  
dtypes: float64(3), int64(4), object(8)
memory usage: 2.4+ MB
```

In [386]:
```
joined_df.head(5)
```

Out[386]:

ar	runtime_minutes	genres	Unnamed: 0	genre_ids	id	original_language
18	122.0	Drama	24185	[35, 18]	299782	er
17	NaN	Horror,Thriller	5872	[27, 878, 12]	117856	er
18	NaN	Action,Horror	5872	[27, 878, 12]	117856	er
18	86.0	Animation,Family	5872	[27, 878, 12]	117856	er
12	91.0	Action,Animation,Comedy	8456	[16, 28, 35, 10751]	116977	er

In [391]:
```
# remove unnecessary columns
joined_df = joined_df.drop \
                        (columns = ['runtime_minutes', 'genre_ids'])
```

In [416]:
```python
# filtering datafarme using more than or equal to 100
joined_df = joined_df[joined_df['vote_count'] >= 100]
joined_df
```

Out[416]:

	tconst	primary_title	original_title	start_year	genres	Unname
13	tt0326592	The Overnight	The Overnight	2010	NaN	145
17	tt0337692	On the Road	On the Road	2012	Adventure,Drama,Romance	53
23	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	Adventure,Comedy,Drama	79
24	tt0360556	Fahrenheit 451	Fahrenheit 451	2018	Drama,Sci-Fi,Thriller	240
25	tt0365545	Nappily Ever After	Nappily Ever After	2018	Comedy,Drama,Romance	240
...	...	...	...	...	...	
20953	tt8632862	Fahrenheit 11/9	Fahrenheit 11/9	2018	Documentary	241
20969	tt8709036	A Christmas Prince: The Royal Wedding	A Christmas Prince: The Royal Wedding	2018	Drama,Romance	242
20986	tt8802728	The Witch	The Witch	2018	Horror	142
21010	tt8954732	The Princess Switch	The Princess Switch	2018	Romance	239
21075	tt9495224	Black Mirror: Bandersnatch	Black Mirror: Bandersnatch	2018	Drama,Mystery,Sci-Fi	240

2545 rows × 13 columns

In [397]:
```
# Checking duplicates in joined_df DataFrame
joined_df.puplicate(). value_counts()
```

```
---------------------------------------------------------------------------
---
AttributeError                            Traceback (most recent call la
st)
~\AppData\Local\Temp\ipykernel_16888\1134796578.py in ?()
      1 # Checking duplicates in joined_df DataFrame
----> 2 joined_df.puplicate(). value_counts()

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_
name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'puplicate'
```

In [417]:
```
#drop duplicate rows
joined_df = joined_df.drop_duplicates\
                        (subset = 'id', keep = 'first')
```

In [418]:
```
# creating twolines of english and non-english movies
joined_df = joined_df \
                    [joined_df['original_language'] == 'en']
joined_df
```

Out[418]:

	tconst	primary_title	original_title	start_year	genres	Unname
13	tt0326592	The Overnight	The Overnight	2010	NaN	145
17	tt0337692	On the Road	On the Road	2012	Adventure,Drama,Romance	53
23	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	Adventure,Comedy,Drama	79
24	tt0360556	Fahrenheit 451	Fahrenheit 451	2018	Drama,Sci-Fi,Thriller	240
25	tt0365545	Nappily Ever After	Nappily Ever After	2018	Comedy,Drama,Romance	240
...	...	...	...	...	...	
20953	tt8632862	Fahrenheit 11/9	Fahrenheit 11/9	2018	Documentary	241
20969	tt8709036	A Christmas Prince: The Royal Wedding	A Christmas Prince: The Royal Wedding	2018	Drama,Romance	242
20986	tt8802728	The Witch	The Witch	2018	Horror	142
21010	tt8954732	The Princess Switch	The Princess Switch	2018	Romance	239
21075	tt9495224	Black Mirror: Bandersnatch	Black Mirror: Bandersnatch	2018	Drama,Mystery,Sci-Fi	240

2545 rows × 13 columns

In [419]:
```
joined_df.shape
```

Out[419]:
```
(2545, 13)
```

In [422]:
```python
# filtering dataframe using more than or equal to 10000
joined_df = joined_df[joined_df['vote_count'] >= 10000]
joined_df
```

	Hallows: Part 2	Hallows: Part 2					
211837	Doctor Strange	Doctor Strange	2016	Action,Adventure,Fantasy	17384	284052	en
228705	Iron Man 2	Iron Man 2	2010	Action,Adventure,Sci-Fi	2	10138	en
323594	Despicable Me	Despicable Me	2010	Animation,Comedy,Family	8	20352	en
345836	The Dark Knight Rises	The Dark Knight Rises	2012	Action,Thriller	5182	49026	en
375666	Inception	Inception	2010	Action,Adventure,Sci-Fi	4	27205	en
386697	Suicide Squad	Suicide Squad	2016	Action,Adventure,Fantasy	17437	297761	en
392170	The Hunger Games	The Hunger Games	2012	Action,Adventure,Sci-Fi	5235	70160	en
392190	Mad Max:	Mad Max:	2015	Action,Adventure,Sci-Fi	14177	76341	en

In [423]:
```python
joined_df.shape
```

Out[423]:
```
(56, 13)
```

In [424]:
```python
joined_df = pd.DataFrame(joined_df)
joined_df
```

183000	Where to Find Them	Where to Find Them	2016	Adventure,Family,Fantasy	17388	259316	en
498820	Captain America: Civil War	Captain America: Civil War	2016	Action,Adventure,Sci-Fi	17382	271110	en
501632	Thor: Ragnarok	Thor: Ragnarok	2017	Action,Adventure,Comedy	20621	284053	en
559388	The Martian	The Martian	2015	Adventure,Drama,Sci-Fi	14188	286217	en
783958	La La Land	La La Land	2016	Comedy,Drama,Music	17443	313369	en
396198	Guardians of the Galaxy Vol. 2	Guardians of the Galaxy Vol. 2	2017	Action,Adventure,Comedy	20622	283995	en
154756	Avengers: Infinity War	Avengers: Infinity War	2018	Action,Adventure,Sci-Fi	23811	299536	en
105098	The Lion King	The Lion King	2019	Adventure,Animation,Drama	2472	8587	en

In [426]:

```
# Query pandasql genres and relase month
sql_query = "SELECT genres, vote_count FROM joined_df"
joined1_df = (ps.sqldf(sql_query, locals()))
print(joined1_df)
```

```
                         genres  vote_count
0        Action,Adventure,Sci-Fi       14056
1       Action,Adventure,Fantasy       12566
2        Action,Adventure,Sci-Fi       12810
3        Action,Adventure,Comedy       11949
4       Action,Adventure,Fantasy       12764
5         Adventure,Drama,Sci-Fi       18597
6        Action,Adventure,Sci-Fi       19673
7       Adventure,Family,Fantasy       12076
8      Adventure,Fantasy,Mystery       10788
9        Action,Adventure,Sci-Fi       10411
10         Biography,Crime,Drama       12411
11     Action,Adventure,Thriller       10441
12              Mystery,Thriller       12625
13       Adventure,Drama,Fantasy       11567
14      Action,Adventure,Fantasy       12582
15       Action,Adventure,Sci-Fi       12368
16       Animation,Comedy,Family       10057
17               Action,Thriller       13933
18       Action,Adventure,Sci-Fi       22186
19      Action,Adventure,Fantasy       13533
20       Action,Adventure,Sci-Fi       14587
21       Action,Adventure,Sci-Fi       14454
22               Horror,Thriller       10931
23       Action,Adventure,Comedy       20175
24          Comedy,Drama,Family       12330
25                   Crime,Drama       12691
26    Action,Adventure,Biography       11064
27         Action,Mystery,Sci-Fi       10626
28       Action,Adventure,Sci-Fi       12365
29       Action,Adventure,Sci-Fi       11034
30                 Drama,Western       15725
31       Action,Adventure,Sci-Fi       10062
32       Action,Adventure,Sci-Fi       11170
33       Action,Adventure,Sci-Fi       10087
34      Action,Adventure,Fantasy       10171
35       Action,Adventure,Comedy       17958
36      Biography,Drama,Thriller       10396
37    Action,Adventure,Animation       10176
38       Action,Adventure,Sci-Fi       11585
39        Drama,Mystery,Thriller       10459
40       Action,Adventure,Sci-Fi       13457
41          Drama,Mystery,Sci-Fi       10387
42          Comedy,Romance,Sport       10375
43        Family,Fantasy,Musical       11023
44       Action,Adventure,Comedy       10287
45        Action,Sci-Fi,Thriller       10019
46         Action,Crime,Thriller       10081
47      Action,Adventure,Fantasy       11991
48      Adventure,Family,Fantasy       12152
49       Action,Adventure,Sci-Fi       14000
50       Action,Adventure,Comedy       11380
51        Adventure,Drama,Sci-Fi       12172
52           Comedy,Drama,Music       10028
53       Action,Adventure,Comedy       12535
```

```
54     Action,Adventure,Sci-Fi       13948
55   Adventure,Animation,Drama       10160
```

In [428]:
```
# MovieBudgets.groupby('release_month')
joined1_df.groupby('genres').mean()
```

Out[428]:

genres	vote_count
Action,Adventure,Animation	10176.000000
Action,Adventure,Biography	11064.000000
Action,Adventure,Comedy	14047.333333
Action,Adventure,Fantasy	12267.833333
Action,Adventure,Sci-Fi	13426.647059
Action,Adventure,Thriller	10441.000000
Action,Crime,Thriller	10081.000000
Action,Mystery,Sci-Fi	10626.000000
Action,Sci-Fi,Thriller	10019.000000
Action,Thriller	13933.000000
Adventure,Animation,Drama	10160.000000
Adventure,Drama,Fantasy	11567.000000
Adventure,Drama,Sci-Fi	15384.500000
Adventure,Family,Fantasy	12114.000000
Adventure,Fantasy,Mystery	10788.000000
Animation,Comedy,Family	10057.000000
Biography,Crime,Drama	12411.000000
Biography,Drama,Thriller	10396.000000
Comedy,Drama,Family	12330.000000
Comedy,Drama,Music	10028.000000
Comedy,Romance,Sport	10375.000000
Crime,Drama	12691.000000
Drama,Mystery,Sci-Fi	10387.000000
Drama,Mystery,Thriller	10459.000000
Drama,Western	15725.000000
Family,Fantasy,Musical	11023.000000
Horror,Thriller	10931.000000
Mystery,Thriller	12625.000000

In [431]:

```
#run an pandaSQL query for movie profit by month
query = """SELECT avg(vote_count), genres
           FROM joined_df
           GROUP BY genres
           ORDER BY avg(vote_count) DESC;
        """
result = sqldf(query, locals())
print(result)
```

```
    avg(vote_count)                     genres
0      15725.000000              Drama,Western
1      15384.500000     Adventure,Drama,Sci-Fi
2      14047.333333     Action,Adventure,Comedy
3      13933.000000            Action,Thriller
4      13426.647059     Action,Adventure,Sci-Fi
5      12691.000000                Crime,Drama
6      12625.000000           Mystery,Thriller
7      12411.000000      Biography,Crime,Drama
8      12330.000000        Comedy,Drama,Family
9      12267.833333    Action,Adventure,Fantasy
10     12114.000000    Adventure,Family,Fantasy
11     11567.000000     Adventure,Drama,Fantasy
12     11064.000000  Action,Adventure,Biography
13     11023.000000      Family,Fantasy,Musical
14     10931.000000            Horror,Thriller
15     10788.000000   Adventure,Fantasy,Mystery
16     10626.000000       Action,Mystery,Sci-Fi
17     10459.000000      Drama,Mystery,Thriller
18     10441.000000   Action,Adventure,Thriller
19     10396.000000    Biography,Drama,Thriller
20     10387.000000        Drama,Mystery,Sci-Fi
21     10375.000000        Comedy,Romance,Sport
22     10176.000000  Action,Adventure,Animation
23     10160.000000   Adventure,Animation,Drama
24     10081.000000       Action,Crime,Thriller
25     10057.000000     Animation,Comedy,Family
26     10028.000000         Comedy,Drama,Music
27     10019.000000       Action,Sci-Fi,Thriller
```

In [433]:
```
genres_vote = pd.DataFrame(result)
genres_vote
```

Out[433]:

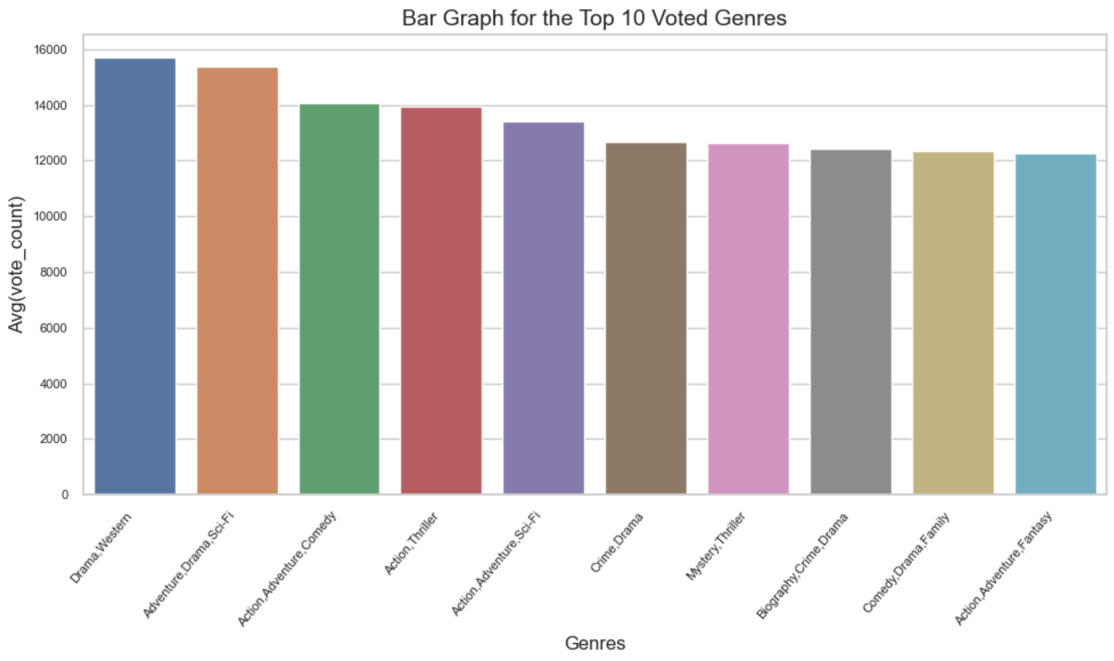
	avg(vote_count)	genres
0	15725.000000	Drama,Western
1	15384.500000	Adventure,Drama,Sci-Fi
2	14047.333333	Action,Adventure,Comedy
3	13933.000000	Action,Thriller
4	13426.647059	Action,Adventure,Sci-Fi
5	12691.000000	Crime,Drama
6	12625.000000	Mystery,Thriller
7	12411.000000	Biography,Crime,Drama
8	12330.000000	Comedy,Drama,Family
9	12267.833333	Action,Adventure,Fantasy
10	12114.000000	Adventure,Family,Fantasy
11	11567.000000	Adventure,Drama,Fantasy
12	11064.000000	Action,Adventure,Biography
13	11023.000000	Family,Fantasy,Musical
14	10931.000000	Horror,Thriller
15	10788.000000	Adventure,Fantasy,Mystery
16	10626.000000	Action,Mystery,Sci-Fi
17	10459.000000	Drama,Mystery,Thriller
18	10441.000000	Action,Adventure,Thriller
19	10396.000000	Biography,Drama,Thriller
20	10387.000000	Drama,Mystery,Sci-Fi
21	10375.000000	Comedy,Romance,Sport
22	10176.000000	Action,Adventure,Animation
23	10160.000000	Adventure,Animation,Drama
24	10081.000000	Action,Crime,Thriller
25	10057.000000	Animation,Comedy,Family
26	10028.000000	Comedy,Drama,Music
27	10019.000000	Action,Sci-Fi,Thriller

In [440]:
```
# Selected top ten votes genres
topten_genres = genres_vote.head(10)
topten_genres
```

Out[440]:

	avg(vote_count)	genres
0	15725.000000	Drama,Western
1	15384.500000	Adventure,Drama,Sci-Fi
2	14047.333333	Action,Adventure,Comedy
3	13933.000000	Action,Thriller
4	13426.647059	Action,Adventure,Sci-Fi
5	12691.000000	Crime,Drama
6	12625.000000	Mystery,Thriller
7	12411.000000	Biography,Crime,Drama
8	12330.000000	Comedy,Drama,Family
9	12267.833333	Action,Adventure,Fantasy

In [446]:
```
# Plot a bargraph
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))  # Adjusting the figure size
ax = sns.barplot(x='genres', y='avg(vote_count)', data=topten_genres)
plt.xlabel('Genres', fontsize=12)
plt.ylabel('Avg(vote_count)', fontsize=12)
plt.title('Bar Graph for the Top 10 Voted Genres', fontsize=14)
# Rotating the x-axis labels
plt.xticks(rotation=50, ha='right', fontsize=8)  # Adjust rotation and ali
plt.yticks(fontsize=8)
# Displaying the bar plot
plt.tight_layout()
plt.show()
```



In [118]:
```
#Create a dataframe for movies budget
MovieBudgets = pd.read_csv("C:\\Users\\user\\Documents\\Phase_1\\tn.movie_
MovieBudgets
```

Out[118]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [119]:
```
MovieBudgets.shape
```

Out[119]:
```
(5782, 6)
```

In [120]:
```
# Summary of MovieBudgets DataFrame
MovieBudgets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

In [130]:
```
# confirm datatype for the prodution_budget column
MovieBudgets["production_budget"].info
```

Out[130]:
```
<bound method Series.info of 0       425000000
1       410600000
2       350000000
3       330600000
4       317000000
          ...    
5777         7000
5778         6000
5779         5000
5780         1400
5781         1100
Name: production_budget, Length: 5782, dtype: int64>
```

In [ ]:
```
```

In [122]:
```
#change datatype of release date to a datetime object to extract the montl
MovieBudgets["release_date"] =pd.to_datetime\
                (MovieBudgets["release_date"])
MovieBudgets
```

Out[122]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	2015-05-01	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...	...
5777	78	2018-12-31	Red 11	$7,000	$0	$0
5778	79	1999-04-02	Following	$6,000	$48,482	$240,495
5779	80	2005-07-13	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	2015-09-29	A Plague So Pleasant	$1,400	$0	$0
5781	82	2005-08-05	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [131]:
```
# Confirm data type for the domestic gross column
MovieBudgets["domestic_gross"].info
```

Out[131]:
```
<bound method Series.info of 0       760507625
1       241063875
2        42762350
3       459005868
4       620181382
          ...
5777            0
5778        48482
5779         1338
5780            0
5781       181041
Name: domestic_gross, Length: 5782, dtype: int64>
```

In [134]:
```
#create profit column subtracting budget from worldwide gross
MovieBudgets['movie_profit'] \
        = MovieBudgets['worldwide_gross']\
        - MovieBudgets['production_budget']
```

In [135]:
```
MovieBudgets
```

Out[135]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	n
0	1	2009-12-18	Avatar	425000000	760507625	2776345279	
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	
...	...	...	...	...	...	...	
5777	78	2018-12-31	Red 11	7000	0	0	
5778	79	1999-04-02	Following	6000	48482	240495	
5779	80	2005-07-13	Return to the Land of Wonders	5000	1338	1338	
5780	81	2015-09-29	A Plague So Pleasant	1400	0	0	
5781	82	2005-08-05	My Date With Drew	1100	181041	181041	

5782 rows × 7 columns

In [137]:
```
#create month column to plot it against profit
MovieBudgets['release_month'] = \
        pd.DatetimeIndex(MovieBudgets['release_date']).month
```

In [138]:
```
MovieBudgets
```

Out[138]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	n
0	1	2009-12-18	Avatar	425000000	760507625	2776345279	
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	
...	...	...	...	...	...	...	
5777	78	2018-12-31	Red 11	7000	0	0	
5778	79	1999-04-02	Following	6000	48482	240495	
5779	80	2005-07-13	Return to the Land of Wonders	5000	1338	1338	
5780	81	2015-09-29	A Plague So Pleasant	1400	0	0	
5781	82	2005-08-05	My Date With Drew	1100	181041	181041	

5782 rows × 8 columns

In [144]:
```
# Create and SQL qyuery of the Data
```

In [172]:
```
import pandas as pd
from pandasql import sqldf
```

In [180]:
```
# Query pandasql movie_profits and relase month
sql_query = "SELECT movie_profit, release_month FROM MovieBudgets WHERE re
MoviesBudgets_1 = (ps.sqldf(sql_query, locals()))
print(MoviesBudgets_1)

# FROM MovieBudgets WHERE release_date > 2000-01-01 ORDER BY movie_profit
```

```
      movie_profit  release_month
0       2351345279             12
1        635063875              5
2       -200237650              6
3       1072413963              5
4        999721747             12
...            ...            ...
5777         -7000             12
5778        234495              4
5779         -3662              7
5780         -1400              9
5781        179941              8

[5782 rows x 2 columns]
```

In [182]:
```
# MovieBudgets.groupby('release_month')
MoviesBudgets_1.groupby('release_month').mean()
```

Out[182]:

release_month	movie_profit
1	2.572033e+07
2	4.349811e+07
3	4.985129e+07
4	3.611743e+07
5	1.151328e+08
6	9.942391e+07
7	9.841746e+07
8	3.542232e+07
9	2.488078e+07
10	2.907190e+07
11	9.314157e+07
12	6.844157e+07

In [210]:
```
# MoviesBudgets_1.info()
```

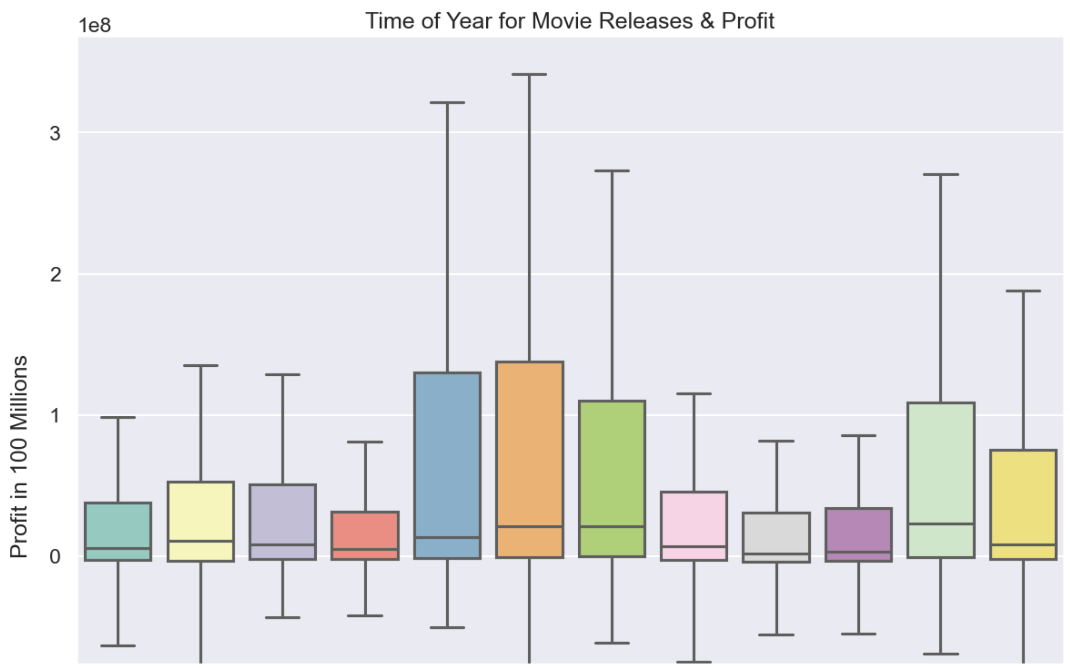
In [219]:
```
#run an pandaSQL query for movie profit by month
query = """SELECT avg(movie_profit), release_month
           FROM MovieBudgets
           GROUP BY release_month
           ORDER BY avg(movie_profit) DESC;
        """
result = sqldf(query, locals())
print(result)
```

```
    avg(movie_profit)  release_month
0        1.151328e+08              5
1        9.942391e+07              6
2        9.841746e+07              7
3        9.314157e+07             11
4        6.844157e+07             12
5        4.985129e+07              3
6        4.349811e+07              2
7        3.611743e+07              4
8        3.542232e+07              8
9        2.907190e+07             10
10       2.572033e+07              1
11       2.488078e+07              9
```

In [223]:
```
import matplotlib.pyplot as plt
```

In [225]:

```
# #create a boxplot using release month and profit
x = MovieBudgets['release_month']
y = MovieBudgets['movie_profit']
f, ax = plt.subplots(figsize=(14,12))
sns.set_style('darkgrid')
sns.set_context('talk')
sns.boxplot(x=MovieBudgets.release_month, y=MovieBudgets.movie_profit, pal
plt.title('Time of Year for Movie Releases & Profit')
plt.ylabel('Profit in 100 Millions')
plt.xlabel('Month of Movie Release')
plt.show()
```



In [79]:

In [ ]: