Final Project Submission

Please fill out:

- Student name: Sruthi Dasari
- Student pace: part time
- Scheduled project review date/time: 15-10-2023
- Instructor name: Hardik Idnani
- Blog post URL:

In [2]:

```
# First import the packages we will use in this project

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from matplotlib import style
```

In [3]:

```
# Now we need to read in the data and look into the data

df1 = pd.read_csv('zippedData/imdb.title.basics.csv.gz')
df1
```

Out[3]:

	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
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
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

146144 rows × 6 columns

In [4]:

```
df2 = pd.read_csv('zippedData/imdb.title.ratings.csv.gz')
df2
```

Out[4]:

	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 [5]:

```
#merge two datasets

data = pd.merge(df1,df2, on= 'tconst')
data
```

Out[5]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	77
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.2	43
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	6.9	4517
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1	13
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5	119
...	...	...	...	...	...	...	...	...
73851	tt9913084	Diabolik sono io	Diabolik sono io	2019	75.0	Documentary	6.2	6
73852	tt9914286	Sokagin Çocuklari	Sokagin Çocuklari	2019	98.0	Drama,Family	8.7	136
73853	tt9914642	Albatross	Albatross	2017	NaN	Documentary	8.5	8
73854	tt9914942	La vida sense la Sara Amat	La vida sense la Sara Amat	2019	NaN	NaN	6.6	5
73855	tt9916160	Drømmeland	Drømmeland	2019	72.0	Documentary	6.5	11

73856 rows × 8 columns

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 73856 entries, 0 to 73855
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           73856 non-null  object 
 1   primary_title    73856 non-null  object 
 2   original_title   73856 non-null  object 
 3   start_year       73856 non-null  int64  
 4   runtime_minutes  66236 non-null  float64
 5   genres           73052 non-null  object 
 6   averagerating    73856 non-null  float64
 7   numvotes         73856 non-null  int64  
dtypes: float64(2), int64(2), object(4)
memory usage: 5.1+ MB
```

In [7]:
```
# We need to see if we have any missing data
data.isna().any()
```

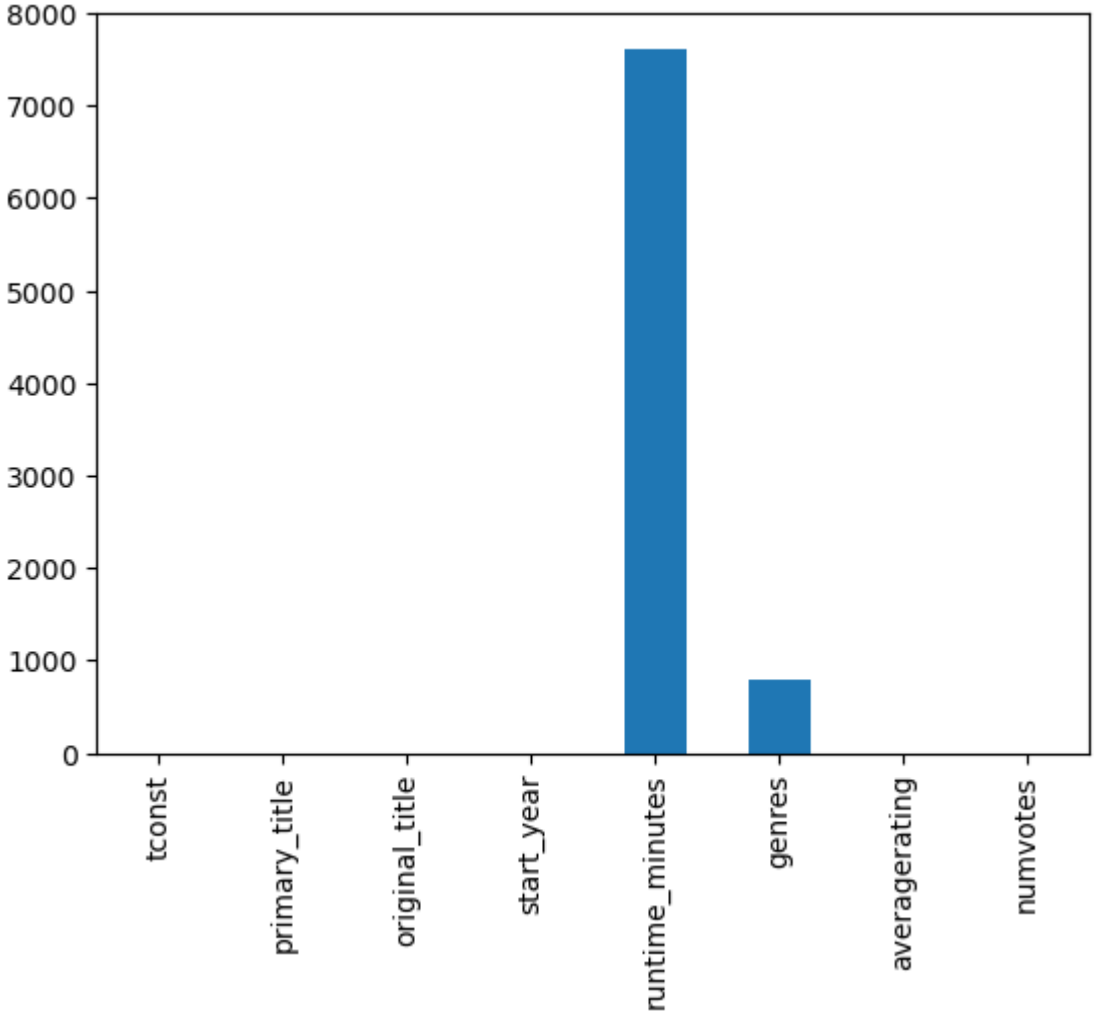
Out[7]:
```
tconst             False
primary_title      False
original_title     False
start_year         False
runtime_minutes     True
genres              True
averagerating      False
numvotes           False
dtype: bool
```

In [8]:
```
data.isna().sum()
```

Out[8]:
```
tconst               0
primary_title        0
original_title       0
start_year           0
runtime_minutes   7620
genres             804
averagerating        0
numvotes             0
dtype: int64
```

In [9]:

```
data.isna().sum().plot.bar()
plt.show()
```



In [10]:
```
#missing data in persentage
missing_data =data.isna().sum()*100/len(data)
missing_data
```

Out[10]:
```
tconst              0.000000
primary_title       0.000000
original_title      0.000000
start_year          0.000000
runtime_minutes    10.317374
genres              1.088605
averagerating       0.000000
numvotes            0.000000
dtype: float64
```

In [11]:
```
#now we need to drop all missing values
data.dropna(axis = 0, inplace = True)
```

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 65720 entries, 0 to 73855
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   tconst           65720 non-null  object
 1   primary_title    65720 non-null  object
 2   original_title   65720 non-null  object
 3   start_year       65720 non-null  int64
 4   runtime_minutes  65720 non-null  float64
 5   genres           65720 non-null  object
 6   averagerating    65720 non-null  float64
 7   numvotes         65720 non-null  int64
dtypes: float64(2), int64(2), object(4)
memory usage: 4.5+ MB
```

In [13]:
```
data.isna().sum()
```

Out[13]:
```
tconst             0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
dtype: int64
```

In [14]:
```
# now we need to find duplicate values in data set
data.duplicated()
```

Out[14]:
```
0        False
1        False
2        False
4        False
6        False
         ...
73849    False
73850    False
73851    False
73852    False
73855    False
Length: 65720, dtype: bool
```

In [15]:
```
data.duplicated().any()
```

Out[15]:
```
False
```

In [16]:
```
data.columns
```

Out[16]:
```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'averagerating', 'numvotes'],
      dtype='object')
```

In [17]:

```
#to check primary_title and original_title vlaue are same or not

(data['primary_title'] == data['original_title']).all()
```

Out[17]:

```
False
```

In [18]:

```
#now we need to rename original_title to title

data = data.rename(columns={'original_title': 'title'})
```

In [19]:

```
#now delete primary_title

data = data.drop('primary_title', axis=1)
```

In [20]:
```
data
```

Out[20]:

	tconst	title	start_year	runtime_minutes	genres	averagerating	numvotes
0	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	77
1	tt0066787	Ashad Ka Ek Din	2019	114.0	Biography,Drama	7.2	43
2	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9	4517
4	tt0100275	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	6.5	119
6	tt0137204	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy	8.1	263
...	...	...	...	...	...	...	...
73849	tt9911774	Padmavyuhathile Abhimanyu	2019	130.0	Drama	8.4	365
73850	tt9913056	Swarm Season	2019	86.0	Documentary	6.2	5
73851	tt9913084	Diabolik sono io	2019	75.0	Documentary	6.2	6
73852	tt9914286	Sokagin Çocuklari	2019	98.0	Drama,Family	8.7	136
73855	tt9916160	Drømmeland	2019	72.0	Documentary	6.5	11

65720 rows × 7 columns

In [21]:

```python
## Now we need to read in the data and look into the data
df3= pd.read_csv('zippedData/bom.movie_gross.csv.gz')
df3.head()
df3.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 [22]:

```python
#merge two datasets

big_data = pd.merge(data,df3, on= 'title')
```

In [23]:

```
big_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2400 entries, 0 to 2399
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   tconst           2400 non-null   object 
 1   title            2400 non-null   object 
 2   start_year       2400 non-null   int64  
 3   runtime_minutes  2400 non-null   float64
 4   genres           2400 non-null   object 
 5   averagerating    2400 non-null   float64
 6   numvotes         2400 non-null   int64  
 7   studio           2397 non-null   object 
 8   domestic_gross   2382 non-null   float64
 9   foreign_gross    1535 non-null   object 
 10  year             2400 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 225.0+ KB
```

In [24]:

```
big_data
```

Out[24]:

	tconst	title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross	foreig
0	tt0315642	Wazir	2016	103.0	Action,Crime,Drama	7.1	15378	Relbig.	1100000.0	
1	tt0337692	On the Road	2012	124.0	Adventure,Drama,Romance	6.1	37886	IFC	744000.0	
2	tt4339118	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.0	
3	tt5647250	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.0	
4	tt0359950	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	7.3	275300	Fox	58200000.0	12
...	...	...	...	...	...	...	...	...	...	
2395	tt8097306	Nobody's Fool	2018	110.0	Comedy,Drama,Romance	4.6	3618	Par.	31700000.0	
2396	tt8108198	Andhadhun	2018	139.0	Crime,Thriller	8.5	43409	Eros	1200000.0	
2397	tt8427036	Helicopter Eela	2018	135.0	Drama	5.4	673	Eros	72000.0	
2398	tt8549902	Oolong Courtyard	2018	103.0	Comedy	4.6	61	CL	37700.0	
2399	tt9151704	Burn the Stage: The Movie	2018	84.0	Documentary,Music	8.8	2067	Trafalgar	4200000.0	

2400 rows × 11 columns

In [25]:

```
# We need to see if we have any missing data

big_data.isna().sum()
```

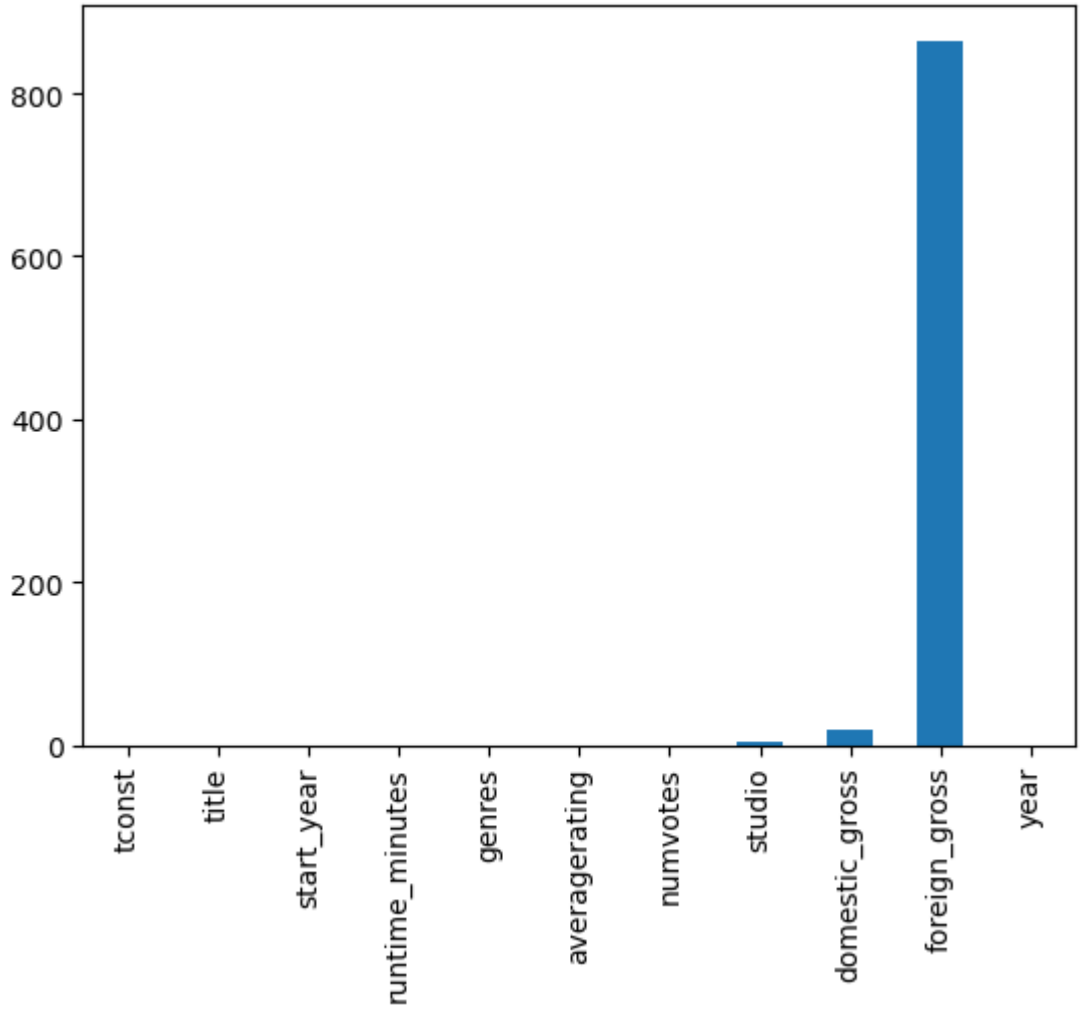
Out[25]:

```
tconst               0
title                0
start_year           0
runtime_minutes      0
genres               0
averagerating        0
numvotes             0
studio               3
domestic_gross      18
foreign_gross      865
year                 0
dtype: int64
```

In [26]:

```
#missing data in bar graph

big_data.isna().sum().plot.bar()
plt.show()
```



In [27]:

```
#now we need to drop all missing values
big_data.dropna(axis = 0, inplace = True)
big_data.isna().sum()
```

Out[27]:

```
tconst             0
title              0
start_year         0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
studio             0
domestic_gross     0
foreign_gross      0
year               0
dtype: int64
```

In [28]:

```
# we need to check any duplicate values in data set

big_data.duplicated().any()
```

Out[28]:

```
False
```

In [29]:

```
#to change genres object to list
genres_list = []
for i in big_data['genres']:
    genres_list.append(i.split(','))
genres_list
```

Out[29]:

```
[['Adventure', 'Drama', 'Romance'],
 ['Drama'],
 ['Drama'],
 ['Adventure', 'Comedy', 'Drama'],
 ['Action', 'Crime', 'Drama'],
 ['Action', 'Adventure', 'Sci-Fi'],
 ['Action', 'Crime', 'Drama'],
 ['Action', 'Comedy', 'Crime'],
 ['Comedy', 'Drama'],
 ['Comedy', 'Family'],
 ['Adventure', 'Animation', 'Comedy'],
 ['Action', 'Adventure', 'Sci-Fi'],
 ['Action', 'Sci-Fi', 'Thriller'],
 ['Comedy'],
 ['Action', 'Adventure', 'Thriller'],
 ['Horror', 'Mystery', 'Thriller'],
 ['Action', 'Drama', 'Family'],
 ['Drama', 'Romance', 'Sci-Fi'],
 ['Adventure', 'Animation', 'Comedy'],
```

In [30]:

```
big_data['genres'] = genres_list
big_data.head()
```

Out[30]:

	tconst	title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross	foreign_gross	year
1	tt0337692	On the Road	2012	124.0	[Adventure, Drama, Romance]	6.1	37886	IFC	744000.0	8000000	2012
2	tt4339118	On the Road	2014	89.0	[Drama]	6.0	6	IFC	744000.0	8000000	2012
3	tt5647250	On the Road	2016	121.0	[Drama]	5.7	127	IFC	744000.0	8000000	2012
4	tt0359950	The Secret Life of Walter Mitty	2013	114.0	[Adventure, Comedy, Drama]	7.3	275300	Fox	58200000.0	129900000	2013
5	tt0365907	A Walk Among the Tombstones	2014	114.0	[Action, Crime, Drama]	6.5	105116	Uni.	26300000.0	26900000	2014

In [31]:

```
#change each list of genres to individuals
big_data= big_data.explode('genres')
big_data
```

Out[31]:

	tconst	title	start_year	runtime_minutes	genres	averagerating	numvotes	studio	domestic_gross	foreign_gross	year
1	tt0337692	On the Road	2012	124.0	Adventure	6.1	37886	IFC	744000.0	8000000	2012
1	tt0337692	On the Road	2012	124.0	Drama	6.1	37886	IFC	744000.0	8000000	2012
1	tt0337692	On the Road	2012	124.0	Romance	6.1	37886	IFC	744000.0	8000000	2012
2	tt4339118	On the Road	2014	89.0	Drama	6.0	6	IFC	744000.0	8000000	2012
3	tt5647250	On the Road	2016	121.0	Drama	5.7	127	IFC	744000.0	8000000	2012
...	...	...	...	...	...	...	...	...	...	...	...
2395	tt8097306	Nobody's Fool	2018	110.0	Comedy	4.6	3618	Par.	31700000.0	1800000	2018
2395	tt8097306	Nobody's Fool	2018	110.0	Drama	4.6	3618	Par.	31700000.0	1800000	2018
2395	tt8097306	Nobody's Fool	2018	110.0	Romance	4.6	3618	Par.	31700000.0	1800000	2018
2399	tt9151704	Burn the Stage: The Movie	2018	84.0	Documentary	8.8	2067	Trafalgar	4200000.0	16100000	2018
2399	tt9151704	Burn the Stage: The Movie	2018	84.0	Music	8.8	2067	Trafalgar	4200000.0	16100000	2018

3777 rows × 11 columns

In [32]:

```
#to get overall statistics about data
big_data.describe()
```

Out[32]:

	start_year	runtime_minutes	averagerating	numvotes	domestic_gross	year
count	3777.000000	3777.000000	3777.000000	3.777000e+03	3.777000e+03	3777.000000
mean	2013.682023	108.950755	6.434525	1.186741e+05	5.926911e+07	2013.757744
std	2.638891	18.948308	0.977512	1.652693e+05	8.534017e+07	2.620473
min	2010.000000	3.000000	1.600000	5.000000e+00	7.000000e+02	2010.000000
25%	2011.000000	96.000000	5.900000	1.758500e+04	4.800000e+06	2011.000000
50%	2014.000000	106.000000	6.500000	6.104600e+04	3.100000e+07	2014.000000
75%	2016.000000	119.000000	7.100000	1.448210e+05	7.320000e+07	2016.000000
max	2019.000000	184.000000	9.200000	1.841066e+06	7.001000e+08	2018.000000

In [33]:

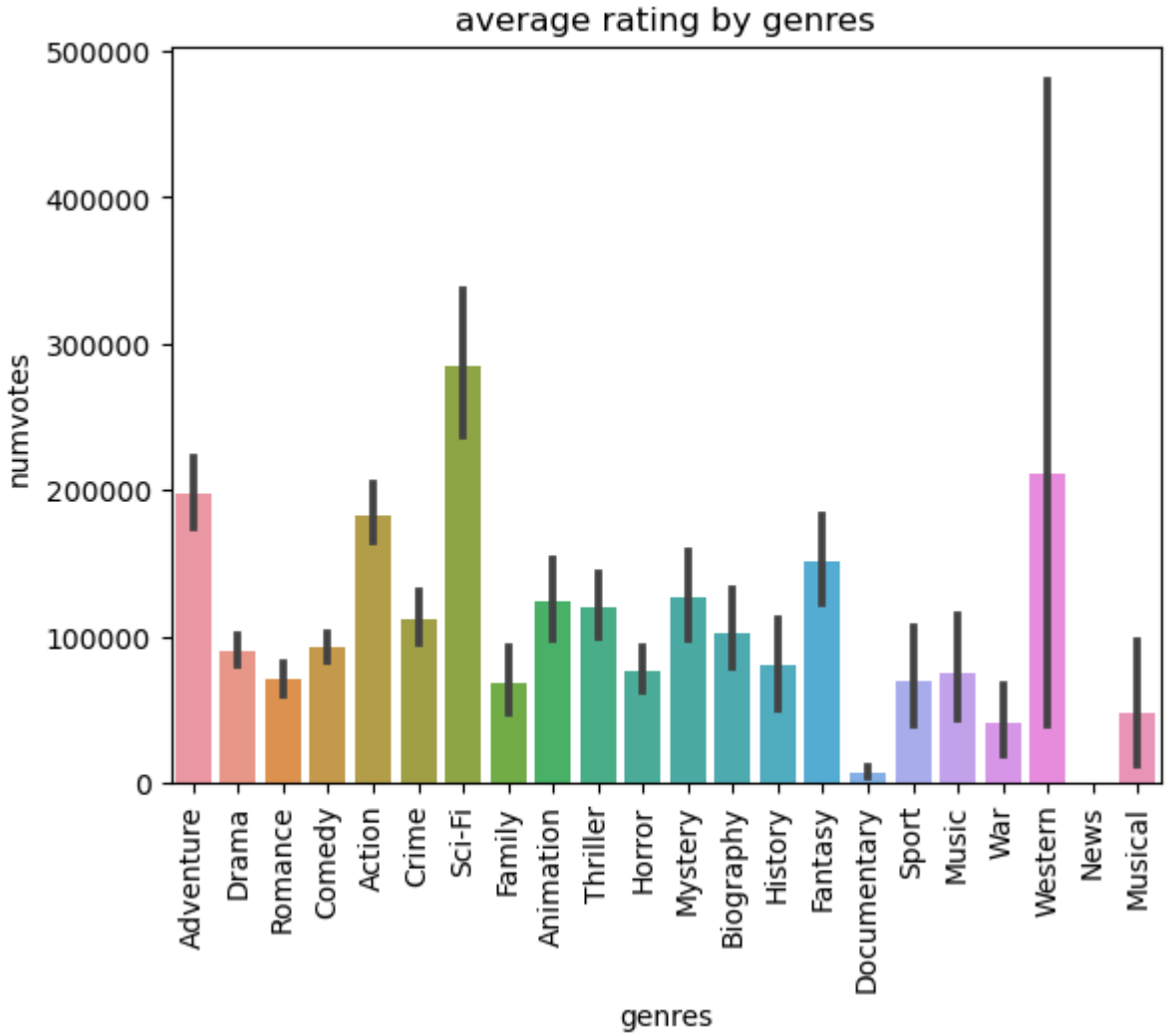
```
#to plot numvotes  and genres
round(big_data.groupby('genres')['numvotes'].mean().sort_values(ascending = False) , 2)
```

Out[33]:

```
genres
Sci-Fi         284557.14
Western        211835.67
Adventure      197881.45
Action         183216.10
Fantasy        151937.22
Mystery        127260.49
Animation      123762.24
Thriller       120452.02
Crime          111856.59
Biography      102643.64
Comedy          92426.49
Drama           90286.30
History         80219.52
Horror          76821.69
Music           75045.27
Romance         70723.04
Sport           69910.83
Family          68003.96
Musical         47826.33
War             41220.43
Documentary      7517.60
News                8.00
Name: numvotes, dtype: float64
```

In [34]:

```
sns.barplot(y = 'numvotes' , x = 'genres', data = big_data)
plt.xticks(rotation='vertical')
plt.title('average rating by genres')
plt.show()
```



In [35]:

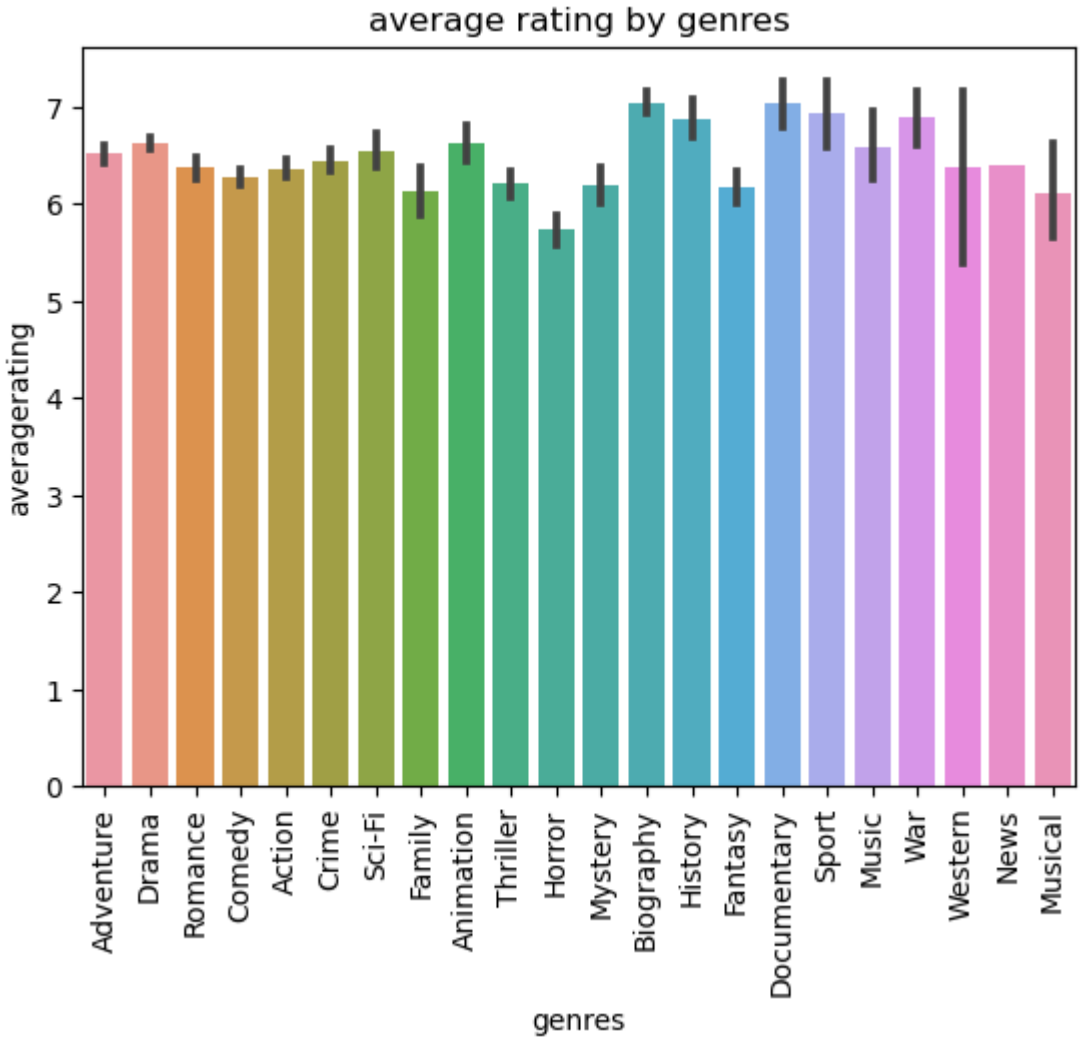
```
#to plot average rating  and genres
round(big_data.groupby('genres')['averagerating'].mean().sort_values(ascending = False) , 2)
```

Out[35]:

```
genres
Biography      7.04
Documentary    7.03
Sport          6.94
War            6.89
History        6.87
Drama          6.62
Animation      6.62
Music          6.59
Sci-Fi         6.54
Adventure      6.51
Crime          6.44
News           6.40
Western        6.38
Romance        6.36
Action         6.36
Comedy         6.26
Thriller       6.20
Mystery        6.19
Fantasy        6.17
Family         6.12
Musical        6.11
Horror         5.73
Name: averagerating, dtype: float64
```

In [36]:

```
sns.barplot(y = 'averagerating' , x = 'genres', data = big_data)
plt.xticks(rotation='vertical')
plt.title('average rating by genres')
plt.show()
```



In [37]:

```
# highest average gross as per genres
big_data['foreign_gross'] = [float(str(i).replace(",", "")) for i in big_data['foreign_gross']]
big_data['total_gross'] = big_data['foreign_gross'] + big_data['domestic_gross']
```

In [38]:

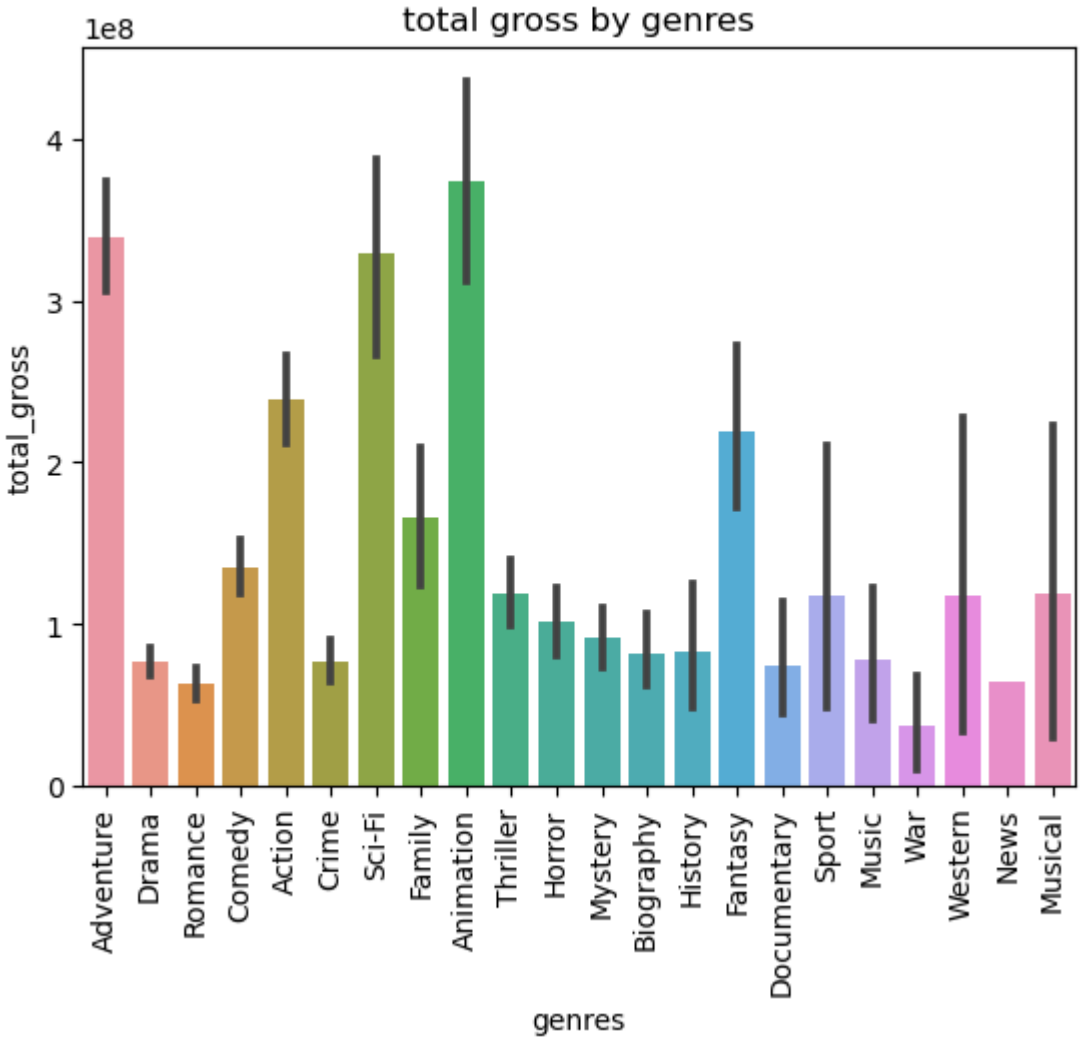
```
big_data.groupby('genres')['total_gross'].mean().sort_values(ascending=False)
```

Out[38]:

```
genres
Animation      3.736899e+08
Adventure      3.389146e+08
Sci-Fi         3.287856e+08
Action         2.384007e+08
Fantasy        2.187723e+08
Family         1.655285e+08
Comedy         1.352298e+08
Musical        1.191816e+08
Thriller       1.186467e+08
Sport          1.176932e+08
Western        1.173332e+08
Horror         1.013936e+08
Mystery        9.139734e+07
History        8.261405e+07
Biography      8.155982e+07
Music          7.877163e+07
Crime          7.740127e+07
Drama          7.715433e+07
Documentary    7.493128e+07
News           6.460000e+07
Romance        6.299393e+07
War            3.772676e+07
Name: total_gross, dtype: float64
```

In [39]:

```
sns.barplot(x='genres' , y= 'total_gross', data = big_data)
plt.xticks(rotation='vertical')
plt.title('total gross by genres')
plt.show()
```



In [75]:

```
#using groupby method create numvotes, totla_gross, averagerating using genres
grouped = big_data.groupby('genres')['numvotes', 'total_gross', 'averagerating'].mean()
grouped.reset_index()
```

```
C:\Users\vasun\AppData\Local\Temp\ipykernel_20664\2137210240.py:2: FutureWarning: Indexing with multiple keys (i
mplicitly converted to a tuple of keys) will be deprecated, use a list instead.
  grouped = big_data.groupby('genres')['numvotes', 'total_gross', 'averagerating'].mean()
```

Out[75]:

	genres	numvotes	total_gross	averagerating
0	Action	183216.101010	2.384007e+08	6.361364
1	Adventure	197881.454545	3.389146e+08	6.506667
2	Animation	123762.244898	3.736899e+08	6.618367
3	Biography	102643.641892	8.155982e+07	7.036486
4	Comedy	92426.491429	1.352298e+08	6.263619
5	Crime	111856.591928	7.740127e+07	6.435874
6	Documentary	7517.602564	7.493128e+07	7.029487
7	Drama	90286.300261	7.715433e+07	6.618668
8	Family	68003.962963	1.655285e+08	6.120988
9	Fantasy	151937.222222	2.187723e+08	6.168519
10	History	80219.516667	8.261405e+07	6.871667
11	Horror	76821.687075	1.013936e+08	5.727211
12	Music	75045.272727	7.877163e+07	6.586364
13	Musical	47826.333333	1.191816e+08	6.111111
14	Mystery	127260.491935	9.139734e+07	6.187903
15	News	8.000000	6.460000e+07	6.400000
16	Romance	70723.037915	6.299393e+07	6.363507
17	Sci-Fi	284557.137615	3.287856e+08	6.544954
18	Sport	69910.828571	1.176932e+08	6.937143
19	Thriller	120452.015326	1.186467e+08	6.197701
20	War	41220.428571	3.772676e+07	6.892857
21	Western	211835.666667	1.173332e+08	6.377778

In [52]:

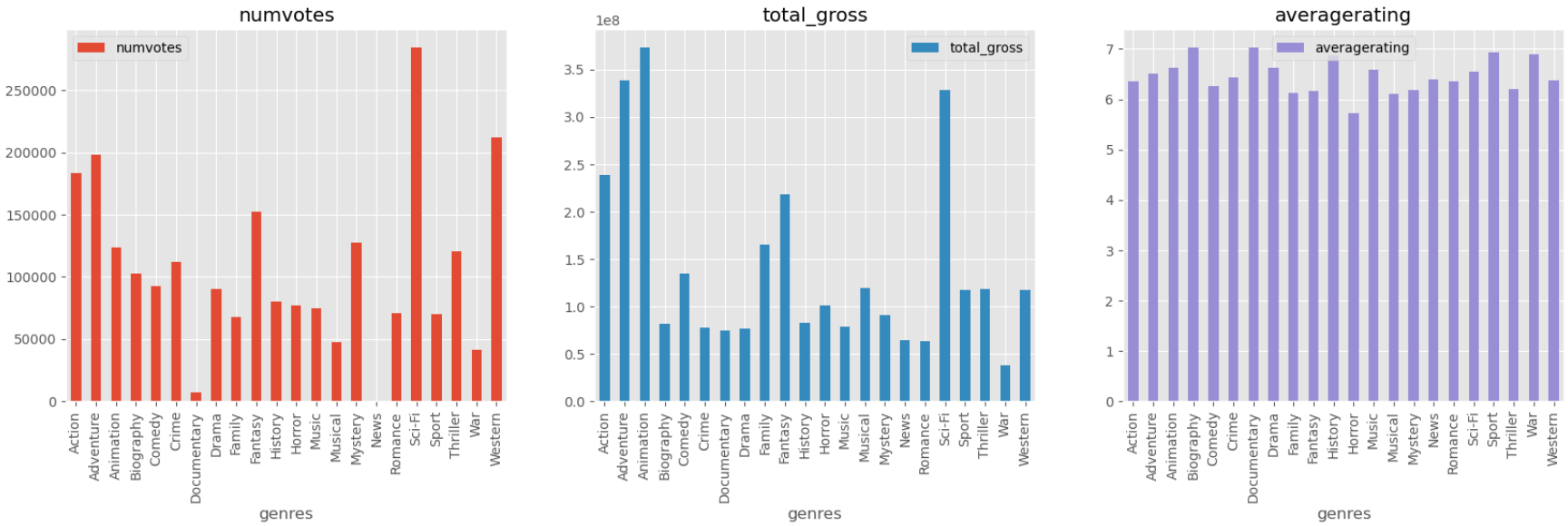
```
grouped.describe(include='all')
```

Out[52]:

	numvotes	total_gross	averagerating
count	22.000000	2.200000e+01	22.000000
mean	106155.092278	1.410327e+08	6.470829
std	67622.765755	9.679249e+07	0.336945
min	8.000000	3.772676e+07	5.727211
25%	70113.880907	7.774386e+07	6.214181
50%	91356.395845	1.093634e+08	6.417937
75%	126385.930176	1.579538e+08	6.618593
max	284557.137615	3.736899e+08	7.036486

In [41]:

```
#create plots
style.use('ggplot')
grouped.plot(subplots=True, layout=(1, 3), figsize=(20, 5), sharex=True , kind= 'bar')
plt.show()
```



In [45]:

```
#to disply highest rating genres with titles
big_data.nlargest(10, 'averagerating')[['title' , 'genres']]
```

Out[45]:

	title	genres
162	The Runaways	Adventure
575	The Wall	Documentary
452	Inception	Action
452	Inception	Adventure
452	Inception	Sci-Fi
2399	Burn the Stage: The Movie	Documentary
2399	Burn the Stage: The Movie	Music
928	Eyes Wide Open	Documentary
928	Eyes Wide Open	History
94	Interstellar	Adventure

In [61]:

```
#to disply highest numvotes genres with titles
big_data.nlargest(10, 'numvotes')[['title' , 'numvotes']]
```

Out[61]:

	title	numvotes
452	Inception	1841066
452	Inception	1841066
452	Inception	1841066
429	The Dark Knight Rises	1387769
429	The Dark Knight Rises	1387769
94	Interstellar	1299334
94	Interstellar	1299334
94	Interstellar	1299334
1127	Django Unchained	1211405
1127	Django Unchained	1211405

In [87]:

```
#to disply highest total_gross genres with titles
big_data[big_data['total_gross'].max() == big_data['total_gross']][['title', 'genres']]
```

Out[87]:

	title	genres
1563	Avengers: Age of Ultron	Action
1563	Avengers: Age of Ultron	Adventure
1563	Avengers: Age of Ultron	Sci-Fi

In [81]:

```
#to calculate the number of votes garnered by the 70% movie.
big_data['numvotes'].quantile(0.70)
```

Out[81]: 119789.0

In [97]:

```
#to display the movies (title, runtime) longer than 30 minutes or shorter than 360 minutes.
big_data[(big_data['runtime_minutes'] <30) | (big_data['runtime_minutes']>360)][['title' , 'genres']]
```

Out[97]:

	title	genres
293	Limitless	Biography
293	Limitless	Documentary

In [104]:

```
#to display the movies (title, number of votes) that received specified number of votes.
n = 5000
big_data[big_data['numvotes'] >= n][['title', 'numvotes']].sort_values(by='numvotes', ascending=False)
```

Out[104]:

	title	numvotes
452	Inception	1841066
452	Inception	1841066
452	Inception	1841066
429	The Dark Knight Rises	1387769
429	The Dark Knight Rises	1387769
...	...	...
2057	My Little Pony: The Movie	5133
2057	My Little Pony: The Movie	5133
2057	My Little Pony: The Movie	5133
1378	I Still See You	5010
1378	I Still See You	5010

3297 rows × 2 columns

In [ ]:

In [ ]: