Importing Libraries

In [1]:
```
import pandas as pd
import glob
import csv
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import json
```

Loading the datasets

In [2]:
```
#Load the datasets
movie_gross = pd.read_csv("bom.movie_gross.csv")

movie_gross.shape
```

Out[2]: (3387, 5)

In [3]:
```
movie_gross.rename(columns={'title': 'primary_title'}, inplace=True)

movie_gross.head()
```

Out[3]:

	primary_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 [4]:
```
name_basics = pd.read_csv("name.basics.csv")

name_basics.head()
```

Out[4]:

	nconst	primary_name	birth_year	death_year	primary_profession
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator

In [5]:
```
title_akas = pd.read_csv("title.akas.csv")

title_akas.head()
```

Out[5]:

	title_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик свят	BG	bg	NaN	NaN	0.0
1	tt0369610	11	Jurashikku warudo	JP	NaN	imdbDisplay	NaN	0.0
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	NaN	imdbDisplay	NaN	0.0
3	tt0369610	13	O Mundo dos Dinossauros	BR	NaN	NaN	short title	0.0
4	tt0369610	14	Jurassic World	FR	NaN	imdbDisplay	NaN	0.0

In [6]:
```
title_basics = pd.read_csv("title.basics.csv")

title_basics.head()
```

Out[6]:

	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 [7]:
```
title_crew = pd.read_csv("title.crew.csv")

title_crew.head()
```

Out[7]:

	tconst	directors	writers
0	tt0285252	nm0899854	nm0899854
1	tt0438973	NaN	nm0175726,nm1802864
2	tt0462036	nm1940585	nm1940585
3	tt0835418	nm0151540	nm0310087,nm0841532
4	tt0878654	nm0089502,nm2291498,nm2292011	nm0284943

In [8]:
```
title_principals = pd.read_csv("title.principals.csv")

title_principals.head()
```

Out[8]:

	tconst	ordering	nconst	category	job	characters
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]
1	tt0111414	2	nm0398271	director	NaN	NaN
2	tt0111414	3	nm3739909	producer	producer	NaN
3	tt0323808	10	nm0059247	editor	NaN	NaN
4	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]

In [9]:
```
title_ratings = pd.read_csv("title.ratings.csv")

title_ratings.head()
```

Out[9]:

	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 [10]:
```
tmdb_movies = pd.read_csv("tmdb.movies.csv")

tmdb_movies.head()
```

Out[10]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vc
0	0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	

In [11]:
```
tn_movie_budgets = pd.read_csv("tn.movie_budgets.csv")

tn_movie_budgets.head()
```

Out[11]:

	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

Merging Relevant Dataframes

In [12]:
```
# First Merge
# Merge title_basics,title_rating on the common identifier 'tconst'

Merged_movies = pd.merge(title_basics,title_ratings, on = 'tconst', how='left')

Merged_movies.head()
```

Out[12]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagera
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 [13]:
```
# Second Merge
# Merging merged_movies,movie_gross on the common identifier 'primary_title'

final_merged_movies = pd.merge(Merged_movies,movie_gross, on = 'primary_title', ho

final_merged_movies.head()
```

Out[13]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagera
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 [14]:
```
final_merged_movies.shape
```

Out[14]:
```
(146146, 12)
```

In [16]:
```
# final_merged_movies.to_csv("D:\df.csv", index=False)
```

In [17]:
```
# Cleaning the final_merged_movies
# dropping null values under domestic gross and foreign gross
```

In [18]:
```
final_merged_movies.isnull().sum()
```

Out[18]:
```
tconst                  0
primary_title           0
original_title         21
start_year              0
runtime_minutes     31739
genres               5408
averagerating       72288
numvotes            72288
studio             142783
domestic_gross     142804
foreign_gross      144103
year               142780
dtype: int64
```

Data Cleaning And Preparation

In [19]:
```
print("Any missing values?", final_merged_movies.isnull().values.any())
```

```
Any missing values? True
```

In [20]:
```
clean_movies = final_merged_movies.dropna()

clean_movies
```

Out[20]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	a
48	tt0337692	On the Road	On the Road	2012	124.0	Adventure,Drama,Romance	
54	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	
58	tt0365907	A Walk Among the Tombstones	A Walk Among the Tombstones	2014	114.0	Action,Crime,Drama	
60	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adventure,Sci-Fi	
61	tt0372538	Spy	Spy	2011	110.0	Action,Crime,Drama	
...	...	...	...	...	...	...	
140828	tt9151704	Burn the Stage: The Movie	Burn the Stage: The Movie	2018	84.0	Documentary,Music	
141376	tt9225192	Unstoppable	Seongnan hwangso	2018	116.0	Action,Crime	
142619	tt9392532	Neighbors	Neighbors	2018	90.0	Comedy,Drama	
142940	tt9447594	The Gambler	The Gambler	2019	121.0	Action,Sci-Fi,Thriller	
146080	tt9906218	Unstoppable	Unstoppable	2019	84.0	Documentary	

1767 rows × 12 columns

In [21]:
```
clean_movies.isnull().sum()
```

Out[21]:
```
tconst             0
primary_title      0
original_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
```

Data Selection From Final Merged, cleaned Dataframe

In [22]:
```
# Selecting Top 20 movies from clean_movies for analysis

top_20_movies_averagerating = clean_movies.nlargest(20,'averagerating')

top_20_movies_averagerating.sample(1)
```

Out[22]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating
118871	tt7130472	Stronger	Stronger	2016	47.0	Action,Sport	8.4

In [23]:
```
# Selecting top 20 from the top 100 for detailed analysis

top_20_movies_gross = clean_movies.nlargest(20,'domestic_gross')

top_20_movies_gross.head()
```

Out[23]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	av
19050	tt1825683	Black Panther	Black Panther	2018	134.0	Action,Adventure,Sci-Fi	
72821	tt4154756	Avengers: Infinity War	Avengers: Infinity War	2018	149.0	Action,Adventure,Sci-Fi	
60	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adventure,Sci-Fi	
42224	tt2527336	Star Wars: The Last Jedi	Star Wars: Episode VIII - The Last Jedi	2017	152.0	Action,Adventure,Fantasy	
62742	tt3606756	Incredibles 2	Incredibles 2	2018	118.0	Action,Adventure,Animation	

In [24]:
```
# Selecting top 20 movies from the top 100 movies based on domestic gross for deta

top_20_movies_domestic_gross = clean_movies.nlargest(20,'domestic_gross')

top_20_movies_domestic_gross.head()
```

Out[24]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	av
19050	tt1825683	Black Panther	Black Panther	2018	134.0	Action,Adventure,Sci-Fi	
72821	tt4154756	Avengers: Infinity War	Avengers: Infinity War	2018	149.0	Action,Adventure,Sci-Fi	
60	tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adventure,Sci-Fi	
42224	tt2527336	Star Wars: The Last Jedi	Star Wars: Episode VIII - The Last Jedi	2017	152.0	Action,Adventure,Fantasy	
62742	tt3606756	Incredibles 2	Incredibles 2	2018	118.0	Action,Adventure,Animation	

Analysis of top_20_movies

In [25]:
```
# Leading Business Questions

# 1. What are the top 20 movies by rating that Microsoft as a company can consider

# 2. Which genres of the movies are highly rated by consumers?

# 3. Which movies have better return on investment on the local market?
```

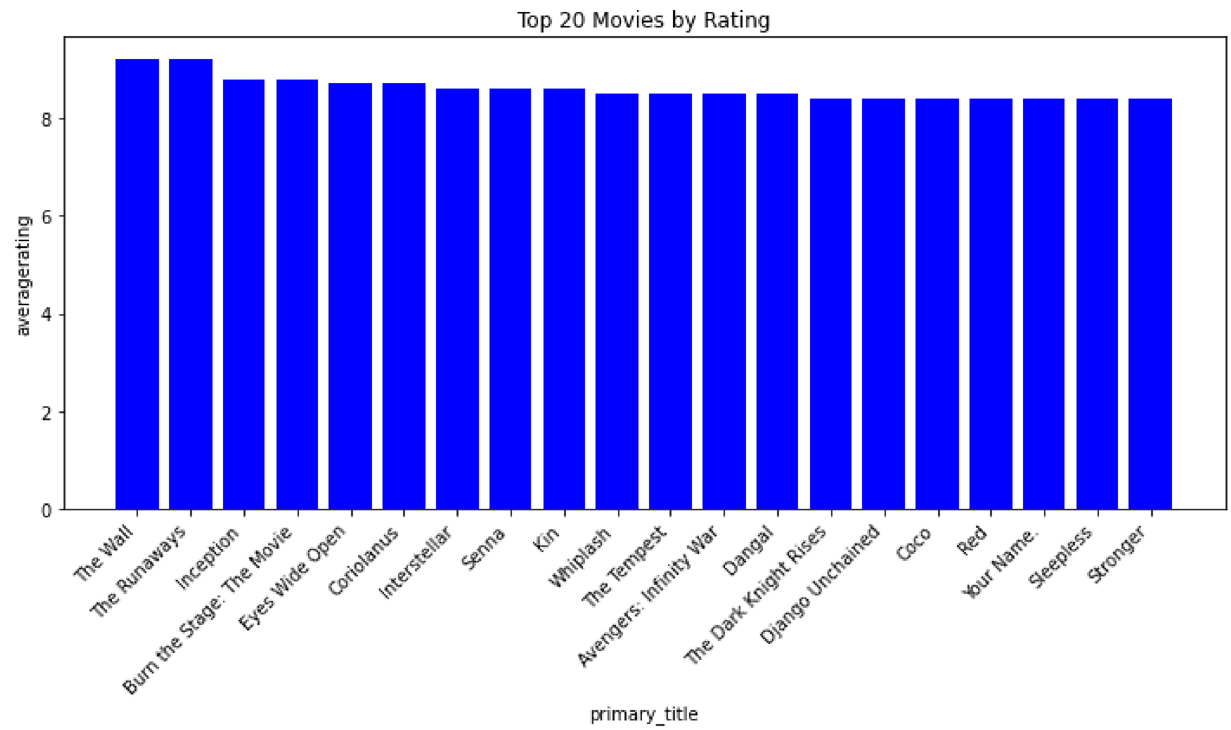
In [26]:
```
pd.options.display.float_format = '{:.2f}'.format
top_20_movies_gross.describe()
```

Out[26]:

	start_year	runtime_minutes	averagerating	numvotes	domestic_gross	year
count	20.00	20.00	20.00	20.00	20.00	20.00
mean	2015.05	124.80	7.22	473962.70	485555000.00	2015.40
std	2.61	26.98	0.94	325397.60	105038521.63	2.41
min	2010.00	60.00	4.20	13.00	400700000.00	2010.00
25%	2013.00	114.25	7.00	217729.25	411700000.00	2013.00
50%	2016.00	131.50	7.30	501837.50	421200000.00	2016.00
75%	2017.00	143.00	7.73	666927.00	551300000.00	2017.00
max	2018.00	164.00	8.50	1387769.00	700100000.00	2018.00

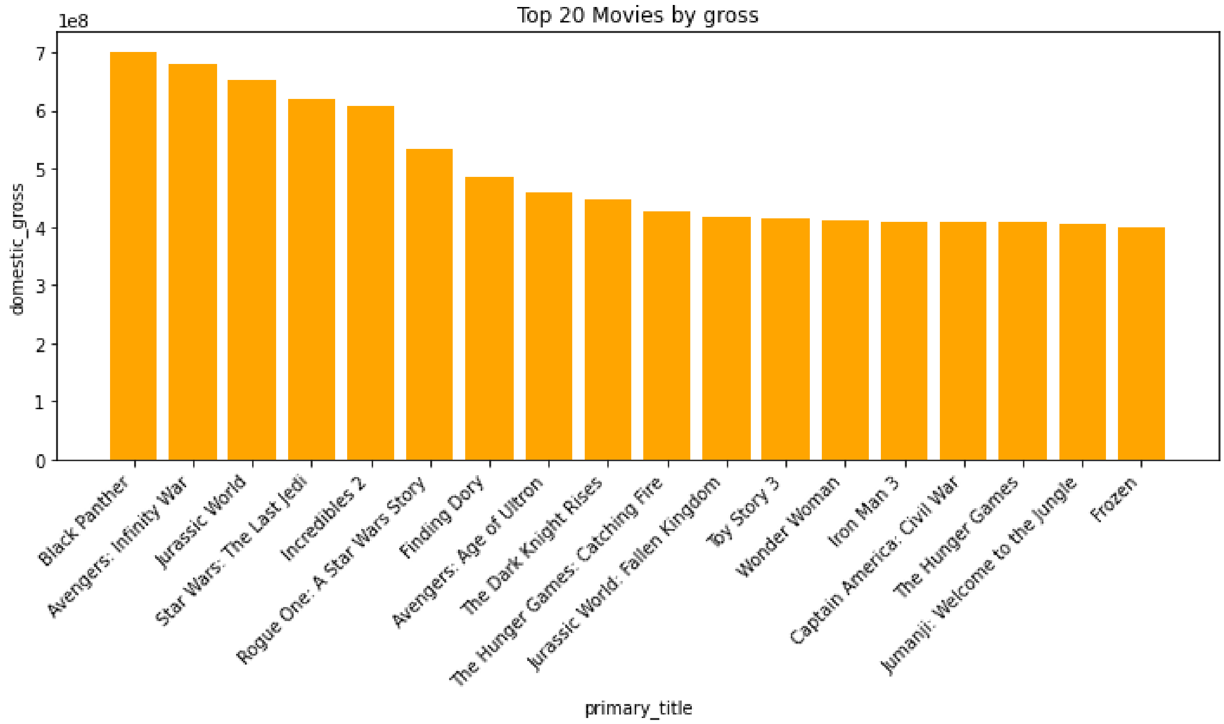
In [27]:

```
# Creating a bar plot of the top 20 movies by rating
# sns.barplot (x = "primary_title" , y = "averagerating", data = top_100_movies_len

plt.figure(figsize=(10, 6))
plt.bar(top_20_movies_averagerating['primary_title'], top_20_movies_averagerating[
plt.xlabel('primary_title')
plt.ylabel('averagerating')
plt.title('Top 20 Movies by Rating')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



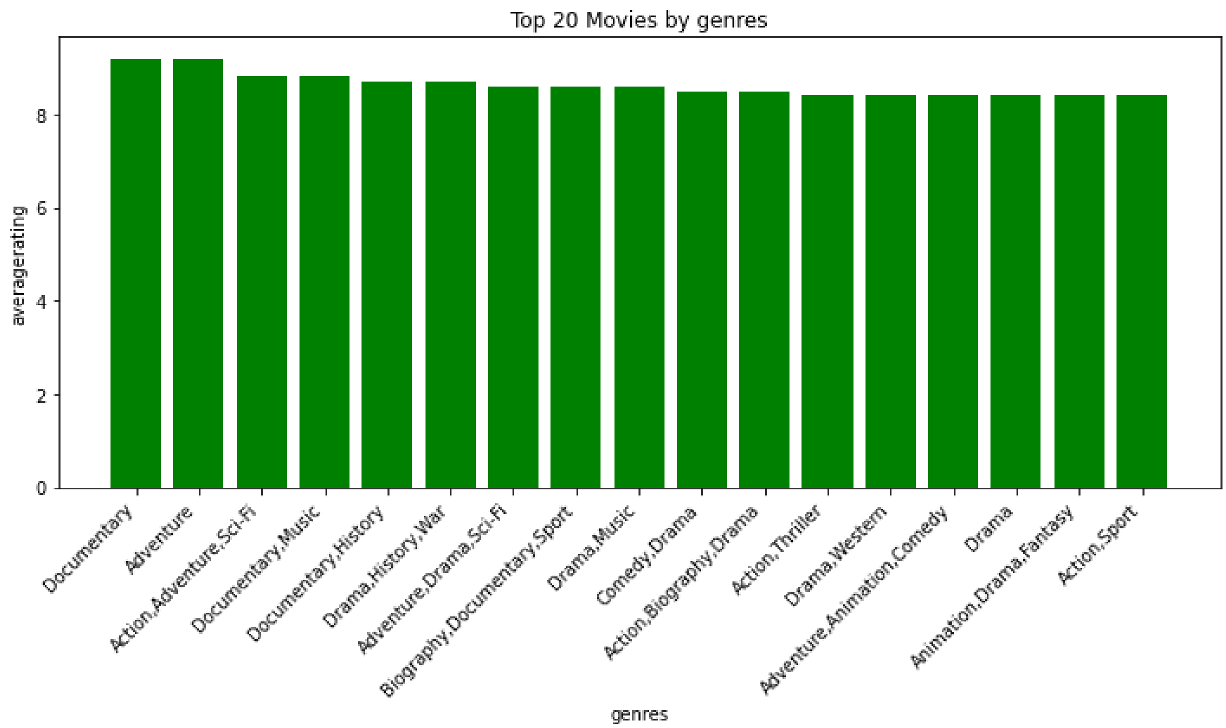
In [28]:

```
# Creating a bar plot of the top 20 movies by gross
# sns.barplot (x = "primary_title" , y = "domestic_gross", data = top_100_movies_le

plt.figure(figsize=(10, 6))
plt.bar(top_20_movies_gross['primary_title'], top_20_movies_gross['domestic_gross'
plt.xlabel('primary_title')
plt.ylabel('domestic_gross')
plt.title('Top 20 Movies by gross')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [29]:

```
# Creating a bar plot of the top 20 movies by genres
# sns.barplot (x = "primary_title" , y = "averagerating", data = top_100_movies_len

plt.figure(figsize=(10, 6))
plt.bar(top_20_movies_averagerating['genres'], top_20_movies_averagerating['averag
plt.xlabel('genres')
plt.ylabel('averagerating')
plt.title('Top 20 Movies by genres')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Selecting Top 20 movies by runtime_minutes from the top 100 movies

In [31]:
```
# Selecting top 20 movies by runtime_minutes

top_20_movies_runtime_minutes = clean_movies.nlargest(20,'runtime_minutes')

top_20_movies_runtime_minutes.head()
```

Out[31]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	ave
6067	tt1236371	Mysteries of Lisbon	Mistérios de Lisboa	2010	272.00	Drama,Mystery,Romance	
56700	tt3313066	Coriolanus	National Theatre Live: Coriolanus	2014	192.00	Drama,History,War	
73162	tt4169250	M.S. Dhoni: The Untold Story	M.S. Dhoni: The Untold Story	2016	184.00	Biography,Drama,Sport	
7292	tt1403047	Aurora	Aurora	2010	181.00	Drama	
100713	tt5886728	Another Year	You yi nian	2016	181.00	Documentary	

Selecting movies which have a runtime => 180

In [32]:
```
top_20_movies_runtime_minutes.columns
```

Out[32]:
```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'averagerating', 'numvotes', 'studio',
       'domestic_gross', 'foreign_gross', 'year'],
      dtype='object')
```

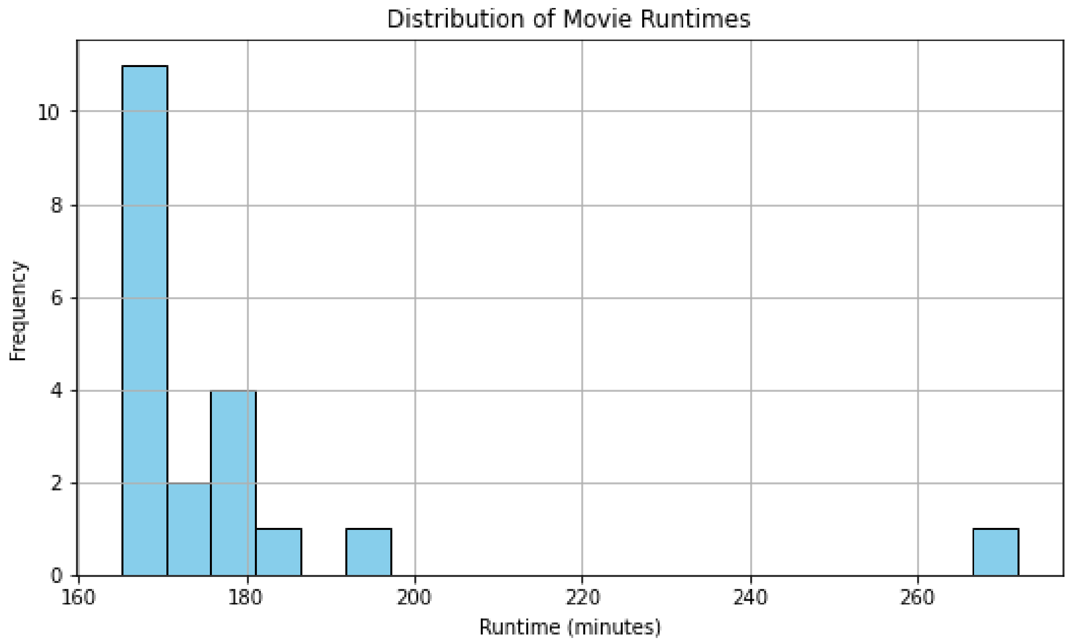
In [33]:
```
top_20_movies_runtime_minutes["runtime_minutes"] >= 180
```

Out[33]:
```
6067       True
56700      True
73162      True
7292       True
100713     True
545        True
34528      True
7039      False
19452     False
83627     False
311       False
434       False
67533     False
59560     False
61577     False
83933     False
5477      False
20342     False
28764     False
126461    False
Name: runtime_minutes, dtype: bool
```

In [34]:
```
long_movies = top_20_movies_runtime_minutes[top_20_movies_runtime_minutes['runtime_

print(long_movies['primary_title'])
```

```
6067                Mysteries of Lisbon
56700                        Coriolanus
73162     M.S. Dhoni: The Untold Story
7292                             Aurora
100713                     Another Year
545             The Wolf of Wall Street
34528         Blue Is the Warmest Color
Name: primary_title, dtype: object
```

In [35]:
```
plt.figure(figsize=(9, 5))
plt.hist(top_20_movies_runtime_minutes['runtime_minutes'], bins=20, color='skyblue
plt.title('Distribution of Movie Runtimes')
plt.xlabel('Runtime (minutes)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [36]:
```
### Selecting the year the highest numvotes:

top_20_movies_numvotes = clean_movies.nlargest(20,'numvotes')

top_20_movies_numvotes.head()
```

Out[36]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	avera
7066	tt1375666	Inception	Inception	2010	148.00	Action,Adventure,Sci-Fi	
6900	tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.00	Action,Thriller	
311	tt0816692	Interstellar	Interstellar	2014	169.00	Adventure,Drama,Sci-Fi	
20342	tt1853728	Django Unchained	Django Unchained	2012	165.00	Drama,Western	
545	tt0993846	The Wolf of Wall Street	The Wolf of Wall Street	2013	180.00	Biography,Crime,Drama	

In [37]:
```
top_20_movies_numvotes.columns
```

Out[37]:
```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'averagerating', 'numvotes', 'studio',
       'domestic_gross', 'foreign_gross', 'year'],
      dtype='object')
```

In [38]:
```
clean_movies.sample()
```

Out[38]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres	avera
78956	tt4530422	Overlord	Overlord	2018	110.00	Action,Adventure,Horror	

In [45]:
```
clean_movies['foreign_gross'].dtype
```

Out[45]:
```
dtype('O')
```

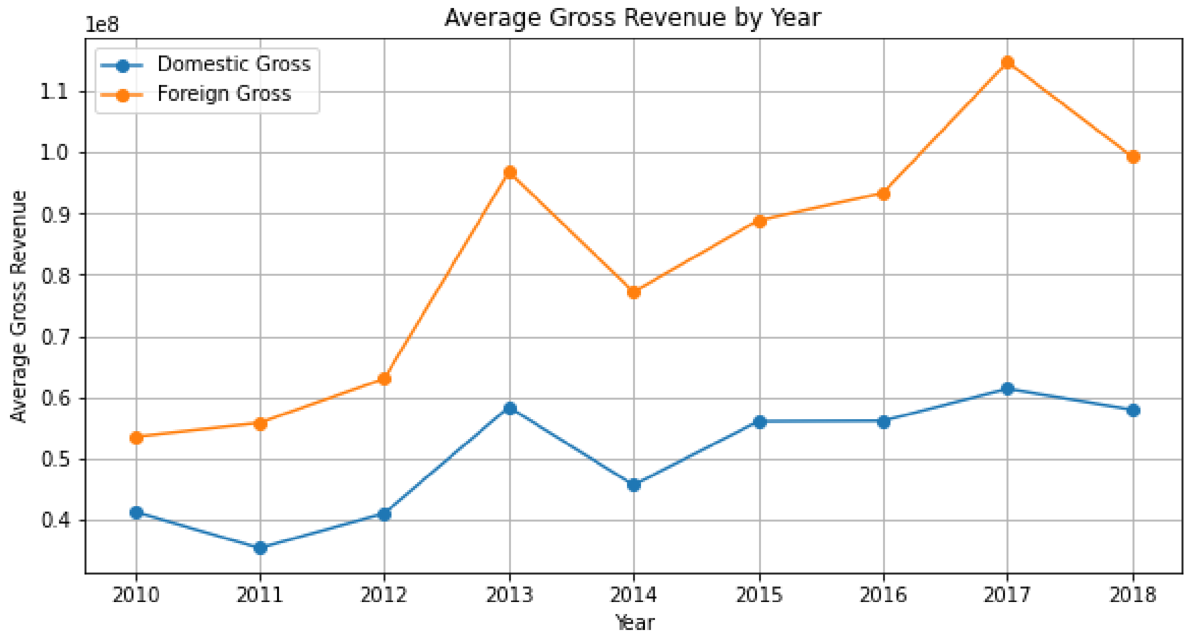
In [56]:
```
# Convert 'foreign_gross' column to numeric dtype
clean_movies['foreign_gross'] = pd.to_numeric(clean_movies['foreign_gross'], error:

# Drop rows with missing or invalid foreign_gross values
clean_movies = clean_movies.dropna(subset=['foreign_gross'])

# Select top 20 movies based on foreign_gross for detailed analysis
top_20_movies_foreign_gross = clean_movies.nlargest(20, 'foreign_gross')

# Display the first few rows of the resulting DataFrame
print(top_20_movies_foreign_gross.head())
```

```
          tconst                    primary_title  \
39010  tt2395427          Avengers: Age of Ultron
84415  tt4881806  Jurassic World: Fallen Kingdom
6647   tt1323045                           Frozen
10824  tt1611845                           Frozen
35107  tt2294629                           Frozen

                       original_title  start_year  runtime_minutes  \
39010         Avengers: Age of Ultron        2015           141.00
84415  Jurassic World: Fallen Kingdom        2018           128.00
6647                           Frozen        2010            93.00
10824              Wai nei chung ching        2010            92.00
35107                           Frozen        2013           102.00

                           genres  averagerating  numvotes studio  \
39010     Action,Adventure,Sci-Fi           7.30 665594.00     BV
84415     Action,Adventure,Sci-Fi           6.20 219125.00   Uni.
6647        Adventure,Drama,Sport           6.20  62311.00     BV
10824             Fantasy,Romance           5.40     75.00     BV
35107  Adventure,Animation,Comedy           7.50 516998.00     BV

       domestic_gross  foreign_gross    year
39010    459000000.00   946400000.00 2015.00
84415    417700000.00   891800000.00 2018.00
6647     400700000.00   875700000.00 2013.00
10824    400700000.00   875700000.00 2013.00
35107    400700000.00   875700000.00 2013.00

<ipython-input-56-8832bc76ed8c>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.
org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  clean_movies['foreign_gross'] = pd.to_numeric(clean_movies['foreign_gross'], er
rors='coerce')
```

Calculate average gross revenue by year

In [62]:

```
# Calculate average gross revenue by year
average_revenue_by_year = clean_movies .groupby('year')[['domestic_gross', 'foreig
# Plot line chart
plt.figure(figsize=(10, 5)) b
plt.plot(average_revenue_by_year.index, average_revenue_by_year['domestic_gross'],
plt.plot(average_revenue_by_year.index, average_revenue_by_year['foreign_gross'],
plt.title('Average Gross Revenue by Year')
plt.xlabel('Year')
plt.ylabel('Average Gross Revenue')
plt.legend()
plt.grid(True)
plt.show()
```



In [42]:
```
top_20_movies_per_year = pd.DataFrame()
for year, group in top_20_movies_numvotes.groupby("year"):
    top_20_movies = group.nlargest(20, "numvotes")
    top_20_movies_per_year = pd.concat([top_20_movies_per_year, top_20_movies])

print(top_20_movies_per_year)
```

```
         tconst                        primary_title  \
7066   tt1375666                           Inception
5640   tt1130884                      Shutter Island
104    tt0435761                         Toy Story 3
281    tt0800369                                Thor
141    tt0458339   Captain America: The First Avenger
6900   tt1345836               The Dark Knight Rises
20342  tt1853728                    Django Unchained
7212   tt1392170                    The Hunger Games
434    tt0903624    The Hobbit: An Unexpected Journey
545    tt0993846              The Wolf of Wall Street
7779   tt1454468                             Gravity
6453   tt1300854                          Iron Man 3
311    tt0816692                        Interstellar
25432  tt2015381              Guardians of the Galaxy
34178  tt2267998                           Gone Girl
19899  tt1843866  Captain America: The Winter Soldier
7213   tt1392190                  Mad Max: Fury Road
63705  tt3659388                          The Martian
7543   tt1431045                             Deadpool
72821  tt4154756               Avengers: Infinity War

                            original_title  start_year  runtime_minutes  \
7066                             Inception        2010           148.00
5640                        Shutter Island        2010           138.00
104                            Toy Story 3        2010           103.00
281                                   Thor        2011           115.00
141     Captain America: The First Avenger        2011           124.00
6900                 The Dark Knight Rises        2012           164.00
20342                     Django Unchained        2012           165.00
7212                      The Hunger Games        2012           142.00
434      The Hobbit: An Unexpected Journey        2012           169.00
545                The Wolf of Wall Street        2013           180.00
7779                               Gravity        2013            91.00
6453                        Iron Man Three        2013           130.00
311                           Interstellar        2014           169.00
25432              Guardians of the Galaxy        2014           121.00
34178                            Gone Girl        2014           149.00
19899  Captain America: The Winter Soldier        2014           136.00
7213                    Mad Max: Fury Road        2015           120.00
63705                          The Martian        2015           144.00
7543                              Deadpool        2016           108.00
72821               Avengers: Infinity War        2018           149.00

                          genres  averagerating   numvotes   studio  \
7066       Action,Adventure,Sci-Fi           8.80 1841066.00       WB
5640              Mystery,Thriller           8.10 1005960.00     Par.
104    Adventure,Animation,Comedy           8.30  682218.00       BV
281      Action,Adventure,Fantasy           7.00  683264.00     Par.
141       Action,Adventure,Sci-Fi           6.90  668137.00     Par.
6900              Action,Thriller           8.40 1387769.00       WB
20342              Drama,Western           8.40 1211405.00    Wein.
7212      Action,Adventure,Sci-Fi           7.20  795227.00      LGF
434      Adventure,Family,Fantasy           7.90  719629.00  WB (NL)
545         Biography,Crime,Drama           8.20 1035358.00     Par.
7779         Drama,Sci-Fi,Thriller           7.70  710018.00       WB
6453      Action,Adventure,Sci-Fi           7.20  692794.00       BV
311         Adventure,Drama,Sci-Fi           8.60 1299334.00     Par.
```

```
25432     Action,Adventure,Comedy          8.10  948394.00       BV
34178      Drama,Mystery,Thriller          8.10  761592.00      Fox
19899     Action,Adventure,Sci-Fi          7.80  666252.00       BV
7213      Action,Adventure,Sci-Fi          8.10  780910.00       WB
63705      Adventure,Drama,Sci-Fi          8.00  680116.00      Fox
7543      Action,Adventure,Comedy          8.00  820847.00      Fox
72821     Action,Adventure,Sci-Fi          8.50  670926.00       BV

       domestic_gross foreign_gross    year
7066     292600000.00     535700000 2010.00
5640     128000000.00     166800000 2010.00
104      415000000.00     652000000 2010.00
281      181000000.00     268300000 2011.00
141      176700000.00     193900000 2011.00
6900     448100000.00     636800000 2012.00
20342    162800000.00     262600000 2012.00
7212     408000000.00     286400000 2012.00
434      303000000.00     718100000 2012.00
545      116900000.00     275100000 2013.00
7779     274100000.00     449100000 2013.00
6453     409000000.00     805800000 2013.00
311      188000000.00     489400000 2014.00
25432    333200000.00     440200000 2014.00
34178    167800000.00     201600000 2014.00
19899    259800000.00     454500000 2014.00
7213     153600000.00     224800000 2015.00
63705    228400000.00     401700000 2015.00
7543     363100000.00     420000000 2016.00
72821    678800000.00       1,369.5 2018.00
```

In [55]:
```
sns.barplot( x = "numvotes" , y= "primary_title" , data = top_20_movies_numvotes)
plt.title("Top 20 Highly Voted Movies")
plt.show()
```