Project: Investigate TMDB (The Movies Database) Dataset

Table of Contents

Introduction

Hello! This is my first project exploring a dataset, thanks to FWD program and Udacity Nanodegree.

After going through the provided datasets, I found myself attracted to the TMDB dataset due to my passion for watching movies as well as watching a huge number of movies that inspired me throug this project.

Let's have a small idea about this dataset:

This dataset was collected by Kaggle through TMDB Website with more than 10,000 movies as rows(entries) and more than 20 columns of different types of information about these movies

I have already gone through the dataset and checked other projects for inspiration and made up my mind about how we are going to explore it;

The dataset provides for each movie: popularity, revenue, budget, list of cast, director, votes, genre, ...etc

So, I have devided my questions to 4 categories:

1: **Comparisons**: Here we are going to compare the most and least value for some properties and see which movies

2: **Popularity**: Here we are going to see what properties affect popularity of a movie

3: **Profitability**: Here we are going to see what properties affect profitability of a movie

4: **Frequency**: finally we are going to check the frequency of an actor, director, genre,...etc

So! are you ready to explore this dataset with me and see what interesting findings we could extract? Let's go!

First, Let's import the main libraries (pandas, numpy, and matplotlib for plotting)

In [1]:

```
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

Data Wrangling

General Properties

Second, let's load our dataset here and explore its properties

In [2]:

```
df = pd.read_csv('tmdb-movies.csv')
df.head()
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.j
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.ma
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.cor

	id	imdb_id	popularity	budget	revenue	original_title	cast	
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://w

5 rows × 21 columns

So, the dataset has 21 columns, and we can see that some columns such as "cast" having a "|" between each name, which is not best for data manipulation

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

It shows up here that there are 10866 movies, however not all the columns are filled up, there is much missing data

In [4]:

```
df.dtypes
```

Out[4]:

```
id                        int64
imdb_id                  object
popularity              float64
budget                    int64
revenue                   int64
original_title           object
cast                     object
homepage                 object
director                 object
tagline                  object
```

```
keywords                 object
overview                 object
runtime                   int64
genres                   object
production_companies     object
release_date             object
vote_count                int64
vote_average            float64
release_year              int64
budget_adj              float64
revenue_adj             float64
dtype: object
```

Up here we checked the data type of each column, I can see that these types are so far so good, unless we find out any issue later on, we can then manipulate the data types to our favor. Only, the release date column need to be "datetime" type

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

After checking the columns, I decided to drop the columns that I don't need in my analysis

Data Cleaning

- Of course I am not going to use 'ID' or "IMDB ID' in my analysis, so I am going to drop them

- Also, I dont need 'Home Page', 'Tagline', 'Keywords' and 'Overview'

- There are two types of budget and revenue; the real numbers ('budget', 'revenue') columns, and ('budget_adj', 'revenue_adj') columns which according to the documentation they are the budget and revenue but accounting for inflation over the years. So, I thought that *_adj columns will give more accurate analysis, so I am going to drop 'budget' and 'revenue' columns

- I am going to count on the release year in my analysis so I am not going to use the release date, will drop that as well

- Votes, and production company columns are irrelevant to my analysis, so we will drop them

In [6]:
```
drop_columns = ['imdb_id','homepage','tagline','keywords','overview','budget','revenue','v
df.drop(drop_columns,axis=1,inplace=True)
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      10866 non-null  float64
 1   original_title  10866 non-null  object 
 2   cast            10790 non-null  object 
 3   director        10822 non-null  object 
 4   runtime         10866 non-null  int64  
 5   genres          10843 non-null  object 
 6   release_year    10866 non-null  int64  
 7   budget_adj      10866 non-null  float64
 8   revenue_adj     10866 non-null  float64
dtypes: float64(3), int64(2), object(4)
memory usage: 764.1+ KB
```

So, after dropping the unwanted columns we are left with these 8 columns, with correct data types

Let's check if the numeric columns contain NaN values masked as zeros

In [8]:
```
df.query('budget_adj == 0')
```

Out[8]:

	popularity	original_title	cast	director	runtime	genres	release_year	b
30	3.927333	Mr. Holmes	Ian McKellen|Milo Parker|Laura Linney|Hattie M...	Bill Condon	103	Mystery|Drama	2015	
36	3.358321	Solace	Abbie Cornish|Jeffrey Dean Morgan|Colin Farrel...	Afonso Poyart	101	Crime|Drama|Mystery	2015	
72	2.272044	Beyond the Reach	Michael Douglas|Jeremy Irvine|Hanna Mangan Law...	Jean-Baptiste LÃ©onetti	95	Thriller	2015	
74	2.165433	Mythica: The Darkspore	Melanie Stone|Kevin Sorbo|Adam Johnson|Jake St...	Anne K. Black	108	Action|Adventure|Fantasy	2015	
75	2.141506	Me and Earl and the Dying Girl	Thomas Mann|RJ Cyler|Olivia Cooke|Connie Britt...	Alfonso Gomez-Rejon	105	Comedy|Drama	2015	
...	...	...	...	...	...	...	...	
10860	0.087034	Carry On Screaming!	Kenneth Williams|Jim Dale|Harry H. Corbett|Joa...	Gerald Thomas	87	Comedy	1966	
10861	0.080598	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	Bruce Brown	95	Documentary	1966	
10862	0.065543	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	John Frankenheimer	176	Action|Adventure|Drama	1966	

	popularity	original_title	cast	director	runtime	genres	release_year	b
10863	0.065141	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	Eldar Ryazanov	94	Mystery|Comedy	1966	
10864	0.064317	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	Woody Allen	80	Action|Comedy	1966	

5696 rows × 9 columns

OOPS, as expected, it seems to be there are alot of zeros embedded in these columns, so lets replace those zeros with NaN values

In [9]:
```
zeros_list = ['revenue_adj','budget_adj','runtime']
df[zeros_list] = df[zeros_list].replace(0,np.nan)
```

Let's replace the NaN values with the mean of each column

In [10]:
```
df.budget_adj.fillna(df.budget_adj.mean(), inplace=True)
df.revenue_adj.fillna(df.revenue_adj.mean(), inplace=True)
df.runtime.fillna(df.runtime.mean(), inplace=True)
```

Let's drop duplicates

In [11]:
```
df.drop_duplicates(inplace=True)
```

And the rows with NaN values

In [12]:
```
df.dropna(inplace=True)
```

We found earlier that the string type columns are separated with "|" so let's separate them into sublists

Remove " | " from genres and cast columns

In [13]:
```
df.genres = df.genres.str.split(pat='|')
df.cast = df.cast.str.split(pat='|')
```

Let's check the final shape of the dataframe after cleaning

In [14]:
```
df.head(1)
```

Out[14]:

	popularity	original_title	cast	director	runtime	genres	release_year	budget_adj	revenue_adj

	popularity	original_title	cast	director	runtime	genres	release_year	budget_adj	revenue_adj
0	32.985763	Jurassic World	[Chris Pratt, Bryce Dallas Howard, Irrfan Khan...	Colin Trevorrow	124.0	[Action, Adventure, Science Fiction, Thriller]	2015	1.379999e+08	1.392446e+09

In [15]:

```
df.info()
```

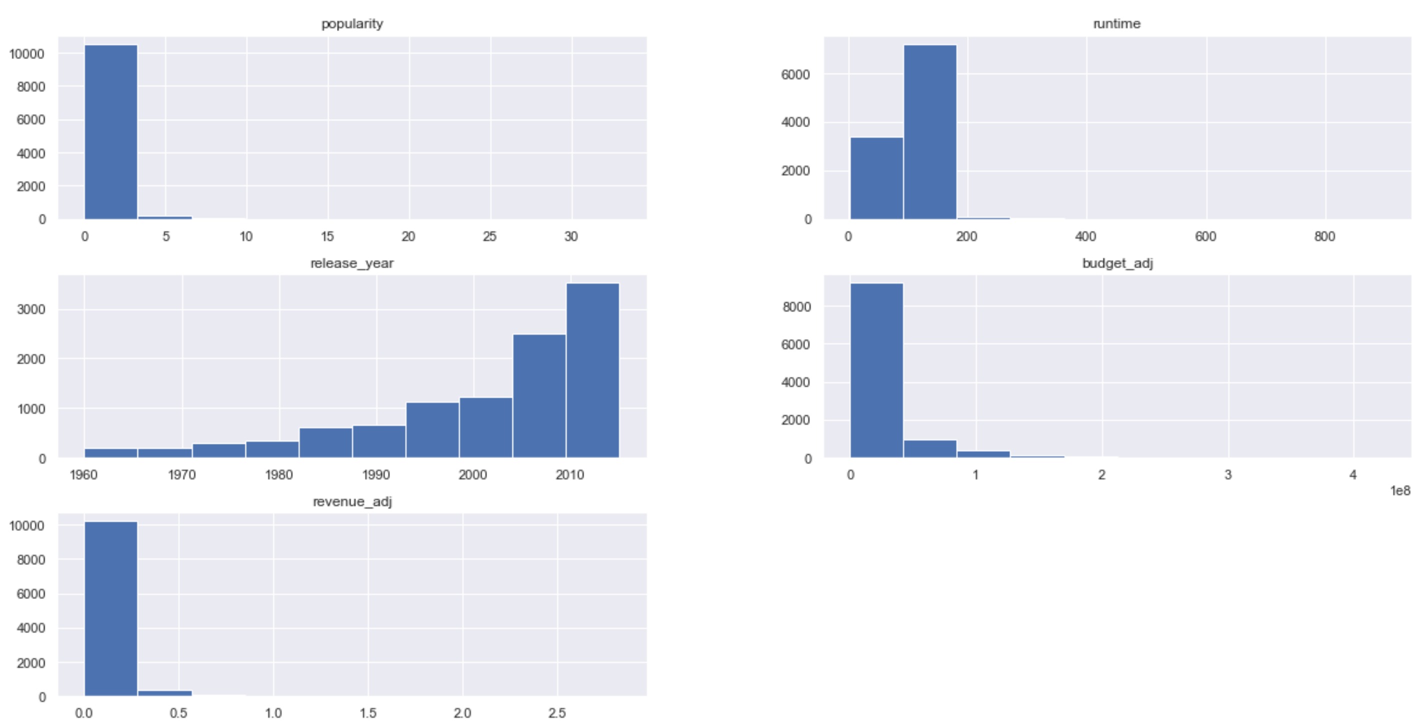
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10731 entries, 0 to 10865
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      10731 non-null  float64
 1   original_title  10731 non-null  object 
 2   cast            10731 non-null  object 
 3   director        10731 non-null  object 
 4   runtime         10731 non-null  float64
 5   genres          10731 non-null  object 
 6   release_year    10731 non-null  int64  
 7   budget_adj      10731 non-null  float64
 8   revenue_adj     10731 non-null  float64
dtypes: float64(4), int64(1), object(4)
memory usage: 838.4+ KB
```

Great! Now our dataframe has 10731 rows and 8 usable columns, without any NaN or duplicate values, and correct data types.

Let's do a quick check with a histogram plot to all numeric columns

In [16]:

```
df.hist(figsize=(20,10));
```



Our Columns are in good shape, So let's go to the next step after cleaning our data frame, which is the most interesting step, <u>The **EDA** step</u>

Exploratory Data Analysis

Comaprisons

As mentioned in the introduction section, this category of exploration aims to compare the movies in terms of highes and lowest

Define a function to take a column as an argument and return the most and least values in the same column

In [17]:
```
def compare(column):
    return df.loc[[df[column].idxmax(),df[column].idxmin()]].T
```

Q: Which movies had the most and least budget?

In [18]:
```
compare('budget_adj')
```

Out[18]:

	2244	1151
popularity	0.25054	0.177102
original_title	The Warrior's Way	Fear Clinic
cast	[Kate Bosworth, Jang Dong-gun, Geoffrey Rush, ...	[Thomas Dekker, Robert Englund, Cleopatra Cole...
director	Sngmoo Lee	Robert Hall
runtime	100.0	95.0
genres	[Adventure, Fantasy, Action, Western, Thriller]	[Horror]
release_year	2010	2014
budget_adj	425000000.0	0.921091
revenue_adj	11087569.0	115077354.868005

"The Warrior's Way" had the most budget with 425 million dollars! however it had lower revenue than was spent

Also "Fear Clinic" had the least budget which is less than 1 dollar!!!!!!!!!!

Q: Which movies had the most and least revenue?

In [19]:
```
compare('revenue_adj')
```

Out[19]:

	1386	5067
popularity	9.432768	0.462609
original_title	Avatar	Shattered Glass
cast	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[Hayden Christensen, Peter Sarsgaard, ChloÃ« S...
director	James Cameron	Billy Ray
runtime	162.0	94.0
genres	[Action, Adventure, Fantasy, Science Fiction]	[Drama, History]
release_year	2009	2003
budget_adj	240886902.887613	7112115.868695
revenue_adj	2827123750.41189	2.370705

I knew it!, the movie "Avatar" made the most revenue of all the movies with 2.9 **Billion** dollars! WOW

Can you believe it! There is a movie that made only 2.3 dollars as a revenue!

Q: Which movies had the most and least runtime?

In [20]:
```
compare('runtime')
```

Out[20]:

	3894	1112
popularity	0.006925	0.202776
original_title	The Story of Film: An Odyssey	Batman: Strange Days
cast	[Mark Cousins, Jean-Michel Frodon, Cari Beauch...	[Kevin Conroy, Brian George, Tara Strong]
director	Mark Cousins	Bruce Timm
runtime	900.0	3.0
genres	[Documentary]	[Action, Animation]
release_year	2011	2014
budget_adj	36887736.695452	36887736.695452
revenue_adj	115077354.868005	115077354.868005

Interesting! There is a movie that is 900 minutes(15 hours) long!!

Q: Whcih movies are most and least popular?

In [21]:
```
compare('popularity')
```

Out[21]:

	0	9977
popularity	32.985763	0.000188
original_title	Jurassic World	The Hospital

	0	9977
cast	[Chris Pratt, Bryce Dallas Howard, Irrfan Khan...	[George C. Scott, Diana Rigg, Richard Dysart, ...
director	Colin Trevorrow	Arthur Hiller
runtime	124.0	103.0
genres	[Action, Adventure, Science Fiction, Thriller]	[Mystery, Comedy, Drama]
release_year	2015	1971
budget_adj	137999939.280026	36887736.695452
revenue_adj	1392445892.5238	115077354.868005

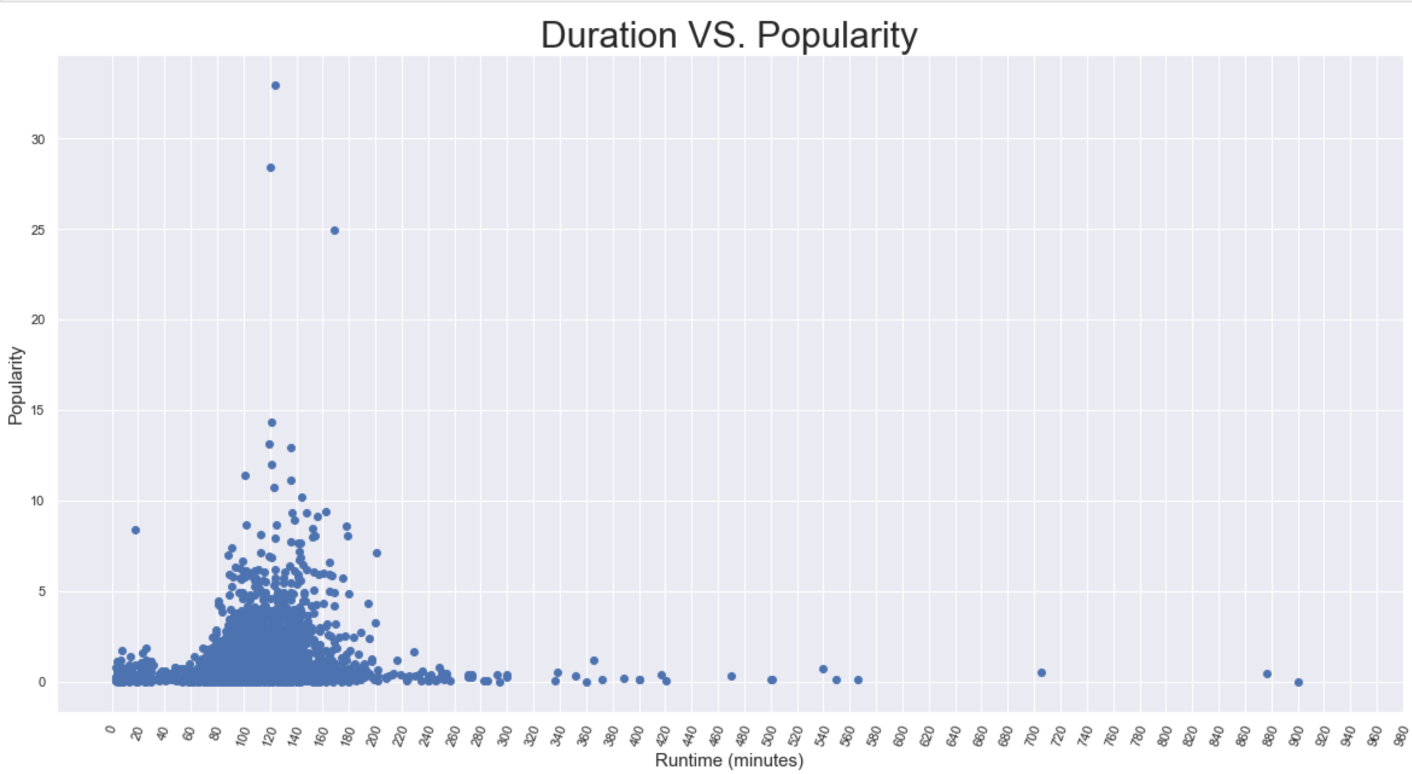
Popularity

In this category we are going to explore our dataset in terms of movies' popularity

Q: How does duration affect popularity

In [22]:

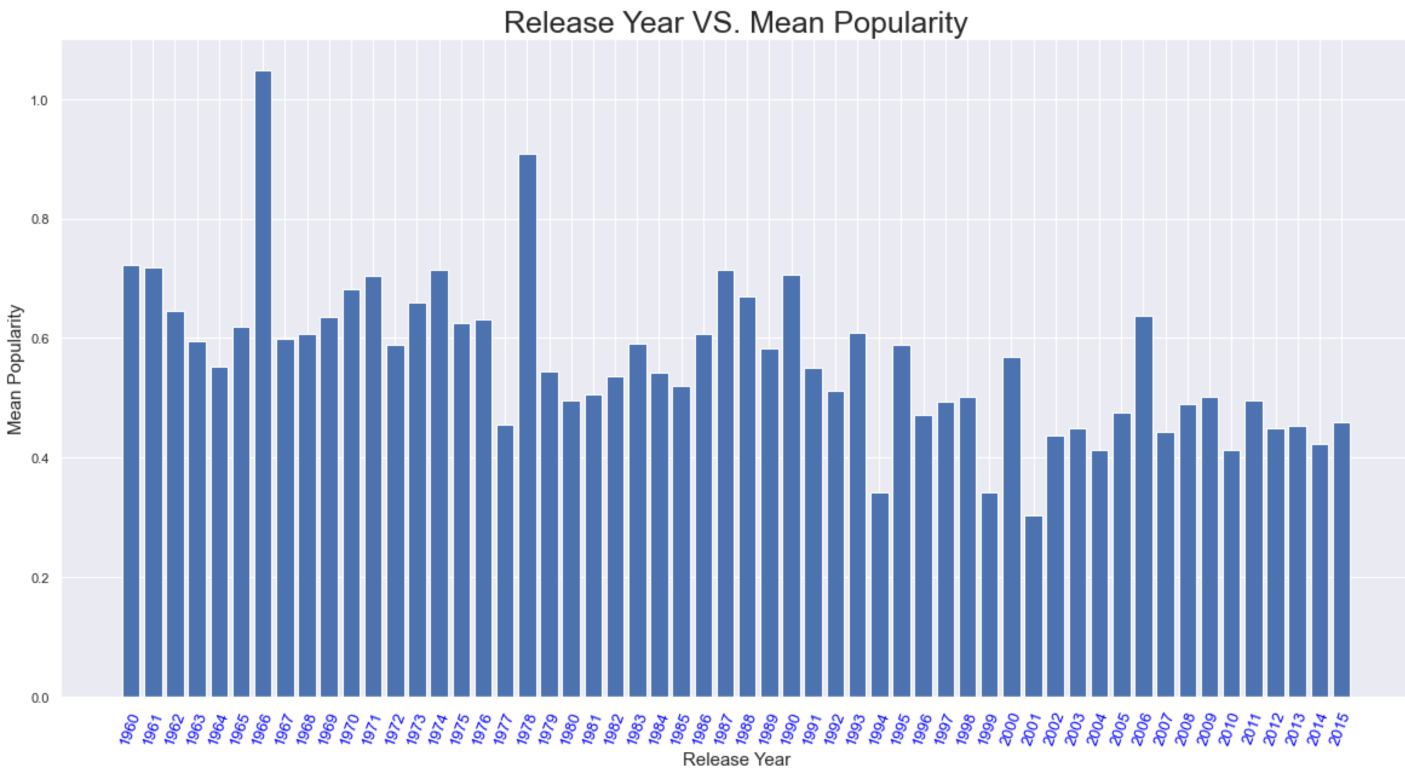
```
ticks = np.arange(0,1000,20)
f, ax = plt.subplots(figsize=(20,10))
plt.scatter(df.runtime, df.popularity)
plt.title('Duration VS. Popularity', fontsize=30)
plt.xlabel('Runtime (minutes)', fontsize=15)
plt.ylabel('Popularity',fontsize=15)
plt.xticks(ticks, rotation=70);
```



The scatter plot here shows that the movies with duration range of **80 - 200 minutes** are the most popular movies

Q: What is the average popularity per year?

In [23]:
```
mean_popularity = df.groupby('release_year').popularity.mean()
release_year = df.release_year.unique()
f, ax = plt.subplots(figsize=(20,10))
plt.bar(release_year, mean_popularity)
plt.title('Release Year VS. Mean Popularity', fontsize=25)
plt.xlabel('Release Year', fontsize=15)
plt.ylabel('Mean Popularity', fontsize=15)
plt.xticks(release_year, rotation=70, fontsize=13, color='blue');
```



From the bar plot above we can see that movies produced in the year 1966 are the most popular ones (average popularity)

Q: What are the genres that are most popular?

We will assign only one genre (the first of each sublist or the main genre) to each movie and add it to a new dataframe

copy to a new dataframe

In [24]:
```
df_genres = df.copy()
```

extract genres column to a list

In [25]:
```
list_of_genres = df_genres.genres.tolist()
```

def a function that takes list of lists as argument and return list of the first element of each sublist

In [26]:
```
def extract(lst):
```

```
    return [item[0] for item in lst]
```

convert list of genres to pandas series to replace original genres column

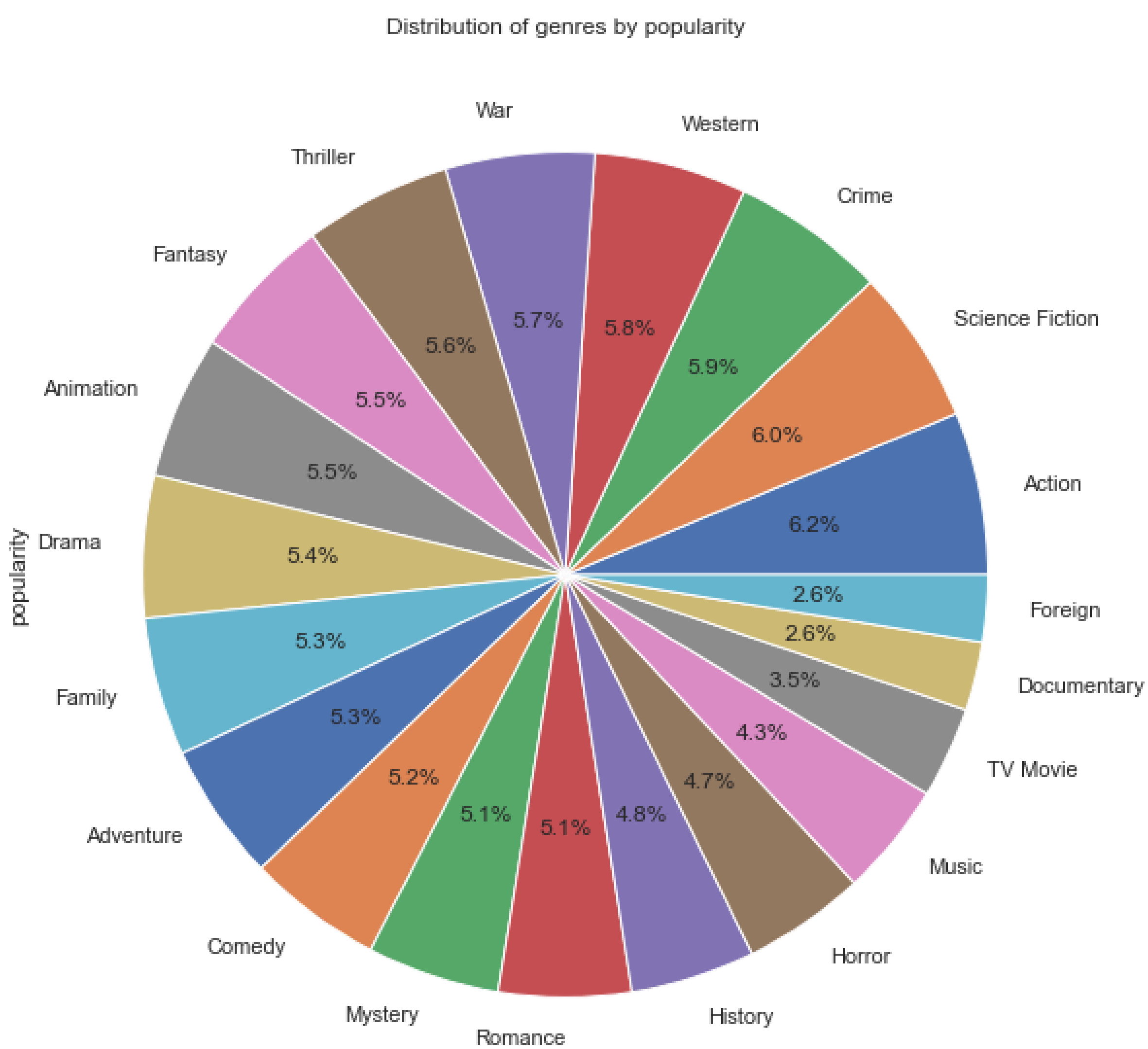
In [27]:
```
list_of_genres = pd.Series(extract(list_of_genres))
```

replace genres column with new genres list

In [28]:
```
df_genres.genres = list_of_genres
```

plot a pie chart for popularity according to genres

In [29]:
```
popularity_by_genres = df_genres.groupby('genres').popularity.mean().sort_values(ascending
popularity_by_genres.plot(kind='pie', figsize=(20,10), autopct = '%1.1f%%', title='Distrib
```



So, the most popular genre is **action** followed by **science fiction**

To my surprise, **Documentaries** comes second to last!

Q: Who is the main actor that most affect popularity?

I did some research on the internet and found that the first actor in 'cast' is the main actor of each movie, so let's edit this column to be of main actors only

Make a new dataframe and extract the first actor of each cast to a new series then replace the cast column with the new one

In [30]:

```
df_cast = df.copy()
cast_list = df_cast.cast.tolist()
cast_list = pd.Series(extract(cast_list))
df_cast.cast = cast_list
```

Let's check how many different actors are there

In [31]:

```
df_cast.cast.nunique()
```

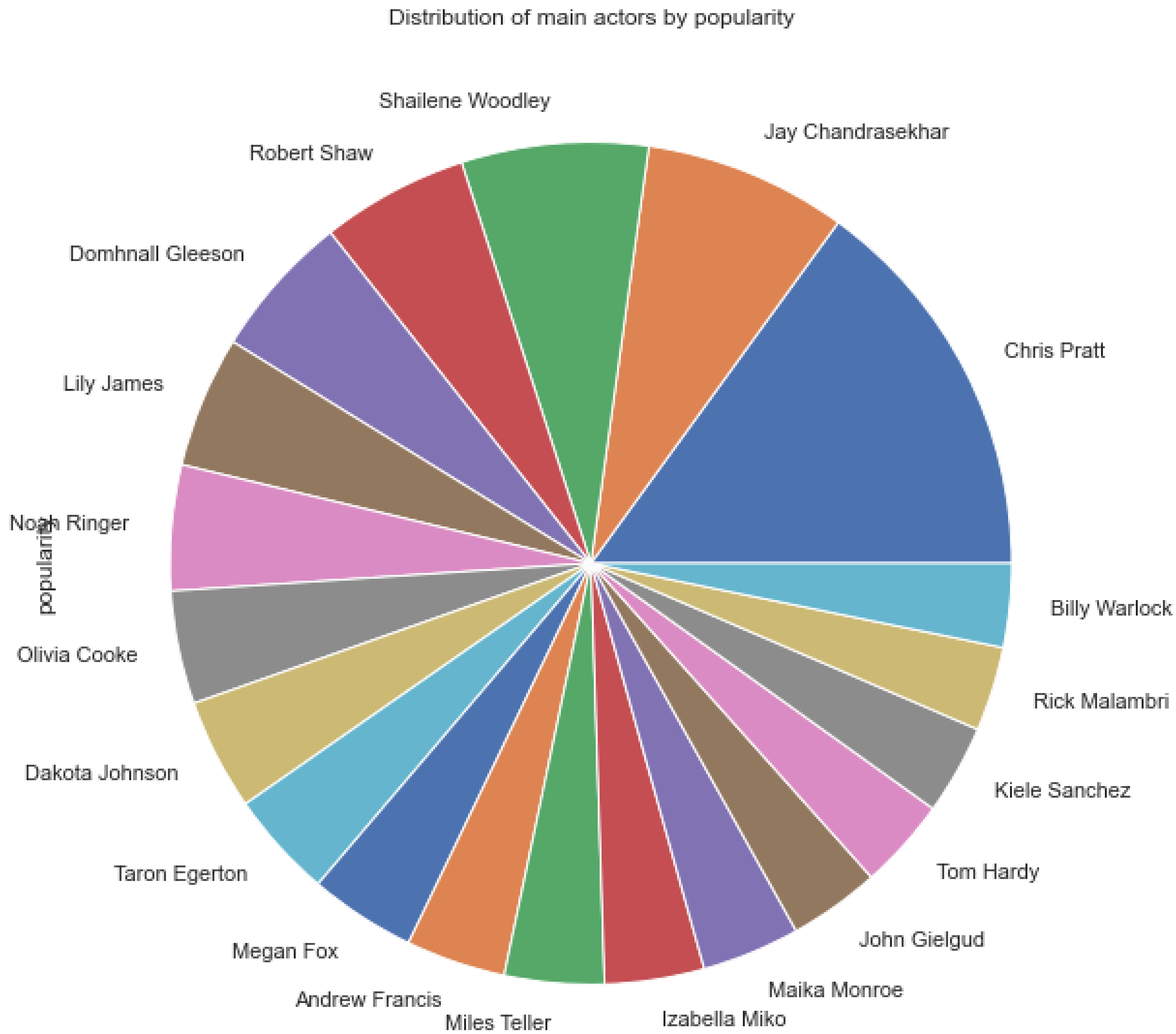
Out[31]:

```
4233
```

So it wouldn't look nice to plot 4233 actors in one plot, instead let's check who are the top 20 actors that affect popularity

In [32]:

```
popularity_by_actor = df_cast.groupby('cast').popularity.mean().sort_values(ascending=Fals
popularity_by_actor.plot(kind='pie', figsize=(20,10), title='Distribution of main actors k
```

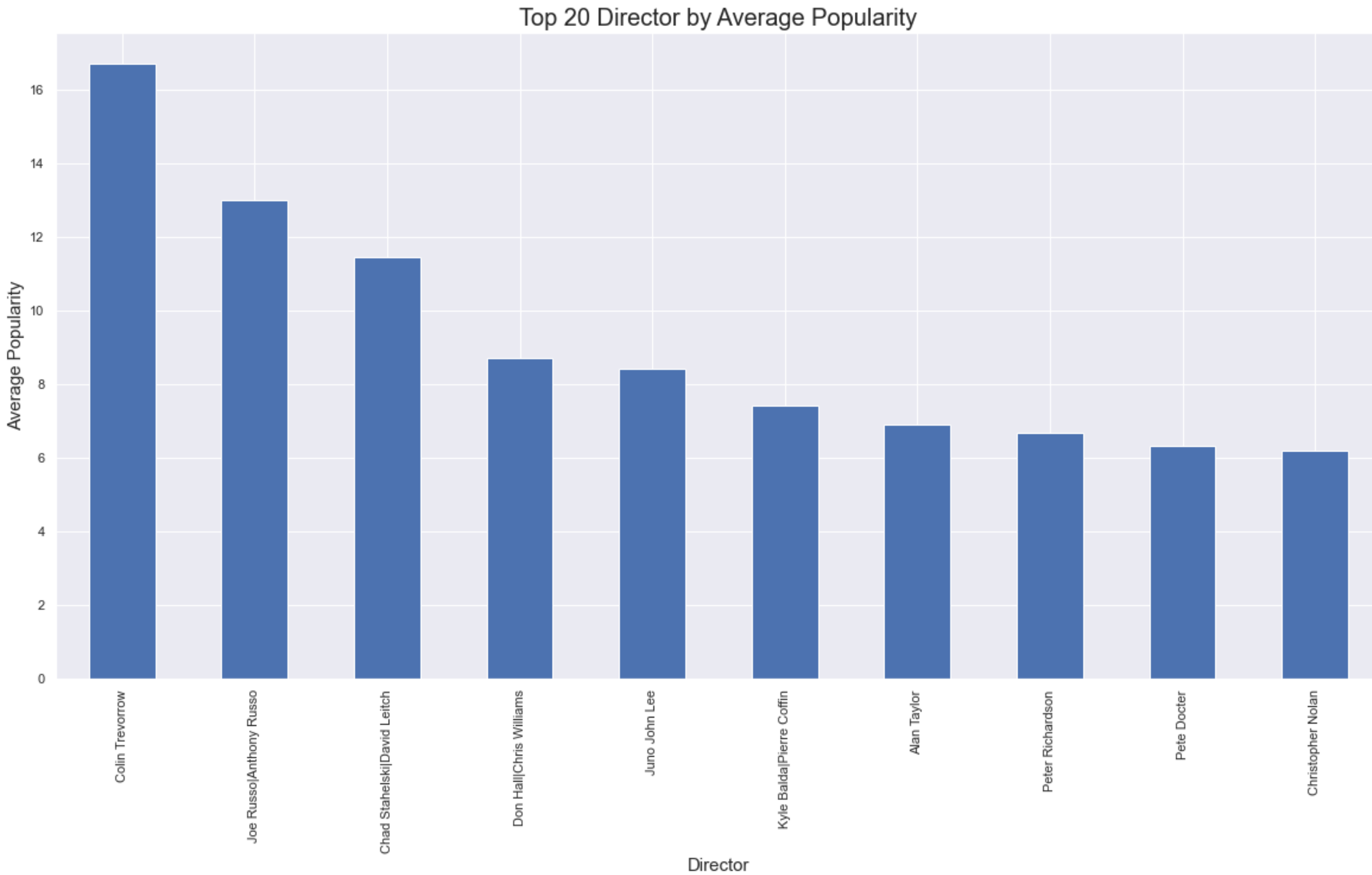


So, it seems Chris Pratt contributes most to popularity -However I think that's because his movie Jurassic World is the most popular among them all-

Q: Which director contributes most to popularity?

In [33]:

```
popularity_by_director = df.groupby('director').popularity.mean().sort_values(ascending=Fa
director_plot = popularity_by_director.plot(kind='bar', figsize=(20,10))
director_plot.set_title('Top 20 Director by Average Popularity', fontsize=20)
director_plot.set_xlabel('Director', fontsize=15)
director_plot.set_ylabel('Average Popularity', fontsize=15)
director_plot;
```



So, the plot shows that **Colin Trevorrow** (Director of Jurassic world saga) movies are the most popular, with **Christopher Nolan** in <u>tenth</u> position!

Profitability

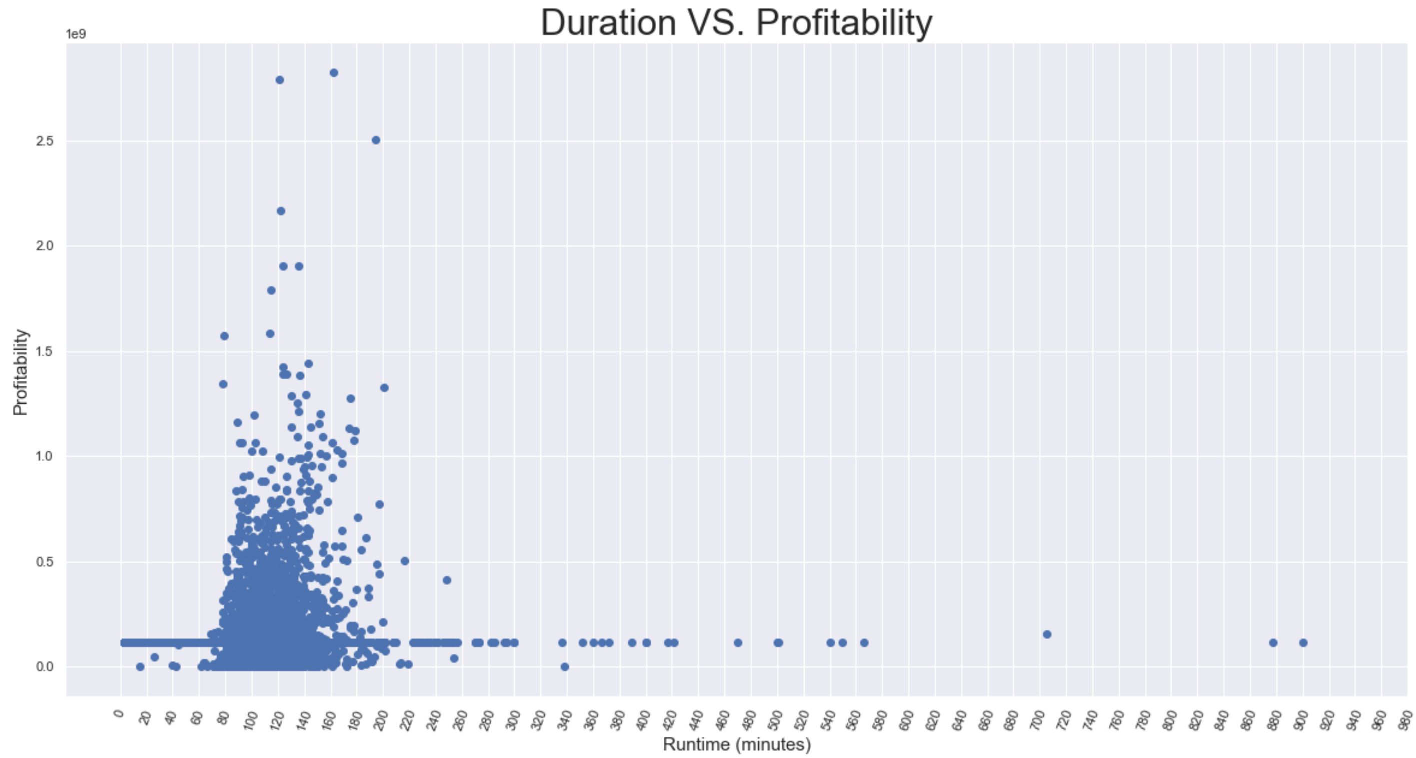
In this category, we will explore our dataset according to profitability

These reults may have a degree of inaccuracy because of the **<u>data limitations</u>** which is due to filling out the NaN values in revenue_adj column with the mean value

Q: How does movie duration affect profitability?

In [34]:

```
f, ax = plt.subplots(figsize=(20,10))
plt.scatter(df.runtime, df.revenue_adj)
plt.title('Duration VS. Profitability', fontsize=30)
plt.xlabel('Runtime (minutes)', fontsize=15)
plt.ylabel('Profitability',fontsize=15)
plt.xticks(ticks, rotation=70);
```

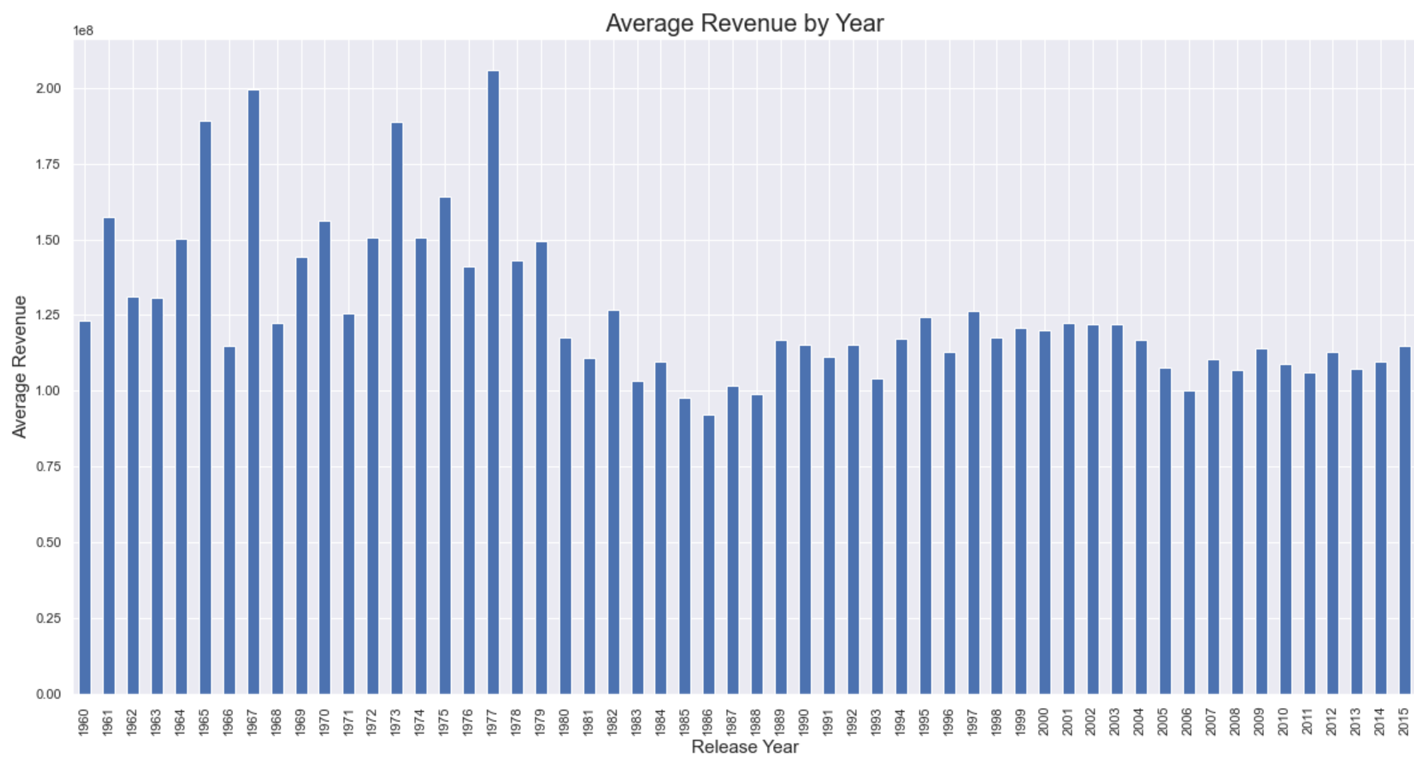


It seems here that the most profitable movies are in the duration range of 80 - 200 minutes, which is nearly the same range of the most popular movies

Q: What is the average revenue by year?

In [35]:

```
average_revenue_by_year = df.groupby('release_year').revenue_adj.mean()
year_plot = average_revenue_by_year.plot(kind='bar', figsize=(20,10))
year_plot.set_title('Average Revenue by Year', fontsize=20)
year_plot.set_xlabel('Release Year', fontsize=15)
year_plot.set_ylabel('Average Revenue', fontsize=15)
year_plot;
```

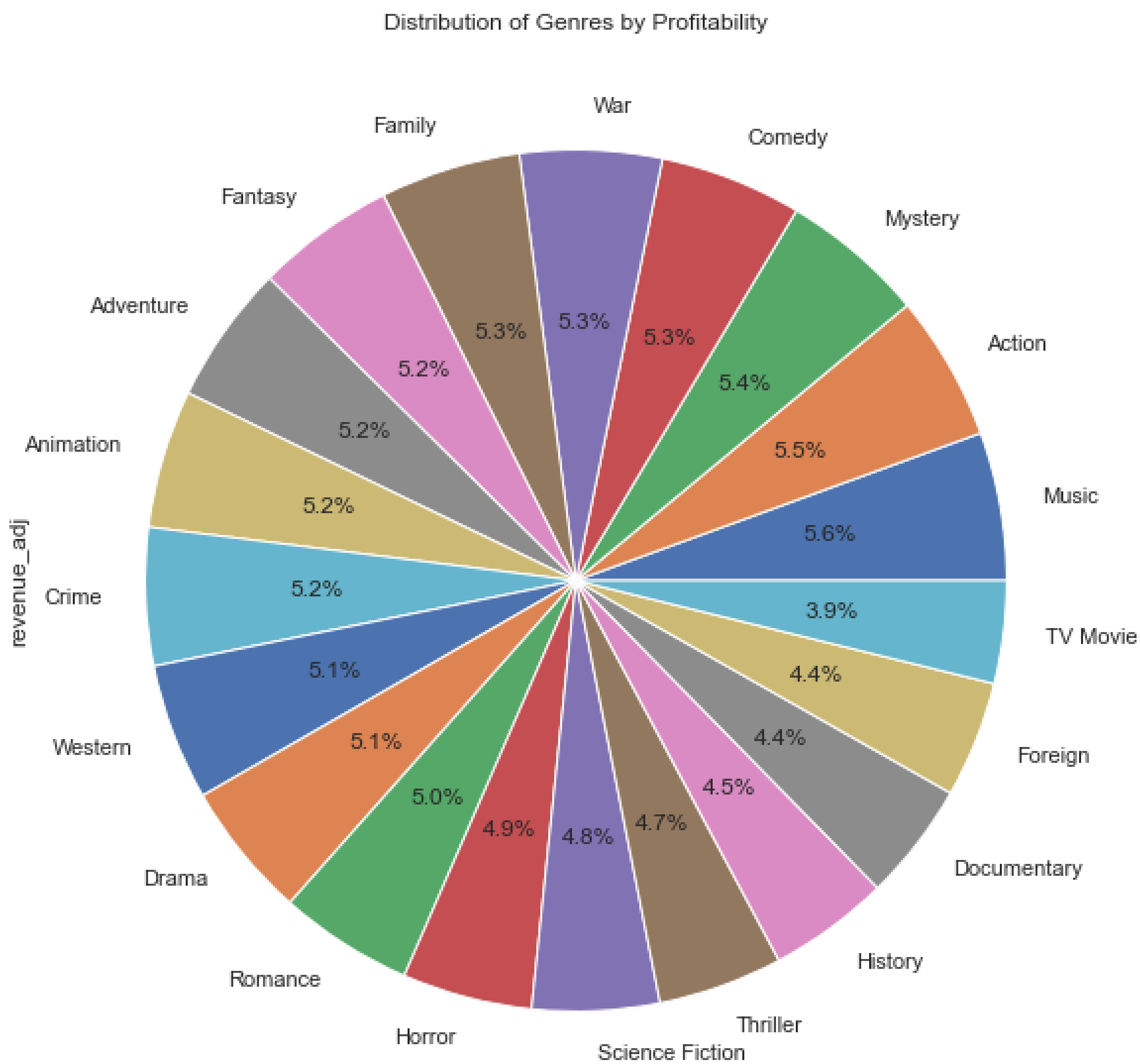


From the bar plot above, it shows that the movies in the years from 1960 - 1980 have made more average revenue than later years, with year **1977** making the most average revenue while **1986** making the least average revenue

Q: Which genres are the most profitable?

In [36]:

```
profitability_by_genre = df_genres.groupby('genres').revenue_adj.mean().sort_values(ascend
profitability_by_genre.plot(kind='pie', figsize=(20,10), autopct = '%1.1f%%', title='Distr
```



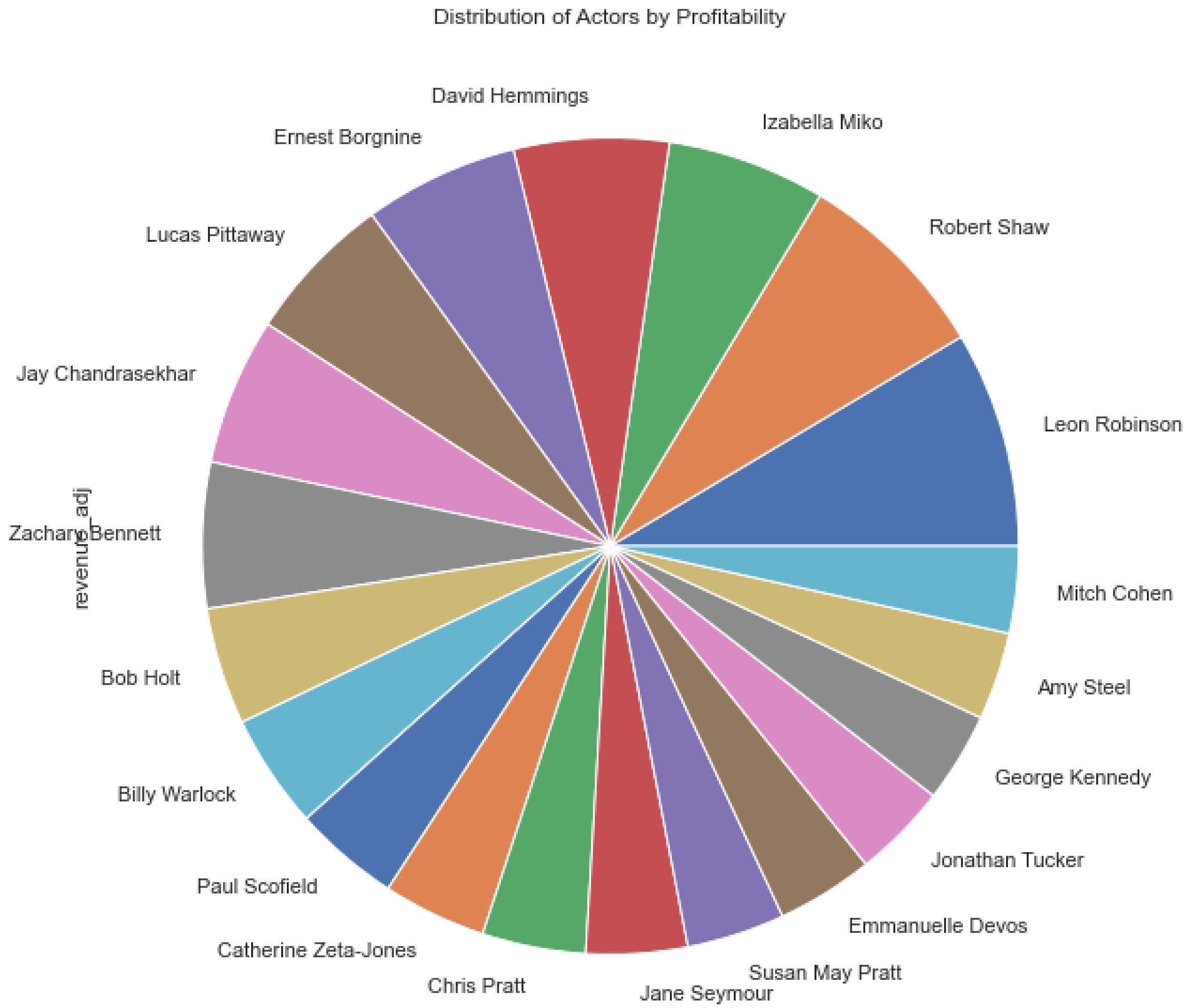
That's interesting! Although Music Genre is one of the least 5 popular genres, it comes **first** in terms of profitability! However it's nearly equal to Action Genre in terms of profitability which is the most popular genre (logic)

Q: Which main actor contributes most to profitability?

Let's plot the top 20 actors affecting profitability

In [37]:

```
profitability_by_actor = df_cast.groupby('cast').revenue_adj.mean().sort_values(ascending=
profitability_by_actor.plot(kind='pie', figsize=(20,10), title='Distribution of Actors by
```



The pie chart shows that **Leon Robinson** is the main actor that contributed most to profitability <u>who is not even in the top 20 most popular main actors!</u>

Frequency

In this last category we are going to explore our dataset with frequency of properties

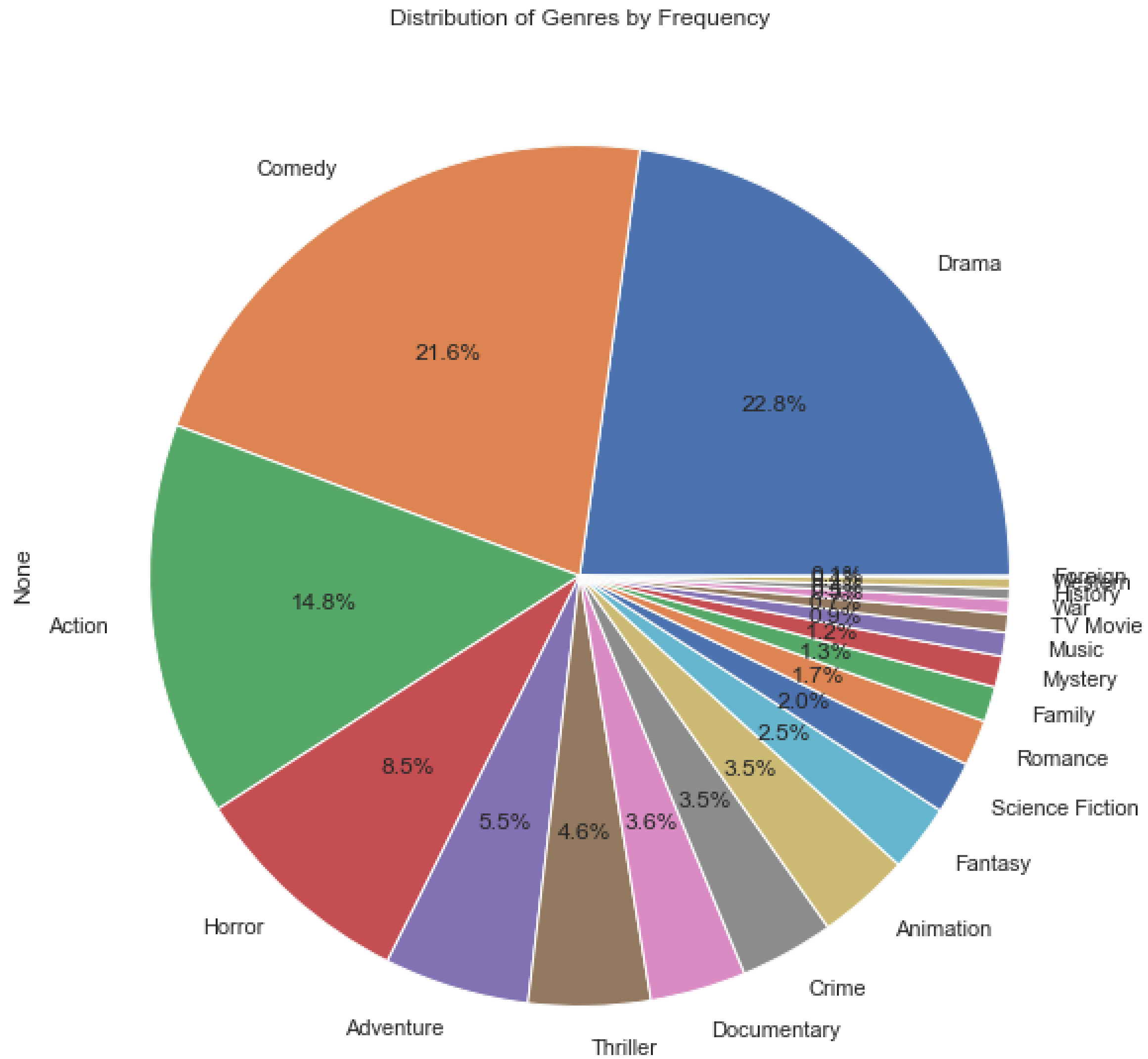
Q: What is the most frequent genre?

In [38]:
```
list_of_genres.mode()[0]
```

Out[38]:
```
'Drama'
```

Generate a pie chart distributing the genres frequency

In [39]:
```
list_of_genres.value_counts().plot(kind='pie', figsize=(20,10), autopct = '%1.1f%%', title
```



Interesting!

Although the most popular genre is (Action), it comes <u>**third**</u> in terms of frequency.

Also, Although (Drama) appears to be the most frequent genre, it comes <u>**ninth**</u> in terms of popularity

Q: Who is the most appearing actor?

Define a function to take a column of list of lists as an argument and returns a pandas series of flat list of all items in sublists

In [40]:

```
def flat(column):
    flat_list=[]
    list_of_sublists = df[column].tolist()
    for sublist in list_of_sublists:
        for item in sublist:
            flat_list.append(item)
    flat_list = pd.Series(flat_list)
    return flat_list
```

Make a flat list of all actors

In [41]:
```
list_of_actors = flat('cast')
```

The most frequent actor

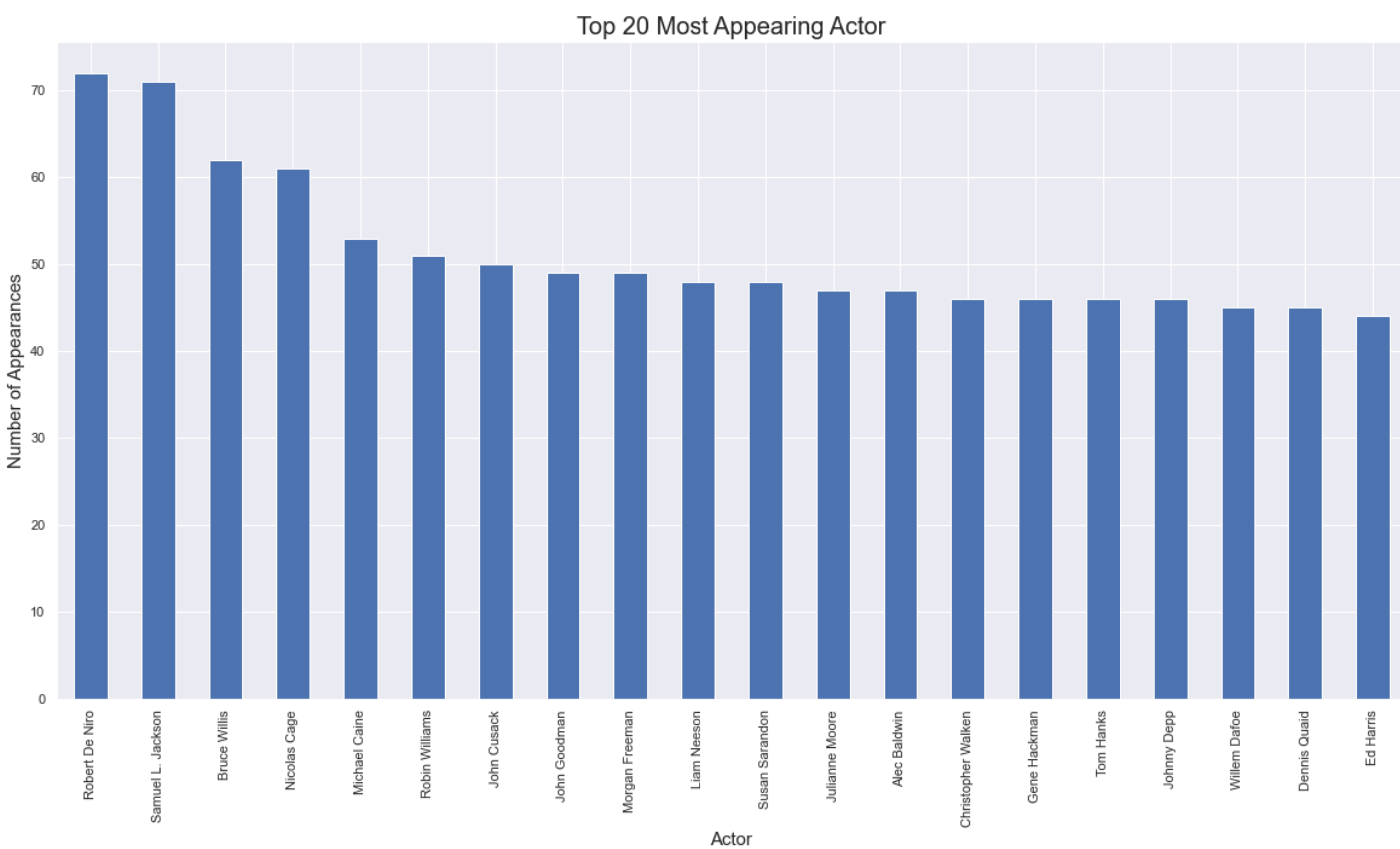
In [42]:
```
list_of_actors.mode()[0]
```

Out[42]:
```
'Robert De Niro'
```

Of course it's **Robert De Niro**!

Plot the top 20 most appearing actors

In [43]:
```
actors_plot = list_of_actors.value_counts()[0:20].plot(kind='bar', figsize=(20,10))
actors_plot.set_title('Top 20 Most Appearing Actor', fontsize=20)
actors_plot.set_xlabel('Actor', fontsize=15)
actors_plot.set_ylabel('Number of Appearances', fontsize=15)
actors_plot;
```



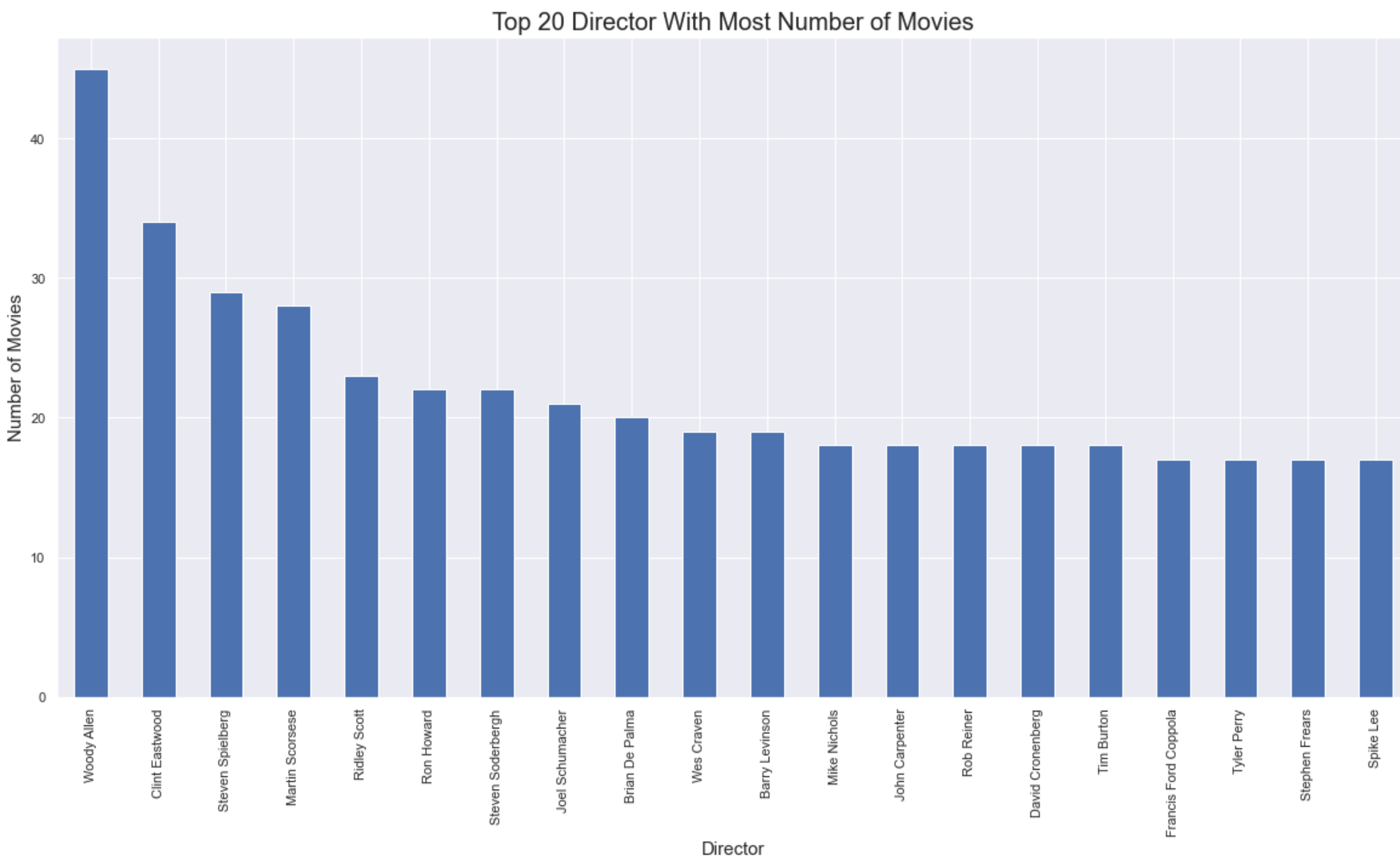
Q: Who is the most frequent director?

In [44]:
```
df.director.mode()[0]
```

Out[44]:
```
'Woody Allen'
```

Plot the top 20 most frequent director

In [45]:
```
director_frequency = df.director.value_counts()[0:20].plot(kind='bar', figsize=(20,10))
director_frequency.set_title('Top 20 Director With Most Number of Movies', fontsize=20)
director_frequency.set_xlabel('Director', fontsize = 15)
director_frequency.set_ylabel('Number of Movies', fontsize = 15);
```



It appears that **Woody Allen** is the director that made the most number of movies with more than 40 movies! However in terms of popularity he is not even in the top 20!

Conclusions

At the end of this project, I realized that it's not necessary if a director for example has produced a large number of movies that he be the most succesful or the most popular, as we saw "Woody Allen" for example; he made more than 40 movies, however he is not the one with the most average popular or profitable movie, It depends on other factors.

Also we can use the analysis of this dataset to conclude that if a production company wants to make a popular and profitable movie at the same time, this movie should fall in the range of **80-200 minutes** duration not more or less. Also the production company would go for Action, Science Fiction movies for popularity while it should go for Musicals for profitability, However "Action" genre appears to be the most succesful genre as it lies within the top in terms of popularity , profitability, and frequency.

I always thought that documentaries are so popular, but after investigating here, it lies within the least popular movies, also they are not so profitable.

Although "Drama" movies are not the most popular nor the most profitable genre, production companies go for it most of the time, maybe there is another factor that drives them, which does not appear here in this dataset! (Interesting, this needs more investigation)

Limitations: Filling the missing values in budget, revenue, and runtime columns with the mean value of each column (more than 4000 entry) could affect the results and provide a degree of inaccuracy

Finally, For me this was a very interesting project, I learned alot going through every part of it, I came to great findings about something that I am passionate for, and also learned that what you think about sth. may not always be right, just go and do some investigations to prove your point of view or maybe you find that your point of view is not the best. The project has inspired me to do more investigations about the things I am passionate for in order to come out with very interesting findings and conclusions.