# UDACITY - DATA ANALYST NANODEGREE

`DATE - MAY 8`

## Project By - Sagar Sahu

`Delhi,India`

# Project: Investigate TMDb Movies Dataset

## Table of Contents

## Introduction :-

For my project, I will be using TMDb Dataset\ TMDb Dataset has 10k+ rows and 21 columns.

In [1]:

```
#Importing the Libraries
import pandas as pd
import numpy as np
import csv
from datetime import datetime
import matplotlib.pyplot as plt
```

## Data Wrangling :-

### General Properties

In [2]:

```
# Load your data and print out a few lines. Perform operations to inspect data
#   types and look for instances of missing or possibly errant data.

#loading the csv file and storing it in the variable "tmbd_data"
df = pd.read_csv('tmdb_movies_data.csv')

#printing first five rows with defined columns of tmdb-movies database
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... | |
| **1** | 76341 | tt1392190 | 28.419936 | 150000000 | 378436354 | Mad Max: Fury Road | Tom Hardy\|Charlize Theron\|Hugh Keays-Byrne\|Nic... | |
| **2** | 262500 | tt2908446 | 13.112507 | 110000000 | 295238201 | Insurgent | Shailene Woodley\|Theo James\|Kate Winslet\|Ansel... | http://wv |
| **3** | 140607 | tt2488496 | 11.173104 | 200000000 | 2068178225 | Star Wars: The Force Awakens | Harrison Ford\|Mark Hamill\|Carrie Fisher\|Adam D... | h |
| **4** | 168259 | tt2820852 | 9.335014 | 190000000 | 1506249360 | Furious 7 | Vin Diesel\|Paul Walker\|Jason Statham\|Michelle ... | |

5 rows × 21 columns

In [3]:

```
# Scanning the dataframe for incorrect datatypes and missing Values.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

In [4]:

```
# Scanning the Dataframe for Null Values
df.isnull().sum()
```

Out[4]:

```
id                        0
imdb_id                  10
popularity                0
budget                    0
revenue                   0
original_title            0
cast                     76
homepage               7930
director                 44
tagline                2824
keywords               1493
overview                  4
runtime                   0
genres                   23
production_companies   1030
release_date              0
vote_count                0
vote_average              0
release_year              0
budget_adj                0
revenue_adj               0
dtype: int64
```

In [5]:

```
# Scanning the dataframe for duplicate values
df.duplicated().any()
```

Out[5]:

```
True
```

In [6]:

```
df.shape
```

Out[6]:

```
(10866, 21)
```

In [7]:

```
# Searching for 0's in dataframe
y = df.query('budget == 0').shape
print('Number of Rows with "0" value in budget column : {}'.format(y[0]))
```

```
Number of Rows with "0" value in budget column : 5696
```

In [8]:

```
# Searching for 0's in dataframe
y = df.query('revenue == 0').shape
print('Number of Rows with "0" value in revenue column : {}'.format(y[0]))
```

```
Number of Rows with "0" value in revenue column : 6016
```

In [9]:

```
# Searching for 0's in dataframe
y = df.query('runtime == 0').shape
print('Number of Rows with "0" value in runtime column : {}'.format(y[0]))
```

```
Number of Rows with "0" value in runtime column : 31
```

## Data Cleaning

```
problem with the Dataframe
```

### 1. Removing Unnecessary Columns

In [10]:

```
#Selecting the columns needed to drop
col_del = ['id','imdb_id','popularity','homepage','keywords','overview','tagline','budget_a

# Droping the columns
df.drop(col_del,axis = 1,inplace = True)

# Checking the columns
df.columns
```

Out[10]:

```
Index(['budget', 'revenue', 'original_title', 'cast', 'director', 'runtime',
       'genres', 'production_companies', 'release_date', 'vote_count',
       'vote_average', 'release_year'],
      dtype='object')
```

### 2. Coverting All Zeroes into Null Values

In [11]:

```
# Selecting the columns
col_del = ['budget','revenue','runtime']

# Replacing the values
df[col_del] = df[col_del].replace(to_replace = 0,value = np.NaN)

# Checking for any 0's
df.query('budget == 0' or 'revenue == 0' or 'runtime == 0')
```

Out[11]:

| budget | revenue | original_title | cast | director | runtime | genres | production_companies | releas |
|---|---|---|---|---|---|---|---|---|

### 3.Drop the Null Values

In [12]:

```
# Droping the Null Values for the selected columns
df.dropna(subset = col_del,inplace = True)

# Droping the Null Values for all the columns
df.dropna(inplace = True)

# Checking for any Null Values
df.isnull().sum()
```

Out[12]:

```
budget                  0
revenue                 0
original_title          0
cast                    0
director                0
runtime                 0
genres                  0
production_companies    0
release_date            0
vote_count              0
vote_average            0
release_year            0
dtype: int64
```

### 4. Duplicate Rows

In [13]:

```
# Droping the Duplicate Rows
df.drop_duplicates(inplace = True)

# Checking for any Duplicate rows
df.duplicated().any()
```

Out[13]:

```
False
```

### 5. Remodelling the datatype of column

In [14]:

```
# Changing the Datatype
df['release_date'] = pd.to_datetime(df['release_date'])
df['budget'] = df['budget'].astype(int)
df['revenue'] = df['revenue'].astype(int)

# Checking the status
df.head(2)
```

Out[14]:

| | budget | revenue | original_title | cast | director | runtime | geni |
|---|---|---|---|---|---|---|---|
| **0** | 150000000 | 1513528810 | Jurassic World | Chris Pratt\|Bryce Dallas Howard\|Irrfan Khan\|Vi... | Colin Trevorrow | 124.0 | Action\|Adventure\|Scier Fiction\|Thri |
| **1** | 150000000 | 378436354 | Mad Max: Fury Road | Tom Hardy\|Charlize Theron\|Hugh Keays-Byrne\|Nic... | George Miller | 120.0 | Action\|Adventure\|Scier Fiction\|Thri |

### 6. Renaming the columns

In [15]:

```
# Renaming the columns
df.rename(columns = {'budget' : 'budget(in $)','revenue' : 'revenue(in $)'},inplace = True)

# Checking the columns
df.head(0)
```

Out[15]:

| budget(in $) | revenue(in $) | original_title | cast | director | runtime | genres | production_companies | re |
|---|---|---|---|---|---|---|---|---|

In [16]:

```
# Getting Shape of dataframe
y = df.shape
print("Total No of Row :{}\nTotal No of Columns:{}".format(y[0],y[1]))
```

```
Total No of Row :3805
Total No of Columns:12
```

```
Our Cleanning process has been completed.\ We can see, Before We've Total 10866 rows and
21 columns\ And Now,After Cleaning the data,\ Now We've Total No of Rows:3805 and
Columns:12
```

## Exploratory Data Analysis :-

Before starting our EDA I will like to add one more columns i.e profit which is one of the major column for answering the questions.

In [17]:

```
# Inserting the new column 'Profit'
df.insert(2,'profit',df['revenue(in $)'] - df['budget(in $)'])

# Checking the columns
df.head(0)
```

Out[17]:

| budget(in $) | revenue(in $) | profit | original_title | cast | director | runtime | genres | production_compa |
|---|---|---|---|---|---|---|---|---|

## A. General Questions

### A1. Which is the Least and Most Profitable Movie?

In [18]:

```
# Function for Easy Code
def high_low(col):

    #taking the index value of the highest number in profit column
    high_id = df[col].idxmax()
    #calling by index number,storing that row info to a variable
    high = pd.DataFrame(df.loc[high_id])

    #taking the index value of the least number in profit column
    low_id = df[col].idxmin()
    #calling by index number,storing that row info to a variable
    low = pd.DataFrame(df.loc[low_id])

    #concatenating two dataframes
    res = pd.concat([high,low],axis = 1)

    return res
```

In [19]:

```
# Calling the function
high_low('profit')
```

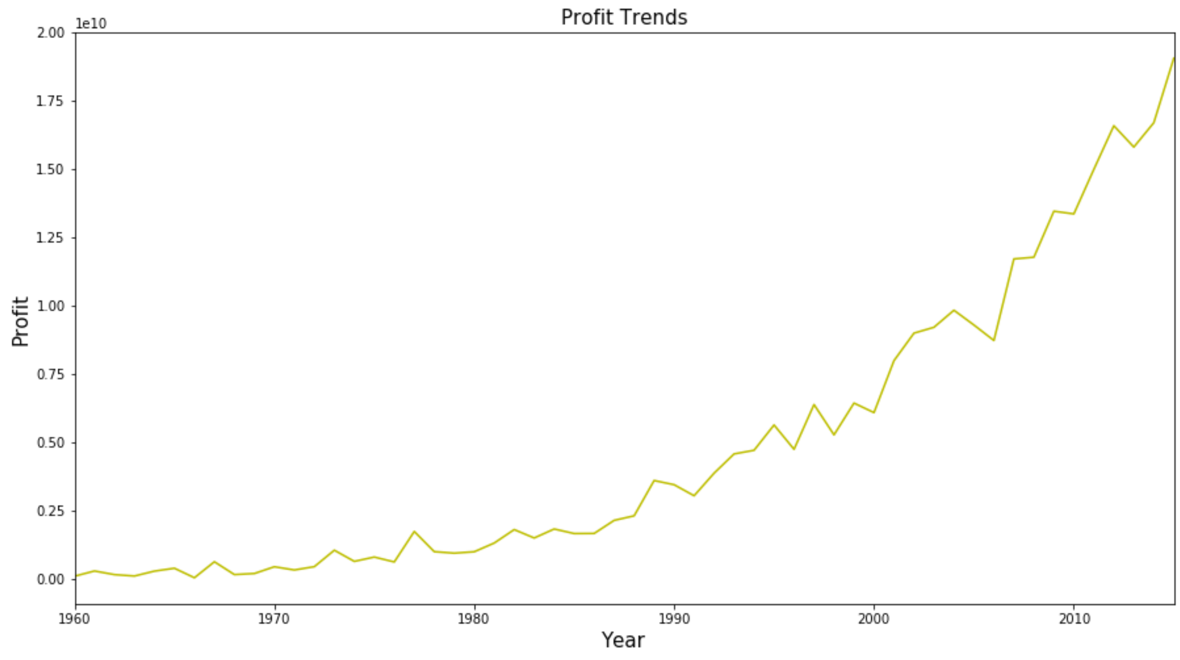
Out[19]:

| | 1386 | 2244 |
|---|---|---|
| **budget(in $)** | 237000000 | 425000000 |
| **revenue(in $)** | -2147483648 | 11087569 |
| **profit** | 1910483648 | -413912431 |
| **original_title** | Avatar | The Warrior's Way |
| **cast** | Sam Worthington\|Zoe Saldana\|Sigourney Weaver\|S... | Kate Bosworth\|Jang Dong-gun\|Geoffrey Rush\|Dann... |
| **director** | James Cameron | Sngmoo Lee |
| **runtime** | 162 | 100 |
| **genres** | Action\|Adventure\|Fantasy\|Science Fiction | Adventure\|Fantasy\|Action\|Western\|Thriller |
| **production_companies** | Ingenious Film Partners\|Twentieth Century Fox ... | Boram Entertainment Inc. |
| **release_date** | 2009-12-10 00:00:00 | 2010-12-02 00:00:00 |
| **vote_count** | 8458 | 74 |
| **vote_average** | 7.1 | 6.4 |
| **release_year** | 2009 | 2010 |

### A2. Profit trends from year to year?

In [20]:

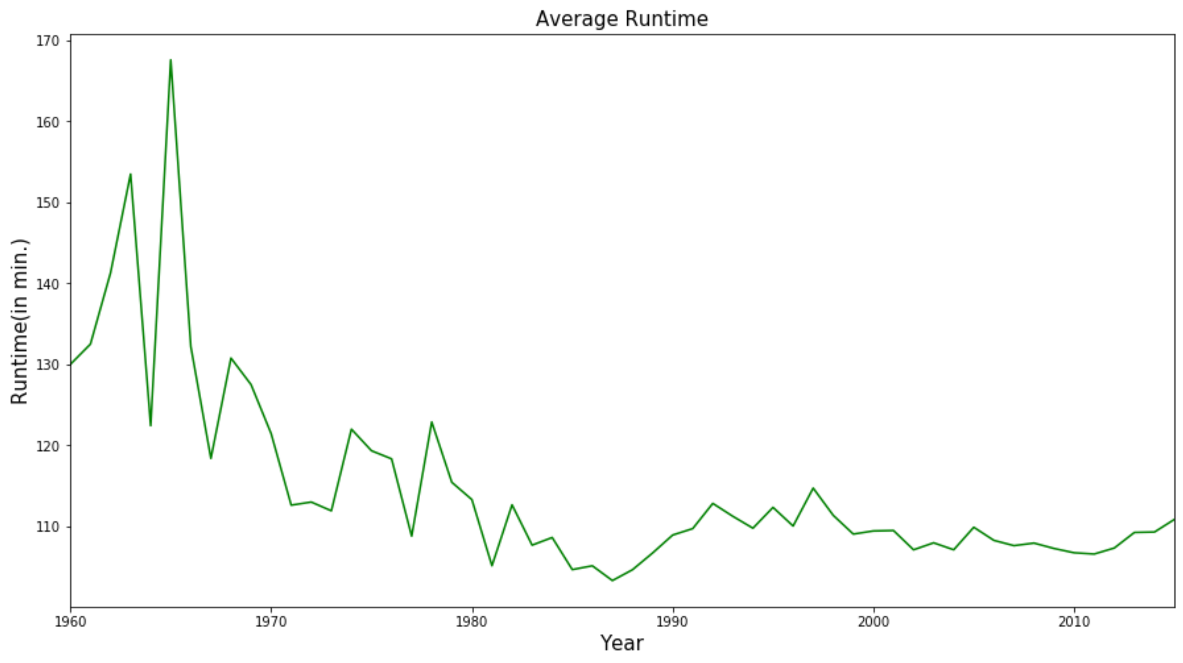
```
# Using the groupby function, calculating the sum and plotting the results
df.groupby('release_year')['profit'].sum().plot(kind = 'line',figsize = (15,8),color = 'y')
plt.title('Profit Trends',fontsize = 15)
plt.xlabel('Year',fontsize = 15)
plt.ylabel('Profit',fontsize = 15);
```



### A3. Average Runtime of Movie Over the Years?

In [21]:

```
# Using the groupby function,calculating the mean() and plotting the results
df.groupby('release_year')['runtime'].mean().plot(kind = 'line',figsize = (15,8),color = 'g
plt.title('Average Runtime',fontsize = 15)
plt.xlabel('Year',fontsize = 15)
plt.ylabel('Runtime(in min.)',fontsize = 15);
```



### A4. Which Movie has the Greatest And Least Budget?

In [22]:

```
# Calling the function
high_low('budget(in $)')
```

Out[22]:

| | 2244 | 2618 |
|---|---|---|
| **budget(in $)** | 425000000 | 1 |
| **revenue(in $)** | 11087569 | 100 |
| **profit** | -413912431 | 99 |
| **original_title** | The Warrior's Way | Lost & Found |
| **cast** | Kate Bosworth\|Jang Dong-gun\|Geoffrey Rush\|Dann... | David Spade\|Sophie Marceau\|Ever Carradine\|Step... |
| **director** | Sngmoo Lee | Jeff Pollack |
| **runtime** | 100 | 95 |
| **genres** | Adventure\|Fantasy\|Action\|Western\|Thriller | Comedy\|Romance |
| **production_companies** | Boram Entertainment Inc. | Alcon Entertainment\|Dinamo Entertainment |
| **release_date** | 2010-12-02 00:00:00 | 1999-04-23 00:00:00 |
| **vote_count** | 74 | 14 |
| **vote_average** | 6.4 | 4.8 |
| **release_year** | 2010 | 1999 |

### A5. Top 3 Cheapest and Expensive Profitable Movies

A5(1). Top 3 Expensive Profitable Movies:

In [23]:

```
def Cheap_exp_profit(val):
    return df.query('profit > 50000000').sort_values('budget(in $)',ascending = val).head(3
```

In [24]:

```
Cheap_exp_profit(False)
```

Out[24]:

| | budget(in $) | revenue(in $) | profit | original_title | cast | director | runtime | |
|---|---|---|---|---|---|---|---|---|
| **3375** | 380000000 | 1021683000 | 641683000 | Pirates of the Caribbean: On Stranger Tides | Johnny Depp\|PenÃ©lope Cruz\|Geoffrey Rush\|Ian M... | Rob Marshall | 136.0 | Ad |
| **7387** | 300000000 | 961000000 | 661000000 | Pirates of the Caribbean: At World's End | Johnny Depp\|Orlando Bloom\|Keira Knightley\|Geof... | Gore Verbinski | 169.0 | Ad |
| **14** | 280000000 | 1405035767 | 1125035767 | Avengers: Age of Ultron | Robert Downey Jr.\|Chris Hemsworth\|Mark Ruffalo... | Joss Whedon | 141.0 | Act |

### *A5(2). Top 3 Expensive Profitable Movies:*

In [25]:

```
Cheap_exp_profit(True)
```

Out[25]:

| | budget(in $) | revenue(in $) | profit | original_title | cast | director | runtim |
|---|---|---|---|---|---|---|---|
| **10495** | 113 | 115103979 | 115103866 | The Karate Kid, Part II | Ralph Macchio\|Pat Morita\|Martin Kove\|Charlie T... | John G. Avildsen | 113.( |
| **7447** | 15000 | 193355800 | 193340800 | Paranormal Activity | Katie Featherston\|Micah Sloat\|Mark Fredrichs\|A... | Oren Peli | 86.( |
| **2449** | 25000 | 248000000 | 247975000 | The Blair Witch Project | Heather Donahue\|Michael C. Williams\|Joshua Leo... | Daniel Myrick\|Eduardo SÃ¡nchez | 81.( |

## B. What are the similar characteristics does the most profitable movie have?

In [26]:

```
# Function for Easy Code
def avg(col):

    #Calculating the mean and returning the result
    return df.query('profit > 50000000')[col].mean()
```

In [27]:

```
# Function for Easy Code
def value(col):
    # Convert column to string and seperate it by '|'
    data = df.query('profit > 50000000')[col].str.cat(sep = '|')

    # Storing the values seperately in a Pandas series
    data = pd.Series(data.split('|'))

    # Counting the data and arraging in descending order
    count = data.value_counts(ascending = False)

    return count
```

### B1. Budget

In [28]:

```
# Calling the function
budget = avg('budget(in $)')
print('The Average Budget of a profitable movie is ${0:.1f}'.format(budget))
```

```
The Average Budget of a profitable movie is $60483360.9
```

### B2. Cast

In [29]:

```
# Calling the function and storing the data in the variable
cast = value('cast')
# Top 5 Data
cast.head()
```

Out[29]:

```
Tom Cruise           27
Brad Pitt            25
Tom Hanks            22
Sylvester Stallone   21
Cameron Diaz         20
dtype: int64
```

### B3. Director

In [30]:

```
# Calling the function and storing the data in the variable
director = value('director')
# Top 5 Data
director.head()
```

Out[30]:

```
Steven Spielberg    23
Robert Zemeckis     13
Clint Eastwood      12
Tim Burton          11
Ridley Scott        10
dtype: int64
```

### B4. Runtime

In [31]:

```
# Calling function
runtime = avg('runtime')
print('The Average Runtime of a profitable movie is {0:.1f} minutes.'.format(runtime))
```

```
The Average Runtime of a profitable movie is 113.6 minutes.
```

### B5. Genres

In [32]:

```
# Calling the function
genres = value('genres')
genres
```

Out[32]:

```
Comedy             492
Drama              480
Action             463
Thriller           404
Adventure          379
Family             229
Romance            215
Science Fiction    206
Fantasy            201
Crime              193
Horror             123
Animation          122
Mystery            112
Music               47
War                 46
History             39
Western             14
Documentary          4
dtype: int64
```

In [33]:

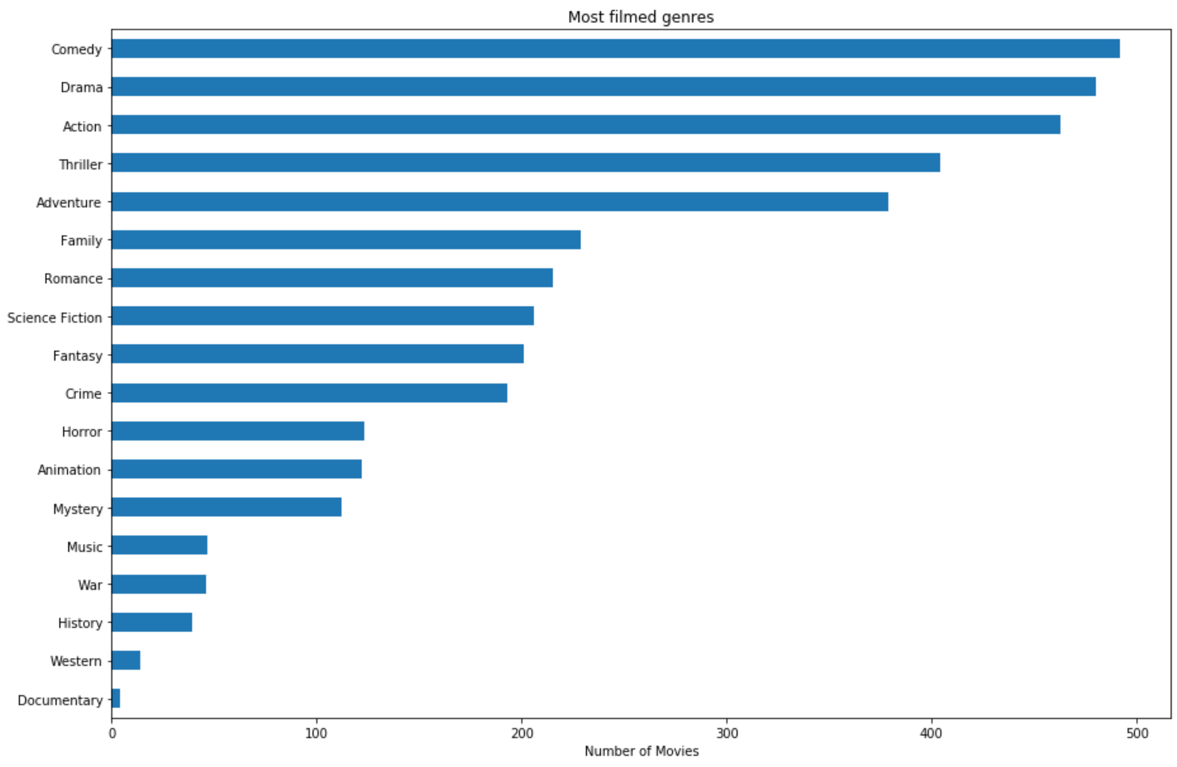
```
# Top 5 Genres
genres.head()
```

Out[33]:

```
Comedy        492
Drama         480
Action        463
Thriller      404
Adventure     379
dtype: int64
```

In [34]:

```
# Sorting the Genres in ascending Order and plotting the bar graph
genres.sort_values(ascending = True,inplace = True)
genres.plot(kind = 'barh',figsize = (15,10))
plt.title('Most filmed genres')
plt.xlabel('Number of Movies');
```



## B6. Production Companies

In [35]:

```
# Calling the function and storing the data in the variable
pd_cmp = value('production_companies')
# Top 20 Data
pd_cmp.head(20)
```

Out[35]:

```
Universal Pictures                          156
Warner Bros.                                144
Paramount Pictures                          130
Twentieth Century Fox Film Corporation      118
Columbia Pictures                            93
Walt Disney Pictures                         78
New Line Cinema                              67
Columbia Pictures Corporation                51
Relativity Media                             50
Touchstone Pictures                          46
DreamWorks SKG                               43
Metro-Goldwyn-Mayer (MGM)                    42
Amblin Entertainment                         40
Village Roadshow Pictures                    35
Dune Entertainment                           34
Regency Enterprises                          32
Fox 2000 Pictures                            26
DreamWorks Animation                         25
TriStar Pictures                             25
Legendary Pictures                           24
dtype: int64
```

In [36]:

```

# Top 5 Production Companies
pd_cmp.head(5)
```

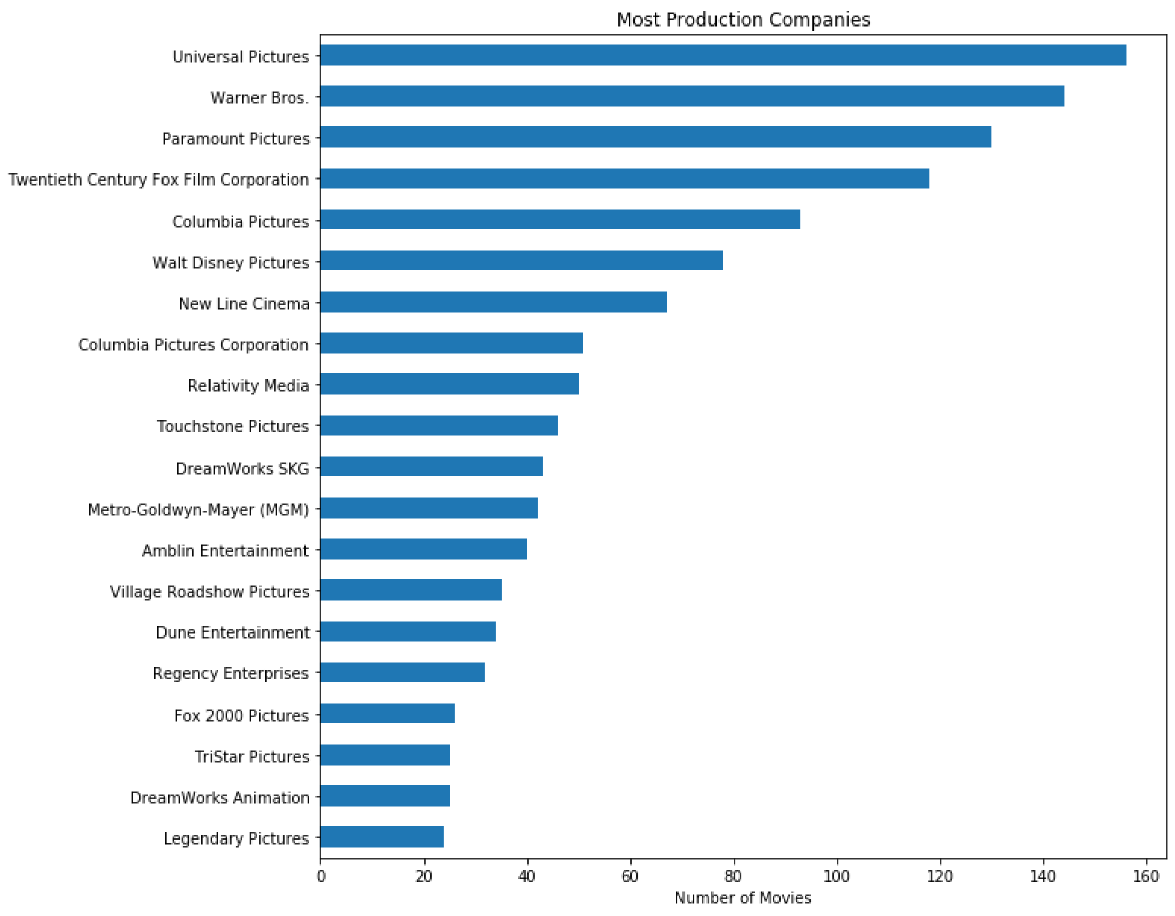
Out[36]:

```
Universal Pictures                          156
Warner Bros.                                144
Paramount Pictures                          130
Twentieth Century Fox Film Corporation      118
Columbia Pictures                            93
dtype: int64
```

In [37]:

```
# Top 20 Production Companies
pd_cmp = pd_cmp.head(20)

# Sorting the values in ascending order and plotting the bar graph
pd_cmp.sort_values(ascending = True,inplace = True)
pd_cmp.plot(kind = 'barh',figsize = (10,10))
plt.title('Most Production Companies')
plt.xlabel('Number of Movies');
```



### B7. Revenue

In [38]:

```
# Calling the function
rev = avg('revenue(in $)')
print("The Average Revenue of Profitable Movie is: ${0:.1f}".format(rev))
```

```
The Average Revenue of Profitable Movie is: $251404908.8
```

## Conclusions :-

```
From the above analysis,we can conclud that to have profitable/successful movies we
should have:
```

```
1. Budget of minimum $60 Million.

2. Cast should be one or more of : Tom Cruise,Brad Pitt,Tom Hanks,Sylvester
Stallone,Cameron Diaz.

3. Director should be any one or more of : Steven Spielberg,Robert Zemeckis,Clint
Eastwood,Tim Burton,Tony Scott.

4. Runtime should be minimum of 113 Minutes or 2 Hours.

5. Generes should be one or more of :Comedy,Drama,Action,Thriller,Adventure.

6. Production Companies should be one or more of :Universal Pictures,Warner
Bros,Paramount Pictures,Twentieth Century Fox Film Corporation,Columbia Pictures
```

```
By doing this,the revenue of should be around $255 Million.
```