Project: Investigate a Dataset (TMDb Movies Dataset)

By: Mohamed Wakiel

Table of Contents

Introduction

Overview

To complete my Data Analysis project I will be using the TMDb movie dataset.

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue. ● Certain columns, like 'cast' and 'genres', contain multiple values separated by pipe (|) characters. ● There are some odd characters in the 'cast' column. Don't worry about cleaning them. You can leave them as is. ● The final two columns ending with "_adj" show the budget and revenue of the associated movie in terms of 2010 dollars, accounting for inflation over time.

Questions:

1. Movies which had most and least budgets
2. Movies with most and least earned revenue
3. Movies which had most and least profit
4. Movies with the longest and shortest runtime
5. Average runtime of the movies
6. In which year we had most no.of profitable movies
7. Characteristics of profitable movies

   For this question I will explore the following properties of the movies with respect to the profitable movies:

   - Successful Genres.
   - Average Budget of the movies.
   - Average Revenue earned by the movies
   - Average duration of the movies

- Most Frequent Cast

In [46]:
```
# Use this cell to set up import statements for all of the packages that you
#   plan to use.

# Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html

import pandas as pd
import numpy as np
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

Data Wrangling

> After loading the data set, I will use Pandas to explore some ways to evaluate and build intuition about the data set, and then I will continue the cleaning process to include only the relevant issues that I will apply to the analysis process on the data set data. And delete unused data.

General Properties

In [3]:
```
# 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 'df'
df = pd.read_csv('tmdb-movies.csv')

#printing first five rows
df.head()
```

Out[3]:

	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://www.

	id	imdb_id	popularity	budget	revenue	original_title	cast	
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http:
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	

5 rows × 21 columns

In [5]:
```
#printing the last five rows
df.tail(5)
```

Out[5]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage
10861	21	tt0060371	0.080598	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	NaN
10862	20379	tt0060472	0.065543	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	NaN
10863	39768	tt0060161	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	NaN
10864	21449	tt0061177	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	NaN
10865	22293	tt0060666	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	NaN

5 rows × 21 columns

In [6]:
```
#The dimensions of the dataset
df.shape
```

Out[6]: (10866, 21)

Cleaning Check

In [7]:
```
# this will display a concise summary of the dataframe,
# including the number of non-null values in each column

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
```

In [8]:
```
# check for duplicates in the data
sum(df.duplicated())
```

Out[8]: 1

In [9]:
```
# although the datatype for release_date appears to be object in Pandas, further
# investigation shows it's a string

type(df['release_date'][0])
```

Out[9]: str

In [10]:
```
df.describe()
```

Out[10]:

	id	popularity	budget	revenue	runtime	vote_count	vote_a
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6

	id	popularity	budget	revenue	runtime	vote_count	vote_a
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9

> ## Obsevations from the data set
>
> 1.) The currency unit is not mentioned in the data set. Therefore, for my analysis, I will use the US dollar because it is the most widely used currency worldwide.
>
> 2.) vote_count is widely different for all the movies, so we cannot directly conclude the popularity of the movies based on the average vote count

Data Cleaning (Removing unused information from the dataset)

> 1. Remove unnecessary columns such as 'id', 'imdb_id', 'popularity', 'budget_adj', 'revenue_adj', 'homepage', production_company, 'keywords', 'overview', 'production_companies', 'vote_count' and 'vote_average'.
> 2. Changing release date column from string to date format.
> 3. Removing the duplicated values.
> 4. Replacing zero with NAN in runtime column.
> 5. replace all the values from '0' to NAN in 'budget' and 'revenue' columns, then removing them.
> 6. Changing format of budget and revenue columns.

1. Removing unused columns

> Just to be clear, these columns I want to delete are just irrelevant to my process to investage this dataset, the columns I'm going to dismiss: id, imdb_id, popularity, budget_adj, revenue_adj, homepage, keywords, overview, production_companies, vote_count and vote_average:

In [13]:
```
# After discussing the structure of the data and any problems that need to be
#   cleaned, perform those cleaning steps in the second part of this section.

# a list of columns we want to remove
del_col = [ 'id', 'imdb_id', 'popularity', 'budget_adj', 'revenue_adj', 'homepage',
#deleting the columns from the database
df = df.drop(del_col, 1)

# previewing the new dataset
df.head()
```

Out[13]:

	budget	revenue	original_title	cast	director	tagline	runtime	
0	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	124	Action|Adv

	budget	revenue	original_title	cast	director	tagline	runtime	
1	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	120	Action|Adv
2	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	One Choice Can Destroy You	119	Adv
3	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Every generation has a story.	136	Action|Adv
4	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Vengeance Hits Home	137	Actio

2. Changing the release date column into standard date format

In [14]:
```
df.release_date = pd.to_datetime(df['release_date'])
```

In [15]:
```
type(df['release_date'][0])
```

Out[15]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

In [16]:
```
df.head()
```

Out[16]:

	budget	revenue	original_title	cast	director	tagline	runtime	
0	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	124	Action|Adv
1	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	120	Action|Adv
2	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	One Choice Can Destroy You	119	Adv
3	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Every generation has a story.	136	Action|Adv
4	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Vengeance Hits Home	137	Actio

3. Removing the duplicated values

In [17]:
```
df.drop_duplicates(keep ='first', inplace=True)
```

In [18]:
```
rows, col = df.shape

print('There are now {} total entries in our dataset after removing the duplicates.'
```

```
There are now 10864 total entries in our dataset after removing the duplicates.
```

4. Replacing zero with NAN in runtime column.

In [19]:
```
#replacing 0 with NaN of runtime column in the dataset
df['runtime'] =df['runtime'].replace(0, np.NAN)
```

5.Removing (0) from budget and revenue columns

In [29]:
```
# I think you need create boolean DataFrame by compare all filtered columns values b
# and then check all Trues per rows by all:
df = df[(df[['revenue','budget']] != 0).all(axis=1)]
rows, col = df.shape
print('Now we have only {} no.of movies.'.format(rows-1))
```

```
Now we have only 3853 no.of movies.
```

6. Changing format of budget and revenue columns

Earlier when we I checked the data information summary for the dataset columns, each of the budget and revenue columns were float data types.

In [30]:
```
change_type=['budget', 'revenue']

#changing data type
df[change_type]=df[change_type].applymap(np.int64)

#printing data types of the dataset to see the changed information
df.dtypes
```

Out[30]:
```
budget                    int64
revenue                   int64
original_title           object
cast                     object
director                 object
tagline                  object
runtime                 float64
genres                   object
release_date      datetime64[ns]
release_year               int64
dtype: object
```

Exploratory Data Analysis

> **Tip**: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

Research Question 1 : Movies which had most and least budgets

In [31]:
```
import pprint
#defining the function
def calculate(column):
    #for highest value
    high= df[column].idxmax()
    high_details=pd.DataFrame(df.loc[high])

    #for lowest value
    low= df[column].idxmin()
    low_details=pd.DataFrame(df.loc[low])

    #collecting data in one place
    info=pd.concat([high_details, low_details], axis=1)

    return info

#calling the function
calculate('budget')
```

Out[31]:

	2244	2618
budget	425000000	1
revenue	11087569	100
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
tagline	Assassin. Hero. Legend.	A comedy about a guy who would do anything to ...
runtime	100	95
genres	Adventure|Fantasy|Action|Western|Thriller	Comedy|Romance
release_date	2010-12-02 00:00:00	1999-04-23 00:00:00
release_year	2010	1999

> The entery with id no. 2244 shows the largest budget with total of 425000000 dollar.
>
> While the entery with id no. 2618 have the lowest budget with total of 1 dollar.

Research Question 2 : Movies with most and least earned revenue

In [32]:
```
# Calling the same function **calculate(column)** again for this analysis
calculate('revenue')
```

Out[32]:

	1386	5067
budget	237000000	6000000
revenue	2781505847	2
original_title	Avatar	Shattered Glass

	1386	5067
cast	Sam Worthington|Zoe Saldana|Sigourney Weaver|S...	Hayden Christensen|Peter Sarsgaard|ChloÃ« Sevi...
director	James Cameron	Billy Ray
tagline	Enter the World of Pandora.	NaN
runtime	162	94
genres	Action|Adventure|Fantasy|Science Fiction	Drama|History
release_date	2009-12-10 00:00:00	2003-11-14 00:00:00
release_year	2009	2003

> The entery with id no. 1386 shows the largest revenue with total of 237000000 dollar.
>
> While the entery with id no. 5067 have the lowest revenue with total of 2 dollar.

Research Question 3 : Movies which had most and least profit

We don't have a column represents the total profit of each movie in our dataset, to find the most and least profit movies so I will answer this question in two steps:

A. Calculating the profit of each movie

Adding a new column to the dataset represents the total profit for each movie in our dataset by subtracting the revenue from the budget.

In [39]:
```
#insert function with three parameters(index of the column in the dataset, name of t
df.insert(2,'profit',df['revenue']-df['budget'])

#previewing the changes in the dataset
df.head(2)
```

Out[39]:

	budget	revenue	profit	original_title	cast	director	tagline	runtime	
0	150000000	1513528810	1363528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	124.0	Ac
1	150000000	378436354	228436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	120.0	Ac

B. Movies with most and least profit

In [40]:
```
# Calling the same function **calculate(column)** again for this analysis
calculate('profit')
```

Out[40]:

	1386	2244

	1386	2244
budget	237000000	425000000
revenue	2781505847	11087569
profit	2544505847	-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
tagline	Enter the World of Pandora.	Assassin. Hero. Legend.
runtime	162	100
genres	Action|Adventure|Fantasy|Science Fiction	Adventure|Fantasy|Action|Western|Thriller
release_date	2009-12-10 00:00:00	2010-12-02 00:00:00
release_year	2009	2010

> The entery with id no. 1386 shows the largest profit with total of 2544505847 dollar.
>
> While the entery with id no. 2244 have the lowest budget with total of -413912431 dollar. Meaning, The movie made them lose 413912431.

Research Question 4 : Movies with longest and shortest runtime

In [41]:
```
# CAlling the same function **calculate(column)** again for this analysis
calculate('runtime')
```

Out[41]:

	2107	5162
budget	18000000	10
revenue	871279	5
profit	-17128721	-5
original_title	Carlos	Kid's Story
cast	Edgar RamÃrez|Alexander Scheer|Fadi Abi Samra...	Clayton Watson|Keanu Reeves|Carrie-Anne Moss|K...
director	Olivier Assayas	Shinichiro Watanabe
tagline	The man who hijacked the world	NaN
runtime	338	15
genres	Crime|Drama|Thriller|History	Science Fiction|Animation
release_date	2010-05-19 00:00:00	2003-06-02 00:00:00
release_year	2010	2003

> The movie with id no. 2107 shows the longest runtime or duration about 338 minutes.

> While the movie with id no. 5162 shows the shortest runtime about 15 minutes.

Research Question 5 : Average runtime of the movies

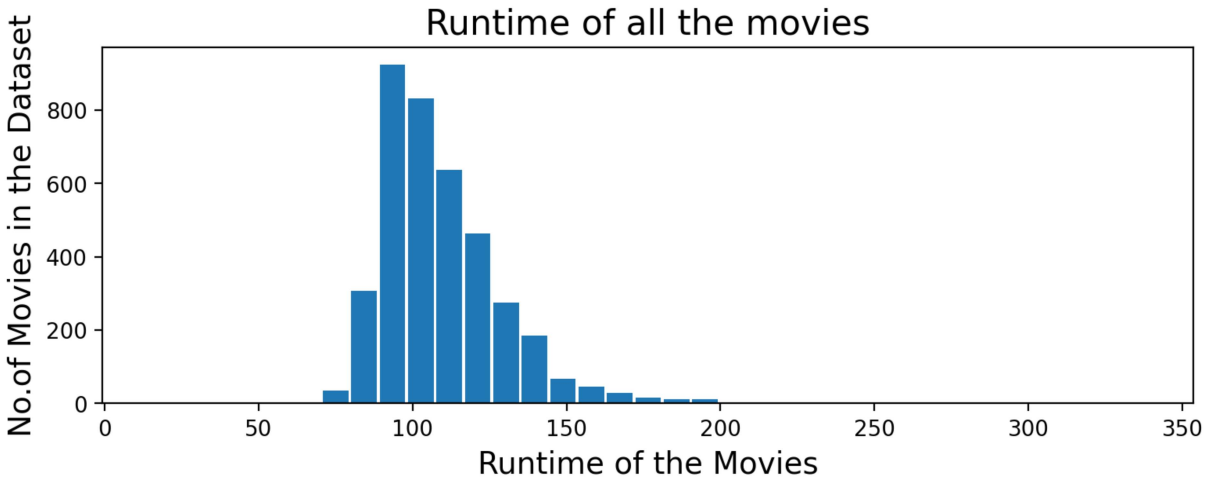
In [42]:
```
# defining a function to find average of a column
def avg_fun(column):
    return df[column].mean()
```

In [43]:
```
#calling above function
avg_fun('runtime')
```

Out[43]: 109.22029060716139

> The average runtime of all movies is 109 minutes. Now I'll analyse this value from a visual perspective.

In [67]:
```
#plotting a histogram of runtime of movies

#giving the figure size(width, height)
plt.figure(figsize=(9,3), dpi = 100)

#On x-axis
plt.xlabel('Runtime of the Movies', fontsize = 14)
#On y-axis
plt.ylabel('No.of Movies in the Dataset', fontsize=14)
#Name of the graph
plt.title('Runtime of all the movies', fontsize=16)

#giving a histogram plot
plt.hist(df['runtime'], rwidth = 0.9, bins =35)
#displays the plot
plt.show()
```

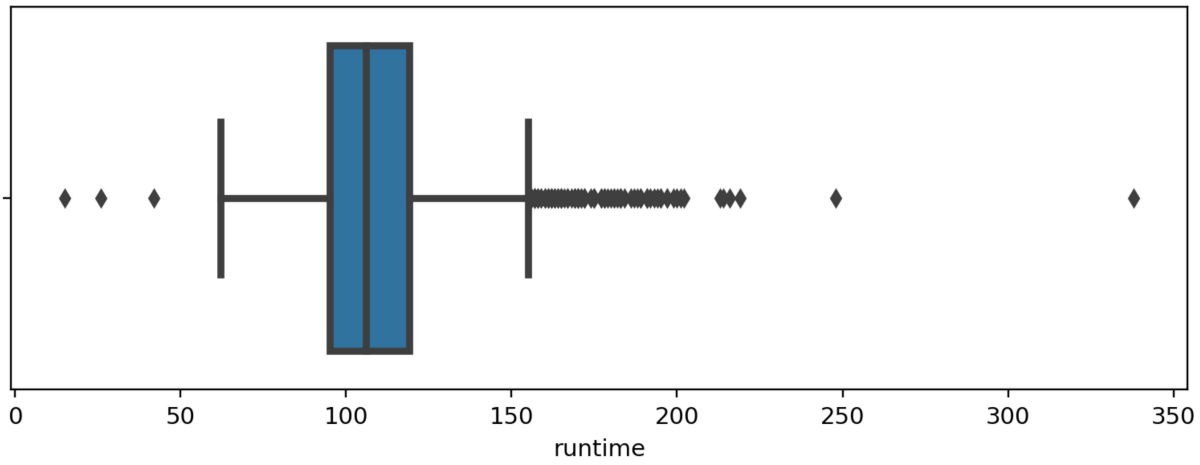


> The above distribution graph is positively or right skewed! Most of the movies duarations arrange from 80 to 115 minutes. More than 800 movies fall in this criteria.

Lets take another look at runtime of the movie using different kind of plots i.e Box Plot and Data Point Plot

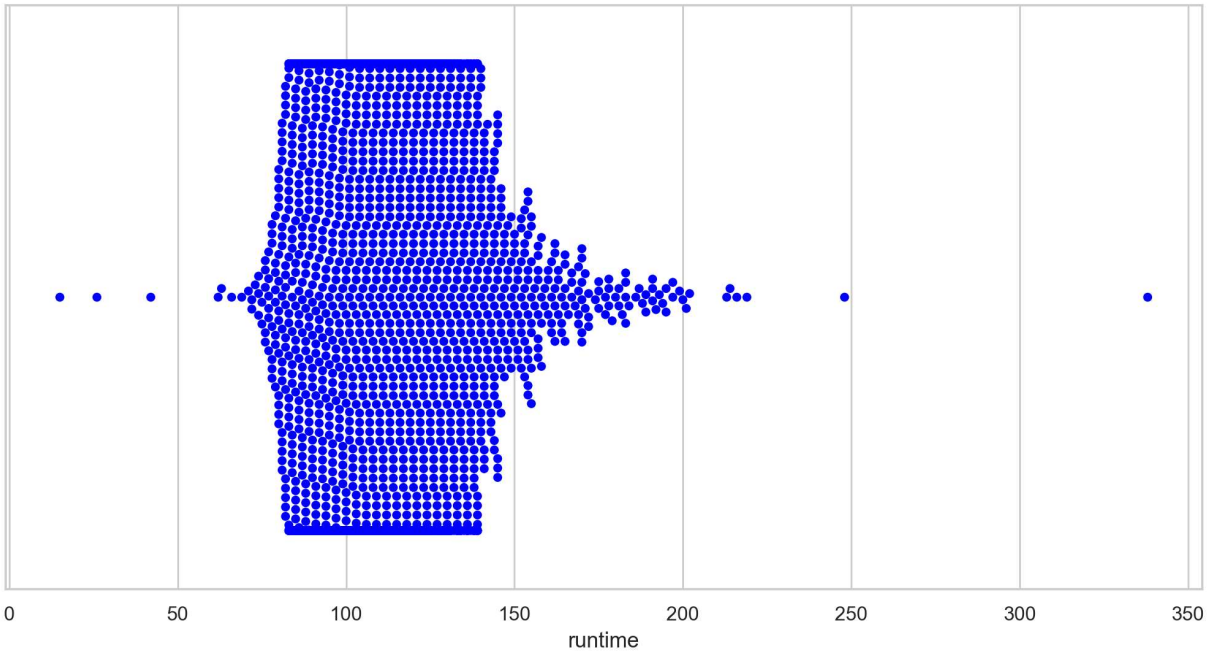
In [66]:
```
import seaborn as sns
#The First plot is box plot of the runtime of the movies
```

```
plt.figure(figsize=(9,3), dpi = 105)

#using seaborn to generate the boxplot
data =df['runtime']
sns.boxplot(x=data, linewidth = 3)
#diplaying the plot
plt.show()
```



In [89]:
```
sns.set(style="whitegrid")
#The Second plot is the data points plot of runtime of movies

plt.figure(figsize=(12,6), dpi = 105)
#using seaborn to generate the plot
data = df['runtime']
sns.swarmplot(x=data, color = 'blue')
#displaying the plot
plt.show()
```

```
C:\Users\Wakiel\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarnin
g: 65.7% of the points cannot be placed; you may want to decrease the size of the ma
rkers or use stripplot.
  warnings.warn(msg, UserWarning)
```



In [56]:
```
#getting specific runtime
df['runtime'].describe()
```

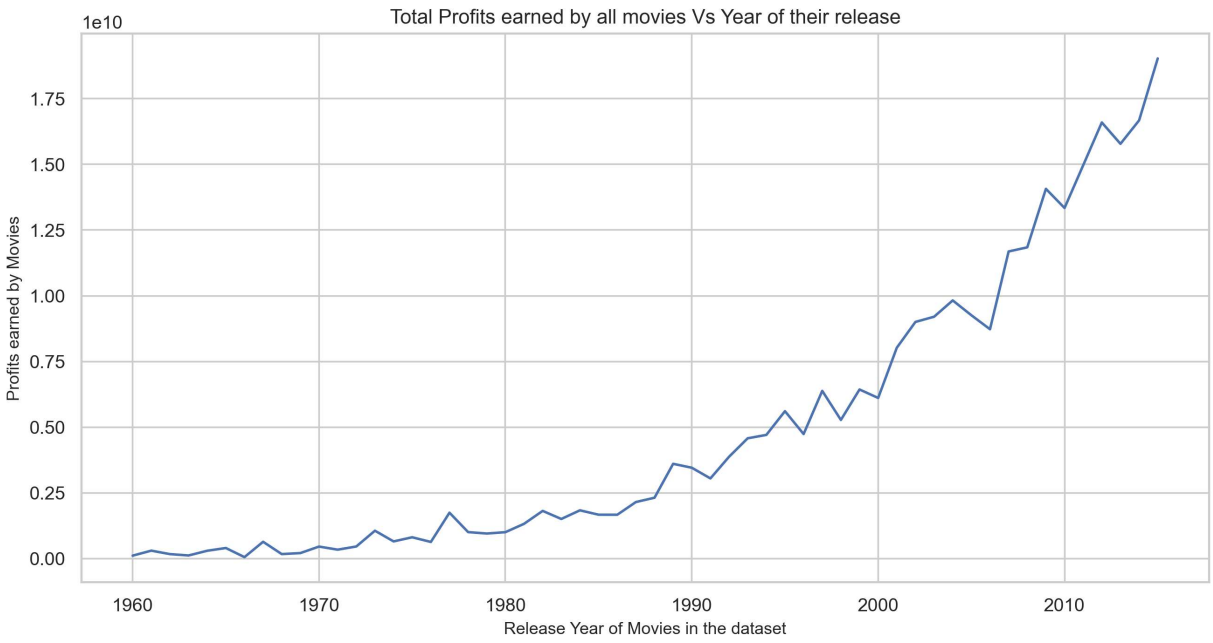
Out[56]:
```
count    3854.000000
mean      109.220291
```

```
std        19.922820
min        15.000000
25%        95.000000
50%       106.000000
75%       119.000000
max       338.000000
Name: runtime, dtype: float64
```

> The first plot above "the box-plot" gives us an overall idea of how spreaded the ditribution is in case of runtime of the movies. It also shows the outliers here.
>
> The second plot "Data Point Plot" generated above gives a visual of the distribution of runtime of movies by plotting the points againts their respective positions in the ditribution.
>
> By looking at both of the plots and calculations, we can get that:
>
> 1. 25% of the movies have a runtime of less than 95 minutes
> 2. 50% of the movies have a runtime of less than 106 minutes.
> 3. 75% of the movies have a runtime of less than 119 minutes

Research Question 6 : In which year we had most no.of profitable movies.

In [85]:
```
#Using Line plot for this analysis
#Since we want to know the profits of movies for every year therefore we have to sum

profits_year = df.groupby('release_year')['profit'].sum()

#figure size(width, height)
plt.figure(figsize=(12,6), dpi = 130)

#on x-axis
plt.xlabel('Release Year of Movies in the dataset', fontsize = 10)
#on y-axis
plt.ylabel('Profits earned by Movies', fontsize = 10)
#title of the line plot
plt.title('Total Profits earned by all movies Vs Year of their release')

#plotting the graph
plt.plot(profits_year)

#displaying the line plot
plt.show()
```



In [81]:
```
#To find that which year made the highest profit?
profits_year.idxmax()
```

Out[81]: 2015

> So we can conclude both graphically as well as by calculations that year 2015 was the year where movies made the highest profit.

Research Question 7 :With respect to the profitable movies

> We will now find characteristics of profitable movies, Before moving further we need to clean our data again. We will be considering only those movies who have earned a significant amount of profit.
>
> **So lets fix this amount by selecting the movies having profit of $60M or more :**

In [102…
```
#selecting the movies having profit of $50M or more
profit_data = df[df['profit'] >= 60000000]

#reindexing new data
profit_data.index = range(len(profit_data))

#we will start from 1 instead of 0
profit_data.index = profit_data.index + 1

#printing the changed dataset
profit_data.head()
```

Out[102…

	budget	revenue	profit	original_title	cast	director	tagline	runtin
1	150000000	1513528810	1363528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	The park is open.	124

	budget	revenue	profit	original_title	cast	director	tagline	runtin
2	150000000	378436354	228436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	What a Lovely Day.	120
3	110000000	295238201	185238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	Robert Schwentke	One Choice Can Destroy You	119
4	200000000	2068178225	1868178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	J.J. Abrams	Every generation has a story.	136
5	190000000	1506249360	1316249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	James Wan	Vengeance Hits Home	137

In [103…
```
len(profit_data)
```

Out[103… 1197

> So our dataset is reduced to 1197 rows from 3853 (in earlier case).

A. Succesful Genres

In [104…
```
#function which will take any column as argument from and keep its track
def data(column):
    #will take a column, and separate the string by '|'
    data = profit_data[column].str.cat(sep = '|')

    #giving pandas series and storing the values separately
    data = pd.Series(data.split('|'))

    #arranging in descending order
    count = data.value_counts(ascending = False)

    return count
```

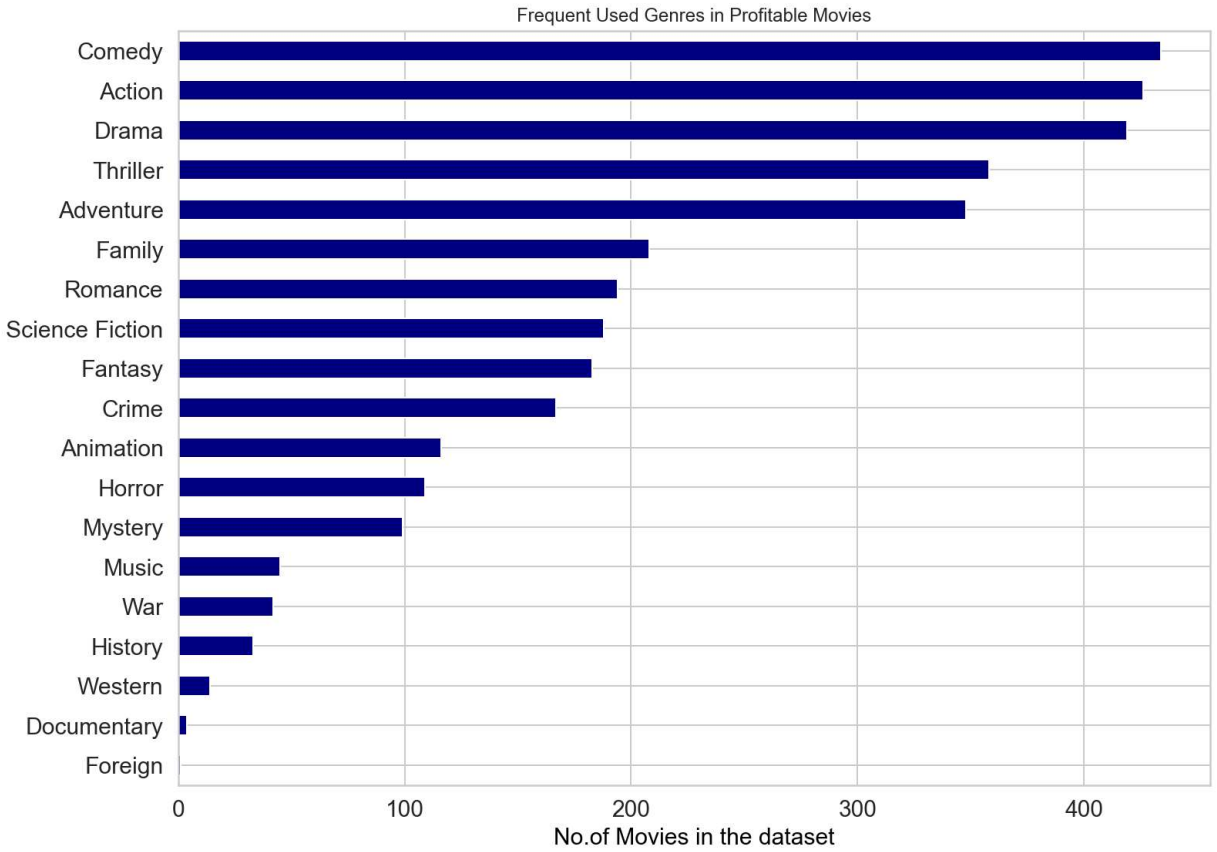
In [107…
```
#variable to store the retured value
count = data('genres')
#printing top 5 values
count.head()
```

Out[107…
```
Comedy       434
Action       426
Drama        419
Thriller     358
Adventure    348
dtype: int64
```

Here's a graphical analysis of the above collected data

In [110…
```
#Lets plot the points in descending order top to bottom as we have data in same form
count.sort_values(ascending = True, inplace = True)
```

```
#ploting
pt = count.plot.barh(color = 'navy', fontsize = 15)

#title
pt.set(title = 'Frequent Used Genres in Profitable Movies')

# on x axis
pt.set_xlabel('No.of Movies in the dataset', color = 'black', fontsize = '15')

#figure size(width, height)
pt.figure.set_size_inches(12, 9)

#ploting the graph
plt.show()
```



The visualization above shows Frequent Used Genres in Profitable Movies from all movies.

B. Average Budget of the movies

In [111…
```
#New function to find average
def profit_avg(column):
    return profit_data[column].mean()
```

In [112…
```
# calling the above function for budget
profit_avg('budget')
```

Out[112… 63757867.39515455

> So the movies having profit of 60 million dollar and more have an average budget of 64 million dollar.

C. Average Revenue earned by the movies

In [113…
```
# calling avg function for revenue
profit_avg('revenue')
```

Out[113… 274739298.8086884

> So the movies having profit of 60 million dollar and more have an average revenue of 274 million dollar.

D. Average duration of the movies

In [114…
```
# calling avg function for runtime
profit_avg('runtime')
```

Out[114… 114.06850459482038

> So the movies having profit of 60 million dollar and more have an average duration of 114 minutes.

E. Most Frequent Cast

We will call the avg function **data(column)** again for this analysis.

In [115…
```
#variable to store the retured value
count = data('cast')
#printing top 5 values
count.head()
```

Out[115…
```
Tom Cruise            26
Tom Hanks             22
Brad Pitt             22
Sylvester Stallone    21
Cameron Diaz          20
dtype: int64
```

> Tom Cruise is on the top for appearing the most in movies profiting more than $60M with total of 26 cast followed by Brad Pitt and Tom Hanks with 22 followed by Sylvester Stallone with 21.

Conclusions

> After this interesting analysis for TMDb dataset and abstract conclusions after each analysis and visualtion, here are some final thoughts:
>
> **For a Movie in order to be considered in a successful criteria**
>
> - Average Budget must be around 64 millon dollar.
> - Average duration of the movie must be 114 minutes.
> - Any one of these actors should be in the cast :Tom Cruise, Brad Pitt, Tom Hanks, Sylvester Stallone,Cameron Diaz.
> - Genre must be : Action, Adventure, Thriller, Comedy, Drama.

> - By doing all this the movie might be one of the hits and hence can earn an average revenue of around 274 million dollar.
>
> Final observation: This analysis was done considering the movies which had a significant amount of profit of around 60 million dollar. This might not be completely error free but by following these suggestions one can increase the probability of a movie to become a hit. Moreover we are not sure if the data provided to us is completely correct and up-to-date. As mentioned before the budget and revenue column do not have currency unit, it might be possible different movies have budget in different currency according to the country they are produce in. So an inconsistency appears here which can state the complete analysis wrong. Dropping the rows with missing values also affected the overall analysis.

In [ ]: