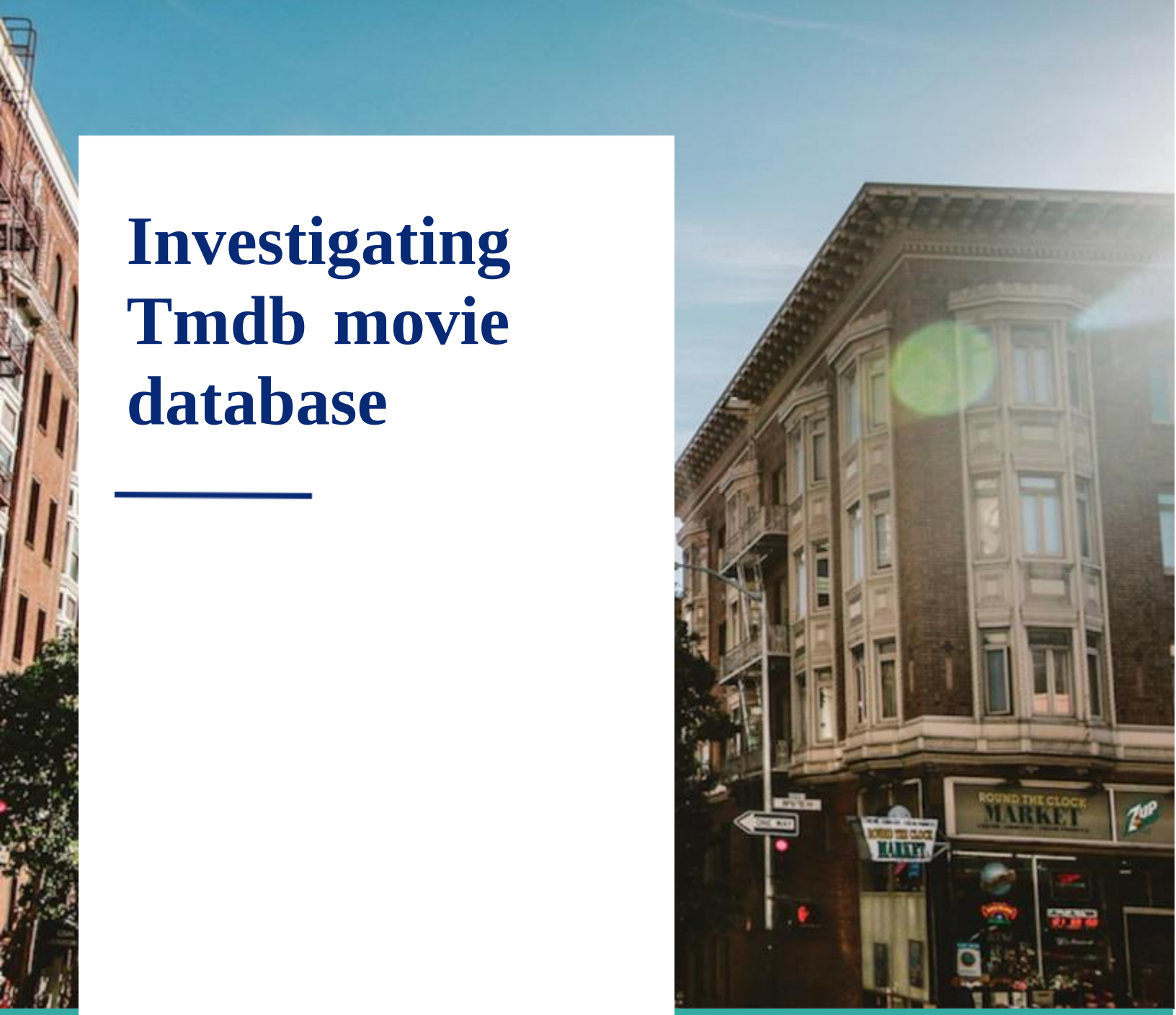


Investigating Tmdb movie database

SEPTEMBER 14

Udacity Data Analyst Nanodegree
Authored by: Neha Singh

Project: Investigate a Dataset (TMDb_Movies Dataset)

Overview

For our 2nd Data Analysis project I have opted for the TMDb movies dataset.

This data set contains information about 10 thousand movies collected from The Movie Database (TMDb), including user ratings and revenue

Questions which I analyzed from this database :

1. Movies, which had most and least profit.
2. Movies with largest and lowest budgets.
3. Movies with most and least earned revenue.
4. Movies with longest and shortest runtime values.
5. Average runtime of all the movies.
6. Year of release vs profitability
7. Average budget of movies which earned $40 million or more
8. Average runtime of movies which earned $40 million or more

Used Code:

 # import required libraries

```
import numpy as np
import pandas as pd
import datetime from datetime
from matplotlib import pyplot as plt
```

DATA WRANGLING

 # Since this dataset contains a lot of undesired data, we will just weed out the unnecessary columns/data so as to make our calculations easy

 # loading data from csv file

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

 # after previewing the dataset with .head() function, we came to know that

1. *there are 21 columns and many of them are unimportant for our observations.*
2. *No currency has been given in the dataset.*

DATA CLEANING

 # removing the unused columns

 # we will delete several unnecessary columns in order to simplify our calculations.

```
# we will do this by assigning them to an array and then dropping them
 useless_col = [ 'id', 'imdb_id', 'popularity',
'budget_adj', 'revenue_adj', 'homepage',
'keywords', 'overview', 'production_companies',
'vote_count',    'vote_average']

movie_data = movie_data.drop(useless_col   , 1)

# checking for duplicates in dataset
Sum(movie_data.duplicated())

# dropping the duplicate data
 movie_data.drop_duplicates(keep    'first' , inplace
 = = True)

# creating a seperate list of revenue and budget column so as to remove 0's from these series
change_list = ['revenue' , 'budget']


# replacing all 0's with NAN'
 movie_data[change_list]  =
movie_data[change_list].replace(0   , np.NAN)

 #dropping NAN
movie_data.dropna(subset  = change_list  , inplace =
True)

# replacing the 0 values of runtime column with NAN
movie_data['runtime']  =
movie_data['runtime'].replace(0   , np.NAN)
```

```
# Checking the datatypes of the given dataset
movie_data.dtypes
```

```
budget             float64
revenue            float64
original_title      object
cast                object
director            object
tagline             object
runtime              int64
genres              object
release_date        object
release_year         int64
dtype: object
```

```
# since the budget and revenue columns is in float, we will change it to int
change_dtype = ['budget' , 'revenue']
```

```
 # changing the datatype to int
movie_data[change_dtype]  =
movie_data[change_dtype].applymap(np.int64)
```

Data Analysis

3- Calculating the profit of each movie

```
# for this we are going use the function insert to create a new
column  called profit_earned
```

```
 movie_data.insert(2  , 'profit_earned'  ,
movie_data['revenue']-movie_data['budget'])
```

Question 1- Which movies had the most and least profit?

```
# we are going to define a function for gettig the answer to this question
def calculate_max_min(column):
    highest_profit_id  = movie_data[column].idxmax()

    highest_profit_id_details  =
pd.DataFrame(movie_data.loc[highest_profit_id])

    lowest_profit_id  = movie_data[column].idxmin()

    lowest_profit_id_details  =
pd.DataFrame(movie_data.loc[lowest_profit_id])

#collecting data in one frame
show = pd.concat([highest_profit_id_details   ,
lowest_profit_id_details]  , axis = 1) return
show

# calling the function
calculate_max_min('profit_earned')
```

The results show that AVATAR had the highest profit while THE WARRIOR'S WAY had the lowest profit

Question 2- which movies had the largest & smallest budget?

```
calculate_max_min('budget')
```

The results show that THE WARRIOR'S WAY had the largest budget while LOST & FOUND had the smallest budget

Question 3- which movies earned the largest & smallest revenue?

```
calculate_max_min('revenue')
```

The results show that AVATAR earned the largest revenue while SHATTERED GLASS earned the smallest revenue

Question 4- movies with longest and shortest runtime?

```
calculate_max_min('runtime')
```

The results show that CARLOS had the longest runtime while KID'S STORY HAD the smallest runtime

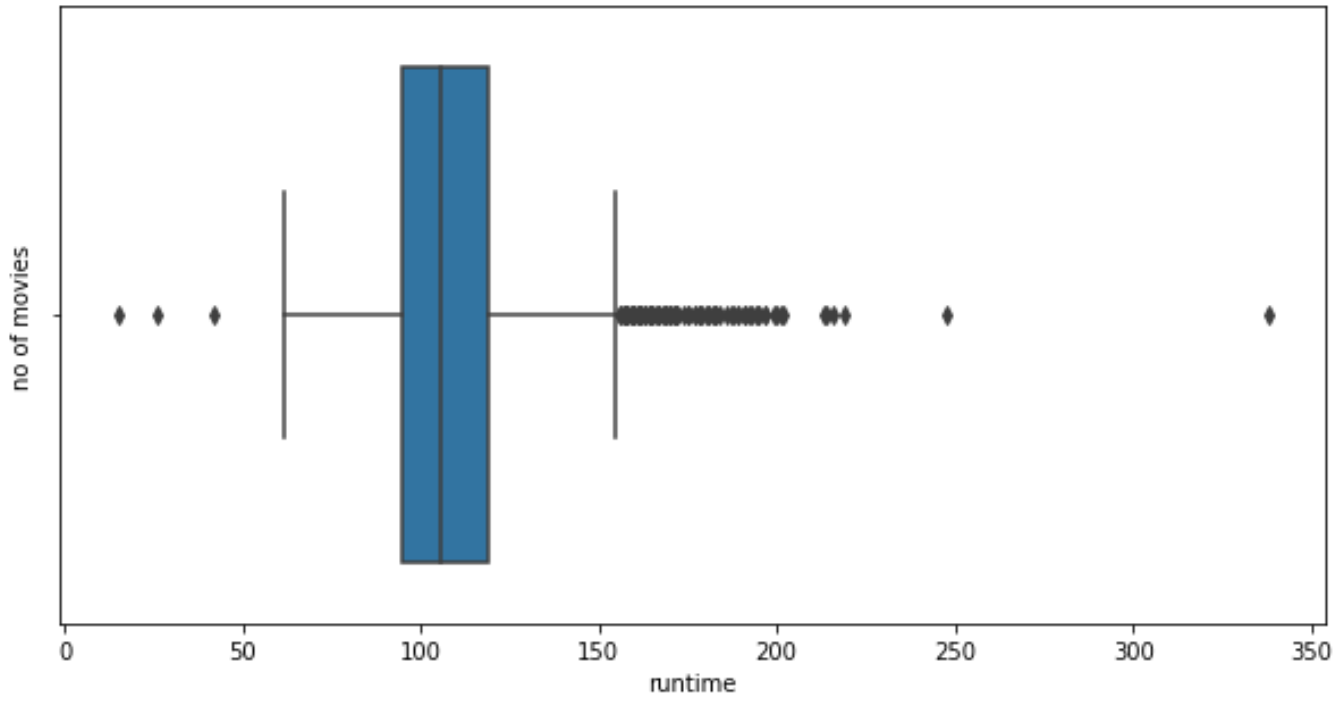
Question 5- average runtime of all movies

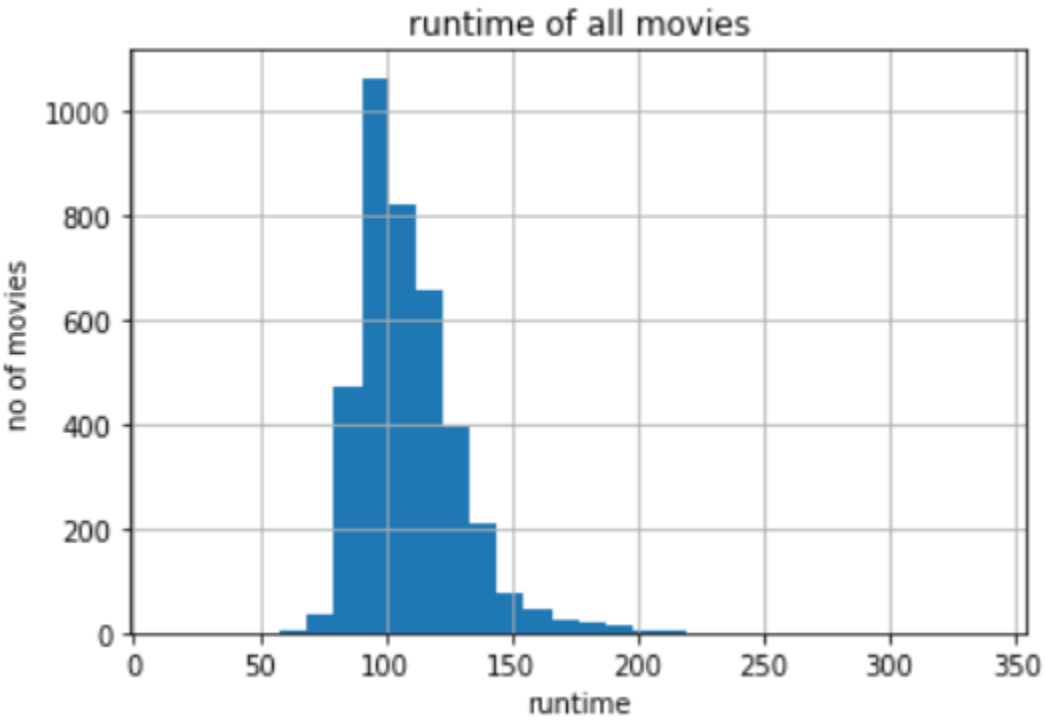
```
def average_run(column):
    return movie_data[column].mean()

# calling the function
average_run('runtime')
```

The results show that all movies had an average runtime of 109.22 minutes

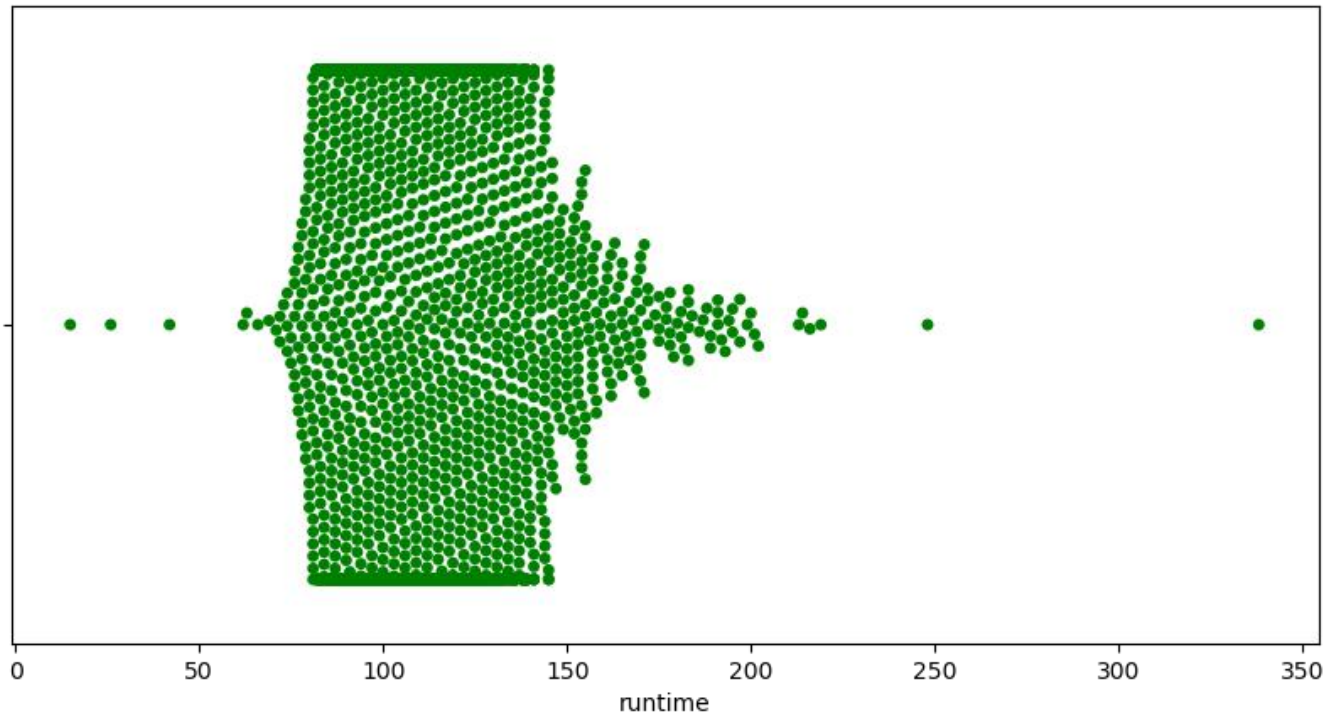
We also plotted a histogram, a boxplot & a swarmplot to get a bigger picture of runtime of all movies.

Since visualisation often helps in better , fair and more beautiful communication of results , We have also included a line plot between Year of release and total profitability so as to check which year has given us the most profits.




runtime of all movies
no of movies
runtime
0
200
400
600
800
1000
0
50
100
150
200
250
300
350

Another data point plot of runtime of movies:



The swarmplot generated above provides a complete visual of runtime of movies by plotting the points against their respective positions in the distribution.

Whereas, the boxplot provides us with an idea about how spreaded our distribution is in case of the runtime of the movies. It also provides us any outliers which are present in our data.

The bar graph is right skewed, i.e most of the movies are timed between 80-115 mnutes. There are almost 1000 movies which fall into this criteria.

We also used the .describe() function on the runtime column and with the help of this + the above visualisations, we can say that:

25% of the movies had a runtime of less than 95 minutes

50% of the movies had a runtime of less than 106 minutes

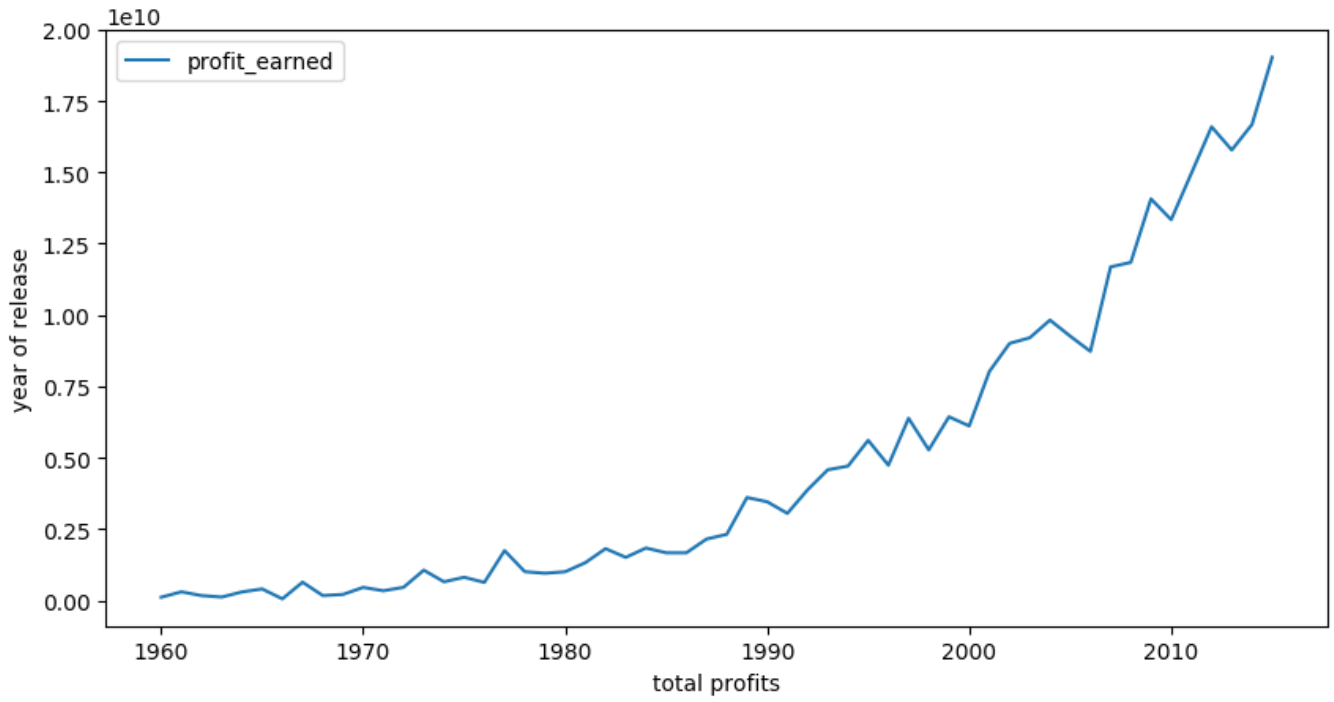
75% of the movies had a runtime of less than 119 minutes

Question 6- Which year saw the most profits(combined) ?

For this we generated a line plot between year of release & profitablity

 # year of release vs profitability

```
profits_earned_per_year =
movie_data.groupby('release_year')['profit_earned'].sum()

plt.figure(figsize = (10 , 5) , dpi = 100)

plt.plot(profits_earned_per_year)

plt.legend()

plt.xlabel('total profits')

plt.ylabel('year of release')

plt.show()
```



So we can graphically conclude that year 2015 was the time when movies made the most profit.

Now let's analyse our data in the light of the fact that we will consider only those movies under the profitable category which have earned a profit of more than $ 40 million

So we will clean out our data once again using the following code-

```
most_profitable_movies  =
movie_data[movie_data['profit_earned']
>=40000000] most_profitable_movies.index  =
range(len(most_profitable_movies))
```

Question 7- average budget of movies earning $40 million or more?

```
# we will define another function for calulating the avg. budget
def avg_profit(column):
    return most_profitable_movies[column].mean()

# We called the above function on :
avg_profit('budget')
```

Movies, which earned $40 million or more, were on an average budget of $57308877

Questiom 8- average runtime of movies earning $40 million or more?

```
# average duration of most profitable movies


avg_profit('runtime')
```

Movies, which earned $40 million or more, had an average runtime of 113 minutes

Limitations and Conclusions

Conclusions:

- Average runtime of a good movie should be around 113 minutes
- Average budget should be around $57 million

If one follows the above guidelines, his/her movie can become one of the hits and can earn a revenue of $236306323 or more

Limitations:

- The database hasn′t been updated from a long time which resulted in a decreased accuracy of the results
- The currency we took in our case i.e $ might be not the same for movies produced in different countries. Morever, it dosen’t accounts the money for inflation occurred all these years. All this may result in a completely different picture of the whole data.

References:

1- Stackoverflow.com for excellent material on idmax() & idmin() functions.

2- Python documentation for insights on all the used functions.

3- DataCamp.com for basic ideas on functions.