Project: TMDB 5000 Movie Dataset

Table of Contents

Introduction

Dataset Description

In this project, we will be analyzing data about TMDB movies. This dataset contains information about 5000 movies collected from The Movie Database (TMDb), including user ratings and revenue.

most colums in the dataset are self explonatory but there are certain columns, like 'genres' that contain multiple values separated by pipe (|) characters, but it's nothing to worry about we will clean it up. The orignal data can be found on kaggle https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata.

Question(s) for Analysis

In this project We are interested in exploring and answering the following questions using python libraries such as Pandas, numpy, and matplotlib etc:

Q1: Which movies have the highest budget spend on?

Q2: What are the top 5 highest profit making movies and the 5 lowest movies?

Q3: What is the relationships between, [Budget & Revenue], and [profit and revenue]?

Q4: Which year has the highest release of movies?

Q5: Which Month has the highest release movies?

Q6: Which months made the highest average profits?

Import Packages

In [163…

```python
### In this section we will import necessary packages we need to analyze the dataset

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

Data Wrangling

In this section of the report, we will load, access, and explore the data to understand the characteristics of our data.

In [164…

```python
# load the data and read the first five rows.

df = pd.read_csv("tmdb_5000_movies.csv")
df.head(2)
```

Out[164…

	budget	genres	homepage	id	keywords	original_language	original_title	overview	po
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139

In [165…

```python
# check the data dimensions
df.shape
```

Out[165…

```
(4803, 20)
```

In [166…

```python
# let's inspect the data and get a good understanding of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

observing the data dimensions and the info() function result, we see that we have 4803 row and 20 columns in our data. We also notice there are some columns that have some null values.for instance, the homage and tagline have a lot of missing value. The release date is missing 1 value and is an object(string) type, so in our next section on Data Cleaning we will work on this columns, drop the homage, tagline and the Overview column as they are not neccesary for our analysis. We also change the date datatype and fill or drop the missing values for release_date and runtime as they only have few null values

In [167…

```python
# Now let check our data statics using describe() and transpose() to make our data easy to read.
df.describe().transpose()
```

Out[167…

	count	mean	std	min	25%	50%	75%	max
budget	4803.0	2.904504e+07	4.072239e+07	0.0	790000.00000	1.500000e+07	4.000000e+07	3.800000e+08
id	4803.0	5.716548e+04	8.869461e+04	5.0	9014.50000	1.462900e+04	5.861050e+04	4.594880e+05
popularity	4803.0	2.149230e+01	3.181665e+01	0.0	4.66807	1.292159e+01	2.831350e+01	8.755813e+02
revenue	4803.0	8.226064e+07	1.628571e+08	0.0	0.00000	1.917000e+07	9.291719e+07	2.787965e+09
runtime	4801.0	1.068759e+02	2.261193e+01	0.0	94.00000	1.030000e+02	1.180000e+02	3.380000e+02
vote_average	4803.0	6.092172e+00	1.194612e+00	0.0	5.60000	6.200000e+00	6.800000e+00	1.000000e+01
vote_count	4803.0	6.902180e+02	1.234586e+03	0.0	54.00000	2.350000e+02	7.370000e+02	1.375200e+04

Observing the data summery we see we have some 0 values in budget, revenue and in runtime. we will treat this as missing data as and change it Nans

In [168…

```python
# replace 0 in budget, revenue, runtime

df['budget'] = df['budget'].replace(0, np.NaN)
df['revenue'] = df['revenue'].replace(0, np.NaN)
df['runtime'] = df['runtime'].replace(0, np.NaN)
```

In [169…

```python
df.describe().transpose()
```

Out[169…

	count	mean	std	min	25%	50%	75%	max
budget	3766.0	3.704284e+07	4.264651e+07	1.0	8.000000e+06	2.300000e+07	5.000000e+07	3.800000e+08
id	4803.0	5.716548e+04	8.869461e+04	5.0	9.014500e+03	1.462900e+04	5.861050e+04	4.594880e+05
popularity	4803.0	2.149230e+01	3.181665e+01	0.0	4.668070e+00	1.292159e+01	2.831350e+01	8.755813e+02
revenue	3376.0	1.170314e+08	1.834831e+08	5.0	1.535290e+07	5.175184e+07	1.401651e+08	2.787965e+09
runtime	4766.0	1.076607e+02	2.074942e+01	14.0	9.400000e+01	1.040000e+02	1.180000e+02	3.380000e+02
vote_average	4803.0	6.092172e+00	1.194612e+00	0.0	5.600000e+00	6.200000e+00	6.800000e+00	1.000000e+01
vote_count	4803.0	6.902180e+02	1.234586e+03	0.0	5.400000e+01	2.350000e+02	7.370000e+02	1.375200e+04

In [170…

```python
# check for duplicates
df.duplicated().sum()
```

Out[170…

```
0
```

Data Cleaning

In this section we will clean our data, remove unused colums, and change data types where applicable.

In [171…

```python
# drop uneccesary Columns
df.drop(["homepage", "tagline", "overview"],axis=1, inplace=True)
```

In [172…

```python
# confirm the columns are removed
for c in df.columns:
    print(c)
```

```
budget
genres
id
keywords
original_language
original_title
popularity
production_companies
production_countries
release_date
revenue
runtime
spoken_languages
status
title
vote_average
vote_count
```

In [173…

```python
# Convert the release date to datetime format
df["release_date"] = pd.to_datetime(df["release_date"])
```

In [174…

```python
# confirm
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   budget                3766 non-null   float64       
 1   genres                4803 non-null   object        
 2   id                    4803 non-null   int64         
 3   keywords              4803 non-null   object        
 4   original_language     4803 non-null   object        
 5   original_title        4803 non-null   object        
 6   popularity            4803 non-null   float64       
 7   production_companies  4803 non-null   object        
 8   production_countries  4803 non-null   object        
 9   release_date          4802 non-null   datetime64[ns]
 10  revenue               3376 non-null   float64       
 11  runtime               4766 non-null   float64       
 12  spoken_languages      4803 non-null   object        
 13  status                4803 non-null   object        
 14  title                 4803 non-null   object        
 15  vote_average          4803 non-null   float64       
 16  vote_count            4803 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(2), object(9)
memory usage: 638.0+ KB
```

In [175…

```python
release_date_mode = df["release_date"].mode()
```

In [176…

```python
release_date_mode
```

Out[176…

```
0   2006-01-01
Name: release_date, dtype: datetime64[ns]
```

In [177…

```python
# lets fill the missing release_date with mode date
df.fillna({'release_date': "2006-01-01"}, inplace=True)
df.isnull().sum()
```

Out[177…

```
budget                  1037
genres                     0
id                         0
keywords                   0
original_language          0
original_title             0
popularity                 0
production_companies       0
production_countries       0
release_date               0
revenue                 1427
runtime                   37
spoken_languages           0
status                     0
title                      0
vote_average               0
vote_count                 0
dtype: int64
```

In [178…

```python
# similarly lets fill the runtime missing values with the run time mean value
runt_mean = df["runtime"].mean()
```

In [179…

```python
runt_mean
```

Out[179…

```
107.66072177926983
```

In [180…

```python
# Get the runtime mean to replace the runtime missing value as well.
df.fillna({'runtime': runt_mean}, inplace=True)
```

In [181…

```python
# Check if we have any null values in our data
df.isnull().sum()
```

Out[181…

```
budget                  1037
genres                     0
id                         0
keywords                   0
original_language          0
original_title             0
popularity                 0
production_companies       0
production_countries       0
release_date               0
revenue                 1427
runtime                    0
spoken_languages           0
status                     0
title                      0
vote_average               0
vote_count                 0
dtype: int64
```

Now that we have cleaned data, we can move the next step EDA and answer our posted questions.

In [182…

```python
# 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.
```

Exploratory Data Analysis

Question 1. Which top 5 movies have the highes budget?

In [183…

```python
df.columns
```

Out[183…

```
Index(['budget', 'genres', 'id', 'keywords', 'original_language',
       'original_title', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

In [184…

```python
# in order to answer this question we will need to find out the top 5 movies with the highest budget.
# we probably don't need the whole columns to answer this question let take a subset of our data frame
# by creates a new data frame
df1 = df[["release_date", "title", "genres", "runtime", "popularity", "budget", "revenue"]]
df1
```

Out[184…

	release_date	title	genres	runtime	popularity	budget	revenue
0	2009-12-10	Avatar	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	162.0	150.437577	237000000.0	2.787965e+09
1	2007-05-19	Pirates of the Caribbean: At World's End	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	169.0	139.082615	300000000.0	9.610000e+08
2	2015-10-26	Spectre	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	148.0	107.376788	245000000.0	8.806746e+08
3	2012-07-16	The Dark Knight Rises	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	165.0	112.312950	250000000.0	1.084939e+09
4	2012-03-07	John Carter	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	132.0	43.926995	260000000.0	2.841391e+08
...	...	...	...	...	...	...	...
4798	1992-09-04	El Mariachi	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	81.0	14.269792	220000.0	2.040920e+06
4799	2011-12-26	Newlyweds	[{"id": 35, "name": "Comedy"}, {"id": 10749, "...	85.0	0.642552	9000.0	NaN
4800	2013-10-13	Signed, Sealed, Delivered	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	120.0	1.444476	NaN	NaN
4801	2012-05-03	Shanghai Calling	[]	98.0	0.857008	NaN	NaN
4802	2005-08-05	My Date with Drew	[{"id": 99, "name": "Documentary"}]	90.0	1.929883	NaN	NaN

4803 rows × 7 columns

In [185…

```python
# Let's get the top 10 movies with the highst budget spent on, sort decending and plot
highest_budget10 = df1.nlargest(n=5, columns="budget")
highest_bufget10 = df1.sort_values("budget", ascending=True)
```

In [186…

```python
highest_budget10
```

Out[186…

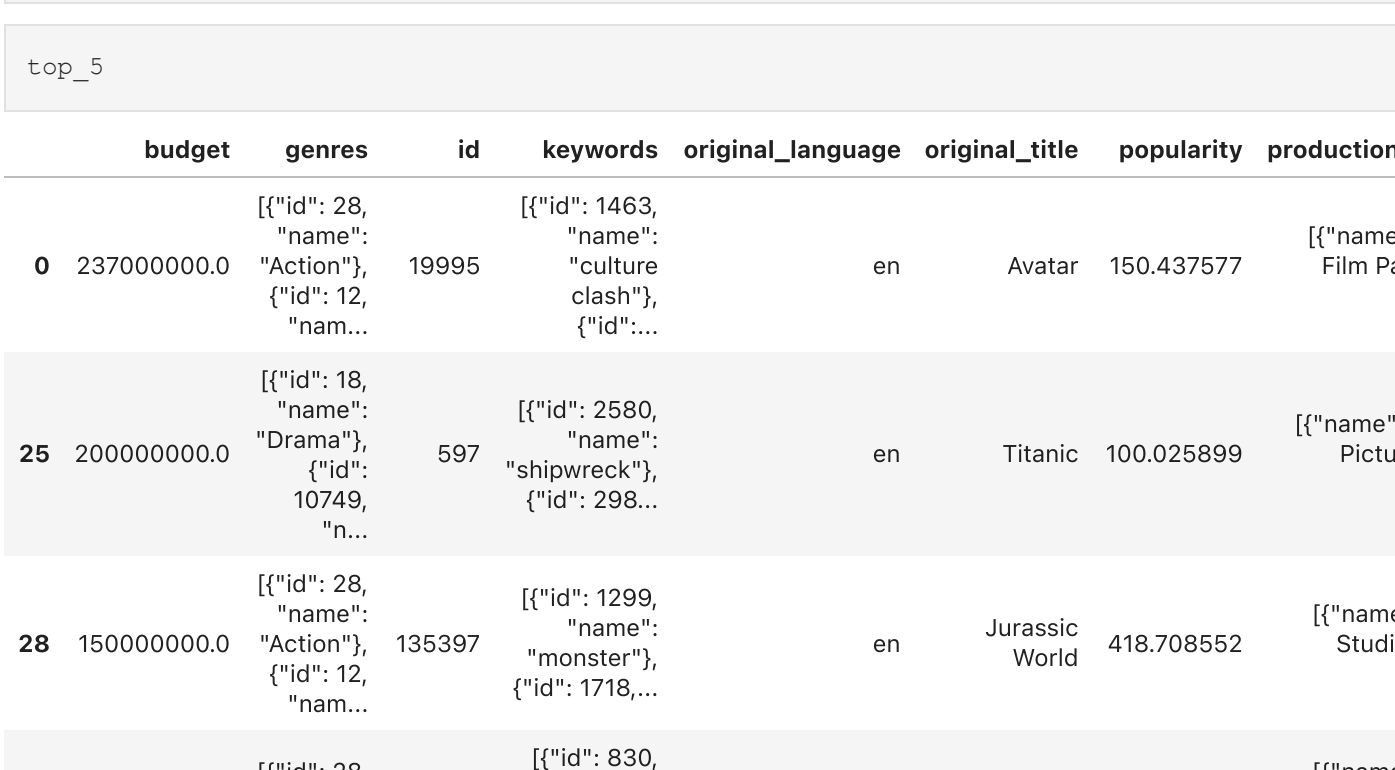
	release_date	title	genres	runtime	popularity	budget	revenue
17	2011-05-14	Pirates of the Caribbean: On Stranger Tides	[{"id": 12, "name": "Adventure"}, {"id": 28, "...	136.0	135.413856	380000000.0	1.045714e+09
1	2007-05-19	Pirates of the Caribbean: At World's End	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	169.0	139.082615	300000000.0	9.610000e+08
7	2015-04-22	Avengers: Age of Ultron	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	141.0	134.279229	280000000.0	1.405404e+09
10	2006-06-28	Superman Returns	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	154.0	57.925623	270000000.0	3.910812e+08
4	2012-03-07	John Carter	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	132.0	43.926995	260000000.0	2.841391e+08

In [187…

```python
plt.figure(figsize=(8,4))
# bar plot with matplotlib
plt.barh('title', 'budget',data=highest_budget10)
plt.xlabel("Budget value in Million USD Dollars", size=15)
plt.ylabel("Movie Title", size=15)
plt.title("Top 5 Moveies on TMDB movies", size=18)
```

Out[187…

```
Text(0.5, 1.0, 'Top 5 Moveies on TMDB movies')
```



Obeservations

."Pirates of the Caribbean: On Stranger Tides, \ . At world's end \ . Avengers: Age of Utron \ . Superman returns \ . John Carter are top 5 highes based the budget.

Question 2 what are the top 5 highest profit making movies and the 5 lowest movies

In [188…

```python
#calculate Profit for each of the movie
#This line of code is going to create a new column in our data called 'Profit'
df['profit'] = df['revenue'] - df['budget']
```

In [189…

```python
df1.columns
```

Out[189…

```
Index(['release_date', 'title', 'genres', 'runtime', 'popularity', 'budget',
       'revenue'],
      dtype='object')
```

In [190…

```python
# the top 5 hight profit making movies
top_5 = df.nlargest(5, "profit")
```

In [191…

```python
top_5
```

Out[191…

	budget	genres	id	keywords	original_language	original_title	popularity	production_companies	production_cou
0	237000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1" "name": " Sta
25	200000000.0	[{"id": 18, "name": "Drama"}, {"id": 10749, "n...	597	[{"id": 2580, "name": "shipwreck"}, {"id": 298...	en	Titanic	100.025899	[{"name": "Paramount Pictures", "id": 4}, {"na...	[{"iso_3166_1" "name": " Sta
28	150000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	135397	[{"id": 1299, "name": "monster"}, {"id": 1718,...	en	Jurassic World	418.708552	[{"name": "Universal Studios", "id": 13}, {"na...	[{"iso_3166_1" "name": " Sta
44	190000000.0	[{"id": 28, "name": "Action"}]	168259	[{"id": 830, "name": "car race"}, {"id": 3428,...	en	Furious 7	102.322217	[{"name": "Universal Pictures", "id": 33}, {"n...	[{"iso_3166_1 "name": "Ja {
16	220000000.0	[{"id": 878, "name": "Science Fiction"}, {"id"...	24428	[{"id": 242, "name": "new york"}, {"id": 5539,...	en	The Avengers	144.448633	[{"name": "Paramount Pictures", "id": 4}, {"na...	[{"iso_3166_1" "name": " Sta

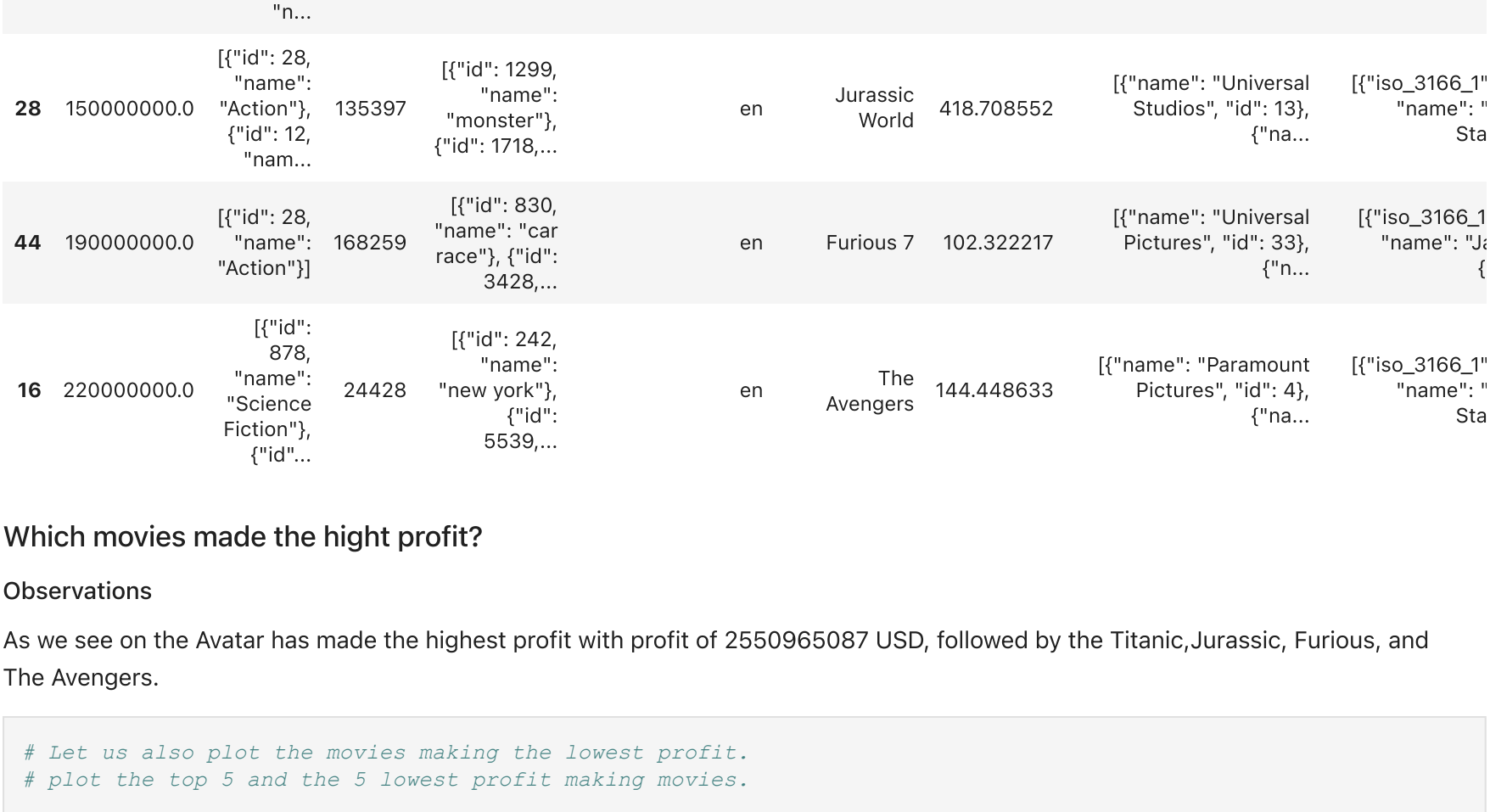
In [192…

```python
lowest_5 = df.nsmallest(5, "profit")
```

In [193…

```python
# plot the top 5 and the 5 lowest profit making movies.

labels = ["Profits", "Budget"]
plt.figure(figsize=(18, 8))
plt.plot(top_5["title"], top_5["profit"])
plt.plot(top_5["title"], top_5["budget"])
plt.xlabel("Movie Title")
plt.ylabel("Value in USD Dollar")
plt.title("Top Five & lowest profit movies")
plt.legend(labels = labels, fontsize= "large")
plt.show()
```



In [194…

```python
top_5
```

Out[194…

	budget	genres	id	keywords	original_language	original_title	popularity	production_companies	production_cou
0	237000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1" "name": " Sta
25	200000000.0	[{"id": 18, "name": "Drama"}, {"id": 10749, "n...	597	[{"id": 2580, "name": "shipwreck"}, {"id": 298...	en	Titanic	100.025899	[{"name": "Paramount Pictures", "id": 4}, {"na...	[{"iso_3166_1" "name": " Sta
28	150000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	135397	[{"id": 1299, "name": "monster"}, {"id": 1718,...	en	Jurassic World	418.708552	[{"name": "Universal Studios", "id": 13}, {"na...	[{"iso_3166_1" "name": " Sta
44	190000000.0	[{"id": 28, "name": "Action"}]	168259	[{"id": 830, "name": "car race"}, {"id": 3428,...	en	Furious 7	102.322217	[{"name": "Universal Pictures", "id": 33}, {"n...	[{"iso_3166_1 "name": "Ja {
16	220000000.0	[{"id": 878, "name": "Science Fiction"}, {"id"...	24428	[{"id": 242, "name": "new york"}, {"id": 5539,...	en	The Avengers	144.448633	[{"name": "Paramount Pictures", "id": 4}, {"na...	[{"iso_3166_1" "name": " Sta

Which movies made the hight profit?

Observations

As we see on the Avatar has made the highest profit with profit of 2550965087 USD, followed by the Titanic,Jurassic, Furious, and The Avengers.

In [195…

```python
# Let us also plot the movies making the lowest profit.
# plot the top 5 and the 5 lowest profit making movies.

labels = ["Profits", "Budget"]
plt.figure(figsize=(18, 8))
plt.plot(lowest_5["title"], lowest_5["profit"])
plt.plot(lowest_5["title"], lowest_5["budget"])
plt.xlabel("Movie Title")
plt.ylabel("Profits")
plt.title("Top Five & lowest profit movies")
plt.legend(labels = labels, fontsize= "large")
plt.show()
```

In [196…

```python
lowest_5
```

Out[196…

	budget	genres	id	keywords	original_language	original_title	popularity	production_companies	production_cou
13	255000000.0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	57201	[{"id": 1556, "name": "texas"}, {"id": 2673, "...	en	The Lone Ranger	49.046956	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "name": "l Stat
338	145000000.0	[{"id": 37, "name": "Western"}, {"id": 36, "na...	10733	[{"id": 1556, "name": "texas"}, {"id": 3776, "...	en	The Alamo	10.660441	[{"name": "Imagine Entertainment", "id": 23}, ...	[{"iso_3166_1": "name": "l Stat
141	150000000.0	[{"id": 12, "name": "Adventure"}, {"id": 16, "...	50321	[{"id": 5202, "name": "boy"}, {"id": 9951, "na...	en	Mars Needs Moms	12.362599	[{"name": "Walt Disney Animation Studios", "id...	[{"iso_3166_1": "name": "l Stat
208	160000000.0	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	1911	[{"id": 616, "name": "witch"}, {"id": 1964, "n...	en	The 13th Warrior	27.220157	[{"name": "Touchstone Pictures", "id": 9195}]	[{"iso_3166_1": "name": "l Stat
311	100000000.0	[{"id": 28, "name": "Action"}, {"id": 35, "nam...	11692	[{"id": 305, "name": "moon"}, {"id": 585, "nam...	en	The Adventures of Pluto Nash	12.092241	[{"name": "Village Roadshow Pictures", "id": 7...	[{"iso_3166_1": "name": "Austr

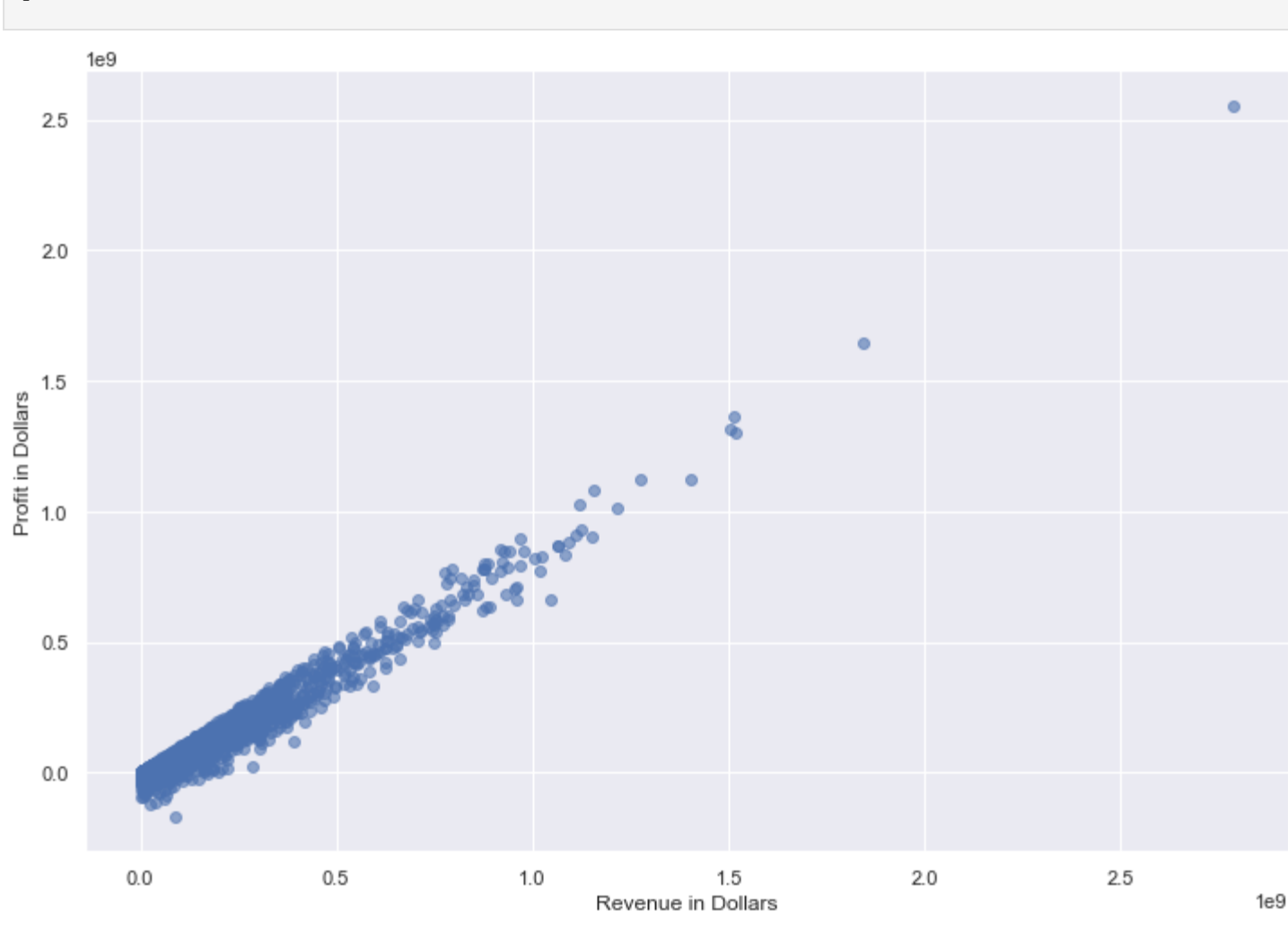
observations

Observing the graph we see that The Lone Ranger is the biggest loser, making the lowest profit of -16571009 USD; followed by: The Alamo, Mars needs Moms, the 13th warrior, the adventures of pluto Nash.

In [197…

```python
# Let us check out if there is a relationship between budget and Revenue

plt.figure(figsize=(12, 8))
plt.scatter(df["budget"],
            df["revenue"],
            alpha=0.7,
            )
plt.xlabel('Budget in Dollars')
plt.ylabel('Profit in Dollars')
plt.show()
```



Observing the data points there is no relationship between budget and profits

In [198…

```python
plt.figure(figsize=(12, 8))
plt.scatter(df["revenue"],
            df["profit"],
            alpha=0.6,
            )
# x-axis
plt.xlabel('Revenue in Dollars')
# y-axis
plt.ylabel('Profit in Dollars')
plt.show()
```

In [ ]:

Inspecting the scatter we can see the relationship between profit and revenue is strong relationship, thus the higher higher the revenue is the higher profit the movie will make.

Which year has the highest release of movies?

In [199…

```python
# to answer this question let's create a new column for the release by extracting the release date

df['year'] = df['release_date'].dt.year
df['year']
```

Out[199…

```
0       2009
1       2007
2       2015
3       2012
4       2012
        ... 
4798    1992
4799    2011
4800    2013
4801    2012
4802    2005
Name: year, Length: 4803, dtype: int64
```

In [200…

```python
# let's count how many movies are released each year by grouping them by year
count_movies=df.groupby('year').count()['id']
count_movies.head()
```

Out[200…

```
year
1916    1
1925    1
1927    1
1929    2
1930    1
Name: id, dtype: int64
```

In [201…

```python
# create a line chart
plt.figure(figsize=(20, 8))
count_movies.plot(xticks = np.arange(1916,2017,4))
plt.xlabel("Year")
plt.ylabel("Number of movies")
plt.show()
```

In [202…

```python
count_movies.tail(10)
```

Out[202…

```
year
2008    227
2009    247
2010    225
2011    223
2012    208
2013    231
2014    238
2015    216
2016    104
2017      1
Name: id, dtype: int64
```

Inspecting the plot and the tail function result, we see that year 2009 year has the highest release of movies with 247 movies released.

In which month has the highest Number of released movies?

In [203…

```python
# extract a new column for the month the movies are released.
df["released_month"] = df['release_date'].dt.month_name()
```

In [204…

```python
df.columns
```

Out[204…

```
Index(['budget', 'genres', 'id', 'keywords', 'original_language',
       'original_title', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'title', 'vote_average', 'vote_count',
       'profit', 'year', 'released_month'],
      dtype='object')
```

In [213…

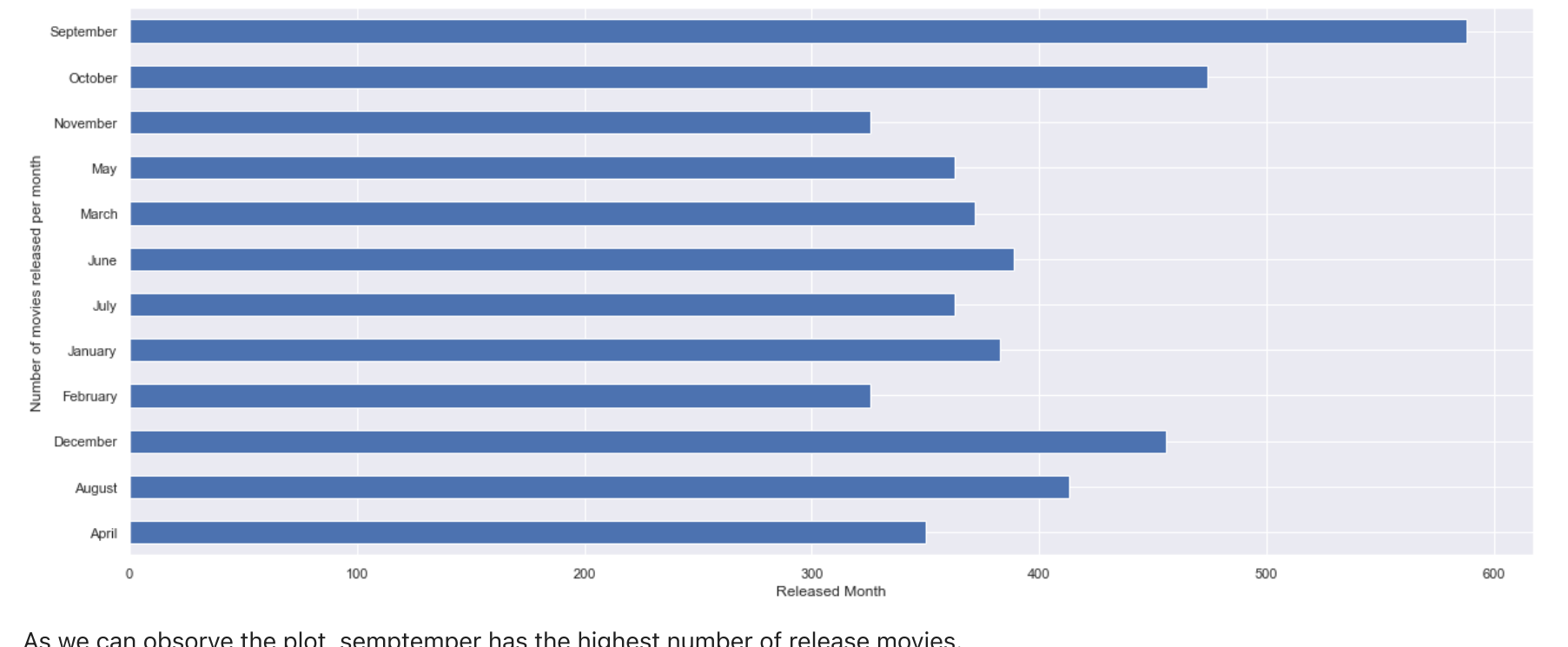
```python
# Count movies by month
count_movies_per_month = df.groupby('released_month').count()["id"]
count_movies_per_month
```

Out[213…

```
released_month
April        350
August       413
December     456
February     326
January      383
July         363
June         389
March        372
May          363
November     326
October      474
September    588
Name: id, dtype: int64
```

In [214…

```python
# plot the result
plt.figure(figsize=(20, 8))
count_movies_per_month.plot(kind='barh')
plt.xlabel("Released Month")
plt.ylabel("Number of movies released per month")
plt.show()
```



As we can obsorve the plot, semptemper has the highest number of release movies.

Which Month Made The Highest average profit?

In [222…

```python
df2 = df[["title", "released_month", "revenue", "profit"]]
df2
```

Out[222…

	title	released_month	revenue	profit
0	Avatar	December	2.787965e+09	2.550965e+09
1	Pirates of the Caribbean: At World's End	May	9.610000e+08	6.610000e+08
2	Spectre	October	8.806746e+08	6.356746e+08
3	The Dark Knight Rises	July	1.084939e+09	8.349391e+08
4	John Carter	March	2.841391e+08	2.413910e+07
...	...	...	...	...
4798	El Mariachi	September	2.040920e+06	1.820920e+06
4799	Newlyweds	December	NaN	NaN
4800	Signed, Sealed, Delivered	October	NaN	NaN
4801	Shanghai Calling	May	NaN	NaN
4802	My Date with Drew	August	NaN	NaN

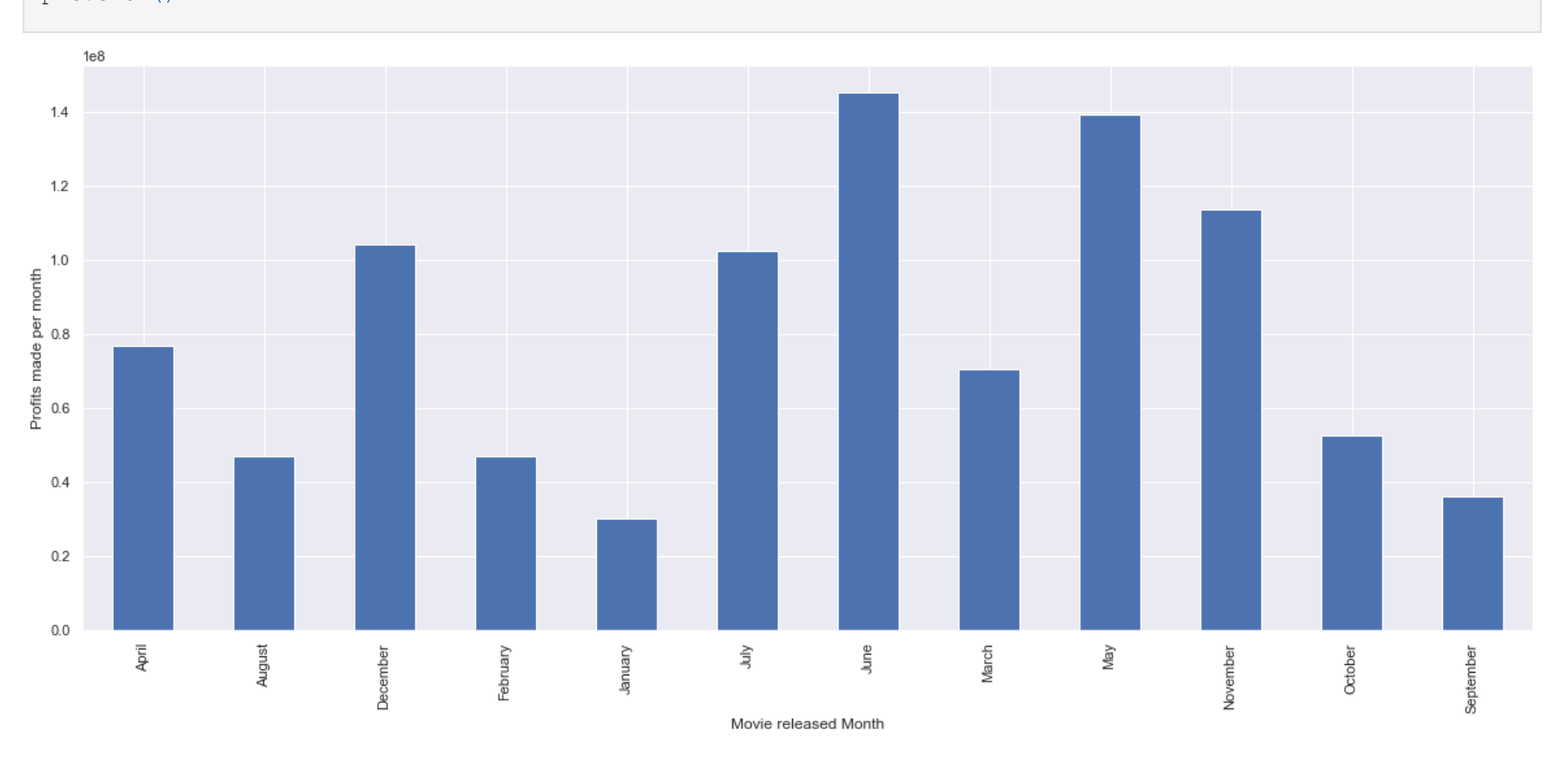
4803 rows × 4 columns

In [228…

```python
profit_per_month = df2.groupby("released_month")["profit"].mean()
```

In [230…

```python
# plot the result
plt.figure(figsize=(20, 8))
profit_per_month.plot(kind='bar')
plt.xlabel("Movie released Month")
plt.ylabel("Profits made per month")
plt.show()
```



Insights

Movies released on July made the highest profits, followed by Jun, May, thus we can say that those months are the best time to release movies.

Conclusions

Submitting your Project

> **Tip**: Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).
>
> **Tip**: Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.
>
> **Tip**: Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

The objective of this project was to practice and put to work for the lesson's we have learned first two weeks of the Udacity Data Analyst Nono degree.

The main goal for the project was to explore the TMDB 5000 Movie dataset, drive insights and answer the questions we posted in the description section and communicate our finds.

upon exploring the data we have answered the posted questions and make recommendations as follows.

Q1: Which movies have the highest budget spend on?

```
Top Highest budget movies:
. Pirates of the Caribbean: On Stranger Tides,
. At world's end
. Avengers: Age of Utron
. Superman returns
. John Carter
```

Q2: What are the top 5 highest profit making movies and the 5 lowest movies?

```
Highest Profit:
    . Avatar
    . titanic
    . Jurassic World Movie Title
    . furious 7, and The Avengers made the highest profits.

Lowest Profit:
  . Lone Ranger made the least profit, followed by: The Alamo,
  . Mars needs Moms,
  . the 13th warrior, and
  The adventures of pluto Nash.
```

Q3: What is the relationships between, [Budget & Revenue], and [profit and revenue]?

```
We found there is no relationship between budget and revenue, but but the relation between
profit and revenue is defined as a strong relationship the higher the revenue is the more
profit the movie will make.
```

Q4: Which year has the highest release of movies?

```
. 2009 has the highest released movies with 247 movies.
```

Q5: Which Month has the highest release movies?

```
. September
```

Q6: Which months made the highest average profits?

```
. Movies released on July made the highest profits, followed by Jun, May. So practically movies
  released during summer makes the most profits, so our recommendationt to release movies to
during this months
  to make the most profits.
```

In [247…

```python
from subprocess import call
call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

Out[247… 255

In [ ]: