In [1]:
```
import pandas as pd
import traceback
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
csv_movies_with_budget_df=pd.read_csv(r'./project_datasets/top-500-movies.c
sv', parse_dates=True)
csv_movies_with_budget_df.head()
```

Out[2]:

	rank	release_date	title	url	production_cost	domestic_gross	world
0	1	2019-04-23	Avengers: Endgame	/movie/Avengers-Endgame-(2019)#tab=summary	400000000	858373000	
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	/movie/Pirates-of-the-Caribbean-On-Stranger-Ti...	379000000	241071802	
2	3	2015-04-22	Avengers: Age of Ultron	/movie/Avengers-Age-of-Ultron#tab=summary	365000000	459005868	
3	4	2015-12-16	Star Wars Ep. VII: The Force Awakens	/movie/Star-Wars-Ep-VII-The-Force-Awakens#tab=...	306000000	936662225	
4	5	2018-04-25	Avengers: Infinity War	/movie/Avengers-Infinity-War#tab=summary	300000000	678815482	

Data Cleaning (Handling missing values and duplicate data: Removing unused information)

In this section, i will try to handle the problem of missing values, duplicated data and format:

- Remove the unused columns that are not useful in my analysis;
- Remove the movies having empty genres;
- Ensure there is no white spaces on the primay key, the movie title
- Remove duplicate rows
- Convert year of release as an int instead of float
- Changing format of release date into datetime format and budget_adj/revenue_adj format from float to int.

Step 1: Drop unused columns and duplicates or rows with null values

```
In [3]: #
        csv_movies_with_budget_df.drop(['url','mpaa'], axis=1, inplace=True)
        req_cols = ['title','genre']
        csv_movies_with_budget_df[req_cols] = csv_movies_with_budget_df[req_cols].a
        pply(lambda col:col.str.strip())
```

```
In [4]: # drop duplicates
        csv_movies_with_budget_df.drop_duplicates(inplace=True)
        # confirm correction by rechecking for duplicates in the data
        sum(csv_movies_with_budget_df.duplicated())
```

```
Out[4]: 0
```

```
In [5]: # drop rows with any null values
        csv_movies_with_budget_df.dropna(subset=['genre'],inplace=True)
        csv_movies_with_budget_df.dropna(inplace=True)
        # checks if any of columns in the data have null values - should print Fals
        e
        csv_movies_with_budget_df.isnull().sum().any()
```

```
Out[5]: False
```

Step 2: Convert 'release_date' type from str to datetime

```
In [6]: # Convert 'release_date' type from str to datetime
        csv_movies_with_budget_df['release_date']=pd.to_datetime(csv_movies_with_bu
        dget_df['release_date'])
        csv_movies_with_budget_df['release_date'].head()
```

```
Out[6]: 0   2019-04-23
        1   2011-05-20
        2   2015-04-22
        3   2015-12-16
        4   2018-04-25
        Name: release_date, dtype: datetime64[ns]
```

Check for Missing values in the dataset

```
In [7]: csv_movies_with_budget_df.isnull().sum()

Out[7]: rank              0
        release_date      0
        title             0
        production_cost   0
        domestic_gross    0
        worldwide_gross   0
        opening_weekend   0
        genre             0
        theaters          0
        runtime           0
        year              0
        dtype: int64
```

All titles are available Although the release date column for a value is null, the year column is not null Some genre is null

```
In [8]: csv_movies_with_budget_df.isnull().any(axis=1).sum()

Out[8]: 0
```

```
In [9]: csv_movies_with_budget_df[csv_movies_with_budget_df['genre'].isnull()].yea
        r.value_counts()

Out[9]: Series([], Name: count, dtype: int64)
```

Only 1 movie belonging to 2015 has an empty genre, the other years are not relevant for my analysis.

Duplicates in the dataset

```
In [10]: sum(csv_movies_with_budget_df.duplicated())

Out[10]: 0
```

Step 3: Check Number of unique values in the dataset*

In [11]:
```
# Returns the number of unique values in each column
csv_movies_with_budget_df.nunique()
```

Out[11]:
```
rank               474
release_date       454
title              472
production_cost     89
domestic_gross     474
worldwide_gross    474
opening_weekend    472
genre               10
theaters           404
runtime             94
year                30
dtype: int64
```

Step 4: Create Range Columns for all cost

In [12]:
```
# Final number of movies
rows, col = csv_movies_with_budget_df.shape
print('After cleaning, we now have only {} entries of movies.'.format(row
s))
```

```
After cleaning, we now have only 474 entries of movies.
```

In [13]:
```
#create the range function which takes 2 parameters, the divisor (convert t
o million) and the range
def create_ranges(df, col,div=1_000_000,step=5):
    cst_min,cst_max = df[col].agg(['min', 'max'])
    cost_range=[(start-step,start) for start in range(int(cst_min/div),int
(cst_max/div)+step,step)]
    labels=[f"{start}-{end}" for start, end in cost_range]
    # Define the bins
    bins=pd.IntervalIndex.from_tuples(cost_range)
    # Use "cut" method to group values into ranges
    return pd.cut(df[col].apply(lambda x : int(x/div)),
                        bins=bins, include_lowest=True
                        ).map(dict(zip(bins, labels)))
```

In [14]:
```
#The idea here is to make the cost more readable. For example, a production
cost of 396,554,223 converts to a range of 396-401 million
csv_movies_with_budget_df['prod_cost_range_million'] =create_ranges(csv_mov
ies_with_budget_df,"production_cost")
csv_movies_with_budget_df['worldwide_gross_range_million'] =create_ranges(c
sv_movies_with_budget_df,"worldwide_gross",step=10)
csv_movies_with_budget_df['domestic_gross_range_million'] =create_ranges(cs
v_movies_with_budget_df,"domestic_gross")
csv_movies_with_budget_df.head()
```

Out[14]:

	rank	release_date	title	production_cost	domestic_gross	worldwide_gross	opening_
0	1	2019-04-23	Avengers: Endgame	400000000	858373000	2797800564	35
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	379000000	241071802	1045713802	9
2	3	2015-04-22	Avengers: Age of Ultron	365000000	459005868	1395316979	19
3	4	2015-12-16	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2064615817	24
4	5	2018-04-25	Avengers: Infinity War	300000000	678815482	2048359754	25

In [15]:
```
#Ensure all rows are accounted for
csv_movies_with_budget_df[csv_movies_with_budget_df["prod_cost_range_millio
n"].isna()]
csv_movies_with_budget_df[csv_movies_with_budget_df["domestic_gross_range_m
illion"].isna()]
csv_movies_with_budget_df[csv_movies_with_budget_df["worldwide_gross_range_
million"].isna()]
```

Out[15]:

rank	release_date	title	production_cost	domestic_gross	worldwide_gross	opening_weeke

From the above, there are no rows with NAN

```
In [16]: # Count the occurrences of each  range
         csv_movies_with_budget_df.worldwide_gross_range_million.value_counts()
```

```
Out[16]: worldwide_gross_range_million
         163-173      15
         283-293      10
         363-373      10
         243-253      10
         393-403      10
                      ..
         1793-1803     0
         1803-1813     0
         1813-1823     0
         1823-1833     0
         1463-1473     0
         Name: count, Length: 290, dtype: int64
```

Step 5: Convert year column into a int datatype so year looks like 2019 instead of 2019.0

In [17]:
```python
csv_movies_with_budget_df["year"] = csv_movies_with_budget_df.year.astype
('int64')
csv_movies_with_budget_df
```

Out[17]:

	rank	release_date	title	production_cost	domestic_gross	worldwide_gross	openi
0	1	2019-04-23	Avengers: Endgame	400000000	858373000	2797800564	
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	379000000	241071802	1045713802	
2	3	2015-04-22	Avengers: Age of Ultron	365000000	459005868	1395316979	
3	4	2015-12-16	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2064615817	
4	5	2018-04-25	Avengers: Infinity War	300000000	678815482	2048359754	
...	...	...	...	...	...	...	
493	494	2008-02-14	The Spiderwick Chronicles	92500000	71195053	162839667	
494	495	2004-10-22	The Incredibles	92000000	261441092	631441092	
495	496	2013-02-06	A Good Day to Die Hard	92000000	67349198	304249198	
496	497	2004-04-09	The Alamo	92000000	22406362	23911362	
498	499	2013-12-19	The Secret Life of Walter Mitty	91000000	58236838	187861183	

474 rows × 14 columns

Step 6: Convert key column title to lowercase

In [18]:
```python
csv_movies_with_budget_df['title'] = csv_movies_with_budget_df.title.str.lo
wer()
```

In [19]:
```python
csv_movies_with_budget_df.reset_index(drop = True, inplace = True)
```

In [20]:
```
csv_movies_with_budget_df.columns
```

Out[20]:
```
Index(['rank', 'release_date', 'title', 'production_cost', 'domestic_gros
s',
       'worldwide_gross', 'opening_weekend', 'genre', 'theaters', 'runtim
e',
       'year', 'prod_cost_range_million', 'worldwide_gross_range_million',
       'domestic_gross_range_million'],
      dtype='object')
```

Step 7: Rearrange Columns

In [21]:
```
re_order_cols = ['rank', 'title','year', 'release_date', 'genre','prod_cost
_range_million', 'worldwide_gross_range_million',
                 'domestic_gross_range_million', 'production_cost', 'domest
ic_gross',
                 'worldwide_gross', 'opening_weekend', 'theaters', 'runtim
e' ]
for i,col in enumerate(re_order_cols):
    tmp = csv_movies_with_budget_df[col]
    csv_movies_with_budget_df.drop(labels=[col],axis=1,inplace=True)
    csv_movies_with_budget_df.insert(i,col,tmp)
```

In [22]:
```
csv_movies_with_budget_df.head()
```

Out[22]:

	rank	title	year	release_date	genre	prod_cost_range_million	worldwide_gross_ra
0	1	avengers: endgame	2019	2019-04-23	Action	396-401	
1	2	pirates of the caribbean: on stranger tides	2011	2011-05-20	Adventure	376-381	
2	3	avengers: age of ultron	2015	2015-04-22	Action	361-366	
3	4	star wars ep. vii: the force awakens	2015	2015-12-16	Adventure	301-306	
4	5	avengers: infinity war	2018	2018-04-25	Action	296-301	

Final data after cleanup

```
In [23]: csv_movies_with_budget_df.shape

Out[23]: (474, 14)
```

```
In [24]: csv_movies_with_budget_df.to_csv(r'./project_datasets/clean-500-movies.cs
         v',index=False)
```

Are there any legal or regulatory guidelines for your data or project topic?

None as this is just a top 500 movies with their production cost.

Did you make any assumptions in cleaning/transforming the data?

I assumed the rank of the movie was opinioned and not based on any logic. I do not plan to use that as part of my analysis.

How was your data sourced / verified for credibility?

This comes from the top 500 movies set from kaggle.

Was your data acquired in an ethical way?

Yes, it was just the top 500 movies.