This file was used to blend our financial data sources

In [54]:

```
import numpy as np #linear algebra
import pandas as pd #data processing
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', None)

# Importing into dataframe
df_Bom_MovieGross = pd.read_csv("./Prj_Data/DownLoadedData_FlatIron/bom.movi
df_TN_Movie_Budgets = pd.read_csv("./Prj_Data/DownLoadedData_FlatIron/tn.mov
df_TN_Movie_Budgets = pd.read_csv("./Prj_Data/DownLoadedData_FlatIron/tn.mov
df_Scrp_Financials = pd.read_excel("./Prj_Data/ImdbScrapingData/df_Financial
df_IMDB_Akas_english = pd.read_excel("./Prj_Data/DownLoadedData_Imdb/df_IMDB
df_IMDB_InflationAdjuster = pd.read_excel("./Prj_Data/ImdbScrapingData/Infla

```

In [55]:

```
#PREPARE THE FINACIAL DATA:  WE USED 2 SOURCES,1 FROM FLATIRON DOWNLOAD, 2 F
#*************************************************************************
#prepraring to work with Movie Gross Df, cleaning up data and create right d
df_Bom_MovieGross.name = "df_Bom_MovieGross"
df_Bom_MovieGross['foreign_gross'] = pd.to_numeric(df_Bom_MovieGross['foreig
df_Bom_MovieGross['year_str_BOM'] = df_Bom_MovieGross['year'].astype(str)
df_Bom_MovieGross['year'] = df_Bom_MovieGross['year'].astype(int)
df_Bom_MovieGross_values = {'domestic_gross':0, 'foreign_gross': 0}
df_Bom_MovieGross.fillna(value=df_Bom_MovieGross_values, inplace=True)
df_Bom_MovieGross["wwg_calc_BOM"] = 0
df_Bom_MovieGross['title'] = df_Bom_MovieGross['title'].str.title() #******K
df_Bom_MovieGross["titleyear"] = df_Bom_MovieGross['title'] + df_Bom_MovieGr
df_Bom_MovieGross["wwg_calc_BOM"] = (df_Bom_MovieGross['domestic_gross'] + d

#Renaming to aid in consolidating between the three sources
df_Bom_MovieGross.rename(columns={"foreign_gross": "fg_BOM", "domestic_gross
                                                  "title":"title_BOM","studio":"s
#-------------------------------------------------------------

#prepraring to work with Movie budgets, cleaning up data and create right da
df_TN_Movie_Budgets.name = "df_TN_Movie_Budgets"
df_TN_Movie_Budgets[df_TN_Movie_Budgets.columns[1:]] = df_TN_Movie_Budgets[d
df_TN_Movie_Budgets['production_budget'] = pd.to_numeric(df_TN_Movie_Budgets
df_TN_Movie_Budgets['domestic_gross'] = pd.to_numeric(df_TN_Movie_Budgets['d
df_TN_Movie_Budgets['worldwide_gross'] = pd.to_numeric(df_TN_Movie_Budgets['

df_TN_Movie_Budgets_values = {'domestic_gross': 0, 'year':0 , 'worldwide_gro
df_TN_Movie_Budgets.fillna(value=df_TN_Movie_Budgets_values, inplace=True)
df_TN_Movie_Budgets["fg_calc"] = 0

df_TN_Movie_Budgets['movie'] = df_TN_Movie_Budgets['movie'].str.title() #***
df_TN_Movie_Budgets["year"] = df_TN_Movie_Budgets['release_date'].str[-4:].a
df_TN_Movie_Budgets["year_str_TN"] = df_TN_Movie_Budgets['release_date'].str
df_TN_Movie_Budgets["titleyear"] = df_TN_Movie_Budgets['movie'] + df_TN_Movi
df_TN_Movie_Budgets["fg_calc_TN"] = (df_TN_Movie_Budgets['worldwide_gross']
df_TN_Movie_Budgets = df_TN_Movie_Budgets.drop('id', axis = 1)
df_TN_Movie_Budgets.rename(columns={"domestic_gross": "dg_TN", "worldwide_gr
                                    "production_budget": "pb_TN" ,"movie":"t

#needed to do this steep to collapse data given there are dups in a few movi
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
q3 = """SELECT titleyear as titleyear, max(title_TN) as title_TN, max(year_T
min(rd_TN) as rd_TN, Sum(pb_TN) as pb_TN,sum(dg_TN) as dg_TN, sum(wwg_TN) as
        FROM df_TN_Movie_Budgets
        GROUP BY titleyear, title_TN
        """
df_TN_Movie_Budgets = pysqldf(q3)
#-------------------------------------------------------------

# Finanally merger data from flatiron
df_fI_Financials = df_TN_Movie_Budgets.merge(df_Bom_MovieGross,
     on='titleyear', how='outer', indicator='Combing_FL_Financials', suffixe

# Add tconst Key to FLatIronTables then drop
df_FI_financials_With_tconst = df_fI_Financials.merge(df_IMDB_Akas_english[[
```

```
57
```

In [56]:
```
# df_FI_financials_With_tconst.info()
```

In [57]:
```
# df_FI_financials_With_tconst.nunique()
```

In [58]:
```
#USEDTO CHECK FOR MERGING ERRORS
# # df_FI_financials_With_tconst[["title_TN","title_BOM","title_FI",]].sort_
# df_FI_financials_With_tconst[["year_TN","year_BOM","year_str_FI",]].sort_v
# df_FI_financials_With_tconst[["year_TN","year_BOM","year_FI",]].sort_value
# # df_Bom_MovieGross[["studio","studio_short","Studio_Desc"]].sort_values(b
```

In [59]:
```
#Use all data scraped from IMDB as the default for domestic, foreign ww sale
df_FI_financials_With_tconst.title_TN.fillna("0", inplace=True)
df_FI_financials_With_tconst.year_str_TN.fillna("0", inplace=True)
df_FI_financials_With_tconst.year_TN.fillna(0, inplace=True)

df_FI_financials_With_tconst.title_BOM.fillna("0", inplace=True)
df_FI_financials_With_tconst.year_str_BOM.fillna("0", inplace=True)
df_FI_financials_With_tconst.year_BOM.fillna(0, inplace=True)

df_FI_financials_With_tconst['title_FI'] = df_FI_financials_With_tconst['tit
df_FI_financials_With_tconst['year_str_FI'] = df_FI_financials_With_tconst['
df_FI_financials_With_tconst['year_FI'] = df_FI_financials_With_tconst['year

df_FI_financials_With_tconst['title_FI'] = df_FI_financials_With_tconst.titl
df_FI_financials_With_tconst['year_str_FI'] = df_FI_financials_With_tconst.y
df_FI_financials_With_tconst['year_FI'] = df_FI_financials_With_tconst.year_

df_FI_financials_With_tconst['title_FI'] = df_FI_financials_With_tconst.appl
df_FI_financials_With_tconst['year_str_FI'] = df_FI_financials_With_tconst.a
df_FI_financials_With_tconst['year_FI'] = df_FI_financials_With_tconst.apply

```

In [60]:
```
df_FI_financials_With_tconst.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2609 entries, 0 to 2608
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   titleyear              2609 non-null   object
 1   title_TN               2609 non-null   object
 2   year_TN                2609 non-null   float64
 3   year_str_TN            2609 non-null   object
 4   rd_TN                  1983 non-null   object
 5   pb_TN                  1983 non-null   float64
 6   dg_TN                  1983 non-null   float64
 7   wwg_TN                 1983 non-null   float64
 8   fg_calc_TN             1983 non-null   float64
 9   title_BOM              2609 non-null   object
 10  studio_BOM             1646 non-null   object
 11  dg_BOM                 1647 non-null   float64
 12  fg_BOM                 1647 non-null   float64
 13  year_BOM               2609 non-null   float64
 14  year_str_BOM           2609 non-null   object
 15  wwg_calc_BOM           1647 non-null   float64
 16  Combing_FL_Financials  2609 non-null   category
 17  tconst                 2609 non-null   object
 18  Adding tconst          2609 non-null   category
 19  title_FI               2609 non-null   object
 20  year_str_FI            2609 non-null   object
 21  year_FI                2609 non-null   float64
dtypes: category(2), float64(10), object(10)
memory usage: 433.3+ KB
```

In [61]:
```
# df_FI_financials_With_tconst.sort_values(by="year_FI").head(100)
```

In [62]:
```
#DROP UNNEEDED FIELDS
df_FI_financials_With_tconst.drop(["year_TN","year_BOM","year_TN","year_BOM"
```

In [63]:
```
# df_Scrp_Financials.info()
```

In [ ]:
```

```

In [64]:
```
#Merg flatiron financials with scraping finacials IMDB Site :https://www.box
#*************************************************************************
#prepraring to work with Movie Gross Df, cleaning up data and create right d
df_MasterFinancials = df_Scrp_Financials.merge(df_FI_financials_With_tconst,
                                               on='tconst', how='outer', suf
fieldsToConvert = {'dg_IMDB': 0, 'fg_IMDB':0 , 'ww_IMDB': 0, 'Domestic Openi
                   'dg_TN': 0, 'wwg_TN':0 , 'fg_calc_TN': 0,'dg_BOM': 0,
                  'wg_calc_BOM': 0}
df_MasterFinancials.fillna(value=fieldsToConvert, inplace=True)
```

In [65]:
```
# df_MasterFinancials.info()
```

In [66]:

```
#Clean up dataframe - consolidate, remove unwanted columns, all remove metad
df_MasterFinancials = df_MasterFinancials.reindex(sorted(df_MasterFinancials
col_names = ['tconst','Adding tconst', 'Combing_FL_Financials',
       'Genres_IMDB', 'MPAA', 'Running Time_IMDB', 'Unnamed: 0', 'Domestic O
       'dg_IMDB', 'dg_TN', 'fg_BOM', 'fg_IMDB', 'fg_calc_TN', 'genres',
       'isAdult', 'mergingfinancials', 'originalTitle', 'pb_IMDB', 'pb_TN',
       'rd_IMDB', 'rd_TN', 'runtimeMinutes', 'studio_BOM', 'studio_IMDB',
        'title_BOM', 'title_IMDB', 'title_TN', 'titleyear_IM',
       'titleyear_fl', 'wwg_calc_BOM', 'ww_IMDB', 'wwg_TN',
       'year', 'year_BOM', 'year_TN']
df_MasterFinancials.reindex(columns=col_names)


# df_MasterFinancials.drop(["Adding tconst", 'Unnamed: 0', 'isAdult', 'mergi

df_MasterFinancials_Only = df_MasterFinancials[['tconst','Domestic Opening',
                                                'fg_BOM', 'fg_calc_TN','fg_I
                                                'title_BOM', 'title_IMDB', '
                                                'wwg_calc_BOM', 'wwg_TN','ww

```

In [67]:

```
#Use all data scraped from IMDB as the default for domestic, foreign ww sale
df_MasterFinancials_Only['WW_Gross'] = df_MasterFinancials_Only['ww_IMDB'].a
df_MasterFinancials_Only['Dom_Gross'] = df_MasterFinancials_Only['dg_IMDB'].
df_MasterFinancials_Only['Frgn_Gross'] = df_MasterFinancials_Only['fg_IMDB']
df_MasterFinancials_Only['P_Cost'] = df_MasterFinancials_Only['pb_IMDB'].app
df_MasterFinancials_Only['year'] = df_MasterFinancials_Only['year'].apply(la



#fill in the blanks from max valus from the flatIron finance tables
df_MasterFinancials_Only['dg_Fl_Max'] = df_MasterFinancials_Only[["dg_BOM",
df_MasterFinancials_Only['fg_Fl_Max'] = df_MasterFinancials_Only[["fg_BOM",
df_MasterFinancials_Only['ww_Fl_Max'] = df_MasterFinancials_Only['dg_Fl_Max'

df_MasterFinancials_Only['WW_Gross'] = df_MasterFinancials_Only.apply(lambda
df_MasterFinancials_Only['Dom_Gross'] = df_MasterFinancials_Only.apply(lambd
df_MasterFinancials_Only['Frgn_Gross'] = df_MasterFinancials_Only.apply(lamb
df_MasterFinancials_Only['P_Cost'] = df_MasterFinancials_Only.apply(lambda x
df_MasterFinancials_Only['P_Cost']=df_MasterFinancials_Only['P_Cost'].replac

df_MasterFinancials_Only['year'] = df_MasterFinancials_Only['year'].fillna(0
df_MasterFinancials_Only['year'] = df_MasterFinancials_Only.apply(lambda x:



df_MasterFinancials_Only['titleyear'] = df_MasterFinancials_Only["titleyear_
df_MasterFinancials_Only.titleyear.fillna(df_MasterFinancials_Only["titleyea

# df_MasterFinancials_MetaData['titleyear'] = df_MasterFinancials_Only["titl
# df_MasterFinancials_MetaData.titleyear.fillna(df_MasterFinancials_Only["ti



```

```
<ipython-input-67-e68ce41ead83>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_MasterFinancials_Only['WW_Gross'] = df_MasterFinancials_Only['ww_IMDB'].
apply(lambda x: x if x>0 else 0)
<ipython-input-67-e68ce41ead83>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df_MasterFinancials_Only['Dom_Gross'] = df_MasterFinancials_Only['dg_IMD
```

In [68]:
```
1 df_MasterFinancials_Only.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5540 entries, 0 to 5539
Data columns (total 28 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   tconst            5540 non-null   object
 1   Domestic Opening  5540 non-null   float64
 2   dg_BOM            5540 non-null   float64
 3   dg_TN             5540 non-null   float64
 4   dg_IMDB           5540 non-null   float64
 5   fg_BOM            5540 non-null   float64
 6   fg_calc_TN        5540 non-null   float64
 7   fg_IMDB           5540 non-null   float64
 8   pb_IMDB           5540 non-null   float64
 9   pb_TN             5540 non-null   float64
 10  title_BOM         2609 non-null   object
 11  title_IMDB        4920 non-null   object
 12  title_TN          2609 non-null   object
 13  titleyear_IM      4920 non-null   object
 14  titleyear_fl      2609 non-null   object
 15  wwg_calc_BOM      5540 non-null   float64
 16  wwg_TN            5540 non-null   float64
 17  ww_IMDB           5540 non-null   float64
 18  year              5540 non-null   float64
 19  year_FI           2609 non-null   float64
 20  WW_Gross          5540 non-null   float64
 21  Dom_Gross         5540 non-null   float64
 22  Frgn_Gross        5540 non-null   float64
 23  P_Cost            5540 non-null   float64
 24  dg_Fl_Max         5540 non-null   float64
 25  fg_Fl_Max         5540 non-null   float64
 26  ww_Fl_Max         5540 non-null   float64
 27  titleyear         5540 non-null   object
dtypes: float64(21), object(7)
memory usage: 1.2+ MB
```

In [69]:
```
1 # df_MasterFinancials_Only.sort_values(by="year").head(100)
```

In [70]:
```
1 df_MasterFinancials_Only = df_MasterFinancials_Only[['tconst','P_Cost','Dom_
```

In [71]:
```
1 #_____Add Justments for Inflation and Calculate ROI_____
```

In [ ]:
```
1
```

In [72]:
```
1 df_MasterFinancials_Only = df_MasterFinancials_Only.merge(df_IMDB_InflationA
```

In [73]:
```
1 df_MasterFinancials_Only.head()
```

Out[73]:

	tconst	P_Cost	Dom_Gross	Frgn_Gross	WW_Gross	Domestic Opening	year	TicketPrice
0	tt0120667	100000000.0	154696080.0	178839854.0	333535934.0	56061504.0	2005.0	6.41
1	tt0121164	30000000.0	53359111.0	64731725.0	118090836.0	388166.0	2005.0	6.41
2	tt0121766	113000000.0	380270577.0	488119983.0	868390560.0	108435841.0	2005.0	6.41
3	tt0200465	20000000.0	30060660.0	34767761.0	64828421.0	5935256.0	2008.0	7.18
4	tt0206634	76000000.0	35552383.0	35043081.0	70595464.0	501003.0	2006.0	6.55

In [74]:
```
1 df_MasterFinancials_Only.columns
```

Out[74]:
```
Index(['tconst', 'P_Cost', 'Dom_Gross', 'Frgn_Gross', 'WW_Gross',
       'Domestic Opening', 'year', 'TicketPrice', 'EstInflation',
       'Multiplier'],
      dtype='object')
```

In [75]:
```
1 df_MasterFinancials_Only["adj_P_Cost"] = df_MasterFinancials_Only["P_Cost"]
2 df_MasterFinancials_Only["adj_Frgn_Gross"] = df_MasterFinancials_Only["Frgn_
3 df_MasterFinancials_Only["adj_WW_Gross"] = df_MasterFinancials_Only["WW_Gros
4 df_MasterFinancials_Only["Profits"] = df_MasterFinancials_Only["WW_Gross"] -
5 df_MasterFinancials_Only["adj_Profits"] = df_MasterFinancials_Only["Profits"
6
7 df_MasterFinancials_Only["ROI"] = df_MasterFinancials_Only.apply(lambda row
8
```

In [76]:
```
1 df_MasterFinancials_Only.head()
```

Out[76]:

	tconst	P_Cost	Dom_Gross	Frgn_Gross	WW_Gross	Domestic Opening	year	TicketPrice
0	tt0120667	100000000.0	154696080.0	178839854.0	333535934.0	56061504.0	2005.0	6.41
1	tt0121164	30000000.0	53359111.0	64731725.0	118090836.0	388166.0	2005.0	6.41
2	tt0121766	113000000.0	380270577.0	488119983.0	868390560.0	108435841.0	2005.0	6.41
3	tt0200465	20000000.0	30060660.0	34767761.0	64828421.0	5935256.0	2008.0	7.18
4	tt0206634	76000000.0	35552383.0	35043081.0	70595464.0	501003.0	2006.0	6.55

In [77]:
```
1 df_MasterFinancials_Only.to_excel("df_Movie_Financials.xlsx")
```

In [ ]:
```
1
```

In [ ]:

```
#THIS CODE IS FOR CHECKING THE SYNCHING STEP
# df_MasterFinancials_MetaData[["title_TN","title_BOM","title_IMDB","title_f
# df_MasterFinancials_MetaData[["genres_fin","genres",]].sort_values(by="gen
# df_MasterFinancials_MetaData[["year_BOM","year_TN","year"]].sort_values(by
# df_MasterFinancials_MetaData[["titleyear_fl","titleyear_IM", "titleyear_fi
# df_MasterFinancials_MetaData[["Running Time_IMDB","runtimeMinutes", "Runni
#df_MasterFinancials_MetaData[["rd_IMDB","rd_TN", "rd_fin", "rd_fin1", "rd_f

```