This File was used to get our baseline financial data from imdb, we blended this data with the flatiron financial data

In [ ]:
```
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)

```

In [ ]:
```
df_IMDB_Title_Akas = pd.read_csv("./Prj_Data/DownLoadedData_Imdb/imdb_title_
df_IMDB_Title_Akas.name = "df_IMDB_Title_Akas"

df_IMDB_Title_Akas = df_IMDB_Title_Akas.loc[(df_IMDB_Title_Akas['titleType']
df_IMDB_Title_Akas = df_IMDB_Title_Akas.loc[(df_IMDB_Title_Akas['isAdult']==
df_IMDB_Title_Akas = df_IMDB_Title_Akas.loc[~(df_IMDB_Title_Akas['genres']==
df_IMDB_Title_Akas.drop(['endYear', 'isAdult'], axis=1, inplace = True)

df_IMDB_Title_Akas["startYear"] = df_IMDB_Title_Akas.startYear.replace(r'\N'
df_IMDB_Title_Akas['startYear'] = df_IMDB_Title_Akas['startYear'].astype('in


df_IMDB_Title_Akas["runtimeMinutes"] = df_IMDB_Title_Akas.runtimeMinutes.rep
df_IMDB_Title_Akas['runtimeMinutes'] = df_IMDB_Title_Akas['runtimeMinutes'].

# df_IMDB_Title_Akas = df_IMDB_Title_Akas[df_IMDB_Title_Akas.primaryTitle.st
df_IMDB_Title_Akas.dropna(subset=['primaryTitle'], how='all', inplace=True)
df_IMDB_Title_Akas.fillna({"startYear":0,"runtimeMinutes":0}, inplace=True)
df_IMDB_Title_Akas = df_IMDB_Title_Akas.loc[(df_IMDB_Title_Akas['startYear']


df_IMDB_Title_Akas = df_IMDB_Title_Akas[df_IMDB_Title_Akas['primaryTitle'].s
df_IMDB_Title_Akas = df_IMDB_Title_Akas.loc[~(df_IMDB_Title_Akas['primaryTit


df_IMDB_Title_Ratings = pd.read_csv("./Prj_Data/DownLoadedData_Imdb/Ratings.
df_Imdb_MoviesWithRatings = df_IMDB_Title_Akas.merge(df_IMDB_Title_Ratings,

```

In [ ]:

```
#____________________________________________Indexes To Feed Into Scraping

#df_ttTolookup = pd.read_excel("df_ttTolookup.xlsx")

#df_0_7776 = df_ttTolookup.iloc[0:7776,]
        #df_0_7776.index = df_ttTolookup.iloc[0:7776,].index last row to com
#df_11236_14000 = df_ttTolookup.iloc[11236:14000,]
#df_11236_14000["index_O"] = df_ttTolookup.iloc[11236:14000,].index
# df_14000_19999 = df_ttTolookup.iloc[14000:20000,]
# df_14000_19999["index_O"] = df_ttTolookup.iloc[14000:20000,].index

#df_20000_24999v1 = df_ttTolookup_Backup.iloc[20000:25000,]
# df_20000_24999v1["index_O"] = df_ttTolookup_Backup.iloc[20000:25000,].inde


# df_20000_21910 = df_ttTolookup.iloc[20000:21910,]
# df_21910_24999["index_O"] = df_ttTolookup.iloc[20000:21910,].index

# df_21909_24999 = df_ttTolookup.iloc[21910:25000,]
# df_21909_24999["index_O"] = df_ttTolookup.iloc[21910:25000,].index

# df_25000_29999 = df_ttTolookup.iloc[21910:30000,]
# df_25000_29999["index_O"] = df_ttTolookup.iloc[21910:30000,].index

# df_29290_36177 = df_ttTolookup.iloc[29290:36177,]
# df_29290_36177["index_O"] = df_ttTolookup.iloc[29290:36177,].index


# df_36177_39999 = df_ttTolookup.iloc[36177:40000,]
# df_36177_39999["index_O"] = df_ttTolookup.iloc[36177:40000,].index

# df_40000_40629 = df_ttTolookup.iloc[40000:40629,]
# df_40000_40629["index_O"] = df_ttTolookup.iloc[40000:40629,].index

# df_40628_45000 = df_ttTolookup.iloc[40629:45000,]
# df_40628_45000["index_O"] = df_ttTolookup.iloc[40629:45000,].index

# df_45000_47252 = df_ttTolookup.iloc[45000:47252,]
# df_45000_47252["index_O"] = df_ttTolookup.iloc[45000:47252,].index

# df_47252_47778 = df_ttTolookup.iloc[47252:47778,]
# df_47252_47778["index_O"] = df_ttTolookup.iloc[47252:47778,].index

# df_47778_48108 = df_ttTolookup.iloc[47778:48108,]
# df_47778_48108["index_O"] = df_ttTolookup.iloc[47778:48108,].index

# df_48108_49999 = df_ttTolookup.iloc[48108:50000,]
# df_48108_49999["index_O"] = df_ttTolookup.iloc[48108:50000,].index

# df_50000_51418 = df_ttTolookup.iloc[50000:51418,]
# df_50000_51418["index_O"] = df_ttTolookup.iloc[50000:51418,].index


# df_51418_52224 = df_ttTolookup.iloc[51418:52224,]
# df_51418_52224["index_O"] = df_ttTolookup.iloc[51418:52224,].index


# df_52224_52561 = df_ttTolookup.iloc[52224:52561,]
# df_52224_52561["index_O"] = df_ttTolookup.iloc[52224:52561,].index

# df_52561_55000 = df_ttTolookup.iloc[52561:55000,]
# df_52561_55000["index_O"] = df_ttTolookup.iloc[52561:55000,].index

# df_55000_60000 = df_ttTolookup.iloc[55000:60000,]
# df_55000_60000["index_O"] = df_ttTolookup.iloc[55000:60000,].index


# df_60000_63054 = df_ttTolookup.iloc[60000:,]
# df_60000_63054["index_O"] = df_ttTolookup.iloc[60000:,].index




```

In [ ]:

```
# step 1 get top line
from bs4 import BeautifulSoup as bs
import requests as rq # get url
import re

df_Summary_financials = pd.DataFrame(columns = ["tconst", "Index_O", "Domest
df_Summary_details = pd.DataFrame(columns = ["tconst", "Domestic Distributor
                                             "Earliest Release Date", "MPAA"
#if True:
#df_ttToLookup

for index, row in df_60000_63054.iterrows():
    #print(index)
    #print (row)

#for index, row in df_IMDB_Title_Akas.iterrows():
    #print(row["tconst"])
    #print(row["index_O"])
    #break
#     #current_ttconst = 'tt2527338'

    current_ttconst = row["tconst"]
    ddToLookupIndex = row["index_O"]

    summaryheaders = []
    summaryfinancials = []
    summarydata = {}

    detailheaders = []
    detaildata = []
    detailsummary = {}

    the_getString = 'https://www.boxofficemojo.com/title/'+current_ttconst
    r=rq.get(the_getString)
    p=bs(r.text,'html.parser')

    summaryheaders.append("tconst")
    summaryheaders.append("index")

    summaryfinancials.append(current_ttconst)
    summaryfinancials.append(ddToLookupIndex)

    # get Summary data
    b=p.find('div', class_="a-section a-spacing-none mojo-performance-summar
    if b:
        b=p.find('div', class_="a-section a-spacing-none mojo-performance-su
        divs=b.find_all('div', class_="a-section a-spacing-none")
        if divs:
        #append keys
            for div in divs:
                spans = div.find_all('span', class_=["a-size-small","money"]
                if spans:
                    for span in spans:
                        #print(span.text.strip())

                        #remove () and from values so you can have consisten
                        thedata = re.sub(r'\([^)]*\)', '', span.text.strip()

                        # must be a # remove $ signes so we can sum later
                        if re.sub('[^0-9]',"", thedata):
                            finanicals=re.sub('[^0-9-]',"", thedata)
                            finanicals = int(finanicals)
                            summaryfinancials.append(finanicals)
                        else :
                            #if no # in detected above must be a header so i
                            summaryheaders.append(thedata)

    summarydata = dict(zip(summaryheaders, summaryfinancials))
    df_Summary_financials = df_Summary_financials.append(summarydata, ignore
            #print(type(summarydata))
```

In [ ]:

```
df_ttTolookup = pd.read_excel("df_ttTolookup_DomensticWithSales.xlsx")
```

In [ ]:

```
#___________________________________________________________

# step 2 get detailed data
from bs4 import BeautifulSoup as bs
import requests as rq # get url
import re

df_Summary_financials = pd.DataFrame(columns = ["tconst", "Domestic", "Inter
df_Summary_details = pd.DataFrame(columns = ["tconst", "Index_O" "Domestic D
                                              "Earliest Release Date", "MPAA"
#if True:
#df_ttTolookup

for index, row in df_ttTolookup.iterrows():
    #print(index)
    #print (row)

#for index, row in df_IMDB_Title_Akas.iterrows():
    #print(row["tconst"])
    #print(row["index_O"])
    #break
#     #current_ttconst = 'tt2527338'

    current_ttconst = row["tconst"]
#     ddToLookupIndex = row["Index_O"]

    detailheaders = []
    detaildata = []
    detailsummary = {}

    the_getString = 'https://www.boxofficemojo.com/title/'+current_ttconst
    r=rq.get(the_getString)
    p=bs(r.text,'html.parser')


    detailheaders.append("tconst")
#     detailheaders.append("Index_O)
    detaildata.append(current_ttconst)
    #detaildata.append(ddToLookupIndex)


    b=p.find('div', class_="a-section a-spacing-none mojo-summary-values moj
    if b:
        if b.find_all('span'):
            spans=b.find_all('span')
            #print(len(spans))
            iteration = 1
            for span in spans:
                # get rid of "a" tags
                if span.a:
                    next
                # get rid of sub span tags
                if span.span:
                    next
                else:
                    if (iteration % 2) == 0:
                        iteration +=1
                        # this is even - meaning a detail row
                        detaildata.append(span.text.strip())
                    else:
                       # this is odd - meaning a headerrow
                        #print(span.text)
                        iteration +=1
                        detailheaders.append(span.text.strip())
                        #data=[item for item in data]
                        #data=[re.sub('[^0-9]',"", str(item)) for item in da

    detailsummary = dict(zip(detailheaders, detaildata))
    #alldata.append(ttconst)
    #alldata.append(summarydata)
    df_Summary_details = df_Summary_details.append(detailsummary, ignore_ind

```

In [ ]:

```
df_Summary_details.to_excel("df_Summary_details.xlsx", header=True, index=Tr
```