Final Project Submission

Please fill out:

- Student name: Jane Martha
- Student pace: Part Time
- Scheduled project review date/time: 2/18/2024
- Instructor name: Samwel G
- Blog post URL:

```
# Your code here - remember to use markdown cells for comments as
well!

# Import standard packages
import pandas as pd
import numpy as np

# import
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

#loading data
df_title_ratings =
pd.read_csv("zippedData/imdb.title.ratings.csv.gz",)
df_title_basics = pd.read_csv("zippedData/imdb.title.basics.csv.gz",)
df_movie_gross = pd.read_csv("zippedData/bom.movie_gross.csv.gz",)
df_movies= pd.read_csv("zippedData/tmdb.movies.csv.gz",index_col=0)

#accessing the first 5 rows in df_title_basics
df_title_basics.head()

      tconst                    primary_title
original_title  \
0  tt0063540                        Sunghursh
Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek
Din
2  tt0069049       The Other Side of the Wind  The Other Side of the
Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada
Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela
Errante

   start_year  runtime_minutes                genres
0        2013            175.0    Action,Crime,Drama
1        2019            114.0       Biography,Drama
2        2018            122.0                 Drama
```

```
3        2018              NaN         Comedy,Drama
4        2017             80.0  Comedy,Drama,Fantasy
```

```
#Checking metadata of df_title_basics
df_title_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

```
#accessing the last 5 rows in df_title_basics
df_title_basics.tail()
```

```
           tconst                                primary_title  \
146139  tt9916538                          Kuambil Lagi Hatiku   
146140  tt9916622  Rodolpho Teóphilo - O Legado de um Pioneiro   
146141  tt9916706                              Dankyavar Danka   
146142  tt9916730                                       6 Gunn   
146143  tt9916754               Chico Albuquerque - Revelações   

                                     original_title  start_year  \
146139                          Kuambil Lagi Hatiku        2019   
146140  Rodolpho Teóphilo - O Legado de um Pioneiro        2015   
146141                              Dankyavar Danka        2013   
146142                                       6 Gunn        2017   
146143               Chico Albuquerque - Revelações        2013   

        runtime_minutes       genres  
146139            123.0        Drama  
146140              NaN  Documentary  
146141              NaN       Comedy  
146142            116.0          NaN  
146143              NaN  Documentary  
```

```
#Working with title_basics_df
df_title_basics.describe()
```

```
          start_year  runtime_minutes
count  146144.000000    114405.000000
mean     2014.621798        86.187247
std         2.733583       166.360590
```

```
min      2010.000000         1.000000
25%      2012.000000        70.000000
50%      2015.000000        87.000000
75%      2017.000000        99.000000
max      2115.000000     51420.000000
```

```
#Checking out the rows and columns of df_title_basics
df_title_basics.shape
```

```
(146144, 6)
```

```
# checking the data types
df_title_basics.dtypes
```

```
tconst              object
primary_title       object
original_title      object
start_year           int64
runtime_minutes    float64
genres              object
dtype: object
```

```
#Accessing columns of df_title_basics
df_title_basics.columns
```

```
Index(['tconst', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')
```

```
#Accessing the index of df_title_basics
df_title_basics.index
```

```
RangeIndex(start=0, stop=146144, step=1)
```

```
#aceesing the first 5 rows in df_title_ratings
df_title_ratings.head()
```

```
       tconst  averagerating  numvotes
0  tt10356526            8.3        31
1  tt10384606            8.9       559
2   tt1042974            6.4        20
3   tt1043726            4.2     50352
4   tt1060240            6.5        21
```

```
#Accessing concise summary of df_title_ratings

df_title_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   tconst         73856 non-null  object 
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

```
#Accessing the last 3 rows of df_title_ratings.tail
df_title_ratings.tail(3)
```

```
          tconst  averagerating  numvotes
73853  tt9851050            4.7        14
73854  tt9886934            7.0         5
73855  tt9894098            6.3       128
```

```
#accessing description of df_movie_gross
df_title_ratings.describe()
```

```
       averagerating      numvotes
count   73856.000000  7.385600e+04
mean        6.332729  3.523662e+03
std         1.474978  3.029402e+04
min         1.000000  5.000000e+00
25%         5.500000  1.400000e+01
50%         6.500000  4.900000e+01
75%         7.400000  2.820000e+02
max        10.000000  1.841066e+06
```

```
#Analyzing rows and columns of df_title_ratings
df_title_ratings.shape
```

```
(73856, 3)
```

```
#Checking the data type
df_title_ratings.dtypes
```

```
tconst            object
averagerating    float64
numvotes           int64
dtype: object
```

```
#Accessing columns of df_title_ratings
df_title_ratings.columns
```

```
Index(['tconst', 'averagerating', 'numvotes'], dtype='object')
```

```
#Accessing the index of df_title_ratings
df_title_ratings.index
```

```
RangeIndex(start=0, stop=73856, step=1)
```

```
#Analyzing metadata of df_movie_gross.info

df_movie_gross.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   title           3387 non-null   object
 1   studio          3382 non-null   object
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object
 4   year            3387 non-null   int64
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB

#checking the first 5 rows of df_movie_gross
df_movie_gross.head()

                                         title studio  domestic_gross
\
0                                  Toy Story 3     BV     415000000.0

1                   Alice in Wonderland (2010)     BV     334200000.0

2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0

3                                    Inception     WB     292600000.0

4                          Shrek Forever After   P/DW     238700000.0


  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010

#checking the last 5 rows of df_movie_gross
df_movie_gross.tail()

                            title      studio  domestic_gross
foreign_gross  \
3382                    The Quake       Magn.          6200.0
NaN
3383  Edward II (2018 re-release)          FM          4800.0
NaN
3384                     El Pacto        Sony          2500.0
NaN
```

```
3385                     The Swan  Synergetic         2400.0
NaN
3386            An Actor Prepares       Grav.         1700.0
NaN

      year
3382  2018
3383  2018
3384  2018
3385  2018
3386  2018

#print numbers pf rows and columns in bom_movie_gross_df
print(df_movie_gross.shape)

(3387, 5)

#Accessing the columns of df_movie_gross
print(df_movie_gross.columns)

Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'],
dtype='object')

print(df_movie_gross.dtypes)

title              object
studio             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object

#Accessing the index of df_movie_gross
print(df_movie_gross.index)

RangeIndex(start=0, stop=3387, step=1)

#Removing leading and trailing whitespace in df_title_basics columns
[col.strip() for col in df_title_basics.columns]

['tconst',
 'primary_title',
 'original_title',
 'start_year',
 'runtime_minutes',
 'genres']

#Checking out the missing values in df_title_basics
missing_values = df_title_basics.isna().sum()
missing_values
```

```
tconst                 0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
```

```
#Checking out the missing values in df_title_basics
missing_values = df_title_ratings.isna().sum()
missing_values
```

```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

```
#Removing leading and trailing whitespace in
[col.strip() for col in df_movie_gross.columns]
```

```
['title', 'studio', 'domestic_gross', 'foreign_gross', 'year']
```

```
#Checking out the missing values in df_bom_movie_gross
missing_values = df_movie_gross.isna().sum()
missing_values
```

```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         26517 non-null  int64
 1   genre_ids          26517 non-null  object
 2   id                 26517 non-null  int64
 3   original_language  26517 non-null  object
 4   original_title     26517 non-null  object
 5   popularity         26517 non-null  float64
 6   release_date       26517 non-null  object
 7   title              26517 non-null  object
 8   vote_average       26517 non-null  float64
 9   vote_count         26517 non-null  int64
```

```
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB

df_movies.head()

             genre_ids     id original_language  \
0      [12, 14, 10751]  12444                en
1  [14, 12, 16, 10751]  10191                en
2        [12, 28, 878]  10138                en
3      [16, 35, 10751]    862                en
4        [28, 878, 12]  27205                en

                                 original_title  popularity 
release_date  \
0  Harry Potter and the Deathly Hallows: Part 1      33.533   2010-11-
19
1                      How to Train Your Dragon      28.734   2010-03-
26
2                                    Iron Man 2      28.515   2010-05-
07
3                                     Toy Story      28.005   1995-11-
22
4                                     Inception      27.920   2010-07-
16

                                          title  vote_average 
vote_count
0  Harry Potter and the Deathly Hallows: Part 1           7.7 
10788
1                      How to Train Your Dragon           7.7 
7610
2                                    Iron Man 2           6.8 
12368
3                                     Toy Story           7.9 
10174
4                                     Inception           8.3 
22186

df_movies.tail()

       Unnamed: 0        genre_ids      id original_language  \
26512       26512         [27, 18]  488143                en
26513       26513         [18, 53]  485975                en
26514       26514     [14, 28, 12]  381231                en
26515       26515  [10751, 12, 28]  366854                en
26516       26516         [53, 27]  309885                en

              original_title  popularity release_date 
title  \
26512  Laboratory Conditions         0.6   2018-10-13  Laboratory
```

```
Conditions
26513        _EXHIBIT_84xxx_         0.6   2018-05-01
_EXHIBIT_84xxx_
26514           The Last One         0.6   2018-10-01           The
Last One
26515           Trailer Made         0.6   2018-06-22
Trailer Made
26516             The Church         0.6   2018-10-05             The
Church

       vote_average  vote_count
26512           0.0           1
26513           0.0           1
26514           0.0           1
26515           0.0           1
26516           0.0           1
```

```
df_movies.shape
```

```
(26517, 10)
```

```
df_movies.describe()
```

```
        Unnamed: 0             id    popularity  vote_average
vote_count
count  26517.00000   26517.000000  26517.000000  26517.000000
26517.000000
mean   13258.00000  295050.153260      3.130912      5.991281
194.224837
std     7654.94288  153661.615648      4.355229      1.852946
960.961095
min        0.00000      27.000000      0.600000      0.000000
1.000000
25%     6629.00000  157851.000000      0.600000      5.000000
2.000000
50%    13258.00000  309581.000000      1.374000      6.000000
5.000000
75%    19887.00000  419542.000000      3.694000      7.000000
28.000000
max    26516.00000  608444.000000     80.773000     10.000000
22186.000000
```

```
df_movies.dtypes
```

```
Unnamed: 0             int64
genre_ids             object
id                     int64
original_language     object
original_title        object
popularity           float64
release_date          object
```

```
title                      object
vote_average              float64
vote_count                  int64
dtype: object
```

```
df_movies.columns
```

```
Index(['Unnamed: 0', 'genre_ids', 'id', 'original_language',
'original_title',
       'popularity', 'release_date', 'title', 'vote_average',
'vote_count'],
      dtype='object')
```

```
df_movies.index
```

```
RangeIndex(start=0, stop=26517, step=1)
```

```
#Removing leading and trailing whitespace in df_title_ratings columns
[col.strip() for col in df_title_ratings.columns]
```

```
['tconst', 'averagerating', 'numvotes']
```

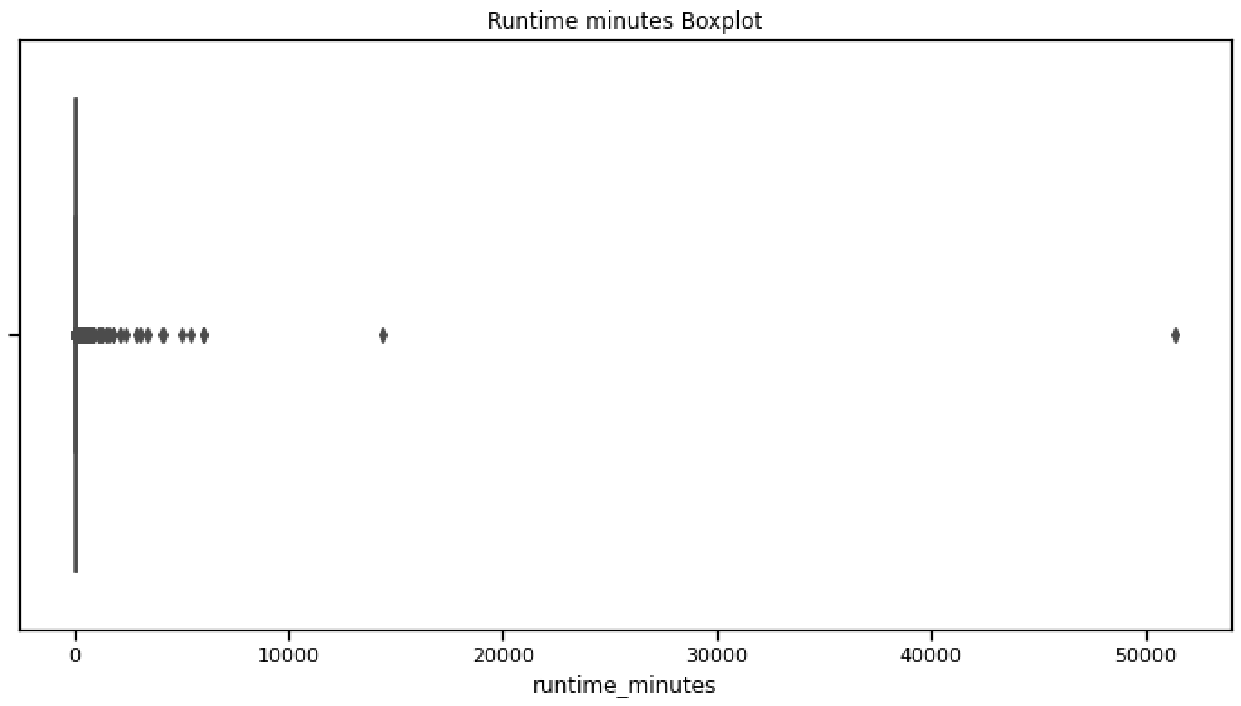
```
#Calculating the percentage of missing values per column with respect
to the entering df a function
def missing_values(data):
    """A simple function to identify data with missing values"""
    #identify the totsl missing values per column
    #sort in order
    miss = data.isnull().sum().sort_values(ascending = False)

    #calculate percentage of missing values
    percentage_miss = (data.isnull().sum() /
len(data)).sort_values(ascending = False)

    #store in a dataframe
    missing = pd.DataFrame({"Missing Values": miss, "Percentage":
percentage_miss}).reset_index()

    return missing
#applying function to the df_title_basics
missing_data = missing_values(df_title_basics)
missing_data
```

```
             index  Missing Values  Percentage
0  runtime_minutes           31739    0.217176
1           genres            5408    0.037005
2   original_title              21    0.000144
3       start_year               0    0.000000
4    primary_title               0    0.000000
5           tconst               0    0.000000
```

```
#genres and original_title missing values are few hence drop them
without causing any effect on data

#Filtering df that only contains Nan and empty strings
missing_genres = df_title_basics.genres.isna()
df_title_basics = df_title_basics[~missing_genres]

missing_title = df_title_basics.original_title.isna()
df_title_basics = df_title_basics[~missing_title]

#Confirming whether Nan values in original_title and genres columns
have all dropped
missing_values(df_title_basics)

             index  Missing Values  Percentage
0  runtime_minutes           28502    0.202524
1           genres               0    0.000000
2       start_year               0    0.000000
3   original_title               0    0.000000
4    primary_title               0    0.000000
5           tconst               0    0.000000

#Confirming whether Nan values in genres columns have all dropped

missing_values(df_title_basics)

             index  Missing Values  Percentage
0  runtime_minutes           28503    0.202528
1   original_title               2    0.000014
2           genres               0    0.000000
3       start_year               0    0.000000
4    primary_title               0    0.000000
5           tconst               0    0.000000

#Visualization of runtime minutes

runtimes = df_title_basics.runtime_minutes

#finding max and mim runtime
min_runtime = runtimes.min()
max_runtime = runtimes.max()
mean_runtime = runtimes.mean()
print(f"minimum runtime: {min_runtime}")
print(f"Maximum runtime: {max_runtime}")
print(f"Mean runtime: {mean_runtime}")

#Choosing boxplot column
col_data = df_title_basics.runtime_minutes

# Creating boxplot
plt.figure(figsize=(12,6))
```

```python
sns.set_context('notebook')
sns.boxplot(x= col_data, color= "yellow")
plt.title('Runtime minutes Boxplot');
```

```
minimum runtime: 1.0
Maximum runtime: 51420.0
Mean runtime: 86.26155641884668
```

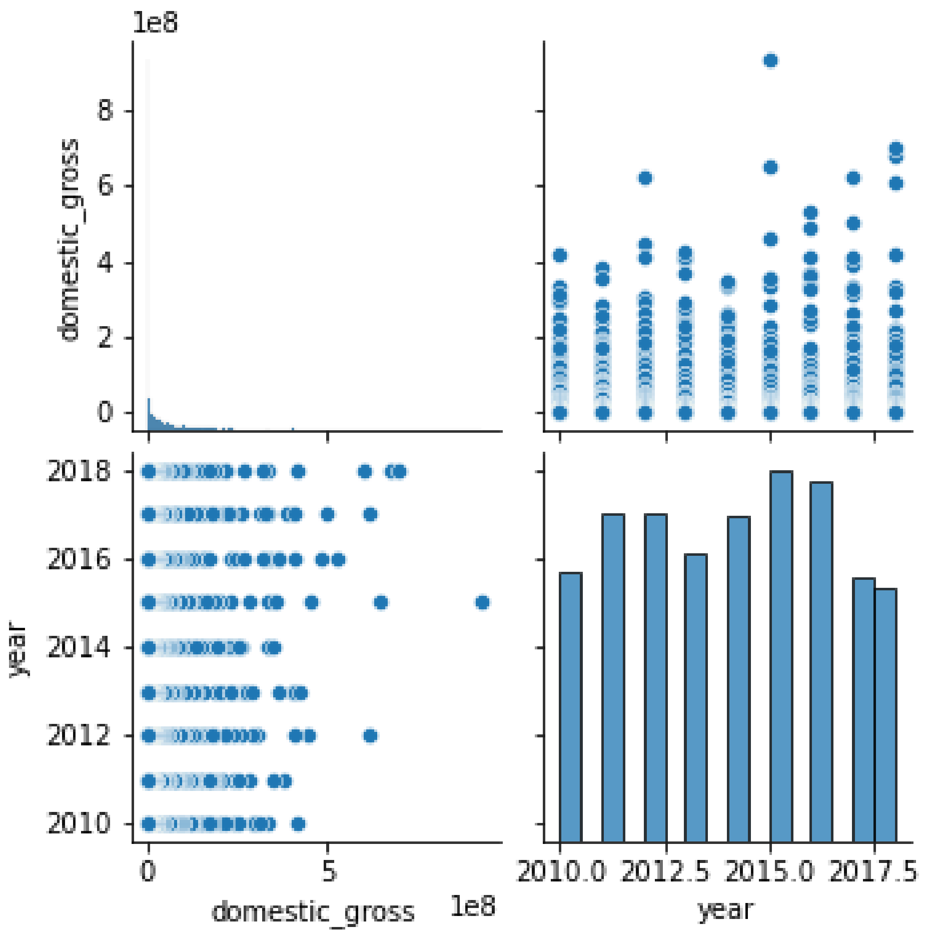


```python
#Checking for the highest runtime
df_title_basics.loc[df_title_basics.runtime_minutes == max_runtime]
```

```
           tconst primary_title original_title  start_year
runtime_minutes  \
132389  tt8273150     Logistics      Logistics        2012
51420.0

             genres
132389  Documentary
```

```python
sns.pairplot(df_movie_gross)
```

```
<seaborn.axisgrid.PairGrid at 0x2a005f73fd0>
```



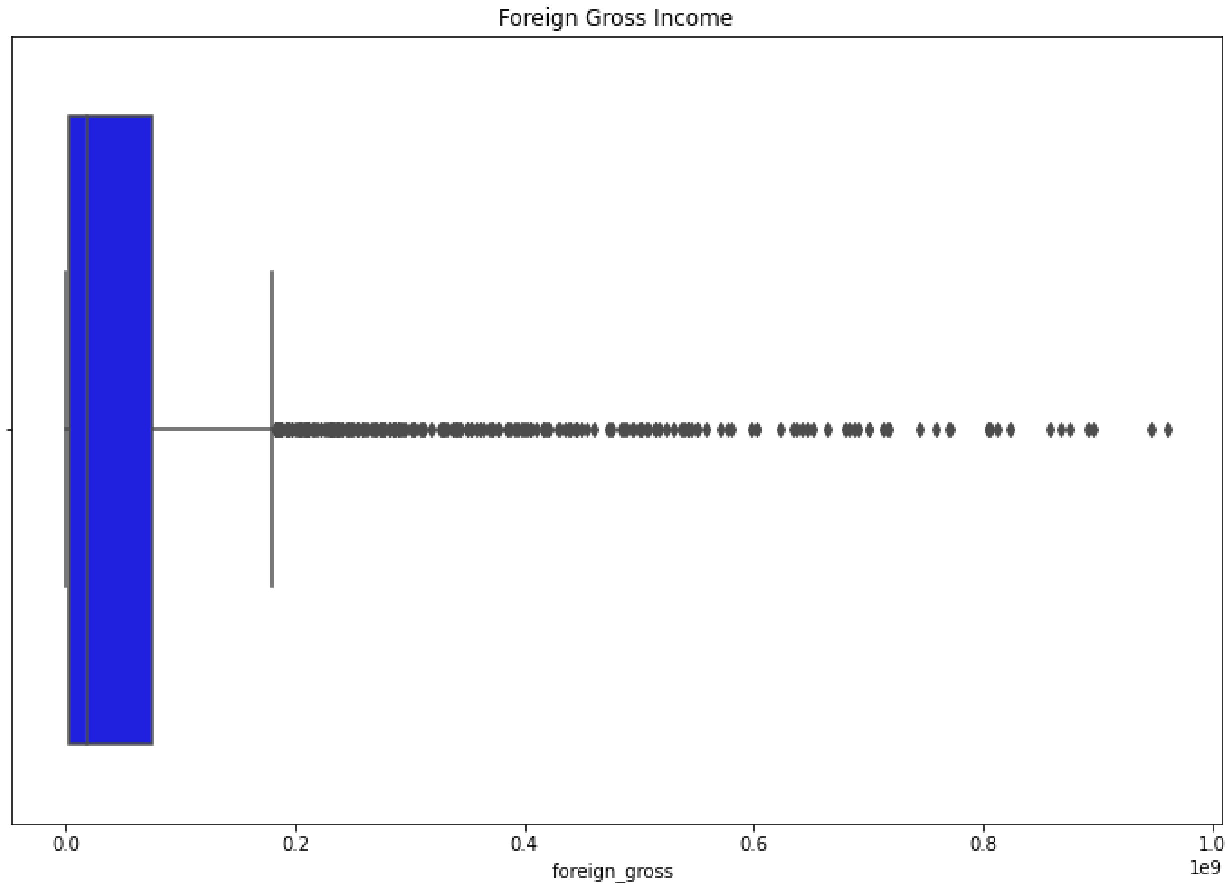
```
#replacing the missing values with  median runtime
df_title_basics.runtime_minutes.fillna(df_title_basics.runtime_minutes
.median(), inplace=True)

#checking for residual of missing values in DF
missing_values

title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64

#cheking duplicates
duplicate_rows = df_title_basics.duplicated().sum()
print(f"Num of Duplicated Rows: {duplicate_rows}")

Num of Duplicated Rows: 0
```

```
#Dealing with df_movie_gross
missing_data = missing_values(df_movie_gross)
missing_data

             index  Missing Values  Percentage
0    foreign_gross            1350    0.398583
1   domestic_gross              28    0.008267
2           studio               5    0.001476
3             year               0    0.000000
4            title               0    0.000000

#Filtering dataframe to remove missing values in df_movie_gross
missing_studio = df_movie_gross.studio.isna()
df_movie_gross = df_movie_gross[~missing_studio]

#checking the residual of missing values
missing_values(df_movie_gross)

             index  Missing Values  Percentage
0    foreign_gross            1349    0.398876
1   domestic_gross              26    0.007688
2             year               0    0.000000
3           studio               0    0.000000
4            title               0    0.000000

#removing unwanted characters like commas etc
df_movie_gross.foreign_gross.replace(',','', inplace=True, regex=True)

df_movie_gross.foreign_gross =
df_movie_gross.foreign_gross.astype('float64')

# checking dtype success
df_movie_gross.dtypes

title              object
studio             object
domestic_gross    float64
foreign_gross     float64
year                int64
dtype: object

#Performing Visualization of df_movie_gross
income_foreign = df_movie_gross.foreign_gross
# find minimum and maximum values in  runtime_minutes coumn
min_foreign = income_foreign.min()
max_foreign = income_foreign.max()
mean_foreign = income_foreign.mean()
print(f"minimum runtime: {min_foreign}")
print(f"Maximum runtime: {max_foreign}")
print(f"Mean runtime: {mean_foreign}")
# selecting the column for the boxplot
```

```
col_data = df_movie_gross.foreign_gross
# Creating boxplot using Seaborn Library
plt.figure(figsize= (12,8))
sns.boxplot(x= col_data, color= "blue")
plt.title('Foreign Gross Income')
plt.xlabel('foreign_gross ')
```

```
minimum runtime: 600.0
Maximum runtime: 960500000.0
Mean runtime: 74954901.2673389
```

```
Text(0.5, 0, 'foreign_gross ')
```

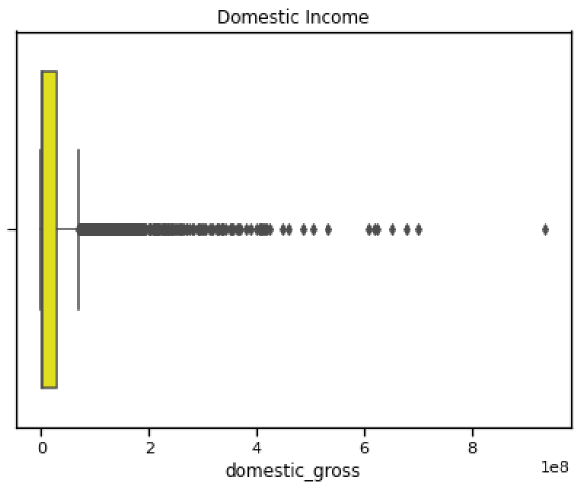


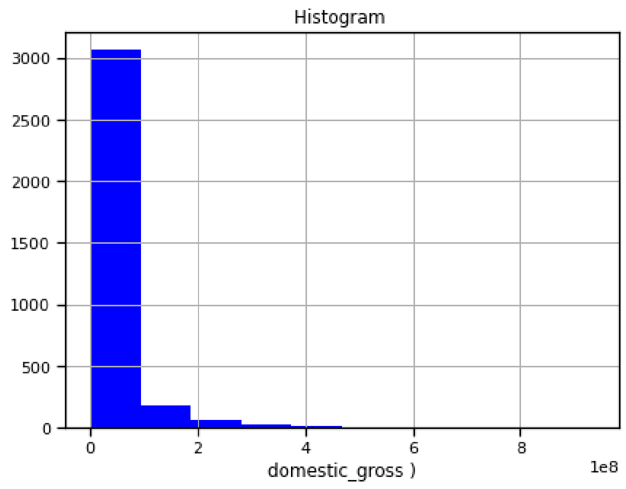
```
#Replacing the missing vlaue with median
df_movie_gross.foreign_gross.fillna(df_movie_gross.foreign_gross.isna(
).median, inplace=True)

#Checking whether the performance was successful
missing_values(df_movie_gross)
```

```
            index  Missing Values  Percentage
0  domestic_gross              28    0.008267
1          studio               5    0.001476
```

```
2             year                0    0.000000
3    foreign_gross                0    0.000000
4            title                0    0.000000

# Visualizing distribution of domestic_gross in our DataFrame
income_domestic = df_movie_gross.domestic_gross

# getting the minimum and maximum values in our runtime_minutes col
min_domestic = income_domestic.min()
max_domestic = income_domestic.max()
mean_domestic = income_domestic.mean()
print(f"minimum runtime: {min_domestic}")
print(f"Maximum runtime: {max_domestic}")
print(f"Mean runtime: {mean_domestic}")

# selecting the column for the boxplot
col_data = df_movie_gross.domestic_gross

# Creating boxplot & histogram using Seaborn Library
fig, ax = plt.subplots(ncols=2, nrows=1, figsize= (15,5))
sns.boxplot(x=col_data, ax=ax[0], color='yellow')
ax[0].set_title('Domestic Income')
ax[0].set_xlabel('domestic_gross')
df_movie_gross.domestic_gross.hist(ax=ax[1], color='blue')
ax[1].set_title('Histogram ')
ax[1].set_xlabel('domestic_gross )')
plt.tight_layout;
```

```
minimum runtime: 100.0
Maximum runtime: 936700000.0
Mean runtime: 28745845.06698422
```





```
#Repacing the missing values with Median
df_movie_gross.domestic_gross.fillna(df_movie_gross.domestic_gross.isn
a().median(), inplace=True)

#Verifying that the operation was successful
#function & checking for any instances of duplicates
print(f"Num of duplicates: {df_movie_gross.duplicated().sum()}")
print(missing_values(df_movie_gross))

Num of duplicates: 0
            index  Missing Values  Percentage
0          studio               5    0.001476
1            year               0    0.000000
2   foreign_gross               0    0.000000
3  domestic_gross               0    0.000000
4           title               0    0.000000

#Feature Engineering
#Merging different dataset in a single dataset
combined_data = df_movie_gross.merge(df_title_basics, left_on='title',
right_on='original_title', how='left')
combined_data = combined_data.merge(df_title_ratings,
left_on='tconst', right_on='tconst', how='left')
combined_data.head()

                                         title studio  domestic_gross
\
0                                  Toy Story 3     BV     415000000.0

1                   Alice in Wonderland (2010)     BV     334200000.0

2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0

3                                    Inception     WB     292600000.0

4                          Shrek Forever After   P/DW     238700000.0


  foreign_gross  year     tconst        primary_title
original_title  \
0     652000000  2010  tt0435761          Toy Story 3          Toy
Story 3
1     691300000  2010        NaN                  NaN
NaN
2     664300000  2010        NaN                  NaN
NaN
3     535700000  2010  tt1375666            Inception
Inception
4     513900000  2010  tt0892791  Shrek Forever After  Shrek Forever
After

   start_year  runtime_minutes                      genres
averagerating  \
0      2010.0           103.0  Adventure,Animation,Comedy
8.3
1         NaN             NaN                         NaN
NaN
2         NaN             NaN                         NaN
NaN
3      2010.0           148.0     Action,Adventure,Sci-Fi
8.8
4      2010.0            93.0  Adventure,Animation,Comedy
6.3

    numvotes
0   682218.0
1        NaN
2        NaN
3  1841066.0
4   167532.0
```

```
#Checking the missing value in the comnined data
missing_values(combined_data)
```

```
              index  Missing Values  Percentage
0     foreign_gross            1557    0.392686
1          numvotes            1522    0.383859
2     averagerating            1522    0.383859
3            genres            1225    0.308953
4   runtime_minutes            1225    0.308953
5        start_year            1225    0.308953
6    original_title            1225    0.308953
7     primary_title            1225    0.308953
8            tconst            1225    0.308953
9    domestic_gross              35    0.008827
10           studio               5    0.001261
11             year               0    0.000000
12            title               0    0.000000
```

```
combined_data.shape
```

```
(3965, 13)
```

```
#Accessing Columns of interest and reassingning to df__movies
df_movies = df_movies.loc[:, ['original_title', 'vote_average',
'vote_count', 'release_date']]

#Checking whether the needed data is successfully extracted
df_movies.head()
```

```
                                 original_title  vote_average
vote_count  \
0  Harry Potter and the Deathly Hallows: Part 1           7.7
10788
```

```
1                          How to Train Your Dragon            7.7
7610
2                                        Iron Man 2            6.8
12368
3                                         Toy Story            7.9
10174
4                                         Inception            8.3
22186

   release_year
0          2010
1          2010
2          2010
3          1995
4          2010

# Convert 'release_date' column to date_time if it's not already in
datetime format
df_movies['release_date'] = pd.to_datetime(df_movies['release_date'])

# Extract the year from 'release_date' and create a new column
'release_year'
df_movies['release_year'] = df_movies['release_date'].dt.year

# Drop the 'release_date' column if you want to remove it
df_movies.drop('release_date', axis=1, inplace=True)

# # Display the first few rows of the modified DataFrame
print(df_movies.head())

                                original_title  vote_average
vote_count  \
0  Harry Potter and the Deathly Hallows: Part 1            7.7
10788
1                          How to Train Your Dragon            7.7
7610
2                                        Iron Man 2            6.8
12368
3                                         Toy Story            7.9
10174
4                                         Inception            8.3
22186

   release_year
0          2010
1          2010
2          2010
3          1995
4          2010
```

```
 #Executing an inner join to preserve only the movies appearing in
both Tables
combined_data = combined_data.merge(df_movies, left_on= 'title',
right_on = 'original_title', how="inner")
combined_data.head()

                      title_x studio  domestic_gross foreign_gross
year  \
0                 Toy Story 3     BV     415000000.0     652000000
2010
1                   Inception     WB     292600000.0     535700000
2010
2         Shrek Forever After   P/DW     238700000.0     513900000
2010
3  The Twilight Saga: Eclipse   Sum.     300500000.0     398000000
2010
4                  Iron Man 2   Par.     312400000.0     311500000
2010

      tconst               primary_title
original_title_x  \
0  tt0435761                 Toy Story 3                 Toy Story 3

1  tt1375666                   Inception                   Inception

2  tt0892791         Shrek Forever After         Shrek Forever After

3  tt1325004  The Twilight Saga: Eclipse  The Twilight Saga: Eclipse

4  tt1228705                  Iron Man 2                  Iron Man 2


   start_year  runtime_minutes  ... Unnamed: 0
genre_ids  \
0      2010.0            103.0  ...          7          [16, 10751,
35]
1      2010.0            148.0  ...          4            [28, 878,
12]
2      2010.0             93.0  ...         38  [35, 12, 14, 16,
10751]
3      2010.0            124.0  ...         15      [12, 14, 18,
10749]
4      2010.0            124.0  ...          2            [12, 28,
878]

      id  original_language            original_title_y  popularity  \
0  10193                 en                 Toy Story 3      24.445
1  27205                 en                   Inception      27.920
2  10192                 en         Shrek Forever After      15.041
3  24021                 en  The Twilight Saga: Eclipse      20.340
```

```
4  10138                 en                 Iron Man 2      28.515

  release_date                     title_y  vote_average vote_count
0   2010-06-17                 Toy Story 3           7.7       8340
1   2010-07-16                   Inception           8.3      22186
2   2010-05-16         Shrek Forever After           6.1       3843
3   2010-06-23  The Twilight Saga: Eclipse           6.0       4909
4   2010-05-07                  Iron Man 2           6.8      12368

[5 rows x 23 columns]

 missing_values(combined_data)

              index  Missing Values  Percentage
0          numvotes             443    0.141669
1     averagerating             443    0.141669
2   runtime_minutes             241    0.077071
3        start_year              84    0.026863
4            genres              84    0.026863
5  original_title_x              84    0.026863
6     primary_title              84    0.026863
7            tconst              84    0.026863
8              year               0    0.000000
9            studio               0    0.000000
10   domestic_gross               0    0.000000
11    foreign_gross               0    0.000000
12     release_year               0    0.000000
13       vote_count               0    0.000000
14 original_title_y               0    0.000000
15     vote_average               0    0.000000
16            title               0    0.000000

# dropping the missing velues in start_year
combined_data.drop('start_year', inplace=True, axis=1)

#Filling missing values in averagerating column with vote_ average
column values
combined_data.averagerating.fillna(combined_data.vote_average,
inplace=True)

#dropping vote_average columns
combined_data.drop('vote_average', axis=1, inplace=True)

#dropping vote_count columns and refilling the null values in
num_votes with median

combined_data.drop('vote_count', inplace=True, axis=1)

#checking whether the vote_count colum has dropped successfully
combined_data.head()
```

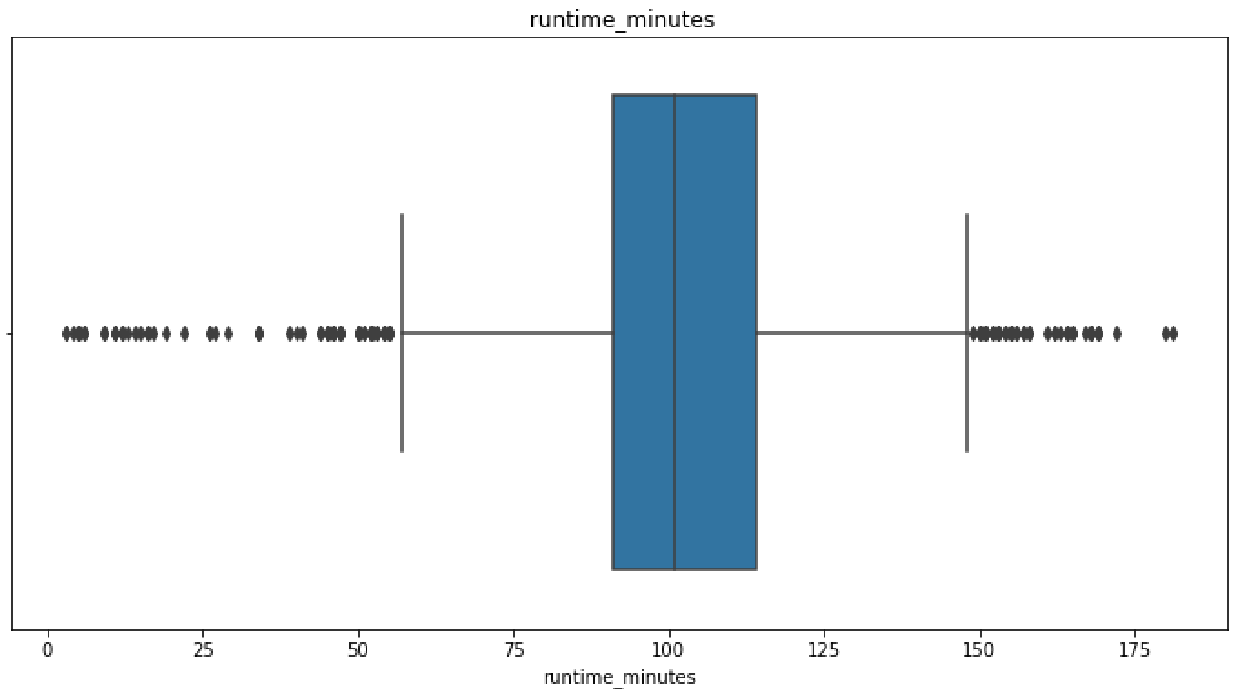
```
                      title_x studio  domestic_gross foreign_gross 
year  \
0                 Toy Story 3     BV     415000000.0     652000000
2010
1                   Inception     WB     292600000.0     535700000
2010
2         Shrek Forever After   P/DW     238700000.0     513900000
2010
3  The Twilight Saga: Eclipse   Sum.     300500000.0     398000000
2010
4                  Iron Man 2   Par.     312400000.0     311500000
2010

      tconst               primary_title
original_title_x  \
0  tt0435761                 Toy Story 3                 Toy Story 3

1  tt1375666                   Inception                   Inception

2  tt0892791         Shrek Forever After         Shrek Forever After

3  tt1325004  The Twilight Saga: Eclipse  The Twilight Saga: Eclipse

4  tt1228705                  Iron Man 2                  Iron Man 2


   runtime_minutes                      genres  averagerating
numvotes  \
0            103.0  Adventure,Animation,Comedy            8.3
682218.0
1            148.0     Action,Adventure,Sci-Fi            8.8
1841066.0
2             93.0  Adventure,Animation,Comedy            6.3
167532.0
3            124.0     Adventure,Drama,Fantasy            5.0
211733.0
4            124.0     Action,Adventure,Sci-Fi            7.0
657690.0

   Unnamed: 0                 genre_ids     id original_language  \
0           7          [16, 10751, 35]  10193                en
1           4            [28, 878, 12]  27205                en
2          38  [35, 12, 14, 16, 10751]  10192                en
3          15      [12, 14, 18, 10749]  24021                en
4           2            [12, 28, 878]  10138                en
```

```
             original_title_y  popularity release_date  \
0                 Toy Story 3      24.445   2010-06-17
1                   Inception      27.920   2010-07-16
2         Shrek Forever After      15.041   2010-05-16
3  The Twilight Saga: Eclipse      20.340   2010-06-23
4                  Iron Man 2      28.515   2010-05-07

                      title_y
0                 Toy Story 3
1                   Inception
2         Shrek Forever After
3  The Twilight Saga: Eclipse
4                  Iron Man 2
```

```
 missing_values(combined_data)
```

```
              index  Missing Values  Percentage
0          numvotes            1522    0.383859
1     averagerating            1522    0.383859
2   runtime_minutes            1364    0.344010
3            genres            1225    0.308953
4    original_title            1225    0.308953
5     primary_title            1225    0.308953
6            tconst            1225    0.308953
7            studio               5    0.001261
8              year               0    0.000000
9     foreign_gross               0    0.000000
10   domestic_gross               0    0.000000
11            title               0    0.000000
```

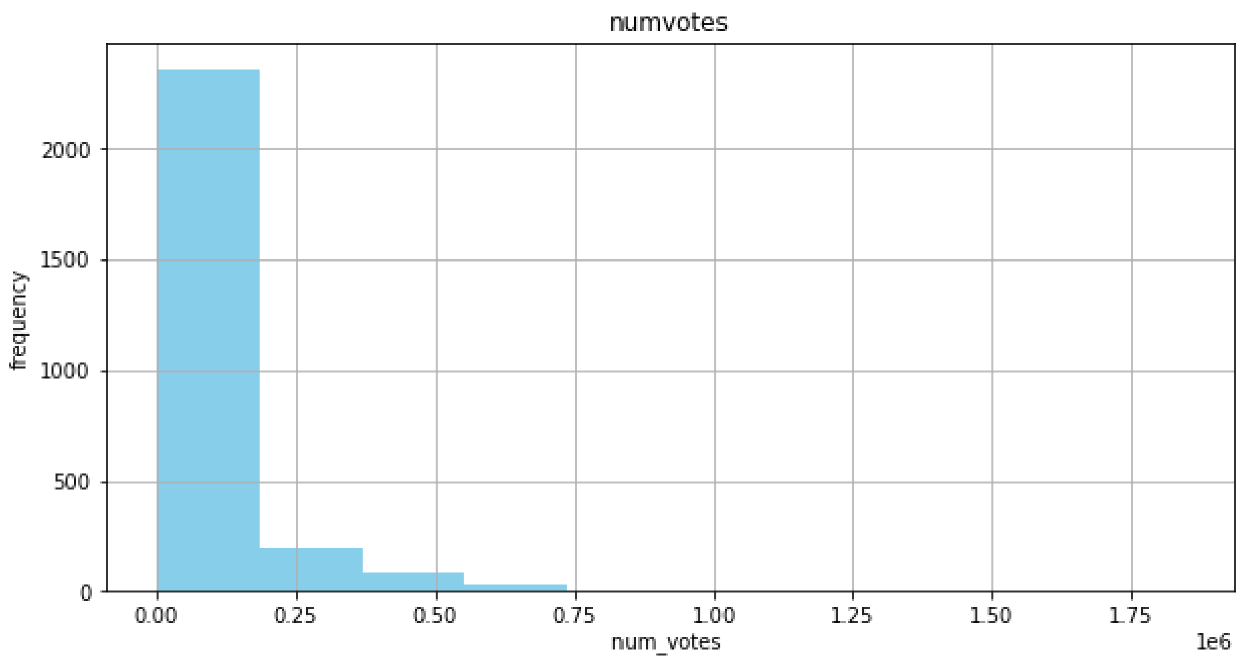
```
#Visualizing the distribution in combined dataframe for
runtime_mintues
col_data = combined_data.runtime_minutes
plt.figure(figsize=(12,6))
sns.boxplot(x=col_data)
plt.title(' runtime_minutes');
```



```
#The data contains outliers
#Using imputation to replace the missing values

combined_data.runtime_minutes.fillna(combined_data.runtime_minutes.mea
n(),inplace=True)

#confirming if the operation was successful
missing_values(combined_data)
```

```
               index  Missing Values  Percentage
0           numvotes             444    0.141944
1             genres              85    0.027174
2   original_title_x              85    0.027174
3      primary_title              85    0.027174
4             tconst              85    0.027174
5             studio               1    0.000320
6       release_year               0    0.000000
7   original_title_y               0    0.000000
8      averagerating               0    0.000000
9    runtime_minutes               0    0.000000
10              year               0    0.000000
11     foreign_gross               0    0.000000
12    domestic_gross               0    0.000000
13             title               0    0.000000
```

```
#visualizing the distribution of numvotes
print(combined_data.numvotes.agg(['mean', 'std', 'min', 'max']))
combined_data.numvotes.hist(color='skyblue', figsize=(10, 5))
plt.title("numvotes")
plt.xlabel('num_votes ')
plt.ylabel('frequency')

mean    7.731767e+04
std     1.362478e+05
min     5.000000e+00
max     1.841066e+06
Name: numvotes, dtype: float64

Text(0, 0.5, 'frequency')
```



```
#Filling the skewed data using imputation median
combined_data.numvotes.fillna(combined_data.numvotes.median(),
inplace=True)

missing_values(combined_data)

                 index  Missing Values  Percentage
0               genres              84    0.026863
1     original_title_x              84    0.026863
2        primary_title              84    0.026863
3               tconst              84    0.026863
4         release_year               0    0.000000
5     original_title_y               0    0.000000
6             numvotes               0    0.000000
7        averagerating               0    0.000000
8      runtime_minutes               0    0.000000
9                 year               0    0.000000
10       foreign_gross               0    0.000000
```

```
11    domestic_gross                0    0.000000
12            studio                0    0.000000
13             title                0    0.000000
```

```
#Checking the most dominant genre using the process

#Calling the value_counts method to find the mode genre
top_genre = combined_data.genres.value_counts().head(1)
print(f'The most common genre in our data is: {top_genre}')
```

```
The most common genre in our data is: Drama    323
Name: genres, dtype: int64
```

```
#Replacing the missing values with mode imputation
combined_data.genres.fillna(combined_data.genres.mode().iloc[0],
inplace=True)

 #confirming that we have no missing values in our dataset
missing_values(combined_data)
```

```
               index  Missing Values  Percentage
0   original_title_x              85    0.027174
1      primary_title              85    0.027174
2             tconst              85    0.027174
3             studio               1    0.000320
4       release_year               0    0.000000
5   original_title_y               0    0.000000
6            numvotes              0    0.000000
7      averagerating               0    0.000000
8             genres               0    0.000000
9     runtime_minutes              0    0.000000
10              year               0    0.000000
11     foreign_gross               0    0.000000
12    domestic_gross               0    0.000000
13             title               0    0.000000
```

```
#Confirming the dtypes of combined dataset
combined_data.dtypes
```

```
title                 object
studio                object
domestic_gross       float64
foreign_gross         object
year                   int64
tconst                object
primary_title         object
original_title_x      object
runtime_minutes      float64
genres                object
averagerating        float64
numvotes             float64
```

```
original_title_y     object
release_year          int64
dtype: object
```

```
#creating a new column that we will use to determine the financia
success of the company
combined_data['total_gross'] =
combined_data.domestic_gross.astype(str) +
combined_data.foreign_gross.astype(str)

#determinng whether new column was created successfully
combined_data.head()
```

```
                                          title studio  domestic_gross
\
0                                   Toy Story 3     BV       415000000

0                                   Toy Story 3     BV       415000000

0                                   Toy Story 3     BV       415000000

1                    Alice in Wonderland (2010)     BV       334200000

2  Harry Potter and the Deathly Hallows Part 1     WB       296000000


  foreign_gross  year     tconst primary_title original_title  \
0     652000000  2010  tt0435761   Toy Story 3    Toy Story 3
0     652000000  2010  tt0435761   Toy Story 3    Toy Story 3
0     652000000  2010  tt0435761   Toy Story 3    Toy Story 3
1     691300000  2010        NaN           NaN            NaN
2     664300000  2010        NaN           NaN            NaN

   runtime_minutes     genres  averagerating  numvotes
total_gross
0            103.0  Adventure            8.3  682218.0
415000000652000000
0            103.0  Animation            8.3  682218.0
415000000652000000
0            103.0     Comedy            8.3  682218.0
415000000652000000
1              NaN        NaN            NaN       NaN
334200000691300000
2              NaN        NaN            NaN       NaN
296000000664300000
```

```
combined_data['total_gross']
```

```
0                                      415000000652000000
0                                      415000000652000000
0                                      415000000652000000
```

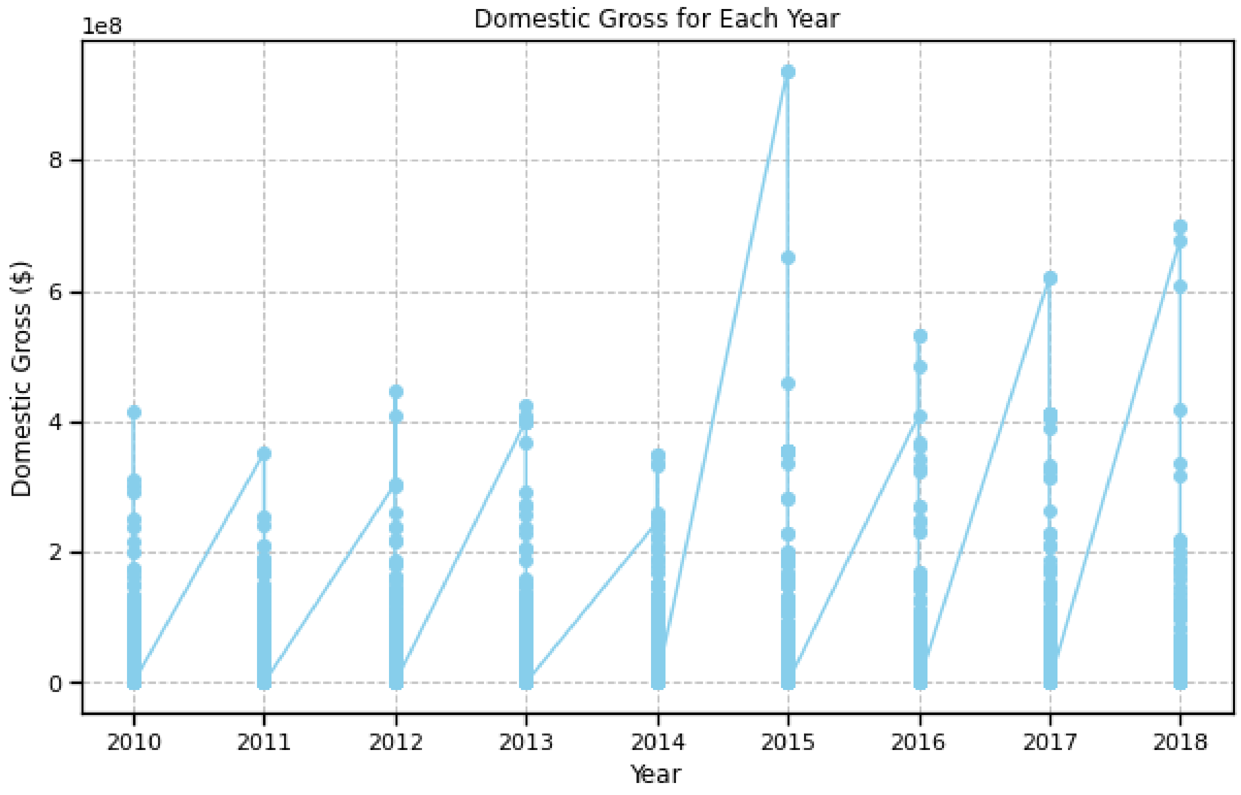
```
1                                     334200000691300000
2                                     296000000664300000
                              ...
3960    6200<bound method Series.median of 0       Fal...
3961    4800<bound method Series.median of 0       Fal...
3962    2500<bound method Series.median of 0       Fal...
3963    2400<bound method Series.median of 0       Fal...
3964    1700<bound method Series.median of 0       Fal...
Name: total_gross, Length: 7452, dtype: object
```

```
#Splitting movie genres and exploded to allow for genre-specific
analyses
combined_data.genres = combined_data.genres.str.split(',')
combined_data.head()
```

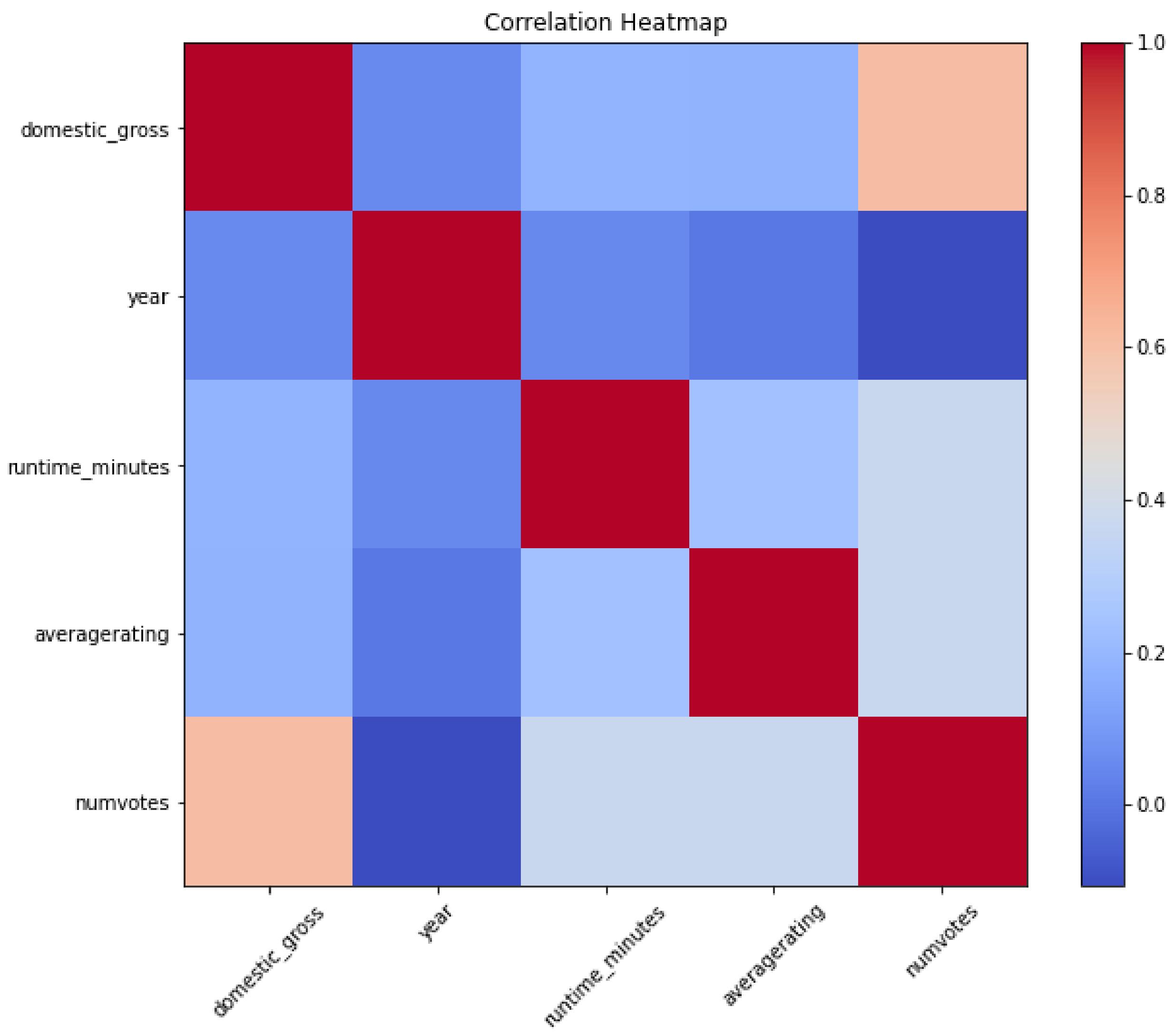
```
                                           title studio  domestic_gross
\
0                                    Toy Story 3     BV     415000000.0

1                     Alice in Wonderland (2010)     BV     334200000.0

2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0

3                                      Inception     WB     292600000.0

4                              Shrek Forever After   P/DW     238700000.0


  foreign_gross  year     tconst        primary_title
original_title  \
0     652000000  2010  tt0435761          Toy Story 3          Toy
Story 3
1     691300000  2010        NaN                  NaN
NaN
2     664300000  2010        NaN                  NaN
NaN
3     535700000  2010  tt1375666            Inception
Inception
4     513900000  2010  tt0892791  Shrek Forever After  Shrek Forever
After

   runtime_minutes                          genres  averagerating
numvotes
0            103.0  [Adventure, Animation, Comedy]            8.3
682218.0
1              NaN                             NaN            NaN
NaN
2              NaN                             NaN            NaN
NaN
3            148.0     [Action, Adventure, Sci-Fi]            8.8
```

```
1841066.0
4              93.0  [Adventure, Animation, Comedy]            6.3
167532.0

 #Exploding the genres
combined_data = combined_data.explode('genres')

 combined_data.head()

         title studio  domestic_gross foreign_gross  year
tconst  \
0  Toy Story 3     BV     415000000.0      6.52e+08  2010  tt0435761

0  Toy Story 3     BV     415000000.0      6.52e+08  2010  tt0435761

0  Toy Story 3     BV     415000000.0      6.52e+08  2010  tt0435761

1    Inception     WB     292600000.0     5.357e+08  2010  tt1375666

1    Inception     WB     292600000.0     5.357e+08  2010  tt1375666


  primary_title original_title_x  runtime_minutes     genres
averagerating  \
0   Toy Story 3      Toy Story 3            103.0  Adventure
8.3
0   Toy Story 3      Toy Story 3            103.0  Animation
8.3
0   Toy Story 3      Toy Story 3            103.0     Comedy
8.3
1     Inception        Inception            148.0     Action
8.8
1     Inception        Inception            148.0  Adventure
8.8

    numvotes original_title_y release_year
0   682218.0      Toy Story 3         2010
0   682218.0      Toy Story 3         2010
0   682218.0      Toy Story 3         2010
1  1841066.0        Inception         2010
1  1841066.0        Inception         2010

# Plotting the domestic gross for each year
plt.figure(figsize=(10, 6))
plt.plot(combined_data['year'], combined_data['domestic_gross'],
marker='o', color='skyblue', linestyle='-')

# Adding labels and title
plt.xlabel('Year')
plt.ylabel('Domestic Gross ($)')
plt.title('Domestic Gross for Each Year')
```

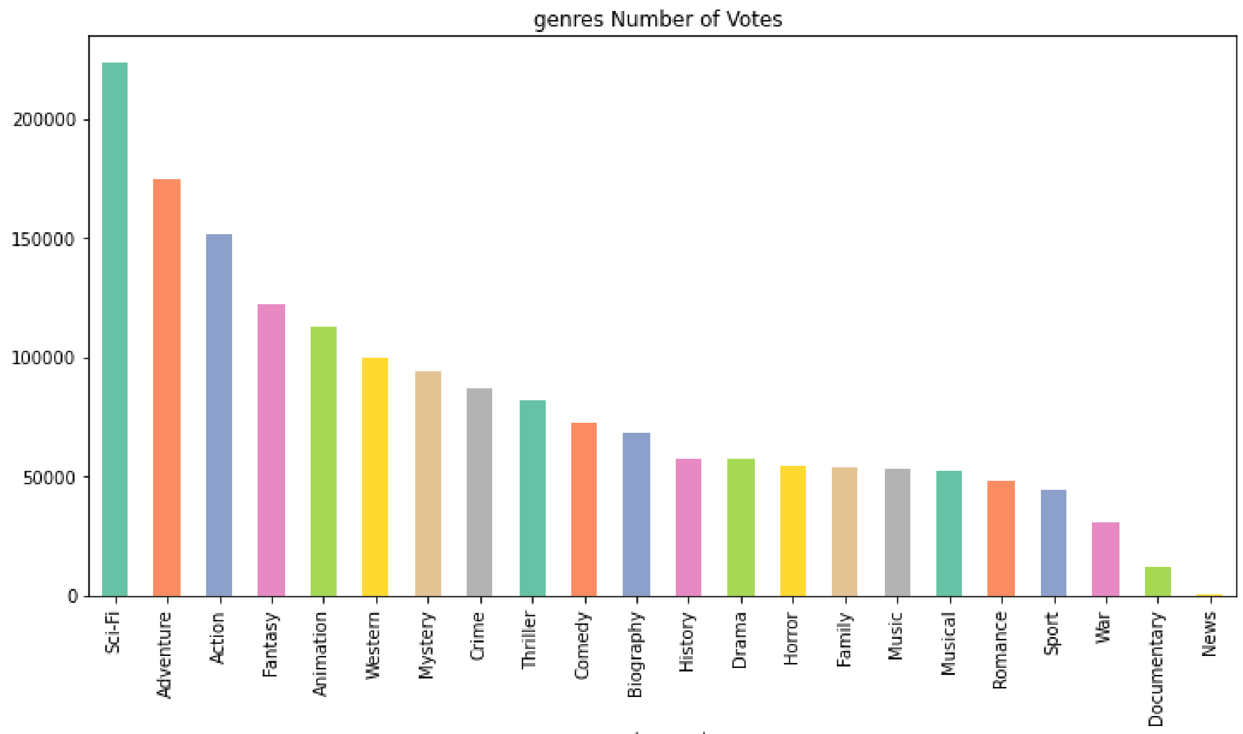
```
# Adding grid
plt.grid(True, linestyle='--', alpha=0.7)

# Show plot
plt.show()
```



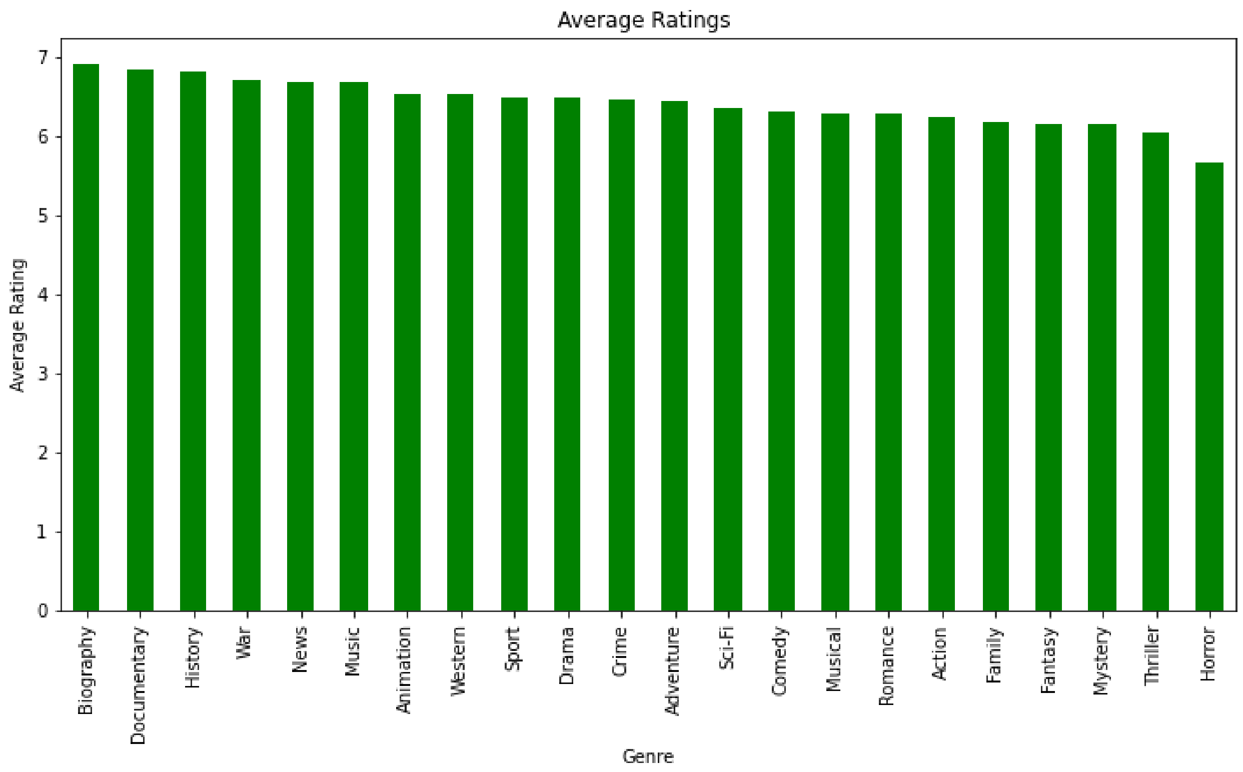
```
# Creating a heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 8))
correlation_matrix = combined_data.corr()
plt.imshow(correlation_matrix, cmap= 'coolwarm',
interpolation='nearest')
plt.colorbar()
plt.xticks(range(len(correlation_matrix)), correlation_matrix.columns,
rotation = 45)
plt.yticks(range(len(correlation_matrix)), correlation_matrix.columns)
plt.title("Correlation Heatmap")
plt.show()
```



```
# Visualizing the relationship between genres and average ratings
plt.figure(figsize=(12, 6))
genre_avg_votes = combined_data.groupby('genres')['numvotes'].mean()
sorted_data = genre_avg_votes.sort_values(ascending=False)
colors = sns.color_palette("Set2", n_colors=len(sorted_data))
sorted_data.plot(kind='bar', color=colors)
plt.title("genres Number of Votes ")
plt.xlabel("vote_count")

Text(0.5, 0, 'vote_count')
```



```
# Visualizing the relationship between genre and average ratings
plt.figure(figsize=(12, 6))
genre_avg_ratings = combined_data.groupby('genres')
['averagerating'].mean()
sorted_data = genre_avg_ratings.sort_values(ascending=False)
sorted_data.plot(kind='bar', color='green')
plt.title("Average Ratings ")
plt.xticks(rotation=90)
plt.xlabel("Genre")
plt.ylabel("Average Rating");
```


Average Ratings
Average Rating
Genre
Biography
Documentary
History
War
News
Music
Animation
Western
Sport
Drama
Crime
Adventure
Sci-Fi
Comedy
Musical
Romance
Action
Family
Fantasy
Mystery
Thriller
Horror
0
1
2
3
4
5
6
7