student

February 21, 2024

0.1 Final Project Submission

Please fill out:

Student name: **Waweru Brian**

Student pace: Self paced / part time

Scheduled project review date/time:

Instructor name: William Okomba, Noah Kandie and Samuel G. Mwangi

Blog post URL: N/A

1 Importing The Necessary Libraries

[1]:

```
# Basic libraries
import pandas as pd
import numpy as np

# Datasets
import csv
import json
import string

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

2 Basic EDA

2.1 (i) Checking how many files we have on the directory by running `! ls *.csv`

[2]:

```
! ls *zippedData/ *.csv
```

```
zippedData/:
bom.movie_gross.csv
bom.movie_gross.csv.gz
imdb.name.basics.csv.gz
imdb.title.akas.csv.gz
imdb.title.basics.csv.gz
imdb.title.crew.csv.gz
imdb.title.principals.csv.gz
imdb.title.ratings.csv.gz
name.basics.csv
rt.movie_info.tsv
rt.movie_info.tsv.gz
rt.reviews.tsv
rt.reviews.tsv.gz
title.akas.csv
title.basics.csv
title.crew.csv
title.principals.csv
title.ratings.csv
tmdb.movies.csv
tmdb.movies.csv.gz
tn.movie_budgets.csv
tn.movie_budgets.csv.gz
```

```
ls: cannot access '*.csv': No such file or directory
```

2.2 (i) importing the CSVs into the notebook

[3]:

```
df_bom = pd.read_csv('zippedData/bom.movie_gross.csv')
df_basics = pd.read_csv('zippedData/name.basics.csv')
df_akas = pd.read_csv('zippedData/title.akas.csv')
df_title_basics = pd.read_csv('zippedData/title.basics.csv')
df_crew = pd.read_csv('zippedData/title.crew.csv')
df_principals = pd.read_csv('zippedData/title.principals.csv')
df_ratings = pd.read_csv('zippedData/title.ratings.csv')
df_movies = pd.read_csv('zippedData/tmdb.movies.csv', index_col=0)
df_movie_budgets = pd.read_csv('zippedData/tn.movie_budgets.csv')

df1 = df_bom.copy()
df2 = df_basics.copy()
df3 = df_akas.copy()
df4 = df_title_basics.copy()
df5 = df_crew.copy()
df6 = df_principals.copy()
df7 = df_ratings.copy()
df8 = df_movies.copy()
df9 = df_movie_budgets.copy()
```

2.3

2.4 Basic Interpretations

.

2.4.1 - There are 9 different dataset tables present in the dataset

2.4.2 - Seemingly with a myriad of entries that include Titles of film, Names of Principals and Directors, as well as their financials.

.

3 Performing Simple Checks of what kind of columns and data present in each database

4 Table 1

[4]:
```
print(df1.shape)
print()
print(df1.info())
print()
```

```
(3387, 5)

<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
None
```

[5]:
```
df1.head(2)
```

[5]:
```
                        title studio  domestic_gross foreign_gross  year
0                 Toy Story 3     BV     415000000.0     652000000  2010
1  Alice in Wonderland (2010)     BV     334200000.0     691300000  2010
```

5 Basic Interpretation

.

5.0.1 - This table has over 3000 entries with only 5 columns.

5.0.2 - The columns give information about the `title` of the movie, the `studio` the title's are produced in, their 'domestic_gross' and `foreign_gross` revenues as well as the year the information was captured.

6 Suggestion(s)

6.0.1 - *Clearly interprete* `studio` *names.*

.

7 Table 2

[6]:
```
print(df2.shape)
print()
print(df2.info())
print()
```

```
(606648, 6)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 606648 entries, 0 to 606647
Data columns (total 6 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   nconst              606648 non-null  object
 1   primary_name        606648 non-null  object
 2   birth_year          82736 non-null   float64
 3   death_year          6783 non-null    float64
 4   primary_profession  555308 non-null  object
 5   known_for_titles    576444 non-null  object
dtypes: float64(2), object(4)
memory usage: 27.8+ MB
None
```

[7]:
```
df2.head(2)
```

[7]:
```
      nconst       primary_name  birth_year  death_year  \
0  nm0061671  Mary Ellen Bauder         NaN         NaN
1  nm0061865       Joseph Bauer         NaN         NaN

                        primary_profession  \
0  miscellaneous,production_manager,producer
1 composer,music_department,sound_department

                       known_for_titles
```

```
0  tt0837562,tt2398241,tt0844471,tt0118553
1  tt0896534,tt6791238,tt0287072,tt1682940
```

8 Basic Interpletation

.

8.0.1 - This table has over 600000 entries with only 6 columns.

8.0.2 - The columns point out the names of professional involved, their individual professional contribution and what film's (title's) they are known for.

9 Suggestion(s)

9.0.1 - *Come up with a python `Function` to create rows for each of the entries on the `known_for_titles`.*

9.0.2 - *Check the data types for the entries in the `birth_year` as well as the `death_years`.*

.

10 Table 3

[8]:

```python
print(df3.shape)
print()
print(df3.info())
print()
```

```
(331703, 8)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331703 entries, 0 to 331702
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   title_id           331703 non-null  object
 1   ordering           331703 non-null  int64
 2   title              331702 non-null  object
 3   region             278410 non-null  object
 4   language           41715 non-null   object
 5   types              168447 non-null  object
 6   attributes         14925 non-null   object
 7   is_original_title  331678 non-null  float64
dtypes: float64(1), int64(1), object(6)
memory usage: 20.2+ MB
```

```
None
```

[9]:

```
df3.head(2)
```

[9]:

```
    title_id  ordering               title region language        types  \
0  tt0369610        10                        BG       bg          NaN
1  tt0369610        11  Jurashikku warudo     JP      NaN  imdbDisplay

  attributes  is_original_title
0        NaN                0.0
1        NaN                0.0
```

11 Basic Interpletation

.

11.0.1 - This table has over 330000 entries with only 7 columns.

11.0.2 - The columns point out the titles, the film's language, the film's attributes and the and whether the film is an original title of not.

12 Suggestion(s)

12.0.1 - *Check the individual meanings of the entries in `types`, `ordering` and `attributes`.* ### - Change the entries in the column `is_original_title` to reflect either `no`, `yes` or `unknown`.

.

13 Table 4

[10]:

```
print(df4.shape)
print()
print(df4.info())
print()
```

```
(146144, 6)

<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    146143 non-null  object
 2   original_title   146122 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
None
```

[11]:

```
df4.head(2)
```

[11]:

```
      tconst                    primary_title   original_title  start_year  \
0  tt0063540                        Sunghursh        Sunghursh        2013
1  tt0066787  One Day Before the Rainy Season  Ashad Ka Ek Din        2019

   runtime_minutes              genres
0            175.0  Action,Crime,Drama
1            114.0     Biography,Drama
```

14 Basic Interpletation

.

14.0.1 - This table has over 146000 entries with only 6 columns.

14.0.2 - The columns point out the original titles, the start year of sale, how long the film is the film and the film's genre.

15 Suggestion(s)

15.0.1 - *Drop primary_title and retain original_title. .

16 Table 5

[12]:

```
print(df5.shape)
print()
print(df5.info())
print()
```

```
(146144, 3)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 3 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   tconst     146144 non-null  object
 1   directors  140417 non-null  object
 2   writers    110261 non-null  object
dtypes: object(3)
```

```
memory usage: 3.3+ MB
None
```

[13]:
```
df5.head(2)
```

[13]:
```
      tconst  directors            writers
0  tt0285252  nm0899854          nm0899854
1  tt0438973        NaN  nm0175726,nm1802864
```

17 Basic Interpletation

.

17.0.1 - This table has over 146000 entries with only 3 columns.

17.0.2 - The columns seem to point out the directors and writers of the film title.

18 Suggestion(s)

18.0.1 - *Find out what the column tconst alludes to. .

19 Table 6

[14]:
```
print(df6.shape)
print()
print(df6.info())
print()
```

```
(1028186, 6)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028186 entries, 0 to 1028185
Data columns (total 6 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   tconst      1028186 non-null  object
 1   ordering    1028186 non-null  int64
 2   nconst      1028186 non-null  object
 3   category    1028186 non-null  object
 4   job         177684 non-null   object
 5   characters  393360 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.1+ MB
None
```

[15]:

```
df6.head(2)
```

[15]:

```
      tconst  ordering     nconst  category  job   characters
0  tt0111414         1  nm0246005     actor  NaN  ["The Man"]
1  tt0111414         2  nm0398271  director  NaN          NaN
```

20 Basic Interpletation

.

20.0.1 - This table has over 100000 entries with only 6 columns

20.0.2 - The columns seem to point out the rating and the number of votes for each title or principal in data

21 Suggestion(s)

21.0.1 - *Find out what the column `Ordering` and `nconst` alludes to. .

22 Table 7

[16]:

```
print(df7.shape)
print()
print(df7.info())
print()
```

```
(73856, 3)

<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
None
```

[17]:

```
df7.head(2)
```

[17]:

```
       tconst  averagerating  numvotes
0  tt10356526            8.3        31
1  tt10384606            8.9       559
```

23 Basic Interpletation

.

23.0.1 - This table has over 73000 entries with only 3 columns

23.0.2 - The columns seem to point out the rating and the number of votes for each title or principal in data

24 Suggestion(s)

24.0.1 - *Find out what the column tconst alludes to. .

25 Table 8

[18]:

```
print(df8.shape)
print()
print(df8.info())
print()
```

```
(26517, 9)

<class 'pandas.core.frame.DataFrame'>
Index: 26517 entries, 0 to 26516
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   genre_ids          26517 non-null  object
 1   id                 26517 non-null  int64
 2   original_language  26517 non-null  object
 3   original_title     26517 non-null  object
 4   popularity         26517 non-null  float64
 5   release_date       26517 non-null  object
 6   title              26517 non-null  object
 7   vote_average       26517 non-null  float64
 8   vote_count         26517 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 2.0+ MB
None
```

[19]:

```
df8.head(2)
```

[19]:

```
                genre_ids     id original_language  \
0      [12, 14, 10751]  12444                en
1  [14, 12, 16, 10751]  10191                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

                                          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
```

26 Basic Interpletation

.

26.0.1 - This table has over 26000 entries with only 9 columns

26.0.2 - The columns seem to point out the populality of each title in data

27 Suggestion(s)

27.0.1 - *Drop the column original_title and retain title.*

.

28 Table 9

[20]:

```
print(df9.shape)
print()
print(df9.info())
print()
```

```
(5782, 6)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
None
```

[21]:

```
df9.head(2)
```

```
[21]:    id  release_date                                        movie  \
      0   1  Dec 18, 2009                                       Avatar
      1   2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides

        production_budget domestic_gross worldwide_gross
      0      $425,000,000   $760,507,625  $2,776,345,279
      1      $410,600,000   $241,063,875  $1,045,663,875
```

29 Basic Interpletation

.

29.0.1 - This table has over 5000 entries with only 6 columns

29.0.2 - This table has the title budget, domestic and worlwide gross incomes

30 Suggestions

30.0.1 - *changing the column name `movie` to `title`.*

.

31 Changing the name of `movie` column to `title`

```
[22]: df9.rename(columns = {'movie':'title'}, inplace=True)
      df9.head(1)
```

```
[22]:    id  release_date   title production_budget domestic_gross worldwide_gross
      0   1  Dec 18, 2009  Avatar      $425,000,000   $760,507,625  $2,776,345,279
```

32 Printing the names of columns Side-by-Side

```
[23]: DFs = [df1, df2, df3, df4, df5, df6, df7, df8, df9]

      for index, DF in enumerate(DFs):
          table_num = index + 1
          print(f'Table {table_num}:-' , sorted(list(DF.columns)),
                sep = '\n', end = '\n\n')
```

```
Table 1:-
['domestic_gross', 'foreign_gross', 'studio', 'title', 'year']

Table 2:-
['birth_year', 'death_year', 'known_for_titles', 'nconst', 'primary_name',
'primary_profession']
```

```
Table 3:-
['attributes', 'is_original_title', 'language', 'ordering', 'region', 'title',
'title_id', 'types']

Table 4:-
['genres', 'original_title', 'primary_title', 'runtime_minutes', 'start_year',
'tconst']

Table 5:-
['directors', 'tconst', 'writers']

Table 6:-
['category', 'characters', 'job', 'nconst', 'ordering', 'tconst']

Table 7:-
['averagerating', 'numvotes', 'tconst']

Table 8:-
['genre_ids', 'id', 'original_language', 'original_title', 'popularity',
'release_date', 'title', 'vote_average', 'vote_count']

Table 9:-
['domestic_gross', 'id', 'production_budget', 'release_date', 'title',
'worldwide_gross']
```

33 Short Interpletation

. > ### - Nearly all tables have a `title`'s column. > ### - This means we can merge the tables along this column.

34 MERGING TABLES

Let us start with **`Table 8`** and **`Table 9`**. And let us call said result **`Table 89`**.

35 **`Function`** to check which columns are common between any 2 dataframes

[24]:

```
def column_check(dat1, dat2):
    """
    This function checks to see whether there are columns
    in common.

    Function counterchecks whether any column in `dat1` has any
    of its columns in `dat2` columns.
    """
    for i in dat1.columns:
        if i in dat2.columns:
            print('Yes, there is a column in common:', i)

    else:
        # print('\nNo, Sadly there is no columns are similar.')
        print('\nThe End!')
```

36 Check common columns in df8 and df9.

[25]:

```
# Check Columns in common
column_check(df8, df9)
```

```
Yes, there is a column in common: id
Yes, there is a column in common: release_date
Yes, there is a column in common: title

The End!
```

[26]:

```
# df8.columns.tolist()
# df9.columns.tolist()
```

[27]:

```
# df8.head(1)
# df9.head(1)

print('The 2 DataFrame have this kind of shapes:-', end = '\n\n')
print('df8 has', df8.shape) # (26517, 9)
print('df9 has', df9.shape) # (5782, 6)
```

```
The 2 DataFrame have this kind of shapes:-

df8 has (26517, 9)
df9 has (5782, 6)
```

[28]:

```
df_89 = df8.merge(df9, how = 'outer', on = 'title', suffixes=('_fr8', '_fr9'))
df_89.head(3)
```

[28]:

```
             genre_ids   id_fr8 original_language  \
0      [12, 14, 10751]  12444.0                en
1  [14, 12, 16, 10751]  10191.0                en
2        [12, 28, 878]  10138.0                en

                                  original_title  popularity release_date_fr8  \
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

                                          title  vote_average  vote_count  \
0  Harry Potter and the Deathly Hallows: Part 1           7.7     10788.0
1                      How to Train Your Dragon           7.7      7610.0
2                                    Iron Man 2           6.8     12368.0

   id_fr9 release_date_fr9 production_budget domestic_gross worldwide_gross
0     NaN              NaN               NaN            NaN             NaN
1    30.0     Mar 26, 2010      $165,000,000   $217,581,232    $494,870,992
2    15.0      May 7, 2010      $170,000,000   $312,433,331    $621,156,389
```

We have merged these 2 tables along the `title` column. Also we have done so with the the condition `outer` inorder to retain the rich myriad entries we would like to analyse later on.

.

Let us start with **`Table 6`** and **`Table 7`**. And let us call said result **`Table 67`**.

[29]:
```python
# df6.head(1)
# df7.head(1)

print('The 2 DataFrame have this kind of shapes:-', end = '\n\n')
print('df6 has', df6.shape) # (1028186, 6)
print('df7 has', df7.shape) # (73856, 3)
```

```
The 2 DataFrame have this kind of shapes:-

df6 has (1028186, 6)
df7 has (73856, 3)
```

[30]:
```python
# Check Columns in common
column_check(df6, df7)
```

```
Yes, there is a column in common: tconst

The End!
```

[31]:
```python
df_67 = df6.merge(df7, how='outer', on='tconst', suffixes=('_fr6', '_fr7'))
df_67.head(3)
```

[31]:
```
      tconst  ordering     nconst  category       job   characters  \
0  tt0111414       1.0  nm0246005     actor       NaN  ["The Man"]
1  tt0111414       2.0  nm0398271  director       NaN          NaN
2  tt0111414       3.0  nm3739909  producer  producer          NaN

   averagerating  numvotes
0            NaN       NaN
```

```
1             NaN       NaN
2             NaN       NaN
```

Let us start with **`Table 5`** and **`Table 67`**. And let us call said result **`Table 67`**.

[32]:
```
# Check Columns in common
column_check(df5, df_67)
```

```
Yes, there is a column in common: tconst

The End!
```

[33]:
```
# df5.columns.tolist()
# df5.head(1)
# df67.head(1)

df_567 = df5.merge(df_67, how='outer', on='tconst', suffixes=('_fr5', '_fr67'))
df_567.head(3)
```

[33]:
```
      tconst  directors    writers  ordering     nconst  category  job  \
0  tt0285252  nm0899854  nm0899854      10.0  nm1077681  composer  NaN
1  tt0285252  nm0899854  nm0899854       1.0  nm0960950     actor  NaN
2  tt0285252  nm0899854  nm0899854       2.0  nm0461311     actor  NaN

          characters  averagerating  numvotes
0                NaN            3.9     219.0
1  ["Darren Fields"]            3.9     219.0
2             ["RJ"]            3.9     219.0
```

Let us start with **`Table 4`** and **`Table 567`**. And let us call said result **`Table 4567`**.

[34]:
```
# Check Columns in common
column_check(df4, df_567)
```

```
Yes, there is a column in common: tconst

The End!
```

First we must make one of the column of titles to `title`.

[35]:
```
# df4.columns.tolist()
# df4.head(2)

df4.rename(columns={'primary_title':'title'}, inplace = True)

df4.head(3)
```

[35]:
```
      tconst                             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

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

Now to check whether the column `title` is common to bother Dataframe.

[36]:
```
# Check Columns in common
column_check(df4, df_567)
```

```
Yes, there is a column in common: tconst

The End!
```

[37]:
```
# df4.head(1)
# df_567.head(1)

df_4to7 = df4.merge(df_567, how='outer', on='tconst', suffixes=('_fr5',␣
 ↪'_fr67'))
df_4to7.head(3)
```

[37]:
```
      tconst      title original_title  start_year  runtime_minutes  \
0  tt0063540  Sunghursh      Sunghursh        2013            175.0
1  tt0063540  Sunghursh      Sunghursh        2013            175.0
2  tt0063540  Sunghursh      Sunghursh        2013            175.0

               genres  directors                                writers  \
0  Action,Crime,Drama  nm0712540  nm0023551,nm1194313,nm0347899,nm1391276
1  Action,Crime,Drama  nm0712540  nm0023551,nm1194313,nm0347899,nm1391276
2  Action,Crime,Drama  nm0712540  nm0023551,nm1194313,nm0347899,nm1391276

   ordering     nconst  category  job                       characters  \
0      10.0  nm0006210  composer  NaN                              NaN
1       1.0  nm0474801     actor  NaN  ["Kundan S. Prasad","Bajrangi"]
2       2.0  nm0904537   actress  NaN      ["Munni","Laila-E-Aasmaan"]

   averagerating  numvotes
0            7.0      77.0
1            7.0      77.0
2            7.0      77.0
```

Let us start with **`Table 3`** and **`Table 4to7`**. And let us call said result **`Table 3to7`**.

[38]:
```
# Check Columns in common
column_check(df3, df_4to7)
```

```
Yes, there is a column in common: ordering
Yes, there is a column in common: title

The End!
```

[39]:

```
# df3.head(1)
# df_4567.head(1)

df_3to7 = df3.merge(df_4to7, how='outer', on='title', suffixes=('_fr3',␣
 ↪'_fr4to7'))
df_3to7.head(3)
```

[39]:

```
    title_id  ordering_fr3                                    title region  \
0  tt0369610          10.0                                        BG        
1  tt0369610          11.0                        Jurashikku warudo     JP   
2  tt0369610          12.0  Jurassic World: O Mundo dos Dinossauros     BR   

  language        types attributes  is_original_title tconst original_title  \
0       bg          NaN        NaN                0.0    NaN            NaN   
1      NaN  imdbDisplay        NaN                0.0    NaN            NaN   
2      NaN  imdbDisplay        NaN                0.0    NaN            NaN   

   …  genres  directors writers ordering_fr4to7 nconst  category  job  \
0  …     NaN        NaN     NaN             NaN    NaN       NaN  NaN   
1  …     NaN        NaN     NaN             NaN    NaN       NaN  NaN   
2  …     NaN        NaN     NaN             NaN    NaN       NaN  NaN   

  characters averagerating numvotes
0        NaN           NaN      NaN
1        NaN           NaN      NaN
2        NaN           NaN      NaN

[3 rows x 22 columns]
```

Let us start with **Table 1** and **Table 3to7**. And let us call said result **Table 1and3to7**.

[40]:

```
# Check Columns in common
column_check(df1, df_3to7)
```

```
Yes, there is a column in common: title

The End!
```

[41]:

```
# df1.head(1)
# df_34567.head(1)

df_1and3to7 = df1.merge(df_3to7, how='outer', on='title', suffixes=('_fr1',␣
 ↪'_fr3to7'))
```

```
df_1and3to7.head(3)
```

[41]:

```
         title studio  domestic_gross foreign_gross    year   title_id  \
0  Toy Story 3     BV     415000000.0     652000000  2010.0  tt0435761
1  Toy Story 3     BV     415000000.0     652000000  2010.0  tt0435761
2  Toy Story 3     BV     415000000.0     652000000  2010.0  tt0435761

   ordering_fr3 region language types  …                      genres  \
0          15.0     DK      NaN   NaN  …  Adventure,Animation,Comedy
1          15.0     DK      NaN   NaN  …  Adventure,Animation,Comedy
2          15.0     DK      NaN   NaN  …  Adventure,Animation,Comedy

   directors                                  writers ordering_fr4to7  \
0  nm0881279  nm0005124,nm0004056,nm0881279,nm1578335            10.0
1  nm0881279  nm0005124,nm0004056,nm0881279,nm1578335             1.0
2  nm0881279  nm0005124,nm0004056,nm0881279,nm1578335             2.0

      nconst  category  job          characters averagerating  numvotes
0  nm0005271  composer  NaN                 NaN           8.3  682218.0
1  nm0000158     actor  NaN           ["Woody"]           8.3  682218.0
2  nm0000741     actor  NaN  ["Buzz Lightyear"]           8.3  682218.0

[3 rows x 26 columns]
```

Let us start with `Table 2` and `Table 1and3to7`. And let us call said result `Table 1to7`.

[42]:

```
# Check Columns in common
column_check(df2, df_1and3to7)
```

```
Yes, there is a column in common: nconst

The End!
```

[43]:

```
df_1to7 = df2.merge(df_1and3to7, how='outer', on='nconst', suffixes=('_fr2', '_fr1_3to7'))
df_1to7.head(3)
```

[43]:

```
      nconst       primary_name  birth_year  death_year  \
0  nm0061671  Mary Ellen Bauder         NaN         NaN
1  nm0061671  Mary Ellen Bauder         NaN         NaN
2  nm0061671  Mary Ellen Bauder         NaN         NaN

                       primary_profession  \
0  miscellaneous,production_manager,producer
1  miscellaneous,production_manager,producer
2  miscellaneous,production_manager,producer

                       known_for_titles                     title studio  \
0  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony
1  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony
2  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony

   domestic_gross foreign_gross  …  runtime_minutes  \
0      45000000.0     152200000  …             90.0
1      45000000.0     152200000  …             90.0
2      45000000.0     152200000  …             90.0

                     genres  directors                        writers  \
0  Adventure,Animation,Comedy  nm0038432  nm1632630,nm0962596,nm0678963
1  Adventure,Animation,Comedy  nm0038432  nm1632630,nm0962596,nm0678963
2  Adventure,Animation,Comedy  nm0038432  nm1632630,nm0962596,nm0678963

  ordering_fr4to7  category       job  characters averagerating numvotes
0             9.0  producer  producer         NaN           6.0  15612.0
1             9.0  producer  producer         NaN           6.0  15612.0
2             9.0  producer  producer         NaN           6.0  15612.0

[3 rows x 31 columns]
```

Let us start with **Table 1to7** and **Table 89**. And let us call said result **Table 1to9**.‘

[44]:
```
# Check Columns in common
column_check(df_1to7, df_89)
```

```
Yes, there is a column in common: title
Yes, there is a column in common: domestic_gross
Yes, there is a column in common: original_title

The End!
```

[45]:
```
df_1to9 = df_1to7.merge(df_89, how='outer', on='title', suffixes=('_fr1to7',␣
 ↪'_fr89'))
df_1to9.head(3)
```

[45]:
```
      nconst       primary_name  birth_year  death_year  \
0  nm0061671  Mary Ellen Bauder         NaN         NaN
1  nm0061671  Mary Ellen Bauder         NaN         NaN
2  nm0061671  Mary Ellen Bauder         NaN         NaN

                      primary_profession  \
0  miscellaneous,production_manager,producer
1  miscellaneous,production_manager,producer
2  miscellaneous,production_manager,producer

                      known_for_titles                     title studio  \
0  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony
```

```
1  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony
2  tt0837562,tt2398241,tt0844471,tt0118553  Smurfs: The Lost Village   Sony

   domestic_gross_fr1to7 foreign_gross  …       original_title_fr89  \
0             45000000.0     152200000  …  Smurfs: The Lost Village
1             45000000.0     152200000  …  Smurfs: The Lost Village
2             45000000.0     152200000  …  Smurfs: The Lost Village

  popularity  release_date_fr8 vote_average vote_count id_fr9  \
0     15.663        2017-04-07          6.2      736.0    5.0
1     15.663        2017-04-07          6.2      736.0    5.0
2     15.663        2017-04-07          6.2      736.0    5.0

  release_date_fr9  production_budget domestic_gross_fr89 worldwide_gross
0      Apr 7, 2017        $60,000,000         $45,020,282    $197,578,586
1      Apr 7, 2017        $60,000,000         $45,020,282    $197,578,586
2      Apr 7, 2017        $60,000,000         $45,020,282    $197,578,586

[3 rows x 44 columns]
```

[46]:

```
# df_1to9.columns.tolist()
```

[47]:

```
cols_1to9 = [
 # film attributes
 'title_id', 'tconst', 'title', 'is_original_title', 'original_title_fr1to7',
 'original_title_fr89', 'studio', 'runtime_minutes', 'attributes', 'genres',␣
 ↪'genre_ids',
 'types', 'category', 'characters',
 # Timelines
 'year', 'start_year', 'release_date_fr8', 'release_date_fr9',
 # Geo
 'region', 'language', 'original_language',
 # finances
 'production_budget', 'domestic_gross_fr1to7', 'domestic_gross_fr89',␣
 ↪'foreign_gross',
 'worldwide_gross',
 # unknowns
 'id_fr8', 'id_fr9', 'ordering_fr3', 'ordering_fr4to7',
 # professionals
 'nconst', 'primary_name', 'birth_year', 'death_year', 'primary_profession',␣
 ↪'known_for_titles',
 'directors', 'writers', 'job',
 # Popularity scores
 'averagerating', 'numvotes', 'popularity', 'vote_average', 'vote_count'
]

# len(cols_1to9) == len(final_1_cols)
# print(len(final_1_cols))
```

37 Delving into the Data Now

[48]:

```
df_19 = df_1to9[cols_1to9]
df_19.head()
```

[48]:

```
    title_id     tconst                     title  is_original_title  \
0  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
1  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
2  tt2398241  tt2398241  Smurfs: The Lost Village                1.0
3  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
4  tt2398241  tt2398241  Smurfs: The Lost Village                0.0

      original_title_fr1to7       original_title_fr89 studio  runtime_minutes  \
0  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
1  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
2  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
3  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
4  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0

  attributes                      genres  …  \
0        NaN  Adventure,Animation,Comedy  …
1        NaN  Adventure,Animation,Comedy  …
2        NaN  Adventure,Animation,Comedy  …
3        NaN  Adventure,Animation,Comedy  …
4        NaN  Adventure,Animation,Comedy  …

                             primary_profession  \
0     miscellaneous,production_manager,producer
1     miscellaneous,production_manager,producer
2     miscellaneous,production_manager,producer
3  art_department,animation_department,director
4  art_department,animation_department,director

                          known_for_titles  directors  \
0  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
1  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
2  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
3  tt0298148,tt0101414,tt0166813,tt0377981  nm0038432
4  tt0298148,tt0101414,tt0166813,tt0377981  nm0038432

                         writers       job  averagerating numvotes popularity  \
0  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
1  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
2  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
```

```
3  nm1632630,nm0962596,nm0678963       NaN            6.0  15612.0     15.663
4  nm1632630,nm0962596,nm0678963       NaN            6.0  15612.0     15.663

  vote_average vote_count
0          6.2      736.0
1          6.2      736.0
2          6.2      736.0
3          6.2      736.0
4          6.2      736.0

[5 rows x 44 columns]
```

38 Thorough Check

```
[49]: df_19.head(3)
```

```
[49]:     title_id     tconst                     title  is_original_title  \
0  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
1  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
2  tt2398241  tt2398241  Smurfs: The Lost Village                1.0

      original_title_fr1to7       original_title_fr89 studio  runtime_minutes  \
0  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
1  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
2  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0

  attributes                      genres  …  \
0        NaN  Adventure,Animation,Comedy  …
1        NaN  Adventure,Animation,Comedy  …
2        NaN  Adventure,Animation,Comedy  …

                          primary_profession  \
0  miscellaneous,production_manager,producer
1  miscellaneous,production_manager,producer
2  miscellaneous,production_manager,producer

                          known_for_titles  directors  \
0  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
1  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
2  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432

                         writers       job  averagerating numvotes popularity  \
0  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
1  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
2  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
```

```
  vote_average vote_count
0          6.2      736.0
1          6.2      736.0
2          6.2      736.0

[3 rows x 44 columns]
```

[50]:

```
# df_19.columns.tolist()

df_19.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2349428 entries, 0 to 2349427
Data columns (total 44 columns):
 #   Column                 Dtype
---  ------                 -----
 0   title_id               object
 1   tconst                 object
 2   title                  object
 3   is_original_title      float64
 4   original_title_fr1to7  object
 5   original_title_fr89    object
 6   studio                 object
 7   runtime_minutes        float64
 8   attributes             object
 9   genres                 object
 10  genre_ids              object
 11  types                  object
 12  category               object
 13  characters             object
 14  year                   float64
 15  start_year             float64
 16  release_date_fr8       object
 17  release_date_fr9       object
 18  region                 object
 19  language               object
 20  original_language      object
 21  production_budget      object
 22  domestic_gross_fr1to7  float64
 23  domestic_gross_fr89    object
 24  foreign_gross          object
 25  worldwide_gross        object
 26  id_fr8                 float64
 27  id_fr9                 float64
 28  ordering_fr3           float64
 29  ordering_fr4to7        float64
 30  nconst                 object
 31  primary_name          object
 32  birth_year            float64
 33  death_year            float64
 34  primary_profession    object
 35  known_for_titles      object
 36  directors             object
 37  writers               object
 38  job                   object
 39  averagerating         float64
 40  numvotes              float64
 41  popularity            float64
 42  vote_average          float64
 43  vote_count            float64
dtypes: float64(16), object(28)
memory usage: 788.7+ MB
```

39 (i) Film attributes

[51]:

```
film_attributes = ['title_id', 'tconst', 'title', 'is_original_title',␣
 ↪'original_title_fr1to7',
 'original_title_fr89', 'studio', 'runtime_minutes', 'attributes', 'genres',␣
 ↪'genre_ids',
 'types', 'category', 'characters']

df_19[film_attributes].head(3)
```

[51]:

```
    title_id     tconst                     title  is_original_title  \
0  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
1  tt2398241  tt2398241  Smurfs: The Lost Village                0.0
2  tt2398241  tt2398241  Smurfs: The Lost Village                1.0

      original_title_fr1to7       original_title_fr89 studio  runtime_minutes  \
0  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
1  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0
2  Smurfs: The Lost Village  Smurfs: The Lost Village   Sony             90.0

  attributes                       genres            genre_ids        types  \
0        NaN  Adventure,Animation,Comedy  [12, 16, 35, 10751]          NaN
1        NaN  Adventure,Animation,Comedy  [12, 16, 35, 10751]  imdbDisplay
2        NaN  Adventure,Animation,Comedy  [12, 16, 35, 10751]     original

   category characters
0  producer        NaN
1  producer        NaN
2  producer        NaN
```

40 Film Attributes Actionables

.(i) We can remove the `original_title_fr1to7` as well as `original_title_fr89` from **film Attributes** because they are similar if not the same as the entries in the `title`.

.(ii) We can change the atributes on the `is_original_title` to `Yes`, `No` and `Unknown`.

.(iii) We can change the dtype of values along the `runtime_minutes` from a `float64` to `int64`.

.(iv) Check the values along `genres` and `genres_ids`.

[52]:

```
# Executable 1
film_attributes = ['title_id', 'tconst', 'title', 'is_original_title',␣
 ↪'studio', 'runtime_minutes', 'attributes', 'genres', 'genre_ids',
 'types', 'category', 'characters']

# Executable 2
df_19['is_original_title'].replace({0.0: 'No', 1.0: 'Yes', np.nan:'Unknown'},␣
 ↪inplace = True)

df_19[film_attributes].head(3)
```

```
C:\Users\rurig\AppData\Local\Temp\ipykernel_15900\2988573533.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_19['is_original_title'].replace({0.0: 'No', 1.0: 'Yes', np.nan:'Unknown'},
inplace = True)
```

[52]:

```
    title_id     tconst                     title is_original_title studio  \
0  tt2398241  tt2398241  Smurfs: The Lost Village                No   Sony
1  tt2398241  tt2398241  Smurfs: The Lost Village                No   Sony
2  tt2398241  tt2398241  Smurfs: The Lost Village               Yes   Sony

   runtime_minutes attributes                       genres  \
0             90.0        NaN  Adventure,Animation,Comedy
1             90.0        NaN  Adventure,Animation,Comedy
2             90.0        NaN  Adventure,Animation,Comedy

             genre_ids        types  category characters
0  [12, 16, 35, 10751]          NaN  producer        NaN
1  [12, 16, 35, 10751]  imdbDisplay  producer        NaN
2  [12, 16, 35, 10751]     original  producer        NaN
```

41 NOTE

[53]:

```
shape_0 = df_19.shape

print(f'There are {shape_0} currently in the dataframe.')
```

```
There are (2349428, 44) currently in the dataframe.
```

[54]:

```
# First check how many null values in datasets
nulls_on_runtime = len(df_19.loc[df_19.runtime_minutes.isnull()])
print(f'There are {nulls_on_runtime} null values on the runtime column.') #␣
  ↪518745

# NExt, drop `null` values along the columns
df_19.dropna(subset = ['runtime_minutes'], inplace = True)
```

```
There are 518745 null values on the runtime column.
```

```
C:\Users\rurig\AppData\Local\Temp\ipykernel_15900\2224123142.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_19.dropna(subset = ['runtime_minutes'], inplace = True)
```

42 NOTE

[55]:

```
shape_1 = df_19.shape

print(f"""There are {shape_1} currently in the dataframe. This is a right after
dropping all null values on our runtime minutes column.""")
```

```
There are (1830683, 44) currently in the dataframe. This is a right after
dropping all null values on our runtime minutes column.
```

[56]:

```
df_19[film_attributes][:3].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, 0 to 2
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title_id           3 non-null      object
 1   tconst             3 non-null      object
 2   title              3 non-null      object
 3   is_original_title  3 non-null      object
 4   studio             3 non-null      object
 5   runtime_minutes    3 non-null      float64
 6   attributes         0 non-null      object
 7   genres             3 non-null      object
 8   genre_ids          3 non-null      object
 9   types              2 non-null      object
 10  category           3 non-null      object
 11  characters         0 non-null      object
dtypes: float64(1), object(11)
memory usage: 312.0+ bytes
```

[57]:

```
# Executable 3

print(df_19.runtime_minutes.dtype)
df_19.runtime_minutes = df_19.runtime_minutes.astype('int64')
# df_19.runtime_minutes.apply(lambda runtime_minutes: runtime_minutes.
  ↪astype('int64'))
```

```
float64

C:\Users\rurig\AppData\Local\Temp\ipykernel_15900\2517915611.py:4:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_19.runtime_minutes = df_19.runtime_minutes.astype('int64')
```

[58]:

```
print(f'Now the .dtype has been changed to: {df_19.runtime_minutes.dtype}.')
```

```
Now the .dtype has been changed to: int64.
```

43 NOTE

[59]:

```
# Executable 4
print(f'There are {df_19[film_attributes].genres.isnull().sum()} null values on␣
  ↪the genre column.')
print()
print(f'Also, there are {df_19[film_attributes].genre_ids.isnull().sum()} null␣
  ↪values on the genre column.')
```

```
There are 18014 null values on the genre column.

Also, there are 969767 null values on the genre column.
```

The `genres_ids` have significant null values than the `genre`. It would be prompt for us to remove the column altogether.

[60]:

```
film_attributes = [ 'title_id', 'tconst', 'title', 'is_original_title',␣
 ↪'studio',
                   'runtime_minutes', 'attributes', 'genres', 'types',␣
 ↪'category',
                   'characters']
```

44 (ii) Timelines and Geo attributes

[61]:

```
timelines_geo_cols = [
 # Timelines
 'year', 'start_year', 'release_date_fr8', 'release_date_fr9',
 # Geo
 'region', 'language', 'original_language'
]
```

[62]:

```
df_19[timelines_geo_cols].head(3)
```

[62]:

```
     year  start_year release_date_fr8 release_date_fr9 region language  \
0  2017.0      2017.0       2017-04-07      Apr 7, 2017     US      NaN
1  2017.0      2017.0       2017-04-07      Apr 7, 2017     CA       en
2  2017.0      2017.0       2017-04-07      Apr 7, 2017    NaN      NaN

  original_language
0                en
1                en
2                en
```

[63]:

```
df_19[timelines_geo_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1830683 entries, 0 to 2337272
Data columns (total 7 columns):
 #   Column             Dtype
---  ------             -----
 0   year               float64
 1   start_year         float64
 2   release_date_fr8   object
 3   release_date_fr9   object
 4   region             object
 5   language           object
 6   original_language  object
dtypes: float64(2), object(5)
memory usage: 111.7+ MB
```

45 Timeline and Geo Actionables

.(i) Change the `.dtype` of the `year` and `start_year` from `float64` to `int64`.

.(ii) Drop drop the `release_date_fr9` column since it is similar to the entries along `release_date_fr8` columns.

.(i) Check the `language` and `original_language` columns. See whether they are somewhat similar and decide which to drop.

[64]:
```
# Executable 1
print(df_19.year.dtype)
print(df_19.start_year.dtype)
```

```
float64
float64
```

[65]:
```
# df_19.loc[df_19.year.isnull()]
print(f"""The number of null values along the `year` column are {df_19.year.
 ↪isnull().sum()}.""",
      end ='\n\n')

print(f'Also, the number of null values along the `start_year` column are␣
 ↪{df_19.start_year.isnull().sum()}.')
```

```
The number of null values along the `year` column are 1617649.

Also, the number of null values along the `start_year` column are 0.
```

NOTE

From this analysis; it would be sufficive to remove the `year` column altogether and retain the `start_year` column since the former has more *null* values whereas `start_year` has none.

[66]:
```
# checking whether we have a null value in this column
np.nan in df_19.start_year.unique()
```

[66]:
```
False
```

[67]:
```
df_19.start_year = df_19.start_year.astype('int64')

print(f'Now, we have:-')
print(df_19.start_year.dtype)
```

```
Now, we have:-
int64
```

```
C:\Users\rurig\AppData\Local\Temp\ipykernel_15900\1475042411.py:1:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_19.start_year = df_19.start_year.astype('int64')
```

[68]:
```
df_19[timelines_geo_cols].head(3)
```

[68]:
```
     year  start_year release_date_fr8 release_date_fr9 region language  \
0  2017.0        2017       2017-04-07      Apr 7, 2017     US      NaN
1  2017.0        2017       2017-04-07      Apr 7, 2017     CA       en
2  2017.0        2017       2017-04-07      Apr 7, 2017    NaN      NaN

  original_language
0                en
1                en
2                en
```

NOTE

From this analysis; the elements in `release_date_fr8` and `release_date_fr9` are similar. We shall retain the former because of it robust numerical elements.

[69]:
```
# Executable 2

timelines_geo_cols = [
 # Timelines
 'start_year', 'release_date_fr8',
 # Geo
 'region', 'language', 'original_language'
]
```

[70]:
```
# Executable 3

nulls_in_lang = df_19[timelines_geo_cols].language.unique()
no_nulls_in_lang = len(df_19[timelines_geo_cols].language.unique())

print(f'{nulls_in_lang} They are {no_nulls_in_lang} in number.', end = '\n\n')

nulls_in_orig_lang = df_19[timelines_geo_cols].original_language.unique()
no_nulls_in_orig_lang = len(df_19[timelines_geo_cols].original_language.
 ↪unique())

print(f"""{nulls_in_orig_lang} They are {no_nulls_in_orig_lang} in number.""",␣
 ↪end = '\n\n')
```

```
[nan 'en' 'tr' 'fr' 'he' 'ar' 'sv' 'bg' 'cmn' 'fa' 'ca' 'hi' 'nl' 'te'
 'de' 'it' 'id' 'ta' 'ml' 'af' 'es' 'bn' 'ur' 'lt' 'hr' 'kn' 'bs' 'mr'
 'pt' 'qbn' 'yue' 'ps' 'pa' 'gd' 'gu' 'gl' 'tl' 'th' 'sr'] They are 39 in
number.
```

```
['en' nan 'lo' 'de' 'ru' 'he' 'fr' 'es' 'sv' 'it' 'hi' 'pl' 'id' 'cn' 'zh'
 'uk' 'nl' 'tl' 'fa' 'ko' 'ja' 'no' 'el' 'mr' 'hr' 'te' 'pt' 'hu' 'tr'
 'vi' 'cs' 'da' 'xx' 'ar' 'sr' 'ca' 'is' 'ta' 'ro' 'sq' 'eu' 'ml' 'fi'
 'th' 'kn' 'dz' 'lv' 'gu' 'ur' 'ab' 'mi' 'ka' 'et' 'bg' 'kk' 'ku' 'lt'
 'cy' 'bn' 'bo' 'pa' 'hy' 'sn' 'sw' 'hz' 'yi' 'ky' 'ne' 'xh' 'af' 'cr'
 'ha'] They are 72 in number.
```

NOTE > - These are the languages in the `language` column:- [**nan ‘en’ ‘tr’ ‘fr’ ‘he’ ‘ar’ ‘sv’ ‘bg’ ‘cmn’ ‘fa’ ‘ca’ ‘hi’ ‘nl’ ‘te’ ‘de’ ‘it’ ‘id’ ‘ta’ ‘ml’ ‘af’ ‘es’ ‘bn’ ‘ur’ ‘lt’ ‘hr’ ‘kn’ ‘bs’ ‘mr’ ‘pt’ ‘qbn’ ‘yue’ ‘ps’ ‘pa’ ‘gd’ ‘gu’ ‘gl’ ‘tl’ ‘th’ ‘sr’**] They are 39 in number.

  - These are the languages in the `original_language` columns- [**‘en’ nan ‘lo’ ‘de’ ‘ru’ ‘he’ ‘fr’ ‘es’ ‘sv’ ‘it’ ‘hi’ ‘pl’ ‘id’ ‘cn’ ‘zh’ ‘uk’ ‘nl’ ‘tl’ ‘fa’ ‘ko’ ‘ja’ ‘no’ ‘el’ ‘mr’ ‘hr’ ‘te’ ‘pt’ ‘hu’ ‘tr’ ‘vi’ ‘cs’ ‘da’ ‘xx’ ‘ar’ ‘sr’ ‘ca’ ‘is’ ‘ta’ ‘ro’ ‘sq’ ‘eu’ ‘ml’ ‘fi’ ‘th’ ‘kn’ ‘dz’ ‘lv’ ‘gu’ ‘ur’ ‘ab’ ‘mi’ ‘ka’ ‘et’ ‘bg’ ‘kk’ ‘ku’ ‘lt’ ‘cy’ ‘bn’ ‘bo’ ‘pa’ ‘hy’ ‘sn’ ‘sw’ ‘hz’ ‘yi’ ‘ky’ ‘ne’ ‘xh’ ‘af’ ‘cr” ‘ha’**] They are 72 in number.mber.</

.

- Also, We need to check rows with null values.font>

[71]:
```python
# Checking to see how many null values are
# present in each of these cols

print(df_19.language.isnull().sum())
print(df_19.original_language.isnull().sum())
```

```
1570627
969767
```

NOTE

From this analysis, We note there are over 1,500,000 and 900000 null values in the `language` and `original_language` columns respectively.

It would only be prompt for us to drop the `language` and retain `original_language` since it has few null values than the `language` column.

[72]:
```python
timelines_geo_cols = [
 # Timelines
 'start_year', 'release_date_fr8',
 # Geo
 'region', 'original_language'
]
```

[73]:
```python
df_19[timelines_geo_cols].head()
```

[73]:

```
   start_year release_date_fr8 region original_language
0        2017       2017-04-07     US                en
1        2017       2017-04-07     CA                en
2        2017       2017-04-07    NaN                en
3        2017       2017-04-07     US                en
4        2017       2017-04-07     CA                en
```

46 (iii) Finances

[74]:

```
finances = [
 'production_budget', 'domestic_gross_fr1to7', 'domestic_gross_fr89',
 'foreign_gross', 'worldwide_gross'
]
```

[75]:

```
df_19[finances].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1830683 entries, 0 to 2337272
Data columns (total 5 columns):
 #   Column                 Dtype
---  ------                 -----
 0   production_budget      object
 1   domestic_gross_fr1to7  float64
 2   domestic_gross_fr89    object
 3   foreign_gross          object
 4   worldwide_gross        object
dtypes: float64(1), object(4)
memory usage: 83.8+ MB
```

NOTE

- There are inconsistencies in the `domestic_gross_fr1to7` and `domestic_gross_fr89` columns in terms of values.
- There are inconsistencies in the `foreign_gross` and `worldwide_gross` columns in terms of values.
- Convert the `dtypes` of the entries in `production_budget`, `domestic_gross_fr89` and `worldwide_gross`.

  From this analysis, we shall drop the columns with inconsitent values. One reason for this `domestic_gross_fr1to7` and `foreign_gross` has too exact figures that would raise an integrity issue.

[76]:

```
# executable 1 and 2
finances = [
 'production_budget', 'domestic_gross_fr89', 'worldwide_gross'
]
```

For the third executable, we create a `function` that will change the datatype to an `int`

[77]:
```
# # Executable 3
# def monetize(item):
#     if item != np.nan:
#         return int(item[1:].replace(',',''))

# df_19.production_budget = df_19.production_budget.map(monetize)
# df_19.domestic_gross = df_19.domestic_gross.map(monetize)
# df_19.worldwide_gross = df_19.worldwide_gross.map(monetize)
```

[78]:
```
df_19[finances].head(3)
```

[78]:
```
  production_budget domestic_gross_fr89 worldwide_gross
0       $60,000,000         $45,020,282    $197,578,586
1       $60,000,000         $45,020,282    $197,578,586
2       $60,000,000         $45,020,282    $197,578,586
```

Suggestion

- Check on how to numerise the values along the `production_budget`.
- Change the column name of `domestic_gross_fr89`.

47 (iv) Unknowns

[79]:
```
unknowns = [
 # unknowns
 'id_fr8', 'id_fr9', 'ordering_fr3', 'ordering_fr4to7',
 # # professionals
 # 'nconst', 'primary_name', 'birth_year', 'death_year', 'primary_profession',␣
 ↪'known_for_titles',
 # 'directors', 'writers', 'job',
 # # Popularity scores
 # 'averagerating', 'numvotes', 'popularity', 'vote_average', 'vote_count',
]
```

[80]:
```
df_19[unknowns].head(3)
```

[80]:
```
     id_fr8  id_fr9  ordering_fr3  ordering_fr4to7
0  137116.0     5.0          16.0              9.0
1  137116.0     5.0          26.0              9.0
2  137116.0     5.0           9.0              9.0
```

NOTE

There seem to be no fathomable entries along this columns. We shall ignore them for now.

48 (v) Professionals

[81]:

```
professionals = [
 'nconst', 'primary_name', 'birth_year', 'death_year', 'primary_profession',␣
 ↪'known_for_titles',
 'directors', 'writers', 'job'
]
```

[82]:

```
df_19[professionals].head(3)
```

[82]:

```
      nconst       primary_name  birth_year  death_year  \
0  nm0061671  Mary Ellen Bauder         NaN         NaN
1  nm0061671  Mary Ellen Bauder         NaN         NaN
2  nm0061671  Mary Ellen Bauder         NaN         NaN

                         primary_profession  \
0  miscellaneous,production_manager,producer
1  miscellaneous,production_manager,producer
2  miscellaneous,production_manager,producer

                         known_for_titles  directors  \
0  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
1  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
2  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432

                      writers       job
0  nm1632630,nm0962596,nm0678963  producer
1  nm1632630,nm0962596,nm0678963  producer
2  nm1632630,nm0962596,nm0678963  producer
```

49 Professionals Actionables

.(i) The entries on the `primary_profession` and `known_for_titles` as well as `writers` presents a need to unarchive the entries on each row for ease of analysis and interpretation of the data.

.(ii) We shall need to convert the `birth_year` and `death_year` to `int64` since they are discrete numerals.

.

.**Although** for now, we shall skip this step and take a look into it later.

[83]:

```
# check whether there are null values in 'birth_year' and 'death year'

np.nan in df_19.birth_year.unique()
np.nan in df_19.death_year.unique()
```

[83]:

```
False
```

[84]:

```
df_19.birth_year
```

[84]:

```
0             NaN
1             NaN
2             NaN
3          1960.0
4          1960.0
            …
2337245       NaN
2337253       NaN
2337260       NaN
2337271       NaN
2337272       NaN
Name: birth_year, Length: 1830683, dtype: float64
```

.**Although** for now, we shall skip this step and take a look into it later.

50 (vii) Popularity Scores

[85]:

```
popularity_scores = [
 'averagerating', 'numvotes', 'popularity', 'vote_average', 'vote_count'
]
```

[86]:

```
df_19[popularity_scores].head()
```

[86]:

```
   averagerating  numvotes  popularity  vote_average  vote_count
0            6.0   15612.0      15.663           6.2       736.0
1            6.0   15612.0      15.663           6.2       736.0
2            6.0   15612.0      15.663           6.2       736.0
3            6.0   15612.0      15.663           6.2       736.0
4            6.0   15612.0      15.663           6.2       736.0
```

[87]:

```
df_19[popularity_scores].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1830683 entries, 0 to 2337272
Data columns (total 5 columns):
 #   Column         Dtype
---  ------         -----
 0   averagerating  float64
 1   numvotes       float64
 2   popularity     float64
 3   vote_average   float64
 4   vote_count     float64
dtypes: float64(5)
memory usage: 83.8 MB
```

51 Popularity Scores Actionables

.(i) Convert the `numvotes` and the `vote_count` to `int64` since those would be discrete numerals.

.(i) The first 3 columns, i.e. `averagerating`, `numvotes` and `popularity`, are related and they include a `popularity` score.

.**We shall drop the last 2 columns and only use the first 3.**

```
[88]: # the final dataset is:-

      popularity_scores = [
       'averagerating', 'numvotes', 'popularity'
      ]
```

Final Dataset

```
[89]: final_cols = [
       # film_attributes
       'title_id', 'tconst', 'title', 'is_original_title', 'studio',␣
       ↪'runtime_minutes',
       'attributes', 'genres', 'types', 'category', 'characters',
       # Timelines
       'start_year', 'release_date_fr8',
       # Geo
       'region', 'original_language',
       # finances
       'production_budget', 'domestic_gross_fr89', 'worldwide_gross',
       # professionals
       'nconst', 'primary_name', 'birth_year', 'death_year', 'primary_profession',␣
       ↪'known_for_titles',
       'directors', 'writers', 'job',
       # popularity_scores
       'averagerating', 'numvotes', 'popularity'
      ]
```

```
[90]: # A glance at what we have now
      final_df = df_19[final_cols].copy()
      final_df.head(1)
```

```
[90]:     title_id     tconst                     title is_original_title studio  \
      0  tt2398241  tt2398241  Smurfs: The Lost Village                No   Sony

         runtime_minutes attributes                      genres types  category  \
      0              90        NaN  Adventure,Animation,Comedy   NaN  producer

         … birth_year  death_year                       primary_profession  \
      0  …        NaN         NaN  miscellaneous,production_manager,producer
```

```
                         known_for_titles  directors  \
0  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432

                          writers       job averagerating numvotes popularity
0  nm1632630,nm0962596,nm0678963  producer           6.0  15612.0     15.663

[1 rows x 30 columns]
```

SUGGESTIONS

(i) Rename the column names of `domestic_gross_fr89` to `domestic_gross`.

(ii) Change the datatypes of the Financial columns, e.g. `domestic_gross` to floats.

[91]:
```
# Follow-up on Suggestions
# (i)
# changing the column name of `domestic_gross_fr89`
final_df.rename(columns={'domestic_gross_fr89': 'domestic_gross'}, inplace =␣
 ↪True)

# Checking where the type has been made
'domestic_gross' in final_df.columns
```

[91]:
```
True
```

[92]:
```
# Follow-up on Suggestions
# (ii) Changing the datatypes of the financial columns
final_df['production_budget'] = final_df.production_budget.replace('[\$,]', '',␣
 ↪regex=True).astype(float)
final_df['domestic_gross'] = final_df['domestic_gross'].replace('[\$,]', '',␣
 ↪regex=True).astype(float)
final_df['worldwide_gross'] = final_df['worldwide_gross'].replace('[\$,]', '',␣
 ↪regex=True).astype(float)
```

[93]:
```
# Basic information on columns
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1830683 entries, 0 to 2337272
Data columns (total 30 columns):
 #   Column              Dtype
---  ------              -----
 0   title_id            object
 1   tconst              object
 2   title               object
 3   is_original_title   object
 4   studio              object
 5   runtime_minutes     int64
 6   attributes          object
 7   genres              object
 8   types               object
 9   category            object
 10  characters          object
 11  start_year          int64
 12  release_date_fr8    object
 13  region              object
 14  original_language   object
 15  production_budget   float64
 16  domestic_gross      float64
 17  worldwide_gross     float64
 18  nconst              object
 19  primary_name        object
 20  birth_year          float64
 21  death_year          float64
 22  primary_profession  object
 23  known_for_titles    object
 24  directors           object
 25  writers             object
 26  job                 object
 27  averagerating       float64
 28  numvotes            float64
 29  popularity          float64
dtypes: float64(8), int64(2), object(20)
memory usage: 433.0+ MB
```

[94]:

```
# Basic information on columns
final_df.describe()
```

[94]:

```
       runtime_minutes    start_year  production_budget  domestic_gross  \
count     1.830683e+06  1.830683e+06       3.238380e+05    3.238380e+05
mean      9.269764e+01  2.014385e+03       4.022453e+07    5.056320e+07
std       1.013134e+02  2.618585e+00       5.365119e+07    8.278120e+07
min       1.000000e+00  2.010000e+03       1.400000e+03    0.000000e+00
25%       8.000000e+01  2.012000e+03       4.357373e+06    1.543300e+04
50%       9.100000e+01  2.014000e+03       1.300000e+07    1.271149e+07
75%       1.050000e+02  2.017000e+03       5.100000e+07    5.825080e+07
max       5.142000e+04  2.022000e+03       4.250000e+08    7.605076e+08

       worldwide_gross     birth_year    death_year  averagerating  \
count     3.238380e+05  533065.000000  23946.000000   1.396436e+06
mean      1.247131e+08    1967.908733   1987.597386   6.214570e+00
std       2.117233e+08      23.442662     80.262915   1.351983e+00
min       0.000000e+00       1.000000     17.000000   1.000000e+00
25%       2.483790e+06    1960.000000   1999.000000   5.400000e+00
50%       4.479317e+07    1971.000000   2014.000000   6.400000e+00
```

```
75%      1.310118e+08   1980.000000   2017.000000   7.100000e+00
max      2.776345e+09   2014.000000   2019.000000   1.000000e+01

            numvotes     popularity
count  1.396436e+06  860916.000000
mean   1.478212e+04       5.114811
std    6.625076e+04       6.434947
min    5.000000e+00       0.600000
25%    3.300000e+01       0.701000
50%    2.040000e+02       2.379000
75%    1.502000e+03       7.620000
max    1.841066e+06      80.773000
```

52 EXPORT DATASET

[114]:
```
# Export the DataFrame
# final_df.to_csv('final_df.csv')



# > Commented out since it generate a `csv` that is quite sizable
```

[96]:
```
# Checking if it has been written on disk
# 'final_df.csv' in (! ls *.csv)
! ls *final_df.csv
```

```
final_df.csv
```

53 ANALYSIS

[97]:
```
df = final_df.copy()
df.head(3)
```

[97]:
```
    title_id     tconst                     title is_original_title studio  \
0  tt2398241  tt2398241  Smurfs: The Lost Village                No   Sony
1  tt2398241  tt2398241  Smurfs: The Lost Village                No   Sony
2  tt2398241  tt2398241  Smurfs: The Lost Village               Yes   Sony

   runtime_minutes attributes                     genres        types  \
0               90        NaN  Adventure,Animation,Comedy          NaN
1               90        NaN  Adventure,Animation,Comedy  imdbDisplay
2               90        NaN  Adventure,Animation,Comedy     original

   category  … birth_year  death_year  \
0  producer  …        NaN         NaN
1  producer  …        NaN         NaN
2  producer  …        NaN         NaN
```

```
                          primary_profession  \
0  miscellaneous,production_manager,producer
1  miscellaneous,production_manager,producer
2  miscellaneous,production_manager,producer

                          known_for_titles  directors  \
0  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
1  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432
2  tt0837562,tt2398241,tt0844471,tt0118553  nm0038432

                         writers       job  averagerating numvotes popularity
0  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
1  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663
2  nm1632630,nm0962596,nm0678963  producer            6.0  15612.0     15.663

[3 rows x 30 columns]
```

[98]:

```
# Finding the number of items in df's columns
# df.columns

df = df[['title', 'is_original_title', 'studio', 'runtime_minutes',
       'genres', 'start_year', 'original_language','region',
       'directors', 'writers', 'production_budget', 'domestic_gross',
       'worldwide_gross', 'averagerating', 'numvotes', 'popularity']]

df.head(3)
```

[98]:

```
                      title is_original_title studio  runtime_minutes  \
0  Smurfs: The Lost Village                No   Sony               90
1  Smurfs: The Lost Village                No   Sony               90
2  Smurfs: The Lost Village               Yes   Sony               90

                       genres  start_year original_language region  directors  \
0  Adventure,Animation,Comedy        2017                en     US  nm0038432
1  Adventure,Animation,Comedy        2017                en     CA  nm0038432
2  Adventure,Animation,Comedy        2017                en    NaN  nm0038432

                         writers  production_budget  domestic_gross  \
0  nm1632630,nm0962596,nm0678963         60000000.0      45020282.0
1  nm1632630,nm0962596,nm0678963         60000000.0      45020282.0
2  nm1632630,nm0962596,nm0678963         60000000.0      45020282.0

   worldwide_gross  averagerating  numvotes  popularity
0      197578586.0            6.0   15612.0      15.663
1      197578586.0            6.0   15612.0      15.663
2      197578586.0            6.0   15612.0      15.663
```

[99]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1830683 entries, 0 to 2337272
Data columns (total 16 columns):
 #   Column             Dtype
---  ------             -----
 0   title              object
 1   is_original_title  object
 2   studio             object
 3   runtime_minutes    int64
 4   genres             object
 5   start_year         int64
 6   original_language  object
 7   region             object
 8   directors          object
 9   writers            object
 10  production_budget  float64
 11  domestic_gross     float64
 12  worldwide_gross    float64
 13  averagerating      float64
 14  numvotes           float64
 15  popularity         float64
dtypes: float64(6), int64(2), object(8)
memory usage: 237.4+ MB
```

[100]:
```
df.describe()
```

[100]:
```
       runtime_minutes    start_year  production_budget  domestic_gross  \
count     1.830683e+06  1.830683e+06       3.238380e+05    3.238380e+05
mean      9.269764e+01  2.014385e+03       4.022453e+07    5.056320e+07
std       1.013134e+02  2.618585e+00       5.365119e+07    8.278120e+07
min       1.000000e+00  2.010000e+03       1.400000e+03    0.000000e+00
25%       8.000000e+01  2.012000e+03       4.357373e+06    1.543300e+04
50%       9.100000e+01  2.014000e+03       1.300000e+07    1.271149e+07
75%       1.050000e+02  2.017000e+03       5.100000e+07    5.825080e+07
max       5.142000e+04  2.022000e+03       4.250000e+08    7.605076e+08

       worldwide_gross  averagerating      numvotes     popularity
count     3.238380e+05   1.396436e+06  1.396436e+06  860916.000000
mean      1.247131e+08   6.214570e+00  1.478212e+04       5.114811
std       2.117233e+08   1.351983e+00  6.625076e+04       6.434947
min       0.000000e+00   1.000000e+00  5.000000e+00       0.600000
25%       2.483790e+06   5.400000e+00  3.300000e+01       0.701000
50%       4.479317e+07   6.400000e+00  2.040000e+02       2.379000
75%       1.310118e+08   7.100000e+00  1.502000e+03       7.620000
max       2.776345e+09   1.000000e+01  1.841066e+06      80.773000
```

53.1 Guiding Questions

1. How many titles are there
2. Original_title
3. Studio
4. Genre
5. length
6. region
7. original_language

53.2 1. How many titles are there

[115]:
```
print(df.columns)

# titles in DF
# len(df.title) # 1830683

# unique titles
df.title.nunique() #107459

# no of repeated titles
# len(df.title) - df.title.nunique()
```

```
Index(['title', 'is_original_title', 'studio', 'runtime_minutes', 'genres',
       'start_year', 'original_language', 'region', 'directors', 'writers',
       'production_budget', 'domestic_gross', 'worldwide_gross',
       'averagerating', 'numvotes', 'popularity'],
      dtype='object')
```

[115]:
```
107459
```

53.3 2. Original_title

[134]:
```
# number of original titles
original_ser = df.groupby('is_original_title').is_original_title.count()#.
 ↪to_frame()
Orig_Titles = pd.DataFrame(original_ser).rename(columns = {'is_original_title':␣
 ↪'Count'})
x = list(Orig_Titles.values)
x

Orig_Titles.values
```

[134]:
```
array([[1407172],
       [  74571],
       [ 348940]], dtype=int64)
```

[103]:
```
# df.groupby(['title', 'production_budget', 'domestic_gross']).value_counts().
 ↪to_frame()[:2]
df[['title','is_original_title','production_budget', 'domestic_gross',␣
 ↪'worldwide_gross']].groupby('title').count().
 ↪sort_values(by=['production_budget', 'domestic_gross'], ascending = False)[:
 ↪15]
```

[103]:
```
               is_original_title  production_budget  domestic_gross  \
title
Home                       76860              76860           76860
The Gift                   11560              11560           11560
Eden                        9576               9576            9576
Robin Hood                  7488               7488            7488
Truth or Dare               4130               4130            4130
Brothers                    3200               3200            3200
Legend                      2790               2790            2790
Split                       2688               2688            2688
Redemption                  2640               2640            2640
The Return                  2332               2332            2332
Life                        2016               2016            2016
The Family                  1932               1932            1932
Trapped                     1830               1830            1830
Silence                     1800               1800            1800
The Promise                 1764               1764            1764

               worldwide_gross
title
Home                     76860
The Gift                 11560
Eden                      9576
Robin Hood                7488
Truth or Dare             4130
Brothers                  3200
Legend                    2790
Split                     2688
Redemption                2640
The Return                2332
Life                      2016
The Family                1932
Trapped                   1830
Silence                   1800
The Promise               1764
```

53.4 3. Studios

[104]:
```
# number of unique studios
df.studio.nunique()

# list of unique studios
df.studio.unique()
```

[104]:
```
array(['Sony', nan, 'Magn.', 'IFC', 'Par.', 'LGF', 'Gold.', 'SPC', 'WB',
       'Fox', 'Strand', 'FM', 'Men.', 'WHE', 'FoxS', 'MBox', 'ParV',
       'LGP', 'Wein.', 'Focus', 'BV', 'SD', 'Uni.', 'EC', 'Rela.',
       'ENTMP', 'A24', 'Zeit.', 'FRun', 'KL', 'Imax', 'Affirm', 'BST',
       'Anch.', 'MR', 'PDA', 'CGld', 'Osci.', 'FR', 'CFI', 'LG/S',
       'Aviron', 'STX', 'Rel.', 'Eros', 'CJ', 'MNE', 'XL', 'PFR', 'Imag.',
       'KE', 'RTWC', 'WB (NL)', 'Annapurna', 'Orch.', 'Da.', 'RAtt.',
       'ORF', 'Distrib.', 'Over.', 'Relbig.', 'Blue Fox', 'NYer', 'Arrow',
       'P/DW', 'Vari.', 'HC', 'VPD', 'Ghop', 'SGem', 'Amazon', 'P4', 'VE',
       'MPFT', 'BG', 'Sum.', 'W/Dim.', 'EOne', 'Cdgm.', 'Cohen', 'SHO',
       'Free', 'Trib.', 'UTV', 'FOAK', 'FD', 'Arth.', 'TriS', 'Abr.',
       'GK', 'FIP', 'TA', 'Global Road', 'HTR', 'FUN', 'WGUSA', 'Rocket',
       'CL', 'UEP', 'PNT', 'Fathom', 'BH Tilt', 'Grindstone', 'FCW',
       'Jan.', 'LD', 'KS', 'Drft.', 'Scre.', 'Mira.', 'Ampl.', 'ALP',
       'Grav.', 'PI', 'FInd.', 'BM&DH', 'NGE', 'Rialto', 'FOR',
       'CineGalaxy', 'Elev.', 'SEG', 'BBC', 'JBG', 'AF', 'MPI', 'CBS',
       'IM', 'Lorb.', 'CF&SR', 'DF', 'Greenwich', 'MUBI', 'FEF', 'Saban',
       'First', 'CE', 'Mont.', 'TAFC', 'P/108', 'Kino', 'Studio 8', 'ITL',
       'ADC', 'TFA', 'SM', 'PH', 'OutF', 'CLS', 'Asp.', 'Alc', 'AGF',
       'OMNI/FSR', 'Yash', 'A23', 'Crimson', 'ATO', 'RF', 'SMod', 'CAVU',
       'UTMW', 'NAV', 'ELS', 'Vita.', 'Good Deed', 'Triu', 'U/P', 'SV',
       'AM', 'App.', 'WOW', 'TVC', 'Neon', 'Viv.', 'PackYourBag', 'KC',
       'Trafalgar', 'Dreamwest', 'Crnth', 'NM', 'MGM', 'BSC', 'Shout!',
       'Electric', 'SDS', '3D', 'EXCL', 'CLF', 'Icar.', 'Rog.', 'Zee',
       'AZ', 'Cleopatra', 'Gaatri', 'WAMCR', 'KKM', 'AR', 'Abk.', 'RLJ',
       'BWP', 'PM&E', 'Outs', 'Linn', 'Super', 'Hann.', 'DR', 'Orion',
       'RME', 'EF', 'BGP', 'Pala.', 'MOM', 'B360', 'ICir', 'EpicPics',
       'GrtIndia', 'Proud'], dtype=object)
```

[105]:
```
# # df.groupby(['title', 'production_budget', 'domestic_gross']).value_counts().
 ↪to_frame()[:2]
df[['title','studio', 'region','production_budget', 'domestic_gross',␣
 ↪'worldwide_gross']].groupby('title').count(
).sort_values(by=['production_budget', 'domestic_gross'], ascending = False)[:
 ↪15]
```

[105]:
```
                  studio  region  production_budget  domestic_gross  \
title
Home                   0   61488              76860           76860
```

```
The Gift            0    8160              11560           11560
Eden             9576    7980               9576            9576
Robin Hood       7488    6786               7488            7488
Truth or Dare    4130    2950               4130            4130
Brothers            0    2800               3200            3200
Legend           2790    1860               2790            2790
Split            2688    2016               2688            2688
Redemption          0    2145               2640            2640
The Return          0    2014               2332            2332
Life                0    1584               2016            2016
The Family          0    1518               1932            1932
Trapped             0    1586               1830            1830
Silence             0    1560               1800            1800
The Promise         0    1470               1764            1764

               worldwide_gross
title
Home                     76860
The Gift                 11560
Eden                      9576
Robin Hood                7488
Truth or Dare             4130
Brothers                  3200
Legend                    2790
Split                     2688
Redemption                2640
The Return                2332
Life                      2016
The Family                1932
Trapped                   1830
Silence                   1800
The Promise               1764
```

53.5 4. Genre

[106]:

```python
print(df.genres.unique())

# Best performing genres by gross
df.groupby(['genres', 'domestic_gross', 'worldwide_gross']
          ).genres.value_counts().to_frame().sort_values(
        by=['domestic_gross'], ascending = False)[:10]
```

```
['Adventure,Animation,Comedy' 'Comedy,Romance' 'Action,Adventure,Family'
 … 'Biography,History,Musical' 'Adventure,Musical,Sci-Fi'
 'Adult,Romance']
```

[106]:
```
                                                               count
genres                    domestic_gross worldwide_gross
Horror                    760507625.0    2.776345e+09            10
Action,Adventure,Sci-Fi   700059566.0    1.348258e+09           260
                          678815482.0    2.048134e+09           130
Family                    659363944.0    2.208208e+09            10
Action,Adventure,Sci-Fi   652270625.0    1.648855e+09           140
                          623279547.0    1.517936e+09            50
Action,Adventure,Animation 608581744.0   1.242521e+09            20
Action,Adventure,Sci-Fi   532177324.0    1.049103e+09           180
Family,Fantasy,Musical    504014165.0    1.259200e+09           180
Drama,Fantasy,Romance     504014165.0    1.259200e+09            90
```

53.6 5. length

[107]:
```
df.runtime_minutes.to_frame().sort_values(by=['runtime_minutes'], ascending =␣
 ↪False)
```

[107]:
```
         runtime_minutes
2147569            51420
2147574            51420
2147570            51420
2147571            51420
2147572            51420
…                    …
1514194                1
1482434                1
1482433                1
1482432                1
834038                 1

[1830683 rows x 1 columns]
```

[108]:
```
# Best performing genres by gross
df.groupby(['runtime_minutes', 'domestic_gross', 'worldwide_gross']).genres.
 ↪value_counts(
).to_frame().sort_values(by=['domestic_gross'], ascending = False)[:10]
```

[108]:
```
                                                                        count
runtime_minutes domestic_gross worldwide_gross genres
93              760507625.0    2.776345e+09    Horror                      10
134             700059566.0    1.348258e+09    Action,Adventure,Sci-Fi    260
149             678815482.0    2.048134e+09    Action,Adventure,Sci-Fi    130
115             659363944.0    2.208208e+09    Family                      10
124             652270625.0    1.648855e+09    Action,Adventure,Sci-Fi    140
143             623279547.0    1.517936e+09    Action,Adventure,Sci-Fi     50
118             608581744.0    1.242521e+09    Action,Adventure,Animation  20
```

```
133              532177324.0    1.049103e+09    Action,Adventure,Sci-Fi       180
112              504014165.0    1.259200e+09    Drama,Fantasy,Romance          90
60               504014165.0    1.259200e+09    Family,Fantasy,Musical         90
```

[109]:
```
top_10_avg_runtime_df = df.groupby(['runtime_minutes', 'title',␣
 ↪'domestic_gross', 'worldwide_gross']).count().
 ↪sort_values(by='worldwide_gross', ascending = False)

top_10_avg_runtime_df = top_10_avg_runtime_df.reset_index()
top_10_avg_runtime_df.runtime_minutes.mean()
```

[109]:
```
97.30133657351155
```

53.7 6. Region

[110]:
```
# How many unique region do we have?
print(df.region.nunique())

# Which are the unique regions?
df.region.unique()
```

```
207
```

[110]:
```
array(['US', 'CA', nan, 'FR', 'AU', 'PH', 'CN', 'DE', 'RU', 'XWW', 'SE',
       'MZ', 'BE', 'TR', 'IT', 'PT', 'PL', 'EG', 'ZA', 'AT', 'SI', 'AR',
       'KW', 'HR', 'IL', 'IE', 'XEU', 'BR', 'TW', 'GB', 'ES', 'IR', 'GT',
       'XKV', 'VE', 'GR', 'CL', 'NO', 'EC', 'MN', 'IN', 'KR', 'TH', 'UY',
       'HU', 'NZ', 'HK', 'MX', 'LB', 'FI', 'NL', 'DK', 'BG', 'LT', 'RO',
       'JP', 'AE', 'CZ', 'UA', 'LU', 'ID', 'LK', 'CH', 'HN', 'AM', 'AO',
       'MK', 'NG', 'CO', 'IS', 'BO', 'PE', 'MO', 'RS', 'MY', 'SG', 'PR',
       'GE', 'JM', 'ME', 'CU', 'PF', 'LV', 'PS', 'CD', 'IQ', 'JO', 'DO',
       'QA', 'SK', 'DZ', 'AZ', 'KE', 'EE', 'KZ', 'BD', 'KY', 'BA', 'PK',
       'CM', 'BY', 'GH', 'MD', 'UG', 'MA', 'MT', 'MC', 'VN', 'TT', 'XKO',
       'SN', 'BH', 'CI', 'ET', 'SV', 'GP', 'AL', 'KH', 'NI', 'SY', 'TZ',
       'GL', 'PA', 'CR', 'AF', 'MM', 'NP', 'BF', 'CY', 'BS', 'SL', 'AD',
       'VC', 'RW', 'ZM', 'BT', 'TN', 'MW', 'MU', 'FO', 'HT', 'PY', 'ML',
       'LS', 'KP', 'TJ', 'DM', 'XAS', 'BB', 'TD', 'TL', 'PG', 'AN', 'YE',
       'UZ', 'GU', 'AQ', 'XNA', 'SA', 'SO', 'SZ', 'VI', 'KG', 'BJ', 'SD',
       'CSXX', 'NE', 'GW', 'LI', 'CG', 'GA', 'SM', 'ER', 'MR', 'WF', 'BN',
       'BZ', 'LA', 'FJ', 'IM', 'AG', 'ZW', 'VU', 'BM', 'LR', 'AW', 'TO',
       'CV', 'MQ', 'RE', 'MG', 'KN', 'MV', 'TG', 'GM', 'NC', 'OM', 'BI',
       'AS', 'MH', 'SR', 'AI', 'SB', 'BUMM', 'CF', 'LY', 'EH', 'LC'],
      dtype=object)
```

[111]:
```
df[['region', 'title', 'studio', 'genres','production_budget', 'domestic_gross',
    'worldwide_gross']].groupby('region').count().
 ↪sort_values(by=['worldwide_gross',
    'worldwide_gross'], ascending = False)[:15]
```

[111]:

```
         title  studio  genres  production_budget  domestic_gross  \
region
US      455048   39281  452335              73402           73402
XWW     195983   14549  194331              41382           41382
GB       82582    8171   81879              16238           16238
CA       58143    7962   57726               9394            9394
FR       48179    9118   47219               8941            8941
DE       42968    5874   42563               8746            8746
ES       29611    7596   29199               6784            6784
AU       25377    2235   25223               6544            6544
IL        9815    3131    9774               5783            5783
IT       30810    5028   30572               5095            5095
IN       59511    3834   59277               4807            4807
GR       11842    4878   11715               4377            4377
BE        9876    1007    9783               3967            3967
NL       12044     753   11985               3944            3944
PL       11311    3608   11276               3621            3621

        worldwide_gross
region
US                73402
XWW               41382
GB                16238
CA                 9394
FR                 8941
DE                 8746
ES                 6784
AU                 6544
IL                 5783
IT                 5095
IN                 4807
GR                 4377
BE                 3967
NL                 3944
PL                 3621
```

53.8 7. Original language

[112]:

```
# how many unique languages in the dataset
print(df.original_language.nunique())

# what are the unique languages in the dataset
df.original_language.unique()
```

```
71
```

```
[112]: array(['en', nan, 'lo', 'de', 'ru', 'he', 'fr', 'es', 'sv', 'it', 'hi',
              'pl', 'id', 'cn', 'zh', 'uk', 'nl', 'tl', 'fa', 'ko', 'ja', 'no',
              'el', 'mr', 'hr', 'te', 'pt', 'hu', 'tr', 'vi', 'cs', 'da', 'xx',
              'ar', 'sr', 'ca', 'is', 'ta', 'ro', 'sq', 'eu', 'ml', 'fi', 'th',
              'kn', 'dz', 'lv', 'gu', 'ur', 'ab', 'mi', 'ka', 'et', 'bg', 'kk',
              'ku', 'lt', 'cy', 'bn', 'bo', 'pa', 'hy', 'sn', 'sw', 'hz', 'yi',
              'ky', 'ne', 'xh', 'af', 'cr', 'ha'], dtype=object)
```

```
[113]: df[['title', 'studio', 'region', 'genres', 'original_language',␣
        ↪'production_budget', 'domestic_gross',
          'worldwide_gross']].groupby('original_language').count().
        ↪sort_values(by=['worldwide_gross',
          'worldwide_gross'], ascending = False)[:15]
```

```
[113]:                      title  studio  region  genres  production_budget  \
       original_language
       en                  732740  169861  548405  728799             275854
       ru                   20387    1591   16586   20174              13064
       es                   15118    2609   12500   14895               3144
       fr                   23003    5762   19948   22896               3012
       zh                    7838    1949    6846    7773               1664
       de                    9604    2818    8049    9513               1342
       ar                    1410    1004    1272    1410                864
       sv                    2938     762    2525    2938                846
       hi                    8362    1772    5629    8292                832
       no                    2652    1596    2324    2652                525
       pt                    2605    1102    2098    2605                474
       th                     769     440     563     769                360
       el                     923     581     824     923                341
       it                    2332     774    1912    2332                334
       he                    1733     784    1369    1733                294

                           domestic_gross  worldwide_gross
       original_language
       en                          275854           275854
       ru                           13064            13064
       es                            3144             3144
       fr                            3012             3012
       zh                            1664             1664
       de                            1342             1342
       ar                             864              864
       sv                             846              846
       hi                             832              832
       no                             525              525
       pt                             474              474
       th                             360              360
       el                             341              341
```

```
it                            334              334
he                            294              294
```