Data_Cleaning

February 13, 2025

0.1 Exploring the data

Since there is so much data, we need to figure out what the data is and how, if we want to, to combine all the data. We also need to check if anything needs to be cleaned.

[22]:
```
import pandas as pd
import numpy as np
import sqlite3
```

[23]:
```
df_gross = pd.read_csv('../data/bom.movie_gross.csv', index_col=0)
df_budgets = pd.read_csv('../data/tn.movie_budgets.csv', index_col=0)
df_movies = pd.read_csv('../data/tmdb.movies.csv', index_col=0)
df_reviews = pd.read_csv('../data/rt.reviews.tsv', index_col=0, sep='\t',␣
↪encoding='latin-1')
df_info = pd.read_csv('../data/rt.movie_info.tsv', index_col=0, sep='\t')
conn = sqlite3.connect('../data/im.db')
```

This dataset holds the amount of money a movie made domestically and foreignly, the studio associated with the movie and the year it came out.

[24]:
```
#df_gross['domestic_gross'] = df_gross['domestic_gross'].map(lambda x: int(x)␣
↪if pd.notnull(x) else x) Not working??
df_gross
```

[24]:
```
                                             studio  domestic_gross  \
title
Toy Story 3                                      BV     415000000.0
Alice in Wonderland (2010)                       BV     334200000.0
Harry Potter and the Deathly Hallows Part 1      WB     296000000.0
Inception                                        WB     292600000.0
Shrek Forever After                            P/DW     238700000.0
…                                                 …               …
The Quake                                     Magn.          6200.0
Edward II (2018 re-release)                      FM          4800.0
El Pacto                                       Sony          2500.0
The Swan                                 Synergetic          2400.0
An Actor Prepares                             Grav.          1700.0

                                           foreign_gross  year
```

```
title
Toy Story 3                                     652000000  2010
Alice in Wonderland (2010)                      691300000  2010
Harry Potter and the Deathly Hallows Part 1     664300000  2010
Inception                                       535700000  2010
Shrek Forever After                             513900000  2010
…                                                     …     …
The Quake                                             NaN  2018
Edward II (2018 re-release)                           NaN  2018
El Pacto                                              NaN  2018
The Swan                                              NaN  2018
An Actor Prepares                                     NaN  2018

[3387 rows x 4 columns]
```

This dataset is somewhat similar to the one above, except it has a more specific release date, and also the budget spent on the movie. Upon checking the gross columns, there are no missing boxes, so we can strip the columns and turn them into ints to make them easier to deal with. We will also drop movies before the 2000s as inflation will make the price comparisons not equal.

[25]:

```
#df_budgets[['production_budget', 'domestic_gross', 'wordlwide_gross']] =
 ↪df_budgets[['production_budget', 'domestic_gross', 'worldwide_gross']].
 ↪apply(lambda x: x.replace('$', '').replace(',', ''))
#Remove all symbols and convert to int
df_budgets['production_budget'] = df_budgets['production_budget'].map(lambda x:
 ↪int(x.replace('$', '').replace(',', '')))
df_budgets['domestic_gross'] = df_budgets['domestic_gross'].map(lambda x: int(x.
 ↪replace('$', '').replace(',', '')))
df_budgets['worldwide_gross'] = df_budgets['worldwide_gross'].map(lambda x:
 ↪int(x.replace('$', '').replace(',', '')))
#convert to datetime, remove everything before 2000s
df_budgets['release_date'] = pd.to_datetime(df_budgets['release_date'])
df_budgets = df_budgets[df_budgets['release_date'] > pd.
 ↪to_datetime('1999-12-31')]
df_budgets
```

[25]:

```
   release_date                                     movie  \
id
1    2009-12-18                                    Avatar
2    2011-05-20  Pirates of the Caribbean: On Stranger Tides
3    2019-06-07                              Dark Phoenix
4    2015-05-01                   Avengers: Age of Ultron
5    2017-12-15             Star Wars Ep. VIII: The Last Jedi
..          …                                         …
77   2004-12-31                           The Mongol King
78   2018-12-31                                    Red 11
80   2005-07-13               Return to the Land of Wonders
```

```
81   2015-09-29                             A Plague So Pleasant
82   2005-08-05                                My Date With Drew

    production_budget  domestic_gross  worldwide_gross
id
1           425000000       760507625       2776345279
2           410600000       241063875       1045663875
3           350000000        42762350        149762350
4           330600000       459005868       1403013963
5           317000000       620181382       1316721747
..                ...             ...              ...
77               7000             900              900
78               7000               0                0
80               5000            1338             1338
81               1400               0                0
82               1100          181041           181041

[4387 rows x 5 columns]
```

What we can do with this dataset is create a value that measures how much the movie made compared to its production budget to get a simplified value of return.

[26]:
```
df_budgets['return_ratio'] = ((df_budgets['worldwide_gross'] -
 ↪df_budgets['production_budget']) / df_budgets['production_budget'] ).round(2)
df_budgets.sort_values('return_ratio', ascending=False)
```

[26]:
```
   release_date                movie  production_budget  domestic_gross  \
id
93   2009-09-25  Paranormal Activity             450000       107918810
80   2015-07-10          The Gallows             100000        22764410
10   2004-05-07         Super Size Me              65000        11529368
82   2005-08-05     My Date With Drew               1100          181041
57   2007-05-16                  Once             150000         9445857
..          ...                  ...                ...             ...
4    2011-12-31              Tracker            6500000               0
81   2016-10-16           Mi America            2100000            3330
83   2012-12-31             Infected            2100000               0
52   2015-12-11    The Ridiculous 6           60000000               0
14   2015-02-13           Girlhouse            1500000               0

    worldwide_gross  return_ratio
id
93        194183034        430.52
80         41656474        415.56
10         22233808        341.06
82           181041        163.58
57         23323631        154.49
..              ...           ...
```

```
4               3149         -1.00
81              3330         -1.00
83                 0         -1.00
52                 0         -1.00
14                 0         -1.00

[4387 rows x 6 columns]
```

This dataset has an interesting column genre_ids, which holds arrays of numbers. These numbers presumably can be associated with a dict of some sort that holds what genre it is from the number. It also has a popularity number which isn't obvious what it is, the vote_average, which is presumably out of 10, and the vote count. We will drop the genre_ids since there doesn't seem to be another table that links it to the actual genres.

[27]:
```
df_movies.drop(columns='genre_ids', inplace=True)
df_movies.set_index('id', inplace=True)

df_movies['release_date'] = pd.to_datetime(df_movies['release_date'])
df_movies = df_movies[df_movies['release_date'] > pd.to_datetime('1999-12-31')]
df_movies
```

[27]:
```
       original_language                                    original_title  \
id
12444                 en       Harry Potter and the Deathly Hallows: Part 1
10191                 en                           How to Train Your Dragon
10138                 en                                         Iron Man 2
27205                 en                                          Inception
32657                 en  Percy Jackson & the Olympians: The Lightning T…
…                   …                                                 …
488143                en                              Laboratory Conditions
485975                en                                    _EXHIBIT_84xxx_
381231                en                                       The Last One
366854                en                                       Trailer Made
309885                en                                         The Church

        popularity release_date  \
id
12444       33.533   2010-11-19
10191       28.734   2010-03-26
10138       28.515   2010-05-07
27205       27.920   2010-07-16
32657       26.691   2010-02-11
…             …           …
488143       0.600   2018-10-13
485975       0.600   2018-05-01
381231       0.600   2018-10-01
366854       0.600   2018-06-22
309885       0.600   2018-10-05
```

```
                                                     title  vote_average  \
id
12444          Harry Potter and the Deathly Hallows: Part 1           7.7
10191                                 How to Train Your Dragon           7.7
10138                                               Iron Man 2           6.8
27205                                                Inception           8.3
32657   Percy Jackson & the Olympians: The Lightning T…          6.1
…                                                      …            …
488143                                 Laboratory Conditions           0.0
485975                                       _EXHIBIT_84xxx_           0.0
381231                                          The Last One           0.0
366854                                          Trailer Made           0.0
309885                                            The Church           0.0

        vote_count
id
12444        10788
10191         7610
10138        12368
27205        22186
32657         4229
…             …
488143           1
485975           1
381231           1
366854           1
309885           1

[26398 rows x 7 columns]
```

This dataset holds a large amount of reviews with the text, and the score associated with it. This one seems to have many missing rating numbers, some not even being numbers. The fresh section shows this came from rotten tomatoes.

[28]:
```
df_reviews
```

[28]:
```
                                                 review rating   fresh  \
id
3     A distinctly gallows take on contemporary fina…    3/5   fresh
3     It's an allegory in search of a meaning that n…    NaN  rotten
3     … life lived in a bubble in financial dealin…    NaN   fresh
3     Continuing along a line introduced in last yea…    NaN   fresh
3                …  a perverse twist on neorealism…    NaN   fresh
…                                                   …    …     …
2000  The real charm of this trifle is the deadpan c…    NaN   fresh
2000                                                NaN    1/5  rotten
2000                                                NaN    2/5  rotten
```

```
2000                                                      NaN   2.5/5  rotten
2000                                                      NaN     3/5   fresh

                  critic  top_critic          publisher                date
id
3             PJ Nabarro           0   Patrick Nabarro   November 10, 2018
3         Annalee Newitz           0          io9.com         May 23, 2018
3           Sean Axmaker           0  Stream on Demand     January 4, 2018
3          Daniel Kasman           0              MUBI   November 16, 2017
3                    NaN           0      Cinema Scope    October 12, 2017
…                    …          …                 …                  …
2000       Laura Sinagra           1     Village Voice  September 24, 2002
2000   Michael Szymanski           0        Zap2it.com  September 21, 2005
2000         Emanuel Levy          0    EmanuelLevy.Com       July 17, 2005
2000     Christopher Null          0    Filmcritic.com   September 7, 2003
2000      Nicolas Lacroix          0      Showbizz.net   November 12, 2002

[54432 rows x 7 columns]
```

This dataset has a lot of info in it. The genre, director, writer, dates, runtime, studio, to name a few which might not be found in the other datasets.

[29]:
```
df_info['box_office'] = df_info['box_office'].map(lambda x: int(str(x).
 ↪replace(',', '')) if pd.notnull(x) else x)
df_info
```

[29]:
```
                                               synopsis rating  \
id
1     This gritty, fast-paced, and innovative police…      R
3     New York City, not-too-distant-future: Eric Pa…      R
5     Illeana Douglas delivers a superb performance …      R
6     Michael Douglas runs afoul of a treacherous su…      R
7                                                   NaN     NR
…                                                   …    …
1996  Forget terrorists or hijackers -- there's a ha…      R
1997  The popular Saturday Night Live sketch was exp…     PG
1998  Based on a novel by Richard Powell, when the l…      G
1999  The Sandlot is a coming-of-age story about a g…     PG
2000  Suspended from the force, Paris cop Hubert is …      R

                                               genre          director  \
id
1                Action and Adventure|Classics|Drama   William Friedkin
3                  Drama|Science Fiction and Fantasy   David Cronenberg
5                  Drama|Musical and Performing Arts     Allison Anders
6                         Drama|Mystery and Suspense     Barry Levinson
7                                      Drama|Romance     Rodney Bennett
…                                                   …                …
```

```
1996   Action and Adventure|Horror|Mystery and Suspense                 NaN
1997                Comedy|Science Fiction and Fantasy        Steve Barron
1998  Classics|Comedy|Drama|Musical and Performing Arts      Gordon Douglas
1999    Comedy|Drama|Kids and Family|Sports and Fitness  David Mickey Evans
2000  Action and Adventure|Art House and Internation…                 NaN

                                                 writer  theater_date  \
id
1                                       Ernest Tidyman   Oct 9, 1971
3                          David Cronenberg|Don DeLillo  Aug 17, 2012
5                                       Allison Anders  Sep 13, 1996
6                       Paul Attanasio|Michael Crichton   Dec 9, 1994
7                                         Giles Cooper           NaN
…                                                   …            …
1996                                               NaN  Aug 18, 2006
1997  Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner  Jul 23, 1993
1998                                               NaN   Jan 1, 1962
1999                    David Mickey Evans|Robert Gunter   Apr 1, 1993
2000                                        Luc Besson  Sep 27, 2001

          dvd_date currency  box_office      runtime             studio
id
1     Sep 25, 2001      NaN         NaN  104 minutes                NaN
3      Jan 1, 2013        $    600000.0  108 minutes  Entertainment One
5     Apr 18, 2000      NaN         NaN  116 minutes                NaN
6     Aug 27, 1997      NaN         NaN  128 minutes                NaN
7              NaN      NaN         NaN  200 minutes                NaN
…              …        …          …           …                …
1996   Jan 2, 2007        $  33886034.0  106 minutes    New Line Cinema
1997  Apr 17, 2001      NaN         NaN   88 minutes  Paramount Vantage
1998  May 11, 2004      NaN         NaN  111 minutes                NaN
1999  Jan 29, 2002      NaN         NaN  101 minutes                NaN
2000  Feb 11, 2003      NaN         NaN   94 minutes  Columbia Pictures

[1560 rows x 11 columns]
```

The rotten tomato datasets may be one we decide to not use. Not only are the columns missing a lot of values, neither datasets have titles that we can tie the data to. Another option would be to instead just atribute the data to studios, but even the studio data is pretty sparse and missing a lot of data.

We need to figure out the structure of this database first. There are quite a few tables and the key that connects them is unknown. There might be data that isn't shown in the other datasets, such as principals, movie_akas, and known_for. So for now, we will explore those since the other tables likely have redundant data.

[30]:
```
pd.read_sql(
"""
```

```
SELECT name
FROM sqlite_master
WHERE type='table'
""", conn)
```

[30]:

```
              name
0    movie_basics
1       directors
2       known_for
3      movie_akas
4   movie_ratings
5         persons
6      principals
7         writers
```

It seems like in these databases, people are not referred to by name, instead by an id that will link them to another table. In any case, this table shows the people who worked on a movie, their roles, and potentially the characters name they played.

[31]:

```
pd.read_sql(
"""
SELECT *
FROM principals
""", conn)
```

[31]:

```
          movie_id  ordering   person_id  category       job  \
0        tt0111414         1   nm0246005     actor      None
1        tt0111414         2   nm0398271  director      None
2        tt0111414         3   nm3739909  producer  producer
3        tt0323808        10   nm0059247    editor      None
4        tt0323808         1   nm3579312   actress      None
…              …        …          …         …       …
1028181  tt9692684         1   nm0186469     actor      None
1028182  tt9692684         2   nm4929530      self      None
1028183  tt9692684         3  nm10441594  director      None
1028184  tt9692684         4   nm6009913    writer    writer
1028185  tt9692684         5  nm10441595  producer  producer

                     characters
0                   ["The Man"]
1                          None
2                          None
3                          None
4              ["Beth Boothby"]
…                           …
1028181  ["Ebenezer Scrooge"]
1028182    ["Herself","Regan"]
1028183                    None
```

```
1028184                       None
1028185                       None

[1028186 rows x 6 columns]
```

This table seems to be for movies that have different titles since they're in a different language.

[32]:
```
pd.read_sql(
"""
SELECT *
FROM movie_akas
""", conn)
```

[32]:
```
         movie_id  ordering                                    title region  \
0       tt0369610        10                                              BG
1       tt0369610        11                        Jurashikku warudo     JP
2       tt0369610        12  Jurassic World: O Mundo dos Dinossauros     BR
3       tt0369610        13                  O Mundo dos Dinossauros     BR
4       tt0369610        14                           Jurassic World     FR
…             …         …                                         …      …
331698  tt9827784         2                       Sayonara kuchibiru   None
331699  tt9827784         3                            Farewell Song    XWW
331700  tt9880178         1                              La atención   None
331701  tt9880178         2                              La atención     ES
331702  tt9880178         3                            The Attention    XWW

       language        types   attributes  is_original_title
0            bg         None         None                0.0
1          None  imdbDisplay         None                0.0
2          None  imdbDisplay         None                0.0
3          None         None  short title                0.0
4          None  imdbDisplay         None                0.0
…             …            …            …                  …
331698     None     original         None                1.0
331699       en  imdbDisplay         None                0.0
331700     None     original         None                1.0
331701     None         None         None                0.0
331702       en  imdbDisplay         None                0.0

[331703 rows x 8 columns]
```

This table relates a person to the movies they worked on.

[33]:
```
pd.read_sql(
"""
SELECT *
FROM known_for
""",conn)
```

[33]:
```
         person_id   movie_id
0        nm0061671  tt0837562
1        nm0061671  tt2398241
2        nm0061671  tt0844471
3        nm0061671  tt0118553
4        nm0061865  tt0896534
…              …          …
1638255  nm9990690  tt9090932
1638256  nm9990690  tt8737130
1638257  nm9991320  tt8734436
1638258  nm9991320  tt9615610
1638259  nm9993380  tt8743182

[1638260 rows x 2 columns]
```

1 Printing out the rest of the data

[34]:
```python
pd.read_sql(
"""
SELECT *
FROM movie_basics
""",conn)
```

[34]:
```
         movie_id                                 primary_title  \
0       tt0063540                                     Sunghursh
1       tt0066787               One Day Before the Rainy Season
2       tt0069049                    The Other Side of the Wind
3       tt0069204                               Sabse Bada Sukh
4       tt0100275                      The Wandering Soap Opera
…             …                                             …
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  \
0                                          Sunghursh        2013
1                                    Ashad Ka Ek Din        2019
2                         The Other Side of the Wind        2018
3                                    Sabse Bada Sukh        2018
4                              La Telenovela Errante        2017
…                                                  …           …
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
0                 175.0    Action,Crime,Drama
1                 114.0       Biography,Drama
2                 122.0                 Drama
3                   NaN          Comedy,Drama
4                  80.0  Comedy,Drama,Fantasy
…                   …                    …
146139            123.0                 Drama
146140              NaN           Documentary
146141              NaN                Comedy
146142            116.0                  None
146143              NaN           Documentary

[146144 rows x 6 columns]
```

[35]:

```
pd.read_sql(
"""
SELECT *
FROM directors
""",conn)
```

[35]:

```
         movie_id   person_id
0       tt0285252   nm0899854
1       tt0462036   nm1940585
2       tt0835418   nm0151540
3       tt0835418   nm0151540
4       tt0878654   nm0089502
…             …          …
291169  tt8999974  nm10122357
291170  tt9001390   nm6711477
291171  tt9001494  nm10123242
291172  tt9001494  nm10123248
291173  tt9004986   nm4993825

[291174 rows x 2 columns]
```

[36]:

```
pd.read_sql(
"""
SELECT *
FROM persons
""",conn)
```

[36]:

```
        person_id         primary_name  birth_year  death_year  \
0       nm0061671    Mary Ellen Bauder         NaN         NaN
```

```
1       nm0061865        Joseph Bauer        NaN          NaN
2       nm0062070          Bruce Baum        NaN          NaN
3       nm0062195        Axel Baumann        NaN          NaN
4       nm0062798         Pete Baxter        NaN          NaN
…              …                   …         …            …
606643  nm9990381        Susan Grobes        NaN          NaN
606644  nm9990690          Joo Yeon So       NaN          NaN
606645  nm9991320      Madeline Smith        NaN          NaN
606646  nm9991786  Michelle Modigliani       NaN          NaN
606647  nm9993380       Pegasus Envoyé       NaN          NaN

                                   primary_profession
0             miscellaneous,production_manager,producer
1            composer,music_department,sound_department
2                            miscellaneous,actor,writer
3       camera_department,cinematographer,art_department
4       production_designer,art_department,set_decorator
…                                                    …
606643                                         actress
606644                                         actress
606645                                         actress
606646                                        producer
606647                             director,actor,writer

[606648 rows x 5 columns]
```

[37]:
```python
pd.read_sql(
"""
SELECT *
FROM writers
""",conn)
```

[37]:
```
         movie_id   person_id
0       tt0285252   nm0899854
1       tt0438973   nm0175726
2       tt0438973   nm1802864
3       tt0462036   nm1940585
4       tt0835418   nm0310087
…             …          …
255868  tt8999892  nm10122246
255869  tt8999974  nm10122357
255870  tt9001390   nm6711477
255871  tt9004986   nm4993825
255872  tt9010172   nm8352242

[255873 rows x 2 columns]
```

[38]:
```
pd.read_sql(
"""
SELECT *
FROM movie_ratings
""",conn)
```

[38]:
```
        movie_id  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
…             …            …        …
73851   tt9805820            8.1        25
73852   tt9844256            7.5        24
73853   tt9851050            4.7        14
73854   tt9886934            7.0         5
73855   tt9894098            6.3       128

[73856 rows x 3 columns]
```

With some basic cleaning done, and a little quick analysis, we can begin doing our own separate analysis and creating business recommendations

[39]:
```
df_gross.to_csv('../data/cleaned_movie_gross.csv', encoding='utf-8')
df_budgets.to_csv('../data/cleaned_budgets.csv', encoding='utf-8')
df_movies.to_csv('../data/cleaned_movies.csv', encoding='utf-8')
df_info.to_csv('../data/cleaned_rt_info.csv', encoding='utf-8')
```