In [16]:

```
import sys
!{sys.executable} -m pip install -r requirements.txt
```

```
Collecting ipython-sql
  Downloading ipython_sql-0.4.0-py3-none-any.whl (19 kB)
Requirement already satisfied: six in ./venv/lib/python3.8/site-packages (from ipython-sql) (1.15.0)
Collecting sqlparse
  Downloading sqlparse-0.4.1-py3-none-any.whl (42 kB)
     |████████████████████████████████| 42 kB 3.7 MB/s  eta 0:00:01
Requirement already satisfied: ipython>=1.0 in ./venv/lib/python3.8/site-packages (from ipython-sql) (7.20.0)
Requirement already satisfied: sqlalchemy>=0.6.7 in ./venv/lib/python3.8/site-packages (from ipython-sql) (1.3.
23)
Collecting prettytable<1
  Downloading prettytable-0.7.2.zip (28 kB)
Requirement already satisfied: ipython-genutils>=0.1.0 in ./venv/lib/python3.8/site-packages (from ipython-sql)
(0.2.0)
Requirement already satisfied: pygments in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql)
(2.8.0)
Requirement already satisfied: traitlets>=4.2 in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython
-sql) (5.0.5)
Requirement already satisfied: jedi>=0.16 in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sq
l) (0.18.0)
Requirement already satisfied: appnope in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql)
(0.1.2)
Requirement already satisfied: pexpect>4.3 in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sq
l) (4.8.0)
Requirement already satisfied: decorator in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql)
(4.4.2)
Requirement already satisfied: backcall in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sql)
(0.2.0)
Requirement already satisfied: setuptools>=18.5 in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipyth
on-sql) (53.0.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in ./venv/lib/python3.8/site-packag
es (from ipython>=1.0->ipython-sql) (3.0.16)
Requirement already satisfied: pickleshare in ./venv/lib/python3.8/site-packages (from ipython>=1.0->ipython-sq
l) (0.7.5)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in ./venv/lib/python3.8/site-packages (from jedi>=0.16->ipyt
hon>=1.0->ipython-sql) (0.8.1)
Requirement already satisfied: ptyprocess>=0.5 in ./venv/lib/python3.8/site-packages (from pexpect>4.3->ipython
>=1.0->ipython-sql) (0.7.0)
Requirement already satisfied: wcwidth in ./venv/lib/python3.8/site-packages (from prompt-toolkit!=3.0.0,!=3.0.
1,<3.1.0,>=2.0.0->ipython>=1.0->ipython-sql) (0.2.5)
Building wheels for collected packages: prettytable
  Building wheel for prettytable (setup.py) ... done
  Created wheel for prettytable: filename=prettytable-0.7.2-py3-none-any.whl size=13699 sha256=e3f16af41a7d8060
72f606653b9d9666d8dea5c6181ff1dd56c4f3a587758e4a
  Stored in directory: /Users/AlesColi/Library/Caches/pip/wheels/48/6d/77/9517cb933af254f51a446f1a5ec9c2be3e45f
```

```
17384940bce68
Successfully built prettytable
Installing collected packages: sqlparse, prettytable, ipython-sql
Successfully installed ipython-sql-0.4.0 prettytable-0.7.2 sqlparse-0.4.1
```

In [2]:

```
import pandas as pd
import numpy as np
from datetime import date, datetime
from pandas_profiling import ProfileReport
import json
import ast
import re
import xml.etree.ElementTree as et
from pprint import pprint
from sqlalchemy import create_engine
```

Read the csv and load it into a Pandas DataFrame object

In order to get the data, download the csv from this link: https://www.kaggle.com/rounakbanik/the-movies-dataset/version/7#movies_metadata.csv

Once the csv file is downloaded, copy or move it to the same directory of this notebook.

While we're loading the data, we are also specifying the columns we want:

- title
- budget
- year
- revenue
- rating
- ratio
- production_companies
- url
- abstract

We're still missing a few columns and we'll extract the extra ones in a bit.

In [3]:

```
mov_metadata = pd.read_csv("movies_metadata.csv",
                           usecols=["title","budget",
```

```
                                "revenue", "production_companies",
                                "release_date", "vote_average"])

mov_metadata = mov_metadata.rename(columns={"vote_average": "rating"})

rows_before = mov_metadata.shape[0]
print(f"\n=== After import ===\nNumber of rows: {rows_before}\nNumber of columns: {mov_metadata.shape[1]}")

mov_metadata = mov_metadata.drop_duplicates()
rows_after = mov_metadata.shape[0]
print(f"\n=== After deduplication ===\nNumber of rows: {rows_after}\nNumber of columns: {mov_metadata.shape[1]}
print(f"\nWe deleted {rows_before - rows_after} duplicated rows")
```

```
=== After import ===
Number of rows: 45466
Number of columns: 6

=== After deduplication ===
Number of rows: 45434
Number of columns: 6

We deleted 32 duplicated rows
```

Enforcing column data types

In order to have control over column types and Null values we enforce the dtypes:

column	dtype	may contain Nulls
title	string (with capitalization)	True
budget	Float64	True
year	Int64	True
revenue	Float64	True
rating (ex vote_average)	Float64	True
production_companies	string (with capitalization)	filled with "[]"

In [4]:

```
mov_metadata["budget"] = pd.to_numeric(mov_metadata["budget"], errors="coerce").astype("Float64")
mov_metadata["revenue"] = pd.to_numeric(mov_metadata["revenue"], errors="coerce").astype("Float64")

mov_metadata["rating"] = pd.to_numeric(mov_metadata["rating"], errors="coerce").astype("Float64")
```

```
mov_metadata["year"] = pd.to_datetime(mov_metadata["release_date"], errors="coerce").dt.year.astype("Int64")
mov_metadata = mov_metadata.drop(columns=["release_date"])

mov_metadata["production_companies"] = mov_metadata["production_companies"].astype("string")
mov_metadata["production_companies"] = mov_metadata.production_companies.fillna("[]")

mov_metadata["title"] = mov_metadata.title.str.title()

print("To confirm the data types of our table we can print out the schema:")
pprint(pd.io.json.build_table_schema(mov_metadata)["fields"])
```

```
To confirm the data types of our table we can print out the schema:
[{'name': 'index', 'type': 'integer'},
 {'name': 'budget', 'type': 'number'},
 {'name': 'production_companies', 'type': 'string'},
 {'name': 'revenue', 'type': 'number'},
 {'name': 'title', 'type': 'string'},
 {'name': 'rating', 'type': 'number'},
 {'name': 'year', 'type': 'integer'}]
```

In [5]:

```
profile = ProfileReport(mov_metadata, title='Movie Metadata Profiling Report', explorative=True)
# profile
```

Let's talk to the stakeholders

At this stage I think it's a good idea to talk to the stakeholders that are going to use the table so that we can decide together how to best handle the data.

In the cell above we created a report of the data and the key findings are:

- budget has 36551 (80.4%) zeros
- revenue has 38032 (83.7%) zeros
- there is high correlation between budget and revenue (makes sense)

The high number of zeros in budget and revenue makes our next move difficult because we want to create a new metric based on revenue/budget.

Since our driver is finding films with high revenue we can start by removing all the rows where `revenue` is 0 because having no revenue makes the row either "corrupt" or "not interesting" for us.

Similarly, since TrueFilm is an investment company it's likely that productions with zero budget are not particularly interesting so I'm going to eliminate all the rows where `budget` = 0

As I said before, these are assumptions I am making using my common sense but since we are losing 80% of the rows I would never recommend this in a real scenario without talking to the stakeholders.

Possible workarounds

A range of possible solutions could be used to impute `budget` and `revenue` :

- Mean values from a combination of production_companies, genre and rating
- Mean values from a combination of genre and rating
- Mean values from a combination of production_companies and rating
- Average values from productions with overlapping or similar keywords(`keywords.csv` file)

Important to note that whenever we impute data we are creating a bias due to our human nature hence the importance of agreeing this with the stakeholders.

My solutions for this exercise

I decided, for this challenge, to remove completely the rows where budget and revenue are zero. This leaves us with a lot of rows where the budget of the revenue are under 1000. To avoid losing even more data I will instead adjust those values arbitrarily. In order to signal that my values have been altered I'm creating extra columns to flag estimated values.

Extract company names from string representing dictionaries

In [6]:

```
def extract_companies(text):
    _list = ast.literal_eval(text)
    if isinstance(_list, list):
        companies = []
        for elem in _list:
            if isinstance(elem, dict):
                companies.append(elem["name"])
        return companies
    else:
        return list()


mov_metadata["production_companies"] = mov_metadata.production_companies.apply(lambda x: extract_companies(x))
```

Run the adjustments for `budget` and `revenue`

In [7]:

```
def adjust_value(num):
    if isinstance(num, float):
        if num < 100:
            return num * 1000000
        elif num >= 100 and num < 1000:
            return num * 1000
        else:
            return num


mov_metadata['budget_estimated'] = False
mov_metadata.loc[mov_metadata.budget < 1000, 'budget_estimated'] = True

mov_metadata['revenue_estimated'] = False
mov_metadata.loc[mov_metadata.budget < 1000, 'revenue_estimated'] = True

mov_metadata['budget'] = mov_metadata['budget'].apply(lambda x: adjust_value(x))
mov_metadata['revenue'] = mov_metadata['revenue'].apply(lambda x: adjust_value(x))
mov_metadata
```

Out[7]:

	budget	production_companies	revenue	title	rating	year	budget_estimated	revenue_estimated
0	30000000.0	[Pixar Animation Studios]	373554033.0	Toy Story	7.7	1995	False	False
1	65000000.0	[TriStar Pictures, Teitler Film, Interscope Co...	262797249.0	Jumanji	6.9	1995	False	False
2	0.0	[Warner Bros., Lancaster Gate]	0.0	Grumpier Old Men	6.5	1995	True	True
3	16000000.0	[Twentieth Century Fox Film Corporation]	81452156.0	Waiting To Exhale	6.1	1995	False	False
4	0.0	[Sandollar Productions, Touchstone Pictures]	76578911.0	Father Of The Bride Part Ii	5.7	1995	True	True
...	...	...	...	...	...	...	...	...
45461	0.0	[]	0.0	Subdue	4.0	<NA>	True	True
45462	0.0	[Sine Olivia]	0.0	Century Of Birthing	9.0	2011	True	True
45463	0.0	[American World Pictures]	0.0	Betrayal	3.8	2003	True	True

	budget	production_companies	revenue	title	rating	year	budget_estimated	revenue_estimated
45464	0.0	[Yermoliev]	0.0	Satan Triumphant	0.0	1917	True	True
45465	0.0	[]	0.0	Queerama	0.0	2017	True	True

45434 rows × 8 columns

Drop the zeros and create ratio column

In [8]:

```
mov_metadata.drop(mov_metadata[mov_metadata.budget == 0].index, inplace=True)
mov_metadata.drop(mov_metadata[mov_metadata.revenue == 0].index, inplace=True)
mov_metadata["ratio"] = mov_metadata.revenue/mov_metadata.budget
mov_metadata
```

Out[8]:

	budget	production_companies	revenue	title	rating	year	budget_estimated	revenue_estimated	ratio
0	30000000.0	[Pixar Animation Studios]	373554033.0	Toy Story	7.7	1995	False	False	12.451801
1	65000000.0	[TriStar Pictures, Teitler Film, Interscope Co...	262797249.0	Jumanji	6.9	1995	False	False	4.043035
3	16000000.0	[Twentieth Century Fox Film Corporation]	81452156.0	Waiting To Exhale	6.1	1995	False	False	5.090760
5	60000000.0	[Regency Enterprises, Forward Pass, Warner Bros.]	187436818.0	Heat	7.7	1995	False	False	3.123947
8	35000000.0	[Universal Pictures, Imperial Entertainment, S...	64350171.0	Sudden Death	5.5	1995	False	False	1.838576
...	...	...	...	...	...	...	...	...	...
45250	12000000.0	[AVM Productions]	19000000.0	Sivaji: The Boss	6.9	2007	False	False	1.583333
45399	750000.0	[Кинокомпания «Lunapark», Инвада фильм]	3000000.0	All At Once	6.0	2014	False	False	4.000000
45409	800000.0	[]	1328612.0	Savages	5.8	2006	False	False	1.660765

	budget	production_companies	revenue	title	rating	year	budget_estimated	revenue_estimated	ratio
45412	2000000.0	[Profit]	1268793.0	Pro Lyuboff	4.0	2010	False	False	0.634397
45422	5000000.0	[]	1413000.0	Antidur	1.0	2007	False	False	0.282600

5378 rows × 9 columns

Load Wikipedia data

Once you have downloaded the file at this link: https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-abstract.xml.gz you'll need to un-zip it and copy/move `enwiki-latest-abstract.xml` to the same folder of this notebook and the previous dataset.

The next operation is going to take a while between 10 and 30 minutes, depending on your computer) as we are parsing the xml into a pandas data frame. For simplicity we will write the parsed data frame to a local csv so that we have a "quick to load" copy in case we need to run this again.

I got the following function from another developer, for reference: https://medium.com/@robertopreste/from-xml-to-pandas-dataframes-9292980b1c1c

In [9]:

```
def parse_xml(xml_file, df_cols):
    """Parse the input XML file and store the result in a pandas
    DataFrame with the given columns.

    The first element of df_cols is supposed to be the identifier
    variable, which is an attribute of each node element in the
    XML data; other features will be parsed from the text content
    of each sub-element.
    """

    xtree = et.parse(xml_file)
    xroot = xtree.getroot()
    rows = []

    for node in xroot:
        res = []
        res.append(node.attrib.get(df_cols[0]))
        for el in df_cols[1:]:
            if node is not None and node.find(el) is not None:
                res.append(node.find(el).text)
            else:
```

```
                res.append(None)
            rows.append({df_cols[i]: res[i]
                         for i, _ in enumerate(df_cols)})

    out_df = pd.DataFrame(rows, columns=df_cols)

    return out_df
```

Once the xml has been parsed you can comment out the next cell as we'll read the dataset from the local file

In [10]:
```
# df_cols = ["doc","title", "url", "abstract", "sublink", "anchor", "link"]
# wiki_df = parse_xml('enwiki-latest-abstract.xml', df_cols)
# wiki_df.to_csv('wiki.csv')
```

In [11]:
```
wiki_df = pd.read_csv("wiki.csv",
                      usecols=["title", "url", "abstract"])
wiki_df["title"] = wiki_df.title.str.title().str.lstrip("Wikipedia: ")
wiki_df.head()
```

Out[11]:

	title	url	abstract
0	Anarchism	https://en.wikipedia.org/wiki/Anarchism	Anarchism is a political philosophy and moveme...
1	Autism	https://en.wikipedia.org/wiki/Autism	| duration = Lifelong
2	Albedo	https://en.wikipedia.org/wiki/Albedo	Albedo () (, meaning 'whiteness') is the measu...
3	A	https://en.wikipedia.org/wiki/A	][][][][][][][][][][]
4	Alabama	https://en.wikipedia.org/wiki/Alabama	(We dare defend our rights)

Merge the 2 datasets

Ideally we would do a merge on some sort of `id` but since the wikipedia xml does not contain anything we can match with, we are forced to match on the movie title vs wikipedia page title. This opens all sorts of complications especially for those "generic" movie titles (eg. "Boy") where we can not do any disambiguation because of the poor wikipedia dataset. Regardless, many url+abstract combinations make sense for the matched title so we can run another ProfileReport to check the data.

In [12]:
```
merged_df = mov_metadata.merge(wiki_df, on="title", how="inner")
merged_df.head()
```

Out[12]:

	budget	production_companies	revenue	title	rating	year	budget_estimated	revenue_estimated	ratio	
0	30000000.0	[Pixar Animation Studios]	373554033.0	Toy Story	7.7	1995	False	False	12.451801	http
1	65000000.0	[TriStar Pictures, Teitler Film, Interscope Co...	262797249.0	Jumanji	6.9	1995	False	False	4.043035	ht
2	60000000.0	[Regency Enterprises, Forward Pass, Warner Bros.]	187436818.0	Heat	7.7	1995	False	False	3.123947	
3	35000000.0	[Universal Pictures, Imperial Entertainment, S...	64350171.0	Sudden Death	5.5	1995	False	False	1.838576	https://e
4	58000000.0	[United Artists, Eon Productions]	352194034.0	Goldeneye	6.6	1995	False	False	6.072311	https

In [13]:

```
profile = ProfileReport(merged_df, title='Pandas Profiling Report', explorative=True)
profile
```

Overview

Dataset statistics

Number of variables	11
Number of observations	4695

Variable types

Numeric	5
Unsupported	1

Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.9 MiB
Average record size in memory	432.1 B

Categorical	2
Boolean	2
URL	1

Warnings

`title` has a high cardinality: 4479 distinct values	**High cardinality**
`abstract` has a high cardinality: 3429 distinct values	**High cardinality**
`budget_estimated` is highly correlated with `revenue_estimated`	**High correlation**
`revenue_estimated` is highly correlated with `budget_estimated`	**High correlation**
`budget_estimated` is highly correlated with `revenue_estimated`	**High correlation**
`revenue_estimated` is highly correlated with `budget_estimated`	**High correlation**
`ratio` is highly skewed (γ1 = 57.92681803)	**Skewed**

Out[13]:

In [14]:
```
merged_df = merged_df[["title","budget", "year","revenue","rating", "ratio",
                       "production_companies", "url", "abstract"]]
merged_df = merged_df.sort_values(by='ratio', ascending=False).head(1000)
merged_df = merged_df.reset_index(drop=True)
merged_df
```

Out[14]:

	title	budget	year	revenue	rating	ratio	production_companies	
0	Paranormal Activity	15000.0	2007	193355800.0	5.9	12890.386667	[Blumhouse Productions, Solana Films]	https://en.wikipedia.org/wiki/Paranormal_Ac

	title	budget	year	revenue	rating	ratio	production_companies	
1	The Blair Witch Project	60000.0	1999	248000000.0	6.3	4133.333333	[Artisan Entertainment, Haxan Films]	https://en.wikipedia.org/wiki/The_Blair_Wit
2	The Tiger: An Old Hunter'S Tale	5000.0	2015	11083449.0	7.5	2216.689800	[Next Entertainment World, Sanai Pictures]	https://en.wikipedia.org/wiki/The_Tiger:_Ar
3	Eraserhead	10000.0	1977	7000000.0	7.5	700.000000	[American Film Institute (AFI), Libra Films]	https://en.wikipedia.org/wiki/Erase
4	Pink Flamingos	12000.0	1972	6000000.0	6.2	500.000000	[Dreamland Productions]	https://en.wikipedia.org/wiki/Pink_Flam
...	...	...	...	...	...	...	...	
995	Shrek The Third	160000000.0	2007	798958165.0	6.0	4.993489	[DreamWorks SKG, DreamWorks Animation]	https://en.wikipedia.org/wiki/Shrek_the_
996	The Lorax	70000000.0	2012	348840316.0	6.3	4.983433	[Universal Pictures, Illumination Entertainment]	https://en.wikipedia.org/wiki/The_
997	Stay Hungry	5000000.0	1976	24854765.0	5.8	4.970953	[United Artists, Outov Productions]	https://en.wikipedia.org/wiki/Stay_H
998	The Help	25000000.0	2011	124272124.0	7.9	4.970885	[DreamWorks SKG, 1492 Pictures, Participant Pr...	https://en.wikipedia.org/wiki/The
999	Identity Thief	35000000.0	2013	173965010.0	5.6	4.970429	[Stuber Productions, Aggregate Films, DumbDumb]	https://en.wikipedia.org/wiki/Identity_

1000 rows × 9 columns

Load the table to postgres

The configuration below is what I used to run the postgres database on my machine. Not being certain of what kind of computer this is going to run on, here is a link to the docs to help you start a server: https://www.postgresql.org/docs/current/server-start.html

In [39]:
```
username = "postgres"
password = "postgres"
host = "localhost"
port = "5432"
database_name = "postgres"
table_name = "truefilm"


engine = create_engine(f"postgresql://{username}:{password}@{host}:{port}/{database_name}")
merged_df.to_sql(table_name, engine)
```

In [35]:
```
query = """
SELECT
    title,
    ratio,
    year,
    rating,
    url
FROM
    "truefilm"
WHERE
    ratio > 2
ORDER BY
    ratio DESC
LIMIT 10
"""
pd.read_sql(query, engine)
```

Out[35]:

	title	ratio	year	rating	url
0	Paranormal Activity	12890.386667	2007	5.9	https://en.wikipedia.org/wiki/Paranormal_Activity
1	The Blair Witch Project	4133.333333	1999	6.3	https://en.wikipedia.org/wiki/The_Blair_Witch_...
2	The Tiger: An Old Hunter'S Tale	2216.689800	2015	7.5	https://en.wikipedia.org/wiki/The_Tiger:_An_Ol...
3	Eraserhead	700.000000	1977	7.5	https://en.wikipedia.org/wiki/Eraserhead

	title	ratio	year	rating	url
4	Pink Flamingos	500.000000	1972	6.2	https://en.wikipedia.org/wiki/Pink_Flamingos
5	Super Size Me	439.616585	2004	6.6	https://en.wikipedia.org/wiki/Super_Size_Me
6	The Gallows	426.644100	2015	4.9	https://en.wikipedia.org/wiki/The_Gallows
7	Open Water	420.522723	2004	5.3	https://en.wikipedia.org/wiki/Open_Water
8	The Texas Chain Saw Massacre	363.047059	1974	7.1	https://en.wikipedia.org/wiki/The_Texas_Chain_...
9	Bambi	311.709965	1942	6.8	https://en.wikipedia.org/wiki/Bambi

In [38]:

```
query = """
SELECT
    *
FROM
    "truefilm"
ORDER BY
    ratio DESC
LIMIT 10
"""
pd.read_sql(query, engine)
```

Out[38]:

	index	title	budget	year	revenue	rating	ratio	production_companies	
0	3106	Paranormal Activity	15000.0	2007	193355800.0	5.9	12890.386667	{"Blumhouse Productions","Solana Films"}	https://en.wikipedia.org/wiki/Paranormal_
1	940	The Blair Witch Project	60000.0	1999	248000000.0	6.3	4133.333333	{"Artisan Entertainment","Haxan Films"}	https://en.wikipedia.org/wiki/The_Blair_W
2	4579	The Tiger: An Old Hunter'S Tale	5000.0	2015	11083449.0	7.5	2216.689800	{"Next Entertainment World","Sanai Pictures"}	https://en.wikipedia.org/wiki/The_Tiger:_
3	1242	Eraserhead	10000.0	1977	7000000.0	7.5	700.000000	{"American Film Institute (AFI)","Libra Films"}	https://en.wikipedia.org/wiki/Eras
4	801	Pink Flamingos	12000.0	1972	6000000.0	6.2	500.000000	{"Dreamland Productions"}	https://en.wikipedia.org/wiki/Pink_Fla

	index	title	budget	year	revenue	rating	ratio	production_companies	
5	2168	Super Size Me	65000.0	2004	28575078.0	6.6	439.616585	{"Kathbur Pictures"}	https://en.wikipedia.org/wiki/Super_S
6	4260	The Gallows	100000.0	2015	42664410.0	4.9	426.644100	{"New Line Cinema","Blumhouse Productions","Ma...	https://en.wikipedia.org/wiki/The_(
7	2316	Open Water	130000.0	2004	54667954.0	5.3	420.522723	{"Plunge Pictures LLC"}	https://en.wikipedia.org/wiki/Oper
8	852	The Texas Chain Saw Massacre	85000.0	1974	30859000.0	7.1	363.047059	{"New Line Cinema",Vortex}	https://en.wikipedia.org/wiki/The_Texas_C
9	682	Bambi	858000.0	1942	267447150.0	6.8	311.709965	{"Walt Disney Productions"}	https://en.wikipedia.org/wiki

In [ ]: