Real-world Data Wrangling

```
#install kaggle package
!pip install kaggle

#install collected packages
!pip install --target=/workspace ucimlrepo numpy==1.24.3
```

1. Gather data

1.1. Problem Statement

In this project, I will download two datasets: a movie dataset containing 45,466 records with variables such as movie details, production information, popularity metrics, financial data, and user ratings, and a Netflix dataset containing 8,807 records based on similar information.

By combining these two datasets, I aim to explore the question:

What are the Top 5 Highest-Grossing Movie Collections by Director

1.2. Gather at least two datasets using two different data gathering methods

Dataset 1

Type: .csv file

Method: Data was downloaded programmatically with a Kaggle API

Dataset variables:

- `adult`: Indicates whether the movie is intended for adult audiences (true/false).
- `belongs_to_collection`: The name of the collection this movie belongs to (if applicable).
- `budget`: The total budget of the movie in monetary terms.
- `genres`: The genres associated with the movie (e.g., Action, Drama, Comedy).
- `homepage`: The official website of the movie.
- `id`: Unique identifier for the movie.
- `imdb_id`: The unique IMDb identifier for the movie.
- `original_language`: The original language in which the movie was filmed.
- `original_title`: The original title of the movie in its native language.
- `overview`: A brief synopsis or description of the movie's plot.
- `popularity`: A measure of the movie's popularity, typically based on user interactions and viewership.
- `poster_path`: URL or path to the movie's poster image.
- `production_companies`: List of companies that produced the movie.

- `production_countries`: Countries where the movie was produced.
- `release_date`: The official release date of the movie.
- `revenue`: The total box office revenue the movie earned.
- `runtime`: The duration of the movie in minutes.
- `spoken_languages`: Languages spoken in the movie.
- `status`: The current status of the movie (e.g., Released, Post-production, etc.).
- `tagline`: A short, catchy phrase or slogan associated with the movie.
- `title`: The title of the movie.
- `video`: Indicates whether a video is available (true/false).
- `vote_average`: The average rating given by users (e.g., on IMDb).
- `vote_count`: The total number of votes or ratings the movie has received.

```
import os
os.environ['PATH'] = os.environ['PATH'] + ':/home/student/.local/bin'

!kaggle datasets download -d rounakbanik/the-movies-dataset -
p ./movies --unzip

Dataset URL: https://www.kaggle.com/datasets/rounakbanik/the-movies-
dataset
License(s): CC0-1.0
Downloading the-movies-dataset.zip to ./movies
100%|████████████████████████████████████████| 228M/228M
[00:00<00:00, 252MB/s]
100%|████████████████████████████████████████| 228M/228M
[00:00<00:00, 253MB/s]

import pandas as pd

raw_metadata = pd.read_csv("./movies/movies_metadata.csv")

/tmp/ipykernel_396/1241359816.py:3: DtypeWarning: Columns (10) have
mixed types. Specify dtype option on import or set low_memory=False.
  raw_metadata = pd.read_csv("./movies/movies_metadata.csv")

raw_metadata.head()

   adult                              belongs_to_collection    budget
\
0  False  {'id': 10194, 'name': 'Toy Story Collection', ...  30000000

1  False                                                NaN  65000000

2  False  {'id': 119050, 'name': 'Grumpy Old Men Collect...         0

3  False                                                NaN  16000000

4  False  {'id': 96871, 'name': 'Father of the Bride Col...         0
```

```
                                                genres  \
0  [{'id': 16, 'name': 'Animation'}, {'id': 35, '...
1  [{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2  [{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
3  [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
4                    [{'id': 35, 'name': 'Comedy'}]

                                homepage     id    imdb_id
original_language  \
0  http://toystory.disney.com/toy-story    862  tt0114709
en
1                                   NaN   8844  tt0113497
en
2                                   NaN  15602  tt0113228
en
3                                   NaN  31357  tt0114885
en
4                                   NaN  11862  tt0113041
en

                original_title  \
0                    Toy Story
1                      Jumanji
2             Grumpier Old Men
3            Waiting to Exhale
4  Father of the Bride Part II

                                             overview  ... release_date
\
0  Led by Woody, Andy's toys live happily in his ...  ...   1995-10-30

1  When siblings Judy and Peter discover an encha...  ...   1995-12-15

2  A family wedding reignites the ancient feud be...  ...   1995-12-22

3  Cheated on, mistreated and stepped on, the wom...  ...   1995-12-22

4  Just when George Banks has recovered from his ...  ...   1995-02-10


       revenue runtime
spoken_languages  \
0  373554033.0    81.0           [{'iso_639_1': 'en', 'name':
'English'}]
1  262797249.0   104.0  [{'iso_639_1': 'en', 'name': 'English'},
{'iso...
2          0.0   101.0           [{'iso_639_1': 'en', 'name':
'English'}]
3   81452156.0   127.0           [{'iso_639_1': 'en', 'name':
'English'}]
```

```
4   76578911.0   106.0           [{'iso_639_1': 'en', 'name':
'English'}]

     status                                              tagline  \
0  Released                                                 NaN
1  Released          Roll the dice and unleash the excitement!
2  Released  Still Yelling. Still Fighting. Still Ready for...
3  Released  Friends are the people who let you be yourself...
4  Released  Just When His World Is Back To Normal... He's ...

                         title  video vote_average vote_count
0                    Toy Story  False          7.7     5415.0
1                      Jumanji  False          6.9     2413.0
2             Grumpier Old Men  False          6.5       92.0
3            Waiting to Exhale  False          6.1       34.0
4  Father of the Bride Part II  False          5.7      173.0

[5 rows x 24 columns]
```

Dataset 2

Type: .csv file

Method: Data was downloaded from kaggle manually to my local computer and uploaded to the jupyter working directory.

Dataset variables:

- `show_id`: A unique identifier for each show or movie in the dataset.
- `type`: The type of content (e.g., Movie, TV Show).
- `title`: The title of the show or movie.
- `director`: The director(s) responsible for creating the show or movie.
- `cast`: A list of key actors and actresses in the show or movie.
- `country`: The country or countries where the show or movie takes place.
- `date_added`: The date the show or movie was added to the Netflix platform.
- `release_year`: The year the show or movie was originally released.
- `rating`: The age rating of the content (e.g., PG, R, TV-MA).
- `duration`: The total length of the movie in minutes or the number of seasons/episodes for a TV show.
- `listed_in`: The categories or genres the show or movie is listed under on Netflix (e.g., Action, Comedy, Drama).
- `description`: A brief synopsis or summary of the show or movie's plot.

```
raw_netflix = pd.read_csv("netflix_titles.csv")
raw_netflix.head()

  show_id     type                  title         director  \
0      s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1      s2  TV Show          Blood & Water              NaN
```

```
2      s3  TV Show              Ganglands  Julien Leclercq
3      s4  TV Show  Jailbirds New Orleans              NaN
4      s5  TV Show           Kota Factory              NaN

                                                  cast        country  \
0                                                  NaN  United States
1  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...   South Africa
2  Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...            NaN
3                                                  NaN            NaN
4  Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...          India

           date_added  release_year rating   duration  \
0  September 25, 2021          2020  PG-13     90 min
1  September 24, 2021          2021  TV-MA  2 Seasons
2  September 24, 2021          2021  TV-MA   1 Season
3  September 24, 2021          2021  TV-MA   1 Season
4  September 24, 2021          2021  TV-MA  2 Seasons

                                        listed_in  \
0                                   Documentaries
1    International TV Shows, TV Dramas, TV Mysteries
2  Crime TV Shows, International TV Shows, TV Act...
3                            Docuseries, Reality TV
4  International TV Shows, Romantic TV Shows, TV ...

                                         description
0  As her father nears the end of his life, filmm...
1  After crossing paths at a party, a Cape Town t...
2  To protect his family from a powerful drug lor...
3  Feuds, flirtations and toilet talk go down amo...
4  In a city of coaching centers known to train I...
```

2. Assess data

Quality Issue - Metadata - Completeness

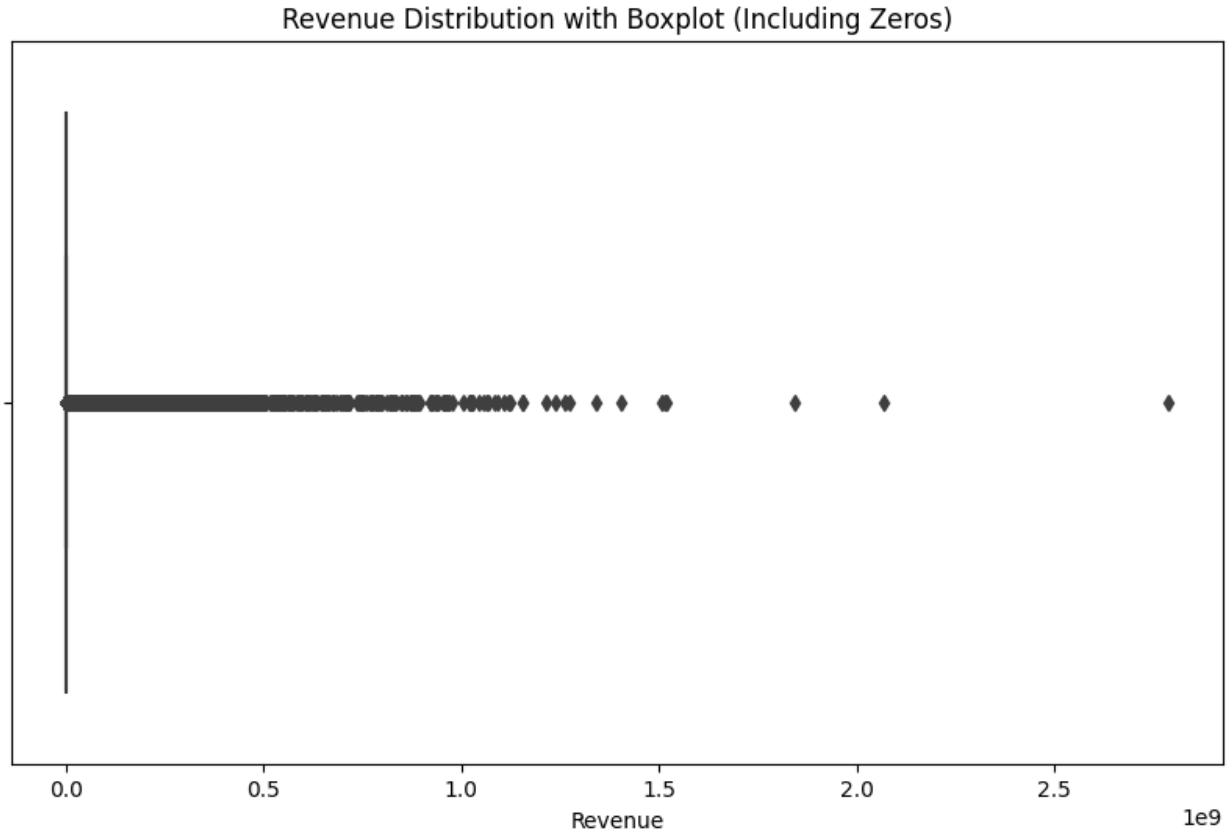
The Movie Metadata dataset from Kaggle contains missing values in the revenue variable, where these missing values are incorrectly represented as zeros. This is a data quality issue under the Completeness pillar, as zeros are not valid substitutes for missing data.

Zeros may falsely indicate no revenue, leading to misleading conclusions. To address this, I will replace the zeros with NaN (Not a Number) values and later remove these entries. This approach ensures the analysis reflects true revenue generation.

The issue is illustrated both visually in a distribution plot and programmatically by using the `isnull()` function to identify missing values after replacing zeros with NaN.

```
#Inspecting the dataframe visually
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Boxplot to visualize revenue distribution and potential outliers
(zeros)
plt.figure(figsize=(10, 6))
sns.boxplot(x=raw_metadata['revenue'])
plt.title('Revenue Distribution with Boxplot (Including Zeros)')
plt.xlabel('Revenue')
plt.show()
```



As you can see in the boxplot above, there are no boxes actually visiable, this is because most of the revenue entries are inputed as zeros. This creates many outliers before we remove them from the dataset

```
#Inspecting the dataframe programmatically
import numpy as np

metadata_copy = raw_metadata.copy()

# Replace all zeros in the 'revenue' column with NaN in the copy
metadata_copy['revenue'] = metadata_copy['revenue'].replace(0, np.nan)

# Check how many NaN (missing) values are now in the 'revenue' column
in the copy
```

```
missing_count = metadata_copy['revenue'].isnull().sum()
print(f"Number of missing values in 'revenue' after replacement:
{missing_count}")
```

```
Number of missing values in 'revenue' after replacement: 38058
```

Justification:

Having 38,058 missing values as zeros can be problematic because it suggests that these movies generated no revenue at all. This is misleading because the actual revenue for these movies is unknown, and assuming zero revenue can skew financial analysis. The large number of zeros can artificially lower the overall revenue estimate, which in turn affects the reliability and accuracy of any conclusions what movie collections are the highest grossing.

Tidiness Issue - Metadata - Each variable forms a column

The Metadata dataframe has an issue of "each variable forms a column" because the current structure of the `belongs_to_collection` variable stores multiple pieces of information (such as the movie's ID, series name, and other details) in a single column as a comma-separated list. The second entry in this list specifically describes the series name, while the other entries are not relevant for analysis. It can be confusing to identify what series the movie belows to with all this information

```
#Inspecting the dataframe visually
print(raw_metadata['belongs_to_collection'].head())
```

```
0    {'id': 10194, 'name': 'Toy Story Collection', ...
1                                                  NaN
2    {'id': 119050, 'name': 'Grumpy Old Men Collect...
3                                                  NaN
4    {'id': 96871, 'name': 'Father of the Bride Col...
Name: belongs_to_collection, dtype: object
```

```
#Inspecting the dataframe programmatically
unique_types = raw_metadata['belongs_to_collection'].unique()
print(f"Unique values in 'belongs_to_collection' column:
{unique_types}")
```

```
Unique values in 'belongs_to_collection' column: ["{'id': 10194,
'name': 'Toy Story Collection', 'poster_path':
'/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg', 'backdrop_path':
'/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}"
 nan
 "{'id': 119050, 'name': 'Grumpy Old Men Collection', 'poster_path':
'/nLvUdqgPgm3F85NMCii9gVFUcet.jpg', 'backdrop_path':
'/hypTnLot2z8wpFS7qwsQHW1uV8u.jpg'}"
 ...
 "{'id': 148603, 'name': 'Ducobu Collection', 'poster_path':
'/rd7AWZUy2QFPIblNWToVmdfXQcA.jpg', 'backdrop_path':
'/7mzKmoIrvGapvsSbAVlX4HtCnFj.jpg'}"
```

```
 "{'id': 152918, 'name': 'Mister Blot Collection', 'poster_path':
'/44PYEwwjGts8pAob59RHd6zlkKc.jpg', 'backdrop_path':
'/5uoPsNiFpUYNamSGqE8okN27VRK.jpg'}"
 "{'id': 200641, 'name': 'Red Lotus Collection', 'poster_path':
'/yf9Eod9ANXyHTzDpAxz9ezgvlL4.jpg', 'backdrop_path':
'/3fhHbLeO3DqdHvgHg5szs399eBb.jpg'}"]
```

Justification:

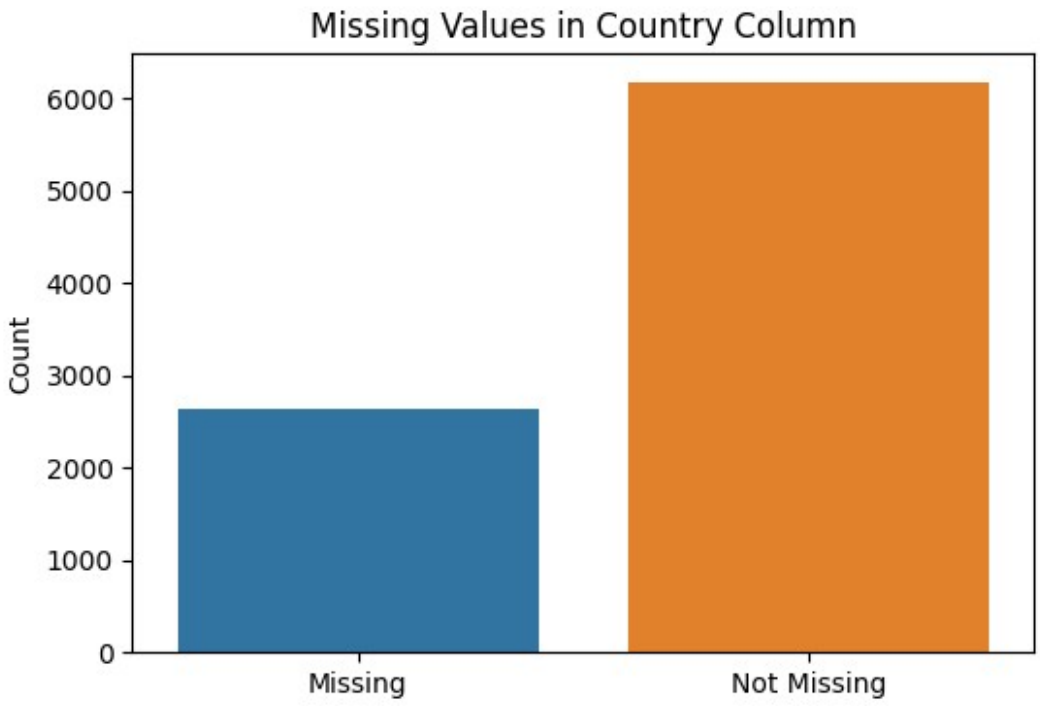
Having each variable in its own column is a core principle of tidy data because it ensures consistency, clarity, and ease of analysis. When each variable is separate, the dataset is structured in a way that makes it simple to manipulate, aggregate, filter, and visualize the data. It allows for clear relationships between variables, making analysis more straightforward and intuitive.

Quality Issue - Netflix - Completeness

The Netflix dataframe has a completeness issue with the `director` variable. In order to answer the reasearch question we will want look at who directed the netflix movie with highest-gross revenue

```
#Inspecting the dataframe visually
#Count the number of missing values under the country variable
missing_values = raw_netflix['director'].isnull().sum()

# Plot the missing values vs the not missing values
plt.figure(figsize=(6, 4))
sns.barplot(x=['Missing', 'Not Missing'], y=[missing_values,
len(raw_netflix) - missing_values])
plt.title('Missing Values in Country Column')
plt.ylabel('Count')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future
version.
  order = pd.unique(vector)
```




```
#Inspecting the dataframe programmatically
missing_count = raw_netflix['director'].isnull().sum()
print(f"Number of missing values in 'director': {missing_count}")
```

```
Number of missing values in 'director': 2634
```

Justification:

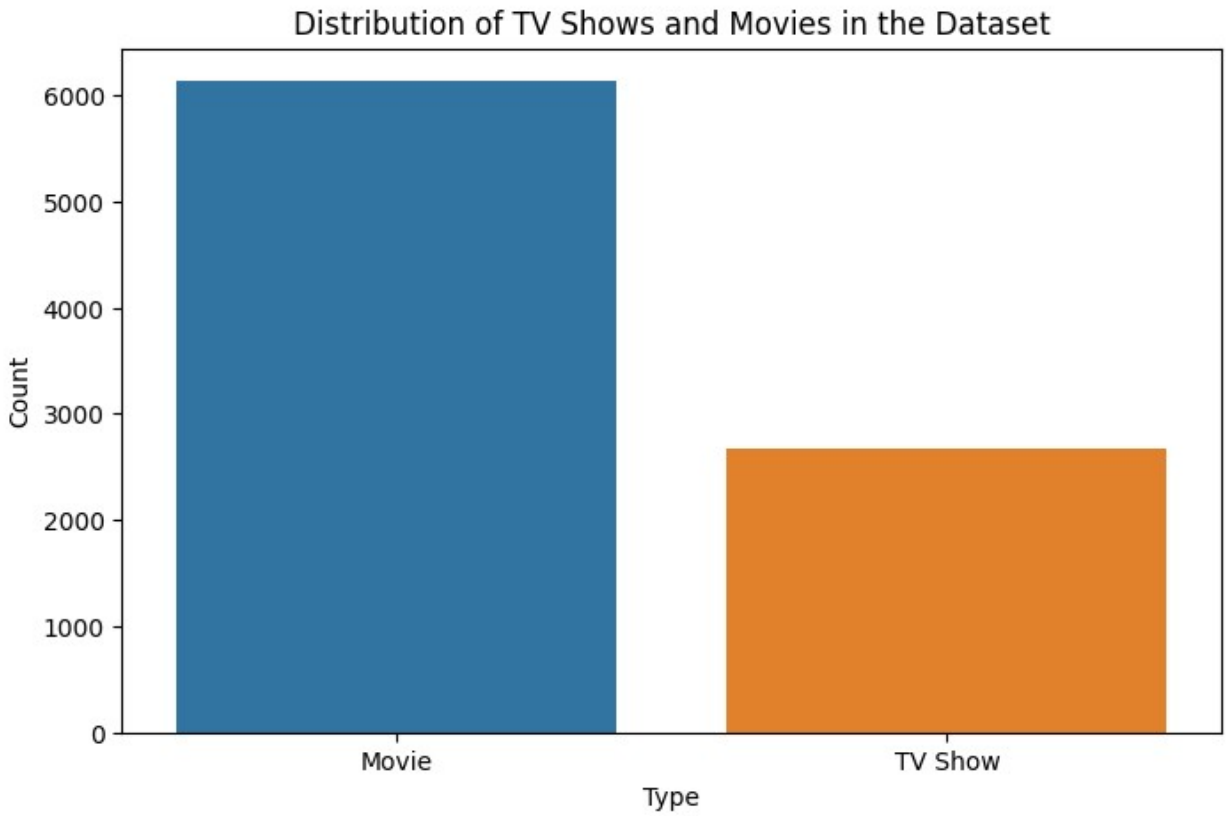
When we want to ultimately answer our research question, **What are the Top 5 Highest-Grossing Movie Collections by Director** we dont want the movie series to have no director at all. Here it is important to check that there are more values that arent missing so that when we join the dataframes together we still have enough values to analyze. As you can see above there are less missing values in the dataframe, so we can ultimately conclude which directors produced the highest grossing movies.

Tidy Issue - Netflix - Each type of observational unit forms a table

The Netflix dataframe has observational unit issues. The `type` variable is used to label the data as either a TV Show or a Movie. Since our analysis focuses only on movies, we will need to filter the dataset to retain only the rows labeled as "Movie." This ensures consistency within the dataset, as we aim to merge this filtered dataset with another containing only movies.

```
#Inspecting the dataframe visually
# Count plot to show distribution of TV Shows and Movies
plt.figure(figsize=(8, 5))
sns.countplot(data=raw_netflix, x='type')
plt.title('Distribution of TV Shows and Movies in the Dataset')
```

```
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



Here we can see that the Netflix dataframe includes both Movies and TV Shows

```
#Inspecting the dataframe programmatically
# Check the unique values in the 'type' column
unique_types = raw_netflix['type'].unique()
print(f"Unique values in 'type' column: {unique_types}")

Unique values in 'type' column: ['Movie' 'TV Show']
```

Justification

The observational unit issue arises because the Netflix dataset contains both TV shows and movies under the `type` column, which can lead to misleading analysis if not addressed. When analyzing the research question, **What are the Top 5 Highest-Grossing Movie Collections by Director** mixing in TV shows introduces irrelevant data, skewing results and distorting comparisons. This inconsistency undermines the integrity of the analysis, as it becomes impossible to accurately assess or compare metrics for movies alone

3. Clean data

```
cleaned_meta = raw_metadata.copy()
cleaned_netflix = raw_netflix.copy()
```

Quality Issue - Metadata - Completeness

The Movie Metadata dataset from Kaggle contains missing values in the revenue variable, where these missing values are incorrectly represented as zeros. This is a data quality issue under the Completeness pillar, as zeros are not valid substitutes for missing data.

```
# Apply the cleaning strategy
# Replace all zeros with NAN values
cleaned_meta['revenue'] = cleaned_meta['revenue'].replace(0, np.nan)

# Remove rows where 'revenue' is NaN
cleaned_meta = cleaned_meta.dropna(subset=['revenue'])

# Validate the cleaning was successful
missing_count = cleaned_meta['revenue'].isnull().sum()
print(f"Number of missing values in 'revenue' after cleaning:
{missing_count}")
```

```
Number of missing values in 'revenue' after cleaning: 0
```

Justification

First I replaced all zero values with Na values, this way I can drop all missing values at the same time in the next step. That where I ran the `.dropna()` function which dropped all entries in the revenue column with an assigned value of NA. This will allow us to further analyze movies with revenue in the future. We can confirm that all entries now have a revenue value because there are zero missing values upon the completion of cleaning. We used the `.isnull()` function to confirm.

Tidiness Issue - Metadata - Each variable forms a column

The Metadata dataframe has an issue of "each variable forms a column" because the current structure of the `belongs_to_collection` variable stores multiple pieces of information (such as the movie's ID, series name, and other details) in a single column as a comma-separated list. The second entry in this list specifically describes the series name, while the other entries are not relevant for analysis. It can be confusing to identify what series the movie belows to with all this information

```
import json
# Apply the cleaning strategy
# Function to safely parse a JSON string and extract the 'name' field
def safe_json_parse(x):
    try:
        # Check if x is a string and replace single quotes with double
```

```
quotes
        # to make it valid JSON, then parse it using json.loads().
        # If successful, extract the 'name' field.
        return json.loads(x.replace("'", '"')).get('name') if
isinstance(x, str) else None
    except json.JSONDecodeError:
        # If JSON parsing fails (due to malformed JSON), return None
        return None

# Apply the safe_json_parse function to the 'belongs_to_collection'
column.
cleaned_meta['belongs_to_collection'] =
cleaned_meta['belongs_to_collection'].apply(safe_json_parse)

# Replace any occurrences of 0 in the 'belongs_to_collection' column
with NaN (Not a Number)
cleaned_meta['belongs_to_collection'] =
cleaned_meta['belongs_to_collection'].replace(0, np.nan)

# Remove rows where the 'belongs_to_collection' column is NaN.
# This ensures that only rows with valid collection names remain in
the dataframe.
cleaned_meta = cleaned_meta.dropna(subset=['belongs_to_collection'])

#Validate the cleaning was successful
cleaned_meta.head()
```

```
    adult           belongs_to_collection    budget  \
0   False            Toy Story Collection  30000000
4   False  Father of the Bride Collection         0
9   False           James Bond Collection  58000000
12  False                Balto Collection         0
18  False          Ace Ventura Collection  30000000

                                                 genres  \
0   [{'id': 16, 'name': 'Animation'}, {'id': 35, '...
4                     [{'id': 35, 'name': 'Comedy'}]
9   [{'id': 12, 'name': 'Adventure'}, {'id': 28, '...
12  [{'id': 10751, 'name': 'Family'}, {'id': 16, '...
18  [{'id': 80, 'name': 'Crime'}, {'id': 35, 'name...

                                       homepage     id    imdb_id  \
0           http://toystory.disney.com/toy-story    862  tt0114709
4                                            NaN  11862  tt0113041
9   http://www.mgm.com/view/movie/757/Goldeneye/    710  tt0113189
12                                           NaN  21032  tt0112453
18                                           NaN   9273  tt0112281

   original_language                original_title  \
0                 en                     Toy Story
```

```
4                  en     Father of the Bride Part II
9                  en                       GoldenEye
12                 en                           Balto
18                 en  Ace Ventura: When Nature Calls

                                              overview  ...
release_date  \
0   Led by Woody, Andy's toys live happily in his ...  ...   1995-10-
30
4   Just when George Banks has recovered from his ...  ...   1995-02-
10
9   James Bond must unmask the mysterious head of ...  ...   1995-11-
16
12  An outcast half-wolf risks his life to prevent...  ...   1995-12-
22
18  Summoned from an ashram in Tibet, Ace finds hi...  ...   1995-11-
10

        revenue runtime
spoken_languages  \
0   373554033.0    81.0           [{'iso_639_1': 'en', 'name':
'English'}]
4    76578911.0   106.0           [{'iso_639_1': 'en', 'name':
'English'}]
9   352194034.0   130.0  [{'iso_639_1': 'en', 'name': 'English'},
{'iso...
12   11348324.0    78.0           [{'iso_639_1': 'en', 'name':
'English'}]
18  212385533.0    90.0           [{'iso_639_1': 'en', 'name':
'English'}]

      status                                              tagline  \
0   Released                                                 NaN
4   Released  Just When His World Is Back To Normal... He's ...
9   Released               No limits. No fears. No substitutes.
12  Released                    Part Dog. Part Wolf. All Hero.
18  Released            New animals. New adventures. Same hair.

                           title  video vote_average vote_count
0                      Toy Story  False          7.7     5415.0
4      Father of the Bride Part II  False          5.7      173.0
9                      GoldenEye  False          6.6     1194.0
12                         Balto  False          7.1      423.0
18  Ace Ventura: When Nature Calls  False          6.1     1128.0

[5 rows x 24 columns]
```

```
missing_count = cleaned_meta['belongs_to_collection'].isnull().sum()
print(f"Number of missing values in belongs_to_collection' after
cleaning: {missing_count}")
```

```
Number of missing values in belongs_to_collection' after cleaning: 0
```

Justification

I first created a function that parses the JSON string and extracts the `name` field. If the parsing fails due to invalid JSON, it returns `None`. After defining the function, I applied it to the `belongs_to_collection` variable. This ensures that each movie is either assigned its collection name or `None`, if no valid collection is available. Then, I replaced all `None` values with `NaN` to standardize missing data, and used `.dropna()` function to remove rows with `NaN` values in the `belongs_to_collection` column. Finally, I confirmed the results by inspecting the first 5 values, where each movie now has a valid collection name with no missing values remaining.

Tidy Issue - Netflix - Each type of observational unit forms a table

The Netflix dataframe has observational unit issues. The `type` variable is used to label the data as either a TV Show or a Movie. Since our analysis focuses only on movies, we will need to filter the dataset to retain only the rows labeled as "Movie." This ensures consistency within the dataset, as we aim to merge this filtered dataset with another containing only movies.

```
#Filtering out any TV Show from the netflix dataframe
cleaned_netflix = cleaned_netflix[cleaned_netflix['type'] != 'TV
Show']

#Validate the cleaning was successful
unique_types = cleaned_netflix['type'].unique()
print(f"Unique values in 'type' column: {unique_types}")

Unique values in 'type' column: ['Movie']
```

Justification

Since we eventually are going to combine the two dataframes together, we have no use for any tv show. I update the cleaned_netflix dataframe by dropping all "TV Show" entries under the `type` variable. This leaves only "Movie" types in the net dataframe. I confirmed by checking the unique values and the only one is "Movie".

Quality Issue - Netflix - Completeness

The Netflix dataframe has a completeness issue with the `director` variable. In order to answer the reasearch question we will want look at who directed the netflix movie with highest-gross revenue

```
#Replacing zeros as Na
cleaned_netflix['director'] = cleaned_netflix['director'].replace(0,
np.nan)
```

```
# Remove rows where 'revenue' is NaN
cleaned_netflix = cleaned_netflix.dropna(subset=['director'])

# Validate the cleaning was successful
missing_count = cleaned_netflix['director'].isnull().sum()
print(f"Number of missing values in 'director' after cleaning:
{missing_count}")
```

```
Number of missing values in 'director' after cleaning: 0
```

Justification:

First I replaced all zero values with Na values, this way I can drop all missing values at the same time in the next step. Thats where I ran the `.dropna()` function which dropped all entries in the director column with an assigned value of NA. This will allow us to further analyze movies with a director in the future. We can confirm that all entries now have a director value because there are zero null values after cleaning. We used the `.isnull()` function to confirm

Remove unnecessary variables and combine datasets

These dataframes are not fully cleaned yet, but we've addressed the four key issues necessary to answer my research question. Further cleaning will take place as we remove variables and combine the datasets.

```
#Merge the two dataframes together by movie
merged_movies =cleaned_netflix.merge(cleaned_meta[['original_title',
'revenue', 'belongs_to_collection']],
                left_on='title',
                right_on='original_title',
                how='left')

# Drop the 'original_title' column from 'meta' after merge if it's no
longer needed
merged_movies = merged_movies.drop(columns=['original_title'])

# Drop any Netflix movies that did not record a revenue
merged_movies = merged_movies.dropna(subset=['revenue'])
# Confirm merge was cleaned
merged_movies.head()
```

```
   show_id   type                title          director  \
12     s28  Movie           Grown Ups      Dennis Dugan
19     s42  Movie                Jaws  Steven Spielberg
20     s43  Movie              Jaws 2    Jeannot Szwarc
22     s45  Movie   Jaws: The Revenge    Joseph Sargent
70    s128  Movie  A Cinderella Story       Mark Rosman

                                                 cast
country  \
12  Adam Sandler, Kevin James, Chris Rock, David S...          United
```

```
States
19  Roy Scheider, Robert Shaw, Richard Dreyfuss, L...          United
States
20  Roy Scheider, Lorraine Gary, Murray Hamilton, ...          United
States
22  Lorraine Gary, Lance Guest, Mario Van Peebles,...          United
States
70  Hilary Duff, Chad Michael Murray, Jennifer Coo...  United States,
Canada

            date_added  release_year rating duration  \
12  September 20, 2021          2010  PG-13  103 min
19  September 16, 2021          1975     PG  124 min
20  September 16, 2021          1978     PG  116 min
22  September 16, 2021          1987  PG-13   91 min
70   September 1, 2021          2004     PG   95 min

                                       listed_in  \
12                                      Comedies
19    Action & Adventure, Classic Movies, Dramas
20             Dramas, Horror Movies, Thrillers
22  Action & Adventure, Horror Movies, Thrillers
70            Children & Family Movies, Comedies

                                          description      revenue  \
12  Mourning the loss of their beloved junior high...  271430189.0
19  When an insatiable great white shark terrorize...  470654000.0
20  Four years after the last deadly shark attacks...  187884007.0
22  After another deadly shark attack, Ellen Brody...   51881013.0
70  Teen Sam meets the boy of her dreams at a danc...   70067909.0

          belongs_to_collection
12         Grown Ups Collection
19          The Jaws Collection
20          The Jaws Collection
22          The Jaws Collection
70  Cinderella Story Collection
```

As you can see, we've successfully merged the two dataframes by adding the `revenue` and `belongs_to_collection` variables to the Netflix dataframe. While this merged dataframe is still a work in progress, for the purpose of answering our research question, we only need a few key variables: `title`, `director`, `revenue`, and `belongs_to_collection`. We'll remove the others as we continue cleaning the data.

```
# Delete all irrelevant variable
merged_movies = merged_movies[['title', 'director', 'revenue',
'belongs_to_collection']]

merged_movies.head()
```

```
                 title          director      revenue  \
12           Grown Ups      Dennis Dugan  271430189.0
19                Jaws  Steven Spielberg  470654000.0
20              Jaws 2    Jeannot Szwarc  187884007.0
22   Jaws: The Revenge    Joseph Sargent   51881013.0
70  A Cinderella Story       Mark Rosman   70067909.0

          belongs_to_collection
12         Grown Ups Collection
19          The Jaws Collection
20          The Jaws Collection
22          The Jaws Collection
70  Cinderella Story Collection
```

Tidy Issue - Movies - Validty

As you can see below, the revenue variable includes variables that are hard to interpret. The min is shown as `1.030000e+02` and the max is shown as `1.118889e+09`. The minimum revenue in the dataset is actually 103 dollars and the the maximum revenue is 1,118,889,000, which is slightly more than 1.1 billion dollars. To make this easier to read we will rescale these to in millions.

```
#Description of revenue
merged_movies['revenue'].describe()
```

```
count    1.600000e+02
mean     2.264185e+08
std      2.162254e+08
min      1.030000e+02
25%      6.948545e+07
50%      1.576485e+08
75%      3.233775e+08
max      1.118889e+09
Name: revenue, dtype: float64
```

```
#Rescaling the revenue variable to miilions
merged_movies['revenue_millions'] =merged_movies['revenue'] /
1_000_000
merged_movies = merged_movies.drop(columns=['revenue'])

# Confirm that revenue is now in millions and that original revenue
variable is removed
merged_movies.head()
```

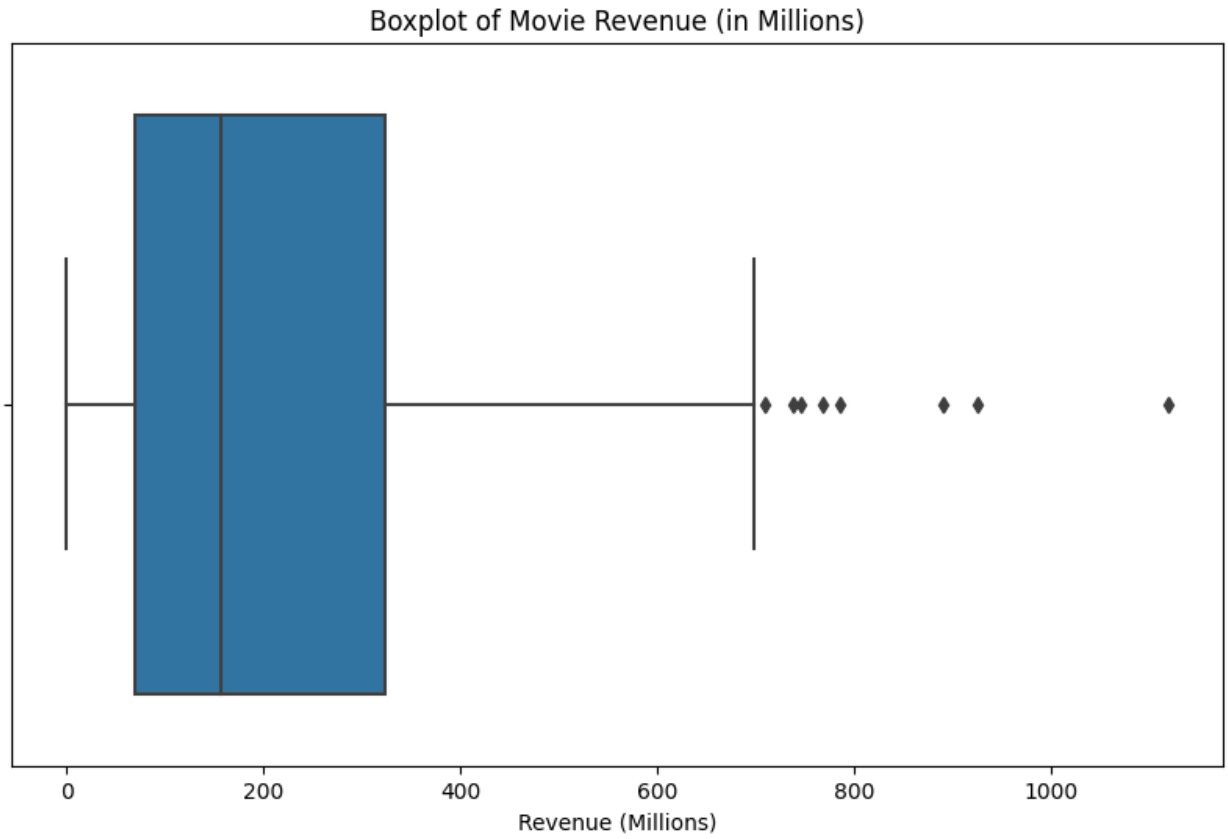
```
                 title          director        belongs_to_collection
\
12           Grown Ups      Dennis Dugan         Grown Ups Collection

19                Jaws  Steven Spielberg          The Jaws Collection
```

```
20              Jaws 2    Jeannot Szwarc          The Jaws Collection

22   Jaws: The Revenge    Joseph Sargent          The Jaws Collection

70  A Cinderella Story       Mark Rosman  Cinderella Story Collection


    revenue_millions
12        271.430189
19        470.654000
20        187.884007
22         51.881013
70         70.067909
```

Justification

Rescaling the revenue data to millions makes the numbers easier to interpret and compare, especially when dealing with very large values. It simplifies the values, reduces the risk of misinterpretation, and aligns with common practices in financial reporting, making the data more accessible and readable.

```
#Boxplot of Movie Revenue in Millions
plt.figure(figsize=(10, 6))
sns.boxplot(x=merged_movies['revenue_millions'])
plt.title('Boxplot of Movie Revenue (in Millions)')
plt.xlabel('Revenue (Millions)')
plt.show()
```



4. Update your data store

Update your local database/data store with the cleaned data, following best practices for storing your cleaned data:

- Must maintain different instances / versions of data (raw and cleaned data)
- Must name the dataset files informatively
- Ensure both the raw and cleaned data is saved to your database/data store

```
#check for workspace directory
import os
current_directory = os.getcwd()
print(current_directory)
```

```
/workspace
```

```
#FILL IN - saving data

#Create the folders
os.makedirs('/workspace/raw', exist_ok=True)
os.makedirs('/workspace/cleaned', exist_ok=True)
os.makedirs('/workspace/merged', exist_ok=True)

# Save raw data in a 'raw' subfolder
```

```
raw_netflix.to_csv('/workspace/raw/raw_netflix.csv', index=False)
raw_metadata.to_csv('/workspace/raw/raw_metadata.csv', index=False)

#Save cleaned data to 'cleaned' subfolder
cleaned_netflix.to_csv('/workspace/cleaned/cleaned_netflix.csv',
index=False)
cleaned_meta.to_csv('/workspace/cleaned/cleaned_meta.csv',
index=False)

# Save the merged data in a 'merged' subfolder
merged_movies.to_csv('/workspace/merged/merged_movies.csv',
index=False)
```

5. Answer the research question

5.1: Define and answer the research question:

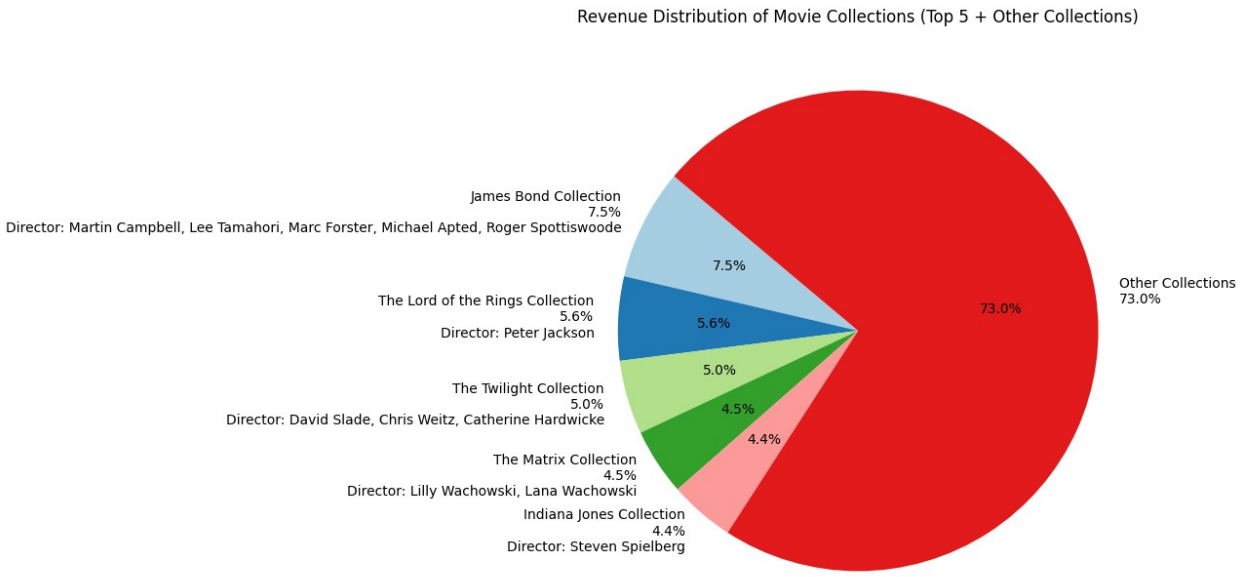
What are the Top 5 Highest-Grossing Movie Collections by Director

```
#Visual 1

# Group by 'belongs_to_collection' and sum the revenue
# Concatenate all directors for each collection
collection_revenue =
merged_movies.groupby('belongs_to_collection').agg({
    'revenue_millions': 'sum',
    'director': lambda x: ', '.join(x.unique())  # Concatenate unique
directors
}).reset_index()

# Sort the collections by total revenue (descending order) and select
the top 5
top_collections = collection_revenue.sort_values('revenue_millions',
ascending=False).head(5)

# Plot the bar chart with color mapping using 'director' for hue
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='belongs_to_collection', y='revenue_millions',
data=top_collections,
                 hue='director', dodge=False, palette='Set1')

# Add director names as annotations on the bars
for i, row in top_collections.iterrows():
    ax.text(i, row['revenue_millions'] + 5, row['director'],
color='black', ha='center')

# Add labels and title
plt.title('Top 5 Highest-Grossing Movie Collections by Director')
plt.xlabel('Movie Collection')
plt.ylabel('Total Revenue (in Millions)')
```

```
# Rotate x-axis labels for readability
plt.xticks(rotation=45, ha='right')

# Show plot
plt.show()
```

```
#Visual 2
# Group by collection and aggregate both revenue and director
collection_info = merged_movies.groupby('belongs_to_collection').agg({
    'revenue_millions': 'sum',
    'director': lambda x: ', '.join(x.unique())  # Join multiple
directors if there are any
}).reset_index()

# Sort collections by total revenue and get the top 5
top_5_collections = collection_info.sort_values('revenue_millions',
ascending=False).head(5)

# Combine all other collections into "Other Collections"
other_collections =
collection_info[~collection_info['belongs_to_collection'].isin(top_5_c
ollections['belongs_to_collection'])]
other_revenue = other_collections['revenue_millions'].sum()

# Create a new row for "Other Collections" with no director info
other_row = pd.DataFrame({
    'belongs_to_collection': ['Other Collections'],
    'revenue_millions': [other_revenue],
    'director': ['']  # No director info for 'Other Collections'
})

# Concatenate the top 5 collections and other collections into a
single DataFrame
top_5_collections = pd.concat([top_5_collections, other_row],
ignore_index=True)

# Plot the pie chart
plt.figure(figsize=(8, 8))
colors = plt.cm.Paired.colors[:len(top_5_collections)]  # Adjust
number of colors
wedges, texts, autotexts =
plt.pie(top_5_collections['revenue_millions'],

labels=top_5_collections['belongs_to_collection'],
                                    autopct='%1.1f%%', startangle=140,
```

```
colors=colors)

# Update labels to show collection name, percentage, and director only
for top 5 collections
for i, text in enumerate(texts):
    director = top_5_collections['director'].iloc[i]
    if director:  # Only add director info if it's available

text.set_text(f"{top_5_collections['belongs_to_collection'].iloc[i]}\
n"
                      f"{autotexts[i].get_text()}\n"
                      f"Director: {director}")
    else:
        # No director info for "Other Collections"

text.set_text(f"{top_5_collections['belongs_to_collection'].iloc[i]}\
n"
                      f"{autotexts[i].get_text()}")

# Add a title
plt.title('Revenue Distribution of Movie Collections (Top 5 + Other
Collections)')

# Show plot
plt.show()
```



What are the Top 5 Highest-Grossing Movie Collections by Director - ANSWERED

```
# Group by 'belongs_to_collection' and sum the revenue
# Concatenate all directors for each collection
collection_revenue =
merged_movies.groupby('belongs_to_collection').agg({
    'revenue_millions': 'sum',
    'director': lambda x: ', '.join(x.unique())  # Concatenate unique
directors
}).reset_index()

# Sort the collections by total revenue (descending order) and select
the top 5
top_collections = collection_revenue.sort_values('revenue_millions',
ascending=False).head(5)

# Format the revenue:
# If revenue is greater than or equal to 1000 million (1 billion),
format as billion (e.g., 2.02 Billion).
top_collections['revenue_millions'] =
top_collections['revenue_millions'].apply(
    lambda x: f"${x/1000:.2f} Billion" if x >= 1000 else f"${x:,.2f}M"
)

# Add a 'Rank' column to explicitly rank the collections
top_collections['Rank'] = range(1, len(top_collections) + 1)

# Print the results in the desired format
for _, row in top_collections.iterrows():
    print(f"The {row['Rank']} highest-grossing movie collection is
'{row['belongs_to_collection']}' directed by {row['director']} with a
total revenue of {row['revenue_millions']}.")
```

```
The 1 highest-grossing movie collection is 'James Bond Collection'
directed by Martin Campbell, Lee Tamahori, Marc Forster, Michael
Apted, Roger Spottiswoode with a total revenue of $2.71 Billion.
The 2 highest-grossing movie collection is 'The Lord of the Rings
Collection' directed by Peter Jackson with a total revenue of $2.05
Billion.
The 3 highest-grossing movie collection is 'The Twilight Collection'
directed by David Slade, Chris Weitz, Catherine Hardwicke with a total
revenue of $1.80 Billion.
The 4 highest-grossing movie collection is 'The Matrix Collection'
directed by Lilly Wachowski, Lana Wachowski with a total revenue of
$1.63 Billion.
The 5 highest-grossing movie collection is 'Indiana Jones Collection'
directed by Steven Spielberg with a total revenue of $1.59 Billion.
```

5.2: Reflection

If I had more time, I'd clean all variables more thoroughly, especially `genres` and `spoken_languages`, which have similar issues as `belongs_to_collection`. Once cleaned, I would revisit my research questions. I originally believed that the issues found in the data should inform the research question, which is why mine is more niche. With a fully cleaned dataset, I'd investigate whether being on Netflix affects revenue generation. I also recognize that the analysis is imbalanced since some collections have more movies, likely inflating their revenue.