MovieAnalysis

September 16, 2025

1 Movie Dataset Analysis

2 This notebook explores the MovieLens + TMDB dataset.

Goals:
- Understand trends in movie production and revenue
- Analyze genres, budgets, revenues
- Build insights for recommendation or success prediction

[1]:
```
# sns.set(style="whitegrid")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

[11]:
```
# LOAD DATA

movies = pd.read_csv(r"C:\Users\user\Downloads\movies_metadata.csv",␣
 ↪low_memory=False)
movies.head()
```

[11]:
```
   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 popularity  \
0  Led by Woody, Andy's toys live happily in his …  21.946943
1  When siblings Judy and Peter discover an encha…  17.015539
2  A family wedding reignites the ancient feud be…    11.7129
3  Cheated on, mistreated and stepped on, the wom…   3.859495
4  Just when George Banks has recovered from his …   8.387519

                       poster_path  \
0  /rhIRbceoE9lR4veEXuwCC2wARtG.jpg
1  /vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg
2  /6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg
3  /16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg
4  /e64sOI48hQXyru7naBFyssKFxVd.jpg

                               production_companies  \
0     [{'name': 'Pixar Animation Studios', 'id': 3}]
1  [{'name': 'TriStar Pictures', 'id': 559}, {'na…
2  [{'name': 'Warner Bros.', 'id': 6194}, {'name'…
3  [{'name': 'Twentieth Century Fox Film Corporat…
4  [{'name': 'Sandollar Productions', 'id': 5842}…

                               production_countries release_date  \
0  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-10-30
1  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-15
2  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-22
3  [{'iso_3166_1': 'US', 'name': 'United States o…   1995-12-22
4  [{'iso_3166_1': 'US', 'name': 'United States o…   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
```

[10]:
```
# Inspect Data
pd.set_option("display.max_columns",25)
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   adult                  45466 non-null  object
 1   belongs_to_collection  4494 non-null   object
 2   budget                 45466 non-null  object
 3   genres                 45466 non-null  object
 4   homepage               7782 non-null   object
 5   id                     45466 non-null  object
 6   imdb_id                45449 non-null  object
 7   original_language      45455 non-null  object
 8   original_title         45466 non-null  object
 9   overview               44512 non-null  object
 10  popularity             45461 non-null  object
 11  poster_path            45080 non-null  object
 12  production_companies   45463 non-null  object
 13  production_countries   45463 non-null  object
 14  release_date           45379 non-null  object
 15  revenue                45460 non-null  float64
 16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object
 18  status                 45379 non-null  object
 19  tagline                20412 non-null  object
 20  title                  45460 non-null  object
 21  video                  45460 non-null  object
 22  vote_average           45460 non-null  float64
 23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
```

```
memory usage: 8.3+ MB
```

3 Here we see which columns have missing values. Some columns like homepage, tagline, and belongs_to_collection have too many missing values, so we may drop them.

[6]:
```
movies.isnull().sum().sort_values(ascending=False).head(20)
```

[6]:
```
belongs_to_collection    40972
homepage                 37684
tagline                  25054
overview                   954
poster_path                386
runtime                    263
status                      87
release_date                87
imdb_id                     17
original_language           11
vote_average                 6
vote_count                   6
title                        6
video                        6
spoken_languages             6
revenue                      6
popularity                   5
production_countries         3
production_companies         3
genres                       0
dtype: int64
```

4 We cleaned missing values, fixed date format, and added a new column release_year.

[12]:
```
movies.columns
```

[12]:
```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

[ ]:
```
#  Drop irrelevant/sparse columns
movies = movies.drop(columns=["belongs_to_collection","homepage","tagline"])
```

[ ]:
```
# Handle missing values in critical fields
movies["runtime"].fillna(movies["runtime"].median())
movies["original_language"].fillna(movies["original_language"].mode()[0],␣
↪inplace=True)
```

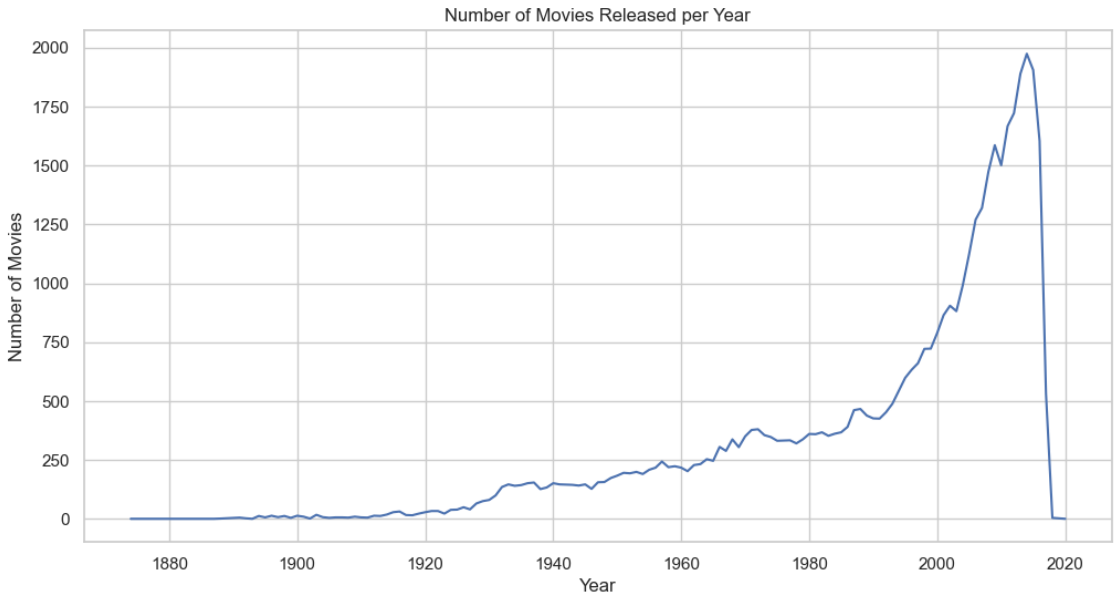
[19]:
```
# Fix release_date
movies['release_date']=pd.to_datetime(movies['release_date'], errors='coerce')
movies = movies.dropna(subset=['release_date'])
```

[22]:
```
# Extract release year
pd.options.mode.copy_on_write = True
movies['release_year'] = movies['release_date'].dt.year
```

5 Exploratory Analysis

6 Shows trends in movie production over time

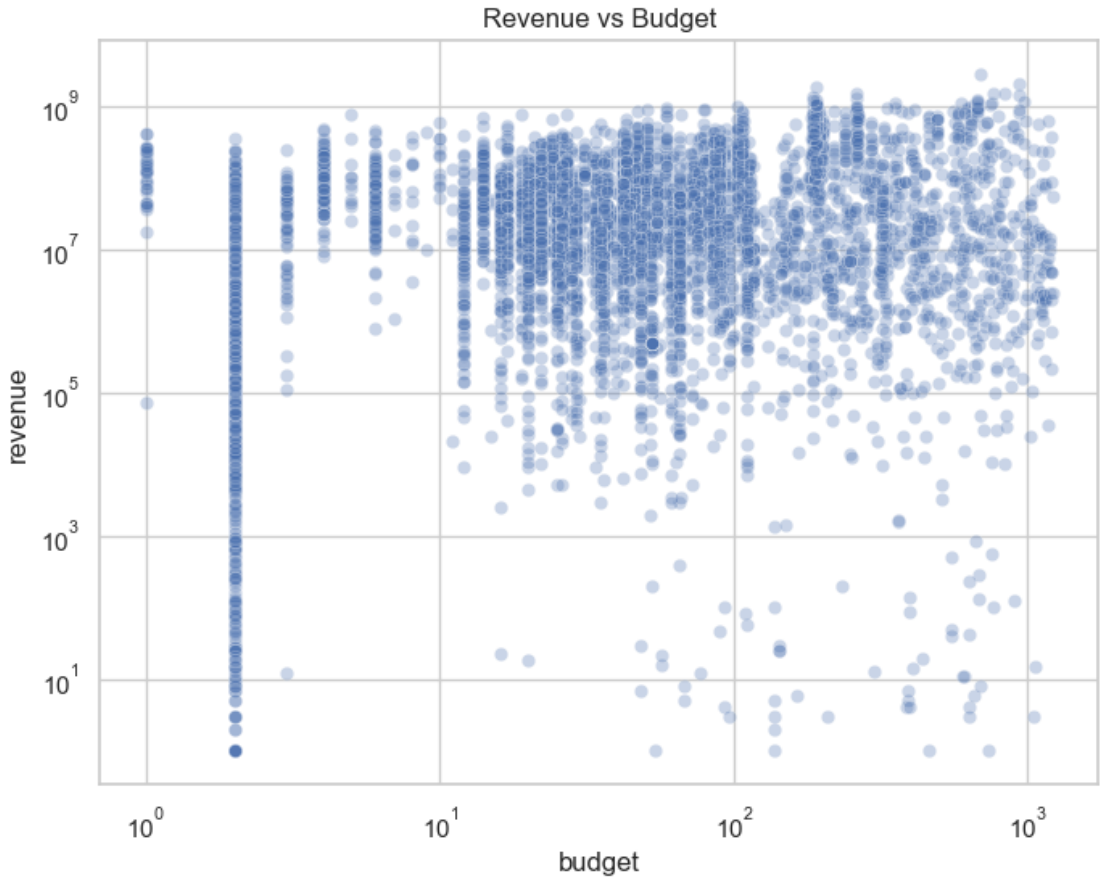
[29]:
```
movies_per_year = movies['release_year'].value_counts().sort_index()
plt.figure(figsize=(12,6))
sns.lineplot(x=movies_per_year.index, y=movies_per_year.values)
plt.title("Number of Movies Released per Year")
plt.xlabel("Year")
plt.ylabel("Number of Movies")
plt.show()
```



7 Revenue vs Budget

8 Helps see if higher budget means higher revenue.

[30]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=movies, x="budget", y="revenue", alpha=0.3)
plt.title("Revenue vs Budget")
plt.xscale("log")
plt.yscale("log")
plt.show()
```



9 Tells us which genres earn the most money.

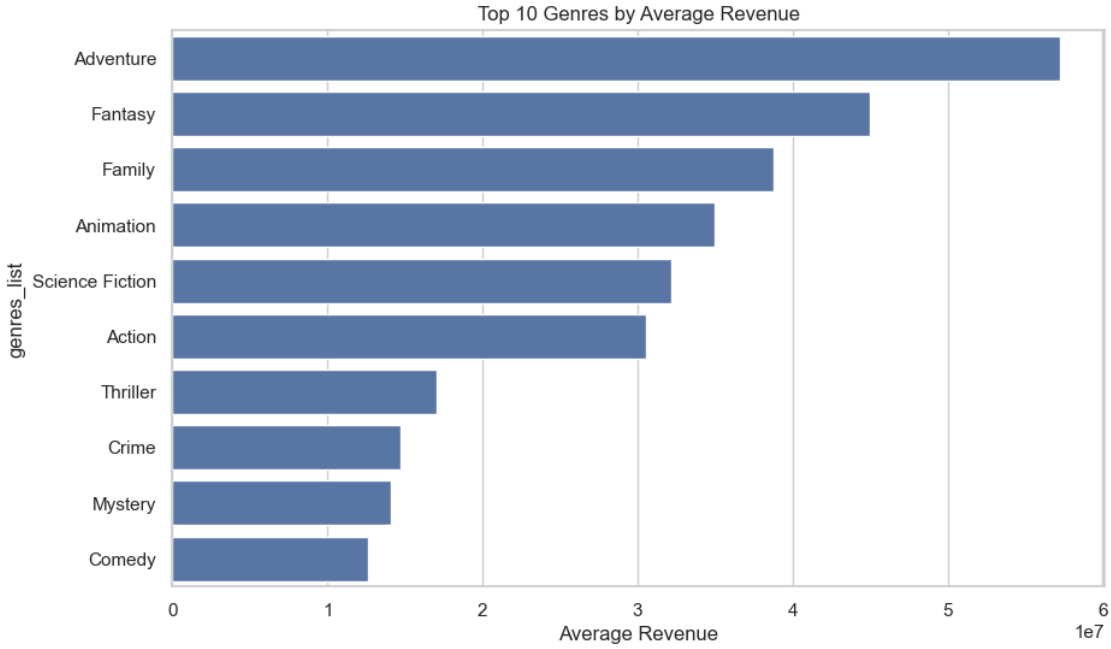
[31]:
```
import ast

# Parse genres
def extract_genres(x):
    try:
```

```
        return [i['name'] for i in ast.literal_eval(x)]
    except:
        return []
movies['genres_list'] = movies['genres'].apply(extract_genres)

# Explode genres
genres_df = movies.explode('genres_list')
genres_revenue = genres_df.groupby('genres_list')['revenue'].mean().
 ↪sort_values(ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=genres_revenue.values, y=genres_revenue.index)
plt.title("Top 10 Genres by Average Revenue")
plt.xlabel("Average Revenue")
plt.show()
```



9.1 Key Insights

- The number of movies produced has increased sharply after 1980.
- Higher budget movies generally make more revenue, but there are many exceptions.
- Animation and Adventure genres tend to earn higher revenues compared to Drama or Horror.
- Median runtime for movies is about 100 minutes.

[ ]: