phase2project

August 6, 2025

1 Phase 2 Project: Movie Industry Exploratory Data Analysis (EDA)

1.1 Project Overview

In this project, I explore data from various movie industry sources to uncover insights that will guide a newly launched movie studio. With the increasing trend of major companies creating original video content, our company is entering the market but lacks experience in filmmaking. The mission is to analyze what types of films perform best at the box office and translate those findings into clear, data-driven business recommendations.

The analysis draws from multiple datasets sourced from platforms such as IMDB, Box Office Mojo, Rotten Tomatoes, TheMovieDB, and The Numbers. Given the varied formats and origins of these datasets, part of the task involves effective data cleaning and merging before meaningful analysis can begin.

1.2 Business Context

The stakeholders for this project seek strategic insights to help decide:

- What kinds of movies (genres, runtimes, budgets) tend to perform best at the box office?
- Which release periods are most favorable?
- What characteristics are associated with successful films?

Since **producing movies is costly**, it is essential to base investment decisions on solid data.

1.3 Project Objectives

1. **Explore and Clean** multiple datasets related to movie performance.
2. **Visualize and analyze** key factors that influence box office success.
3. **Deliver 3 actionable recommendations** to inform movie production decisions.

1.4 Project Plan

This notebook is structured as follows:

1. **Data Loading & Overview**

- Explore each dataset (CSV,SQLite)

- Understand the schema and content

2. **Insights & Recommendations**
   - Summarize key findings

   - Translate insights into business actions

3. **Conclusion**

---

2 Importing the basic libraries...

[113]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
```

3 Loading the Data

[62]:
```
conn = sqlite3.connect("im.db")
csv_gross = pd.read_csv("bom.movie_gross.csv.gz")
csv_budgets = pd.read_csv("tn.movie_budgets.csv.gz")
```

[63]:
```
dfone = pd.read_sql("""SELECT movie_id, start_year,runtime_minutes,genres FROM␣
↪movie_basics""",conn)
dftwo = pd.read_sql("""SELECT * FROM movie_ratings""",conn)
```

4 Quick Look at the different tables !

[64]:
```
dfone
```

[64]:

	movie_id	start_year	runtime_minutes	genres
0	tt0063540	2013	175.0	Action,Crime,Drama
1	tt0066787	2019	114.0	Biography,Drama
2	tt0069049	2018	122.0	Drama
3	tt0069204	2018	NaN	Comedy,Drama
4	tt0100275	2017	80.0	Comedy,Drama,Fantasy
…	…	…	…	…
146139	tt9916538	2019	123.0	Drama
146140	tt9916622	2015	NaN	Documentary
146141	tt9916706	2013	NaN	Comedy
146142	tt9916730	2017	116.0	None
146143	tt9916754	2013	NaN	Documentary

```
[146144 rows x 4 columns]
```

[65]:
```python
dftwo
```

[65]:
```
        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]
```

[66]:
```python
csv_gross
```

[66]:
```
                                             title      studio  domestic_gross  \
0                                      Toy Story 3          BV     415000000.0
1                       Alice in Wonderland (2010)          BV     334200000.0
2     Harry Potter and the Deathly Hallows Part 1          WB     296000000.0
3                                        Inception          WB     292600000.0
4                              Shrek Forever After        P/DW     238700000.0
…                                              …         …              …
3382                                     The Quake       Magn.          6200.0
3383                   Edward II (2018 re-release)          FM          4800.0
3384                                      El Pacto        Sony          2500.0
3385                                      The Swan  Synergetic          2400.0
3386                               An Actor Prepares       Grav.          1700.0

     foreign_gross  year
0        652000000  2010
1        691300000  2010
2        664300000  2010
3        535700000  2010
4        513900000  2010
…              …     …
3382           NaN  2018
3383           NaN  2018
3384           NaN  2018
3385           NaN  2018
3386           NaN  2018
```

```
[3387 rows x 5 columns]
```

[67]:
```
csv_budgets
```

[67]:
```
      id  release_date                                        movie  \
0      1  Dec 18, 2009                                       Avatar
1      2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides
2      3   Jun 7, 2019                                 Dark Phoenix
3      4   May 1, 2015                      Avengers: Age of Ultron
4      5  Dec 15, 2017            Star Wars Ep. VIII: The Last Jedi
…     ..           …                                            …
5777  78  Dec 31, 2018                                       Red 11
5778  79   Apr 2, 1999                                    Following
5779  80  Jul 13, 2005                Return to the Land of Wonders
5780  81  Sep 29, 2015                         A Plague So Pleasant
5781  82   Aug 5, 2005                            My Date With Drew

     production_budget domestic_gross worldwide_gross
0         $425,000,000   $760,507,625  $2,776,345,279
1         $410,600,000   $241,063,875  $1,045,663,875
2         $350,000,000    $42,762,350    $149,762,350
3         $330,600,000   $459,005,868  $1,403,013,963
4         $317,000,000   $620,181,382  $1,316,721,747
…                  …            …            …
5777            $7,000             $0              $0
5778            $6,000        $48,482        $240,495
5779            $5,000         $1,338          $1,338
5780            $1,400             $0              $0
5781            $1,100       $181,041        $181,041

[5782 rows x 6 columns]
```

5 Let's The Cleaning Begin !

[68]:
```
#dfone_cleaning

# Strip whitespace from column names
dfone.columns = dfone.columns.str.strip()

# Drop duplicates
dfone.drop_duplicates(inplace=True)

#  Clean start_year
dfone['start_year'] = pd.to_numeric(dfone['start_year'], errors='coerce')
dfone = dfone[dfone['start_year'].notnull()]  # Remove rows with missing year
dfone['start_year'] = dfone['start_year'].astype(int)
```

```
dfone = dfone[(dfone['start_year'] >= 2021 & (dfone['start_year'] <= 2025))]  #␣
↪keep valid years only

# Clean runtime_minutes
dfone['runtime_minutes'] = pd.to_numeric(dfone['runtime_minutes'],␣
↪errors='coerce')
dfone = dfone[dfone['runtime_minutes'].notnull()]  # remove rows with invalid␣
↪runtimes
dfone = dfone[dfone['runtime_minutes'] > 0]  # remove zero or negative runtimes
dfone = dfone[dfone['runtime_minutes'] < 500]  # filter unrealistic runtimes

# Clean genres
dfone['genres'] = dfone['genres'].fillna('Unknown')
dfone['genres'] = dfone['genres'].str.strip()
dfone['genres'] = dfone['genres'].str.lower()

#Drop rows with completely missing values (just in case)
dfone.dropna(how='all', inplace=True)

#Reset index after cleaning
dfone.reset_index(drop=True, inplace=True)
```

[69]:
```
#dftwo_cleaning

# Strip whitespace from column names
dftwo.columns = dftwo.columns.str.strip()

# Drop duplicates
dftwo.drop_duplicates(inplace=True)

# Clean average_rating
dftwo['averagerating'] = pd.to_numeric(dftwo['averagerating'], errors='coerce')
dftwo = dftwo[dftwo['averagerating'].notnull()]
dftwo = dftwo[(dftwo['averagerating'] >= 8) & (dftwo['averagerating'] <= 10)]

# Clean numvotes
dftwo['numvotes'] = pd.to_numeric(dftwo['numvotes'], errors='coerce')
dftwo = dftwo[dftwo['numvotes'].notnull()]
dftwo = dftwo[dftwo['numvotes'] >= 0]  # remove negative vote counts

# Drop rows with completely missing values (just in case)
dftwo.dropna(how='all', inplace=True)

# Reset index
dftwo.reset_index(drop=True, inplace=True)
```

[102]:
```
#csv_budget_cleaning

# Strip whitespace from column names
csv_budgets.columns = csv_budgets.columns.str.strip()

#  Drop duplicates
csv_budgets.drop_duplicates(inplace=True)

# Parse release_date into datetime
csv_budgets['release_date'] = pd.to_datetime(csv_budgets['release_date'],␣
↪errors='coerce')
csv_budgets = csv_budgets[csv_budgets['release_date'].notnull()]  # remove␣
↪invalid dates

# Drop rows with completely missing data
csv_budgets.dropna(how='all', inplace=True)

#Remove dollar signs and commas, then convert to numeric
for col in ['production_budget', 'domestic_gross', 'worldwide_gross']:
    csv_budgets[col] = csv_budgets[col].astype(str).str.replace(r'[$,]', '',␣
↪regex=True)
    csv_budgets[col] = pd.to_numeric(csv_budgets[col], errors='coerce')

# Reset index
csv_budgets.reset_index(drop=True, inplace=True)
```

[106]:
```
# csv_gross_cleaning

# Drop duplicates
csv_gross.drop_duplicates(inplace=True)

# Clean year
csv_gross = csv_gross[csv_gross['year'].notnull()]
csv_gross = csv_gross[(csv_gross['year'] >= 2018) & (csv_gross['year'] <=␣
↪2025)]  # valid range

#Drop rows with all values missing (just in case)
csv_gross.dropna(inplace=True)

# Clean and convert gross columns
for col in ['domestic_gross', 'foreign_gross']:
    csv_gross[col] = csv_gross[col].astype(str).str.replace(r'[$,]', '',␣
↪regex=True)
    csv_gross[col] = pd.to_numeric(csv_gross[col], errors='coerce')

# Reset index
csv_gross.reset_index(drop=True, inplace=True)
```

[72]:
```
dfone
```

[72]:
```
         movie_id  start_year  runtime_minutes                genres
0       tt0063540        2013            175.0    action,crime,drama
1       tt0066787        2019            114.0       biography,drama
2       tt0069049        2018            122.0                 drama
3       tt0100275        2017             80.0  comedy,drama,fantasy
4       tt0111414        2018             75.0                comedy
…             …           …               …                     …
114345  tt9916170        2019             51.0                 drama
114346  tt9916186        2017             84.0           documentary
114347  tt9916190        2019             90.0        drama,thriller
114348  tt9916538        2019            123.0                 drama
114349  tt9916730        2017            116.0               unknown

[114350 rows x 4 columns]
```

[73]:
```
dftwo
```

[73]:
```
        movie_id  averagerating  numvotes
0     tt10356526            8.3        31
1     tt10384606            8.9       559
2      tt1193623            8.0         5
3      tt1326743            8.4        21
4      tt1403990            8.5        31
…          …               …       …
9443   tt9219848            8.7        18
9444   tt9367004            8.2         5
9445   tt9590776            9.2        37
9446   tt9768966            8.6        27
9447   tt9805820            8.1        25

[9448 rows x 3 columns]
```

[103]:
```
csv_budgets
```

[103]:
```
      id release_date                                       movie  \
0      1   2009-12-18                                      Avatar
1      2   2011-05-20  Pirates of the Caribbean: On Stranger Tides
2      3   2019-06-07                                 Dark Phoenix
3      4   2015-05-01                      Avengers: Age of Ultron
4      5   2017-12-15            Star Wars Ep. VIII: The Last Jedi
…    ..          …                                           …
5777  78   2018-12-31                                      Red 11
5778  79   1999-04-02                                   Following
5779  80   2005-07-13                 Return to the Land of Wonders
5780  81   2015-09-29                          A Plague So Pleasant
```

```
5781  82   2005-08-05                            My Date With Drew

      production_budget  domestic_gross  worldwide_gross
0             425000000       760507625       2776345279
1             410600000       241063875       1045663875
2             350000000        42762350        149762350
3             330600000       459005868       1403013963
4             317000000       620181382       1316721747
…                     …               …                …
5777               7000               0                0
5778               6000           48482           240495
5779               5000            1338             1338
5780               1400               0                0
5781               1100          181041           181041

[5782 rows x 6 columns]
```

[107]:
```
csv_gross
```

[107]:
```
                                                title       studio  \
0                              Avengers: Infinity War           BV
1                                       Black Panther           BV
2                      Jurassic World: Fallen Kingdom         Uni.
3                                        Incredibles 2           BV
4                                             Aquaman           WB
..                                                  …            …
168                                    I Still See You          LGF
169                             The Catcher Was a Spy          IFC
170                                          Time Freak   Grindstone
171  Reign of Judges: Title of Liberty - Concept Short  Darin Southa
172            Antonio Lopez 1970: Sex Fashion & Disco           FM

     domestic_gross  foreign_gross  year
0       678800000.0         1369.5  2018
1       700100000.0    646900000.0  2018
2       417700000.0    891800000.0  2018
3       608600000.0    634200000.0  2018
4       335100000.0    812700000.0  2018
..                …              …     …
168          1400.0      1500000.0  2018
169        725000.0       229000.0  2018
170         10000.0       256000.0  2018
171         93200.0         5200.0  2018
172         43200.0        30000.0  2018

[173 rows x 5 columns]
```

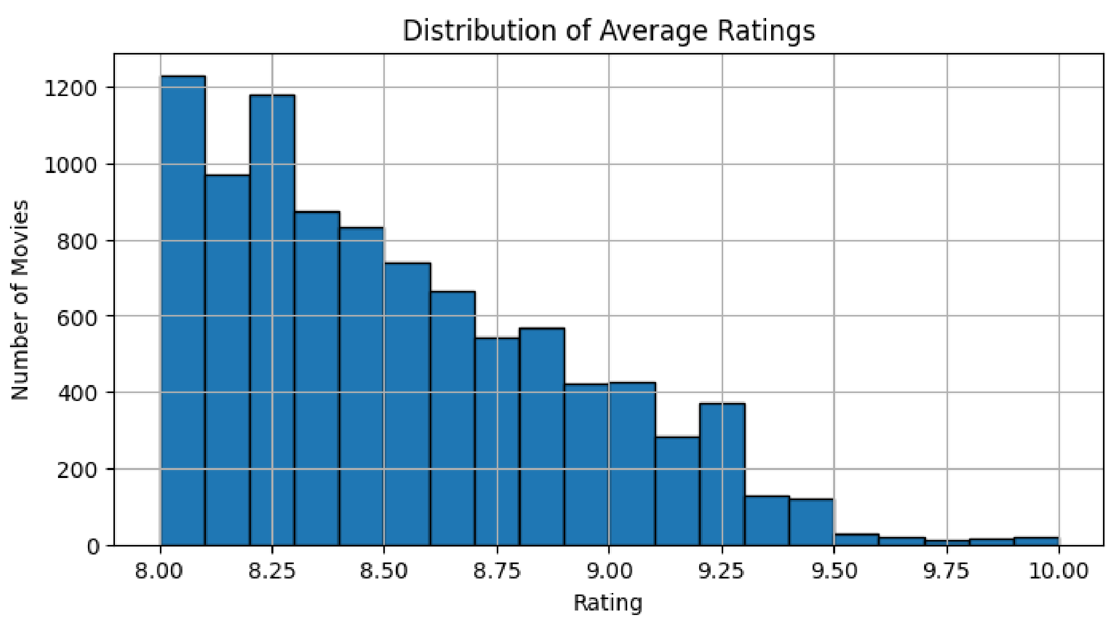
6 Analysis and Insights

[77]:
```
#dfone

top_5_by_votes = dftwo.sort_values(by="numvotes",ascending = False).head(5)
top_5_by_votes
```

[77]:

	movie_id	averagerating	numvotes
8106	tt1375666	8.8	1841066
1066	tt1345836	8.4	1387769
3132	tt0816692	8.6	1299334
4783	tt1853728	8.4	1211405
6177	tt0848228	8.1	1183655

[79]:
```
dftwo['averagerating'].hist(bins=20, edgecolor='black', figsize=(8, 4))
plt.title("Distribution of Average Ratings")
plt.xlabel("Rating")
plt.ylabel("Number of Movies")
plt.show()
```



[87]:
```
bins = [7, 8, 9, 10]
labels = ['7-8', '8-9', '9-10']
dftwo['rating_class'] = pd.cut(dftwo['averagerating'], bins=bins, labels=labels)

rating_dist = dftwo['rating_class'].value_counts().sort_index()
print(pd.DataFrame(rating_dist))
```

```
# Insights : Most ratings go to "[7-8[ class"
```

```
              count
rating_class
7-8            1228
8-9            7220
9-10           1000
```

[93]:
```
# Top 10 Most Frequent Genres

# Explode the genre list
df_genres = dfone.copy()
df_genres['genres'] = df_genres['genres'].str.split(',')
df_genres = df_genres.explode('genres')

# Count most frequent genres
top_genres = df_genres['genres'].value_counts().head(10)
print(pd.DataFrame(top_genres))

#  Insight: These genres are released the most so that indicates market demand␣
↪also production interest.
```

```
             count
genres
documentary  43559
drama        41597
comedy       20838
thriller      9719
horror        8605
biography     8240
action        8108
romance       7783
crime         5850
history       5787
```

[92]:
```
# Number of Movies per Year (Recent Activity)

yearly_release = dfone['start_year'].value_counts().sort_index().tail(10)
print(pd.DataFrame(yearly_release))

# Insight: years with high release counts = stronger market momentum. Ideal␣
↪timeframes for investment.
```

```
            count
start_year
2013        12302
2014        12963
2015        13246
2016        13506
```

```
2017        13457
2018        12208
2019         4500
2020           82
2021            4
2022            3
```

[94]:
```
# Average Runtime by Genre

avg_runtime_by_genre = df_genres.groupby('genres')['runtime_minutes'].mean().
    ↪sort_values(ascending=False).head(10)
print(pd.DataFrame(avg_runtime_by_genre))

#  Insight: Longer runtimes often correlate with more serious or expensive␣
    ↪productions.
```

```
           runtime_minutes
genres
game-show       117.000000
romance         100.219710
action           99.703379
crime            95.421538
musical          94.333919
thriller         94.279967
comedy           93.316345
drama            93.206073
mystery          91.525718
fantasy          91.063330
```

[111]:
```
# Goal: Identify what's making the most money

#(Sum of domestic and foreign gross)

csv_gross['total_gross'] = csv_gross['domestic_gross'] +␣
    ↪csv_gross['foreign_gross']

top_earners = csv_gross.sort_values(by='total_gross', ascending=False)

#Display as Billions (Short Format)
csv_gross['total_gross_billion'] = csv_gross['total_gross'] / 1_000_000_000
print(pd.DataFrame(csv_gross[['title', 'total_gross_billion']]).head(10))


#  Insight: These are top-grossing hits - proven market success, safe to␣
    ↪reinvest or continue series.
```

```
                                           title  total_gross_billion
0                          Avengers: Infinity War             0.678801
```

```
1                        Black Panther          1.347000
2        Jurassic World: Fallen Kingdom          1.309500
3                          Incredibles 2          1.242800
4                                Aquaman          1.147800
5                      Bohemian Rhapsody          0.903600
6                           Venom (2018)          0.855000
7            Mission: Impossible - Fallout          0.791200
8                             Deadpool 2          0.779000
9   Fantastic Beasts: The Crimes of Grindelwald          0.653700
```

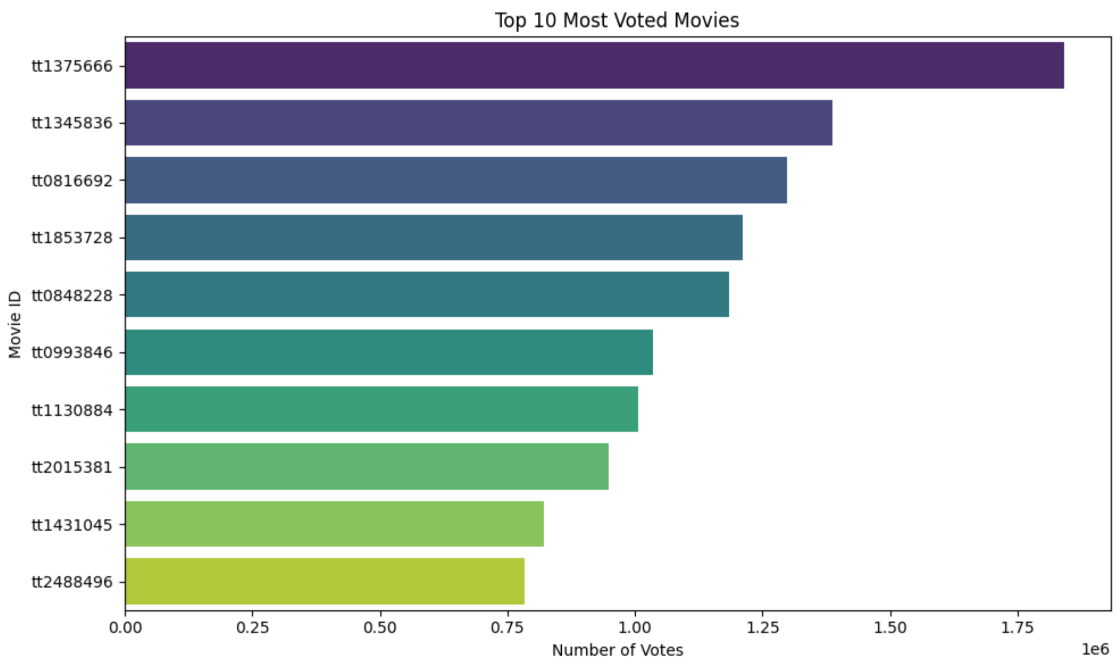
[112]:

```
# Goal: Find the best ROI (Profitability)

# Compute profit and ROI
csv_budgets['profit'] = csv_budgets['worldwide_gross'] -␣
 ↪csv_budgets['production_budget']
csv_budgets['roi'] = csv_budgets['profit'] / csv_budgets['production_budget']

# Sort by ROI (efficiency of investment)
top_roi = csv_budgets.sort_values(by='roi', ascending=False)
print(pd.DataFrame(top_roi[['movie', 'production_budget', 'worldwide_gross',␣
 ↪'profit', 'roi']]).head(10))

#  Insight: These are highly profitable even on low or mid budget - great for␣
 ↪scaling up similar projects.
```

```
                        movie  production_budget  worldwide_gross     profit  \
5745              Deep Throat              25000         45000000   44975000
5613                  Mad Max             200000         99750000   99550000
5492       Paranormal Activity             450000        194183034  193733034
5679              The Gallows             100000         41656474   41556474
5406    The Blair Witch Project             600000        248300000  247700000
5709              Super Size Me              65000         22233808   22168808
5346                    Bambi             858000        268000000  267142000
5773              El Mariachi               7000          2041928    2034928
5676   Night of the Living Dead             114000         30087064   29973064
5210                    Rocky            1000000        225000000  224000000

              roi
5745  1799.000000
5613   497.750000
5492   430.517853
5679   415.564740
5406   412.833333
5709   341.058585
5346   311.354312
5773   290.704000
5676   262.921614
5210   224.000000
```

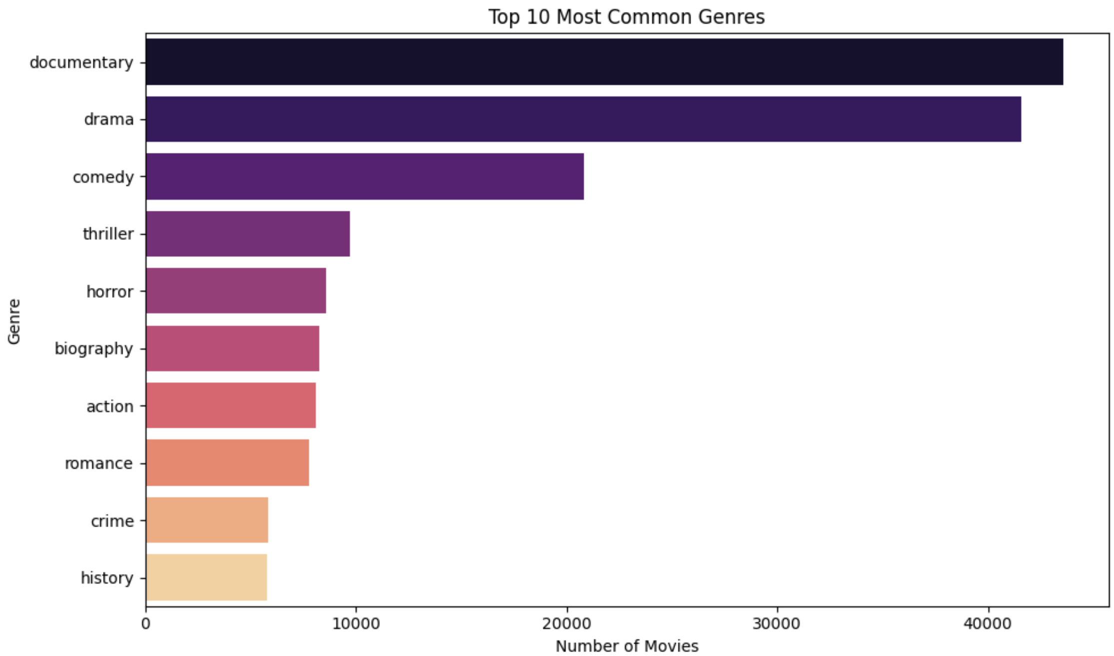
7 Some Visuals

[123]:
```
top_voted = dftwo.sort_values(by='numvotes', ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x='numvotes', y='movie_id', data=top_voted, palette='viridis', hue␣
↪= 'movie_id', legend = False)
plt.title('Top 10 Most Voted Movies')
plt.xlabel('Number of Votes')
plt.ylabel('Movie ID')
plt.tight_layout()
plt.show()
```

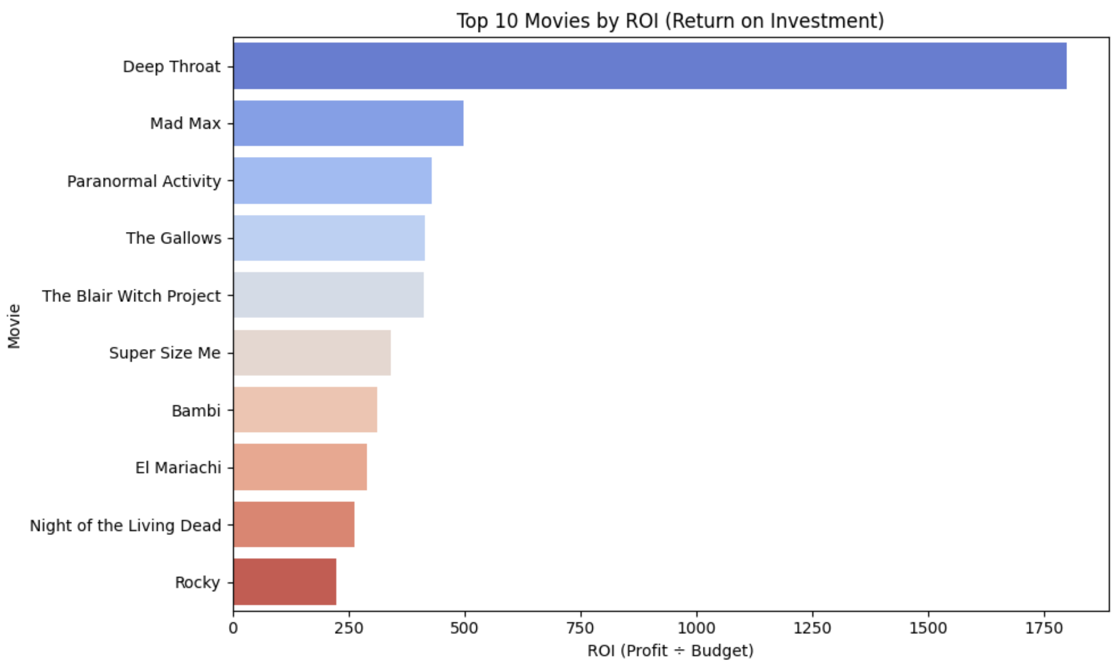


[121]:
```
df_genres = dfone.copy()
df_genres['genres'] = df_genres['genres'].str.split(',')
df_genres = df_genres.explode('genres')

top_genres = df_genres['genres'].value_counts().head(10).reset_index()
top_genres.columns = ['genre', 'count']


plt.figure(figsize=(10, 6))
sns.barplot(x='count', y='genre', data=top_genres, palette='magma', hue =␣
↪'genre', legend = False)
plt.title('Top 10 Most Common Genres')
```

```
plt.xlabel('Number of Movies')
plt.ylabel('Genre')
plt.tight_layout()
plt.show()
```



[122]:
```
top_roi = csv_budgets.sort_values(by='roi', ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x='roi', y='movie', data=top_roi, palette='coolwarm', hue =␣
 ↪'movie', legend = False)
plt.title('Top 10 Movies by ROI (Return on Investment)')
plt.xlabel('ROI (Profit ÷ Budget)')
plt.ylabel('Movie')
plt.tight_layout()
plt.show()
```



8 Overall Analysis & Recommandation

9 Kreyol

9.1 Aksyon I

9.1.1 Nan dataframe "dfone" la, nou te fe yon analiz ki montre nou top film selon vot, kidonk nou deside kenbe 4 film ki gen yon averagerating ant 8.4 et 8.8 donk movie_id : tt137666/tt1345836/tt0816692/tt1853728.

9.2 Aksyon II

9.2.1 Answit, nan analiz nou fe toujou nou te arive remake ke gen "genres" ki pi popile pase lot, yo se : "documentary", "drama" and "comedy" kidonk nou rekonmande biznis la envesti plis nan genres sa yo.

9.3 Aksyon III

9.3.1 dapre rechech ke nou efektye, nou arive konen ke toutotan dire film la "runtime_duration" pi long , se otan kou pou pwodiksyon an enpotan. piske nou fek ap debite nan "industry" an , nou ta sipoze enveti egalman nan film ki gen yon dire mwayen... ant 91 et 94, poun ka divesifye sous revni yo !

9.4 An denye lye, nou panse ke gen 4 fim ke nap konsantre anpil efo sou yo, ke nan repwodiksyon yo oswa nan envesti nan kontinwite yo, yo se : Black Panther, Fallen Kingdom, Incredibles 2 epi Aquaman. Fim sa yo jenere yon revni total de plis ke 1 bilyon dola vet !!