recommendation-system

May 26, 2024

1 Recommendation System

1.0.1 1.Data Collection

Import the necessary libraies

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

Importing the dataset as DataFrame using pandas

[4]:
```
movies = pd.read_csv("//content//drive//MyDrive//CSV_Files//Movies.csv")
credit = pd.read_csv("//content//drive//MyDrive//CSV_Files//Credit.csv")
```

Overview of the dataset

movies dataset

[5]:
```
movies.head(2)
```

[5]:
```
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "…

                                       homepage     id  \
0                   http://www.avatarmovie.com/  19995
1  http://disney.go.com/disneypictures/pirates/    285

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":…                 en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na…                 en

                             original_title  \
```

```
0                                    Avatar
1  Pirates of the Caribbean: At World's End

                                        overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di…  150.437577
1  Captain Barbossa, long believed to be dead, ha…  139.082615

                                production_companies  \
0  [{"name": "Ingenious Film Partners", "id": 289…
1  [{"name": "Walt Disney Pictures", "id": 2}, {"…

                                production_countries release_date     revenue  \
0  [{"iso_3166_1": "US", "name": "United States o…   2009-12-10  2787965087
1  [{"iso_3166_1": "US", "name": "United States o…   2007-05-19   961000000

   runtime                                     spoken_languages    status  \
0    162.0  [{"iso_639_1": "en", "name": "English"}, {"iso…  Released
1    169.0           [{"iso_639_1": "en", "name": "English"}]  Released

                                      tagline  \
0                    Enter the World of Pandora.
1  At the end of the world, the adventure begins.

                                    title  vote_average  vote_count
0                                  Avatar           7.2       11800
1  Pirates of the Caribbean: At World's End           6.9        4500
```

[6]:
```
movies.tail(2)
```

[6]:
```
      budget                                genres  \
4801       0                                    []
4802       0  [{"id": 99, "name": "Documentary"}]

                        homepage      id  \
4801  http://shanghaicalling.com/  126186
4802                          NaN   25975

                                          keywords original_language  \
4801                                            []                en
4802  [{"id": 1523, "name": "obsession"}, {"id": 224…                en

        original_title                                           overview  \
4801   Shanghai Calling  When ambitious New York attorney Sam is sent t…
4802  My Date with Drew  Ever since the second grade when he first saw …

      popularity                                production_companies  \
4801    0.857008                                                []
```

```
4802    1.929883  [{"name": "rusty bear entertainment", "id": 87…

                                  production_countries release_date  revenue  \
4801  [{"iso_3166_1": "US", "name": "United States o…   2012-05-03        0
4802  [{"iso_3166_1": "US", "name": "United States o…   2005-08-05        0

      runtime                           spoken_languages    status  \
4801     98.0  [{"iso_639_1": "en", "name": "English"}]  Released
4802     90.0  [{"iso_639_1": "en", "name": "English"}]  Released

                      tagline              title  vote_average  vote_count
4801  A New Yorker in Shanghai   Shanghai Calling           5.7           7
4802                       NaN  My Date with Drew           6.3          16
```

Info of the movies dataset

[7]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

Describe the movies dataset

[8]:
```
movies.describe()
```

[8]:
```
              budget             id   popularity       revenue      runtime  \
count   4.803000e+03    4803.000000  4803.000000  4.803000e+03  4801.000000
mean    2.904504e+07   57165.484281    21.492301  8.226064e+07   106.875859
std     4.072239e+07   88694.614033    31.816650  1.628571e+08    22.611935
min     0.000000e+00       5.000000     0.000000  0.000000e+00     0.000000
25%     7.900000e+05    9014.500000     4.668070  0.000000e+00    94.000000
50%     1.500000e+07   14629.000000    12.921594  1.917000e+07   103.000000
75%     4.000000e+07   58610.500000    28.313505  9.291719e+07   118.000000
max     3.800000e+08  459488.000000   875.581305  2.787965e+09   338.000000

       vote_average    vote_count
count   4803.000000   4803.000000
mean       6.092172    690.217989
std        1.194612   1234.585891
min        0.000000      0.000000
25%        5.600000     54.000000
50%        6.200000    235.000000
75%        6.800000    737.000000
max       10.000000  13752.000000
```

Shape of movies dataset

[9]:
```
movies.shape
```

[9]:
```
(4803, 20)
```

credit dataset

[10]:
```
credit.head(2)
```

[10]:
```
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End

                                                 cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "…
1  [{"cast_id": 4, "character": "Captain Jack Spa…

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de…
1  [{"credit_id": "52fe4232c3a36847f800b579", "de…
```

[11]:
```
credit.tail(2)
```

[11]:
```
      movie_id              title  \
4801    126186   Shanghai Calling
```

```
4802     25975  My Date with Drew

                                                    cast  \
4801  [{"cast_id": 3, "character": "Sam", "credit_id…
4802  [{"cast_id": 3, "character": "Herself", "credi…

                                                    crew
4801  [{"credit_id": "52fe4ad9c3a368484e16a36b", "de…
4802  [{"credit_id": "58ce021b9251415a390165d9", "de…
```

Info of credit dataset

[12]:
```
credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  4803 non-null   int64
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

Describe the credit dataset

[13]:
```
credit.describe()
```

[13]:
```
            movie_id
count    4803.000000
mean    57165.484281
std     88694.614033
min         5.000000
25%      9014.500000
50%     14629.000000
75%     58610.500000
max    459488.000000
```

Shape of credit dataset

[14]:
```
credit.shape
```

[14]:
```
(4803, 4)
```

1.0.2 2.Preprocessing the dataset

Check Null values in movies dataset

[15]:
```
movies.isnull().sum()
```

[15]:
```
budget                     0
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
dtype: int64
```

Droping the unnecessary column in movies dataset

[16]:
```
movies.drop(columns = ["tagline","homepage","status"],inplace = True)
```

[17]:
```
movies.isnull().sum()
```

[17]:
```
budget                  0
genres                  0
id                      0
keywords                0
original_language       0
original_title          0
overview                3
popularity              0
production_companies    0
production_countries    0
release_date            1
revenue                 0
runtime                 2
spoken_languages        0
title                   0
vote_average            0
vote_count              0
```

```
dtype: int64
```

Filling the null values in overview column in movies dataset

[18]:
```
movies["overview"].isnull().sum()
```

[18]:
```
3
```

[19]:
```
movies["overview"] = movies["overview"].fillna("")
```

[20]:
```
movies["overview"].isnull().sum()
```

[20]:
```
0
```

Droping the null value in the release date column

[21]:
```
movies["release_date"].isnull().sum()
```

[21]:
```
1
```

[22]:
```
release_date_values = movies[movies["release_date"].isnull()].index
```

[23]:
```
movies.drop(release_date_values , inplace = True )
```

[24]:
```
movies["release_date"].isnull().sum()
```

[24]:
```
0
```

Droping the null value in the run time column

[25]:
```
movies["runtime"].isnull().sum()
```

[25]:
```
2
```

[26]:
```
runtime_values = movies[movies["runtime"].isnull()].index
```

[27]:
```
movies.drop(runtime_values , inplace = True)
```

[28]:
```
movies["runtime"].isnull().sum()
```

[28]:
```
0
```

[29]:
```
movies.isnull().sum()
```

[29]:
```
budget                  0
genres                  0
id                      0
keywords                0
```

```
original_language       0
original_title          0
overview                0
popularity              0
production_companies    0
production_countries    0
release_date            0
revenue                 0
runtime                 0
spoken_languages        0
title                   0
vote_average            0
vote_count              0
dtype: int64
```

Check Null values in credit dataset

[30]:
```
credit.isnull().sum()
```

[30]:
```
movie_id    0
title       0
cast        0
crew        0
dtype: int64
```

1.0.3 3.Data Validation

Validate the movies dataset

[31]:
```
movies.count()
```

[31]:
```
budget                  4800
genres                  4800
id                      4800
keywords                4800
original_language       4800
original_title          4800
overview                4800
popularity              4800
production_companies    4800
production_countries    4800
release_date            4800
revenue                 4800
runtime                 4800
spoken_languages        4800
title                   4800
vote_average            4800
vote_count              4800
```

```
dtype: int64
```

Validate the credit dataset

[32]:
```
credit.count()
```

[32]:
```
movie_id    4803
title       4803
cast        4803
crew        4803
dtype: int64
```

4.Data Formatting

Data formatting in the movies dataset

Columns in movies dataset required for content based recommendation

1.genres

2.overview

3.keywords

Convert all columns with revelant content into list of strings

Datatype of overview column

[33]:
```
type(movies.loc[0]["overview"])
```

[33]:
```
str
```

Convert the overview column into list of strings

[34]:
```
movies["overview"] = movies["overview"].apply(lambda x : x.split())
```

[35]:
```
type(movies.loc[0]["overview"])
```

[35]:
```
list
```

Datatype of geners column

[36]:
```
type(movies.loc[0]["genres"])
```

[36]:
```
str
```

Datatype of keywords column

[37]:
```
type(movies.loc[0]["keywords"])
```

[37]:
```
str
```

Datatype of production_countries column

[38]:
```
type(movies.loc[0]["production_countries"])
```

[38]:
```
str
```

Datatype of production_companies column

[39]:
```
type(movies.loc[0]["production_companies"])
```

[39]:
```
str
```

Datatype of spoken_languages column

[40]:
```
type(movies.loc[0]["spoken_languages"])
```

[40]:
```
str
```

Function to convert the columns into list datatype in movies dataset

[41]:
```
import ast
```

[42]:
```
def convert(text) :
    l = []
    for i in ast.literal_eval(text):
        l.append(i["name"])

    return l
```

[43]:
```
movies["genres"] = movies["genres"].apply(convert)
movies["keywords"] = movies["keywords"].apply(convert)
movies["production_countries"] = movies["production_countries"].apply(convert)
movies["production_companies"] = movies["production_companies"].apply(convert)
movies["spoken_languages"] = movies["spoken_languages"].apply(convert)
```

Overview of the movies dataset after conversion of columns into suitable datatypes

[44]:
```
movies.head(2)
```

[44]:
```
      budget                                     genres     id  \
0  237000000  [Action, Adventure, Fantasy, Science Fiction]  19995
1  300000000                [Adventure, Fantasy, Action]    285

                                        keywords original_language  \
```

```
0  [culture clash, future, space war, space colon…                en
1  [ocean, drug abuse, exotic island, east india …                en

                            original_title  \
0                                   Avatar
1  Pirates of the Caribbean: At World's End

                                          overview  popularity  \
0  [In, the, 22nd, century,, a, paraplegic, Marin…  150.437577
1  [Captain, Barbossa,, long, believed, to, be, d…  139.082615

                               production_companies  \
0  [Ingenious Film Partners, Twentieth Century Fo…
1  [Walt Disney Pictures, Jerry Bruckheimer Films…

                        production_countries release_date     revenue  \
0  [United States of America, United Kingdom]   2009-12-10  2787965087
1                  [United States of America]   2007-05-19   961000000

   runtime    spoken_languages                                    title  \
0    162.0  [English, Español]                                   Avatar
1    169.0           [English]  Pirates of the Caribbean: At World's End

   vote_average  vote_count
0           7.2       11800
1           6.9        4500
```

Data formatting in the credit dataset

Columns in credit dataset required for content based recommendation

1.cast

2.crew

Datatype of cast column

[45]:
```
type(credit.loc[0]["cast"])
```

[45]:
```
str
```

Datatype of crew column

[46]:
```
type(credit.loc[0]["crew"])
```

[46]:
```
str
```

Function to convert the columns into list datatype in credit dataset

[47]:
```
def convert_cast(text) :
    l = []
    count = 0
    for i in ast.literal_eval(text):
        count += 1
        if count <= 10 :
            l.append(i["name"])

    return l

def convert_crew(text) :
    l = []
    for i in ast.literal_eval(text):
        if i['job'] == "Director" or  i["job"] == "Original Music Composer" or␣
 ↪i["job"] == "Editor" or i["job"] == "Writer":
            l.append(i["name"])
            break
    return l
```

[48]:
```
credit["cast"] = credit["cast"].apply(convert_cast)
credit["crew"] = credit["crew"].apply(convert_crew)
```

Overview of the credit dataset after converting the columns into suitable datatype

[49]:
```
credit.head(2)
```

[49]:
```
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End

                                                cast                  crew
0  [Sam Worthington, Zoe Saldana, Sigourney Weave…  [Stephen E. Rivkin]
1  [Johnny Depp, Orlando Bloom, Keira Knightley, …     [Gore Verbinski]
```

Conversion of white spaces to avoid error

Function to convert the white spaces

[50]:
```
def convert_whitespaces(text) :
    l = []
    for i in text :
        l.append(i.replace(" ",""))
    return l
```

[51]:
```
movies["genres"] = movies["genres"].apply(convert_whitespaces)
movies["keywords"] = movies["genres"].apply(convert_whitespaces)
movies["overview"] = movies["overview"].apply(convert_whitespaces)
```

```
credit["cast"] = credit["cast"].apply(convert_whitespaces)
credit["crew"] = credit["crew"].apply(convert_whitespaces)
```

[52]:
```
credit["cast"] = credit["cast"].apply(convert_whitespaces)
credit["crew"] = credit["crew"].apply(convert_whitespaces)
```

[53]:
```
movies.head(2)
```

[53]:
```
      budget                                             genres     id  \
0  237000000  [Action, Adventure, Fantasy, ScienceFiction]  19995
1  300000000                  [Adventure, Fantasy, Action]    285

                                    keywords original_language  \
0  [Action, Adventure, Fantasy, ScienceFiction]                en
1                  [Adventure, Fantasy, Action]                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End

                                          overview  popularity  \
0  [In, the, 22nd, century,, a, paraplegic, Marin…  150.437577
1  [Captain, Barbossa,, long, believed, to, be, d…  139.082615

                               production_companies  \
0  [Ingenious Film Partners, Twentieth Century Fo…
1  [Walt Disney Pictures, Jerry Bruckheimer Films…

                        production_countries release_date     revenue  \
0  [United States of America, United Kingdom]   2009-12-10  2787965087
1                  [United States of America]   2007-05-19   961000000

   runtime    spoken_languages                                    title  \
0    162.0  [English, Español]                                   Avatar
1    169.0           [English]  Pirates of the Caribbean: At World's End

   vote_average  vote_count
0           7.2       11800
1           6.9        4500
```

[54]:
```
credit.head(2)
```

[54]:
```
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End

                                                cast                crew
```

```
0  [SamWorthington, ZoeSaldana, SigourneyWeaver, …  [StephenE.Rivkin]
1  [JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste…    [GoreVerbinski]
```

1.0.4 4.Merging the datasets

[55]:
```
dataset = movies.merge(credit , on = "title")
```

[56]:
```
dataset.head(2)
```

[56]:
```
      budget                                         genres     id  \
0  237000000  [Action, Adventure, Fantasy, ScienceFiction]  19995
1  300000000                  [Adventure, Fantasy, Action]    285

                                    keywords original_language  \
0  [Action, Adventure, Fantasy, ScienceFiction]                en
1                  [Adventure, Fantasy, Action]                en

                           original_title  \
0                                  Avatar
1  Pirates of the Caribbean: At World's End

                                        overview  popularity  \
0  [In, the, 22nd, century,, a, paraplegic, Marin…  150.437577
1  [Captain, Barbossa,, long, believed, to, be, d…  139.082615

                              production_companies  \
0  [Ingenious Film Partners, Twentieth Century Fo…
1  [Walt Disney Pictures, Jerry Bruckheimer Films…

                       production_countries release_date     revenue  \
0  [United States of America, United Kingdom]   2009-12-10  2787965087
1                  [United States of America]   2007-05-19   961000000

   runtime    spoken_languages                                    title  \
0    162.0  [English, Español]                                   Avatar
1    169.0           [English]  Pirates of the Caribbean: At World's End

   vote_average  vote_count  movie_id  \
0           7.2       11800     19995
1           6.9        4500       285

                                                cast                crew
0  [SamWorthington, ZoeSaldana, SigourneyWeaver, …  [StephenE.Rivkin]
1  [JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste…    [GoreVerbinski]
```

Combining the required column into a single column to build the recommendation system

[57]:
```
dataset["tags"] = dataset["genres"] + dataset["keywords"]+ dataset["overview"]+␣
↪dataset["cast"]+ dataset["crew"]
```

Convert the text in the tags column to lower case

Function to convert the text to lower case

[58]:
```
def lower_case(text) :
    l = []
    for i in text :
        i = i.lower()
        l.append(i)

    return l
```

[59]:
```
dataset["tags"] = dataset["tags"].apply(lower_case)
```

[60]:
```
dataset.head(2)
```

[60]:

	budget	genres	id	keywords	original_language	original_title	overview	popularity	production_companies	production_countries	…	revenue	runtime	spoken_languages	title	vote_average
0	237000000	[Action, Adventure, Fantasy, ScienceFiction]	19995	[Action, Adventure, Fantasy, ScienceFiction]	en	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin…	150.437577	[Ingenious Film Partners, Twentieth Century Fo…	[United States of America, United Kingdom]	…	2787965087	162.0	[English, Español]	Avatar	7.2
1	300000000	[Adventure, Fantasy, Action]	285	[Adventure, Fantasy, Action]	en	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d…	139.082615	[Walt Disney Pictures, Jerry Bruckheimer Films…	[United States of America]	…	961000000	169.0			

```
1           [English]  Pirates of the Caribbean: At World's End          6.9

   vote_count  movie_id                                                cast  \
0       11800     19995  [SamWorthington, ZoeSaldana, SigourneyWeaver, …
1        4500       285  [JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste…

                crew                                                tags
0  [StephenE.Rivkin]  [action, adventure, fantasy, sciencefiction, a…
1    [GoreVerbinski]  [adventure, fantasy, action, adventure, fantas…

[2 rows x 21 columns]
```

Convert the tags column into string datatype

[61]:
```
dataset["tags"] = dataset["tags"].apply(lambda x : " ".join(x))
```

[62]:
```
df = pd.DataFrame(dataset)
```

[63]:
```
df.drop(columns = ["genres","budget","keywords","original_language","overview",
 ↪"popularity","production_companies","production_countries",
 ↪"spoken_languages","vote_average","vote_count","movie_id",
 ↪"cast","crew","revenue","runtime","release_date","original_title"],inplace =␣
 ↪True)
```

[64]:
```
df.head(2)
```

[64]:
```
      id                                     title  \
0  19995                                    Avatar
1    285  Pirates of the Caribbean: At World's End

                                                tags
0  action adventure fantasy sciencefiction action…
1  adventure fantasy action adventure fantasy act…
```

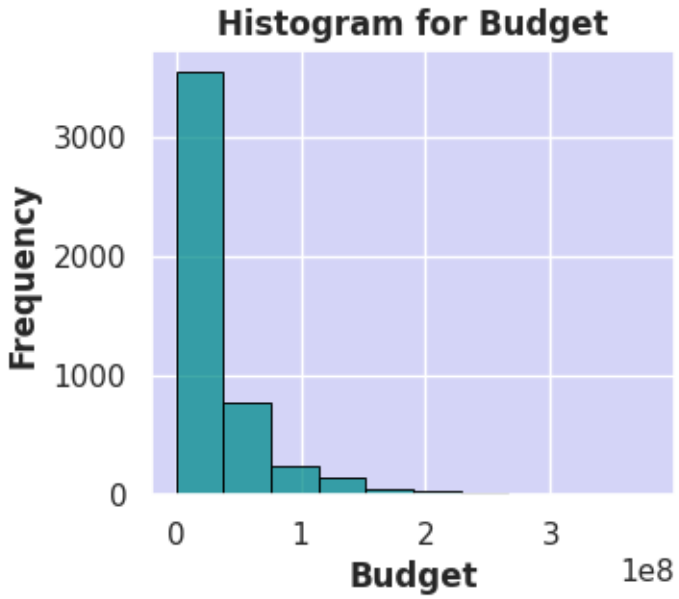
1.0.5 5.Exploratory data analysis

Univariate Analysis

Histogram for budget

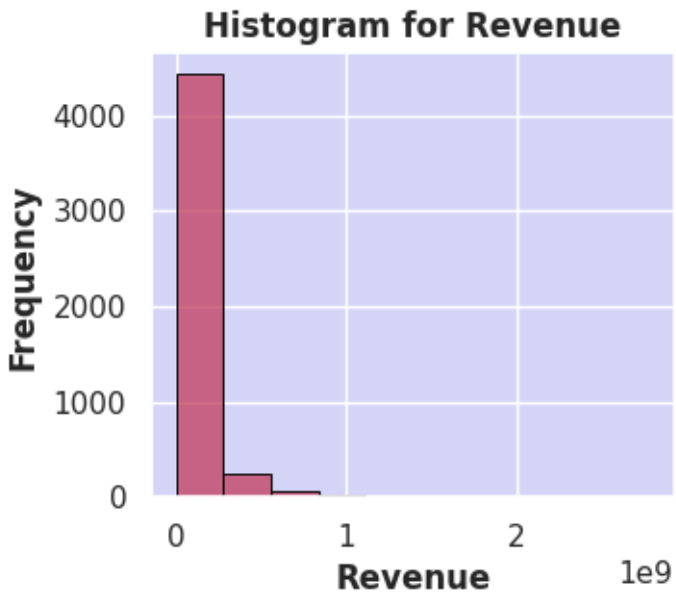
[65]:
```
plt.figure(figsize = (3.5,3))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white'})
sns.histplot(data = dataset , x = "budget" , bins = 10 , color = "darkcyan"␣
 ↪,edgecolor = "black" ,linewidth = 0.65)
plt.xlabel("Budget",weight='bold')
```

```
plt.ylabel("Frequency",weight='bold')
plt.title("Histogram for Budget",weight='bold')
plt.show()
```



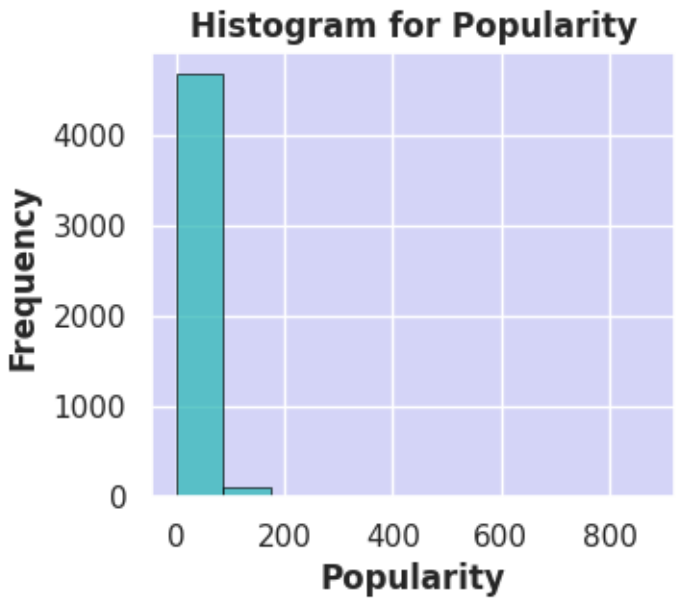
Histogram for Revenue

[66]:
```
plt.figure(figsize = (3.5,3))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white'})
sns.histplot(data = dataset , x = "revenue" , bins = 10 ,color = "#c23d5e"␣
↪,edgecolor = "black" ,linewidth = 0.65)
plt.xlabel("Revenue",weight='bold')
plt.ylabel("Frequency",weight='bold')
plt.title("Histogram for Revenue",weight='bold')
plt.show()
```



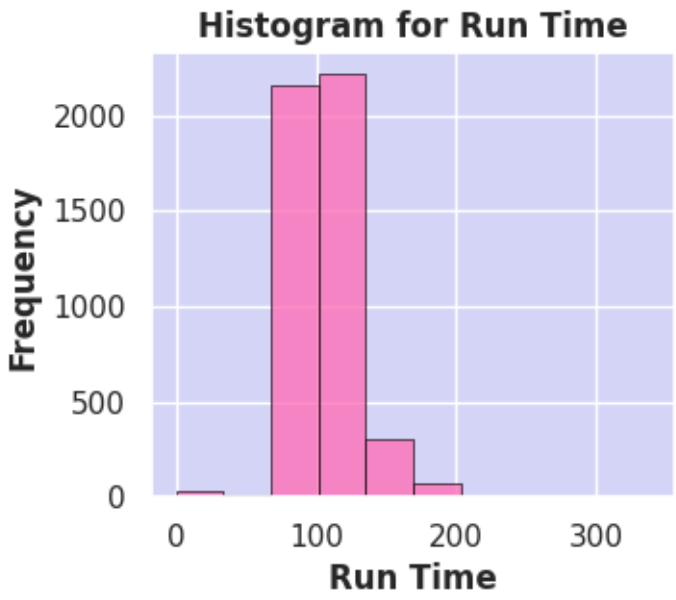
Histogram for Popularity

[67]:
```
plt.figure(figsize = (3.5,3))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white'})
sns.histplot(data = dataset , x = "popularity" , bins = 10 , color = "#2eb8b8"␣
↪,edgecolor = "black" ,linewidth = 0.45)
plt.xlabel("Popularity",weight='bold')
plt.ylabel("Frequency",weight='bold')
plt.title("Histogram for Popularity",weight='bold')
plt.show()
```



Histogram for Run time

[68]:
```
plt.figure(figsize = (3.5,3))
sns.set_style("darkgrid")
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white'})
sns.histplot(data = dataset , x = "runtime" , bins = 10  , color = "hotpink"␣
 ↪,edgecolor = "black",linewidth = 0.45)
plt.xlabel("Run Time",weight='bold')
plt.ylabel("Frequency",weight='bold')
plt.title("Histogram for Run Time",weight='bold')
plt.show()
```

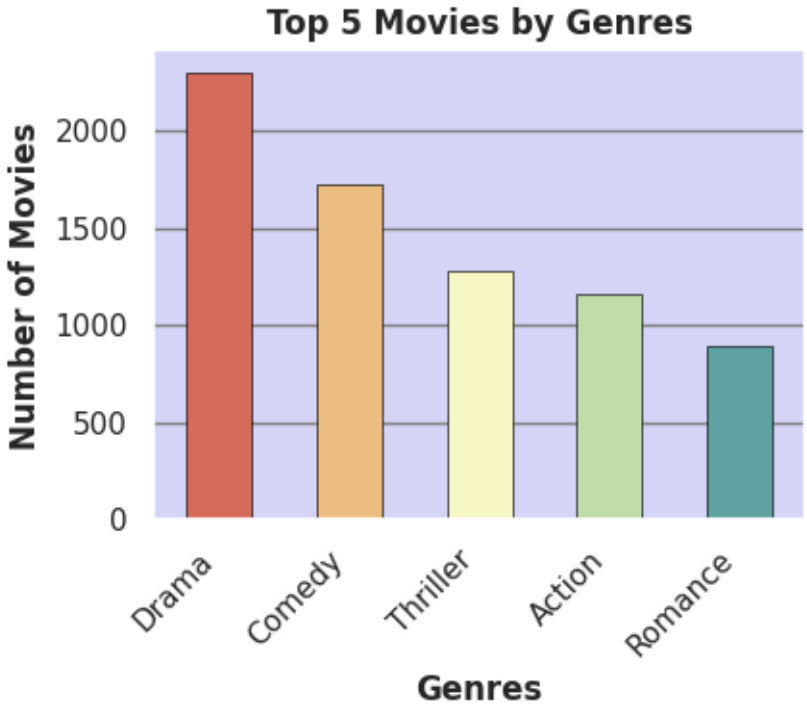


Top 5 geners

[69]:
```
genres_count = dataset["genres"].explode().value_counts()
genres_df = pd.DataFrame(genres_count).reset_index()


plt.figure(figsize = (4.5,4))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white' , "grid.
 ↪color" : "grey"})
sns.barplot(x = genres_df["genres"].head(5),y = genres_df["count"].head(5),data␣
 ↪= df , width = 0.5,edgecolor = "black" ,palette = "Spectral",linewidth = 0.
 ↪45)
plt.xlabel("Genres",weight='bold')
plt.ylabel("Number of Movies",weight='bold')
plt.title("Top 5 Movies by Genres",weight='bold')
plt.xticks(rotation = 45 ,ha = "right")
plt.tight_layout() # Adjust layout to prevent overlap plt.show()
plt.show()
```

```
<ipython-input-69-6069124cad1e>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x = genres_df["genres"].head(5),y =
genres_df["count"].head(5),data = df , width = 0.5,edgecolor = "black" ,palette
```

```
= "Spectral",linewidth = 0.45)
```



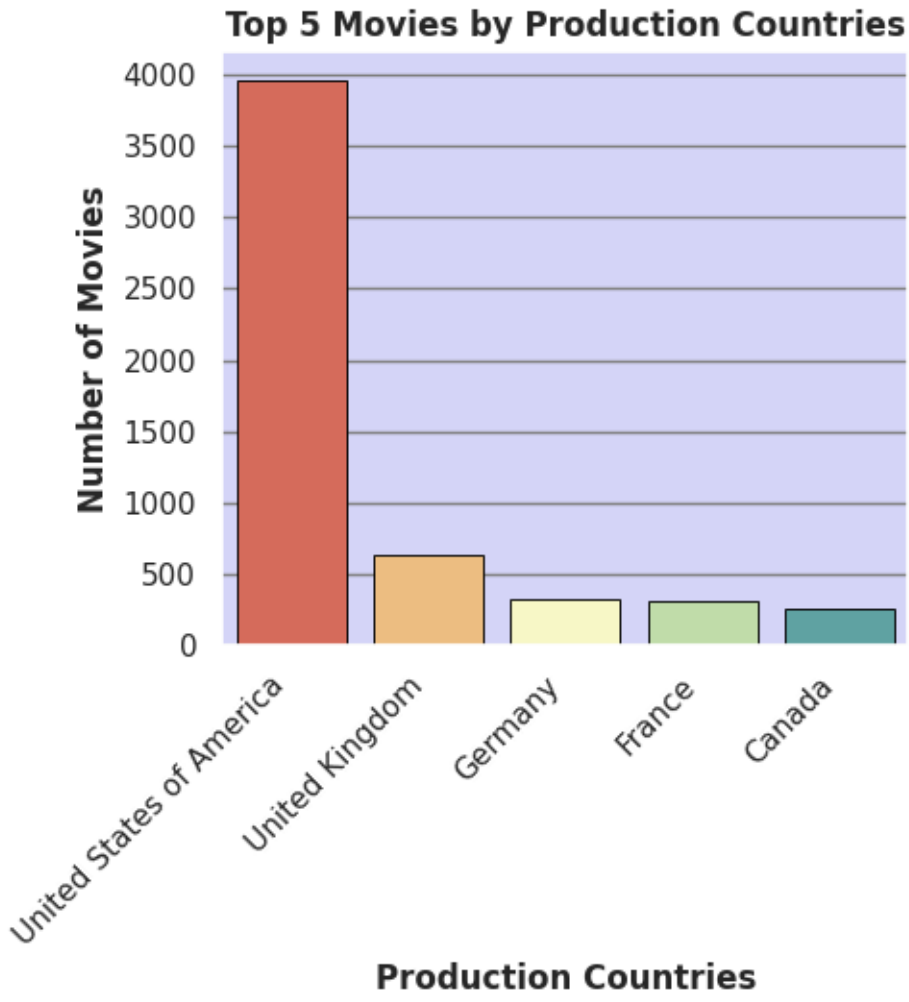
Top 5 Production countries

[70]:
```
countries_count = dataset["production_countries"].explode().value_counts()
countries_count_df = pd.DataFrame(countries_count).reset_index()

plt.figure(figsize = (5,5.5))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white' , "grid.
 ↪color" : "grey"})
sns.barplot(x = countries_count_df["production_countries"].head(5),y =␣
 ↪countries_count_df["count"].head(5),data = df ,linewidth = 0.65 , palette=␣
 ↪"Spectral",edgecolor = "black" )
plt.xlabel("Production Countries",weight='bold')
plt.ylabel("Number of Movies",weight='bold')
plt.title("Top 5 Movies by Production Countries",weight='bold')
plt.xticks(rotation = 45 ,ha = "right")
plt.tight_layout() # Adjust layout to prevent overlap
plt.show()
```

```
<ipython-input-70-5197a708cd76>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
```

effect.

```
  sns.barplot(x = countries_count_df["production_countries"].head(5),y =
countries_count_df["count"].head(5),data = df ,linewidth = 0.65 , palette=
"Spectral",edgecolor = "black" )
```

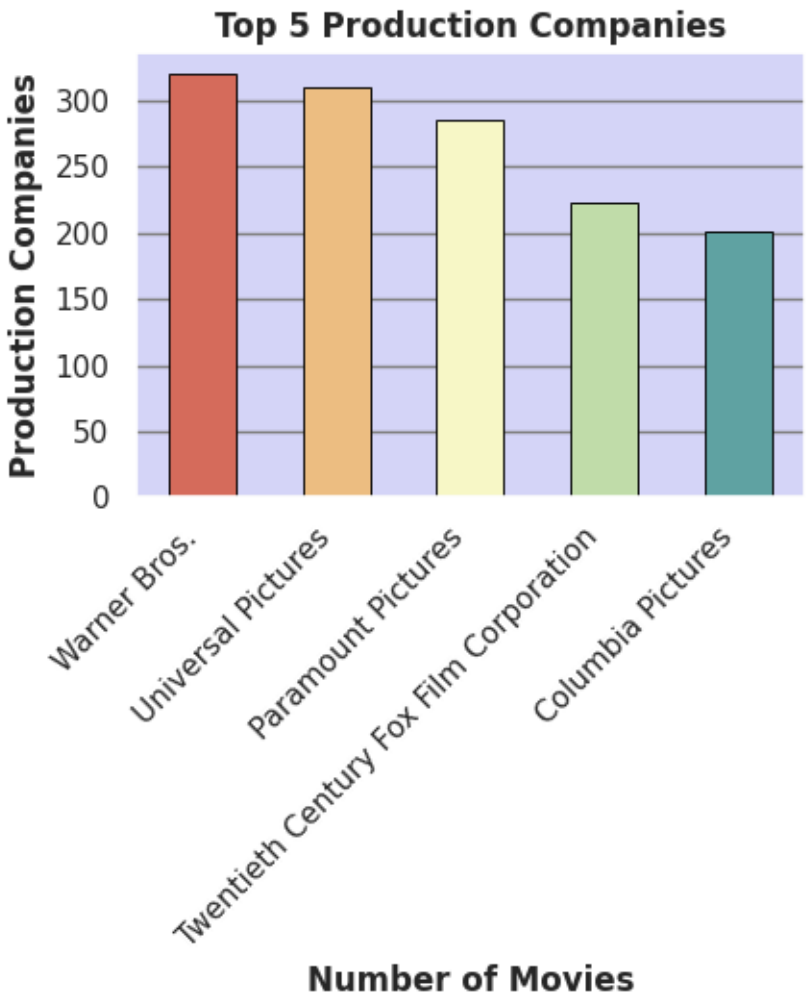


Top 5 Production countries

[71]:
```
company_count = dataset["production_companies"].explode().value_counts()
company_count_df = pd.DataFrame(company_count).reset_index()


plt.figure(figsize = (4.5,5.5))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white', "grid.
 ↪color" : "grey"})
```

```
sns.barplot(data = company_count_df ,x =␣
 ↪company_count_df["production_companies"].head(5), y =␣
 ↪company_count_df["count"].head(5), palette=␣
 ↪"Spectral",edgecolor='black',width = 0.5,linewidth=0.65)
plt.xticks(rotation = 45 ,ha = "right")
plt.ylabel("Production Companies",weight='bold')
plt.xlabel("Number of Movies",weight='bold')
plt.title("Top 5 Production Companies",weight='bold')
plt.tight_layout()
```

```
<ipython-input-71-54b9369d90a6>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(data = company_count_df ,x =
company_count_df["production_companies"].head(5), y =
company_count_df["count"].head(5), palette= "Spectral",edgecolor='black',width =
0.5,linewidth=0.65)
```

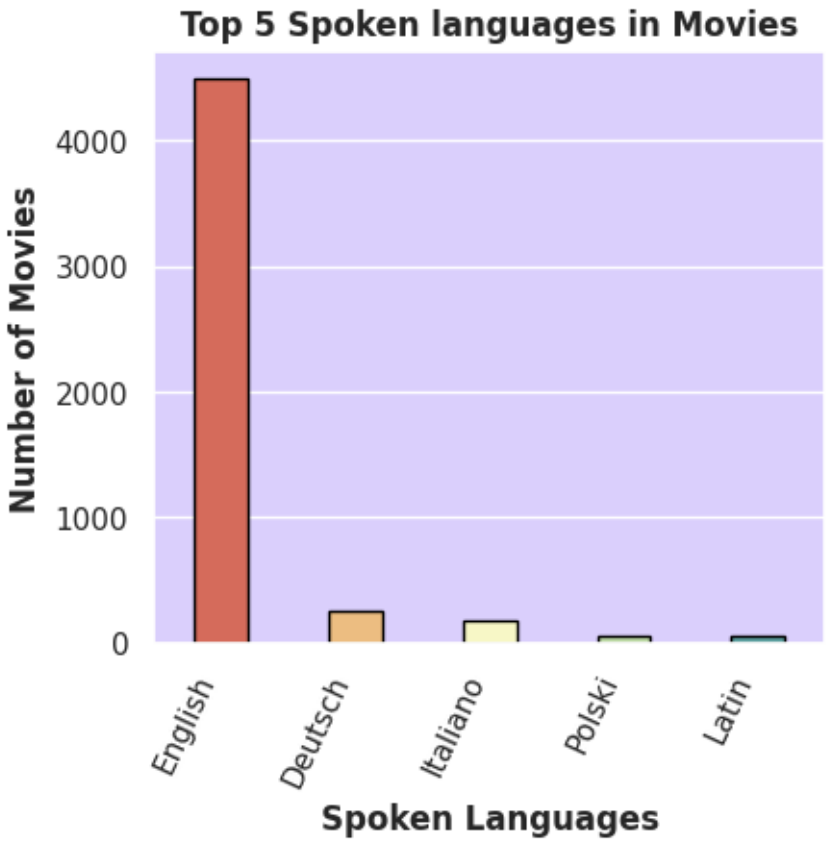


Top 5 Spoken languages

[72]:
```
language_count = dataset["spoken_languages"].explode().value_counts()
language_df = pd.DataFrame(language_count).reset_index()
language_df = language_df[language_df["spoken_languages"].str.
 ↪match(r"[a-zA-Z]+$")]


plt.figure(figsize = (4.5,4))
sns.set(rc = {"axes.facecolor" : "#dacffc" , "figure.facecolor" : "white"})
sns.barplot(x = language_df["spoken_languages"].head(5) , y =␣
 ↪language_df["count"].head(5) , palette= "Spectral" ,edgecolor =␣
 ↪"black",width = 0.4)
plt.xlabel("Spoken Languages",weight='bold')
plt.ylabel("Number of Movies",weight='bold')
plt.title("Top 5 Spoken languages in Movies",weight='bold')
```

```
plt.xticks(rotation = 65 ,ha = "right")
plt.show()
```

```
<ipython-input-72-7e83f6fa27ad>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x = language_df["spoken_languages"].head(5) , y =
language_df["count"].head(5) , palette= "Spectral" ,edgecolor = "black",width =
0.4)
```

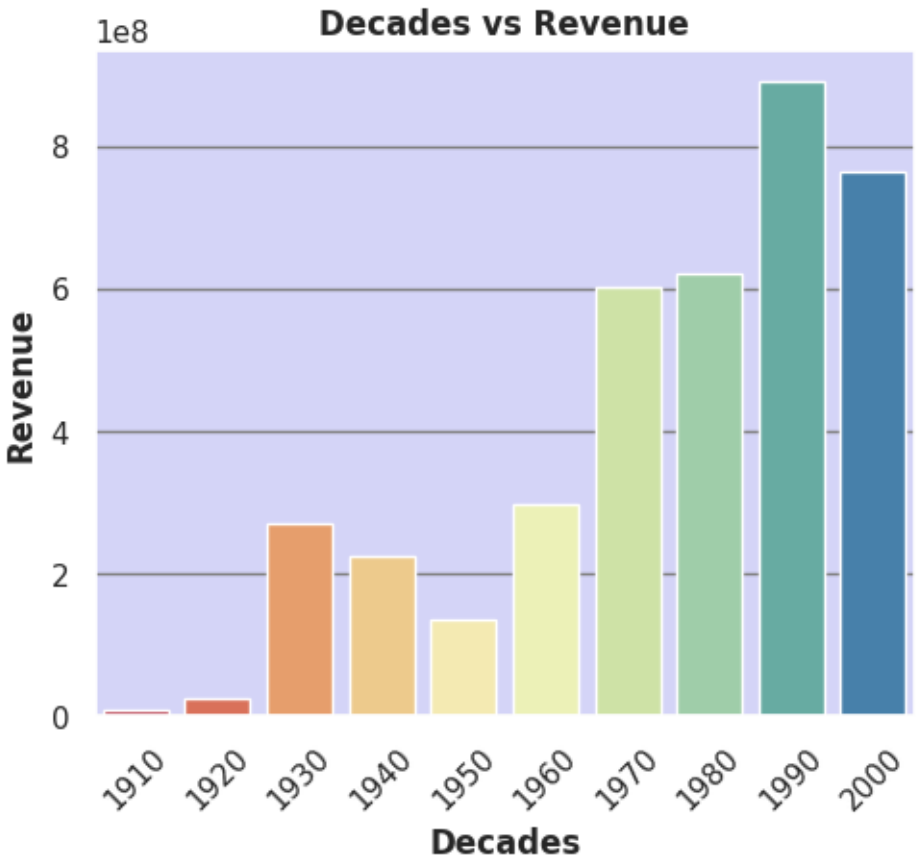


Revenue based on Decades

[73]:
```
revenue_data = pd.DataFrame(dataset)

revenue_data["release_date"] = pd.to_datetime(revenue_data["release_date"])
revenue_data["release_year"] = revenue_data['release_date'].dt.year
```

```
revenue_data = revenue_data.groupby('release_year')['revenue'].mean().
 ↪reset_index()
revenue_data["decade"] = (revenue_data["release_year"]//10) *10
revenue_df = revenue_data.groupby('decade')['revenue'].sum().reset_index()


plt.figure(figsize=(5.5,4.5))
sns.set(rc={'axes.facecolor':'#d4d4f7', 'figure.facecolor':'white' , "grid.
 ↪color" : "grey"})
sns.barplot( x =revenue_df['decade'].head(10) , y = revenue_df['revenue'].
 ↪head(10),data=revenue_df , palette = "Spectral")
plt.xlabel('Decades', weight='bold')
plt.ylabel('Revenue', weight='bold')
plt.title('Decades vs Revenue', weight='bold')
plt.xticks(rotation = 45)
plt.show()
```

```
<ipython-input-73-daba049236a4>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot( x =revenue_df['decade'].head(10) , y =
revenue_df['revenue'].head(10),data=revenue_df , palette = "Spectral")
```

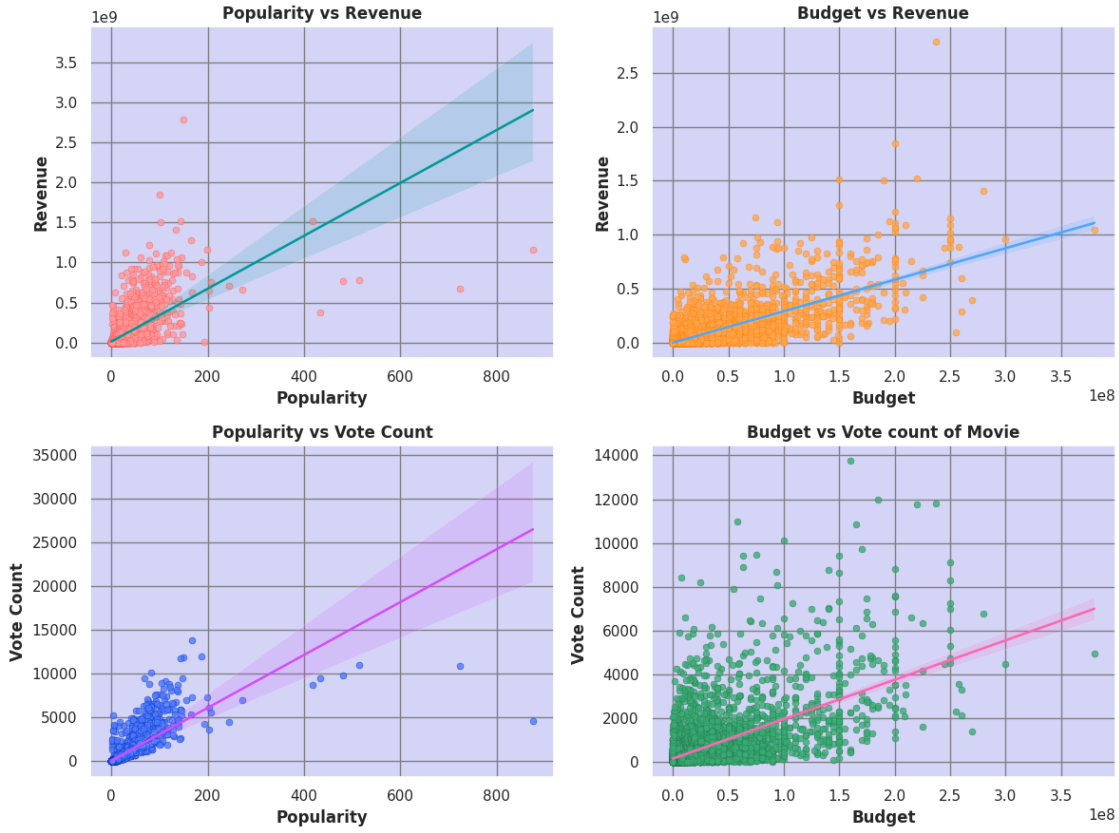


Bivariate analysis

Regression plot

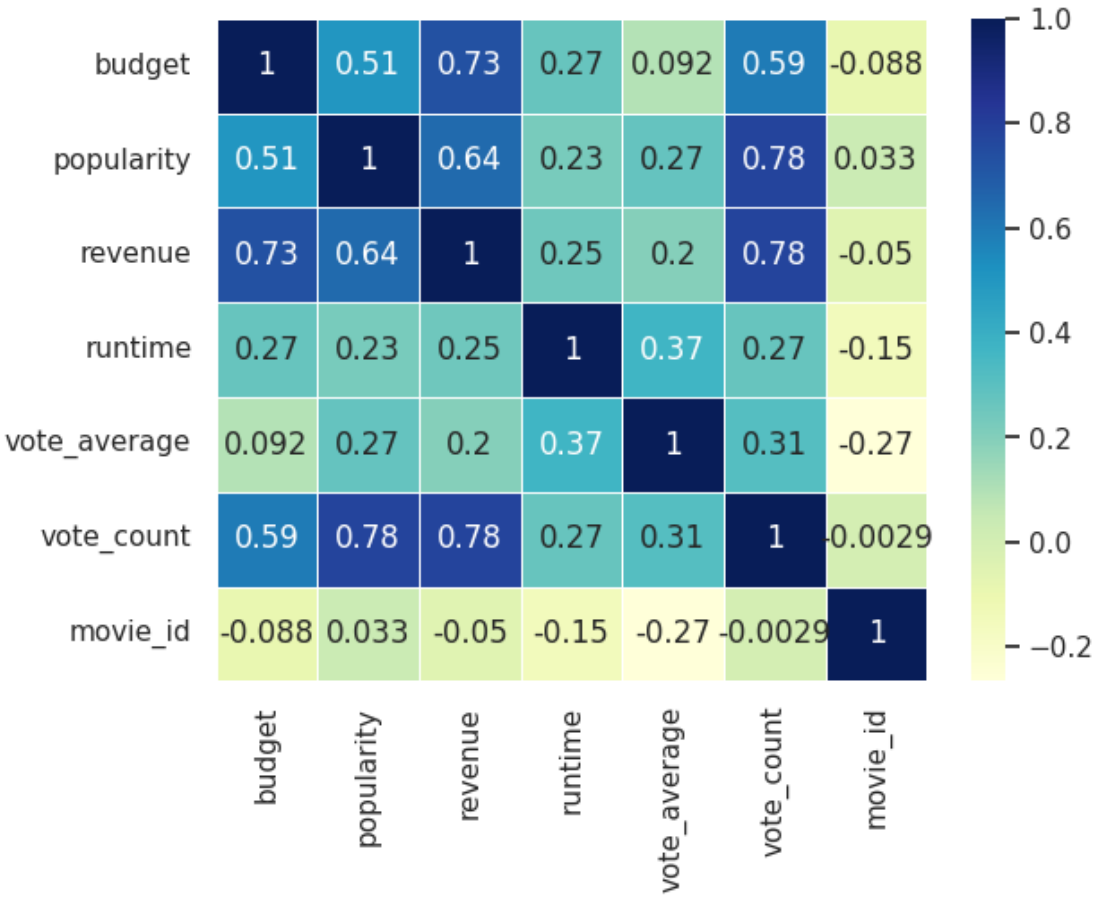
[74]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 9))  # 1 row, 2 columns

# Plotting on the first subplot
sns.regplot(x="popularity", y="revenue", data=dataset ,color = "#ff9999" ,
            scatter_kws={'edgecolor': '#ff5050','linewidths': 0.3,'s' : 25},
            line_kws={'color': '#009999','linewidth':1.8}, ax=axes[0][0])

axes[0][0].set_xlabel("Popularity", fontweight='bold',fontsize = 12)
axes[0][0].set_ylabel("Revenue", fontweight='bold',fontsize = 12)
axes[0][0].set_title("Popularity vs Revenue", fontweight='bold',fontsize =12)


# Plotting on the second subplot
sns.regplot(x = "budget", y = "revenue" , data = dataset ,color = "#ffa64d" ,
            scatter_kws={'edgecolor': '#ff8000','linewidths': 0.3,'s' : 25},
            line_kws={'color': '#4da6ff','linewidth':1.8}, ax=axes[0][1])
```

```
axes[0][1].set_xlabel("Budget", fontweight='bold',fontsize = 12)
axes[0][1].set_ylabel("Revenue", fontweight='bold',fontsize = 12)
axes[0][1].set_title("Budget vs Revenue", fontweight='bold',fontsize =12)


sns.regplot(x = "popularity", y = "vote_count" , data = dataset ,color =␣
 ↪"#4d79ff" ,
            scatter_kws={'edgecolor': '#002db3','linewidths': 0.3,'s' : 25},
            line_kws={'color': '#d24dff','linewidth':1.8}, ax=axes[1][0])

axes[1][0].set_xlabel("Popularity", fontweight='bold',fontsize = 12)
axes[1][0].set_ylabel("Vote Count", fontweight='bold',fontsize = 12)
axes[1][0].set_title("Popularity vs Vote Count", fontweight='bold',fontsize =12)




sns.regplot(x = "budget", y = "vote_count" ,data = dataset ,color = "#39ac73" ,
            scatter_kws={'edgecolor': '#26734d','linewidths': 0.3,'s' : 25},
            line_kws={'color': '#ff66b3','linewidth':1.8}, ax=axes[1][1])

axes[1][1].set_xlabel("Budget", fontweight='bold',fontsize = 12)
axes[1][1].set_ylabel("Vote Count ", fontweight='bold',fontsize = 12)
axes[1][1].set_title("Budget vs Vote count of Movie",␣
 ↪fontweight='bold',fontsize = 12)


# Adjust layout for better display
plt.tight_layout()
plt.show()
```



Multivariate Analysis Heat map

[75]:
```
numerical_data = pd.DataFrame(dataset)
numerical_data.drop(columns =["genres","keywords","original_language",
 ↪"original_title","overview","production_companies","production_countries",
                               "release_date","spoken_languages","title",
                               "cast","crew","id","tags"],inplace = True)

numerical_data.dropna(inplace = True)

corr_matrix = numerical_data.corr()

sns.heatmap(corr_matrix ,annot = True,linewidths=.5 ,cmap="YlGnBu")
plt.show()
```



1.0.6 6. Build Recommendation system

Using Term frequency , Inverse term frequency to create tf_idf matrix and cosine similarity to find similarity among the movies

Overview of the processed dataset to build recommendation system

[76]:
```
df.head()
```

[76]:

	id	title	tags
0	19995	Avatar	action adventure fantasy sciencefiction action…
1	285	Pirates of the Caribbean: At World's End	adventure fantasy action adventure fantasy act…
2	206647	Spectre	
3	49026	The Dark Knight Rises	
4	49529	John Carter	

```
2  action adventure crime action adventure crime …
3  action crime drama thriller action crime drama…
4  action adventure sciencefiction action adventu…
```

Import the libraries needed for find similar movies

[77]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

[78]:
```
df["tags"] = df["tags"].apply(lambda x : "".join(x))
```

[79]:
```
df.head()
```

[79]:
```
       id                                     title  \
0   19995                                    Avatar
1     285  Pirates of the Caribbean: At World's End
2  206647                                   Spectre
3   49026                     The Dark Knight Rises
4   49529                               John Carter

                                                tags
0  action adventure fantasy sciencefiction action…
1  adventure fantasy action adventure fantasy act…
2  action adventure crime action adventure crime …
3  action crime drama thriller action crime drama…
4  action adventure sciencefiction action adventu…
```

Create an instance of TdidfVectorizer

[80]:
```
tf_idf = TfidfVectorizer(stop_words='english')
```

Transforming Text Data into a TF-IDF Matrix

[81]:
```
tfidf_matrix = tf_idf.fit_transform(df['tags'])
```

Calculating Cosine Similarity

[82]:
```
cos_similarity = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

Creating a Mapping of Movie Titles to Indices

[83]:
```
data_set = pd.Series(df.index ,index = df["title"]).drop_duplicates()
```

Function to calculate the similarity between the movies

[114]:
```
def get_recommendation(title ,cosine = cos_similarity) :
  if title not in list(df["title"]) :
    print(f"{title} not present in the dataset")
    return
```

```
  movie_index = data_set[title]
  sim_scores = cosine[movie_index]
  sim_scores = list(enumerate(sim_scores))
  sim_scores = sorted(sim_scores , key = lambda x : x[1] , reverse = True)
  top_recommendation = sim_scores[1:15]
  l = []
  for i in top_recommendation :
    l.append(i[0])

  movies_list = df["title"].iloc[l]

  recommended_list = []
  for movie in  movies_list:
    recommended_list.append(movie)

  return recommended_list
```

[115]:
```
def print_movie(get_recommendation):
  recommended_list = get_recommendation
  print("\nRecommended movies : \n")
  for recommended_movie in recommended_list :
    print(recommended_movie)
```

7.Testing of the Recommendation system

[116]:
```
print_movie(get_recommendation("The Dark Knight Rises"))
```

```
Recommended movies :

The Dark Knight
Batman Begins
Batman Forever
Batman Returns
Batman
Batman
Slow Burn
Inception
Batman: The Dark Knight Returns, Part 2
Batman & Robin
Batman v Superman: Dawn of Justice
JFK
Sin City: A Dame to Kill For
Premium Rush
```

[117]:
```
print_movie(get_recommendation("Spider-Man"))
```

```
Recommended movies :

Spider-Man 2
Spider-Man 3
The Amazing Spider-Man 2
The Amazing Spider-Man
Gremlins 2: The New Batch
Arachnophobia
Charlotte's Web
Hook
Small Soldiers
Spawn
Kick-Ass
Mystery Men
Man of Steel
Heavenly Creatures
```

[97]:
```
print_movie(get_recommendation("Iron Man 3"))
```

```
Recommended movies for Iron Man 3 :

Iron Man
Iron Man 2
The Helix… Loaded
Avengers: Age of Ultron
Captain America: Civil War
Star Trek Into Darkness
Cradle 2 the Grave
X-Men
The Walk
The Avengers
The Abyss
Avatar
Transcendence
A Sound of Thunder
```

[98]:
```
print_movie(get_recommendation("Jurassic World"))
```

```
Recommended movies for Jurassic World :

The Lost World: Jurassic Park
Jurassic Park
The Helix… Loaded
National Lampoon's Vacation
Guardians of the Galaxy
The Cell
```

```
The Nut Job
Impostor
The 5th Wave
The Blood of Heroes
Jurassic Park III
Cloverfield
Terminator Salvation
Knowing
```

[99]:
```
print_movie(get_recommendation("Toy Story 3"))
```

```
Recommended movies for Toy Story 3 :

Toy Story
Toy Story 2
The 40 Year Old Virgin
Factory Girl
Class of 1984
Heartbeeps
The Rugrats Movie
Daddy Day Care
Man on the Moon
Rango
A LEGO Brickumentary
Chicken Little
Monsters, Inc.
How to Train Your Dragon 2
```

[100]:
```
print_movie(get_recommendation("Ice Age"))
```

```
Recommended movies for Ice Age :

Ice Age: Dawn of the Dinosaurs
Ice Age: The Meltdown
Ice Age: Continental Drift
Jimmy Neutron: Boy Genius
The Croods
Frozen
Kung Fu Panda 3
Titan A.E.
Shrek the Third
Bolt
Big Hero 6
The Good Dinosaur
Osmosis Jones
Madagascar
```

8.Model Evaluation

[124]:
```
# Define functions for evaluation metrics
def precision_at_k(actual, predicted, k):
        actual_set = set(actual)
        predicted_set = set(predicted[:k])
        common_items = actual_set.intersection(predicted_set)
        precision = len(common_items) / k if k > 0 else 0
        return precision

def recall_at_k(actual, predicted, k):
        actual_set = set(actual)
        predicted_set = set(predicted[:k])
        common_items = actual_set.intersection(predicted_set)
        recall = len(common_items) / len(actual) if len(actual) > 0 else 0
        return recall

def f1_score(precision, recall):
        f1 = 2 * (precision * recall) / (precision + recall) if (precision +␣
 ↪recall) > 0 else 0
        return f1
```

[131]:
```
import random

def select_elements(lst):
    # Ensure the list is not empty
    if not lst:
        return None  # or raise an exception

    # Randomly select at least one element
    num_elements_to_select = random.randint(1, len(lst))
    selected_elements = random.sample(lst, num_elements_to_select)

    return selected_elements
```

[132]:
```
# Evaluate Precision@K, Recall@K, and F1 Score

# Define actual user preferences of Ice Age
def evaluate(title):

  actual = get_recommendation(title)
  actual = actual[:5]
  actual_preferences = select_elements(actual)

# Generate recommendations using the recommendation function
  recommended_movies = get_recommendation(title)
  k = 3
  precision = precision_at_k(actual_preferences, recommended_movies, k)
```

```
    recall = recall_at_k(actual_preferences, recommended_movies, k)
    f1 = f1_score(precision, recall)

    print("Precision@{}: {:.4f}".format(k, precision))
    print("Recall@{}: {:.4f}".format(k, recall))
    print("F1 Score: {:.4f}".format(f1))
```

[133]:
```
evaluate("Spider-Man")
```

```
Precision@3: 0.3333
Recall@3: 0.5000
F1 Score: 0.4000
```

[136]:
```
evaluate("Ice Age")
```

```
Precision@3: 0.6667
Recall@3: 0.5000
F1 Score: 0.5714
```

[145]:
```
evaluate("Toy Story")
```

```
Precision@3: 1.0000
Recall@3: 0.6000
F1 Score: 0.7500
```

[148]:
```
evaluate("Iron Man")
```

```
Precision@3: 0.3333
Recall@3: 1.0000
F1 Score: 0.5000
```

[ ]:
```

```