# Final Project Submission

Please fill out:

- Student name: CYNTHIA SHEILAH ATIENO
- Student pace: part time
- Scheduled project review date/time:
- Instructor name: SAMUEL KARU
- Blog post URL:

# # Step 0: Imports and Reading Data

In [77]:
```
# Your code here - remember to use markdown cells for comments as well!
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
%matplotlib inline
pd.set_option('max_columns',150)
```

In [5]:
```
#connecting to sqlite3
conn = sqlite3.connect('im.db')
```

In [6]:
```
cur = conn.cursor()
```

In [8]:
```
# SQL query to select the names of all tables in the database
cur.execute("""SELECT name FROM sqlite_master WHERE type = 'table';""")
table_names = cur.fetchall()
table_names
```

Out[8]:
```
[('movie_basics',),
 ('directors',),
 ('known_for',),
 ('movie_akas',),
 ('movie_ratings',),
 ('persons',),
 ('principals',),
 ('writers',)]
```

In [20]:
```
import sqlite3
import pandas as pd

try:
    # Connecting to the SQLite database using a context manager
    with sqlite3.connect(r'C:\Users\DAVID\Documents\im.db') as conn:
        # SQL query to select the names of all tables in the database
        query = "SELECT name FROM sqlite_master WHERE type='table';"

        # Reading the SQL query into a pandas DataFrame
        df_tables = pd.read_sql_query(query, conn)
        table_names = df_tables['name'].tolist()

        # Dictionary to hold data from each table
        tables_data = {}

        for table in table_names:
            # Query each table and store the result in the dictionary
            tables_data[table] = pd.read_sql_query(f"SELECT * FROM {table}", cc

    # Combine all data into one DataFrame
    combined_df = pd.concat(tables_data.values(), ignore_index=True)

    # Display the combined DataFrame
    combined_df.head()

except sqlite3.Error as e:
    print(f"An error occurred: {e}")
```

# # Step 1: Data Understanding

```
Dataframe shape
head and tail
dtypes
describe
```

In [74]:
```
combined_df.shape
```

Out[74]:
```
(4371844, 23)
```

In [78]:
```
combined_df.head()
```

Out[78]:

| gion | language | types | attributes | is_original_title | averagerating | numvotes | primary_name | birth_yea |
|---|---|---|---|---|---|---|---|---|
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |

In [71]:
```
combined_df.tail()
```

Out[71]:

| | movie_id | primary_title | original_title | start_year | runtime_minutes | genres | person_id |
|---|---|---|---|---|---|---|---|
| **4371839** | tt8999892 | NaN | NaN | NaN | NaN | NaN | nm10122246 |
| **4371840** | tt8999974 | NaN | NaN | NaN | NaN | NaN | nm10122357 |
| **4371841** | tt9001390 | NaN | NaN | NaN | NaN | NaN | nm6711477 |
| **4371842** | tt9004986 | NaN | NaN | NaN | NaN | NaN | nm4993825 |
| **4371843** | tt9010172 | NaN | NaN | NaN | NaN | NaN | nm8352242 |

5 rows × 23 columns

In [73]:
```
combined_df.dtypes
```

Out[73]:
```
movie_id               object
primary_title          object
original_title         object
start_year            float64
runtime_minutes       float64
genres                 object
person_id              object
ordering              float64
title                  object
region                 object
language               object
types                  object
attributes             object
is_original_title     float64
averagerating         float64
numvotes              float64
primary_name           object
birth_year            float64
death_year            float64
primary_profession     object
category               object
job                    object
characters             object
dtype: object
```

In [79]:
```
combined_df.columns
```

Out[79]:
```
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'person_id', 'ordering', 'title', 'regio
n',
       'language', 'types', 'attributes', 'is_original_title', 'averageratin
g',
       'numvotes', 'primary_name', 'birth_year', 'death_year',
       'primary_profession', 'category', 'job', 'characters'],
      dtype='object')
```

In [32]:
```
combined_df.describe()
```

Out[32]:

| | start_year | runtime_minutes | ordering | is_original_title | averagerating | numvote |
|---|---|---|---|---|---|---|
| **count** | 146144.000000 | 114405.000000 | 1.359889e+06 | 331678.000000 | 73856.000000 | 7.385600e+0 |
| **mean** | 2014.621798 | 86.187247 | 4.834006e+00 | 0.134769 | 6.332729 | 3.523662e+0 |
| **std** | 2.733583 | 166.360590 | 4.087300e+00 | 0.341477 | 1.474978 | 3.029402e+0 |
| **min** | 2010.000000 | 1.000000 | 1.000000e+00 | 0.000000 | 1.000000 | 5.000000e+0 |
| **25%** | 2012.000000 | 70.000000 | 2.000000e+00 | 0.000000 | 5.500000 | 1.400000e+0 |
| **50%** | 2015.000000 | 87.000000 | 4.000000e+00 | 0.000000 | 6.500000 | 4.900000e+0 |
| **75%** | 2017.000000 | 99.000000 | 7.000000e+00 | 0.000000 | 7.400000 | 2.820000e+0 |
| **max** | 2115.000000 | 51420.000000 | 6.100000e+01 | 1.000000 | 10.000000 | 1.841066e+0 |

# Step 2:Data Preparation

```
Finding missing values
Identifying Duplicated columns
Feature Creation
```

In [84]:
```
# change the birth_year to datetime
combined_df['birth_year']= pd.to_datetime(combined_df['birth_year'])
```

In [85]:
```
## change the death_year to datetime
combined_df['death_year']= pd.to_datetime(combined_df['death_year'])
```

In [86]:
```
combined_df.head()
```

Out[86]:

| | movie_id | primary_title | original_title | start_year | runtime_minutes | genres | persc |
|---|---|---|---|---|---|---|---|
| **0** | tt0063540 | Sunghursh | Sunghursh | 2013.0 | 175.0 | Action,Crime,Drama | |
| **1** | tt0066787 | One Day Before the Rainy Season | Ashad Ka Ek Din | 2019.0 | 114.0 | Biography,Drama | |
| **2** | tt0069049 | The Other Side of the Wind | The Other Side of the Wind | 2018.0 | 122.0 | Drama | |
| **3** | tt0069204 | Sabse Bada Sukh | Sabse Bada Sukh | 2018.0 | NaN | Comedy,Drama | |
| **4** | tt0100275 | The Wandering Soap Opera | La Telenovela Errante | 2017.0 | 80.0 | Comedy,Drama,Fantasy | |

In [88]:

```
#find all the missing values in the dataset.
combined_df.isna().sum()
```

Out[88]:

```
movie_id               606648
primary_title         4225700
original_title        4225721
start_year            4225700
runtime_minutes       4257439
genres                4231108
person_id              551703
ordering              3011955
title                 4040141
region                4093434
language              4330129
types                 4203397
attributes            4356919
is_original_title     4040166
averagerating         4297988
numvotes              4297988
primary_name          3765196
birth_year            4289108
death_year            4365061
primary_profession    3816536
category              3343658
job                   4194160
characters            3978484
dtype: int64
```

In [90]:

```
#finding duplicated values
combined_df.duplicated()
```

Out[90]:

```
0          False
1          False
2          False
3          False
4          False
           ...
4371839    False
4371840     True
4371841     True
4371842     True
4371843    False
Length: 4371844, dtype: bool
```

In [93]:
```
combined_df.loc[combined_df.duplicated()].head()
```

Out[93]:

| gion | language | types | attributes | is_original_title | averagerating | numvotes | primary_name | birth_yea |
|---|---|---|---|---|---|---|---|---|
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |

In [35]:
```
#finding unique values
combined_df.nunique()
```

Out[35]:
```
movie_id              526391
primary_title         136071
original_title        137774
start_year                19
runtime_minutes          367
genres                  1086
person_id             606908
ordering                  61
title                 252781
region                   214
language                  77
types                     11
attributes                78
is_original_title          2
averagerating             91
numvotes                7349
primary_name          577203
birth_year               267
death_year               214
primary_profession      8648
category                  12
job                     2966
characters            174763
dtype: int64
```

# Step 3:Feature Understanding

Plotting Feature Distributions
Histogram
KDE
Boxplot

In [95]:
```
#The most watched movie by title
combined_df['title'].value_counts()
```

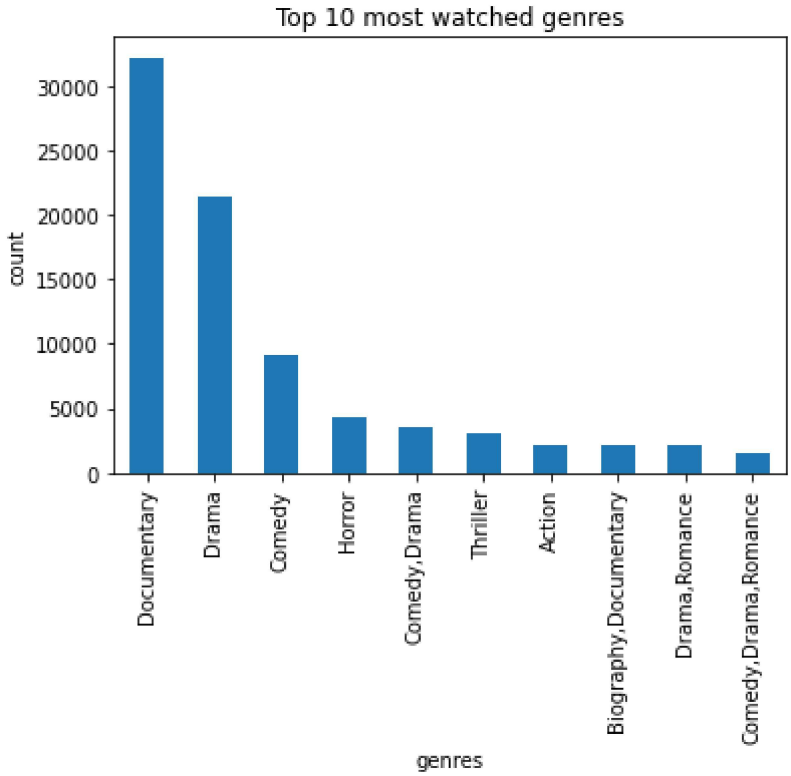
Out[95]:
```
Robin Hood                       32
Home                             30
Alone                            27
Love                             25
Thor                             25
                                 ..
Hashima                           1
Koinowa: Konkatsu Cruising        1
Iskušenik                         1
Любить, пить и петь               1
O Mensageiro dos Espíritos 2      1
Name: title, Length: 252781, dtype: int64
```

In [96]:
```
#Top 10 most watched genres
combined_df['genres'].value_counts()
```

Out[96]:
```
Documentary                     32185
Drama                           21486
Comedy                           9177
Horror                           4372
Comedy,Drama                     3519
                                ...
Biography,Family,Fantasy            1
Sport,Talk-Show                     1
Animation,Mystery,Thriller          1
Animation,Music,Mystery             1
Mystery,Reality-TV,Thriller         1
Name: genres, Length: 1085, dtype: int64
```

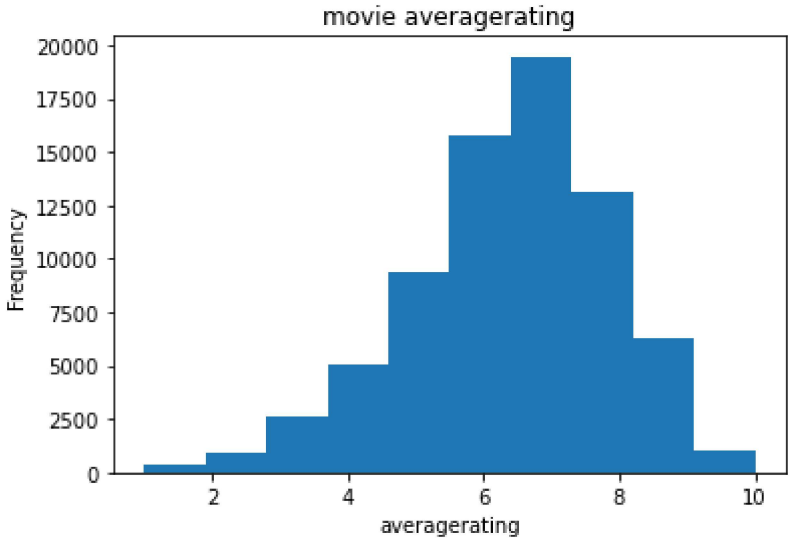
In [100]:
```
#visualizing the top 10 most watched genres using a bar plot
ax= combined_df['genres'].value_counts() \
            .head(10) \
            .plot(kind='bar',title='Top 10 most watched genres')
ax.set_xlabel('genres')
ax.set_ylabel('count')
```

Out[100]: Text(0, 0.5, 'count')




In [106]:
```
#Visualizing the averagerating using a histogram
ax= combined_df['averagerating'].plot(kind='hist',title='movie averagerating')
ax.set_xlabel('averagerating')
```

Out[106]: Text(0.5, 0, 'averagerating')



# # Step 4: Feature Relationships

Heatmap correlation
Groupby comparisons

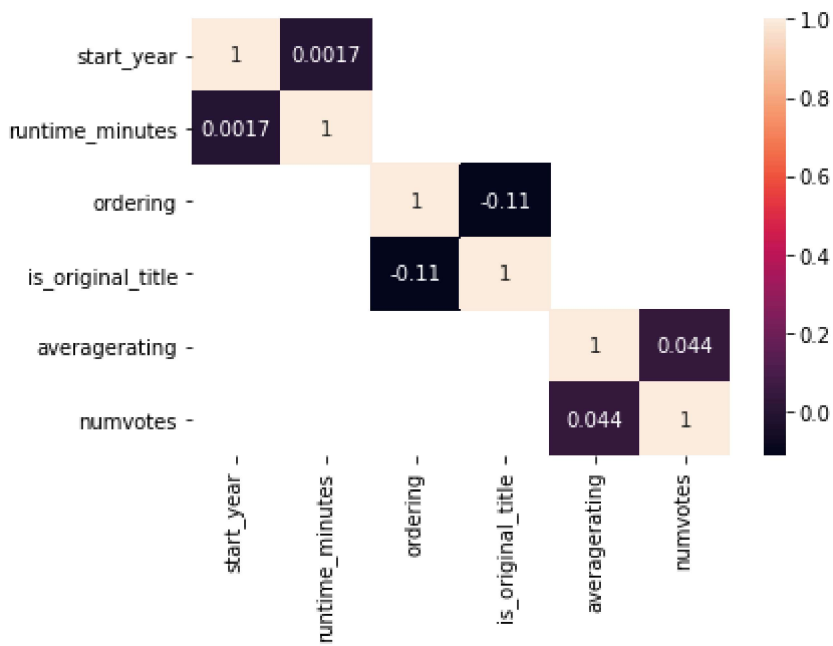
In [123]:
```
#check correlation
combined_df.corr()
```

Out[123]:

| | start_year | runtime_minutes | ordering | is_original_title | averagerating | numvotes |
|---|---|---|---|---|---|---|
| **start_year** | 1.000000 | 0.001729 | NaN | NaN | NaN | NaN |
| **runtime_minutes** | 0.001729 | 1.000000 | NaN | NaN | NaN | NaN |
| **ordering** | NaN | NaN | 1.000000 | -0.111053 | NaN | NaN |
| **is_original_title** | NaN | NaN | -0.111053 | 1.000000 | NaN | NaN |
| **averagerating** | NaN | NaN | NaN | NaN | 1.000000 | 0.044478 |
| **numvotes** | NaN | NaN | NaN | NaN | 0.044478 | 1.000000 |

In [127]:
```
#visualizing the correlation using heatmap
sns.heatmap(combined_df.corr(),annot=True)
```

Out[127]: <AxesSubplot:>



In [128]:
```
#grouping the data by title
combined_df.groupby('title').mean().sort_values(by="ordering",ascending=False)
```

Out[128]:

| title | start_year | runtime_minutes | ordering | is_original_title | averagerating | numv |
|---|---|---|---|---|---|---|
| **Žvaigždžiu karai: galia nubunda** | NaN | NaN | 61.0 | 0.0 | NaN | |
| **Star Wars: Güç Uyaniyor** | NaN | NaN | 60.0 | 0.0 | NaN | |
| **Star Wars: Episódio VII - O Despertar da Força** | NaN | NaN | 59.0 | 0.0 | NaN | |
| **Star Wars: Das Erwachen der Macht** | NaN | NaN | 57.0 | 0.0 | NaN | |
| **Star Wars: O Despertar da Força** | NaN | NaN | 56.5 | 0.0 | NaN | |
| **...** | ... | ... | ... | ... | ... | |
| **Diese verfluchten Stunden am Abend - Häftlingsbordelle im KZ** | NaN | NaN | 1.0 | 0.0 | NaN | |
| **Muodonmuutoksia** | NaN | NaN | 1.0 | 1.0 | NaN | |
| **Dieser eine gemeinsame Tag** | NaN | NaN | 1.0 | 0.0 | NaN | |
| **I principi dell'Indeterminazione: Il Boia** | NaN | NaN | 1.0 | 0.0 | NaN | |
| **Dreckiges Blut - Die Transfusion des Bösen** | NaN | NaN | 1.0 | 0.0 | NaN | |

252781 rows × 6 columns

# Step 5:Ask A Question About The Data

```
Answer a question about the data using a plot or statistic
Questions:
What is the most watched movie by category?
Who is the most loved character?
What is the most watched movie by title?
```

In [129]:
```
combined_df.head()
```

Out[129]:

| gion | language | types | attributes | is_original_title | averagerating | numvotes | primary_name | birth_yea |
|---|---|---|---|---|---|---|---|---|
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |

In [130]:
```
combined_df['genres'].value_counts()
```

Out[130]:
```
Documentary                     32185
Drama                           21486
Comedy                           9177
Horror                           4372
Comedy,Drama                     3519
                                ...
Biography,Family,Fantasy            1
Sport,Talk-Show                     1
Animation,Mystery,Thriller          1
Animation,Music,Mystery             1
Mystery,Reality-TV,Thriller         1
Name: genres, Length: 1085, dtype: int64
```

In [ ]:
```
#The most watched movie by category is Documentaries
```

In [131]:
```
combined_df['title'].value_counts()
```

Out[131]:
```
Robin Hood                       32
Home                             30
Alone                            27
Love                             25
Thor                             25
                                 ..
Hashima                           1
Koinowa: Konkatsu Cruising        1
Iskušenik                         1
Любить, пить и петь               1
O Mensageiro dos Espíritos 2      1
Name: title, Length: 252781, dtype: int64
```

In [ ]:
```
#The most watched movie by title is Robin Hood
```

In [136]:
```
combined_df['primary_name'].value_counts()
```

Out[136]:
```
James Brown        16
Michael Brown      16
David Brown        15
Michael Johnson    14
Dinesh             13
                   ..
Daniel Vitalis      1
Nikhil Upreti       1
Jean Law            1
Mark Ashton         1
Yanjia Chen         1
Name: primary_name, Length: 577203, dtype: int64
```

In [ ]:
```
#The most loved character is James Brown
```

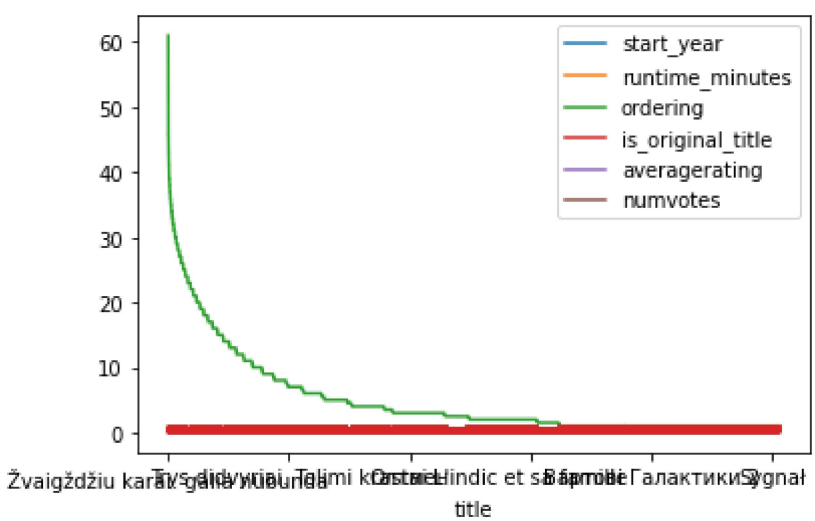
In [135]:
```
combined_df.mode()
```

Out[135]:

| | movie_id | primary_title | original_title | start_year | runtime_minutes | genres | person_id | or |
|---|---|---|---|---|---|---|---|---|
| **0** | tt4050462 | Home | Broken | 2017.0 | 90.0 | Documentary | nm6935209 | |
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

In [143]:
```
#visualizing the most ordered movie
df2=combined_df.groupby('title').mean().sort_values(by="ordering",ascending=Fal
df2.plot()
```

Out[143]:
```
<AxesSubplot:xlabel='title'>
```



# Step 6: Recommendation

In [ ]:
```
From the analysis of the movie data sets, the below are the findings:
    1. Most people prefered watching documentaries as compared to other genres.
    2. The most loved movie was 'Broken' Featuring James Brown
    3. The most watched movie was Titled Robin Hood.
Recommendation to Microsoft.
1.I would recommend microsoft to venture into documentaries as this will bring
2.They should incorporate the top three most loved characters who are James Bro
3.This is definitely a viable business.
```