Student Name:Nzula Priscilla Malombe

Student Pace: Part Time

Scheduled Project Review Date/Time: 5.11.2023 8.00pm

Instructor Name: Samwel Jane

In [2]:
```
#import the relevant libraries
#import pandas as pd
#import numpy as np
#import matplotlib.pyplot as plt
#import seaborn as sns
#%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

to get the file path, use !ls

In [3]:
```
!ls
```

```
CONTRIBUTING.md
LICENSE.md
Presentation Slides project 1.pdf
Presentation Slides project 1.pptx
README.md
awesome.gif
bom.movie_gross.csv
github.pdf
notebook.pdf
rt.reviews.csv
rt.reviews.tsv
student.ipynb
title.basics.csv
title.ratings.csv
zippedData
~$Presentation Slides project 1.pptx
```

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don’t know anything about creating movies. You are charged with exploring what types of films are currently doing the best at the box office. You must then translate those findings into actionable insights that the head of Microsoft's new movie studio can use to help decide what type of films to create.

Data Understanding

The data is contained in a zippedData file wwhich comprises of various movies datasets. We will therefore work with three data files;

1.imdb.title.basics

2.imdb.title.ratings

3.bom.movie_gross

The title.basics.csv file contains 146144 rows and 6 columns which represent the basic information about the movies. Like the genre, title and the start year. The title.ratings.csv file contains 73856 rows and 3 columns which show the averagerating of the movies and the number of votes casted. The movie_gross.csv file contains 3387 rows and 5 columns which basically shows the income generated by various movies. It shows income made both domestically and internationally. We will focus on the follwing features;averagerating,numvotes,domestic_gross, genres and foreign_gross. To get the type of films that are doing the best.

Ratings

Load the title.rating.csv file as df_rating.

In [4]:
```
#Loading the first datasets.
# df_rating

df_rating = pd.read_csv('title.ratings.csv')
df_rating
```

Out[4]:

	tconst	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 × 3 columns

In [5]:
```
#Check for the df_rating data type
#the column data types
#use .dtypes
df_rating.dtypes
```

Out[5]:
```
tconst           object
averagerating    float64
numvotes           int64
dtype: object
```

In [6]:
```
# check for the overview of the rating table
# use .info
df_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   tconst         73856 non-null  object
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

In [7]:
```
# Check for missing values- NULL values
# use .isna().sum()
df_rating.isna().sum()
```

Out[7]:
```
tconst           0
averagerating    0
numvotes         0
dtype: int64
```

In [8]:
```
# check for duplicated vaues
# use .duplicated()
df_rating.duplicated().sum()
```

Out[8]:
```
0
```

There are no missing values and duplicates for df_rating.

In [9]:
```
#To check for the number of rows and columns in the rating table, use .shape
df_rating.shape
```

Out[9]:
```
(73856, 3)
```

Movie Gross

Load the bom.movies_gross.csv file as df_movies.

In [10]:
```
#loading the second data set.
#df_movies

df_movies = pd.read_csv('bom.movie_gross.csv')
df_movies
```

Out[10]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

In [11]:
```
#To check for the number of rows and columns in the rating table
# use .shape
df_movies.shape
```

Out[11]:
```
(3387, 5)
```

In [12]:
```
# check for the overview of the rating table
# use .info
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

In [13]:
```
# Check for missing values- NULL values
# use .isna().sum()
df_movies.isna().sum()
```

Out[13]:
```
title               0
studio              5
domestic_gross     28
foreign_gross    1350
year                0
dtype: int64
```

In [14]:
```
# check for duplicated vaues
# use .duplicated
df_movies.duplicated().sum()
```

Out[14]:
```
0
```

There are no duplicates values. However, there are missing values for three columns in df_movies .

Basics

Load the title.basics.csv file as df_basics.

In [15]:
```
# Loading the third data sets.
# df_basics
df_basics = pd.read_csv('title.basics.csv')
df_basics
```

Out[15]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	NaN
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

In [16]:
```
#To check for the number of rows and columns in the rating table
# use .shape
df_basics.shape
```

Out[16]: (146144, 6)

In [17]:
```
# check for the overview of the rating table
# use .info
df_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   primary_title    146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [18]:
```
# Check for missing values- NULL values
# use .isna().sum()
df_basics.isna().sum()
```

Out[18]:
```
tconst                0
primary_title         0
original_title       21
start_year            0
runtime_minutes    31739
genres              5408
dtype: int64
```

In [19]:
```
# check for duplicated vaues
# use .duplicated
df_movies.duplicated().sum()
```

Out[19]:
```
0
```

There are no duplicated values. However, there are missing values for three columns in df_basics.

Data Cleaning and Data Preparation

There are columns that have missing values such as; studio,domestic_gross and foreign_gross in df_movies. genres, runtime_minutes and original_title in df_basics.

Handling the missing values.

Fill the domestic_gross column ad foreign_gross column in df_movies with 0. This is because fillig both columns with either mean or median would alter the data set completely. A movie that did not yield returns can not be assumed to have contributed to the gross. Also filling the columns with mean or median would result to a biased analysis and wrong decision making by Microsoft.Hence, the financial losses. Drop the studio column.

For the df_basics data set, we drop the rows where the columns- genres,original_title and runtime_minutes have missing values.

Cleaning the df_movies data set to get cleaned_df_movies data set.

In [20]:
```
#Fill the domestic_gross column in df_movies with 0
# use.replace(), inplace = True, regex = False.
df_movies['domestic_gross'].replace(np.nan, 0, inplace = True, regex = False)
df_movies
```

Out[20]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

In [21]:
```
#Fill the foreign_gross column  in df_movies with 0
# use.replace(), inplace = True, regex = False.
df_movies['foreign_gross'].replace(np.nan, 0, inplace = True, regex = False)
df_movies
```

Out[21]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	0	2018
3383	Edward II (2018 re-release)	FM	4800.0	0	2018
3384	El Pacto	Sony	2500.0	0	2018
3385	The Swan	Synergetic	2400.0	0	2018
3386	An Actor Prepares	Grav.	1700.0	0	2018

3387 rows × 5 columns

In [22]:
```
# Drop the studio column in df_movies
# use .drop(),axis=1, inplace =True
df_movies.drop('studio',axis = 1,inplace = True)
df_movies
```

Out[22]:

	title	domestic_gross	foreign_gross	year
0	Toy Story 3	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	296000000.0	664300000	2010
3	Inception	292600000.0	535700000	2010
4	Shrek Forever After	238700000.0	513900000	2010
...	...	...	...	...
3382	The Quake	6200.0	0	2018
3383	Edward II (2018 re-release)	4800.0	0	2018
3384	El Pacto	2500.0	0	2018
3385	The Swan	2400.0	0	2018
3386	An Actor Prepares	1700.0	0	2018

3387 rows × 4 columns

In [23]:
```
# cleaned df_movies data set = cleaned_df_movies
cleaned_df_movies = df_movies
cleaned_df_movies
```

Out[23]:

	title	domestic_gross	foreign_gross	year
0	Toy Story 3	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	296000000.0	664300000	2010
3	Inception	292600000.0	535700000	2010
4	Shrek Forever After	238700000.0	513900000	2010
...	...	...	...	...
3382	The Quake	6200.0	0	2018
3383	Edward II (2018 re-release)	4800.0	0	2018
3384	El Pacto	2500.0	0	2018
3385	The Swan	2400.0	0	2018
3386	An Actor Prepares	1700.0	0	2018

3387 rows × 4 columns

Cleaning the df_basics data set to get cleaned_df_basics.

In [24]:
```
# drop the rows where the columns-genres has missing values.
# use.dropna()
df_basics.dropna(axis = 0, subset = ['genres'], inplace = True)
```

In [25]:
```
# drop the rows where the columns-runtime_minutes has missing values.
# use.dropna()
df_basics.drop('runtime_minutes', axis = 1, inplace = True)
```

In [26]:
```
# drop the columns-original_title.
# use.drop()
df_basics.drop('original_title', axis = 1,inplace = True)
```

In [27]:
```
# drop the columns-start_year also because it is not very necessary in our anan
# use.drop()
df_basics.drop('start_year', axis = 1,inplace = True)
```

In [28]:
```
cleaned_df_basics = df_basics
cleaned_df_basics
```

Out[28]:

	tconst	primary_title	genres
0	tt0063540	Sunghursh	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Biography,Drama
2	tt0069049	The Other Side of the Wind	Drama
3	tt0069204	Sabse Bada Sukh	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	Comedy,Drama,Fantasy
...	...	...	...
146138	tt9916428	The Secret of China	Adventure,History,War
146139	tt9916538	Kuambil Lagi Hatiku	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Documentary
146141	tt9916706	Dankyavar Danka	Comedy
146143	tt9916754	Chico Albuquerque - Revelações	Documentary

140736 rows × 3 columns

In [29]:
```
# check whether the cleaned_df_basics is truly cleaned.
# use .isna().sum()
cleaned_df_basics.isna().sum()
```

Out[29]:
```
tconst          0
primary_title   0
genres          0
dtype: int64
```

In [30]:
```
# check whether the cleaned_df_movies is truly cleaned.
# use .isna().sum()
cleaned_df_movies.isna().sum()
```

Out[30]:
```
title             0
domestic_gross    0
foreign_gross     0
year              0
dtype: int64
```

Merging of Data sets

Merge df_rating and cleaned_df_basics. This is because the two data sets share a similar primary key - tconst.

In [31]:
```
# To merge use .merge()
df_rating_basics = pd.merge(df_rating,cleaned_df_basics)
df_rating_basics
```

Out[31]:

	tconst	averagerating	numvotes	primary_title	genres
0	tt10356526	8.3	31	Laiye Je Yaarian	Romance
1	tt10384606	8.9	559	Borderless	Documentary
2	tt1042974	6.4	20	Just Inès	Drama
3	tt1043726	4.2	50352	The Legend of Hercules	Action,Adventure,Fantasy
4	tt1060240	6.5	21	Até Onde?	Mystery,Thriller
...	...	...	...	...	...
73047	tt9805820	8.1	25	Caisa	Documentary
73048	tt9844256	7.5	24	Code Geass: Lelouch of the Rebellion - Glorifi...	Action,Animation,Sci-Fi
73049	tt9851050	4.7	14	Sisters	Action,Drama
73050	tt9886934	7.0	5	The Projectionist	Documentary
73051	tt9894098	6.3	128	Sathru	Thriller

73052 rows × 5 columns

In [32]:
```
# check for missing values.
df_rating_basics.isna().sum()
```

Out[32]:
```
tconst           0
averagerating    0
numvotes         0
primary_title    0
genres           0
dtype: int64
```

Renaming

To merge the df_rating_basics with cleaned_df_movies, we rename the column 'primary_title'. This means the column 'tilte' in cleaned_df_movies acts as the primary key and is similar to 'primary_title' in df_rating_basics. Renaming the column 'primary_title' to 'title' makes it easy to merge.

In [33]:
```
# Rename using .rename()
#df_merged.rename(columns ={'primary_title': 'title'}, inplace = True)
df_rating_basics.rename(columns = {'primary_title' : 'title'}, inplace = True)
```

In [34]:
```
# check that the column 'primary_title' has been renamed.
# call df-rating_basics
df_rating_basics
```

Out[34]:

	tconst	averagerating	numvotes	title	genres
0	tt10356526	8.3	31	Laiye Je Yaarian	Romance
1	tt10384606	8.9	559	Borderless	Documentary
2	tt1042974	6.4	20	Just Inès	Drama
3	tt1043726	4.2	50352	The Legend of Hercules	Action,Adventure,Fantasy
4	tt1060240	6.5	21	Até Onde?	Mystery,Thriller
...	...	...	...	...	...
73047	tt9805820	8.1	25	Caisa	Documentary
73048	tt9844256	7.5	24	Code Geass: Lelouch of the Rebellion - Glorifi...	Action,Animation,Sci-Fi
73049	tt9851050	4.7	14	Sisters	Action,Drama
73050	tt9886934	7.0	5	The Projectionist	Documentary
73051	tt9894098	6.3	128	Sathru	Thriller

73052 rows × 5 columns

In [35]:
```
# Use .merge() to merge the df_rating_basics and cleaned_df_movies data sets.
# Name the finale data set as df_rating_bascis_movies
df_rating_basics_movies = pd.merge(df_rating_basics,cleaned_df_movies)
df_rating_basics_movies
```

Out[35]:

	tconst	averagerating	numvotes	title	genres	domestic_g
0	tt1043726	4.2	50352	The Legend of Hercules	Action,Adventure,Fantasy	18800(
1	tt1171222	5.1	8296	Baggage Claim	Comedy	21600(
2	tt1181840	7.0	5494	Jack and the Cuckoo-Clock Heart	Adventure,Animation,Drama	
3	tt1210166	7.6	326657	Moneyball	Biography,Drama,Sport	75600(
4	tt1212419	6.5	87288	Hereafter	Drama,Fantasy,Romance	32700(
...	...	...	...	...	...	
3015	tt3399916	6.3	4185	The Dead Lands	Action,Adventure	52
3016	tt3616916	6.7	28167	The Wave	Action,Drama,Thriller	177(
3017	tt3748512	7.4	4977	Hitchcock/Truffaut	Documentary	260(
3018	tt7008872	7.0	18768	Boy Erased	Biography,Drama	68000
3019	tt7048622	7.7	11168	The Insult	Crime,Drama,Thriller	10000

3020 rows × 8 columns

In [36]:
```
# checking for data types for df_rating_basics_movies data set.
#use .dtypes
df_rating_basics_movies.dtypes
```

Out[36]:
```
tconst             object
averagerating     float64
numvotes            int64
title              object
genres             object
domestic_gross    float64
foreign_gross      object
year                int64
dtype: object
```

In [37]:
```
# check the contents of column 'foreign_gross'
df_rating_basics_movies['foreign_gross']
```

Out[37]:
```
0       42400000
1         887000
2        3400000
3       34600000
4       72500000
          ...
3015           0
3016           0
3017           0
3018     5000000
3019           0
Name: foreign_gross, Length: 3020, dtype: object
```

In [38]:
```
# check the column 'domestic_gross'
df_rating_basics_movies['domestic_gross']
```

Out[38]:
```
0       18800000.0
1       21600000.0
2              0.0
3       75600000.0
4       32700000.0
             ...
3015        5200.0
3016      177000.0
3017      260000.0
3018     6800000.0
3019     1000000.0
Name: domestic_gross, Length: 3020, dtype: float64
```

The columns domestic_gross and foreign_gross have different data types. We change the data type of foreign_gross column to 'float64'.

In [39]:
```
# Convert the datatype of column 'foreign_gross' to float.
#use pd.to_numeric()
# sample df['column_name'] = pd.to_numeric(df['column_name'])
df_rating_basics_movies['foreign_gross']= pd.to_numeric(df_rating_basics_movies
```

In [40]:
```
# checking for data types for df_rating_basics_movies data set.
#See if they now have similar data type
#use .dtypes
df_rating_basics_movies.dtypes
```

Out[40]:
```
tconst             object
averagerating     float64
numvotes            int64
title              object
genres             object
domestic_gross    float64
foreign_gross     float64
year                int64
dtype: object
```

In [41]:
```
# Combine the domestic_gross column and foreign_gross column
# as total-gross-income
# sample df['combined col'] = df['col1'] + df['col2']
df_rating_basics_movies['total_gross_income'] = df_rating_basics_movies['domest
```

In [42]:
```
# The datatype float is not suitable to use
# convert the column total_gross-income data type to object.
# use df['col'] = df['col'].astype(str)
df_rating_basics_movies['total_gross_income'] = df_rating_basics_movies['total_
```

In [43]:
```
# check if the new column total_gross_income data type was converted.
# load data set
df_rating_basics_movies.dtypes
```

Out[43]:
```
tconst                object
averagerating        float64
numvotes               int64
title                 object
genres                object
domestic_gross       float64
foreign_gross        float64
year                   int64
total_gross_income    object
dtype: object
```

In [44]:
```
# For easy analysis , convert total gross income data type back to float64
# Converting total_income_gross dtype into 'float64'
# Use .astype()
df_rating_basics_movies['total_gross_income']= df_rating_basics_movies['total_g
```

In [45]:
```
# Checking for null values
# use .isna(), .sum()
df_rating_basics_movies.isna().sum()
```

Out[45]:
```
tconst               0
averagerating        0
numvotes             0
title                0
genres               0
domestic_gross       0
foreign_gross        4
year                 0
total_gross_income   4
dtype: int64
```

In [46]:
```
# Replace all the null values with '0'
# Use .replace()
df_rating_basics_movies['total_gross_income'].replace(np.nan, 0, inplace = True
df_rating_basics_movies
```

Out[46]:

	tconst	averagerating	numvotes	title	genres	domestic_g
0	tt1043726	4.2	50352	The Legend of Hercules	Action,Adventure,Fantasy	18800(
1	tt1171222	5.1	8296	Baggage Claim	Comedy	21600(
2	tt1181840	7.0	5494	Jack and the Cuckoo-Clock Heart	Adventure,Animation,Drama	
3	tt1210166	7.6	326657	Moneyball	Biography,Drama,Sport	75600(
4	tt1212419	6.5	87288	Hereafter	Drama,Fantasy,Romance	32700(
...	...	...	...	...	...	
3015	tt3399916	6.3	4185	The Dead Lands	Action,Adventure	52
3016	tt3616916	6.7	28167	The Wave	Action,Drama,Thriller	177(
3017	tt3748512	7.4	4977	Hitchcock/Truffaut	Documentary	260(
3018	tt7008872	7.0	18768	Boy Erased	Biography,Drama	6800(
3019	tt7048622	7.7	11168	The Insult	Crime,Drama,Thriller	10000

3020 rows × 9 columns

In [47]:
```
# drop the columns, domestic_gross,foreign_gross,tconst also because it is not
# use.drop()
df_rating_basics_movies.drop('domestic_gross', axis = 1,inplace = True)
df_rating_basics_movies.drop('foreign_gross', axis = 1,inplace = True)
df_rating_basics_movies.drop('tconst', axis = 1,inplace = True)
```

In [48]:
```
# Split genres column to remove the repeated genres
# use.split()
df_rating_basics_movies['genres'] = df_rating_basics_movies['genres'].str.split
df_rating_basics_movies
```

Out[48]:

	averagerating	numvotes	title	genres	year	total_gross_income
0	4.2	50352	The Legend of Hercules	Action	2014	61200000.0
1	5.1	8296	Baggage Claim	Comedy	2013	22487000.0
2	7.0	5494	Jack and the Cuckoo-Clock Heart	Adventure	2014	3400000.0
3	7.6	326657	Moneyball	Biography	2011	110200000.0
4	6.5	87288	Hereafter	Drama	2010	105200000.0
...	...	...	...	...	...	...
3015	6.3	4185	The Dead Lands	Action	2015	5200.0
3016	6.7	28167	The Wave	Action	2016	177000.0
3017	7.4	4977	Hitchcock/Truffaut	Documentary	2015	260000.0
3018	7.0	18768	Boy Erased	Biography	2018	11800000.0
3019	7.7	11168	The Insult	Crime	2018	1000000.0

3020 rows × 6 columns

Data Exploration and Analysis

Use .groupby, .sort and aggregate functions to analyse the data. Explore the types of genres with their corresponding total income, average rating and the number of votes. The most popular genre being the one with the highest number of votes. The best genres being the ones with the highest average rating. The most profitable being the one that has the highest total gross income generated.

In [49]:
```
# We look at the diffrent genres, their averagerating,numvotes and the total_gr

df= df_rating_basics_movies.groupby(['genres','averagerating','numvotes'])['tot
df
```

Out[49]:
```
genres    averagerating  numvotes
Action    1.7            7384          271000.0
          2.9            2325          274000.0
          3.0            60          82100000.0
          3.4            2295           99600.0
          3.6            201            44800.0
                                          ...
Thriller  6.6            5           68600000.0
          6.8            6          133500000.0
          7.1            36            970000.0
          7.2            59          22400000.0
          8.0            8           286200000.0
Name: total_gross_income, Length: 3013, dtype: float64
```

To determine the type of genre that is highly watched , we look at the genre that has the highest number of votes.

In [50]:
```
# Use.groupby and sort in descending order
df_rating_basics_movies.groupby(['genres'])['numvotes'].max().sort_values(ascer

# Action,Adventure, Sci-Fri genres has the highest number of votes.
# Action,Adventure, Sci-Fri genres is therefore highly watched while Documentar
```

Out[50]:
```
genres
Action         1841066
Adventure      1299334
Drama          1211405
Biography      1035358
Mystery        1005960
Comedy          621018
Crime           553156
Animation       464511
Horror          400474
Romance         227616
Fantasy         102369
Documentary      74978
Thriller         16318
Music            15592
Sci-Fi            3501
Family             132
Sport               77
Name: numvotes, dtype: int64
```

Look for the genres with the highest averagerating.

In [51]:
```
# Use.groupby and sort in descending order
df_rating_basics_movies.groupby(['genres'])['averagerating'].max().sort_values(

#Adventure,Documentary and Drama respectively have the highest averageratings.
```

Out[51]:
```
genres
Documentary    9.2
Adventure      9.2
Drama          9.1
Comedy         8.9
Action         8.8
Biography      8.6
Crime          8.5
Animation      8.4
Fantasy        8.2
Mystery        8.1
Thriller       8.0
Sport          7.9
Horror         7.7
Romance        7.6
Family         7.3
Music          7.2
Sci-Fi         5.9
Name: averagerating, dtype: float64
```

In [52]:
```
#check for the genres that have the highest income
# use .groupby and aggregate function .max()
df_rating_basics_movies.groupby(['genres'])['total_gross_income'].max()
#Action genres has the highest income contribution.
# folllwed by Adventure.
```

Out[52]:
```
genres
Action         1.405400e+09
Adventure      1.276400e+09
Animation      6.342000e+08
Biography      9.036000e+08
Comedy         5.868000e+08
Crime          8.576000e+08
Documentary    8.219000e+08
Drama          7.232000e+08
Family         8.576000e+08
Fantasy        1.276400e+09
Horror         8.071000e+08
Music          8.000000e+06
Mystery        2.948000e+08
Romance        1.279000e+08
Sci-Fi         8.219000e+08
Sport          5.300000e+06
Thriller       2.862000e+08
Name: total_gross_income, dtype: float64
```

In [96]:
```
# to check for the title of movies in various genres.
# use .groupby.
df_rating_basics_movies.groupby(['title'])['genres'].max()
```

Out[96]:
```
title
'71                             Action
1,000 Times Good Night           Drama
10 Cloverfield Lane              Drama
10 Years                        Comedy
1001 Grams                       Drama
                                ...
Zindagi Na Milegi Dobara        Comedy
Zombeavers                      Action
Zookeeper                       Comedy
Zoolander 2                     Comedy
Zootopia                     Adventure
Name: genres, Length: 2598, dtype: object
```

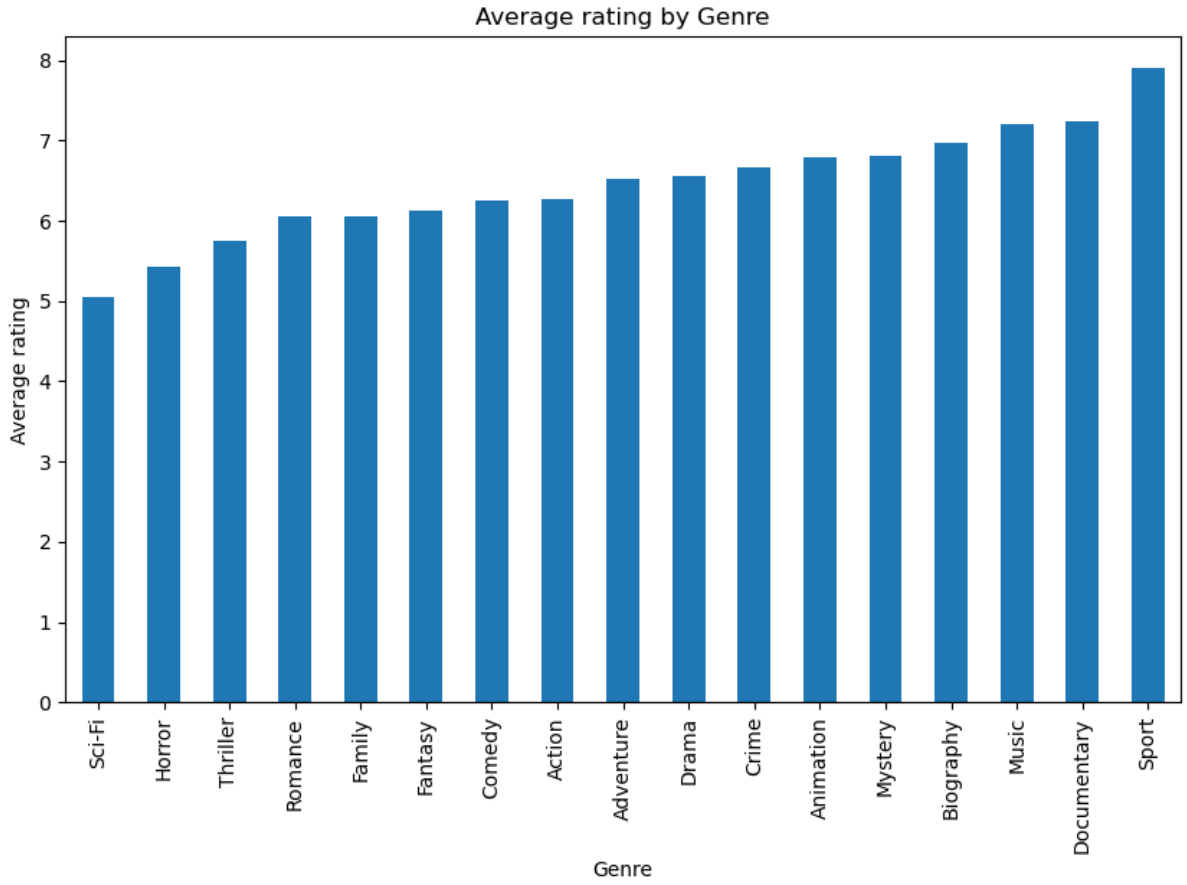
In [54]:
```
# Drama genres is the most popular .
df_rating_basics_movies['genres'].value_counts().sort_values(ascending = False)
```

Out[54]:
```
Drama          777
Action         646
Comedy         617
Biography      251
Adventure      214
Documentary    160
Crime          140
Horror         111
Animation       36
Thriller        27
Fantasy         13
Mystery         10
Romance          9
Family           5
Sci-Fi           2
Sport            1
Music            1
Name: genres, dtype: int64
```

Data Visualization

We visualize the data to help in strategic analysis by making the data more understable and easy to discover various trends. Using matplotlib.pyplot as plt,seaborn as sns and %matplotlib inline libraries. We plot bar graphs and scatterplots that show the trends of various genres. The bar graphs shows the relationship between genres, average rating and number of votes. The scatterplot show the relationship between genres, total gross income and average rating. The bar and scatterplots help in communicating insights to the microsoft stakeholders to help tehm venture in the right type of films.

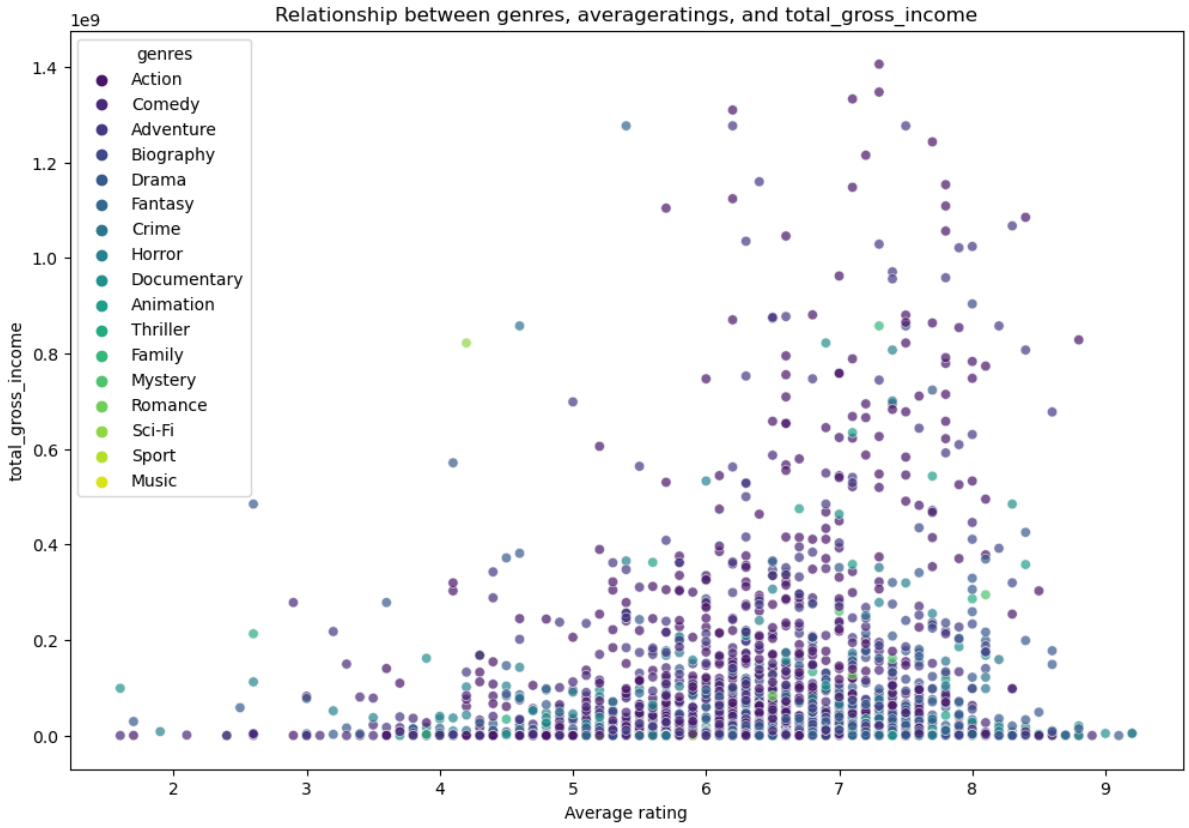
In [55]:
```
# Plot a bar that shows the relationship between genres and the average ratings
#create genres average rating plot
genre_avg_rating = df_rating_basics_movies.groupby('genres')['averagerating'].m

# Plot the bar chart
# figsize = 12,8
plt.figure(figsize=(10,6))

genre_avg_rating.plot(kind='bar')
# labeling the title of the bar.
plt.title('Average rating by Genre')
# label the x- axis
plt.xlabel('Genre')
# label the y- axis
plt.ylabel('Average rating')
# display the bar
plt.show()
```



As per the bar above the Sport and Adventure genres have the highest average ratings while Sci-Fi genres has the lowest average rating.

Most people love the Adventure and Sport genres .

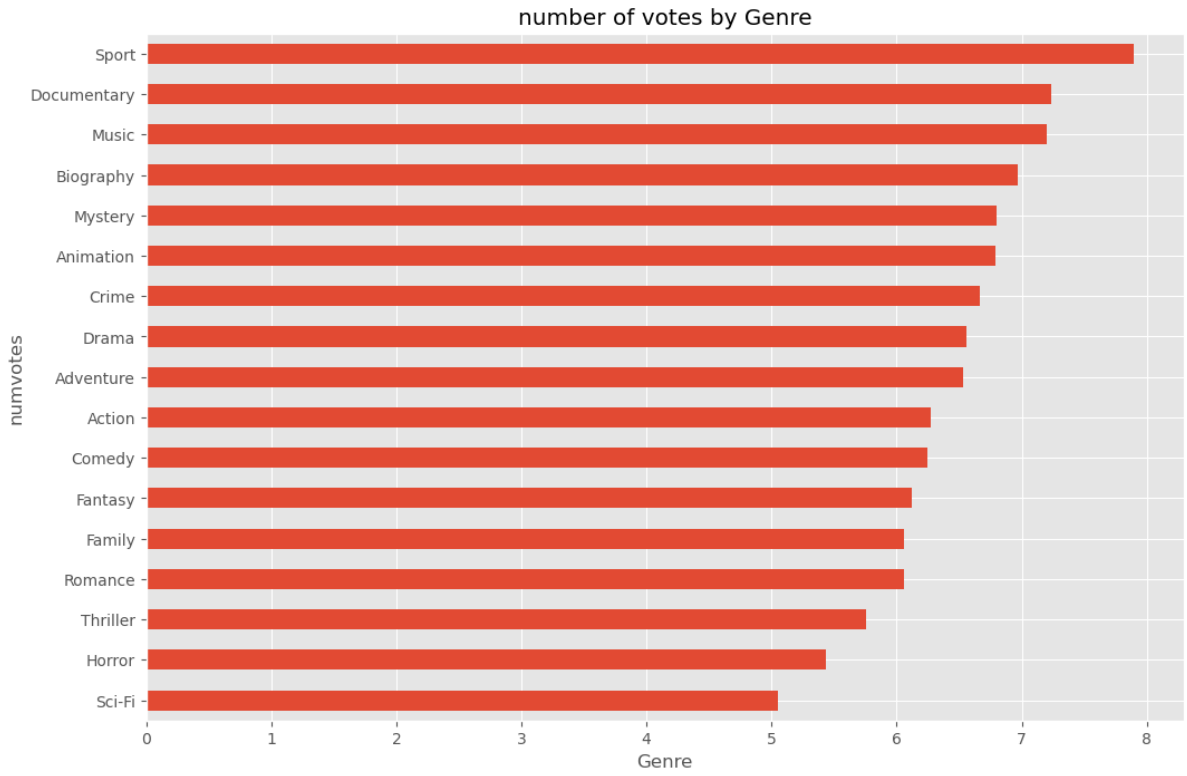
In [56]:
```
# using sns plot a scatter
# plot to show relationship between genres ,average rating and total gross inco
# figsize = 12,8
plt.figure(figsize=(12,8))
# create sns.scatterplot
sns.scatterplot(x='averagerating', y='total_gross_income', hue='genres', data=d
#label title of the plot
plt.title('Relationship between genres, averageratings, and total_gross_income'
# label the x-axis
plt.xlabel('Average rating')
# label y-axis
plt.ylabel('total_gross_income')
#display the plot
plt.show()
```



From the scatterplot above,the adventure, documentary ,drama, comedy and action genres have the highest rating.

While action and adventure having the highest total gross income.

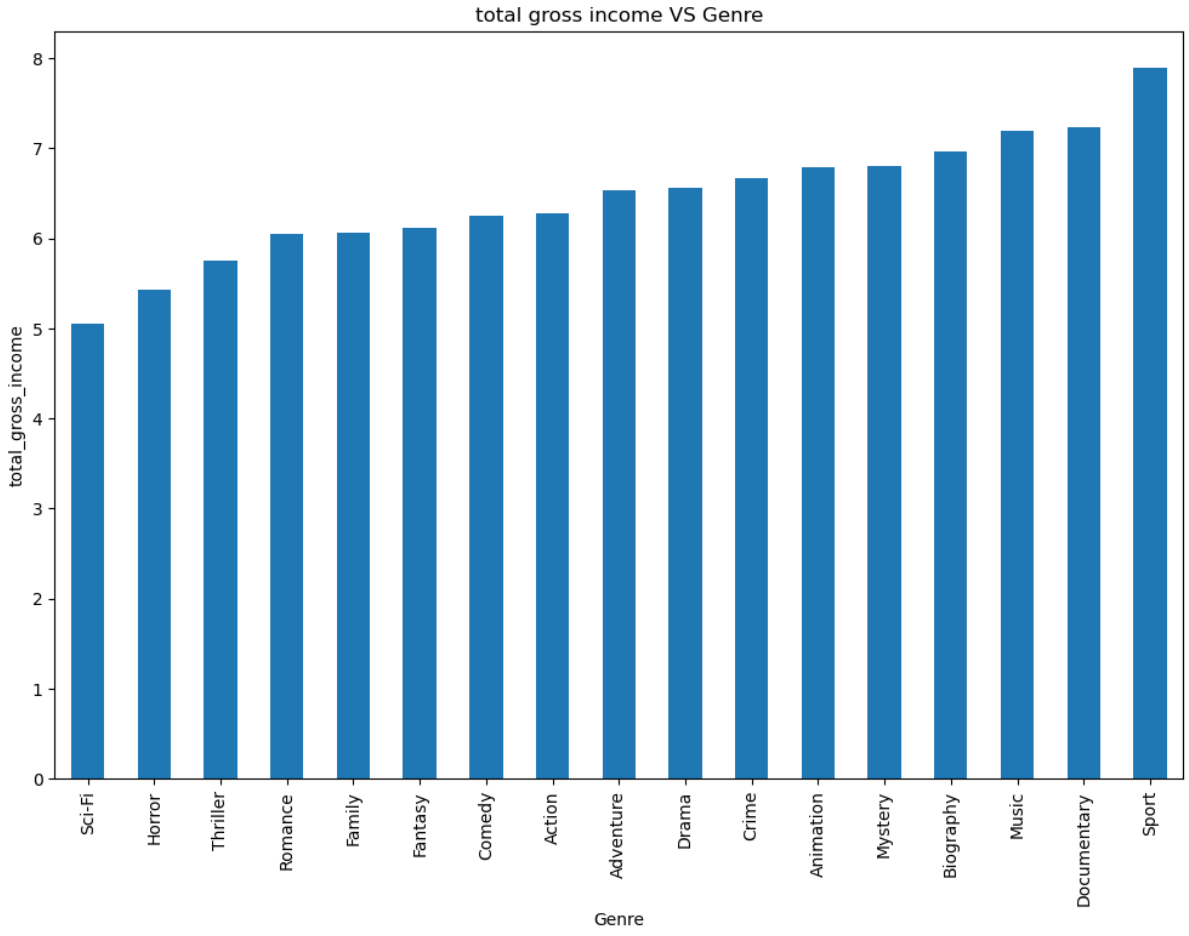
In [71]:
```
# create barplot  for genres avearage rating vs number of votes
genre_numvotes = df_rating_basics_movies.groupby('genres')['numvotes'].mean().s


# figsize = 12,8
plt.figure(figsize=(12,8))
# Plot the bar chart
genre_avg_rating.plot(kind='barh')
# label the title of the bar
plt.title('number of votes by Genre')
# label the x-axis
plt.xlabel('Genre')
# label the y-axis
plt.ylabel('numvotes')
# display the bar
plt.show()
```



From the above bar mystery, action, adventure, crime and biography genres have the highes number of votes.

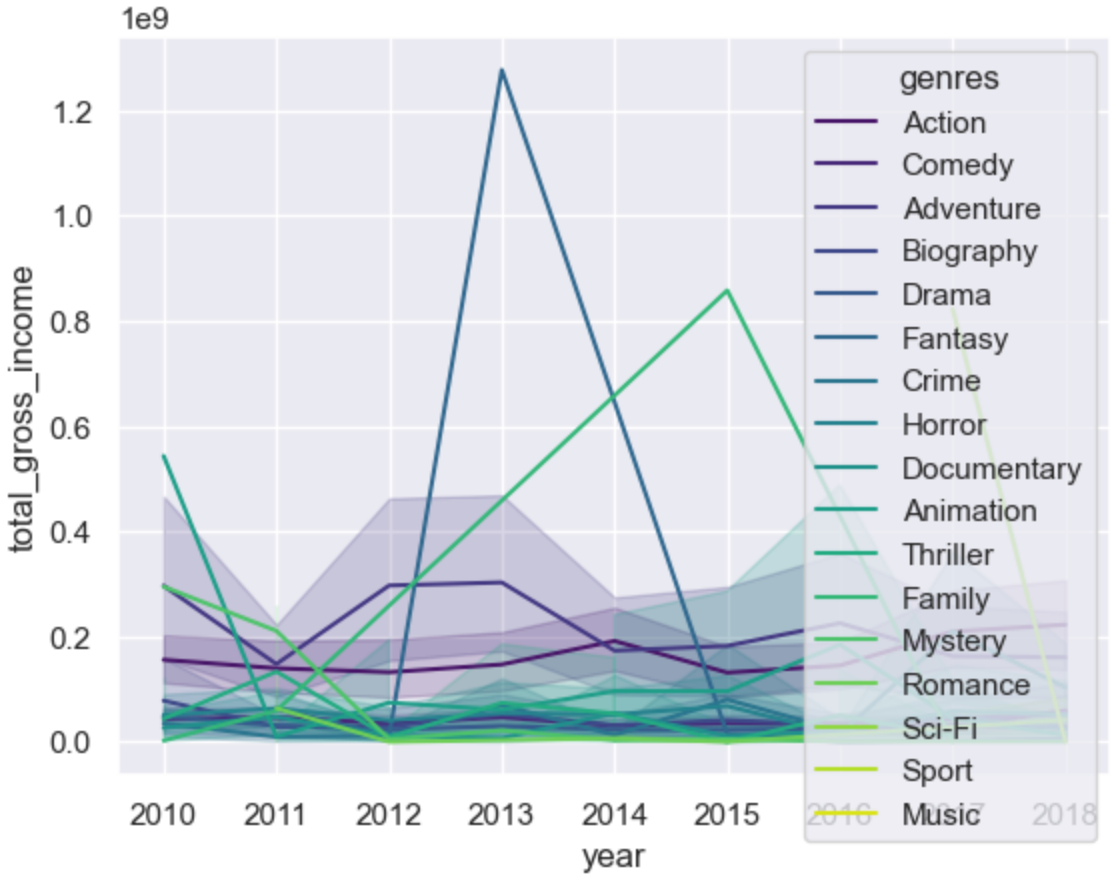
With the family and sport having no votes at all.

In [58]:
```
# create barplot  for genres avearage rating vs total_gross_income
genre_total_gross_income= df_rating_basics_movies.groupby('genres')['total_gros


# figsize = 12,8
plt.figure(figsize=(12,8))
# Plot the bar chart
genre_avg_rating.plot(kind='bar')
# label the title of the bar
plt.title('total gross income VS Genre')
# label the x-axis
plt.xlabel('Genre')
# label the y-axis
plt.ylabel('total_gross_income')
# display the bar
plt.show()
```



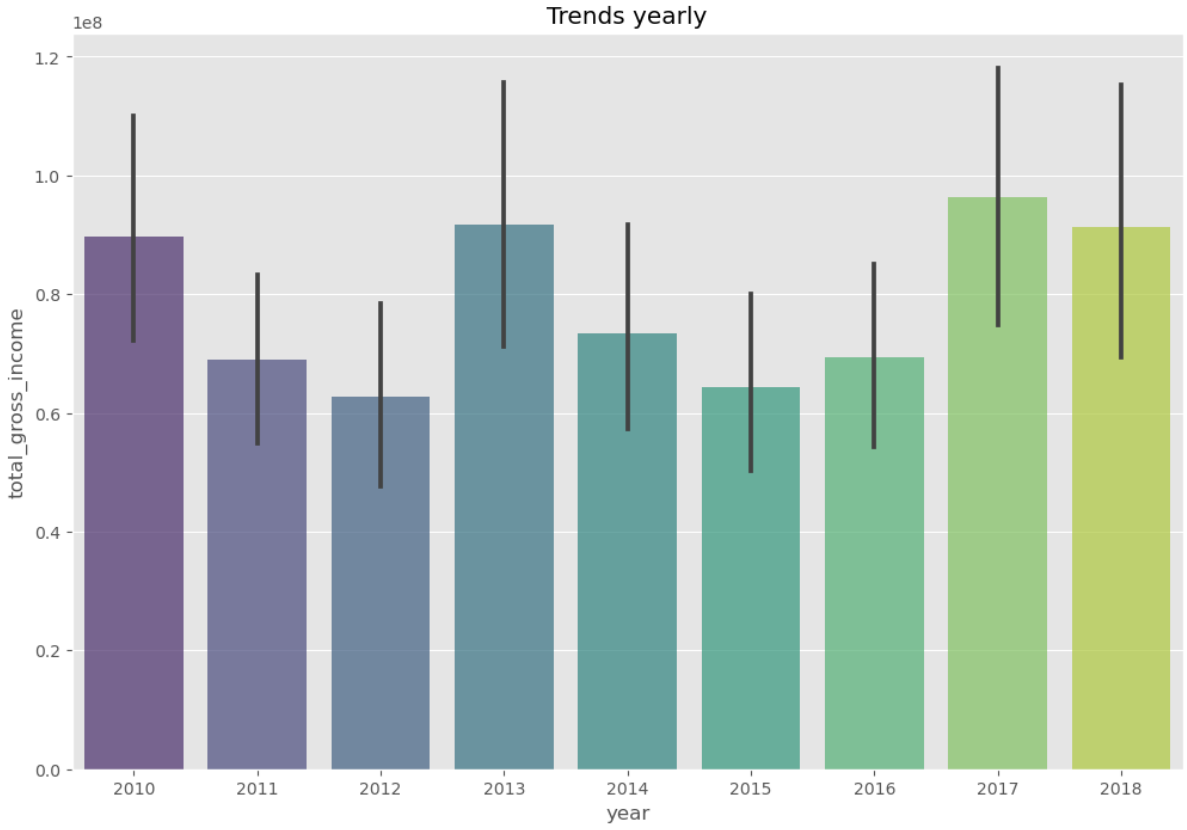
In [93]:
```
#### From the above bar chart, mystery, action, adventure, crime and biography
```

In [92]:
```
# create lineplot
# data=df_rating_basics_movies,hue='genres',palette='viridis'
# label the x and y axis
sns.lineplot(x="year", y="total_gross_income", data=df_rating_basics_movies,hue
#sns.lineplot(x="kepid", y="koi_duration", data= df_rating_basics_movies, alpha
sns.set(style="darkgrid")

plt.show()
```



In [94]:
```
#### From the above sns lineplot, mystery, action, adventure, crime and biograp
#### The total gross income fluctuates with time. One genre could be do well in
#### Hence the need to diversify the genres during creation.
```

In [72]:
```
#plt figure=figsize(12,8)

plt.figure(figsize=(12,8))
# create sns.scatterplot
sns.barplot(x='year', y='total_gross_income',data=df_rating_basics_movies, alph
#label title of the plot
plt.title('Trends yearly')
# label the x-axis
plt.xlabel('year')
# label y-axis
plt.ylabel('total_gross_income')
#display the plot
plt.show()
```



From the above boxplot we note that the total gross income fluctuates with time.

One genre could be do well in a certain year and also perform poorly in another year.

Hence the need to diversify the genres during creation.

Conclusion

From the analysis above, the Microsoft's new studio can create the Adventure, Action, biography , crime and mystery genres . They are the first five genres whose total gross income is very high. And by creating these genres Microsoft is can make good profits both internationally and locally.

Also the mystery ,action ,adventure, crime and biography genres have the highest votes. Meaning they are also preferable by the targeted clientele. However, the total gross income generated by each genre is not constant. There are years a certain genre like action will yield more income and other years that the income will be low. Hence the need for Microsoft to diversify the genres they create to avoid losses. For additional genres the stakeholders can include drama and comedy genres. They have a positive good rating

In [ ]: