AbelO2022 / Phase_1_Microsoft_Project Public

View license

0 stars 0 forks

Star Unwatch

Code Issues Pull requests Actions Projects Wiki Security Insig

main

AbelO2022 Final Submission ... 9 minutes ago 14

View code

README.md

## Final Project Submission

Please fill out:

- Student name: Abel Otieno Odhiambo
- Student pace: full time
- Scheduled project review date/time:
- Instructor name: Antonny Muiko
- Blog post URL:

# Your code here - remember to use markdown cells for comments as well!

## MICROSOFT MOVIE STUDIO DATA ANALYSIS PROJECT

# 1.Business Understanding

In my project Microsoft wants to start a movie studio and my analysis is based on my objectives, which will enable microsoft to come up with a profitable competative movie studio.

### Objectives

- Find the top movie genre
- Find the most popular genre
- Calculate profit and loss for a movie
- Find Distribution locally and worldwide
- Find which is the best movie and what are the features of the Movie

# 2.Data Understanding

In my project i need to get data that shows movie categories and sales

### Collecting Our Data

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns import sqlite3 %matplotlib inline import csv

## 2.1 Loading The data sets to see which datasets suits our project.

### 2.1.1 `bom.movie_gross.csv` File

df = pd.read_csv('bom.movie_gross.csv') df.head()

df.shape

df.isna().sum()

### 2.1.2 `tn.movie_budgets.csv` File

df2 = pd.read_csv('tn.movie_budgets.csv',index_col=0) df2.head()

# Convert "$" amounts to int64

df2["production_budget"] = df2.production_budget.str.replace('[$,]',"").astype("int64")
df2["domestic_gross"] = df2.domestic_gross.str.replace('[$,]',"").astype("int64")
df2["worldwide_gross"] = df2.worldwide_gross.str.replace('[$,]',"").astype("int64")

df2.head()

df2.isna().sum()

### 2.1.3 `tmdb.movies.csv` File

df3 = pd.read_csv('tmdb.movies.csv',index_col=0) df3.head()

df3.rename(columns= {'id':'movie_id','title':'movie_title'}, inplace=True) df3.head()

df3.isna().sum()

### 2.1.4 `rt.movie_info.tsv` File

df4 = pd.read_csv('rt.movie_info.tsv', sep='\t') df4.head()

df4.isna().sum()

### 2.1.5 `im.db` File

conn =sqlite3.connect("im.db")

### Selected data data files for my analysis .

- `tmdb.movies.csv`
- `tn.movie_budgets.csv`
- `im.db`
- `bom.movie_gross.csv`

# 2.2 Cleaning the selected data

### 2.2.1 cleaning `tmdb.movies.csv` (df3)

df3 df3.isna().sum()

df3.tail() df3["vote_average"] df3.drop(index=df3[df3["vote_average"] == 0.0].index,inplace=True) df3.drop(index=df3[df3["vote_count"] < 2].index,inplace=True) df3.tail()

df3['release_date'] = pd.to_datetime(df3['release_date']) df3['release_year'] = df3['release_date'].dt.year

df3.head()

### 2.2.2 `tn.movie_budgets.csv` AS (df2)

df2.isna().sum() df2.tail()

df2.info()

df2['release_date'] = pd.to_datetime(df2['release_date']) df2['release_year'] = df2['release_date'].dt.year

### 2.2.3: `bom.movie_gross.csv` (df)

df.head()

df.tail()

df.isna().sum()

df['foreign_gross'] = df['foreign_gross'].replace(np.nan, 0) df['domestic_gross'] = df['domestic_gross'].replace(np.nan, 0) df['studio'] = df['studio'].replace(np.nan, "no_studio")

# checking for changes made

df.isna().sum()

df.info()

# changing foreign gross to int64

# Removing the ","

df["foreign_gross"] = df['foreign_gross'].str.replace(',',"")

# Replace "nan" with 0

df['foreign_gross'] = df['foreign_gross'].replace(np.nan, 0)

# Change type to float64

df['foreign_gross'] = df['foreign_gross'].astype("float64")

df.info()

## 3.1 Analysing `tmdb.movies.csv` AS df3

# Viewing the data once again

df3.head()

# Describing our data

df3.describe()

# view the shape of the data

df3.shape

## Questions for the data set df3

### 3.1.1 Which Movies has the highest vote_rating?

df3.sort_values(by="vote_average", ascending=False)[:10]

# Does the movie average vote affected by it's popularity?

df3["popularity"].corr(df3["vote_average"])

The relationship is a very week positive correlation, so it doesn't affect.

# Does the the vote count affect the rating?

df3["vote_count"].corr(df3["vote_average"])

The movie rating of a movie is not affected by vote count

### 3.1.2: Which movie has the highest popularity?

# Finding the most watched movie

df3.sort_values(by="popularity", ascending=False)[:10]

df3["vote_count"].corr(df3["vote_average"])

### 3.1.3: Which movie has the highest vote count?

df3.sort_values(by="vote_count", ascending=False)[:10]

### 3.1.4: What is the distribution vote_count and year?

sns.set(style="whitegrid") sns.lineplot(data=df3, x="release_year", y="vote_count", ci=None) plt.title("vote count by year",fontsize=18) plt.xlabel("year",fontsize=15) plt.ylabel("votes",fontsize=15) plt.show()

### 3.2: Analysing `tmdb.movies.csv` (df2)

df2.head()

### 3.2.1: Which movie has the highest Worldwide gross?

df2.sort_values(by="worldwide_gross", ascending=False) df2.head()

### 3.2.2: Which movies made the highest gross profit world_wide and locally?

# Calculating and creating a new column "world_wide_gross_profit"

df2['world_wide_gross_profit'] = df2['worldwide_gross']-df2['production_budget']

# Calculating and creating a new column "worldwide_percentage-profit"

df2['worldwide_percentage_profit'] = (df2['world_wide_gross_profit']/df2['production_budget'])*100

# Sorting the data to view the movies with the highest worldwide profit

df2.sort_values(by="worldwide_percentage_profit", ascending=False)[:10]

# Calculating and creating a new column "domestic_gross_profit"

df2['domestic_gross_profit'] = df2['domestic_gross']-df2['production_budget']

# Calculating and creating a new column "domestic_percentage-profit"

df2['domestic_percentage_profit'] = df2['domestic_gross']/df2['production_budget']*10

# Sorting the data to view the movies with the highest domestic profit

df2.sort_values(by="domestic_percentage_profit", ascending=False)[:10]

### 3.2.3: What is the relationship between a movie budget and the reception it gets?

# Finding the relationship between a movies' production budget and how it sells world

# wide

df2['production_budget'].corr(df2['world_wide_gross_profit'])

There is a positive relationship between a movies budget and the reception it gets worldwide
The higher the movie budget the high positive reception it gets.

# Plotting a scatter plot to visualize

plt.scatter(df2.world_wide_gross_profit, df2.production_budget)
plt.title('Production_budget vs world_wide_profit',fontsize=16)
plt.xlabel("world_wide_gross_profit",fontsize=15)
plt.ylabel("production_budget",fontsize=15) plt.show();

# Finding the relationship between a movies' production budget and how it sells locally

df2['production_budget'].corr(df2['domestic_gross_profit'])

We can interprate this to movies with high budget do not do sell domestically

# Plotting a scatter plot to visualize

plt.scatter(df2.domestic_gross_profit, df2.production_budget)
plt.xlabel("domestic_gross_profit") plt.ylabel("production_budget") plt.show();

## 3.3 Analysing `bom.movie_gross.csv` AS df

df.tail()

# How many studios does the data set have?

df['studio'].nunique()

### 3.3.1: Which studios generates the highest gross?

# Studio with the heighest Domestic gross

studio_with_highest_dom_gross = df.groupby(['studio'])["domestic_gross"].sum()
studio_with_highest_dom_gross.sort_values(ascending=False)

# Studio with the heighest foreign gross

studio_with_highest_for_gross = df.groupby(['studio'])["foreign_gross"].sum()
studio_with_highest_for_gross.sort_values(ascending=False)

### 3.3.2: Whats the gross distribution per year?

# Finding the years in the dataset

df['year'].unique()

# domestic gross

sns.barplot(x="year", y="domestic_gross", data=df, ci=None)
plt.title("domestic_gross_by_year",fontsize=18) plt.xlabel("year",fontsize=15)
plt.ylabel("domestic_gross",fontsize=15) plt.show()

# foreign gross

sns.barplot(x="year", y="foreign_gross", data=df, ci=None)
plt.title("domestic_gross_by_year",fontsize=18) plt.xlabel("year",fontsize=15)
plt.ylabel("foreign_gross",fontsize=15) plt.show()

## 3.4: Analysing `im.db` tables

image.png

conn = sqlite3.connect('im.db')

# Selecting the table names

tables_name = """SELECT name AS 'Table Names' FROM sqlite_master WHERE type='table';"""

pd.read_sql(tables_name, conn)

# lets see how many genre do we have do we have

genres = """ SELECT genres FROM movie_basics GROUP BY genres

; """ data = pd.read_sql(genres ,conn).dropna() data.count()

### 3.4.1: Which Are The Top Genres?

genre_ratings = """ SELECT genres,avg(averagerating) AS average_ratings FROM movie_basics JOIN movie_ratings USING(movie_id) GROUP BY genres ORDER BY average_ratingS DESC

; """ pd.read_sql(genre_ratings, conn).dropna()

### 3.4.2: Which are the most viewed genres?

genre_ratings = """ SELECT genres,sum(numvotes) AS People_viewed FROM movie_basics JOIN movie_ratings USING(movie_id) GROUP BY genres ORDER BY people_viewed desc

; """ pd.read_sql(genre_ratings, conn).dropna()

### 3.4.1: Does the numbers of viewers relate with the ratings?

genre_counts = """ SELECT genres, sum(numvotes) AS people_viewed, avg(averagerating) as average_rating FROM movie_basics JOIN movie_ratings USING(movie_id) GROUP BY genres HAVING people_viewed between 1000 and 50000 AND average_rating between 5.5 and 8 ORDER BY people_viewed desc

; """ pd.read_sql(genre_counts, conn) data = pd.read_sql(genre_counts, conn).dropna() data

# Find outliers from ratings

data["average_rating"].plot(kind='box',vert=False,showfliers=False);

# find outliers from people viewed

plt.boxplot(x=data['people_viewed'], vert=False,showfliers=False); plt.semilogy()

data["average_rating"].corr(data["people_viewed"])

plt.scatter(data.people_viewed, data.average_rating) plt.xlabel("people_viewed") plt.ylabel("average_rating") plt.show();

There is no relationship between number of people viewing a genre and it's rating

### 3.4.2: Does a movie length affect its rating?

movie_length = """ SELECT movie_id, runtime_minutes AS length, averagerating AS rating FROM movie_basics JOIN movie_ratings USING(movie_id) GROUP BY movie_id HAVING length < 200 LIMIT 1000

""" data2 = pd.read_sql(movie_length, conn).dropna() data2

data2.describe()

# Finding relationship between movie rating and legnth

data = pd.read_sql(movie_length,conn).dropna() data2["rating"].corr(data2["length"])

# Plotting a scatter plot

plt.scatter(data2.length, data2.rating) plt.xlabel("length",fontsize=15) plt.ylabel("rating",fontsize=15) plt.title("Movie_length and Rating",fontsize=20)

A movie legnth does not affect its rating

### 3.4.3: Who are the best directors?

# first lets find how many directors we have

directors_details = """ SELECT person_id AS director_id, primary_name AS director_name FROM directors JOIN persons USING(person_id) GROUP BY person_id ; """ data3 = pd.read_sql(directors_details,conn) data3.count()

data3.info()

We have 109,251 directors is our data set.

### 3.4.4: Out of the Director Movie rated which director has the heighest rating?

directors_ratings = """ SELECT person_id AS director_id, primary_name As director_name, COUNT(movie_id) AS movies_rated, avg(averagerating) AS director_movies_average_rating FROM directors LEFT JOIN persons USING(person_id) JOIN movie_ratings USING(movie_id) GROUP BY director_id HAVING movies_rated > 5 ORDER BY director_movies_average_rating DESC

; """ data4 = pd.read_sql(directors_ratings,conn) data4

### 3.4.5: Out of the Director Movie rated which director has the heighest number of movies?

directors_ratings = """ SELECT person_id AS director_id, primary_name As director_name, COUNT(movie_id) AS movies_rated, avg(averagerating) AS director_movies_average_rating FROM directors LEFT JOIN persons USING(person_id) JOIN movie_ratings USING(movie_id) GROUP BY director_id HAVING movies_rated > 5 ORDER BY movies_rated desc

; """ data6 = pd.read_sql(directors_ratings,conn) data6

### 3.4.6: Do we have a relationship between movie ratings and number of movie rated?

data5["director_movies_average_rating"].corr(data5["movies_rated"])

No relationship between directors_movies_rated and the movies rating

plt.scatter(data5.movies_rated, data5.director_movies_average_rating) plt.xlabel("movies_rated") plt.ylabel("average_rating")

### 3.4.7: What is the average movie length?

movie_durations = """ SELECT movie_id,avg(runtime_minutes) AS average_runtime,start_year FROM movie_basics GROUP BY start_year

; """ data7 = pd.read_sql(movie_durations,conn) data7.dropna()

# runtime by movies

plt.hist(data7['average_runtime'],range=(40,200),bins=30) plt.title("Movie_runtime",fontsize=20) plt.xlabel("movie_length_in_mins",fontsize=15) plt.ylabel("number_of_movies",fontsize=15)

# Create year range of 5 years

runtime_by_year = data7.copy() runtime_by_year["start_year"] = ((runtime_by_year['start_year']//5)*5).astype("int64")

# Plotting boxplot to visualize

sns.boxplot(x="start_year",y="average_runtime",data=runtime_by_year, palette='colorblind', showfliers=False)

## Releases

No releases published
Create a new release

## Packages

No packages published
Publish your first package

## Languages

- **Jupyter Notebook** 100.0%