Project Title	**Google Playstore Apps rating Prediction**
Tools	Visual Studio code / jupyter notebook
Technologies	Finance Analyst
Project Difficulties level	Advance

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set]

About Dataset

Context

While many public datasets (on Kaggle and the like) provide Apple App Store data, there are not many counterpart datasets available for Google Play Store apps anywhere on the web. On digging deeper, I found out that iTunes App Store page deploys a nicely indexed appendix-like structure to allow for simple and easy web scraping. On the other hand, Google Play Store uses sophisticated modern-day techniques (like dynamic page load) using JQuery making scraping more challenging.

Content

Each app (row) has values for catergory, rating, size, and more.

Acknowledgements

This information is scraped from the Google Play Store. This app information would not be available without it.

Inspiration

The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market!

About data columns:

App : The name of the app

Category : The category of the app

Rating : The rating of the app in the Play Store

Reviews : The number of reviews of the app

Size : The size of the app

Install : The number of installs of the app

Type : The type of the app (Free/Paid)

Price : The price of the app (0 if it is Free)

Content Rating : The appropiate target audience of the app

Genres: The genre of the app

Last Updated : The date when the app was last updated

Current Ver : The current version of the app

Android Ver : The minimum Android version required to run the app

Example that how you can create project you can get idea from here:

Machine Learning Project: Google Play Store Analysis, EDA & Visualization

Objective:

To analyze the Google Play Store dataset and draw useful insights using exploratory data analysis (EDA), visualization, and machine learning techniques. The dataset contains various app-related attributes such as ratings, reviews, price, size, installs, and more. We will clean the data, perform EDA, and use visualizations to uncover hidden patterns and trends.

Dataset Overview:

We will work with the following columns:

- **App: Name of the application.**
- **Category: Category under which the app is listed.**
- **Rating: User rating of the app.**
- **Reviews: Number of reviews for the app.**
- **Size: Size of the app (in MB).**
- **Install: Number of user installs.**
- **Type: Free or Paid.**
- **Price: Price of the app.**
- **Content Rating: Audience the app is appropriate for.**
- **Genres: App genres.**
- **Last Updated: Last date the app was updated.**
- **Current Ver: Latest version of the app.**
- **Android Ver: Minimum required Android version.**

Step 1: Import Necessary Libraries

```
# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline



# Read the dataset

df = pd.read_csv('googleplaystore.csv')



# Check the first few rows

df.head()
```

Step 2: Data Cleaning and Preprocessing

1. **Handling Missing Values: We will identify and handle missing values in the dataset.**

```
# Check for missing values

df.isnull().sum()



# Drop rows with missing values in important columns

df.dropna(subset=['Rating', 'Reviews', 'Size', 'Installs'],
inplace=True)



# Check the updated data

df.info()
```

2. Converting Columns to Appropriate Data Types:
    - Convert Reviews and Installs to integer types.
    - Convert Price to numeric.
    - Convert Size to a uniform numeric format.

```
# Convert 'Reviews' to integer

df['Reviews'] = df['Reviews'].astype(int)



# Convert 'Installs' by removing '+' and ',' then converting to
integer
```

```python
df['Installs'] = df['Installs'].apply(lambda x: x.replace(',',
'').replace('+', '')).astype(int)


# Convert 'Price' by removing '$' and converting to float

df['Price'] = df['Price'].apply(lambda x: float(x.replace('$', ''))
if '$' in x else float(x))


# Convert 'Size' to numeric (MB) - Convert 'k' to MB

def convert_size(size):

    if 'M' in size:

        return float(size.replace('M', ''))

    elif 'k' in size:

        return float(size.replace('k', '')) / 1000

    else:

        return np.nan


df['Size'] = df['Size'].apply(convert_size)
```

3. **Handling Duplicate Entries:**

```python
# Check for duplicates

df.duplicated().sum()



# Remove duplicates

df.drop_duplicates(inplace=True)
```

Step 3: Exploratory Data Analysis (EDA)

3.1: Distribution of App Ratings

```python
plt.figure(figsize=(8,6))

sns.histplot(df['Rating'].dropna(), bins=20, kde=True)

plt.title('Distribution of App Ratings')

plt.xlabel('Rating')

plt.ylabel('Count')

plt.show()
```

Insight: Visualizing the distribution of app ratings helps us understand if there are more highly rated apps or apps with low ratings.

3.2: Top 10 Categories by Number of Apps

```
plt.figure(figsize=(12,6))

top_categories = df['Category'].value_counts().head(10)

sns.barplot(x=top_categories.index, y=top_categories.values,
palette='coolwarm')

plt.title('Top 10 App Categories by Number of Apps')

plt.ylabel('Number of Apps')

plt.xlabel('Category')

plt.xticks(rotation=45)

plt.show()
```

Insight: This shows the most popular categories on Google Play Store in terms of app count.

3.3: Free vs Paid Apps

```
plt.figure(figsize=(6,4))

sns.countplot(df['Type'], palette='Set2')
```

```
plt.title('Distribution of Free vs Paid Apps')

plt.show()
```

3.4: Correlation Between Reviews and Rating

```
plt.figure(figsize=(10,6))

sns.scatterplot(x='Reviews', y='Rating', data=df, hue='Category')

plt.title('Correlation Between Reviews and Ratings')

plt.show()
```

Insight: This helps to identify if more reviews generally mean higher ratings.

Step 4: Price Analysis

4.1: Price Distribution for Paid Apps

```
paid_apps = df[df['Type'] == 'Paid']

plt.figure(figsize=(10,6))

sns.histplot(paid_apps['Price'], bins=30, color='orange')

plt.title('Price Distribution for Paid Apps')
```

```
plt.xlabel('Price ($)')

plt.ylabel('Count')

plt.show()
```

4.2: Relationship Between Price and Rating

```
plt.figure(figsize=(8,6))

sns.scatterplot(x='Price', y='Rating', data=paid_apps)

plt.title('Price vs Rating for Paid Apps')

plt.show()
```

Step 5: Content Rating Analysis

5.1: Distribution of Content Ratings

```
plt.figure(figsize=(10,6))

content_ratings = df['Content Rating'].value_counts()

sns.barplot(x=content_ratings.index, y=content_ratings.values,
palette='coolwarm')
```

```
plt.title('Distribution of Content Ratings')

plt.xlabel('Content Rating')

plt.ylabel('Count')

plt.show()
```

5.2: Content Rating vs Rating

```
plt.figure(figsize=(10,6))

sns.boxplot(x='Content Rating', y='Rating', data=df, palette='Set1')

plt.title('Content Rating vs App Rating')

plt.show()
```

Step 6: Genre and Installs Analysis

6.1: Top Genres by Install Count

```
plt.figure(figsize=(12,6))

top_genres_installs =
df.groupby('Genres')['Installs'].sum().sort_values(ascending=False).h
ead(10)
```

```
sns.barplot(x=top_genres_installs.index,
y=top_genres_installs.values, palette='Spectral')

plt.xticks(rotation=90)

plt.title('Top 10 Genres by Install Count')

plt.show()
```

Step 7: Machine Learning (Predicting App Rating)

7.1: Prepare Data for Modeling

We will predict the app rating based on the features in the dataset. First, let's prepare the data by encoding categorical variables and splitting it into training and testing sets.

```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder



# Encode categorical columns

label_encoder = LabelEncoder()

df['Category'] = label_encoder.fit_transform(df['Category'])

df['Type'] = label_encoder.fit_transform(df['Type'])
```

```
df['Content Rating'] = label_encoder.fit_transform(df['Content Rating'])

df['Genres'] = label_encoder.fit_transform(df['Genres'])


# Define features and target variable

X = df[['Category', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Genres', 'Content Rating']]

y = df['Rating']


# Handle missing values in target

y.fillna(y.median(), inplace=True)


# Split the data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

7.2: Train a Random Forest Model

```
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error, r2_score
```

```
# Train the model

model = RandomForestRegressor(n_estimators=100, random_state=42)

model.fit(X_train, y_train)



# Make predictions

y_pred = model.predict(X_test)



# Evaluate the model

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)



print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')
```

Conclusion:

- **Data Insights: We explored the distribution of app ratings, prices, and installs. We identified the top categories and genres in terms of the number of apps and installs.**
- **Machine Learning: We built a Random Forest model to predict app ratings based on the dataset, achieving a decent R-squared score.**

Sample link

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

In [209]:

```
data=pd.read_csv("F:/AI/googleplaystore.csv")
```

In [210]:

```
data.sample(10)
```

Out[210]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
7423	CJ Browser - Fast & Private	COMMUNICATION	4.2	5	15M	100+	Free	0	Everyone	Communication	November 7, 2017	1.0	4.0 and up
6328	BJ - Confidential	COMMUNICATION	NaN	0	3.2M	10+	Free	0	Teen	Communication	April 23, 2018	1.7	4.1 and up
2349	Teach Me Anatomy	MEDICAL	4.7	9945	97M	500,000+	Free	0	Everyone	Medical	July 5, 2018	5.11	4.1 and up
10732	Draw with FP sDraw	TOOLS	4.3	3268	467k	100,000+	Free	0	Everyone	Tools	December 16, 2017	6.6	2.0 and up
10574	Lottery Results: Florida	FAMILY	4.2	582	3.2M	100,000+	Free	0	Teen	Entertainment	January 22, 2018	4.0	4.0 and up
5094	AG Subway Simulator	FAMILY	4.5	623	47	5,000+	Pai	$0.9	Everyone	Simulation	June 2, 2018	1.3.0.6	4.1 and

	Mobile				M		d	9					up
3366	Color Call - Caller Screen, LED Flash	PERSONALIZATION	4.7	29485	9.9M	1,000,000+	Free	0	Everyone	Personalization	July 20, 2018	1.0.4	4.0 and up
4578	Samsung Smart Switch Mobile	TOOLS	4.3	146913	24M	100,000,000+	Free	0	Everyone	Tools	July 18, 2018	3.5.02.15	4.0 and up
8793	Dr. Seuss's ABC	FAMILY	4.7	429	12M	10,000+	Paid	$3.99	Everyone	Books & Reference;Education	February 26, 2018	2.05	4.0.3 and up
8968	DV - Digito Verificador	TOOLS	NaN	2	4.2M	500+	Free	0	Everyone	Tools	March 2, 2017	1.0	4.0 and up

In [211]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10841 entries, 0 to 10840

Data columns (total 13 columns):

 #   Column          Non-Null Count  Dtype

---  ------          --------------  -----
```

```
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

In [212]:

```
data['App'].isna().sum()
```

Out[212]:

```
0
```

Category

In [213]:

```
data.shape
```

Out[213]:

```
(10841, 13)
```

In [214]:

```
data['Category'].isnull().sum()
```

Out[214]:

```
0
```

In [215]:

```
data['Category'].unique()
```

Out[215]:

```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
```

```
       'LIBRARIES_AND_DEMO', 'LIFESTYLE', 'GAME', 'FAMILY', 'MEDICAL',
       'SOCIAL', 'SHOPPING', 'PHOTOGRAPHY', 'SPORTS', 'TRAVEL_AND_LOCAL',
       'TOOLS', 'PERSONALIZATION', 'PRODUCTIVITY', 'PARENTING', 'WEATHER',
       'VIDEO_PLAYERS', 'NEWS_AND_MAGAZINES', 'MAPS_AND_NAVIGATION',

       '1.9'], dtype=object)
```

In [216]:

```
data[data['Category'] == '1.9']
```

Out[216]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	February 11, 2018	1.0.19	4.0 and up	NaN

In [220]:

```
data['Category'].loc[10472]=np.nan
```

In [221]:

```
data['Category'].loc[10472]
```

Out[221]:

nan

In [222]:

```python
df_category=data['Category'].value_counts()
df_category
```

```
Out[222]:
Category
FAMILY                 1972
GAME                   1144
TOOLS                   843
MEDICAL                 463
BUSINESS                460
PRODUCTIVITY            424
PERSONALIZATION         392
COMMUNICATION           387
SPORTS                  384
LIFESTYLE               382
FINANCE                 366
HEALTH_AND_FITNESS      341
PHOTOGRAPHY             335
SOCIAL                  295
NEWS_AND_MAGAZINES      283
SHOPPING                260
TRAVEL_AND_LOCAL        258
DATING                  234
BOOKS_AND_REFERENCE     231
VIDEO_PLAYERS           175
EDUCATION               156
ENTERTAINMENT           149
MAPS_AND_NAVIGATION     137
FOOD_AND_DRINK          127
HOUSE_AND_HOME           88
AUTO_AND_VEHICLES        85
LIBRARIES_AND_DEMO       85
```

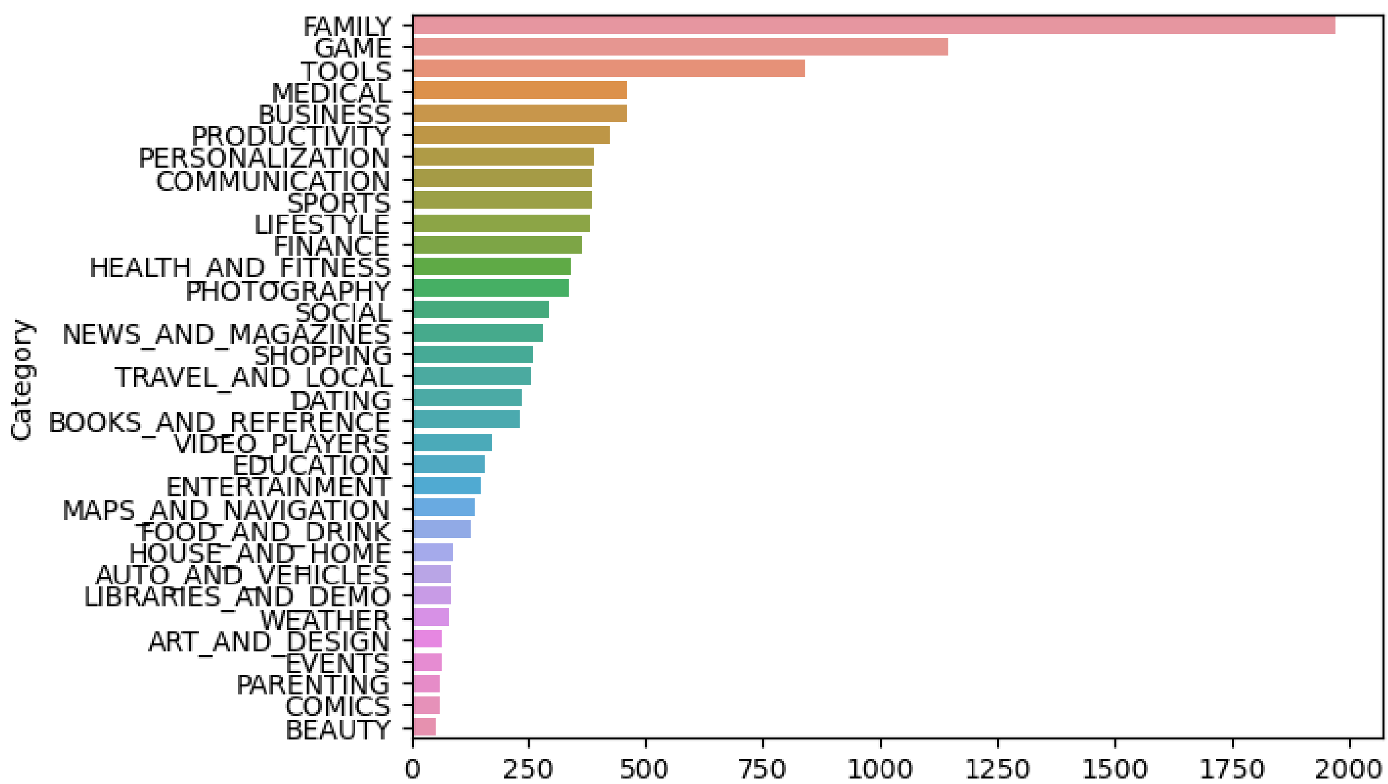
```
WEATHER                  82
ART_AND_DESIGN           65
EVENTS                   64
PARENTING                60
COMICS                   60
BEAUTY                   53

Name: count, dtype: int64
```

In [223]:

```python
sns.barplot(x=df_category.values,y=df_category.index,orient='h')
```

Out[223]:

```
<Axes: ylabel='Category'>
```



In [224]:

```python
data.columns
```

Out[224]:

```
Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
       'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
       'Android Ver'],

      dtype='object')
```

In [225]:

```
data['Rating'].unique
```

Out[225]:

```
<bound method Series.unique of 0        4.1
1        3.9
2        4.7
3        4.5
4        4.3
        ...
10836    4.5
10837    5.0
10838    NaN
10839    4.5
10840    4.5

Name: Rating, Length: 10841, dtype: float64>
```

Reviews

In [226]:

```
data['Reviews'].unique()
```

Out[226]:

```
array(['159', '967', '87510', ..., '603', '1195', '398307'], dtype=object)
```

In [227]:

```
data['Reviews'].dtype
```

Out[227]:

```
dtype('O')
```

In [228]:

```
data['Reviews']=data['Reviews'].replace('3.0M','3000000.0')
```

In [229]:

```
data['Reviews']=data['Reviews'].astype(float)
```

In [230]:

```
data['Reviews'].dtype
```

Out[230]:

```
dtype('float64')
```

size

In [231]:

```
data['Size'].unique()
```

Out[231]:

```
array(['19M', '14M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M',
       '28M', '12M', '20M', '21M', '37M', '2.7M', '5.5M', '17M', '39M',
       '31M', '4.2M', '7.0M', '23M', '6.0M', '6.1M', '4.6M', '9.2M',
       '5.2M', '11M', '24M', 'Varies with device', '9.4M', '15M', '10M',
       '1.2M', '26M', '8.0M', '7.9M', '56M', '57M', '35M', '54M', '201k',
       '3.6M', '5.7M', '8.6M', '2.4M', '27M', '2.5M', '16M', '3.4M',
       '8.9M', '3.9M', '2.9M', '38M', '32M', '5.4M', '18M', '1.1M',
       '2.2M', '4.5M', '9.8M', '52M', '9.0M', '6.7M', '30M', '2.6M',
       '7.1M', '3.7M', '22M', '7.4M', '6.4M', '3.2M', '8.2M', '9.9M',
       '4.9M', '9.5M', '5.0M', '5.9M', '13M', '73M', '6.8M', '3.5M',
       '4.0M', '2.3M', '7.2M', '2.1M', '42M', '7.3M', '9.1M', '55M',
       '23k', '6.5M', '1.5M', '7.5M', '51M', '41M', '48M', '8.5M', '46M',
       '8.3M', '4.3M', '4.7M', '3.3M', '40M', '7.8M', '8.8M', '6.6M',
       '5.1M', '61M', '66M', '79k', '8.4M', '118k', '44M', '695k', '1.6M',
       '6.2M', '18k', '53M', '1.4M', '3.0M', '5.8M', '3.8M', '9.6M',
       '45M', '63M', '49M', '77M', '4.4M', '4.8M', '70M', '6.9M', '9.3M',
       '10.0M', '8.1M', '36M', '84M', '97M', '2.0M', '1.9M', '1.8M',
       '5.3M', '47M', '556k', '526k', '76M', '7.6M', '59M', '9.7M', '78M',
       '72M', '43M', '7.7M', '6.3M', '334k', '34M', '93M', '65M', '79M',
       '100M', '58M', '50M', '68M', '64M', '67M', '60M', '94M', '232k',
       '99M', '624k', '95M', '8.5k', '41k', '292k', '11k', '80M', '1.7M',
       '74M', '62M', '69M', '75M', '98M', '85M', '82M', '96M', '87M',
       '71M', '86M', '91M', '81M', '92M', '83M', '88M', '704k', '862k',
       '899k', '378k', '266k', '375k', '1.3M', '975k', '980k', '4.1M',
       '89M', '696k', '544k', '525k', '920k', '779k', '853k', '720k',
       '713k', '772k', '318k', '58k', '241k', '196k', '857k', '51k',
       '953k', '865k', '251k', '930k', '540k', '313k', '746k', '203k',
       '26k', '314k', '239k', '371k', '220k', '730k', '756k', '91k',
       '293k', '17k', '74k', '14k', '317k', '78k', '924k', '902k', '818k',
       '81k', '939k', '169k', '45k', '475k', '965k', '90M', '545k', '61k',
       '283k', '655k', '714k', '93k', '872k', '121k', '322k', '1.0M',
       '976k', '172k', '238k', '549k', '206k', '954k', '444k', '717k',
```

```
       '210k', '609k', '308k', '705k', '306k', '904k', '473k', '175k',
       '350k', '383k', '454k', '421k', '70k', '812k', '442k', '842k',
       '417k', '412k', '459k', '478k', '335k', '782k', '721k', '430k',
       '429k', '192k', '200k', '460k', '728k', '496k', '816k', '414k',
       '506k', '887k', '613k', '243k', '569k', '778k', '683k', '592k',
       '319k', '186k', '840k', '647k', '191k', '373k', '437k', '598k',
       '716k', '585k', '982k', '222k', '219k', '55k', '948k', '323k',
       '691k', '511k', '951k', '963k', '25k', '554k', '351k', '27k',
       '82k', '208k', '913k', '514k', '551k', '29k', '103k', '898k',
       '743k', '116k', '153k', '209k', '353k', '499k', '173k', '597k',
       '809k', '122k', '411k', '400k', '801k', '787k', '237k', '50k',
       '643k', '986k', '97k', '516k', '837k', '780k', '961k', '269k',
       '20k', '498k', '600k', '749k', '642k', '881k', '72k', '656k',
       '601k', '221k', '228k', '108k', '940k', '176k', '33k', '663k',
       '34k', '942k', '259k', '164k', '458k', '245k', '629k', '28k',
       '288k', '775k', '785k', '636k', '916k', '994k', '309k', '485k',
       '914k', '903k', '608k', '500k', '54k', '562k', '847k', '957k',
       '688k', '811k', '270k', '48k', '329k', '523k', '921k', '874k',
       '981k', '784k', '280k', '24k', '518k', '754k', '892k', '154k',
       '860k', '364k', '387k', '626k', '161k', '879k', '39k', '970k',
       '170k', '141k', '160k', '144k', '143k', '190k', '376k', '193k',
       '246k', '73k', '658k', '992k', '253k', '420k', '404k', '1,000+',
       '470k', '226k', '240k', '89k', '234k', '257k', '861k', '467k',
       '157k', '44k', '676k', '67k', '552k', '885k', '1020k', '582k',

       '619k'], dtype=object)
```

In [232]:

```
data['Size']=data['Size'].str.replace('M','000') # This Converte sizes to Kbytes
data['Size']=data['Size'].replace('Varies with device',np.nan)
data['Size']=data['Size'].str.replace('k','')
data['Size']=data['Size'].replace('1,000+','1000')
```

In [233]:

```
data['Size']=data['Size'].astype(float)
```

In [234]:

```
data['Size'].dtype
```

Out[234]:

```
dtype('float64')
```

Installs

In [235]:

```
data['Installs'].unique()
```

Out[235]:

```
array(['10,000+', '500,000+', '5,000,000+', '50,000,000+', '100,000+',
       '50,000+', '1,000,000+', '10,000,000+', '5,000+', '100,000,000+',
       '1,000,000,000+', '1,000+', '500,000,000+', '50+', '100+', '500+',

       '10+', '1+', '5+', '0+', '0', 'Free'], dtype=object)
```

In [236]:

```
data['Installs']=data['Installs'].str.replace(',','')
data['Installs']=data['Installs'].str.replace('+','')
data['Installs']=data['Installs'].replace('Free',np.nan)
data['Installs']=data['Installs'].astype(float)
```

In [237]:

```
data['Installs'].dtype
```

Out[237]:

```
dtype('float64')
```

Price

In [238]:

```python
data['Price'].unique()
```

Out[238]:

```
array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',
       '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',
       '$10.00', '$24.99', '$11.99', '$79.99', '$16.99', '$14.99',
       '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99',
       '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',
       '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',
       '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',
       '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',
       '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',
       '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',
       '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',
       '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',
       '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',

       '$394.99', '$1.26', 'Everyone', '$1.20', '$1.04'], dtype=object)
```

In [239]:

```python
data['Price']=data["Price"].str.replace('$','')
data['Price']=data["Price"].replace('Everyone',np.nan)
data['Price']=data["Price"].astype(float)
```

In [240]:

```
data['Price'].dtype
```

Out[240]:

```
dtype('float64')
```

Content Rating

In [241]:

```
raiting=data['Content Rating'].value_counts()
raiting
```

Out[241]:

```
Content Rating
Everyone           8714
Teen               1208
Mature 17+          499
Everyone 10+        414
Adults only 18+       3
Unrated               2

Name: count, dtype: int64
```

In [242]:

```
px.scatter(x=raiting.index,y=raiting.values,color=raiting.index,title="Apps Raiting")
```

In [243]:

```python
data.groupby('Category')['Reviews'].sum()
```

```
Out[243]:
Category
ART_AND_DESIGN         1.714440e+06
AUTO_AND_VEHICLES      1.163666e+06
BEAUTY                 3.962400e+05
BOOKS_AND_REFERENCE    2.195907e+07
BUSINESS               1.395455e+07
COMICS                 3.383276e+06
COMMUNICATION          8.154623e+08
DATING                 7.291278e+06
EDUCATION              3.959579e+07
ENTERTAINMENT          5.917815e+07
EVENTS                 1.610180e+05
FAMILY                 4.102263e+08
FINANCE                1.755073e+07
FOOD_AND_DRINK         8.883330e+06
GAME                   1.585422e+09
HEALTH_AND_FITNESS     3.789374e+07
HOUSE_AND_HOME         3.976385e+06
LIBRARIES_AND_DEMO     1.037118e+06
LIFESTYLE              1.288278e+07
MAPS_AND_NAVIGATION    3.065925e+07
MEDICAL                1.585975e+06
NEWS_AND_MAGAZINES     5.440086e+07
PARENTING              9.583310e+05
PERSONALIZATION        8.934614e+07
PHOTOGRAPHY            2.135166e+08
PRODUCTIVITY           1.141170e+08
SHOPPING               1.150412e+08
SOCIAL                 6.212414e+08
```

```
SPORTS                7.083017e+07
TOOLS                 2.731850e+08
TRAVEL_AND_LOCAL      6.261792e+07
VIDEO_PLAYERS         1.103802e+08
WEATHER               1.460474e+07

Name: Reviews, dtype: float64
```

In [244]:

```
data.describe()
```

Out[244]:

	Rating	Reviews	Size	Installs	Price
count	9367.000000	1.084100e+04	9146.000000	1.084000e+04	10840.000000
mean	4.193338	4.443887e+05	19577.388487	1.546434e+07	1.027368
std	0.537431	2.927728e+06	24041.532453	8.502936e+07	15.949703
min	1.000000	0.000000e+00	1.000000	0.000000e+00	0.000000
25%	4.000000	3.800000e+01	5.600000	1.000000e+03	0.000000

50%	4.300000	2.094000e+03	13000.000000	1.000000e+05	0.000000
75%	4.500000	5.479800e+04	30000.000000	5.000000e+06	0.000000
max	19.000000	7.815831e+07	100000.000000	1.000000e+09	400.000000

In [246]:

```
category_review=data.groupby('Category')['Reviews'].max().head(10)
category_review
```

Out[246]:

```
Category
ART_AND_DESIGN            295237.0
AUTO_AND_VEHICLES         271920.0
BEAUTY                    113715.0
BOOKS_AND_REFERENCE      2915189.0
BUSINESS                 1279800.0
COMICS                   1013944.0
COMMUNICATION           69119316.0
DATING                    516917.0
EDUCATION                6290507.0
ENTERTAINMENT            7165362.0

Name: Reviews, dtype: float64
```

In [247]:

```
data['Rating']
```

Out[247]:

```
0        4.1
1        3.9
2        4.7
3        4.5
4        4.3
        ...
10836    4.5
10837    5.0
10838    NaN
10839    4.5
10840    4.5

Name: Rating, Length: 10841, dtype: float64
```

In [248]:

```
def category_rating(rating):

    try:
        rating = round(rating)
        if int(rating) in range(0,3):
            return 'low'
        elif int(rating) in range(3,5):
            return 'Average'
        elif int(rating) in range(4,6):
            return 'High'
    except ValueError as error:
        return 'none'

data['category_rating']=data['Rating'].apply(category_rating)
```

In [251]:

```
category_r=data['category_rating'].value_counts()
```

```
category_r
```

Out[251]:

```
category_rating
Average    7299
High       1917
none       1474
low         150

Name: count, dtype: int64
```

In [256]:

```
px.bar(x=category_r.values,y=category_r.index,color=category_r.index,title="Category Rating")
```

In [285]:

```
data.head()
```

Out[285]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	category_rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159.0	19000.0	10000.0	Free	0.0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up	Average

1	Coloring book moana	ART_AND_DESIGN	3.9	967.0	14000.0	500000.0	Free	0.0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up	Average
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510.0	8.7	5000000.0	Free	0.0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up	High
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644.0	25000.0	50000000.0	Free	0.0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up	Average
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967.0	2.8	100000.0	Free	0.0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up	Average

```
In [ ]:
```

[Reference link]

[Reference link]

[You can practice and get experience from here for sql project]

SQL Project: Google Play Store Analysis, EDA & Visualization

Objective:

The goal of this project is to analyze the Google Play Store dataset using SQL to derive insights through queries, performing exploratory data analysis (EDA) and basic visualizations. We will utilize SQL to filter, group, and aggregate data based on the features like app ratings, installs, categories, and pricing, followed by visualizing those insights in a SQL environment (for example, using PostgreSQL with visualization extensions or integrating with BI tools like Tableau).

Dataset Overview

We will work with the following columns:

- App: Name of the application.
- Category: Category under which the app is listed.
- Rating: User rating of the app.
- Reviews: Number of reviews for the app.
- Size: Size of the app (in MB).
- Install: Number of user installs.
- Type: Free or Paid.
- Price: Price of the app.
- Content Rating: Audience the app is appropriate for.
- Genres: App genres.
- Last Updated: Last date the app was updated.

- Current Ver: Latest version of the app.
- Android Ver: Minimum required Android version.

Step 1: Database Setup and Table Creation

First, let's create the table for the Play Store data in an SQL environment (PostgreSQL in this case).

```sql
-- Creating the Google Play Store table
CREATE TABLE google_play_store (
    App VARCHAR(255),
    Category VARCHAR(50),
    Rating FLOAT,
    Reviews INTEGER,
    Size VARCHAR(50),
    Install VARCHAR(50),
    Type VARCHAR(10),
    Price DECIMAL(10,2),
    Content_Rating VARCHAR(20),
    Genres VARCHAR(50),
    Last_Updated DATE,
    Current_Ver VARCHAR(20),
    Android_Ver VARCHAR(20)
);
```

Step 2: Data Insertion

After the table is created, we would insert the data into it. Assuming we have a CSV file with the Play Store data, we can use the following query to load it into the database (this can also be done using tools like pgAdmin or SQL Workbench).

```
-- Loading data from CSV file into the table
COPY google_play_store (App, Category, Rating, Reviews,
Size, Install, Type, Price, Content_Rating, Genres,
Last_Updated, Current_Ver, Android_Ver)
FROM '/path/to/googleplaystore.csv'
DELIMITER ',' CSV HEADER;
```

Step 3: Basic Data Exploration

3.1: Checking the Structure of the Data

```
-- Previewing the data to check the first few rows
SELECT * FROM google_play_store LIMIT 10;
```

3.2: Checking Missing Values

```
-- Checking for missing or NULL values in the dataset
SELECT
```

```
    COUNT(*) AS total_records,
    SUM(CASE WHEN Rating IS NULL THEN 1 ELSE 0 END) AS
missing_ratings,
    SUM(CASE WHEN Reviews IS NULL THEN 1 ELSE 0 END) AS
missing_reviews,
    SUM(CASE WHEN Size IS NULL THEN 1 ELSE 0 END) AS
missing_size
FROM google_play_store;
```

Step 4: Data Cleaning and Transformation

4.1: Convert Install Column to Integer

We will clean up the `Install` column by removing '+' and ',' from the values and converting it to integers.

```
-- Removing '+' and ',' from the 'Install' column and
converting it to integer
UPDATE google_play_store
SET Install = REPLACE(REPLACE(Install, ',', ''), '+', '');
```

4.2: Handling Missing Ratings

For missing ratings, we can either remove the rows or fill them with a neutral value like the median rating.

```
-- Filling missing ratings with median value of ratings
```

```sql
WITH median_rating AS (
    SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY
Rating) AS median
    FROM google_play_store
)
UPDATE google_play_store
SET Rating = (SELECT median FROM median_rating)
WHERE Rating IS NULL;
```

Step 5: Exploratory Data Analysis (EDA)

5.1: Distribution of App Categories

We want to know how many apps exist in each category.

```sql
-- Counting the number of apps in each category
SELECT Category, COUNT(App) AS num_apps
FROM google_play_store
GROUP BY Category
ORDER BY num_apps DESC;
```

Output Explanation: This query will give us the count of apps in each category, sorted in descending order to show the most popular categories by app count.

5.2: Average Rating per Category

```
-- Calculating the average rating for each category
SELECT Category, AVG(Rating) AS avg_rating
FROM google_play_store
GROUP BY Category
ORDER BY avg_rating DESC;
```

Output Explanation: This query will show the average user rating for each category, helping us identify which categories have better-rated apps on average.

5.3: Most Popular Genres by Installs

```
-- Summing the installs for each genre
SELECT Genres, SUM(CAST(Install AS BIGINT)) AS
total_installs
FROM google_play_store
GROUP BY Genres
ORDER BY total_installs DESC
LIMIT 10;
```

Output Explanation: This query gives the top 10 genres by total installs, allowing us to see which genres are most downloaded.

Step 6: Price and Type Analysis

6.1: Free vs Paid App Distribution

```
-- Counting the number of free and paid apps
SELECT Type, COUNT(App) AS num_apps
FROM google_play_store
GROUP BY Type;
```

Output Explanation: This query counts the number of free and paid apps, which helps us understand the distribution between free and paid apps on the Play Store.

6.2: Average Price of Paid Apps

```
-- Calculating the average price of paid apps
SELECT AVG(Price) AS avg_price
FROM google_play_store
WHERE Type = 'Paid';
```

Output Explanation: This query shows the average price of all paid apps on the Play Store.

Step 7: Content Rating and User Feedback

7.1: Distribution of Content Ratings

```
-- Counting the number of apps for each content rating
SELECT Content_Rating, COUNT(App) AS num_apps
FROM google_play_store
GROUP BY Content_Rating;
```

Output Explanation: This query counts the number of apps for each content rating (e.g., Everyone, Teen, Mature 17+), providing insight into the distribution of apps based on content appropriateness.

7.2: Correlation Between Reviews and Rating

```
-- Finding the correlation between the number of reviews
and rating
SELECT ROUND(CORR(Reviews, Rating), 2) AS
correlation_reviews_rating
FROM google_play_store;
```

Output Explanation: This query calculates the correlation between the number of reviews and app ratings, helping us understand if more reviews tend to result in higher or lower ratings.

Step 8: Advanced Queries and Insights

8.1: Top 10 Most Expensive Apps

```
-- Listing the top 10 most expensive apps
SELECT App, Price, Rating
FROM google_play_store
WHERE Type = 'Paid'
ORDER BY Price DESC
LIMIT 10;
```

Output Explanation: This query lists the top 10 most expensive apps and their ratings, which can provide insight into whether expensive apps have high or low ratings.

8.2: Apps With the Highest Installs and Their Ratings

```
-- Listing the top 10 apps with the highest installs
SELECT App, Install, Rating
FROM google_play_store
ORDER BY CAST(Install AS BIGINT) DESC
LIMIT 10;
```

Output Explanation: This query lists the top 10 apps by the number of installs along with their ratings, allowing us to analyze if the most downloaded apps also have high ratings.

Step 9: Visualization (Optional)

If you're using a tool like Tableau or Power BI, you can connect it to your SQL database and visualize the results from the SQL queries above. Here are some suggested visualizations:

- Bar chart of the number of apps per category.
- Pie chart showing the distribution of free vs paid apps.
- Boxplot of app prices for paid apps.
- Scatter plot showing correlation between reviews and ratings.

Conclusion:

- Data Cleaning: We handled missing values, removed unnecessary characters, and converted columns to appropriate types.
- Data Exploration: We explored the dataset to understand app distribution by category, free vs paid apps, and user feedback (ratings, reviews).
- Insights: We gained insights such as which categories have the most apps, which genres are the most popular, and how app price and reviews correlate with ratings.

This SQL project showcases how to handle and analyze app-related data in a structured and efficient manner using SQL queries.