EXPLORATORY DATA ANALYSIS - GOOGLE PLAY STORE APP

STEP 1 :

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

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

This code imports the necessary libraries for data analysis and visualization:

- **pandas (pd)** ***for data manipulation and analysis***
- **numpy (np)** ***for numerical computing***
- **seaborn (sns)** ***for statistical data visualization***
- **matplotlib.pyplot (plt)** ***for creating plots and visualizations***
- **%matplotlib inline** ***is a Jupyter Notebook magic command that allows the plots to be displayed directly in the notebook***

STEP 2:

```
from google.colab import drive
drive.mount('/content/drive/')
```

- ***This code uses the google.colab library to mount the Google Drive to the Colab notebook. Once the Drive is mounted, you can access the files stored in your Drive from within the Colab environment.***

- ***The drive.mount('/content/drive/') command mounts the Drive to the Colab notebook and creates a link that you can click to get an authorization code. Once you enter the authorization code, the Drive will be mounted, and you can access the files stored in your Drive from within the Colab environment.***

- ***Note that this code is specifically for use in Google Colab, which is a cloud-based platform for running Python code. If you are not using Colab or if you don't need to access files from Google Drive, you can skip this code.***

STEP 3:

```
df= pd.read_csv('drive/My Drive/googleplaystore.csv')
```

- ***This code reads a CSV (Comma-Separated Values) file named "googleplaystore.csv" from Google Drive and stores its contents into a pandas DataFrame called df.***

- ***The pd.read_csv() function from the pandas library is used to read the CSV file. The file path 'drive/My Drive/googleplaystore.csv' indicates that the file is located in the "googleplaystore.csv" file in the "My Drive" directory of the mounted Google Drive.***

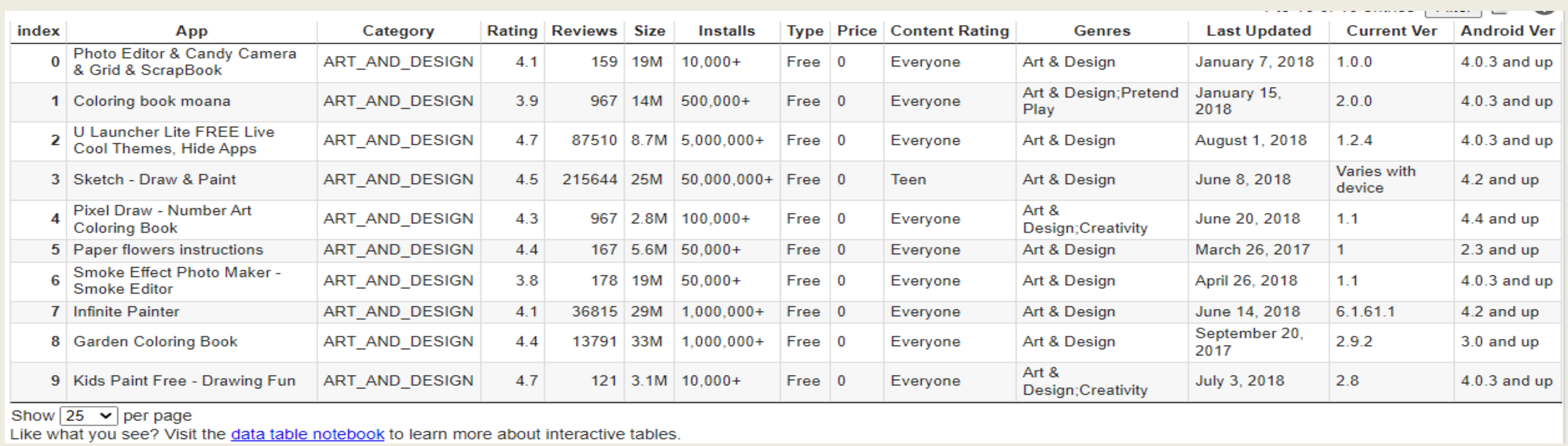
- ***After executing this code, the contents of the CSV file will be stored in the df DataFrame, and you can use various pandas functions to manipulate and analyze the data.***

STEP 4:

```
df= pd.read_csv('drive/My Drive/googleplaystore.csv')
df.head(10)
```

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- ➢ *This code reads the cleaned CSV file "googleplaystore.csv" from Google Drive and stores its contents into a pandas DataFrame called df. Then, it displays the first 10 rows of the DataFrame using the head() method.*
- ➢ *The pd.read_csv('drive/My Drive/googleplaystore.csv ') function is used to read the CSV file from the "My Drive" directory of the mounted Google Drive. Once the file is read, its contents are stored in the df DataFrame.*
- ➢ *The df.head(10) method is used to display the first 10 rows of the DataFrame. This is a useful way to quickly check if the data was imported correctly and to get a sense of what the data looks like.*



index	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite FREE Live Cool Themes, Hide Apps	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone	Art & Design	March 26, 2017	1	2.3 and up
6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0	Everyone	Art & Design	April 26, 2018	1.1	4.0.3 and up
7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free	0	Everyone	Art & Design	June 14, 2018	6.1.61.1	4.2 and up
8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free	0	Everyone	Art & Design	September 20, 2017	2.9.2	3.0 and up
9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0	Everyone	Art & Design;Creativity	July 3, 2018	2.8	4.0.3 and up

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

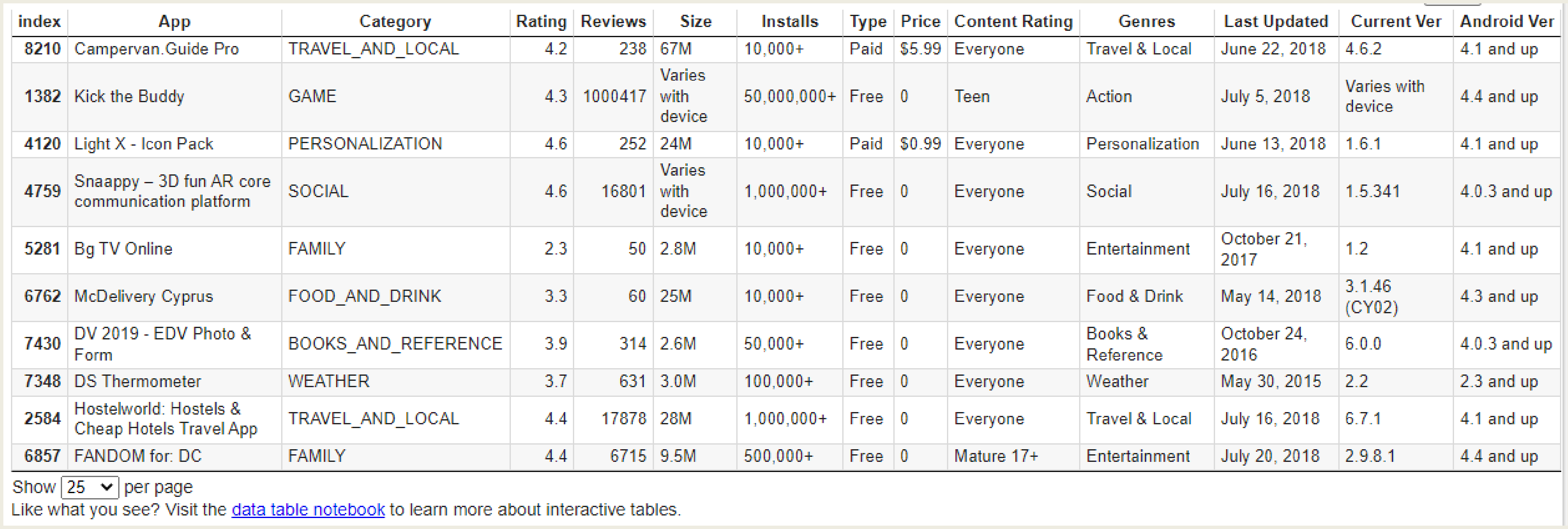
STEP 5:

df.sample(10)

- ***The df.sample(10) method returns a random sample of 10 rows from the DataFrame df. This is a useful way to get a sense of the overall structure and content of the data, and to verify that there are no obvious errors or inconsistencies in the data.***

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- ***The sample() method in pandas is used to randomly sample rows or columns from a DataFrame. By default, it returns a random row from the DataFrame, but you can also specify the number of rows or columns to return. In this case, we have specified to return 10 rows from the DataFrame df.***



index	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
8210	Campervan.Guide Pro	TRAVEL_AND_LOCAL	4.2	238	67M	10,000+	Paid	$5.99	Everyone	Travel & Local	June 22, 2018	4.6.2	4.1 and up
1382	Kick the Buddy	GAME	4.3	1000417	Varies with device	50,000,000+	Free	0	Teen	Action	July 5, 2018	Varies with device	4.4 and up
4120	Light X - Icon Pack	PERSONALIZATION	4.6	252	24M	10,000+	Paid	$0.99	Everyone	Personalization	June 13, 2018	1.6.1	4.1 and up
4759	Snaappy – 3D fun AR core communication platform	SOCIAL	4.6	16801	Varies with device	1,000,000+	Free	0	Everyone	Social	July 16, 2018	1.5.341	4.0.3 and up
5281	Bg TV Online	FAMILY	2.3	50	2.8M	10,000+	Free	0	Everyone	Entertainment	October 21, 2017	1.2	4.1 and up
6762	McDelivery Cyprus	FOOD_AND_DRINK	3.3	60	25M	10,000+	Free	0	Everyone	Food & Drink	May 14, 2018	3.1.46 (CY02)	4.3 and up
7430	DV 2019 - EDV Photo & Form	BOOKS_AND_REFERENCE	3.9	314	2.6M	50,000+	Free	0	Everyone	Books & Reference	October 24, 2016	6.0.0	4.0.3 and up
7348	DS Thermometer	WEATHER	3.7	631	3.0M	100,000+	Free	0	Everyone	Weather	May 30, 2015	2.2	2.3 and up
2584	Hostelworld: Hostels & Cheap Hotels Travel App	TRAVEL_AND_LOCAL	4.4	17878	28M	1,000,000+	Free	0	Everyone	Travel & Local	July 16, 2018	6.7.1	4.1 and up
6857	FANDOM for: DC	FAMILY	4.4	6715	9.5M	500,000+	Free	0	Mature 17+	Entertainment	July 20, 2018	2.9.8.1	4.4 and up

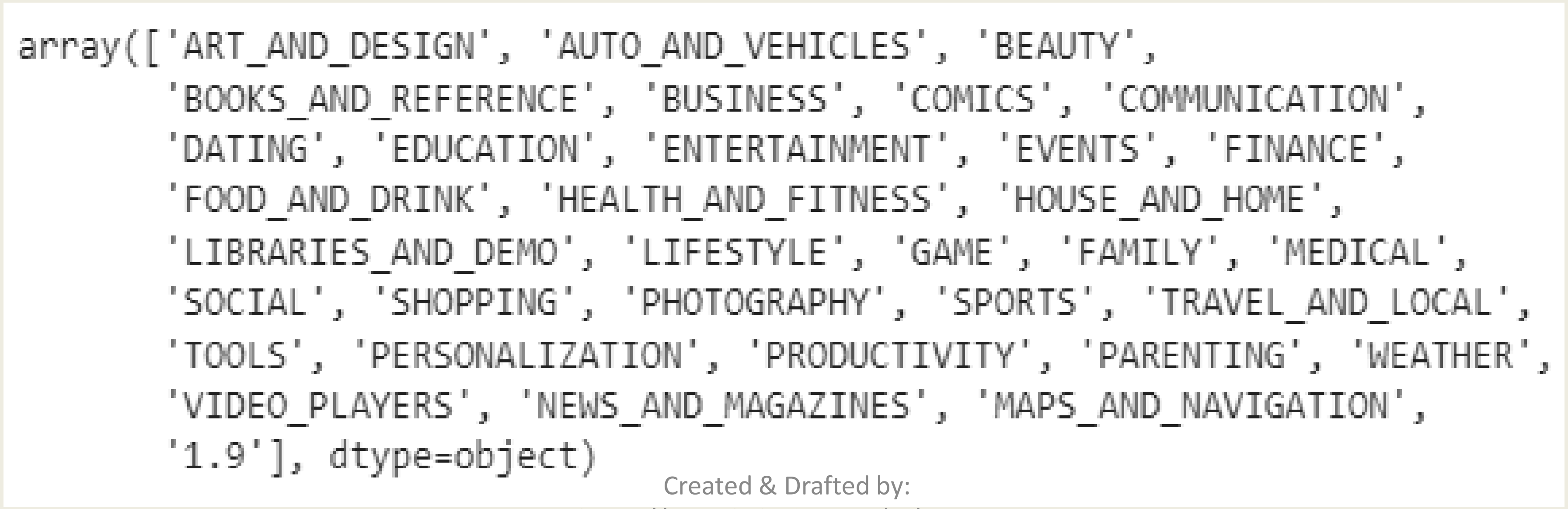
Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

STEP 6:

df['Category'].unique()

- ***The df['Category'].unique() method returns an array of all unique values in the 'Category' column of the DataFrame df. This can be useful for getting a quick overview of the different categories of apps in the dataset.***

- ***By calling unique() on the 'Category' column of df, we can see all the unique values of the 'Category' column. This will help us to understand the different categories of apps that are available on the Google Play Store, and to group or analyze the data based on these categories if necessary.***



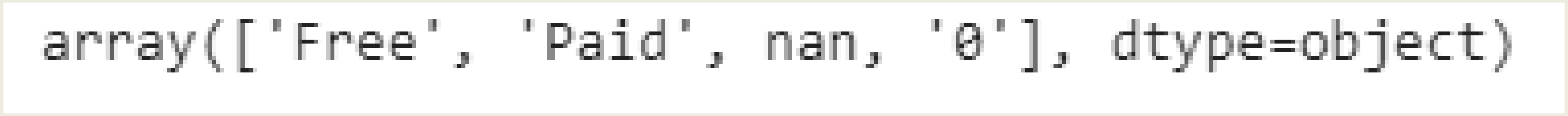
```
array(['ART_AND_DESIGN', 'AUTO_AND_VEHICLES', 'BEAUTY',
       'BOOKS_AND_REFERENCE', 'BUSINESS', 'COMICS', 'COMMUNICATION',
       'DATING', 'EDUCATION', 'ENTERTAINMENT', 'EVENTS', 'FINANCE',
       'FOOD_AND_DRINK', 'HEALTH_AND_FITNESS', 'HOUSE_AND_HOME',
       'LIBRARIES_AND_DEMO', 'LIFESTYLE',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

STEP 7:

df['Type'].unique()

- ***The df['Type'].unique() method returns an array of all unique values in the 'Type' column of the DataFrame df. This can be useful for getting a quick overview of the different types of apps in the dataset.***
- ***By calling unique() on the 'Type' column of df, we can see all the unique values of the 'Type' column. In this case, we expect to see two possible values: 'Free' and 'Paid', which indicate whether an app is free or requires payment to download or use.***
- ***This information can be useful for understanding the distribution of free vs. paid apps in the dataset, and for analyzing the pricing strategies of different app developers.***

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/



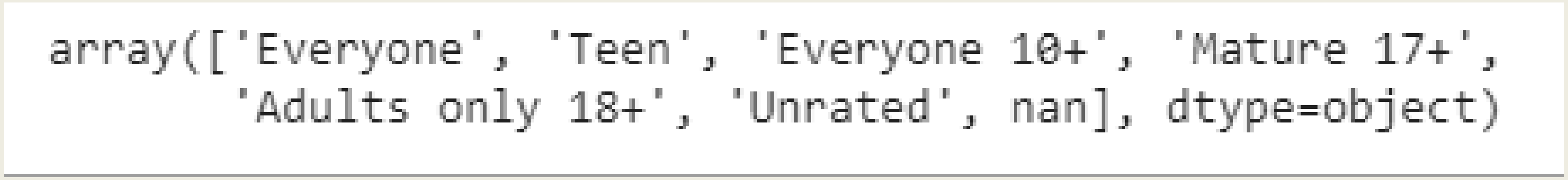
```
array(['Free', 'Paid', nan, '0'], dtype=object)
```

STEP 8:

df['Content Rating'].unique()

- *The df['Content Rating'].unique() method returns an array of all unique values in the 'Content Rating' column of the DataFrame df. This can be useful for getting a quick overview of the different content ratings used for apps in the dataset.*
- *By calling unique() on the 'Content Rating' column of df, we can see all the unique values of the 'Content Rating' column. In this case, we expect to see values like 'Everyone', 'Teen', 'Mature 17+', etc., which indicate the age rating of the app content based on the content rating system used by the Google Play Store.*

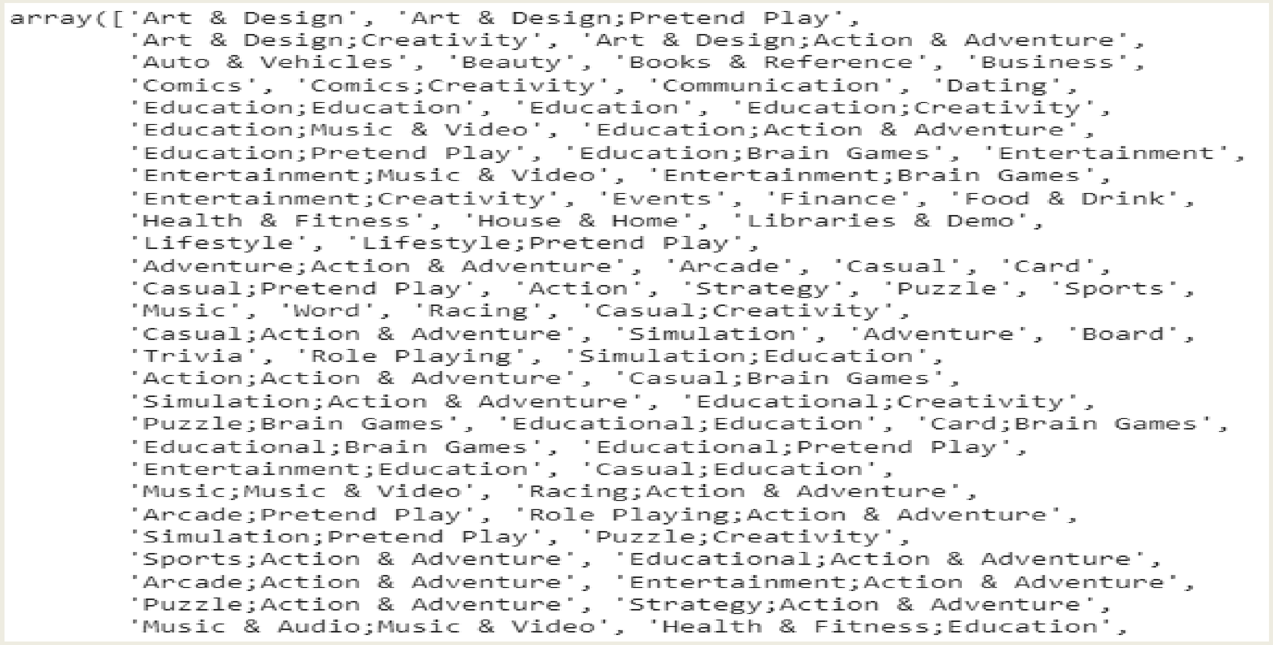
- *This information can be useful for understanding the types of apps that are available in the dataset, and for analyzing differences in user preferences or app usage across different age groups.*



```
array(['Everyone', 'Teen', 'Everyone 10+', 'Mature 17+',
       'Adults only 18+', 'Unrated', nan], dtype=object)
```

STEP 9:

df['Genres'].unique()



```
array(['Art & Design', 'Art & Design;Pretend Play',
       'Art & Design;Creativity', 'Art & Design;Action & Adventure',
       'Auto & Vehicles', 'Beauty', 'Books & Reference', 'Business',
       'Comics', 'Comics;Creativity', 'Communication', 'Dating',
       'Education;Education', 'Education', 'Education;Creativity',
       'Education;Music & Video', 'Education;Action & Adventure',
       'Education;Pretend Play', 'Education;Brain Games', 'Entertainment',
       'Entertainment;Music & Video', 'Entertainment;Brain Games',
       'Entertainment;Creativity', 'Events', 'Finance', 'Food & Drink',
       'Health & Fitness', 'House & Home', 'Libraries & Demo',
       'Lifestyle', 'Lifestyle;Pretend Play',
       'Adventure;Action & Adventure', 'Arcade', 'Casual', 'Card',
       'Casual;Pretend Play', 'Action', 'Strategy', 'Puzzle', 'Sports',
       'Music', 'Word', 'Racing', 'Casual;Creativity',
       'Casual;Action & Adventure', 'Simulation', 'Adventure', 'Board',
       'Trivia', 'Role Playing', 'Simulation;Education',
       'Action;Action & Adventure', 'Casual;Brain Games',
       'Simulation;Action & Adventure', 'Educational;Creativity',
       'Puzzle;Brain Games', 'Educational;Education', 'Card;Brain Games',
       'Educational;Brain Games', 'Educational;Pretend Play',
       'Entertainment;Education', 'Casual;Education',
       'Music;Music & Video', 'Racing;Action & Adventure',
       'Arcade;Pretend Play', 'Role Playing;Action & Adventure',
       'Simulation;Pretend Play', 'Puzzle;Creativity',
       'Sports;Action & Adventure', 'Educational;Action & Adventure',
       'Arcade;Action & Adventure', 'Entertainment;Action & Adventure',
       'Puzzle;Action & Adventure', 'Strategy;Action & Adventure',
       'Music & Audio;Music & Video', 'Health & Fitness;Education',
```

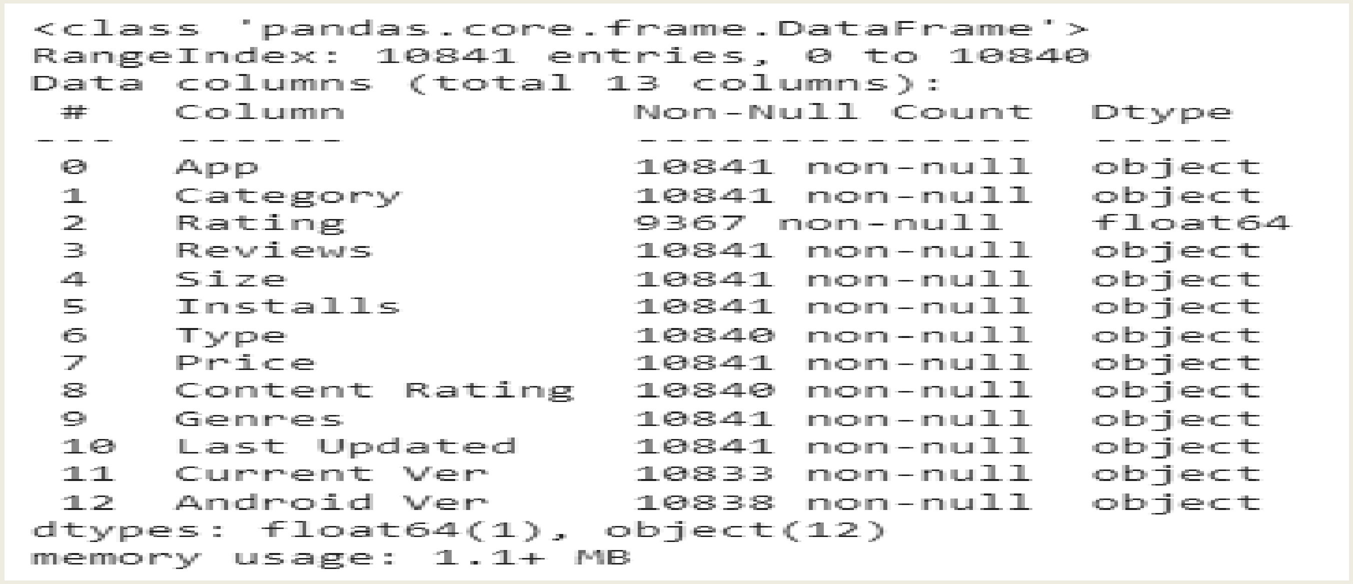
- ***The df['Genres'].unique() method returns an array of all unique values in the 'Genres' column of the DataFrame df. This can be useful for getting a quick overview of the different genres of apps in the dataset.***
- ***By calling unique() on the 'Genres' column of df, we can see all the unique values of the 'Genres' column. In this case, we expect to see values like 'Tools', 'Communication', 'Action', etc., which indicate the genre or category of the app.***

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- ***This information can be useful for understanding the types of apps that are available in the dataset, and for analyzing user preferences or app usage patterns across different genres. It can also be helpful for grouping or filtering the data based on specific genres of interest.***

STEP 10:

df.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
```

The df.info() method provides a summary of the DataFrame df, including information about the number of non-null values in each column, the data type of each column, and the memory usage of the DataFrame.

- ***The output of df.info() typically includes the following information:***
- ***The number of rows in the DataFrame (RangeIndex)***
- ***The name and data type of each column in the DataFrame (Data columns)***
- ***The number of non-null values in each column (Non-Null Count)***
- ***The data type of each column (Dtype)***
- ***The memory usage of the DataFrame (memory usage)***
- ***By examining the output of df.info(), we can get a better understanding of the size and structure of the dataset, and identify any issues that may need to be addressed before we begin analyzing the data.***

STEP 12:

```
# Remove rows with Size = '1,000+'
index = df[df['Size'] == '1,000+'].index
df.drop(axis=0, inplace=True, index=index)

# Convert Size column to float
df['Size'] = df['Size'].astype(str)
df['Size'] = df['Size'].str.replace('M','')
df['Size'] = df['Size'].str.replace('k','')
df['Size'] = pd.to_numeric(df['Size'], errors='coerce')
df['Size'] = df['Size'].fillna(0)
df['Size'] = df['Size'] / 1024  # Convert kilobytes to megabytes
df['Size'] = df['Size'].astype(float)
```

STEP 12:

- *The code provided drops the rows from the DataFrame df where the 'Size' column has a value of '1,000+'. This is likely done because '1,000+' represents an ambiguous and potentially erroneous size value.*
- *After removing the '1,000+' rows, the code calls a function called clean_sizes() on the 'Size' column of the DataFrame df. This function cleans the size data by removing the 'M' and 'k' characters and converting kilobytes to megabytes. If the size value is 'Varies with device', it is replaced with the value 0.*
- *The cleaned size values are added to a new list called cleaned_data. The main code then replaces the 'Size' column in df with the cleaned size data and converts the values to float data type using the astype() method.*
- *This code is useful for cleaning and preparing the size data for analysis. By converting the size data to a numeric format (in megabytes), we can perform mathematical operations on the data and gain insights into the distribution of app sizes and their relationship with other variables in the dataset.*

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

STEP 13:

```
installs = [i for i in df['Installs']]

def clean_installs(installs_list):
    cleaned_data = []
    for install in Installs_list:
        if ',' in install:
            install = install.replace(',', '')
        if '+' in install:
            install = install.replace('+', '')
        install = int(install)
        cleaned_data.append(install)
    return cleaned_data

df['Installs'] = clean_installs(installs)
df['Installs'] = df['Installs'].astype(float)
```

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

STEP 13:

- *The code provided defines a function called clean_installs() that cleans the 'Installs' column of the DataFrame df.*
- *First, it creates a new list called installs that contains all the values from the 'Installs' column of df. The clean_installs() function then iterates through each value in the installs list and performs the following actions:*
- *Removes any commas (',') from the string representation of the value using the replace() method.*

- *Removes any plus signs ('+') from the string representation of the value using the replace() method.*
- *Converts the cleaned value to an integer using the int() function.*
- *Appends the cleaned integer value to a new list called cleaned_data.*
- *Finally, the code replaces the 'Installs' column in df with the cleaned data and converts the values to float data type using the astype() method.*
- *This code is useful for preparing the 'Installs' data for analysis. By converting the string data to a numeric format, we can perform mathematical operations on the data and gain insights into the distribution of app installs and their relationship with other variables in the dataset.*

STEP 14:

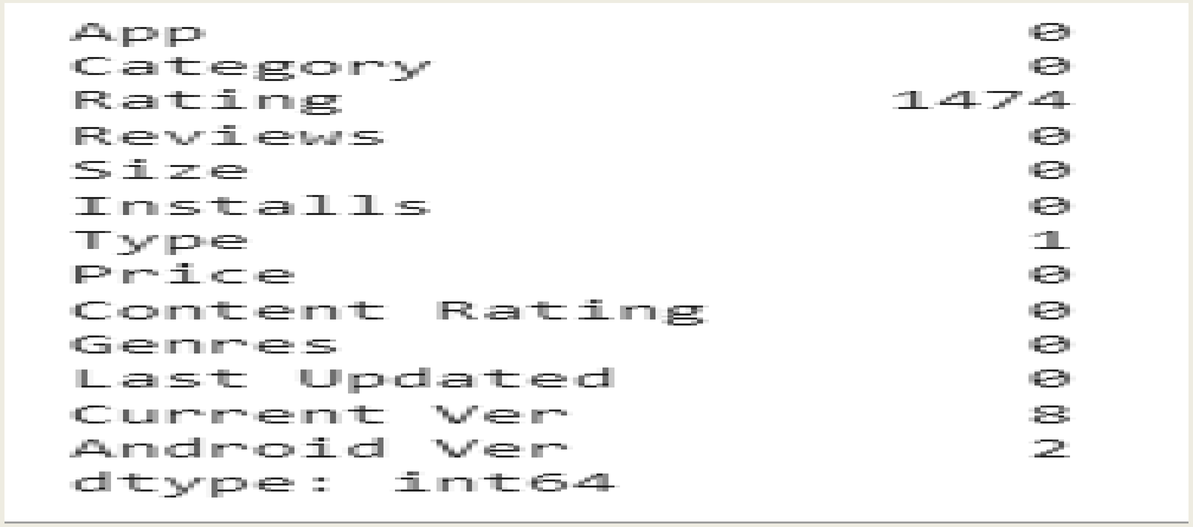
```
prices = [i for i in df['Price']]

def clean_prices(prices_list):
    cleaned_data = []
    for price in prices_list:
        if '$' in price:
            price = price.replace('$', '')
        cleaned_data.append(price)
    return cleaned_data

df['Price'] = clean_prices(prices)
df['Price'] = df['Price'].astype(float)
```

STEP 14:

- *The code you provided seems to be a Python script that cleans a list of prices and converts them to float type. Here's how it works:*
- *The variable "prices" is created by extracting the values in the "Price" column of a dataframe "df".*
- *The function "clean_prices" takes a list of prices as input and cleans them by removing the dollar sign ('$') from each price string, and then appending the cleaned value to a new list called "cleaned_data".*
- *The cleaned_data list is returned by the function.*
- *The "clean_prices" function is called with the "prices" list as its argument, and the returned cleaned_data list is assigned back to the "Price" column of the dataframe "df".*
- *Finally, the "Price" column of "df" is converted to float type using the "astype" method.*
- *Assuming that the input dataframe "df" has a column called "Price" with string values that include a dollar sign ('$'), this code will remove the dollar sign from each value and convert the resulting string to a float. This is useful if you need to perform numerical operations on the price data, since float values are easier to work with than strings.*

STEP 15:

df.isna().sum()



```
App                    0
Category               0
Rating              1474
Reviews                0
Size                   0
Installs               0
Type                   1
Price                  0
Content Rating         0
Genres                 0
Last Updated           0
Current Ver            8
Android Ver            2
dtype: int64
```

- ***The code you provided is used to check for the number of missing values (i.e., NaN values) in each column of a dataframe "df". Here's how it works:***
- ***The "isna()" method is called on the dataframe "df", which returns a boolean dataframe of the same shape as "df" with True values where "df" has missing values (i.e., NaN values) and False values elsewhere.***
- ***The "sum()" method is called on the boolean dataframe returned by "isna()". This sums up the True values for each column (since True is treated as 1 and False as 0), and returns a series with the sum of missing values for each column of "df".***
- ***The resulting series shows the number of missing values for each column of "df".***

- ***In summary, the code "df.isna().sum()" is a useful way to quickly check for missing values in a dataframe and determine which columns have the most missing data.***

STEP 16:

```
def replace_with_median(series):
    """
    Given a series, replace the rows with null values
    with median values
    """
    return series.fillna(series.median())

df['Rating'] = df['Rating'].transform(replace_with_median)
df['Rating'] = df['Rating'].astype(float)
```

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- *The code you provided is used to replace missing (i.e., NaN) values in a dataframe column called "Rating" with the median value of the column, and then convert the column to a float type. Here's how it works:*

- *The function "replace_with_median" takes a series as input, which is assumed to be a column of a dataframe. It fills in any missing values in the series with the median value of the non-missing values in the series, using the "fillna()" method with the argument "series.median()".*

STEP 16:

- *The "transform()" method is called on the "Rating" column of the dataframe "df", passing the "replace_with_median" function as an argument. This applies the function to each element of the "Rating" column and replaces missing values with the median of the non-missing values in the column.*

- *The "astype()" method is called on the "Rating" column of the dataframe "df", passing the "float" type as an argument. This converts the "Rating" column from whatever type it was (e.g., string, integer) to a float type.*

- *In summary, the code*
  "df['Rating'] = df['Rating'].transform(replace_with_median); df['Rating'] = df['Rating'].astype(float)" is a useful way to replace missing values in a column of a dataframe with the median of the non-missing values in that column, and then convert the column to a float type for further analysis.

STEP 17:

```
index = df[df['Type'].isna()].index
df.drop(axis=0, inplace=True, index=index)
```

- *The code you provided drops rows from a dataframe "df" where the value of the "Type" column is missing (i.e., NaN). Here's how it works:*

- *The "isna()" method is called on the "Type" column of the dataframe "df", which returns a boolean series of the same length as "df" with True values where "df" has missing values (i.e., NaN values) in the "Type" column, and False values elsewhere.*

- *The "index" attribute is called on the boolean series returned by "isna()", which returns the index labels of the rows where the "Type" column has missing values.*

STEP 17:

- *The "drop()" method is called on the dataframe "df" with three arguments: "axis=0" (which indicates that rows should be dropped), "inplace=True" (which indicates that the changes should be made to "df" directly rather than returning a new dataframe), and "index=index" (which specifies the index labels of the rows to be dropped).*

- *The resulting dataframe has all rows where the "Type" column had missing values removed.*

- *In summary, the code "index = df[df['Type'].isna()].index; df.drop(axis=0, inplace=True, index=index)" is a useful way to remove rows from a dataframe where a specific column has missing values. This can be helpful when analyzing data, since missing values can cause errors in computations or lead to inaccurate results.*

STEP 18:

```
df = df.groupby(['App', 'Reviews', 'Category', 'Rating', 'Size', 'Installs', 'Type', 'P
rice', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver'],
as_index=False)
df = df['Installs'].mean()
df.sort_values(by='Reviews', ascending=False, inplace=True)
df.drop_duplicates(subset=['App'], inplace=True)
df
```

- ***The code you provided is used to perform several operations on a dataframe "df" and generate a new dataframe with aggregated and cleaned data. Here's how it works:***

- ***The "groupby()" method is called on the dataframe "df" with multiple column names as arguments. This groups the rows of the dataframe by the specified columns and returns a "groupby" object.***
- ***The "mean()" method is called on the "Installs" column of the "groupby" object. This calculates the mean value of the "Installs" column for each group of rows that share the same values in the other columns.***
- ***The resulting dataframe has a multi-level index based on the columns used for grouping, with a single column "Installs" containing the mean value for each group.***

STEP 19:

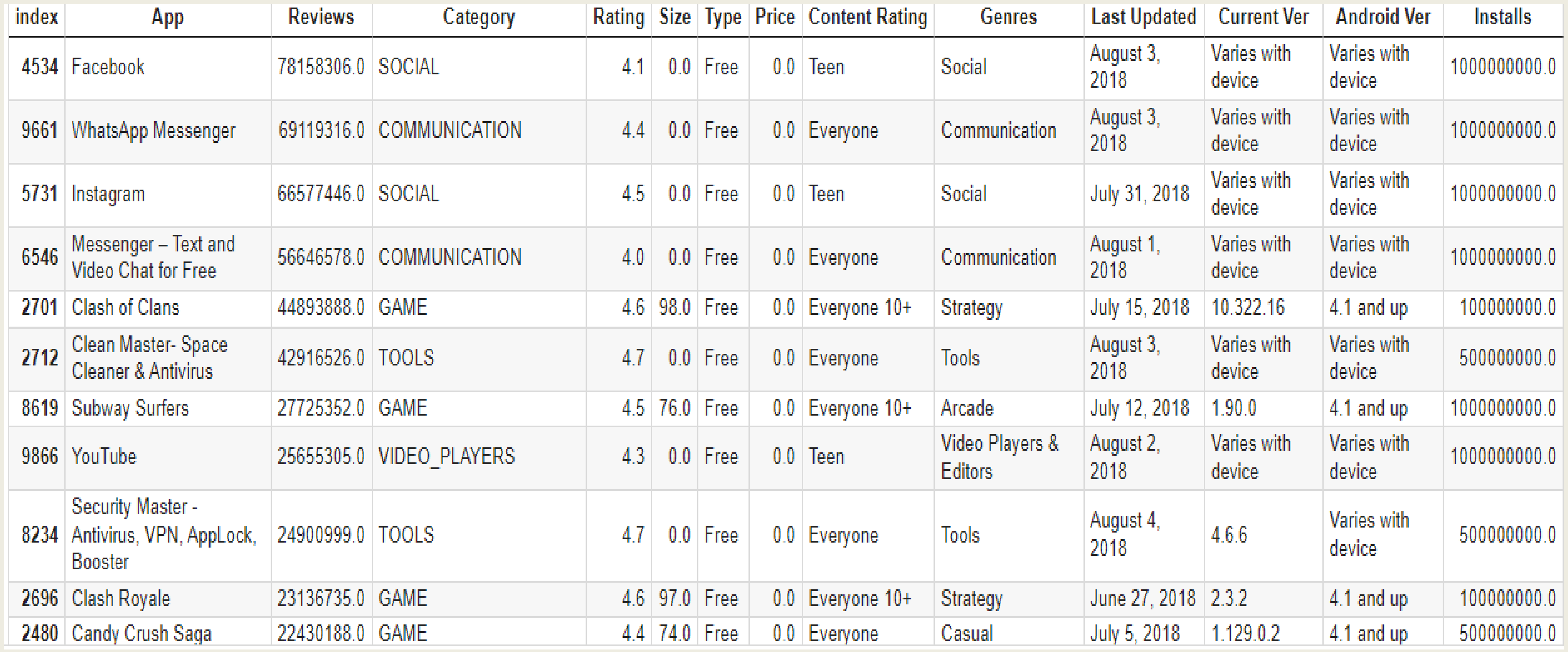
- *The "sort_values()" method is called on the resulting dataframe with two arguments: "by='Reviews'" (which indicates that the dataframe should be sorted by the "Reviews" column) and "ascending=False" (which indicates that the sort should be in descending order).*

- *The "drop_duplicates()" method is called on the resulting dataframe with the argument "subset=['App']". This drops any rows where the "App" column has a duplicate value, keeping only the first occurrence of each unique "App" value.*

- *The resulting dataframe has all rows sorted in descending order by the "Reviews" column, and with duplicates removed based on the "App" column.*

- *In summary, the code you provided is a useful way to clean and aggregate data in a dataframe by grouping rows based on multiple columns, calculating summary statistics for each group, and removing duplicates based on a specific column. This can be helpful when analyzing large datasets with many duplicate or inconsistent values.*

STEP 19:

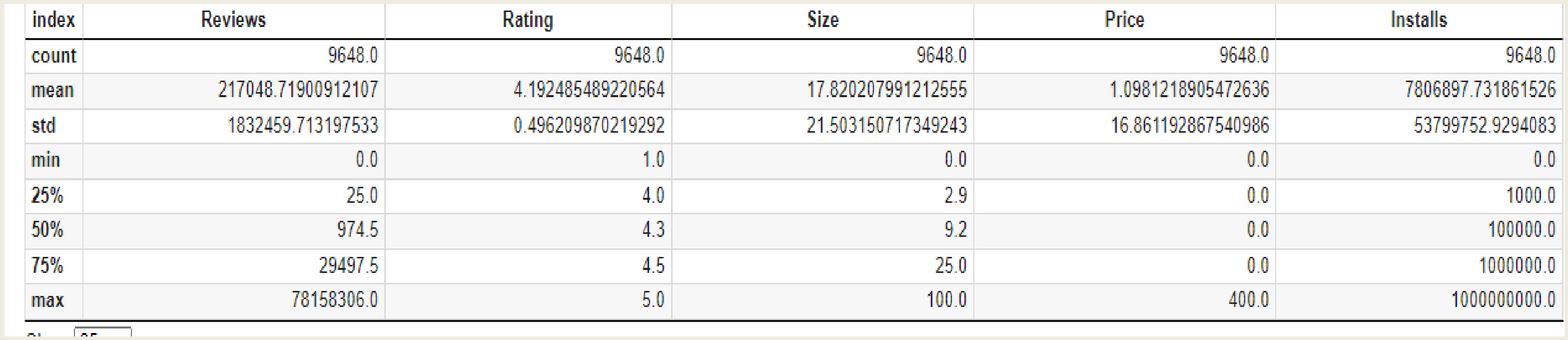


index	App	Reviews	Category	Rating	Size	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Installs
4534	Facebook	78158306.0	SOCIAL	4.1	0.0	Free	0.0	Teen	Social	August 3, 2018	Varies with device	Varies with device	1000000000.0
9661	WhatsApp Messenger	69119316.0	COMMUNICATION	4.4	0.0	Free	0.0	Everyone	Communication	August 3, 2018	Varies with device	Varies with device	1000000000.0
5731	Instagram	66577446.0	SOCIAL	4.5	0.0	Free	0.0	Teen	Social	July 31, 2018	Varies with device	Varies with device	1000000000.0
6546	Messenger – Text and Video Chat for Free	56646578.0	COMMUNICATION	4.0	0.0	Free	0.0	Everyone	Communication	August 1, 2018	Varies with device	Varies with device	1000000000.0
2701	Clash of Clans	44893888.0	GAME	4.6	98.0	Free	0.0	Everyone 10+	Strategy	July 15, 2018	10.322.16	4.1 and up	100000000.0
2712	Clean Master- Space Cleaner & Antivirus	42916526.0	TOOLS	4.7	0.0	Free	0.0	Everyone	Tools	August 3, 2018	Varies with device	Varies with device	500000000.0
8619	Subway Surfers	27725352.0	GAME	4.5	76.0	Free	0.0	Everyone 10+	Arcade	July 12, 2018	1.90.0	4.1 and up	1000000000.0
9866	YouTube	25655305.0	VIDEO_PLAYERS	4.3	0.0	Free	0.0	Teen	Video Players & Editors	August 2, 2018	Varies with device	Varies with device	1000000000.0
8234	Security Master - Antivirus, VPN, AppLock, Booster	24900999.0	TOOLS	4.7	0.0	Free	0.0	Everyone	Tools	August 4, 2018	4.6.6	Varies with device	500000000.0
2696	Clash Royale	23136735.0	GAME	4.6	97.0	Free	0.0	Everyone 10+	Strategy	June 27, 2018	2.3.2	4.1 and up	100000000.0
2480	Candy Crush Saga	22430188.0	GAME	4.4	74.0	Free	0.0	Everyone	Casual	July 5, 2018	1.129.0.2	4.1 and up	500000000.0

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

STEP 20:

df.describe()



index	Reviews	Rating	Size	Price	Installs
count	9648.0	9648.0	9648.0	9648.0	9648.0
mean	217048.71900912107	4.192485489220564	17.820207991212555	1.0981218905472636	7806897.731861526
std	1832459.713197533	0.496209870219292	21.503150717349243	16.861192867540986	53799752.9294083
min	0.0	1.0	0.0	0.0	0.0
25%	25.0	4.0	2.9	0.0	1000.0
50%	974.5	4.3	9.2	0.0	100000.0
75%	29497.5	4.5	25.0	0.0	1000000.0
max	78158306.0	5.0	100.0	400.0	1000000000.0

- ***The describe() method in pandas is used to generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution. When applied to a dataframe df, it will compute the summary statistics for each numerical column in the dataframe.***
- ***Here is an overview of the statistics computed by describe():***
- ***count: number of non-null values in the column.***
- ***mean: arithmetic mean of the values in the column.***
- ***std: standard deviation of the values in the column.***
- ***min: minimum value in the column.***
- ***25%: 25th percentile value in the column.***
- ***50%: 50th percentile value (median) in the column.***
- ***75%: 75th percentile value in the column.***
- ***max: maximum value in the column.***

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- ***Note that describe() only computes the summary statistics for numerical columns, and skips columns with non-numeric data types (e.g., string, datetime).***
- ***When you run df.describe(), it will return a table containing the summary statistics for all numerical columns in the dataframe df.***

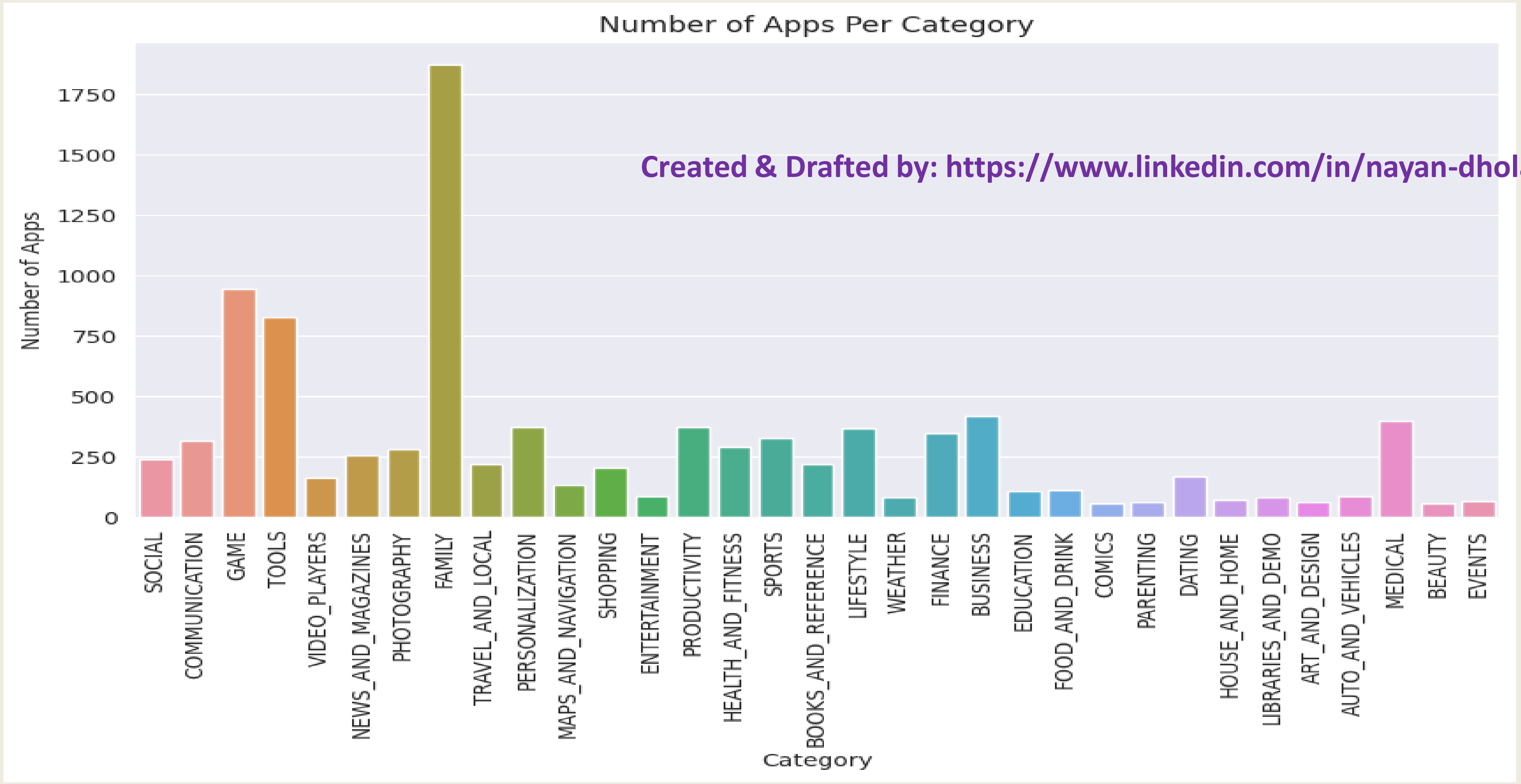
STEP 21:

```
sns.set_style('darkgrid')
plt.figure(figsize=(10, 5))
sns.countplot(x='Category', data=df)
plt.title('Number of Apps Per Category')
plt.xticks(rotation=90)
plt.ylabel('Number of Apps')
plt.show()
```

- ***The code you provided uses the seaborn library to create a countplot of the number of apps per category in the dataframe df. Here is what each line of the code does:***
- ***sns.set_style('darkgrid'): sets the plotting style to "darkgrid" for the seaborn library.***
- ***plt.figure(figsize=(10, 5)): creates a new figure with a width of 10 inches and a height of 5 inches.***
- ***sns.countplot(x='Category', data=df): creates a countplot using the seaborn library, with the "Category" column as the x-axis and the dataframe df as the data source.***
- ***plt.title('Number of Apps Per Category'): adds a title to the plot.***
- ***plt.xticks(rotation=90): rotates the x-axis labels by 90 degrees for better readability.***

STEP 21:

- ***plt.ylabel('Number of Apps'): adds a label to the y-axis.***
- ***plt.show(): displays the plot.***
- ***Overall, this code is a simple and effective way to visualize the distribution of app categories in the dataframe df. The countplot shows the number of apps in each category, making it easy to compare and identify the most popular categories. The rotation of the x-axis labels and the addition of a title and y-axis label help make the plot more informative and easier to read.***



STEP 22:

```
categories = df.groupby('Category')
category_installs_sum_df = categories[['Installs']].sum()
category_installs_sum_df = category_installs_sum_df.reset_index()
plt.figure(figsize=(10, 5))
sns.barplot(x='Category', y='Installs', data=category_installs_sum_df)
plt.xticks(rotation=90)
plt.ylabel('Installs (e+10)')
plt.title('Number of Installs For Each Category')
plt.show()
```

- *The code you provided uses the seaborn library to create a barplot of the number of installs for each app category in the dataframe df. Here is what each line of the code does:*
- *categories = df.groupby('Category'): groups the dataframe df by the "Category" column.*

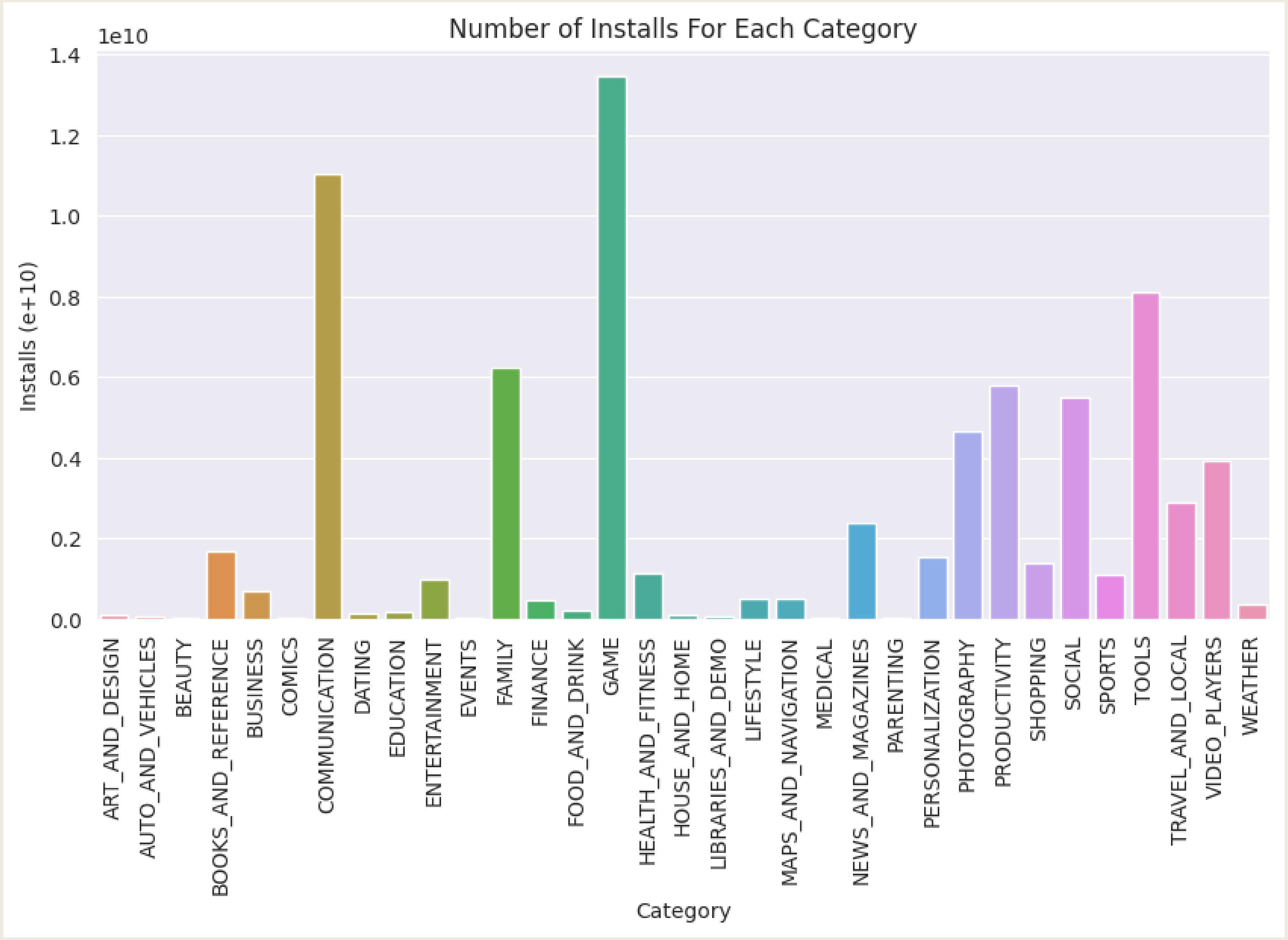
- *category_installs_sum_df = categories[['Installs']].sum(): computes the total number of installs for each app category by summing the "Installs" column within each group, and creates a new dataframe category_installs_sum_df with the result.*
- *category_installs_sum_df = category_installs_sum_df.reset_index(): resets the index of the dataframe category_installs_sum_df.*

STEP 22:

- *plt.figure(figsize=(10, 5)): creates a new figure with a width of 10 inches and a height of 5 inches.*
- *sns.barplot(x='Category', y='Installs', data=category_installs_sum_df): creates a barplot using the seaborn library, with the "Category" column as the x-axis, the "Installs" column as the y-axis, and the dataframe category_installs_sum_df as the data source.*
- *plt.xticks(rotation=90): rotates the x-axis labels by 90 degrees for better readability.*
- *plt.ylabel('Installs (e+10)'): adds a label to the y-axis, representing the number of installs in scientific notation.*
- *plt.title('Number of Installs For Each Category'): adds a title to the plot.*
- *plt.show(): displays the plot.*
- *Overall, this code is a good way to compare the popularity of app categories based on the total number of installs. The barplot shows the total number of installs for each category, making it easy to identify the most popular categories. The rotation of the x-axis labels and the addition of a title and y-axis label help make the plot more informative and easier to read.*

STEP 22:

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/



STEP 23:

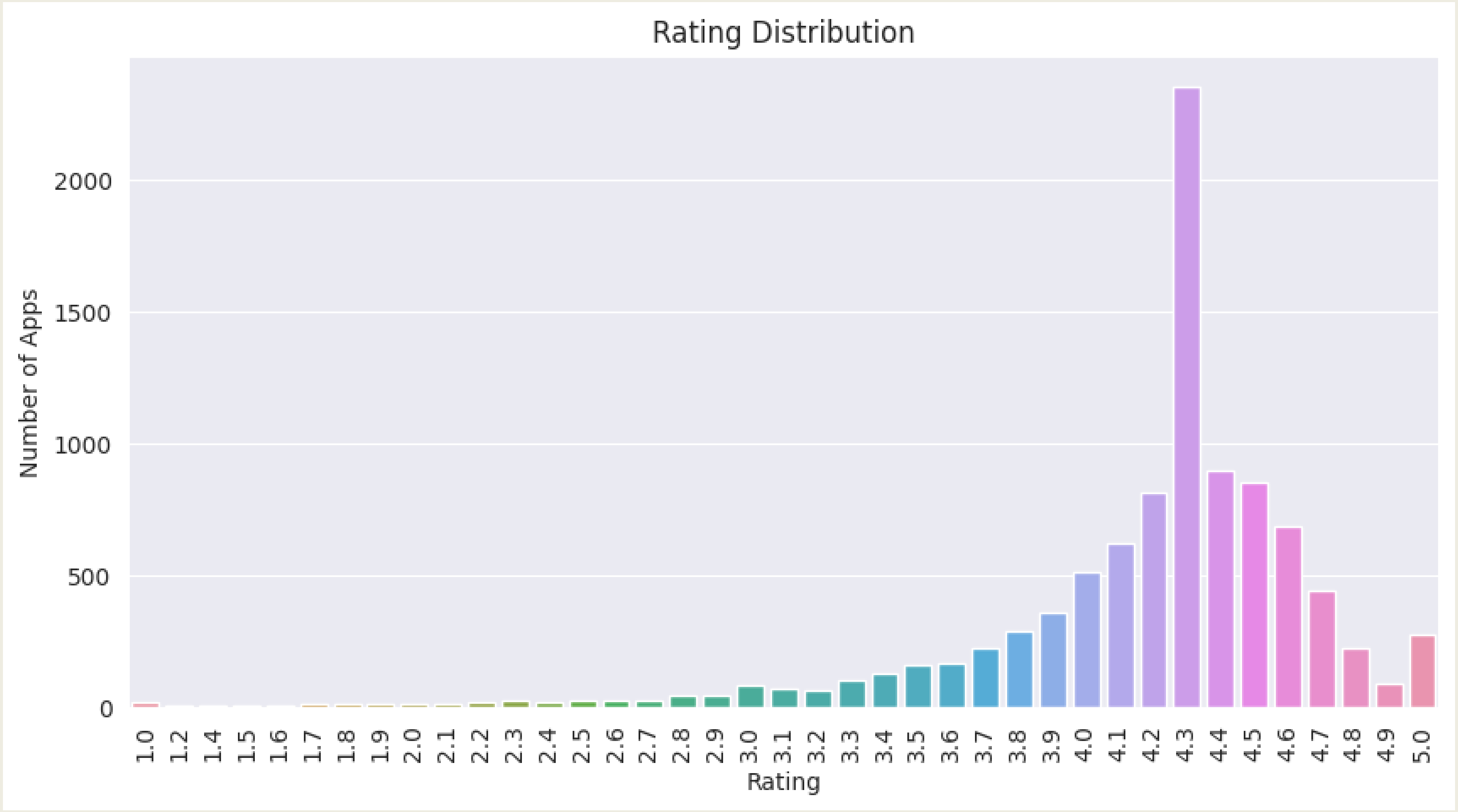
```
plt.figure(figsize=(10, 5))
sns.countplot(x='Rating',data=df)
plt.title('Rating Distribution')
plt.xticks(rotation=90)
plt.ylabel('Number of Apps')
plt.show()
```

- ***The code you provided creates a countplot of the rating distribution of apps in the dataframe df using the seaborn library. Here is what each line of the code does:***
- ***plt.figure(figsize=(10, 5)): creates a new figure with a width of 10 inches and a height of 5 inches.***

- ***sns.countplot(x='Rating',data=df): creates a countplot using the seaborn library, with the "Rating" column as the x-axis and the dataframe df as the data source.***
- ***plt.title('Rating Distribution'): adds a title to the plot.***
- ***plt.xticks(rotation=90): rotates the x-axis labels by 90 degrees for better readability.***
- ***plt.ylabel('Number of Apps'): adds a label to the y-axis, representing the number of apps with each rating.***
- ***plt.show(): displays the plot.***

STEP 23:

- *Overall, this code is a good way to visualize the distribution of app ratings in the dataframe df. The countplot shows how many apps have each rating, making it easy to see the most common ratings. The rotation of the x-axis labels and the addition of a title and y-axis label help make the plot more informative and easier to read.* **Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/**



STEP 24:

```
rating_df = df.groupby('Rating').sum().reset_index()

fig, axes = plt.subplots(1, 4, figsize=(14, 4))

axes[0].plot(rating_df['Rating'], rating_df['Reviews'], 'r')
axes[0].set_xlabel('Rating')
axes[0].set_ylabel('Reviews')
axes[0].set_title('Reviews Per Rating')

axes[1].plot(rating_df['Rating'], rating_df['Size'], 'g')
axes[1].set_xlabel('Rating')
axes[1].set_ylabel('Size')
axes[1].set_title('Size Per Rating')

axes[2].plot(rating_df['Rating'], rating_df['Installs'], 'g')
axes[2].set_xlabel('Rating')
axes[2].set_ylabel('Installs (e+10)')
axes[2].set_title('Installs Per Rating')

axes[3].plot(rating_df['Rating'], rating_df['Price'], 'k')
axes[3].set_xlabel('Rating')
axes[3].set_ylabel('Price')
axes[3].set_title('Price Per Rating')

plt.tight_layout(pad=2)
plt.show()
```

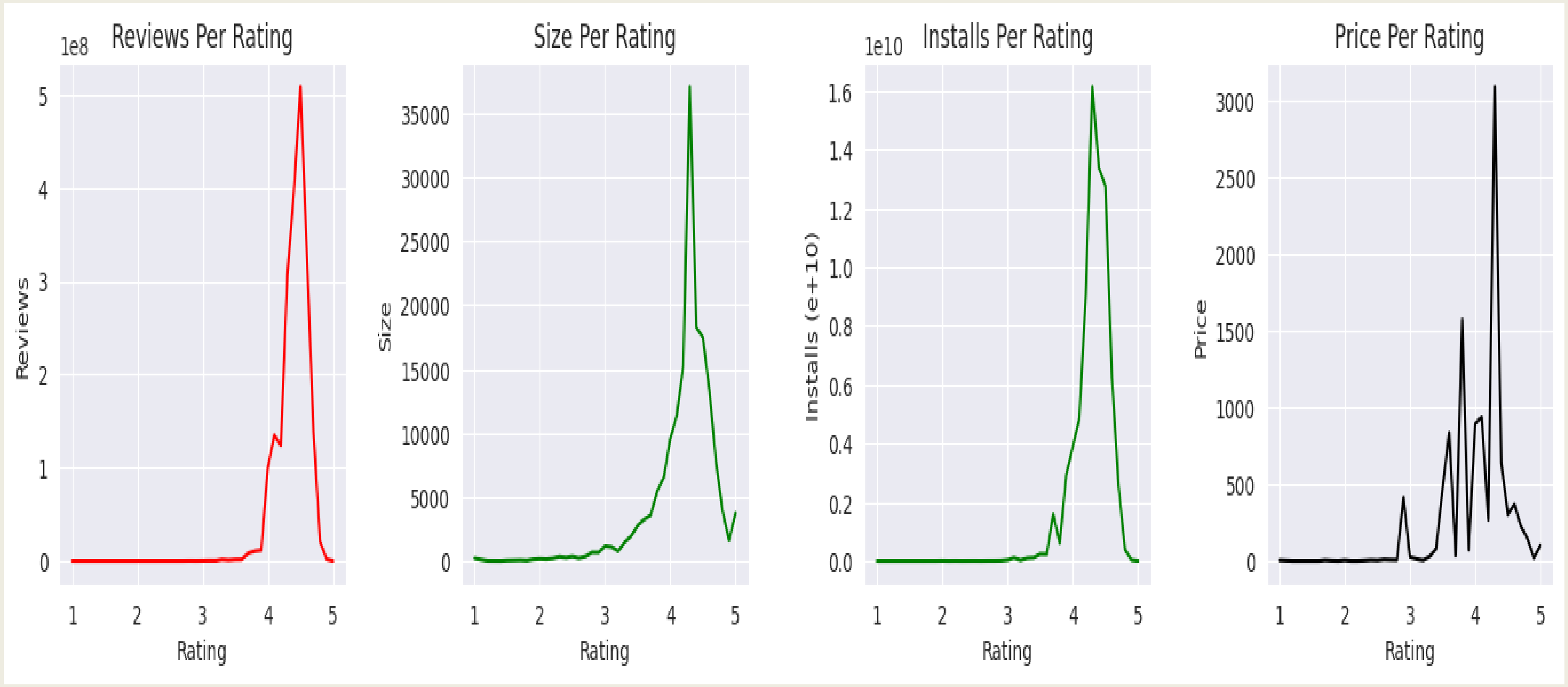
STEP 25:

- *The code you provided creates a figure with four subplots using the matplotlib library to visualize how different variables are related to the app rating in the dataframe df. Here is what each line of the code does:*
- *rating_df = df.groupby('Rating').sum().reset_index(): groups the dataframe df by the "Rating" column and calculates the sum of all other columns for each rating, then resets the index to a default range.*
- *fig, axes = plt.subplots(1, 4, figsize=(14, 4)): creates a new figure with one row and four columns, with a total width of 14 inches and a height of 4 inches. Returns the figure and an array of four axes objects.*
- *axes[0].plot(rating_df['Rating'], rating_df['Reviews'], 'r'): plots a line chart on the first subplot with the app rating on the x-axis and the sum of reviews for each rating on the y-axis, using red color.*
- *axes[0].set_xlabel('Rating'): adds a label to the x-axis.*
- *axes[0].set_ylabel('Reviews'): adds a label to the y-axis.*
- *axes[0].set_title('Reviews Per Rating'): adds a title to the subplot.*
- *The next three lines of code do the same as the first three lines, but for the other subplots, with different variables plotted and different titles and y-axis labels.*
- *plt.tight_layout(pad=2): adjusts the spacing between subplots to avoid overlapping.*
- *plt.show(): displays the figure.*

STEP 25:

- *Overall, this code is a good way to visualize the relationships between the app rating and different variables in the dataframe df. The four subplots show how the sum of reviews, size, installs, and price vary with the app rating, making it easy to see any patterns or trends in the data. The tight layout and labels make the figure informative and easy to read.*

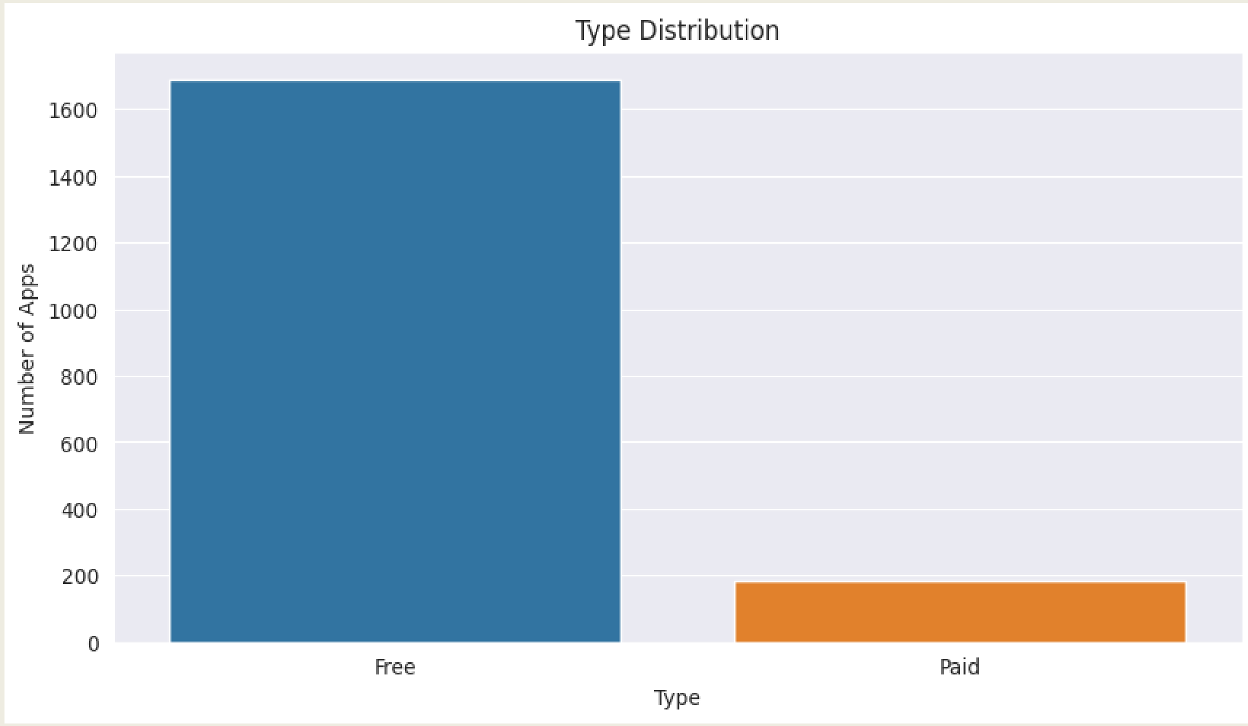
Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/



STEP 26:

```
df['Type'] = df['Type'].astype('category')

plt.figure(figsize=(10, 5))
sns.countplot(x='Type', data=df)
plt.title('Type Distribution')
plt.ylabel('Number of Apps')
plt.show()
```

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/



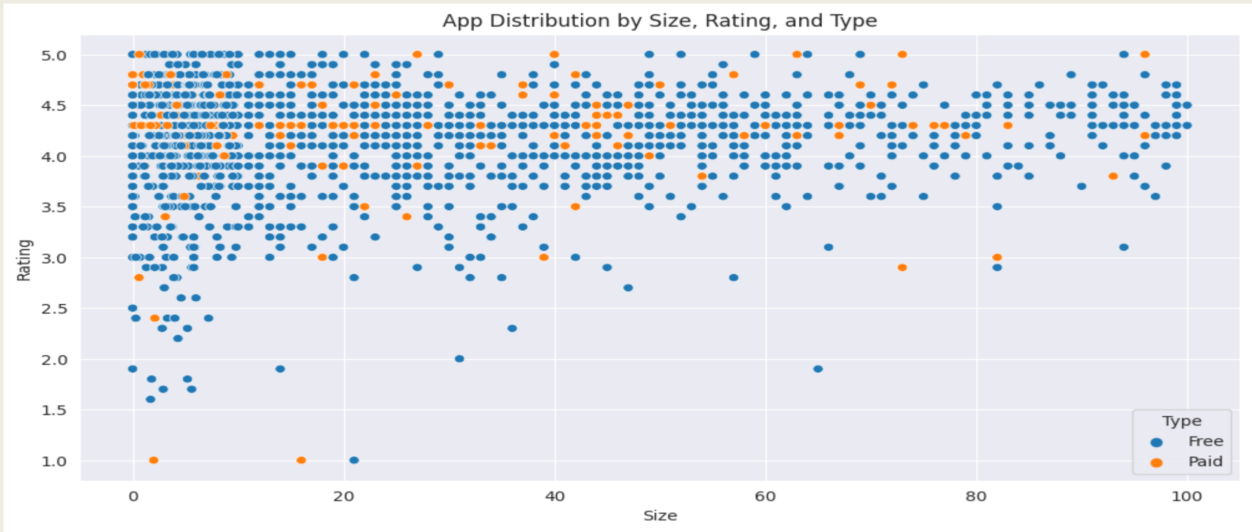
- *The plot shows the distribution of app types in the dataset. There are two types of apps, Free and Paid. The majority of apps in the dataset are free, with only a small percentage being paid. This suggests that users tend to prefer free apps over paid apps.*

STEP 27:

```
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Size',
                y='Rating',
                hue='Type',
                data=df)
plt.title('App Distribution by Size, Rating, and Type')
plt.show()
```

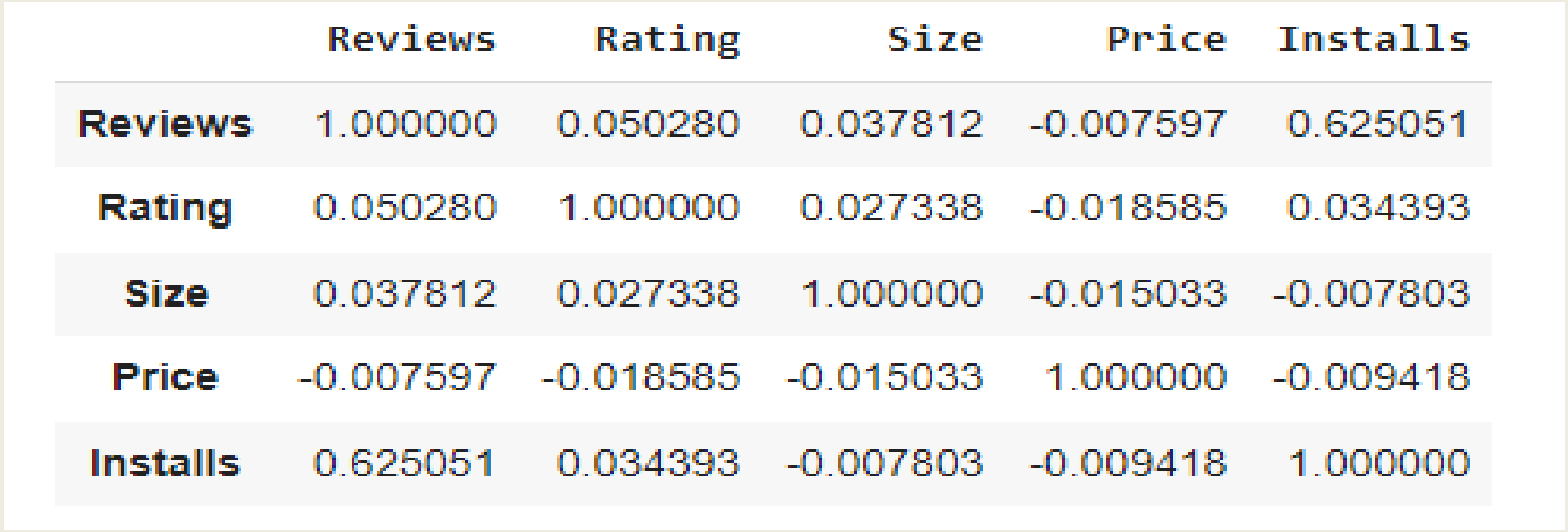
Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

- ➢ *This scatter plot shows the distribution of apps by size, rating, and type. The x-axis represents the size of the app, and the y-axis represents the rating. The color of the dots represents the type of the app, either free or paid.*
- ➢ *The plot shows that most apps are small in size, less than 50MB, and have a rating of around 4.0. Free apps are much more common than paid apps, and they tend to have slightly higher ratings. There is a wide range of ratings for both free and paid apps, from around 1.0 to 5.0. The plot also shows that there are some large apps, greater than 100MB, with high ratings.*



STEP 28:

df.corr()



	Reviews	Rating	Size	Price	Installs
Reviews	1.000000	0.050280	0.037812	-0.007597	0.625051
Rating	0.050280	1.000000	0.027338	-0.018585	0.034393
Size	0.037812	0.027338	1.000000	-0.015033	-0.007803
Price	-0.007597	-0.018585	-0.015033	1.000000	-0.009418
Installs	0.625051	0.034393	-0.007803	-0.009418	1.000000

- ***The corr() method in Pandas computes pairwise correlation of columns, excluding NA/null values.***

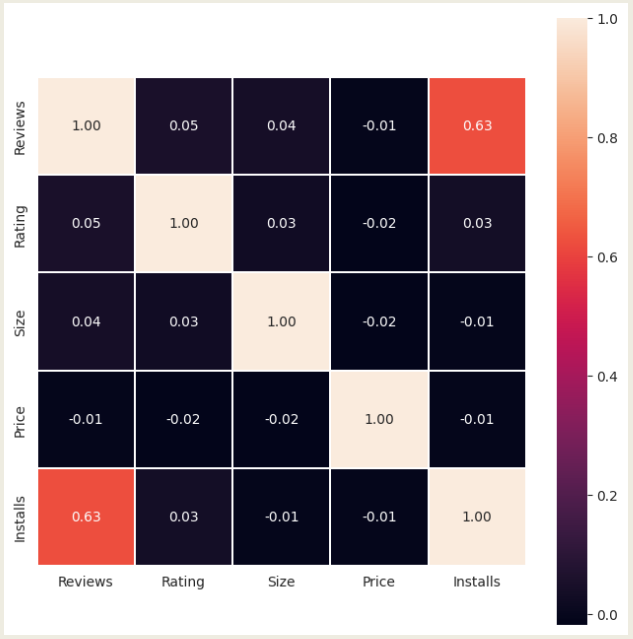
- ***From the correlation matrix, we can see that there is a weak positive correlation between Rating and Size, and a weak positive correlation between Size and Installs. There is no strong correlation between any of the numerical columns in the dataset. The negative correlation between Price and Rating is weak, but it indicates that as the price of an app increases, its rating tends to decrease slightly.***

STEP 28:

```
fig, axes = plt.subplots(figsize=(8, 8))
sns.heatmap(df.corr(), ax=axes, annot=True, linewidths=0.1, fmt='.2f', square=True)
plt.show()
```

- *The heatmap shows the correlation matrix between the numerical columns of the dataframe. The correlation ranges from -1 to 1, where values closer to 1 indicate a strong positive correlation and values closer to -1 indicate a strong negative correlation. A value of 0 indicates no correlation.*

- *In this specific dataframe, the heatmap shows a positive correlation between the number of reviews and the number of installs, as well as between the app size and the number of installs. There is also a weak positive correlation between the rating and the number of installs. On the other hand, the heatmap shows no significant correlation between the price and any of the other variables.*

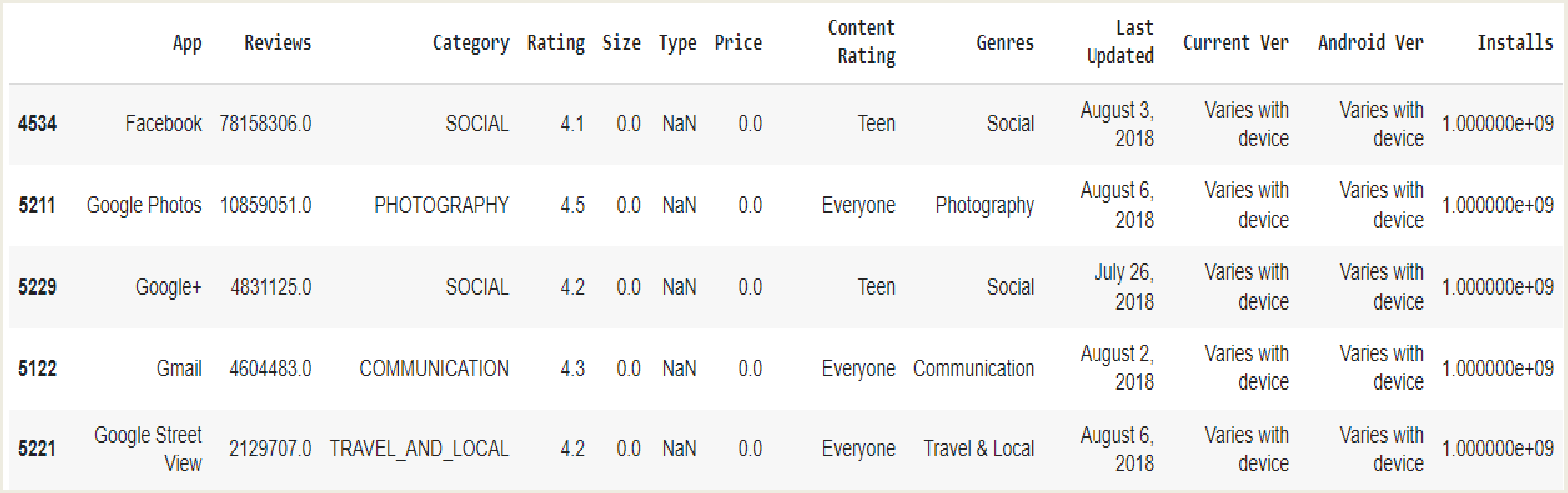


	Reviews	Rating	Size	Price	Installs
Reviews	1.00	0.05	0.04	-0.01	0.63
Rating	0.05	1.00	0.03	-0.02	0.03
Size	0.04	0.03	1.00	-0.02	-0.01
Price	-0.01	-0.02	-0.02	1.00	-0.01
Installs	0.63	0.03	-0.01	-0.01	1.00

STEP 29:

```
# 1. What is the top 5 apps on the basis of installs?
df = df.sort_values(by=['Installs'], ascending=False)
df.head(5)
```

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/



	App	Reviews	Category	Rating	Size	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Installs
4534	Facebook	78158306.0	SOCIAL	4.1	0.0	NaN	0.0	Teen	Social	August 3, 2018	Varies with device	Varies with device	1.000000e+09
5211	Google Photos	10859051.0	PHOTOGRAPHY	4.5	0.0	NaN	0.0	Everyone	Photography	August 6, 2018	Varies with device	Varies with device	1.000000e+09
5229	Google+	4831125.0	SOCIAL	4.2	0.0	NaN	0.0	Teen	Social	July 26, 2018	Varies with device	Varies with device	1.000000e+09
5122	Gmail	4604483.0	COMMUNICATION	4.3	0.0	NaN	0.0	Everyone	Communication	August 2, 2018	Varies with device	Varies with device	1.000000e+09
5221	Google Street View	2129707.0	TRAVEL_AND_LOCAL	4.2	0.0	NaN	0.0	Everyone	Travel & Local	August 6, 2018	Varies with device	Varies with device	1.000000e+09

STEP 30:

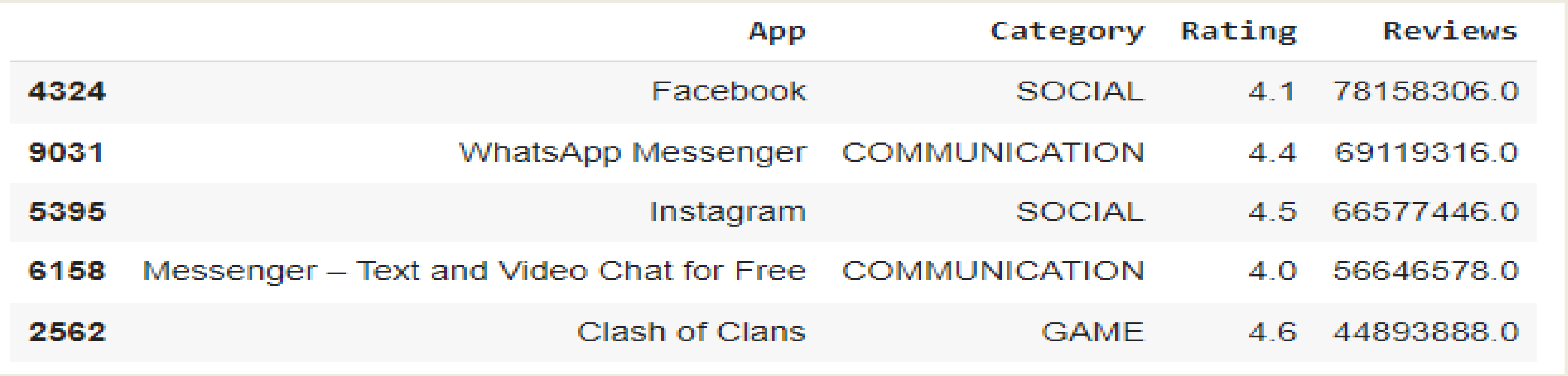
```
print(f'The 5 apps that have the most number of installs are: {", ".join(df["App"].
head(5))}')
```

Output:

```
The 5 apps that have the most number of installs are: Facebook, Google Photos, Google+,
Gmail, Google Street View
```

STEP 31:

```
# 2. What is the top 5 reviewed apps?
df = df.groupby(by=['App', 'Category', 'Rating'])[['Reviews']].sum().reset_index()
df = df.sort_values(by=['Reviews'], ascending=False)
df.head(5)
```



	App	Category	Rating	Reviews
4324	Facebook	SOCIAL	4.1	78158306.0
9031	WhatsApp Messenger	COMMUNICATION	4.4	69119316.0
5395	Instagram	SOCIAL	4.5	66577446.0
6158	Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56646578.0
2562	Clash of Clans	GAME	4.6	44893888.0

STEP 32:

```
print(f'The 5 apps that have the most number of total reviews are: {", ".join(df["App"].head(5))}')
```

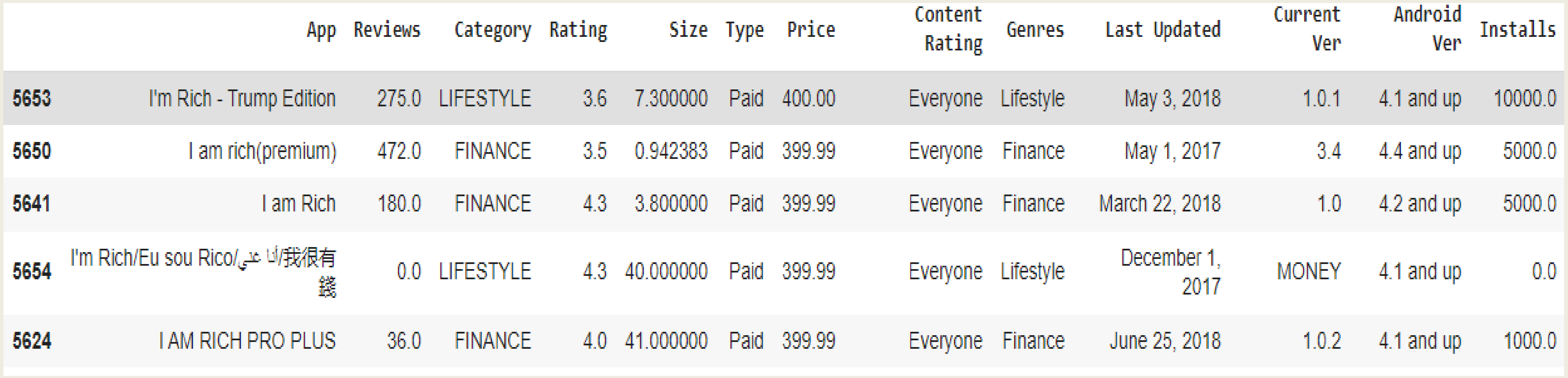
Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

Output:

```
The 5 apps that have the most number of total reviews are: Facebook, WhatsApp Messenger, Instagram, Messenger – Text and Video Chat for Free, Clash of Clans
```

STEP 33:

```
# 3. What is the top 5 expensive apps?
df = df.sort_values(by='Price', ascending=False)
top_5_expensive_apps = df.head(5)
top_5_expensive_apps
```



	App	Reviews	Category	Rating	Size	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Installs
5653	I'm Rich - Trump Edition	275.0	LIFESTYLE	3.6	7.300000	Paid	400.00	Everyone	Lifestyle	May 3, 2018	1.0.1	4.1 and up	10000.0
5650	I am rich(premium)	472.0	FINANCE	3.5	0.942383	Paid	399.99	Everyone	Finance	May 1, 2017	3.4	4.4 and up	5000.0
5641	I am Rich	180.0	FINANCE	4.3	3.800000	Paid	399.99	Everyone	Finance	March 22, 2018	1.0	4.2 and up	5000.0
5654	I'm Rich/Eu sou Rico/أنا غني/我很有錢	0.0	LIFESTYLE	4.3	40.000000	Paid	399.99	Everyone	Lifestyle	December 1, 2017	MONEY	4.1 and up	0.0
5624	I AM RICH PRO PLUS	36.0	FINANCE	4.0	41.000000	Paid	399.99	Everyone	Finance	June 25, 2018	1.0.2	4.1 and up	1000.0

STEP 34:

```
print(f'The top 5 most expensive apps in the store are: {", ".join(df["App"].head(5))}'
```

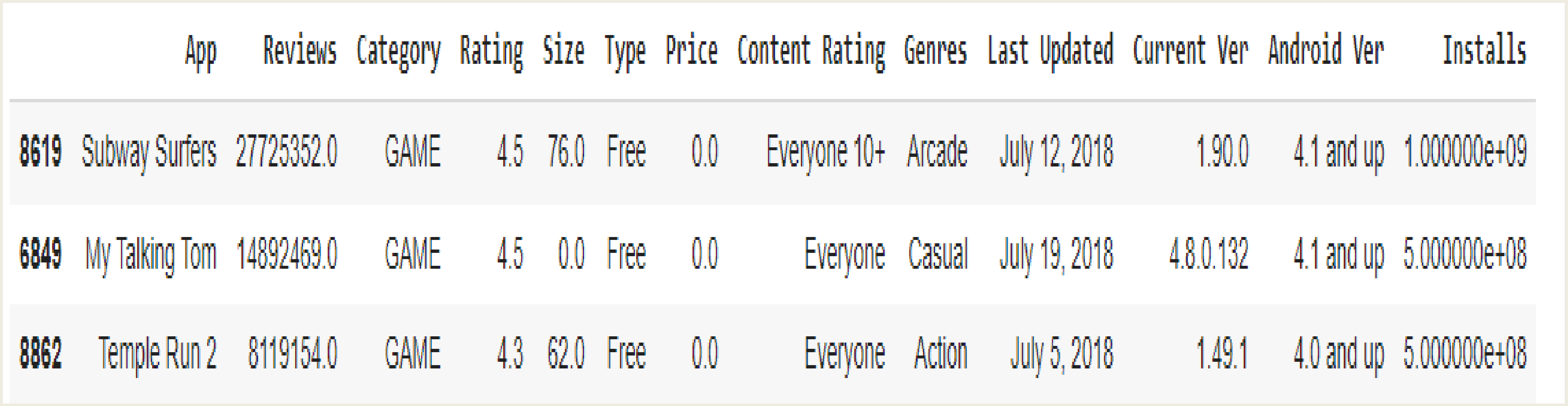
_Output:

The top 5 most expensive apps in the store are: Facebook, WhatsApp Messenger, Instagram, Messenger – Text and Video Chat for Free, Clash of Clans CodeText

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

STEP 35:

```
# 4. What is the top 3 most installed apps in Game category?
# Filter by Game category and sort by Installs in descending order
df_game = df[df['Category'] == 'GAME']
df_game = df_game.sort_values(by='Installs', ascending=False)
# Show top 3 most installed apps in Game category
top_3_installed_games = df_game.head(3)
top_3_installed_games
```



	App	Reviews	Category	Rating	Size	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Installs
8619	Subway Surfers	27725352.0	GAME	4.5	76.0	Free	0.0	Everyone 10+	Arcade	July 12, 2018	1.90.0	4.1 and up	1.000000e+09
6849	My Talking Tom	14892469.0	GAME	4.5	0.0	Free	0.0	Everyone	Casual	July 19, 2018	4.8.0.132	4.1 and up	5.000000e+08
8862	Temple Run 2	8119154.0	GAME	4.3	62.0	Free	0.0	Everyone	Action	July 5, 2018	1.49.1	4.0 and up	5.000000e+08

STEP 36:

```
print(f'The top 3 most expensive apps in the GAME category are: {", ".join(df["App"].head(3))}')
```

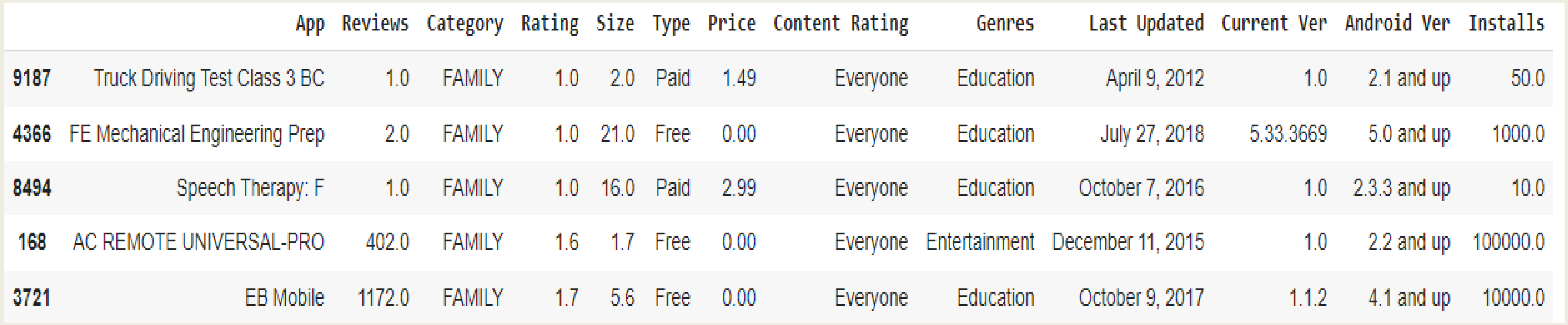
Output:

The top 3 most expensive apps in the GAME category are: Clash of Clans, Subway Surfers, Clash Royale

STEP 37:

Created & Drafted by: https://www.linkedin.com/in/nayan-dholakia/

```
# 5. Which 5 apps from the 'FAMILY' category are having the lowest rating?
df = df[df['Category'] == 'FAMILY']
df = df.sort_values(by=['Rating'], ascending=True)
df.head(5)
```



	App	Reviews	Category	Rating	Size	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Installs
9187	Truck Driving Test Class 3 BC	1.0	FAMILY	1.0	2.0	Paid	1.49	Everyone	Education	April 9, 2012	1.0	2.1 and up	50.0
4366	FE Mechanical Engineering Prep	2.0	FAMILY	1.0	21.0	Free	0.00	Everyone	Education	July 27, 2018	5.33.3669	5.0 and up	1000.0
8494	Speech Therapy: F	1.0	FAMILY	1.0	16.0	Paid	2.99	Everyone	Education	October 7, 2016	1.0	2.3.3 and up	10.0
168	AC REMOTE UNIVERSAL-PRO	402.0	FAMILY	1.6	1.7	Free	0.00	Everyone	Entertainment	December 11, 2015	1.0	2.2 and up	100000.0
3721	EB Mobile	1172.0	FAMILY	1.7	5.6	Free	0.00	Everyone	Education	October 9, 2017	1.1.2	4.1 and up	10000.0

```
print(f'The 5 apps from the FAMILY category having the lowest rating are: {", ".join(df["App"].head(5))}')
```

Output:

The 5 apps from the FAMILY category having the lowest rating are: Truck Driving Test Class 3 BC, FE Mechanical Engineering Prep, Speech Therapy: F, AC REMOTE UNIVERSAL-PRO, EB Mobile