AI

Capstone Project-1

(Exploratory Data Analysis)

Play Store App Review Analysis

colab https://colab.research.google.com/madhavimali

GitHub https://github.com/madhavimali/PlayStore

Team Member

Team DATA DIGGERS

1. SARTHAK ARORA	sarthak1611@gmail.com
2. JAY NANDASANA	nandasanajay@gmail.com
3. MADHAVI MALI	madhvimali1996@gmail.com
4. PRANJALI TETE	pranjalitele@gmail.com
5. ARSHI WANI	arshiwani3@gmail.com

Content

1. Introduction
2. Problem Statement
3. Dataset Description
    a) Play store App Data
    b) User Reviews Data
4. Data Processing Flowchart
5. Import Libraries
6. Data Loading
7. Null values Imputation / Data Cleaning
8. Data Segregation
9. Exploratory & Visualization Analysis
10. Conclusion

Introduction

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

- Taking into account billion of Android users worldwide, mining this data has the potential to reveal user behaviours and trends in the whole global scope. This dataset is obtained from scraping Google Play Store.

Problem Statement

New app maker company is trying to identify various factors to capture android market ,so they are trying to find out the following things from the app store data:

1.Which is the best suitable Android version for new App?
2.Which are the top ‘category’ apps available in the play store?
3.What are the top ‘genre’ apps available in play store?
4.Ratings flow of the Apps
5.Paid and Free App ratio
6.People age group ratio, who are connected with play store
7.Apps vs their updated year ratio
8.Age group and its response against apps
9.Category wise free and paid apps
10.Age wise preferred category

Dataset Description

Two different datasets provided for analysis:

1.Play Store Data.csv

App : Categorical, the app name.
Category : Categorical, category the app belongs to.
Rating : Numerical, range from 0.0 to 5.0,Rating has received from the users.
Reviews : Numerical, the number of reviews that the app received.
Size : Numerical, the size of the app. The suffix M - megabytes, K - kilobytes.
Installs : Numerical, describes the number of installs.
Type : Categorical, a label that indicates whether the app is free or paid.
Price : Numerical, the price value for the paid apps.
Content Rating : Categorical, a categorical rating that indicates the age group for user.
Genre : Categorical, list of genres to which the app belongs.
Last Update : Date Format, the date at which the app was last updated.
Current Version : Version of the app as specified by the developers.
Android Version : The Android OS the app is compatible with.

Dataset Description

2.User Reviews.csv

App : the app name.

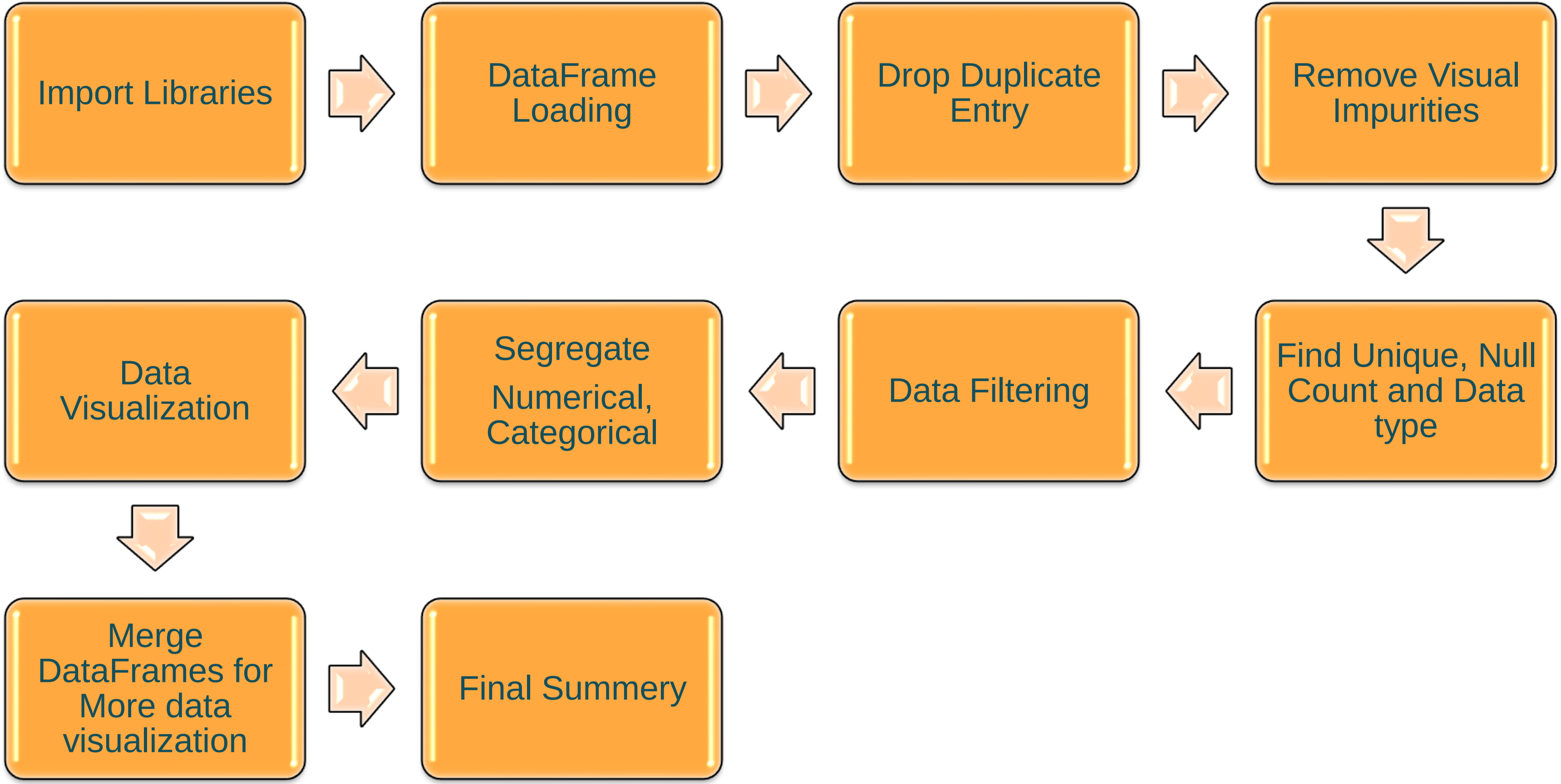
Translated_Review : the review text in English.

Sentiment : the sentiment of the review, positive, neutral, or negative.

Sentiment_Polarity : the sentiment in numerical form, ranging from -1.00 to 1.00.

Sentiment_Subjectivity : a measure of the expression of opinions, evaluations, feelings, and speculations.

Data Processing Flowchart



Import Libraries

DataFrame Loading

Drop Duplicate Entry

Remove Visual Impurities

Find Unique, Null Count and Data type

Data Filtering

Segregate Numerical, Categorical

Data Visualization

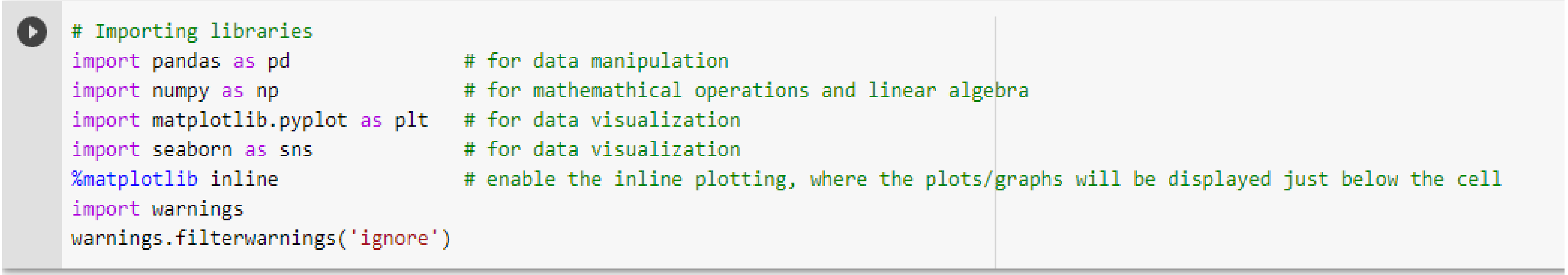
Merge DataFrames for More data visualization

Final Summery

Import Libraries

library is a **collection of related modules**. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer.

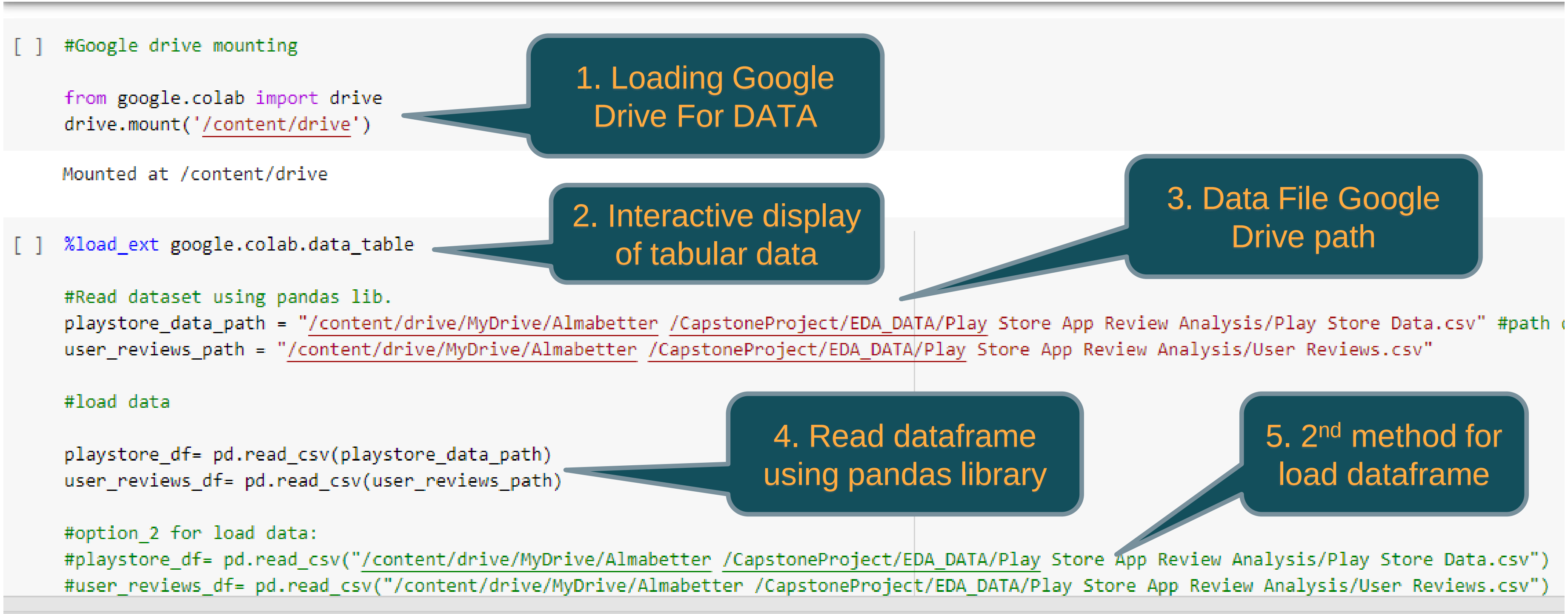
Imporating Require Libraries



```
# Importing libraries
import pandas as pd                # for data manipulation
import numpy as np                 # for mathematical operations and linear algebra
import matplotlib.pyplot as plt    # for data visualization
import seaborn as sns              # for data visualization
%matplotlib inline                 # enable the inline plotting, where the plots/graphs will be displayed just below the cell
import warnings
warnings.filterwarnings('ignore')
```

DataFrame Loading…….

Import Playstore Data and User Review Data



```
[ ]  #Google drive mounting

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

     Mounted at /content/drive

[ ]  %load_ext google.colab.data_table

     #Read dataset using pandas lib.
     playstore_data_path = "/content/drive/MyDrive/Almabetter /CapstoneProject/EDA_DATA/Play Store App Review Analysis/Play Store Data.csv" #path
     user_reviews_path = "/content/drive/MyDrive/Almabetter /CapstoneProject/EDA_DATA/Play Store App Review Analysis/User Reviews.csv"

     #load data

     playstore_df= pd.read_csv(playstore_data_path)
     user_reviews_df= pd.read_csv(user_reviews_path)

     #option_2 for load data:
     #playstore_df= pd.read_csv("/content/drive/MyDrive/Almabetter /CapstoneProject/EDA_DATA/Play Store App Review Analysis/Play Store Data.csv")
     #user_reviews_df= pd.read_csv("/content/drive/MyDrive/Almabetter /CapstoneProject/EDA_DATA/Play Store App Review Analysis/User Reviews.csv")
```

1. Loading Google Drive For DATA

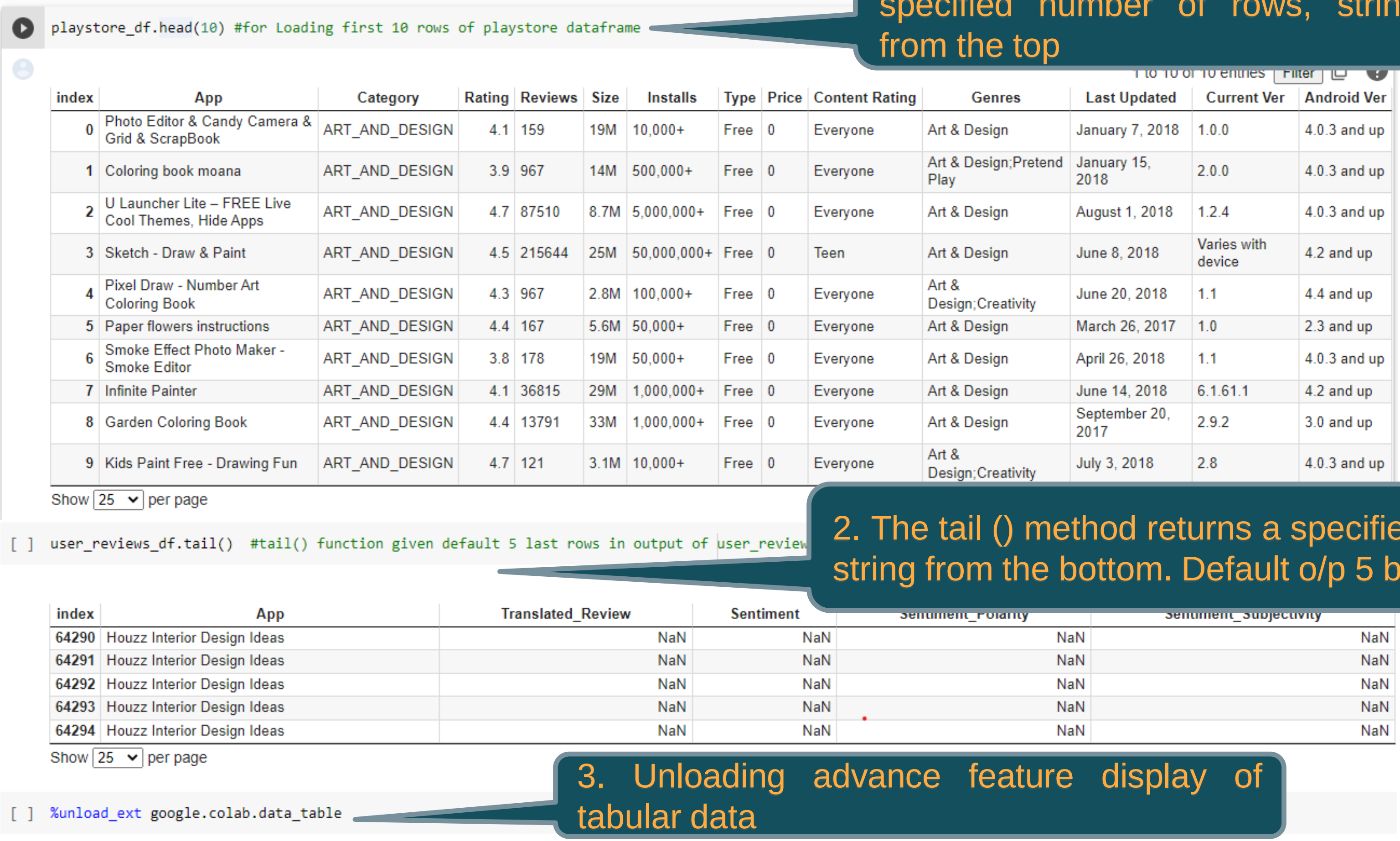
2. Interactive display of tabular data

3. Data File Google Drive path

4. Read dataframe using pandas library

5. 2nd method for load dataframe

DataFrame Loading…….



```
playstore_df.head(10) #for Loading first 10 rows of playstore dataframe
```

1 to 10 of 10 entries Filter

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	1.0	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

```
[ ] user_reviews_df.tail()  #tail() function given default 5 last rows in output of user_review
```

index	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
64290	Houzz Interior Design Ideas	NaN	NaN	NaN	NaN
64291	Houzz Interior Design Ideas	NaN	NaN	NaN	NaN
64292	Houzz Interior Design Ideas	NaN	NaN	NaN	NaN
64293	Houzz Interior Design Ideas	NaN	NaN	NaN	NaN
64294	Houzz Interior Design Ideas	NaN	NaN	NaN	NaN

Show 25 per page

```
[ ] %unload_ext google.colab.data_table
```

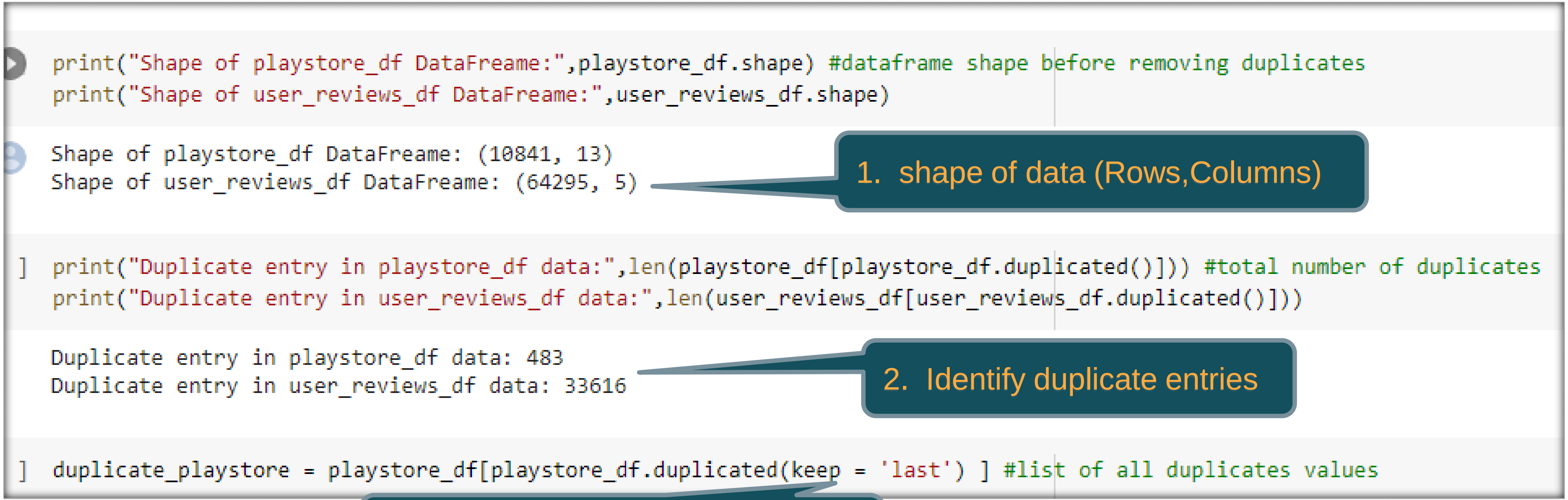
1. The head () method returns a specified number of rows, string from the top

2. The tail () method returns a specified number of rows, string from the bottom. Default o/p 5 bottom rows

3. Unloading advance feature display of tabular data

Drop Duplicate Entry

- Duplicate Enter data is repeated data in dataframe,
- Either multiple columns have same data or multiple rows have same data.
- Repeated Entry add some number of data in final data and it miscalculates the true outcome.
- **df.duplicated()** method is used to identify duplicates from dataframe



```
print("Shape of playstore_df DataFreame:",playstore_df.shape) #dataframe shape before removing duplicates
print("Shape of user_reviews_df DataFreame:",user_reviews_df.shape)
```

```
Shape of playstore_df DataFreame: (10841, 13)
Shape of user_reviews_df DataFreame: (64295, 5)
```

1. shape of data (Rows,Columns)

```
print("Duplicate entry in playstore_df data:",len(playstore_df[playstore_df.duplicated()])) #total number of duplicates
print("Duplicate entry in user_reviews_df data:",len(user_reviews_df[user_reviews_df.duplicated()]))
```

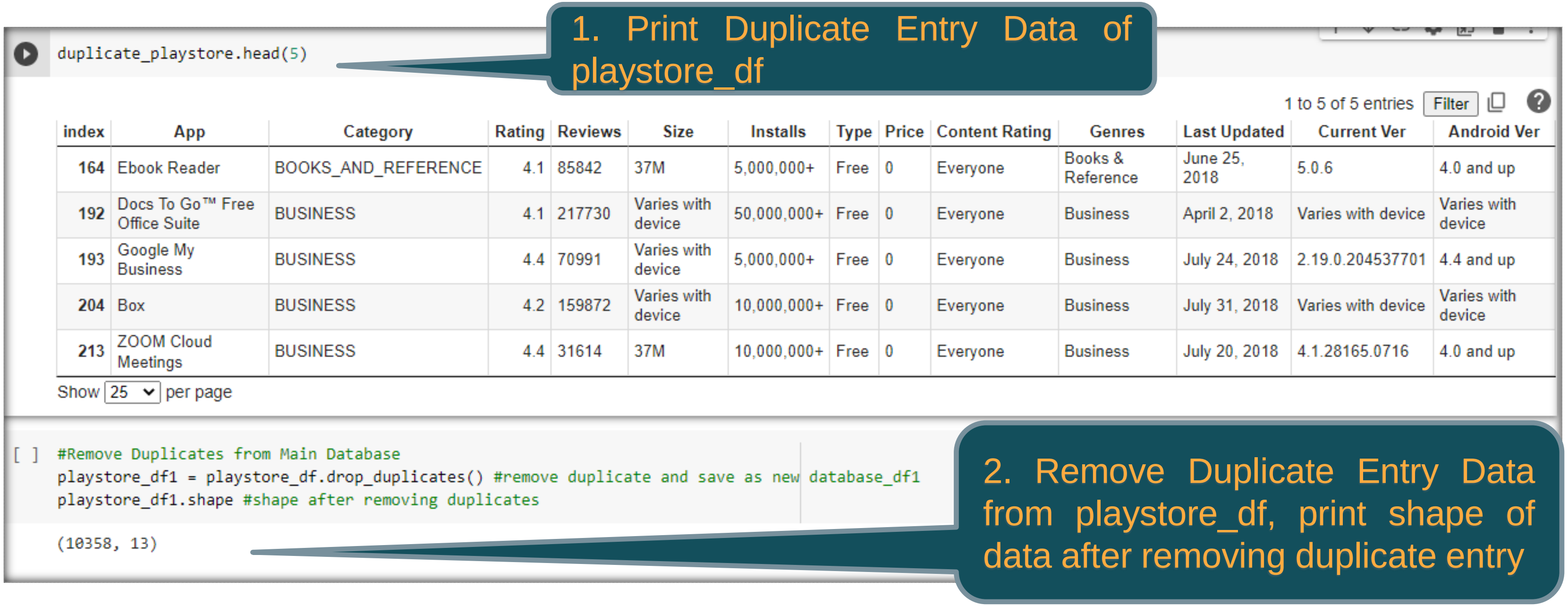
```
Duplicate entry in playstore_df data: 483
Duplicate entry in user_reviews_df data: 33616
```

2. Identify duplicate entries

```
duplicate_playstore = playstore_df[playstore_df.duplicated(keep = 'last') ] #list of all duplicates values
```

3. keep the last instance of a duplicate row in dataframe

Drop Duplicate Entry



```
duplicate_playstore.head(5)
```

1 to 5 of 5 entries Filter

index	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
164	Ebook Reader	BOOKS_AND_REFERENCE	4.1	85842	37M	5,000,000+	Free	0	Everyone	Books & Reference	June 25, 2018	5.0.6	4.0 and up
192	Docs To Go™ Free Office Suite	BUSINESS	4.1	217730	Varies with device	50,000,000+	Free	0	Everyone	Business	April 2, 2018	Varies with device	Varies with device
193	Google My Business	BUSINESS	4.4	70991	Varies with device	5,000,000+	Free	0	Everyone	Business	July 24, 2018	2.19.0.204537701	4.4 and up
204	Box	BUSINESS	4.2	159872	Varies with device	10,000,000+	Free	0	Everyone	Business	July 31, 2018	Varies with device	Varies with device
213	ZOOM Cloud Meetings	BUSINESS	4.4	31614	37M	10,000,000+	Free	0	Everyone	Business	July 20, 2018	4.1.28165.0716	4.0 and up

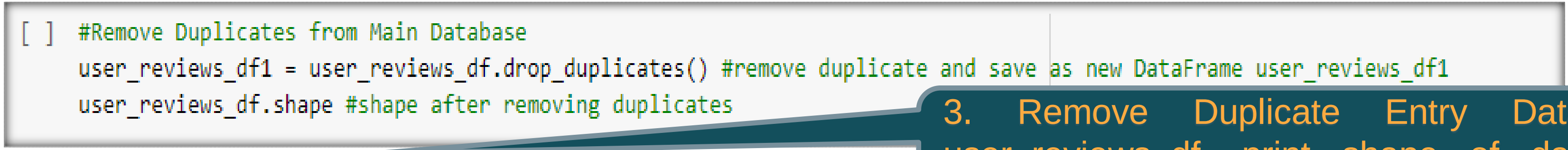
Show 25 per page

1. Print Duplicate Entry Data of playstore_df

```
[ ] #Remove Duplicates from Main Database
    playstore_df1 = playstore_df.drop_duplicates() #remove duplicate and save as new database_df1
    playstore_df1.shape #shape after removing duplicates
```

```
(10358, 13)
```

2. Remove Duplicate Entry Data from playstore_df, print shape of data after removing duplicate entry



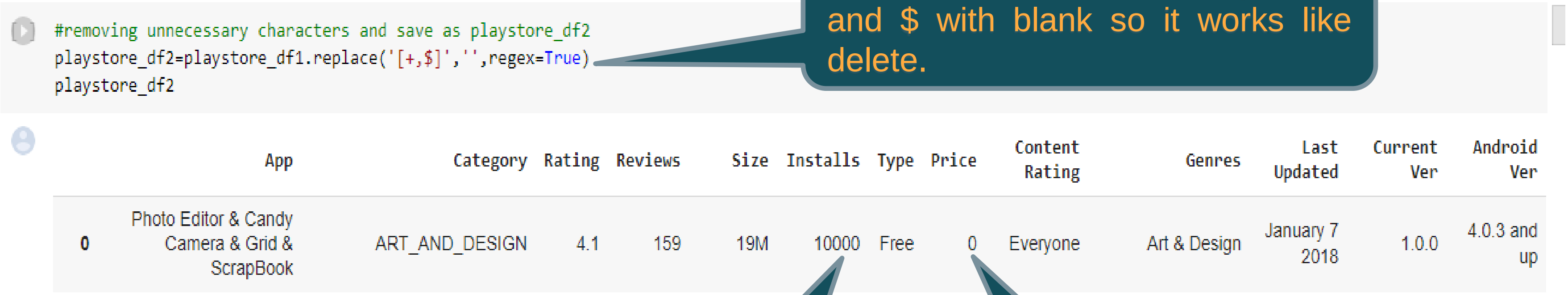
```
[ ] #Remove Duplicates from Main Database
    user_reviews_df1 = user_reviews_df.drop_duplicates() #remove duplicate and save as new DataFrame user_reviews_df1
    user_reviews_df.shape #shape after removing duplicates
```

3. Remove Duplicate Entry Data from user_reviews_df, print shape of data after removing duplicate entry

Remove Visual Impurities

Visual Impurities is like eye sight impurities in dataframe, without additional deep finding visible impurities

- Like: Price Column have “$” sign and Installation column have “+” sign



```
#removing unnecessary characters and save as playstore_df2
playstore_df2=playstore_df1.replace('[+,$]','',regex=True)
playstore_df2
```

	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	10000	Free	0	Everyone	Art & Design	January 7 2018	1.0.0	4.0.3 and up

1. Check dataframe and replace + and $ with blank so it works like delete.

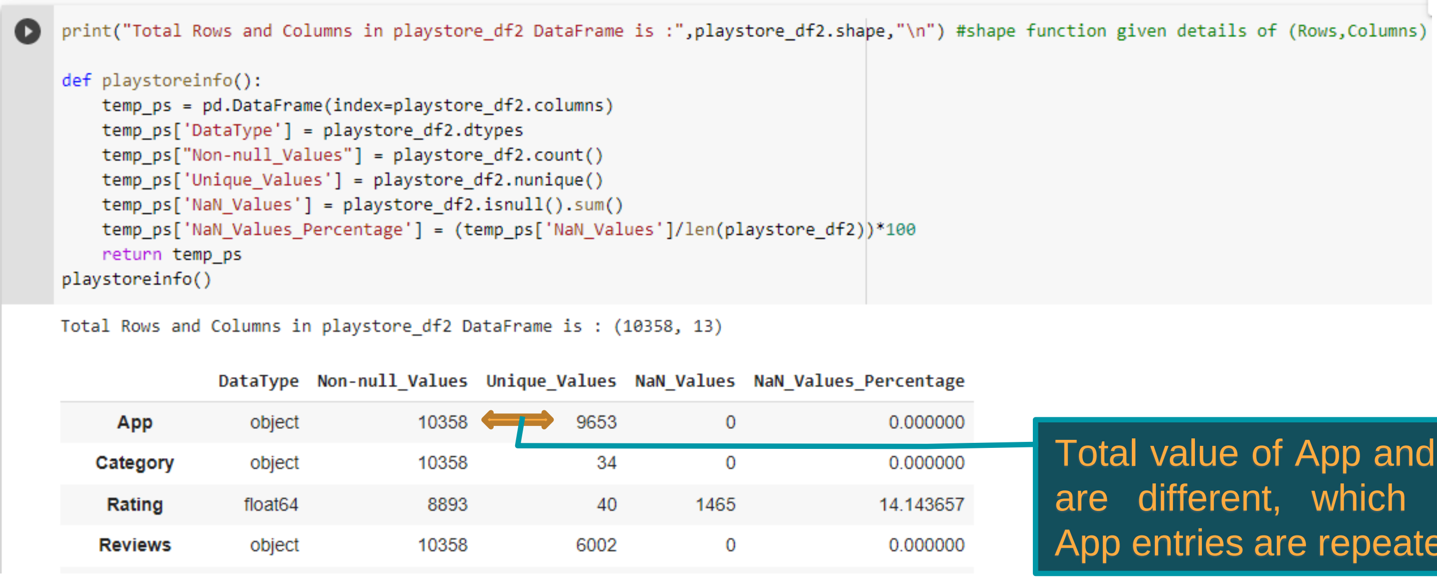
Data after filter +

Data after filter $

Find Unique, Null Count and Data type

For more cleaning of data we identified Unique data – Null Counts and Data type of each column

Unique Data	- Unique value of particular column
Null Count	- NULL means that there is no value
Nan Count	- NaN mean that there is some value, although one that is perhaps not usable
Data type	- Type of data like int, float64, object



```
print("Total Rows and Columns in playstore_df2 DataFrame is :",playstore_df2.shape,"\n") #shape function given details of (Rows,Columns)

def playstoreinfo():
    temp_ps = pd.DataFrame(index=playstore_df2.columns)
    temp_ps['DataType'] = playstore_df2.dtypes
    temp_ps["Non-null_Values"] = playstore_df2.count()
    temp_ps['Unique_Values'] = playstore_df2.nunique()
    temp_ps['NaN_Values'] = playstore_df2.isnull().sum()
    temp_ps['NaN_Values_Percentage'] = (temp_ps['NaN_Values']/len(playstore_df2))*100
    return temp_ps
playstoreinfo()
```

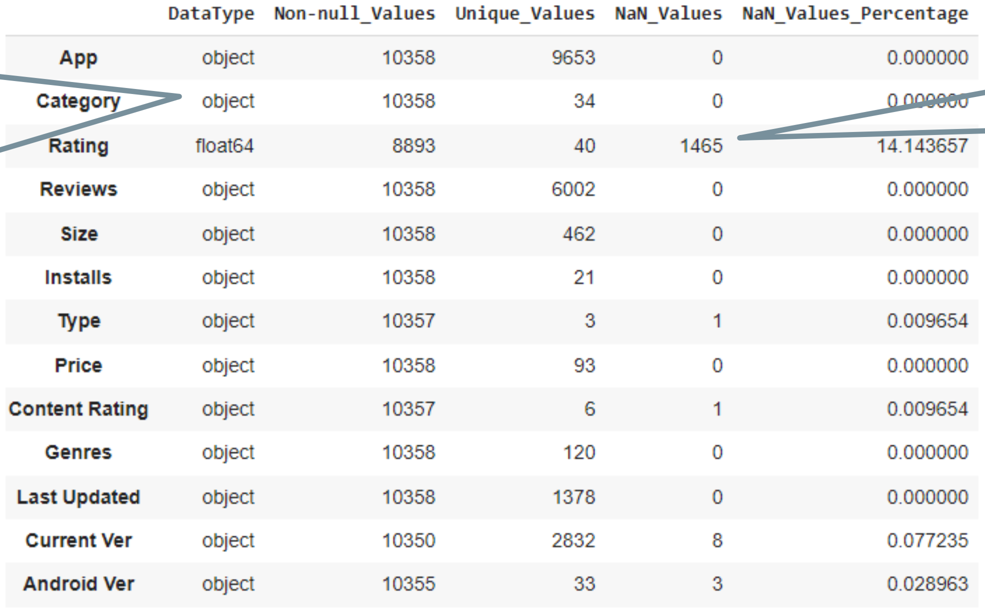
```
Total Rows and Columns in playstore_df2 DataFrame is : (10358, 13)
```

	DataType	Non-null_Values	Unique_Values	NaN_Values	NaN_Values_Percentage
App	object	10358	9653	0	0.000000
Category	object	10358	34	0	0.000000
Rating	float64	8893	40	1465	14.143657
Reviews	object	10358	6002	0	0.000000

Total value of App and Unique value are different, which means some App entries are repeated

Information From Data Digging

Datatype are not correct for some columns like price , size, Installs and Last Update



	DataType	Non-null_Values	Unique_Values	NaN_Values	NaN_Values_Percentage
App	object	10358	9653	0	0.000000
Category	object	10358	34	0	0.000000
Rating	float64	8893	40	1465	14.143657
Reviews	object	10358	6002	0	0.000000
Size	object	10358	462	0	0.000000
Installs	object	10358	21	0	0.000000
Type	object	10357	3	1	0.009654
Price	object	10358	93	0	0.000000
Content Rating	object	10357	6	1	0.009654
Genres	object	10358	120	0	0.000000
Last Updated	object	10358	1378	0	0.000000
Current Ver	object	10350	2832	8	0.077235
Android Ver	object	10355	33	3	0.028963

Some columns have null values that need to be filled or filtered

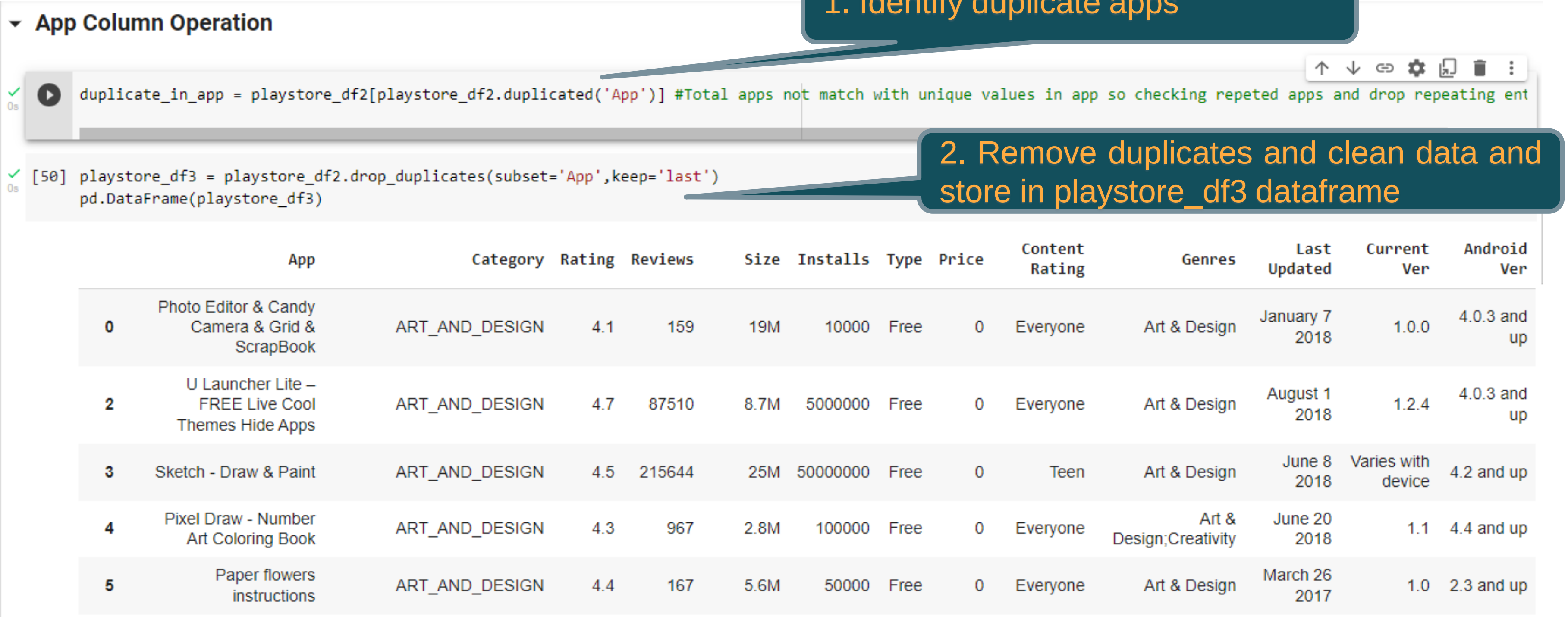
Total count of each rows are not same so database are missing

Data Filtering Column by Column

All Columns have some mistakes and unreliable things that need to be filtered

No	DataType	Mistakes and Unreliable things on specific Column
1	**App**	Repeated Entry in app column indicated some repeated apps are there
2	**Category**	1.9 entry in category is outliner, checked data and shifted the row
3	**Rating**	Null values present, filled it with mean() or median()
4	**Reviews**	Data type mistake, it’s a numerical data type
5	**Size**	Converted the size into one single unit
6	**Installs**	After removing +, corrected the data type to numerical
7	**Type**	1 null value present, filled it with proper data (After Crosscheck with play store fill with "Free")
8	**Price**	After removing $ sign, need to correct data type to numerical
9	**Content Rating**	Did not require any operation
10	**Genres**	Did not require any operation
11	**Last Updated**	Data type needed to replace to Date Format : datetime64[ns]
12	**Current Ver**	Did not require any operation
13	**Android Ver**	Did not require any operation

App Column Operation



1. Identify duplicate apps

2. Remove duplicates and clean data and store in playstore_df3 dataframe

App Column Operation

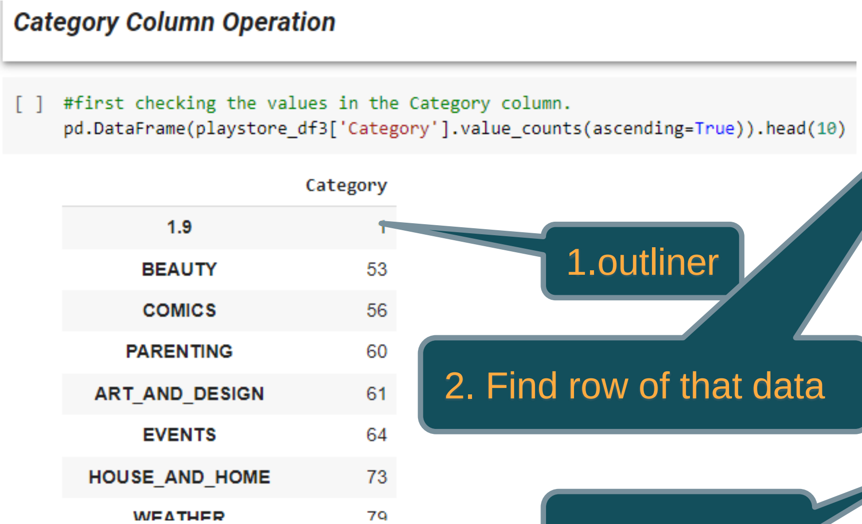
```
duplicate_in_app = playstore_df2[playstore_df2.duplicated('App')] #Total apps not match with unique values in app so checking repeted apps and drop repeating ent
```

```
[50] playstore_df3 = playstore_df2.drop_duplicates(subset='App',keep='last')
     pd.DataFrame(playstore_df3)
```

	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	10000	Free	0	Everyone	Art & Design	January 7 2018	1.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes Hide Apps	ART_AND_DESIGN	4.7	87510	8.7M	5000000	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	50000000	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	100000	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	50000	Free	0	Everyone	Art & Design	March 26 2017	1.0	2.3 and up

Category Column Operation

1.

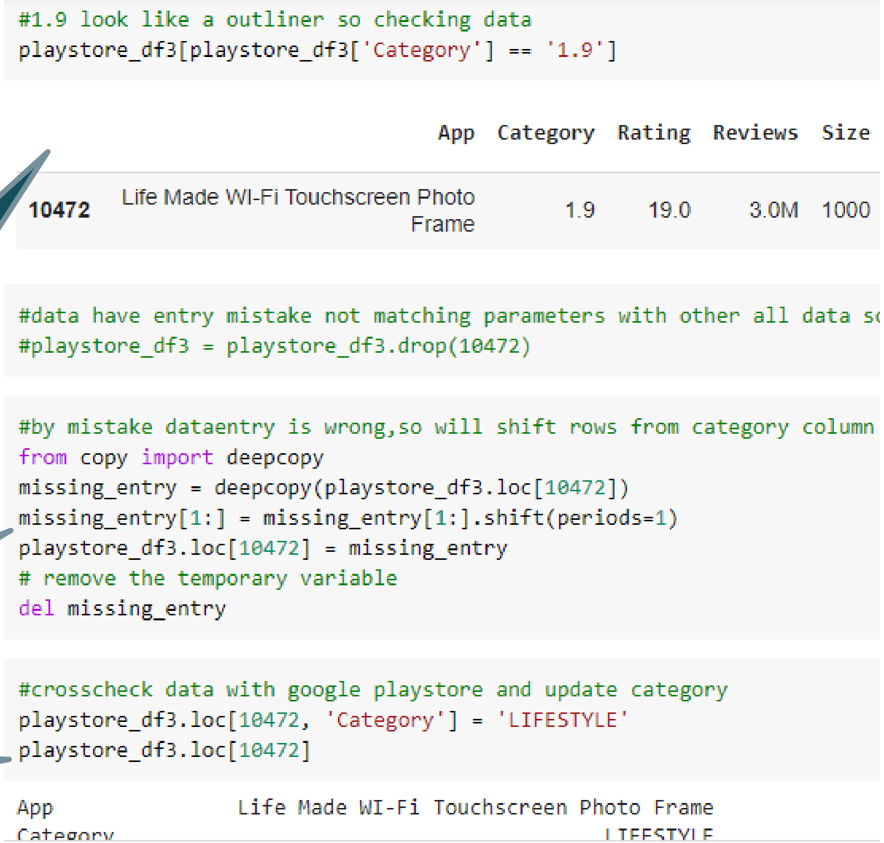


Category Column Operation

```
[ ] #first checking the values in the Category column.
    pd.DataFrame(playstore_df3['Category'].value_counts(ascending=True)).head(10)
```

	Category
1.9	1
BEAUTY	53
COMICS	56
PARENTING	60
ART_AND_DESIGN	61
EVENTS	64
HOUSE_AND_HOME	73
WEATHER	79

2.



```
#1.9 look like a outliner so checking data
playstore_df3[playstore_df3['Category'] == '1.9']
```

	App	Category	Rating	Reviews	Size
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1000

```
#data have entry mistake not matching parameters with other all data sc
#playstore_df3 = playstore_df3.drop(10472)
```

```
#by mistake dataentry is wrong,so will shift rows from category column
from copy import deepcopy
missing_entry = deepcopy(playstore_df3.loc[10472])
missing_entry[1:] = missing_entry[1:].shift(periods=1)
playstore_df3.loc[10472] = missing_entry
# remove the temporary variable
del missing_entry
```

```
#crosscheck data with google playstore and update category
playstore_df3.loc[10472, 'Category'] = 'LIFESTYLE'
playstore_df3.loc[10472]
```

```
App                Life Made WI-Fi Touchscreen Photo Frame
Category                                         LIFESTYLE
```

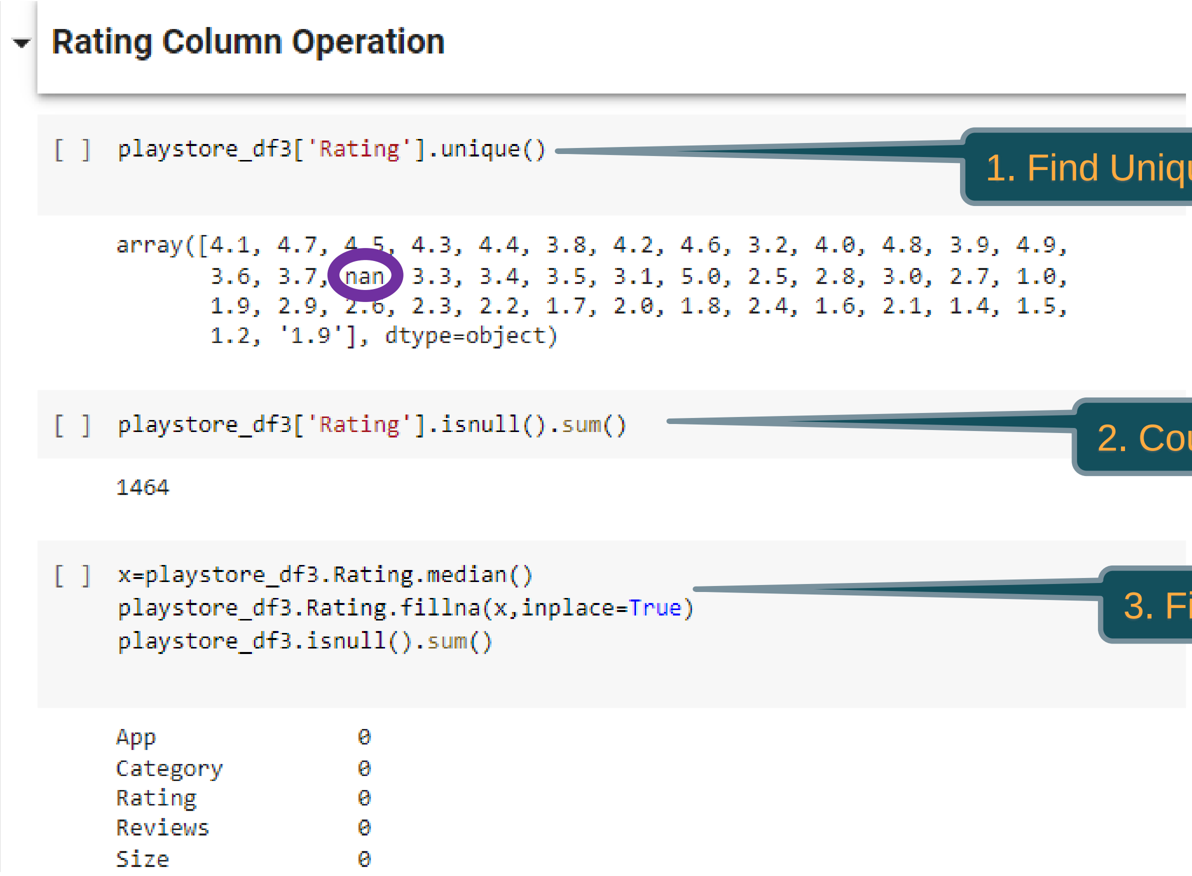
1.outliner

2. Find row of that data

3. Shift data

4. Replace with “LIFESTYLE” (check with playstore)

Rating Column Operation



Rating Column Operation

```
[ ]  playstore_df3['Rating'].unique()
```

```
array([4.1, 4.7, 4.5, 4.3, 4.4, 3.8, 4.2, 4.6, 3.2, 4.0, 4.8, 3.9, 4.9,
       3.6, 3.7, nan, 3.3, 3.4, 3.5, 3.1, 5.0, 2.5, 2.8, 3.0, 2.7, 1.0,
       1.9, 2.9, 2.6, 2.3, 2.2, 1.7, 2.0, 1.8, 2.4, 1.6, 2.1, 1.4, 1.5,
       1.2, '1.9'], dtype=object)
```

```
[ ]  playstore_df3['Rating'].isnull().sum()
```

```
1464
```

```
[ ]  x=playstore_df3.Rating.median()
     playstore_df3.Rating.fillna(x,inplace=True)
     playstore_df3.isnull().sum()
```

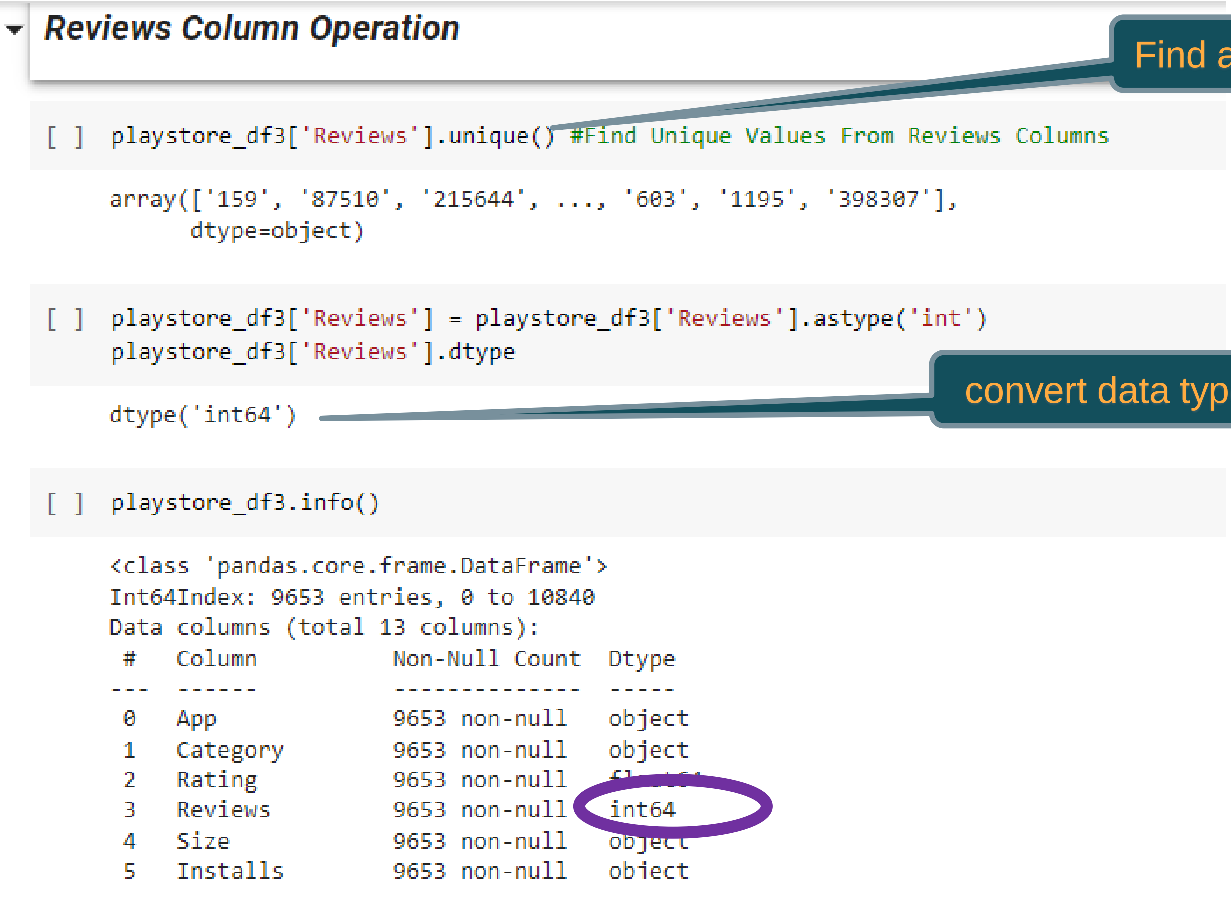
```
App          0
Category     0
Rating       0
Reviews      0
Size         0
```

1. Find Unique value

2. Count Null value

3. Fill with Median

Review Column Operation



Reviews Column Operation

```
[ ]  playstore_df3['Reviews'].unique() #Find Unique Values From Reviews Columns
```

```
array(['159', '87510', '215644', ..., '603', '1195', '398307'],
      dtype=object)
```

```
[ ]  playstore_df3['Reviews'] = playstore_df3['Reviews'].astype('int')
     playstore_df3['Reviews'].dtype
```

```
dtype('int64')
```

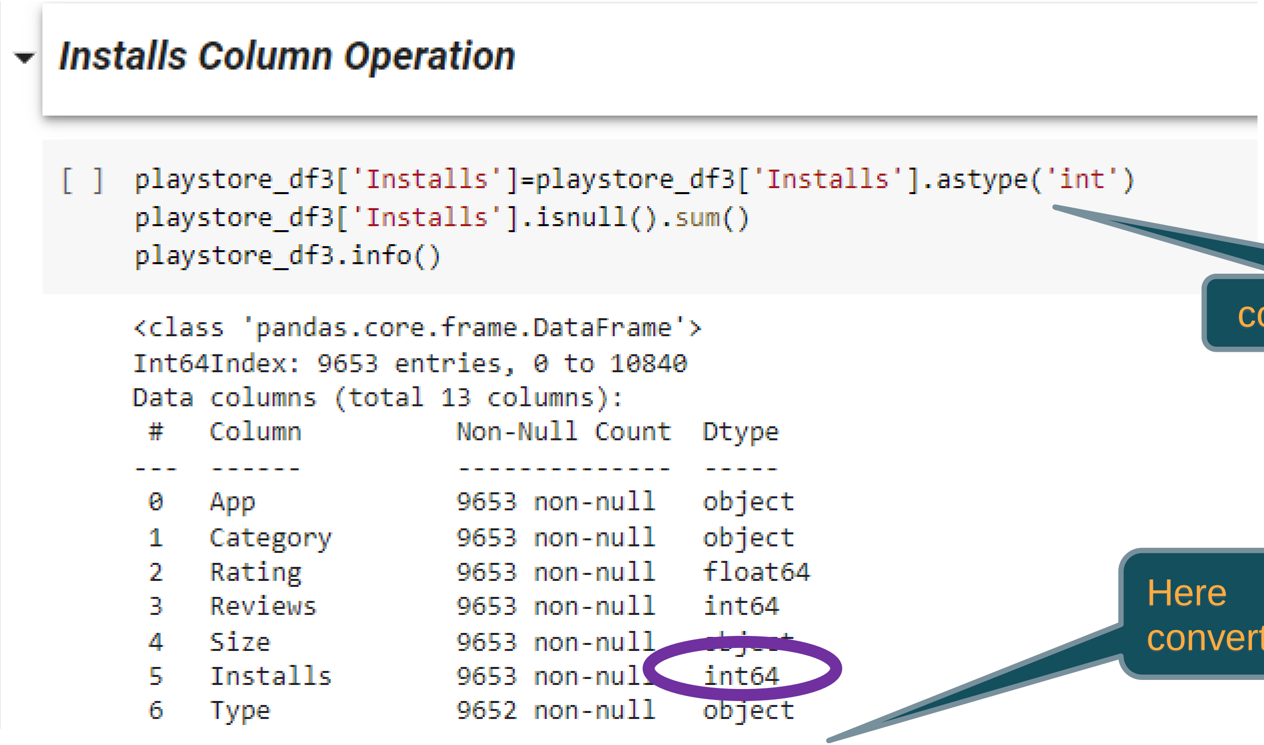
```
[ ]  playstore_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9653 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             9653 non-null   object
 1   Category        9653 non-null   object
 2   Rating          9653 non-null   float64
 3   Reviews         9653 non-null   int64
 4   Size            9653 non-null   object
 5   Installs        9653 non-null   object
```

Find all the unique values.

convert data type to “int”

Installs Column Operation



Installs Column Operation

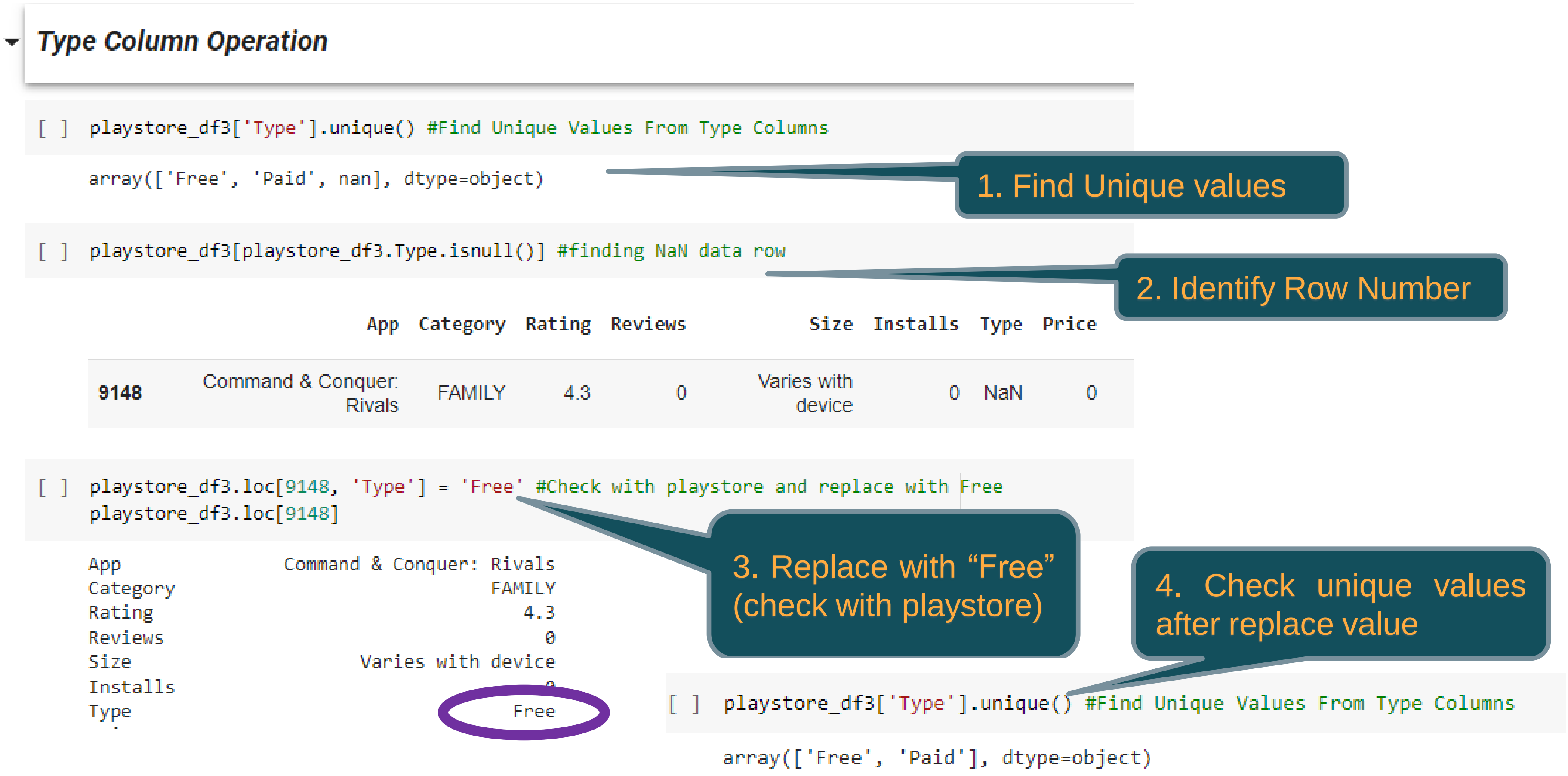
```
[ ]  playstore_df3['Installs']=playstore_df3['Installs'].astype('int')
     playstore_df3['Installs'].isnull().sum()
     playstore_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9653 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             9653 non-null   object
 1   Category        9653 non-null   object
 2   Rating          9653 non-null   float64
 3   Reviews         9653 non-null   int64
 4   Size            9653 non-null   object
 5   Installs        9653 non-null   int64
 6   Type            9652 non-null   object
```

convert data type to “int”

Here we can see it converted into “int” data type

Type Column Operation



Type Column Operation

```
[ ]  playstore_df3['Type'].unique() #Find Unique Values From Type Columns

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

1. Find Unique values

```
[ ]  playstore_df3[playstore_df3.Type.isnull()] #finding NaN data row
```

2. Identify Row Number

	App	Category	Rating	Reviews	Size	Installs	Type	Price
9148	Command & Conquer: Rivals	FAMILY	4.3	0	Varies with device	0	NaN	0

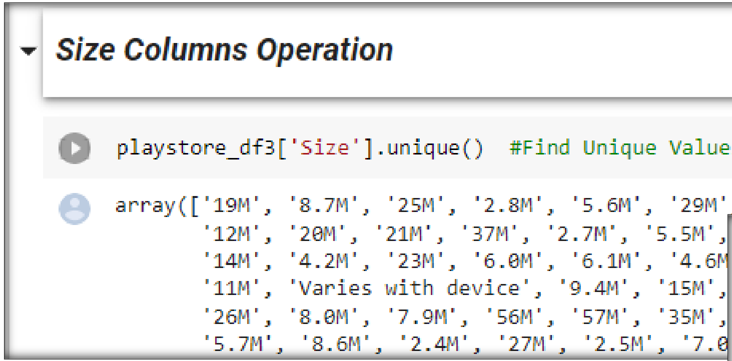
```
[ ]  playstore_df3.loc[9148, 'Type'] = 'Free' #Check with playstore and replace with Free
     playstore_df3.loc[9148]

     App                   Command & Conquer: Rivals
     Category                                 FAMILY
     Rating                                      4.3
     Reviews                                       0
     Size                         Varies with device
     Installs                                      0
     Type                                       Free
```

3. Replace with “Free” (check with playstore)

4. Check unique values after replace value

```
[ ]  playstore_df3['Type'].unique() #Find Unique Values From Type Columns

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

Size Column Operation

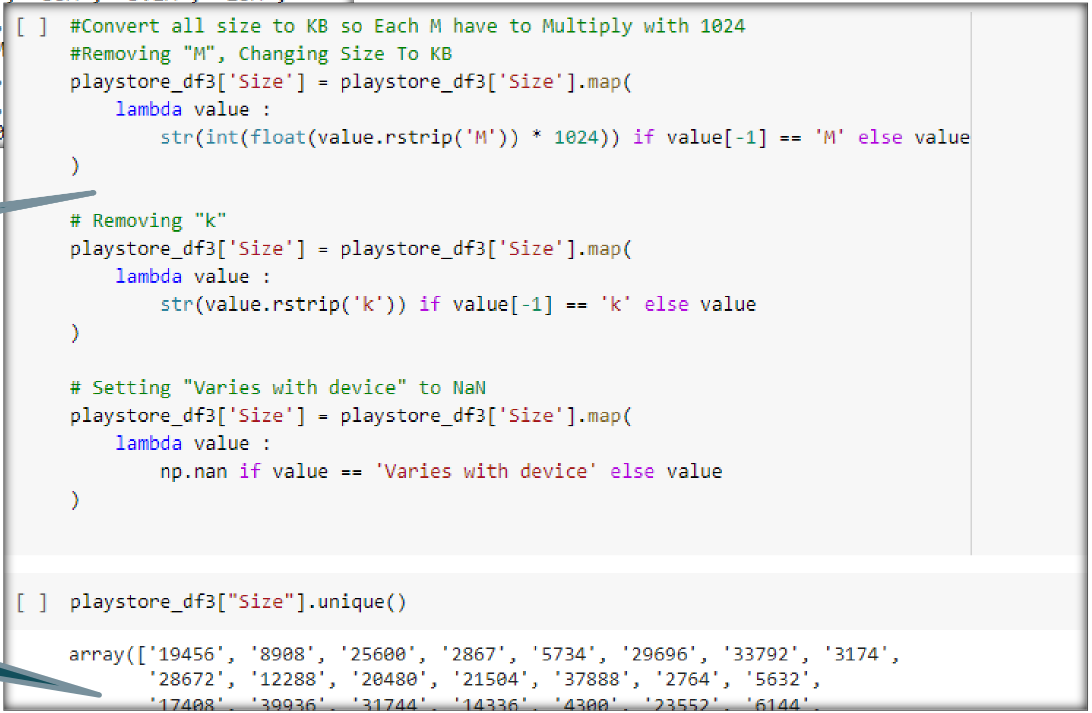


Size Columns Operation

```
playstore_df3['Size'].unique()  #Find Unique Values From Size Columns
```

```
array(['19M', '8.7M', '25M', '2.8M', '5.6M', '29M', '33M', '3.1M', '28M',
       '12M', '20M', '21M', '37M', '2.7M', '5.5M',
       '14M', '4.2M', '23M', '6.0M', '6.1M', '4.6M
       '11M', 'Varies with device', '9.4M', '15M',
       '26M', '8.0M', '7.9M', '56M', '57M', '35M',
       '5.7M', '8.6M', '2.4M', '27M', '2.5M', '7.0
```

1. Value contain ‘M’, ’K’, Varies with device



```
[ ]  #Convert all size to KB so Each M have to Multiply with 1024
     #Removing "M", Changing Size To KB
     playstore_df3['Size'] = playstore_df3['Size'].map(
         lambda value :
             str(int(float(value.rstrip('M')) * 1024)) if value[-1] == 'M' else value
     )

     # Removing "k"
     playstore_df3['Size'] = playstore_df3['Size'].map(
         lambda value :
             str(value.rstrip('k')) if value[-1] == 'k' else value
     )

     # Setting "Varies with device" to NaN
     playstore_df3['Size'] = playstore_df3['Size'].map(
         lambda value :
             np.nan if value == 'Varies with device' else value
     )
```

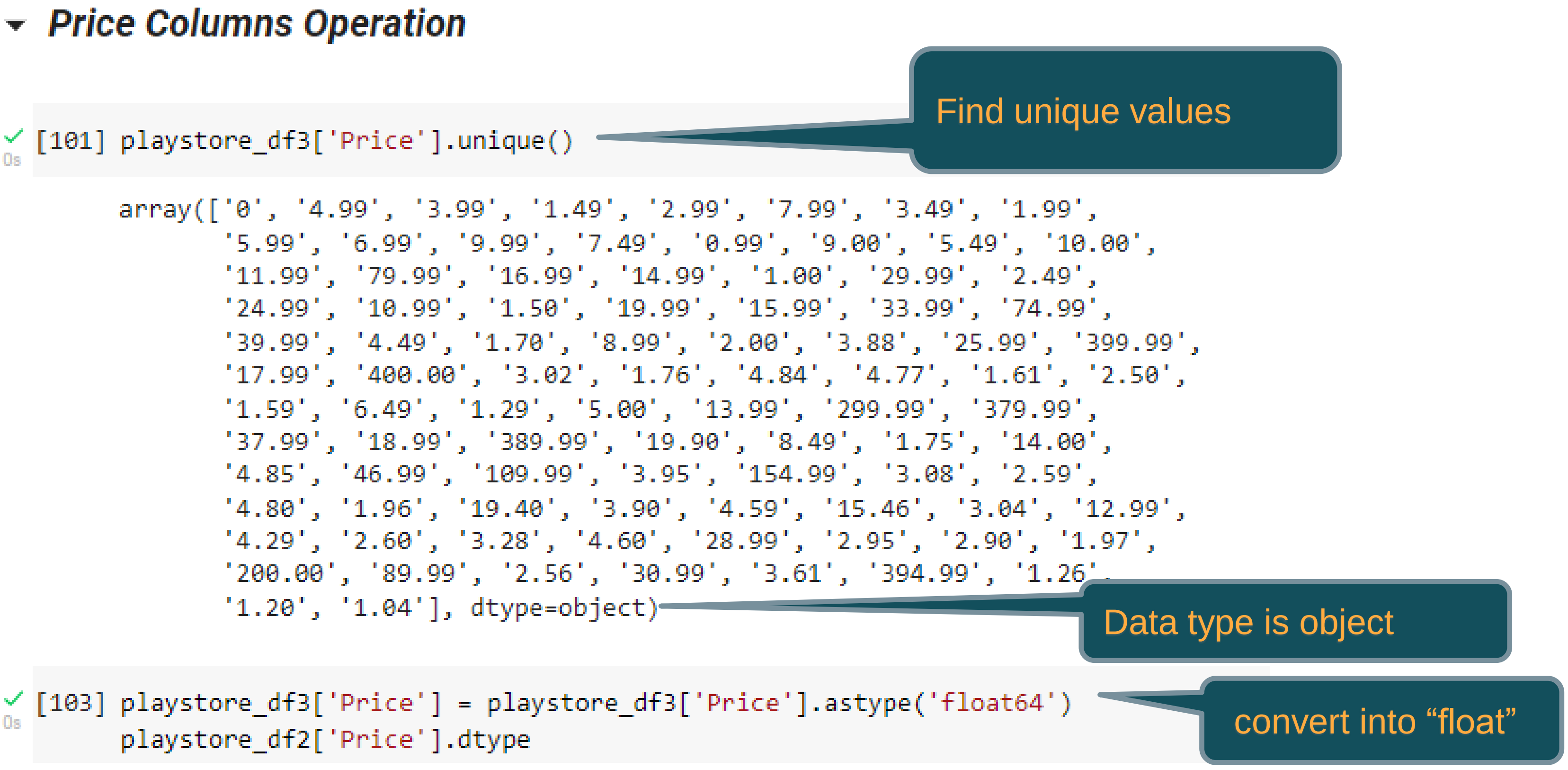
```
[ ]  playstore_df3["Size"].unique()
```

```
     array(['19456', '8908', '25600', '2867', '5734', '29696', '33792', '3174',
            '28672', '12288', '20480', '21504', '37888', '2764', '5632',
            '17408', '39936', '31744', '14336', '4300', '23552', '6144',
```

2. Convert MB to KB and Remove K & M

3. Filter Data output

Price Column Operation



Price Columns Operation

```
[101] playstore_df3['Price'].unique()
```

Find unique values

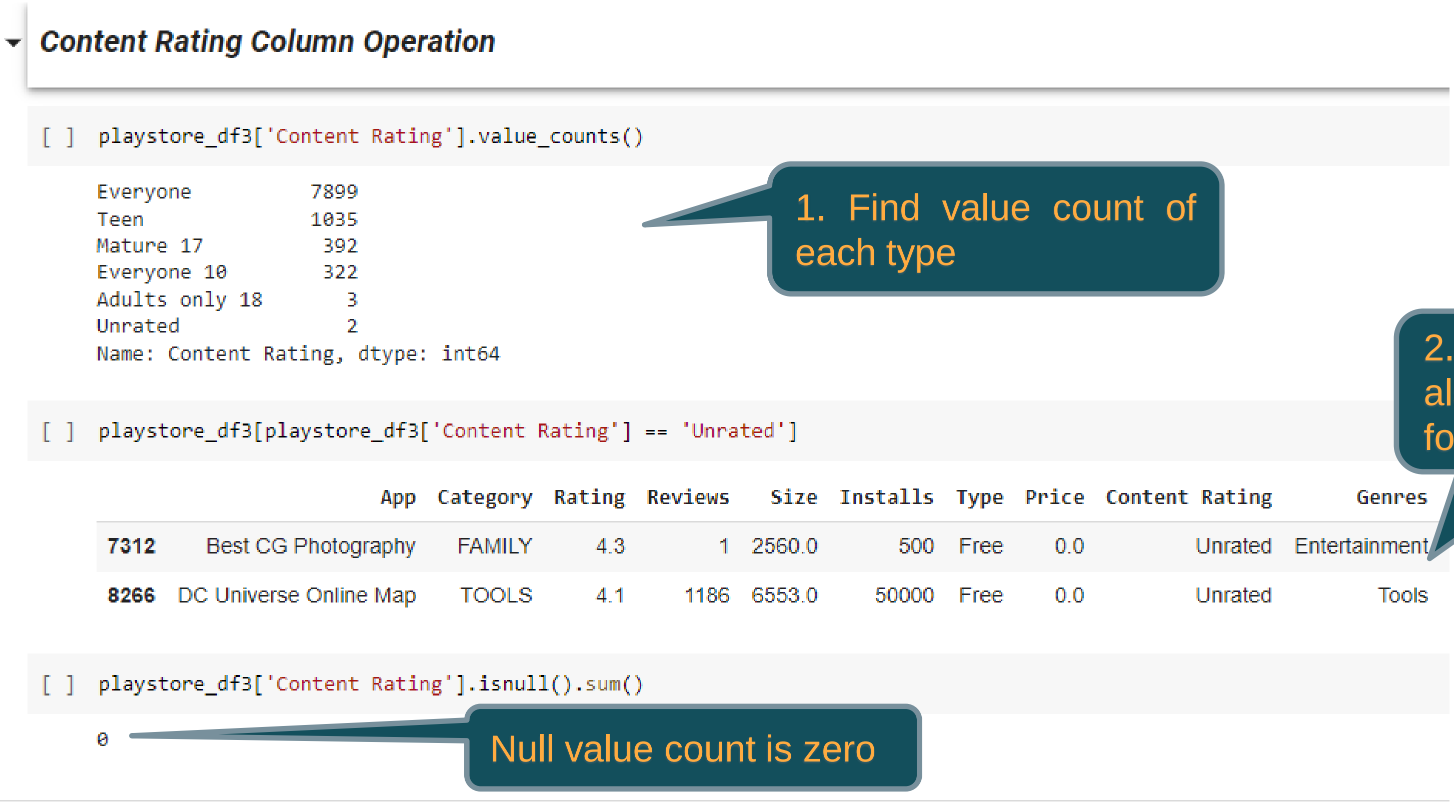
```
array(['0', '4.99', '3.99', '1.49', '2.99', '7.99', '3.49', '1.99',
       '5.99', '6.99', '9.99', '7.49', '0.99', '9.00', '5.49', '10.00',
       '11.99', '79.99', '16.99', '14.99', '1.00', '29.99', '2.49',
       '24.99', '10.99', '1.50', '19.99', '15.99', '33.99', '74.99',
       '39.99', '4.49', '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', '3.95', '154.99', '3.08', '2.59',
       '4.80', '1.96', '19.40', '3.90', '4.59', '15.46', '3.04', '12.99',
       '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',
       '1.20', '1.04'], dtype=object)
```

Data type is object

```
[103] playstore_df3['Price'] = playstore_df3['Price'].astype('float64')
      playstore_df2['Price'].dtype
```

convert into “float”

Content Rating Column Operation



Content Rating Column Operation

```
[ ]  playstore_df3['Content Rating'].value_counts()
```

```
Everyone           7899
Teen               1035
Mature 17           392
Everyone 10         322
Adults only 18        3
Unrated               2
Name: Content Rating, dtype: int64
```

1. Find value count of each type

```
[ ]  playstore_df3[playstore_df3['Content Rating'] == 'Unrated']
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
7312	Best CG Photography	FAMILY	4.3	1	2560.0	500	Free	0.0	Unrated	Entertainment
8266	DC Universe Online Map	TOOLS	4.1	1186	6553.0	50000	Free	0.0	Unrated	Tools

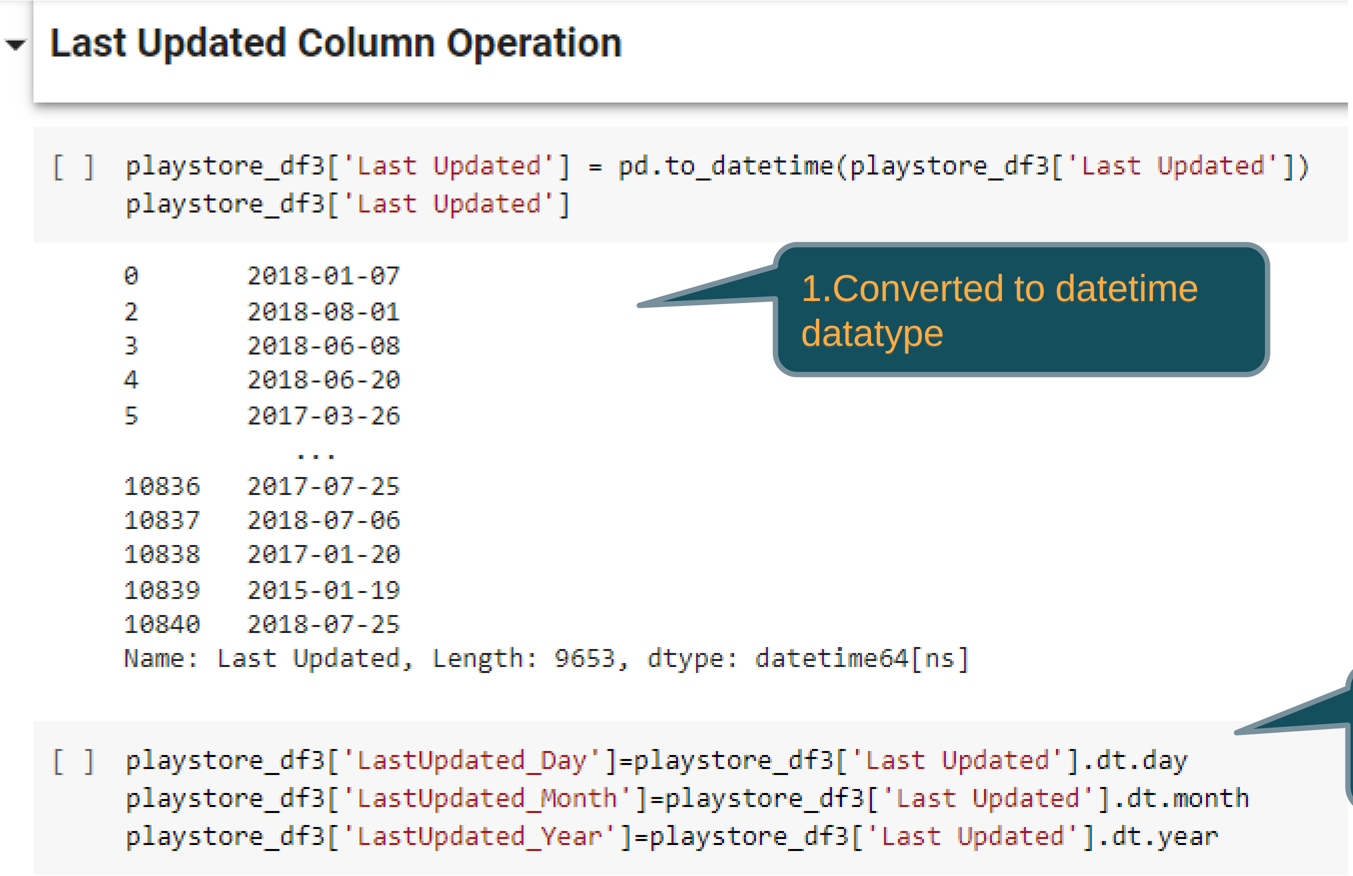
2. Unrated apps also considerable for Everyone

```
[ ]  playstore_df3['Content Rating'].isnull().sum()
```

```
0
```

Null value count is zero

Last Update Column Operation



Last Updated Column Operation

```
[ ]  playstore_df3['Last Updated'] = pd.to_datetime(playstore_df3['Last Updated'])
     playstore_df3['Last Updated']
```

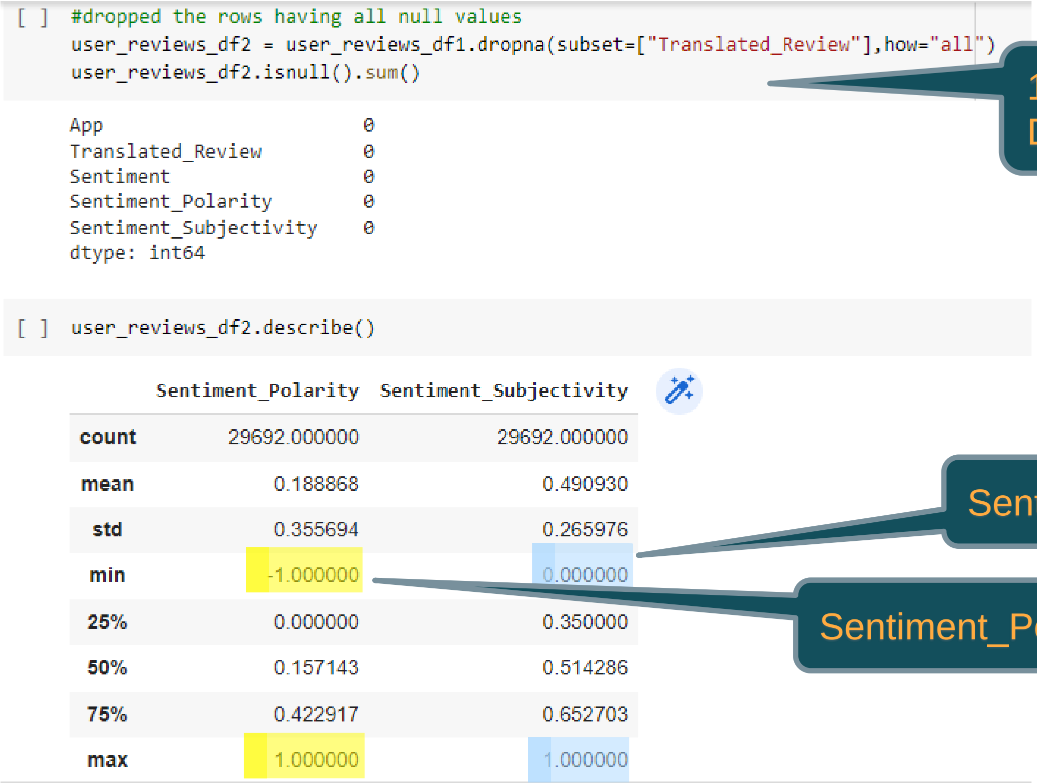
```
0        2018-01-07
2        2018-08-01
3        2018-06-08
4        2018-06-20
5        2017-03-26
            ...
10836    2017-07-25
10837    2018-07-06
10838    2017-01-20
10839    2015-01-19
10840    2018-07-25
Name: Last Updated, Length: 9653, dtype: datetime64[ns]
```

```
[ ]  playstore_df3['LastUpdated_Day']=playstore_df3['Last Updated'].dt.day
     playstore_df3['LastUpdated_Month']=playstore_df3['Last Updated'].dt.month
     playstore_df3['LastUpdated_Year']=playstore_df3['Last Updated'].dt.year
```

1.Converted to datetime datatype

2. Separate Month, Day, year and Store in new column

User Reviews DataFrame Operation



```
[ ] #dropped the rows having all null values
    user_reviews_df2 = user_reviews_df1.dropna(subset=["Translated_Review"],how="all")
    user_reviews_df2.isnull().sum()
```

```
App                       0
Translated_Review         0
Sentiment                 0
Sentiment_Polarity        0
Sentiment_Subjectivity    0
dtype: int64
```

```
[ ] user_reviews_df2.describe()
```

	Sentiment_Polarity	Sentiment_Subjectivity
count	29692.000000	29692.000000
mean	0.188868	0.490930
std	0.355694	0.265976
min	-1.000000	0.000000
25%	0.000000	0.350000
50%	0.157143	0.514286
75%	0.422917	0.652703
max	1.000000	1.000000

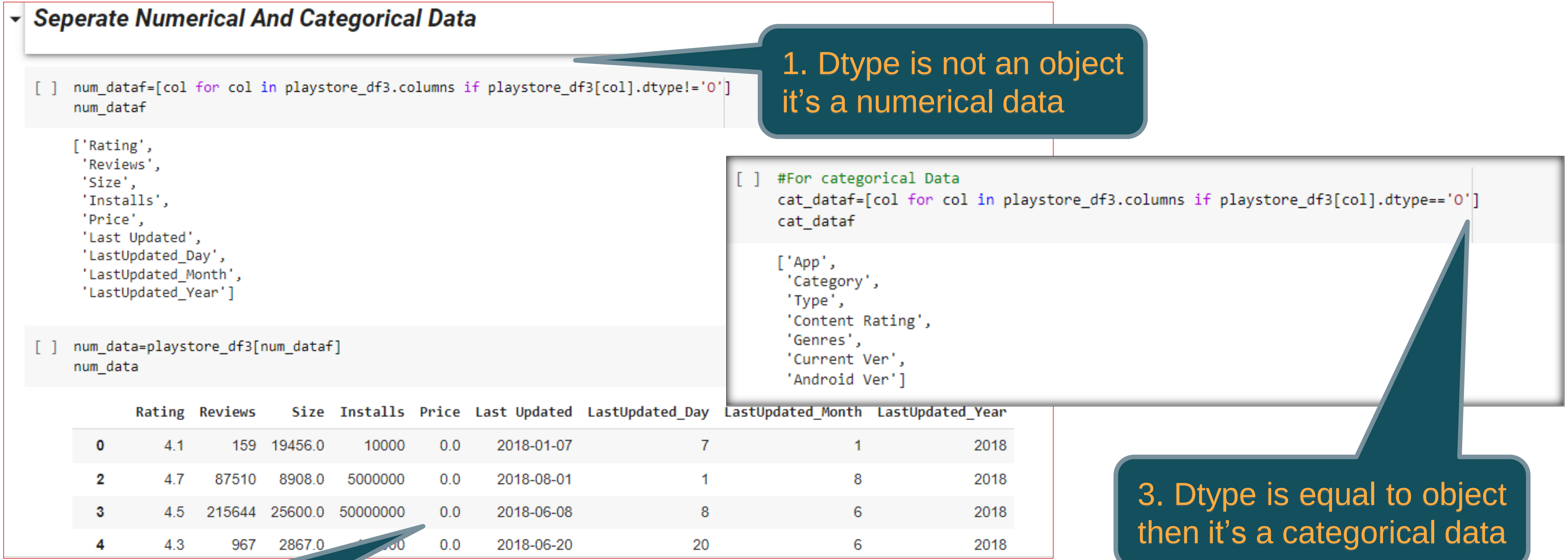
1. Unreviewed Columns Dropped

Sentiment_Subjectivity is Between 0 to 1

Sentiment_Polarity is Between -1 to 1

Segregate : Numerical, Categorical Variable Data

Segregate Numerical and Categorical Variable is useful to do process fast and easily plotting



Seperate Numerical And Categorical Data

```
[ ] num_dataf=[col for col in playstore_df3.columns if playstore_df3[col].dtype!='O']
    num_dataf
```

```
['Rating',
 'Reviews',
 'Size',
 'Installs',
 'Price',
 'Last Updated',
 'LastUpdated_Day',
 'LastUpdated_Month',
 'LastUpdated_Year']
```

```
[ ] num_data=playstore_df3[num_dataf]
    num_data
```

	Rating	Reviews	Size	Installs	Price	Last Updated	LastUpdated_Day	LastUpdated_Month	LastUpdated_Year
0	4.1	159	19456.0	10000	0.0	2018-01-07	7	1	2018
2	4.7	87510	8908.0	5000000	0.0	2018-08-01	1	8	2018
3	4.5	215644	25600.0	50000000	0.0	2018-06-08	8	6	2018
4	4.3	967	2867.0	[illegible]	0.0	2018-06-20	20	6	2018

```
[ ] #For categorical Data
    cat_dataf=[col for col in playstore_df3.columns if playstore_df3[col].dtype=='O']
    cat_dataf
```

```
['App',
 'Category',
 'Type',
 'Content Rating',
 'Genres',
 'Current Ver',
 'Android Ver']
```

1. Dtype is not an object it's a numerical data

2. Output of numerical data

3. Dtype is equal to object then it's a categorical data

Data Visualization

•**Correlations in Relationships:** Without data visualization, it is challenging to identify the correlations between the relationship of independent variables. By making sense of those independent variables, we can make better business decisions.

•**Trends Over Time:** While this seems like an obvious use of data visualization, it is also one of the most valuable applications. It's impossible to make predictions without having the necessary information from the past and present. Trends over time tell us where we were and where we can potentially go.

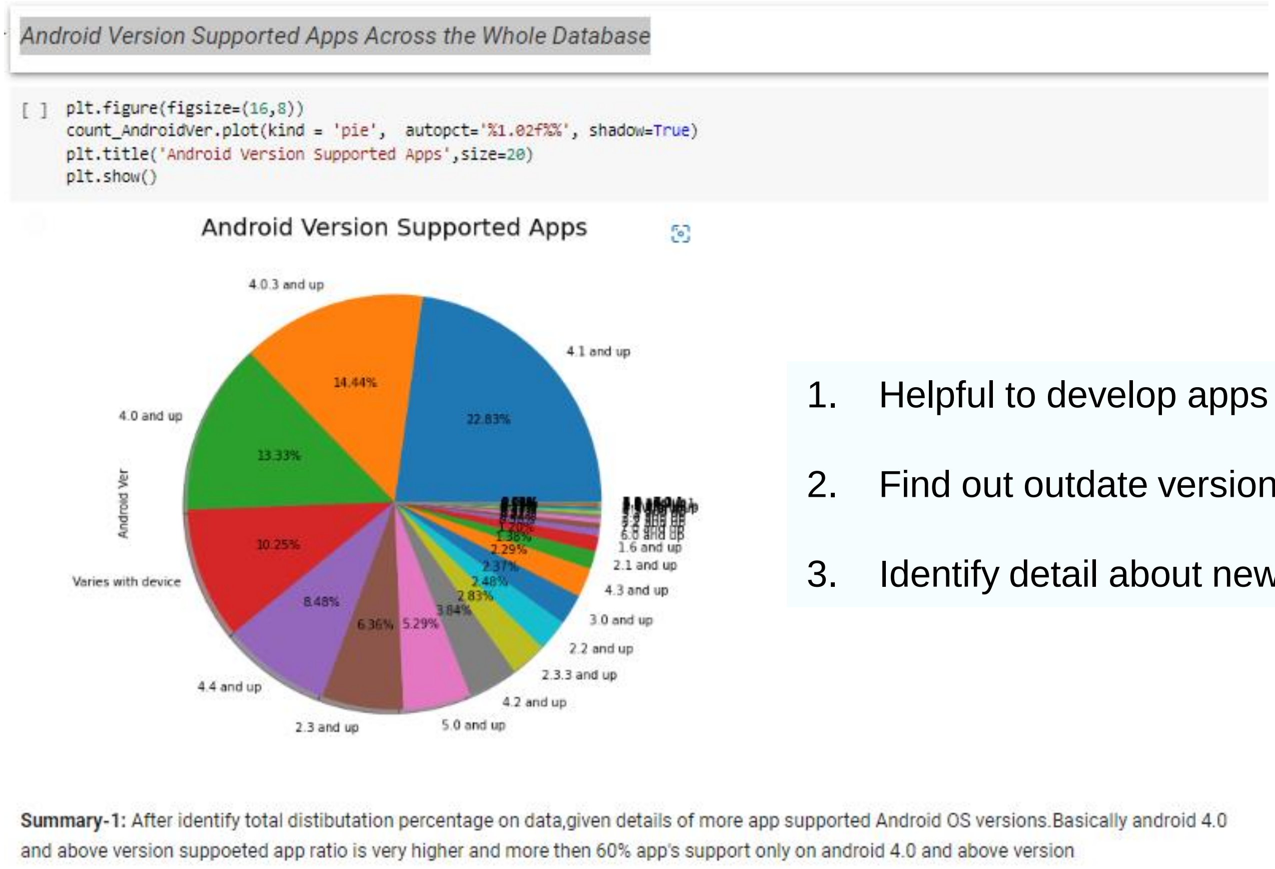
•**Frequency:** Closely related to trends over time is frequency. By examining the rate, or how often, customers purchase and when they buy gives us a better feel for how potential new customers might act and react to different marketing and customer acquisition strategies.

•**Examining the Market:** Data visualization takes the information from different markets to give you insights into which audiences to focus your attention on and which ones to stay away from. We get a clearer picture of the opportunities within those markets by displaying this data on various charts and graphs.

•**Risk and Reward:** Looking at value and risk metrics requires expertise because, without data visualization, we must interpret complicated spreadsheets and numbers. Once information is visualized, we can then pinpoint areas that may or may not require action.

•**Reacting to the Market:** The ability to obtain information quickly and easily with data displayed clearly on a functional dashboard allows businesses to act and respond to findings swiftly and helps to avoid making mistakes.

1. Most Android Ver. Supported Apps in Play store



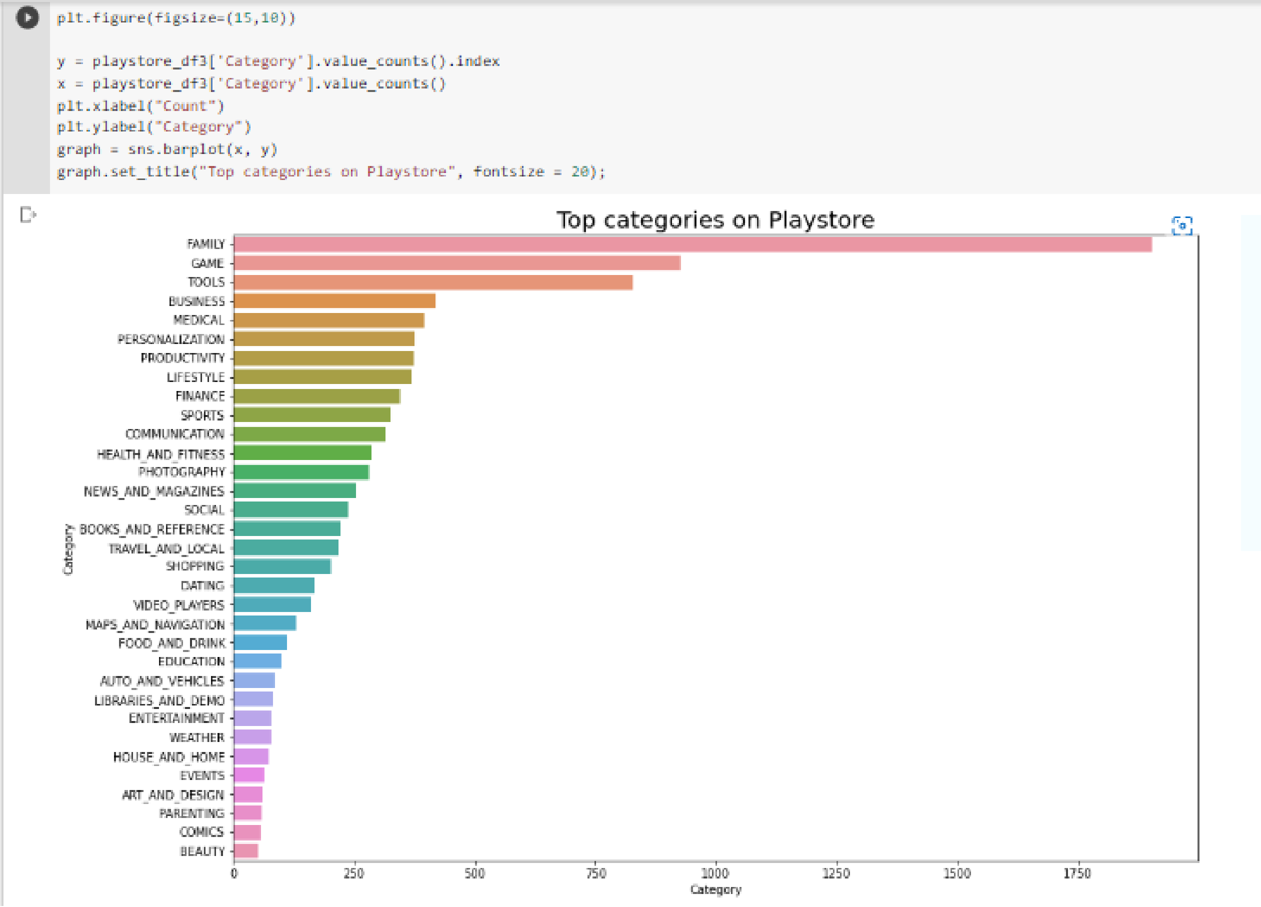
Android Version Supported Apps Across the Whole Database

```
[ ] plt.figure(figsize=(16,8))
    count_AndroidVer.plot(kind = 'pie',  autopct='%1.02f%%', shadow=True)
    plt.title('Android Version Supported Apps',size=20)
    plt.show()
```

Summary-1: After identify total distibutation percentage on data,given details of more app supported Android OS versions.Basically android 4.0 and above version suppoeted app ratio is very higher and more then 60% app's support only on android 4.0 and above version

1. Helpful to develop apps and minimum criteria
2. Find out outdate versions of apps
3. Identify detail about new trend version contribution

2. Top Categories in Play store



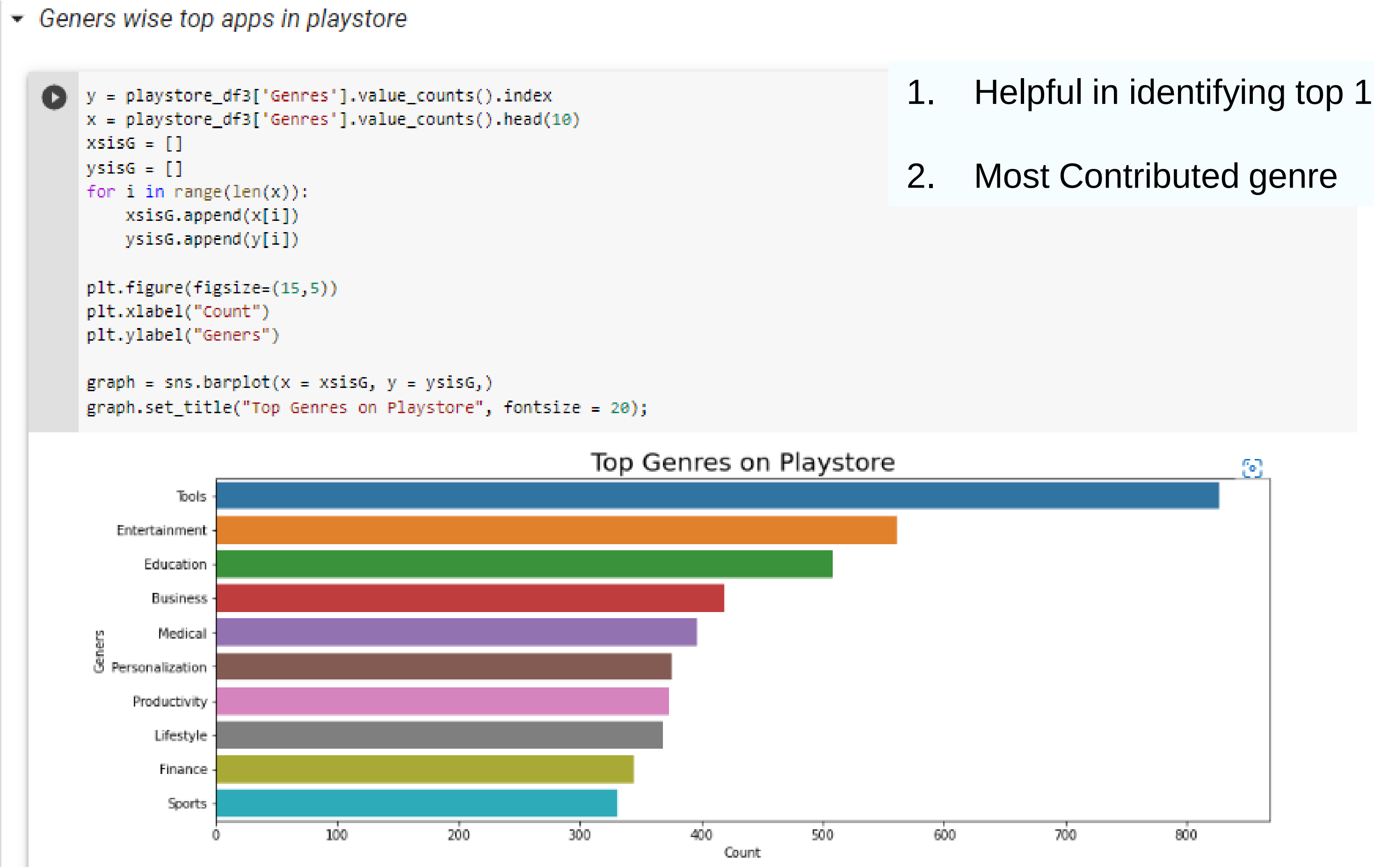
```
plt.figure(figsize=(15,10))

y = playstore_df3['Category'].value_counts().index
x = playstore_df3['Category'].value_counts()
plt.xlabel("Count")
plt.ylabel("Category")
graph = sns.barplot(x, y)
graph.set_title("Top categories on Playstore", fontsize = 20);
```

1. Helpful in identifying the common categories of app
2. Able to find the least number of apps available category wise
3. Each categories' contribution

3. Top 10 Genres in Play store

1. Helpful in identifying top 10 genres in play store

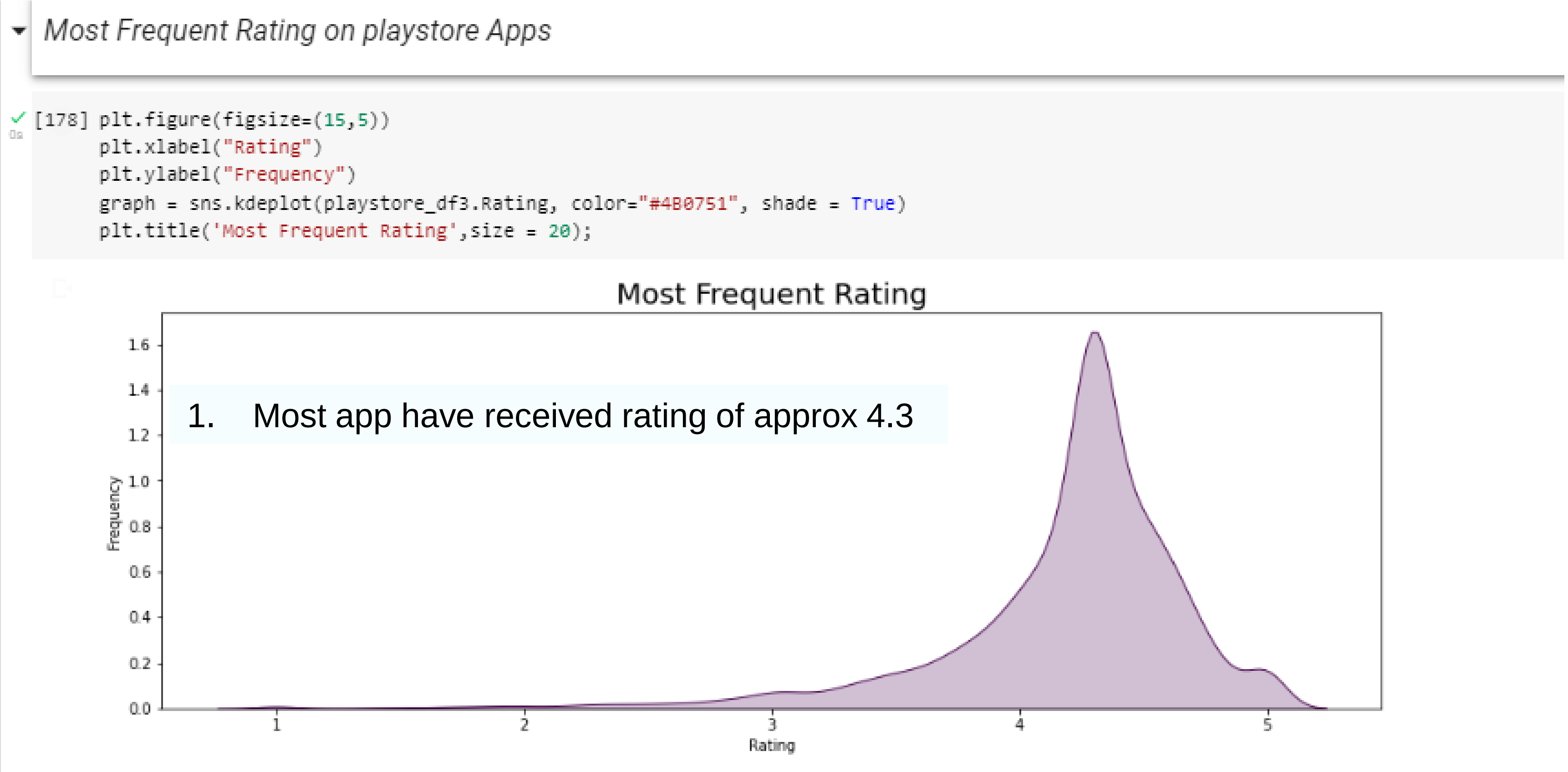
2. Most Contributed genre



Geners wise top apps in playstore

```
y = playstore_df3['Genres'].value_counts().index
x = playstore_df3['Genres'].value_counts().head(10)
xsisG = []
ysisG = []
for i in range(len(x)):
    xsisG.append(x[i])
    ysisG.append(y[i])

plt.figure(figsize=(15,5))
plt.xlabel("Count")
plt.ylabel("Geners")

graph = sns.barplot(x = xsisG, y = ysisG,)
graph.set_title("Top Genres on Playstore", fontsize = 20);
```

4. Most Frequent Rating on play store Apps

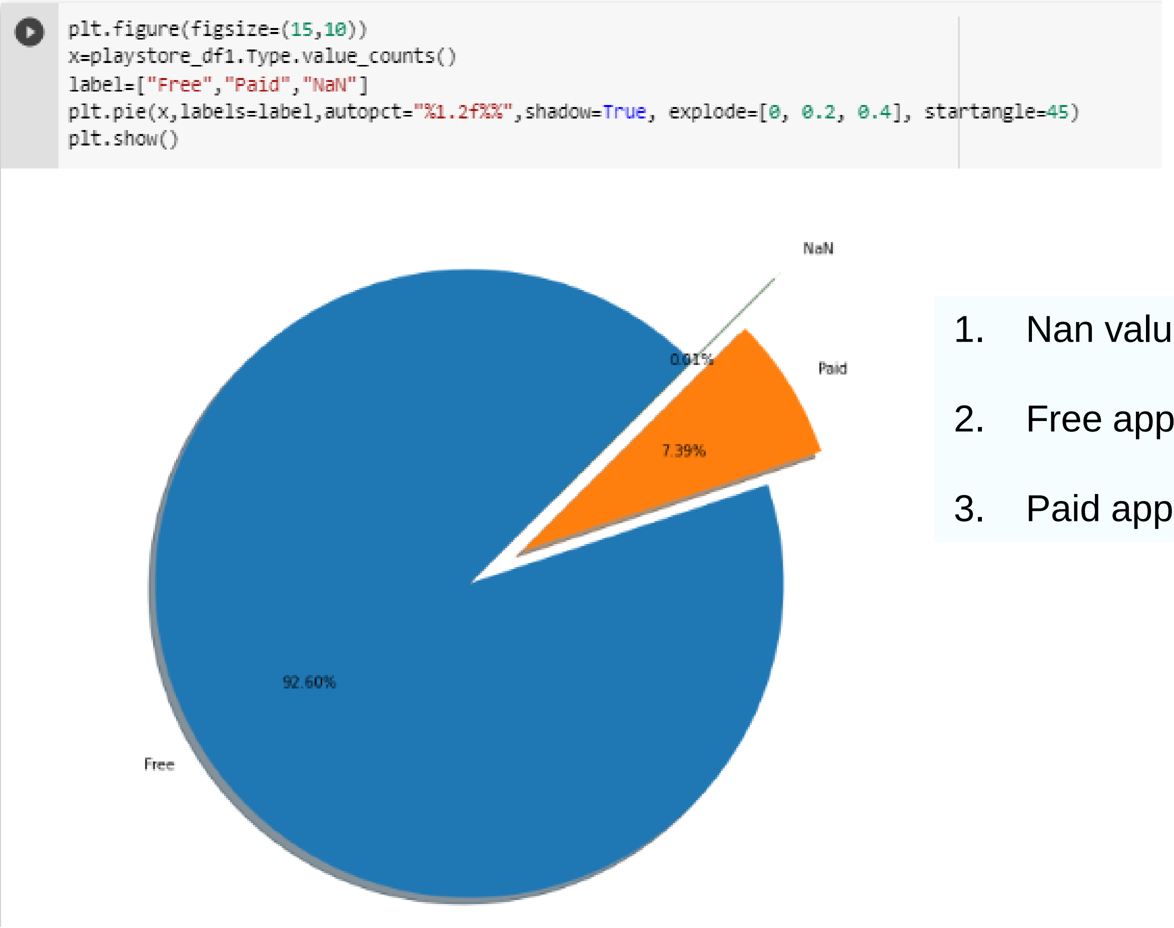


Most Frequent Rating on playstore Apps

```
[178] plt.figure(figsize=(15,5))
      plt.xlabel("Rating")
      plt.ylabel("Frequency")
      graph = sns.kdeplot(playstore_df3.Rating, color="#4B0751", shade = True)
      plt.title('Most Frequent Rating',size = 20);
```

1. Most app have received rating of approx 4.3

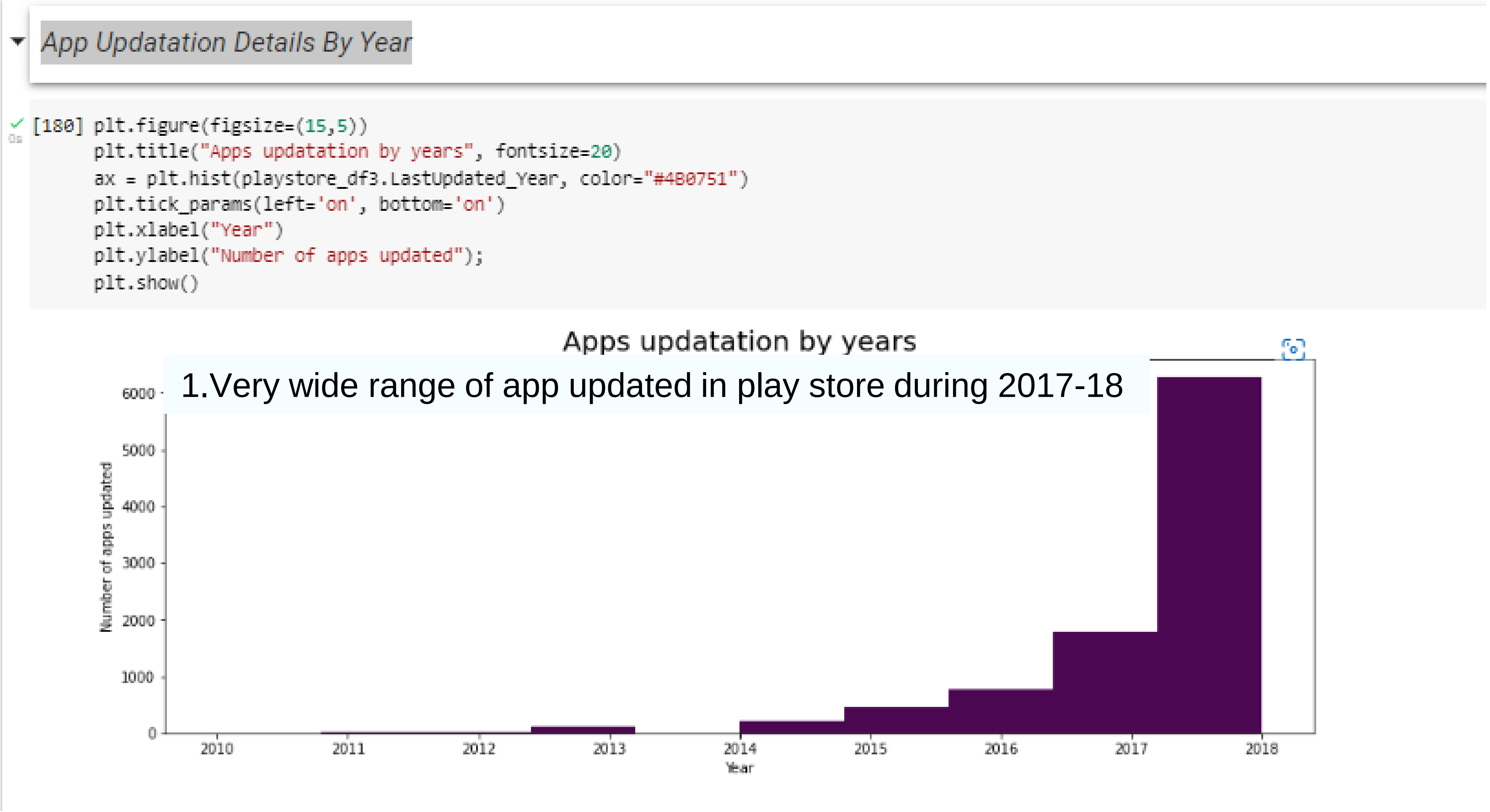
5.Paid and Free Apps Ratio Across play store



```
plt.figure(figsize=(15,10))
x=playstore_df1.Type.value_counts()
label=["Free","Paid","NaN"]
plt.pie(x,labels=label,autopct="%1.2f%%",shadow=True, explode=[0, 0.2, 0.4], startangle=45)
plt.show()
```

1. Nan value in type column are **0.01%**
2. Free apps in playstore are **92.60%**
3. Paid apps in playstore are **7.39%**

6.App Update Details ‘By Year’

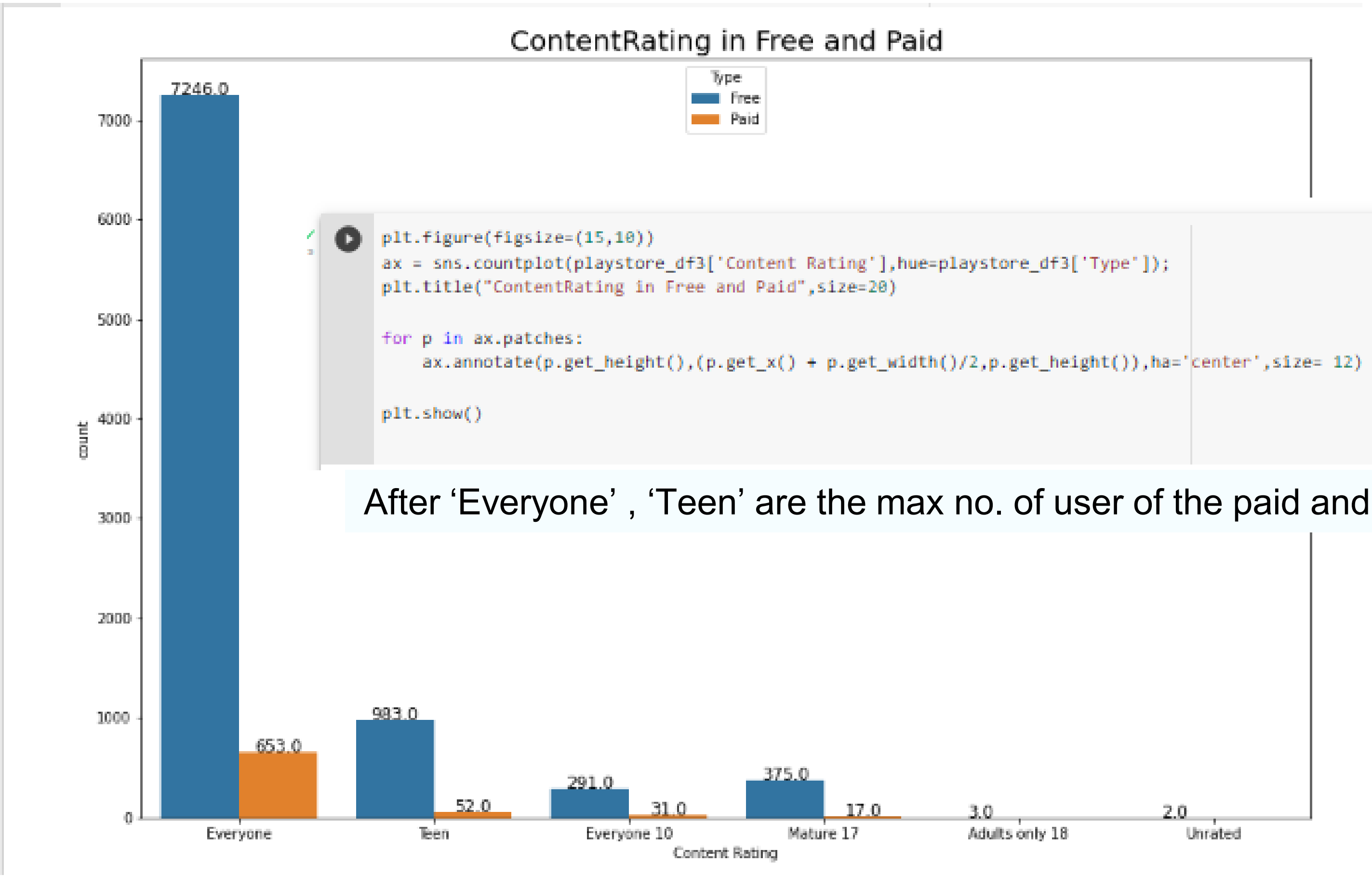


App Updatation Details By Year

```
plt.figure(figsize=(15,5))
plt.title("Apps updatation by years", fontsize=20)
ax = plt.hist(playstore_df3.LastUpdated_Year, color="#4B0751")
plt.tick_params(left='on', bottom='on')
plt.xlabel("Year")
plt.ylabel("Number of apps updated");
plt.show()
```

1.Very wide range of app updated in play store during 2017-18

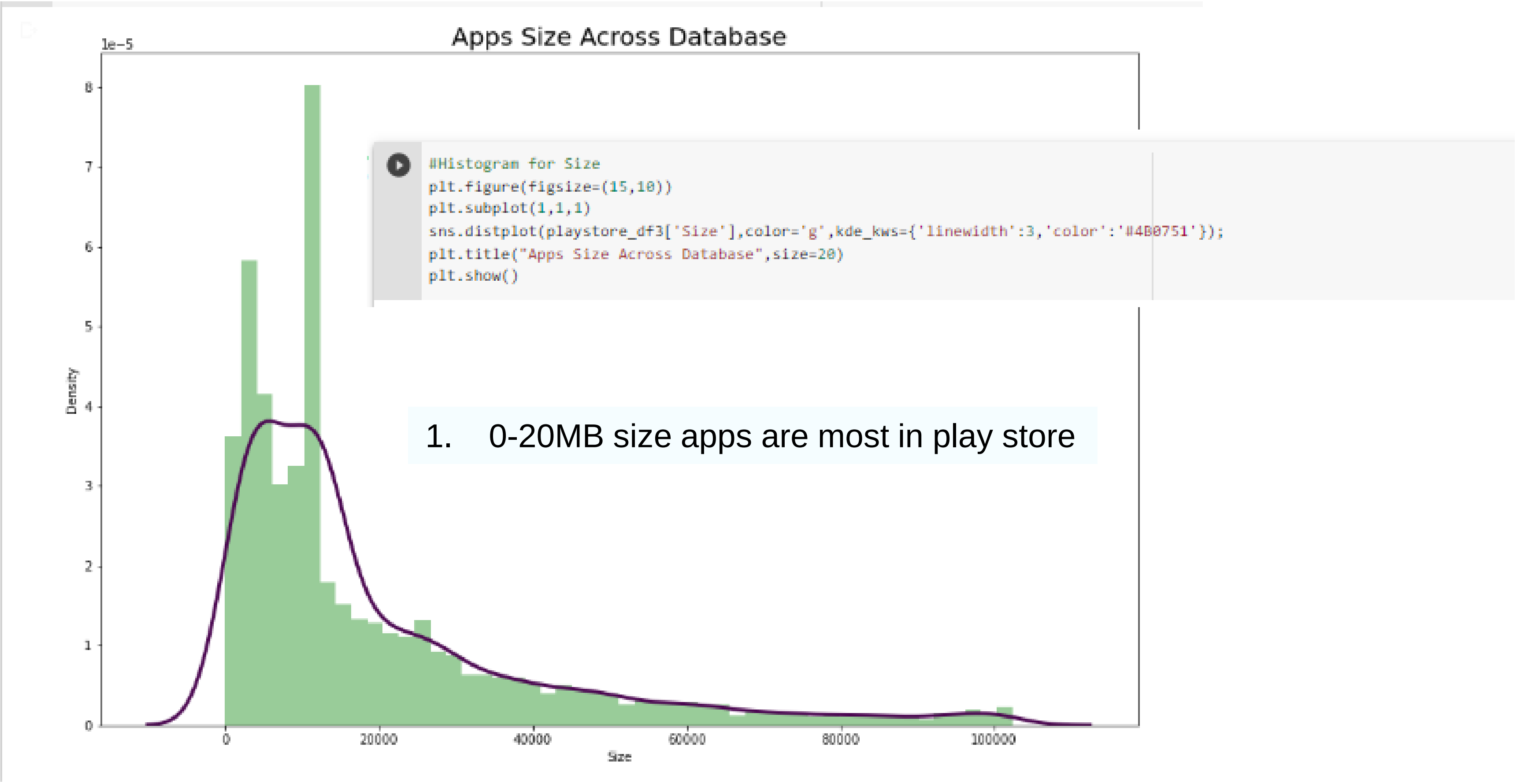
7. Age Wise Free & Paid Apps User’s Detail



```
plt.figure(figsize=(15,10))
ax = sns.countplot(playstore_df3['Content Rating'],hue=playstore_df3['Type']);
plt.title("ContentRating in Free and Paid",size=20)

for p in ax.patches:
    ax.annotate(p.get_height(),(p.get_x() + p.get_width()/2,p.get_height()),ha='center',size= 12)

plt.show()
```

After ‘Everyone’ , ‘Teen’ are the max no. of user of the paid and free apps

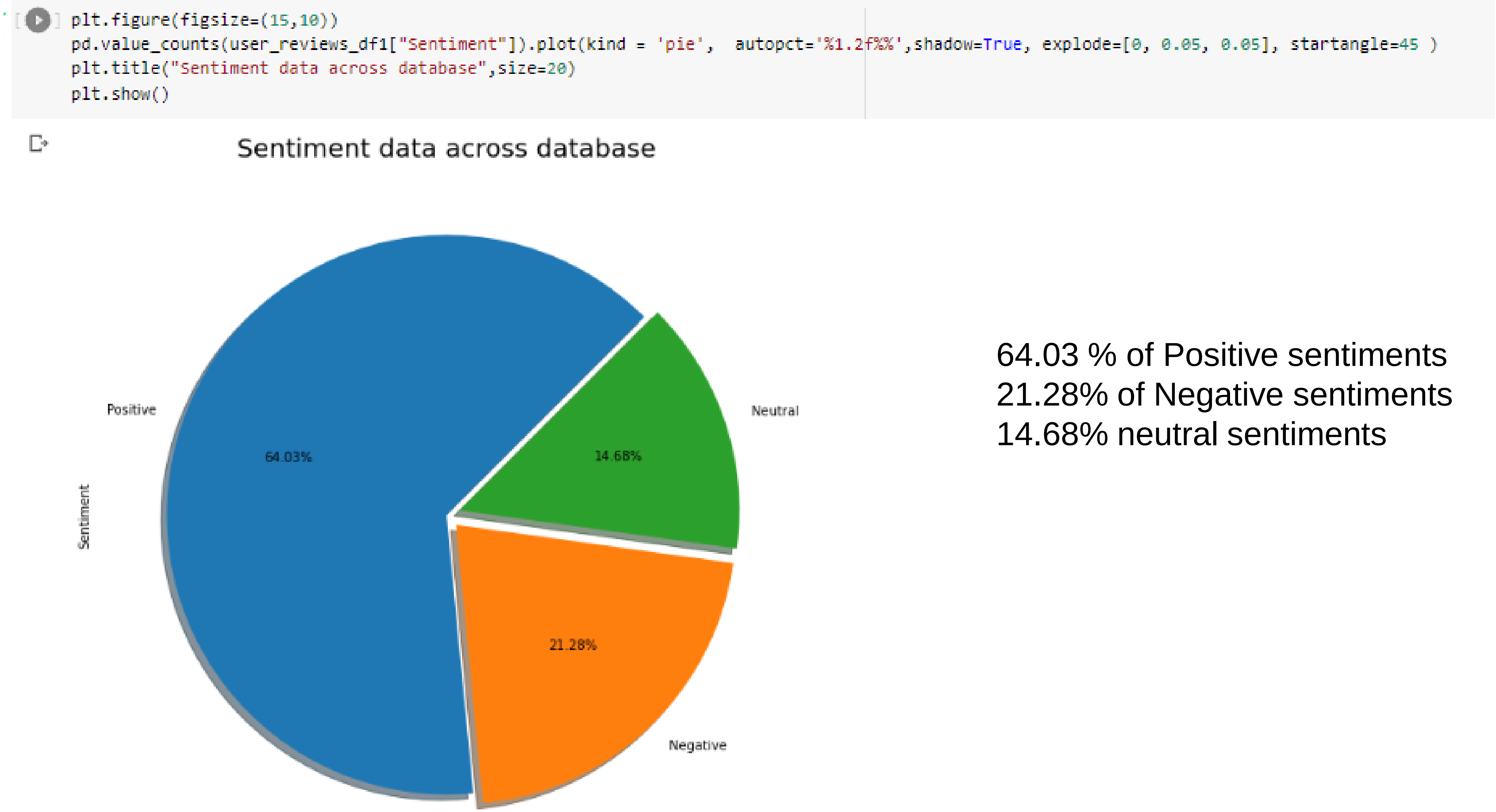
8. Apps Size Across Database



```
#Histogram for Size
plt.figure(figsize=(15,10))
plt.subplot(1,1,1)
sns.distplot(playstore_df3['Size'],color='g',kde_kws={'linewidth':3,'color':'#4B0751'});
plt.title("Apps Size Across Database",size=20)
plt.show()
```

1. 0-20MB size apps are most in play store

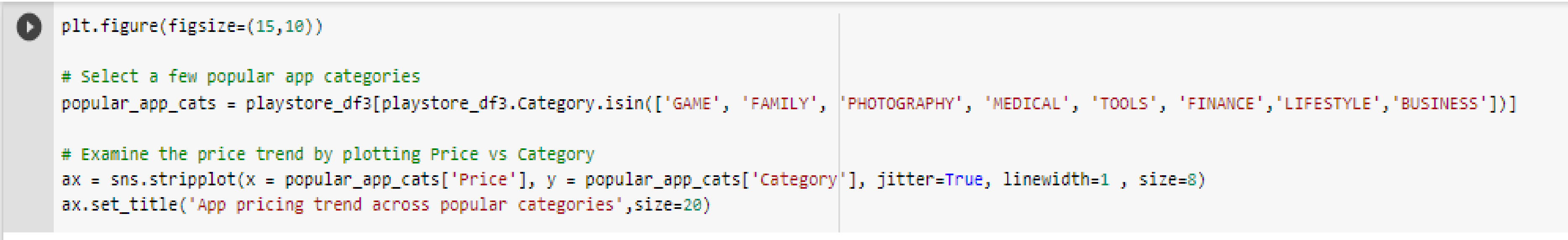
9. Sentiment Data Across the All Reviews



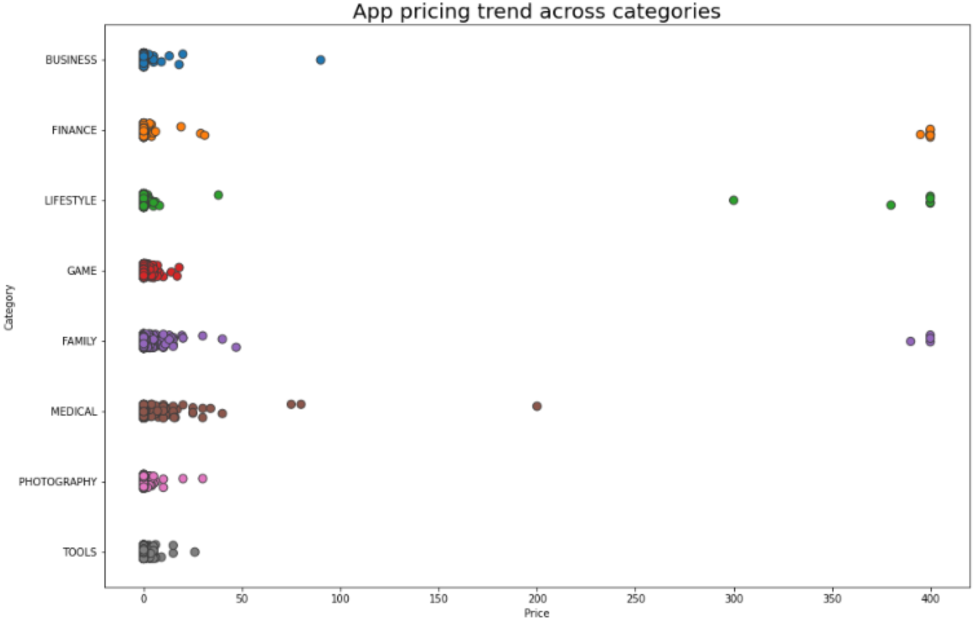
```
plt.figure(figsize=(15,10))
pd.value_counts(user_reviews_df1["Sentiment"]).plot(kind = 'pie',  autopct='%1.2f%%',shadow=True, explode=[0, 0.05, 0.05], startangle=45 )
plt.title("Sentiment data across database",size=20)
plt.show()
```

64.03 % of Positive sentiments
21.28% of Negative sentiments
14.68% neutral sentiments

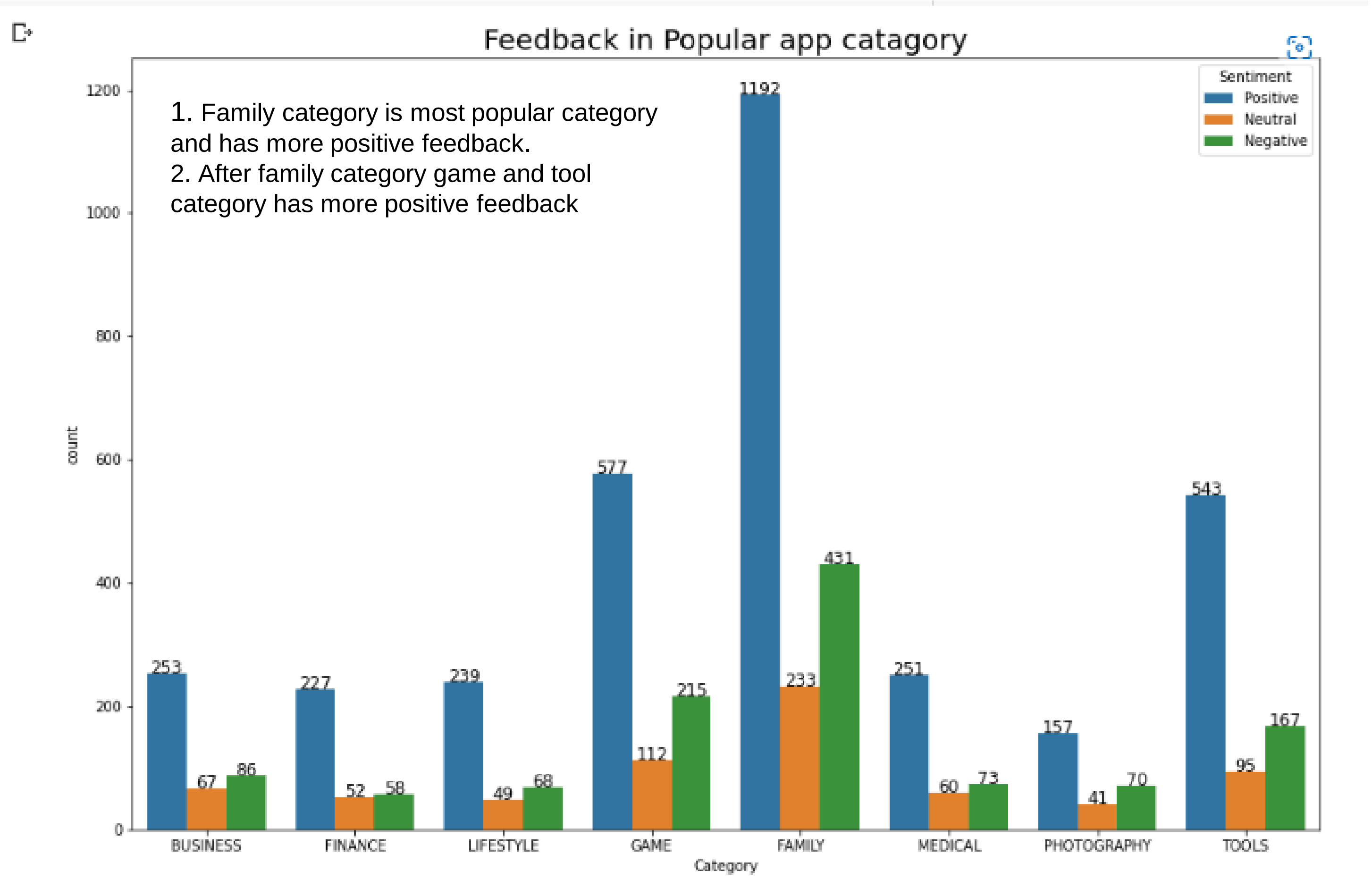
10. App Pricing Trend Across Popular Categories



```
plt.figure(figsize=(15,10))

# Select a few popular app categories
popular_app_cats = playstore_df3[playstore_df3.Category.isin(['GAME', 'FAMILY', 'PHOTOGRAPHY', 'MEDICAL', 'TOOLS', 'FINANCE','LIFESTYLE','BUSINESS'])]

# Examine the price trend by plotting Price vs Category
ax = sns.stripplot(x = popular_app_cats['Price'], y = popular_app_cats['Category'], jitter=True, linewidth=1 , size=8)
ax.set_title('App pricing trend across popular categories',size=20)
```

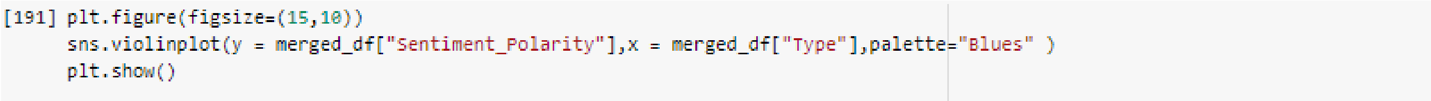


11. Feedback in Popular app catagory

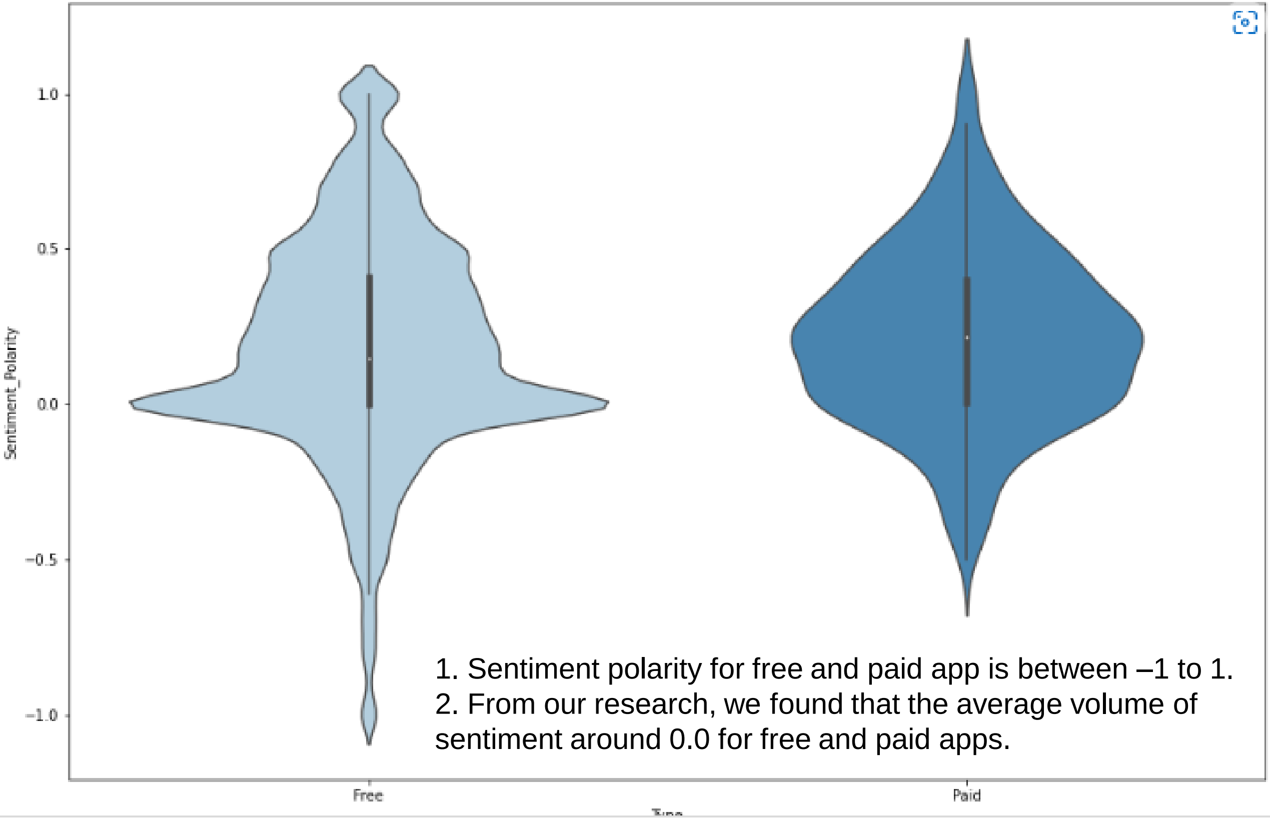


1. Family category is most popular category and has more positive feedback.
2. After family category game and tool category has more positive feedback

12. Sentiment_Polarity relation with paid and Free App

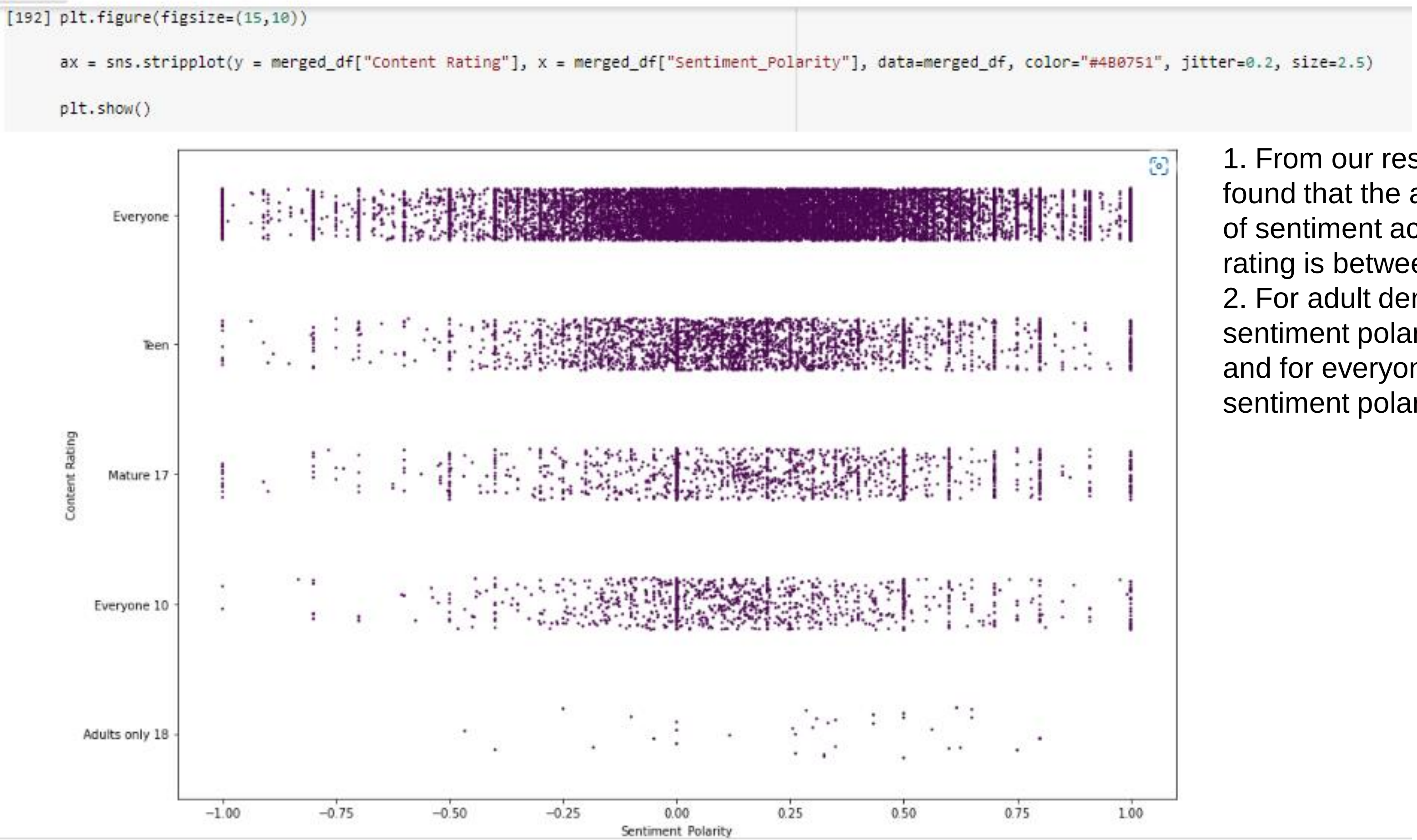


```
[191] plt.figure(figsize=(15,10))
      sns.violinplot(y = merged_df["Sentiment_Polarity"],x = merged_df["Type"],palette="Blues" )
      plt.show()
```



1. Sentiment polarity for free and paid app is between –1 to 1.
2. From our research, we found that the average volume of sentiment around 0.0 for free and paid apps.

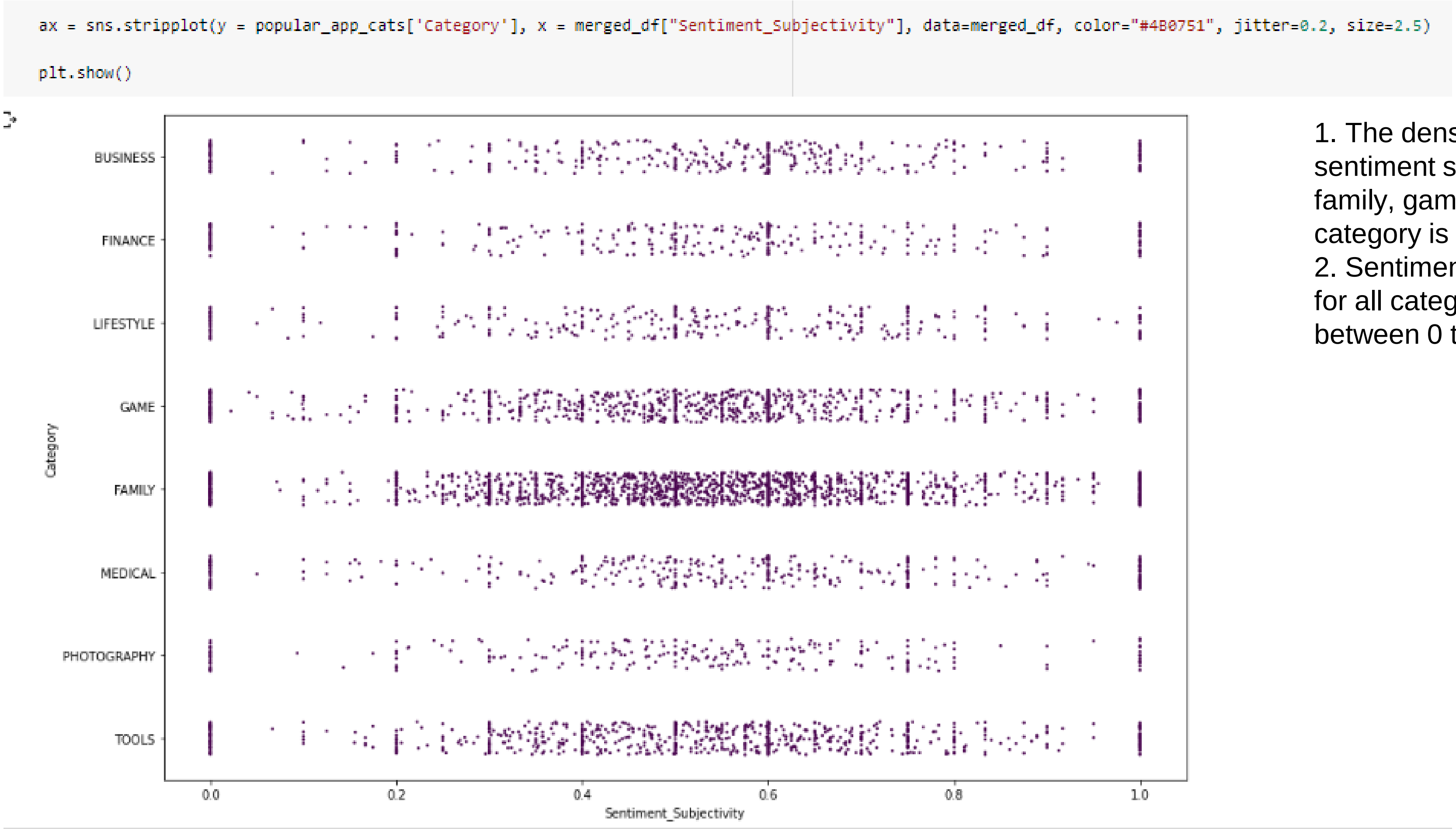
13. Content Rating Relation with Sentiment Polarity



```
[192] plt.figure(figsize=(15,10))

      ax = sns.stripplot(y = merged_df["Content Rating"], x = merged_df["Sentiment_Polarity"], data=merged_df, color="#4B0751", jitter=0.2, size=2.5)

      plt.show()
```

1. From our research, we found that the average volume of sentiment across all content rating is between -0.50 to 0.75
2. For adult density of sentiment polarity is very low and for everyone density of sentiment polarity is high.

14.Categories Relation with Sentiment_Subjectivity

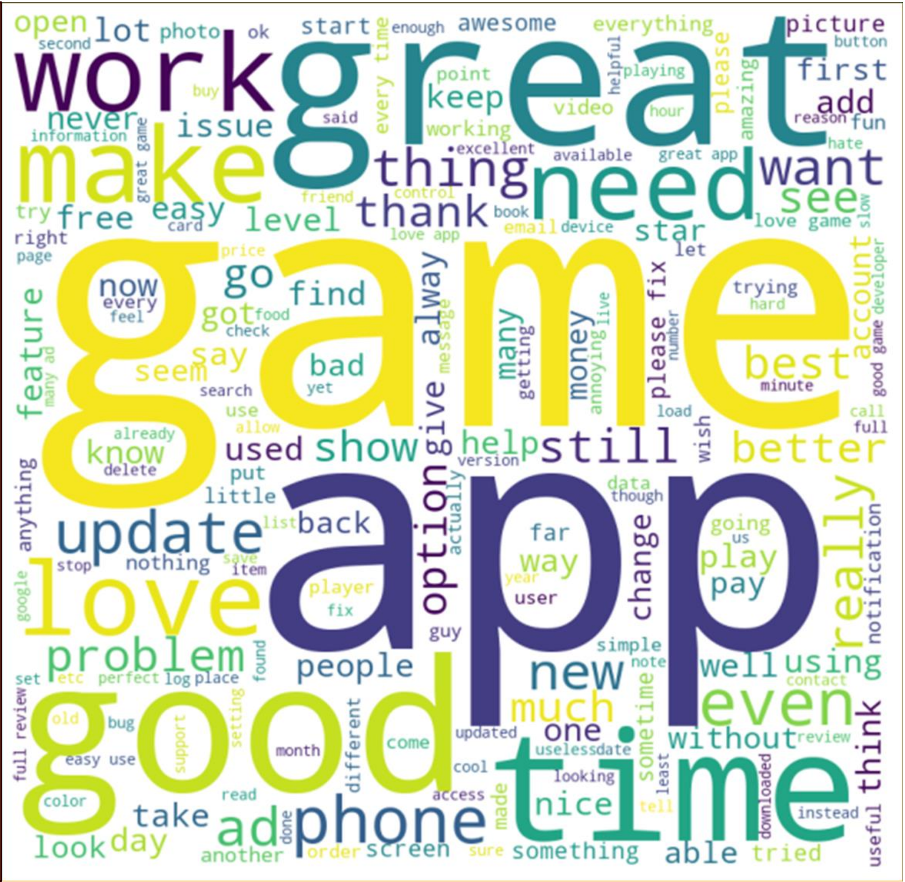


```
ax = sns.stripplot(y = popular_app_cats['Category'], x = merged_df["Sentiment_Subjectivity"], data=merged_df, color="#4B0751", jitter=0.2, size=2.5)

plt.show()
```

1. The density of sentiment subjectivity for family, games and tools category is high.
2. Sentiment subjectivity for all categories is between 0 to 1

Conclusions

- Most Apps(~65%) are compatible with android version 4.0 or above.
- Family , Games and Tools are the top 3 categories of in the given dataset.
- Tools, Entertainment, Education related genres have maximum number of apps among all 119 genres.
- 4.3 rating is the most rating that the apps get .
- Free Apps are approx. 93% compared to 7% Paid apps.
- Most of the apps (around 6k) were updated in the year 2018.
- Content Rating ‘Everyone’ belongs to almost 90% of the Apps followed by the ‘Teens’.
- Apps with size range 0-20mb have the highest share.
- 64% of the apps has positive sentiment.
- Most of the paid apps has a price range under $100.


game
app
great
work
make
need
good
time
love
update
problem
phone
thing
thank
want
see
best
better
still
show
really
even
new
much
nice
option
give
always
help
feature
free
easy
level
now
go
find
bad
know
used
back
people
play
pay
way
far
user
issue
keep
never
lot
open
photo
start
awesome
everything
picture
first
add
fun
star
using
well
without
one
think
look
day
take
ad
screen
something
able
tried