CLEANING THE GOOGLE PLAYSTORE DATASET

In [2]:
```
# Importing the libraries
import re
import pandas as pd
import numpy as np
import os
```

In [3]:
```
current_directory = os.getcwd()
print("Current Directory:", current_directory)

files = os.listdir()
print("Files in Directory:", files)
```

```
Current Directory: C:\Users\Amarachi\Documents\HNG Data Analysis\Task 4
Files in Directory: ['.ipynb_checkpoints', 'appleAppData.csv', 'cleaned_appstore.cs
v', 'cleaned_googleplaystore.csv', 'Google-Playstore.csv', 'googleplaystore.csv', 'g
oogleplaystore_user_reviews.csv', 'HNG_Stage_4_Data_Cleaning.ipynb', 'playstorerevie
ws.csv']
```

In [4]:
```
# Importing the dataset
data = pd.read_csv("Google-Playstore.csv")
```

- Initial Exploration of the Dataset

In [6]:
```
# Check the datatypes of the columns
data.dtypes
```

Out[6]:
```
App Name              object
App Id                object
Category              object
Rating               float64
Rating Count         float64
Installs              object
Minimum Installs     float64
Maximum Installs       int64
Free                    bool
Price                float64
Currency              object
Size                  object
Minimum Android       object
Developer Id          object
Developer Website     object
Developer Email       object
Released              object
Last Updated          object
Content Rating        object
Privacy Policy        object
Ad Supported            bool
In App Purchases        bool
Editors Choice          bool
Scraped Time          object
dtype: object
```

In [7]:
```
# Check the data shape
data.shape
```

Out[7]:
```
(2312944, 24)
```

In [8]:
```
data.head()
```

Out[8]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10+	10.0
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5,000+	5000.0
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50+	50.0
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10+	10.0
4	GROW.me	com.horodyski.grower	Tools	0.0	0.0	100+	100.0

5 rows × 24 columns

In [9]:
```
# Check the columns information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2312944 entries, 0 to 2312943
Data columns (total 24 columns):
 #   Column             Dtype  
---  ------             -----  
 0   App Name           object 
 1   App Id             object 
 2   Category           object 
 3   Rating             float64
 4   Rating Count       float64
 5   Installs           object 
 6   Minimum Installs   float64
 7   Maximum Installs   int64  
 8   Free               bool   
 9   Price              float64
 10  Currency           object 
 11  Size               object 
 12  Minimum Android    object 
 13  Developer Id       object 
 14  Developer Website  object 
 15  Developer Email    object 
 16  Released           object 
 17  Last Updated       object 
 18  Content Rating     object 
 19  Privacy Policy     object 
 20  Ad Supported       bool   
 21  In App Purchases   bool   
 22  Editors Choice     bool   
 23  Scraped Time       object 
dtypes: bool(4), float64(4), int64(1), object(15)
memory usage: 361.8+ MB
```

In [10]:
```
# Check the decriptive statistics of the data
data.describe()
```

Out[10]:

	Rating	Rating Count	Minimum Installs	Maximum Installs	Price
count	2.290061e+06	2.290061e+06	2.312837e+06	2.312944e+06	2.312944e+06
mean	2.203152e+00	2.864839e+03	1.834452e+05	3.202017e+05	1.034992e-01
std	2.106223e+00	2.121626e+05	1.513144e+07	2.355495e+07	2.633127e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00	5.000000e+01	8.400000e+01	0.000000e+00
50%	2.900000e+00	6.000000e+00	5.000000e+02	6.950000e+02	0.000000e+00
75%	4.300000e+00	4.200000e+01	5.000000e+03	7.354000e+03	0.000000e+00
max	5.000000e+00	1.385576e+08	1.000000e+10	1.205763e+10	4.000000e+02

In [11]:
```
# Convert the date columns from object to datetime
data['Released'] = pd.to_datetime(data['Released'])
data['Last Updated'] = pd.to_datetime(data['Last Updated'])
```

```
#Create a new column for the age of the apps on Google Playstore
data['Released'] = data['Released'].dt.tz_localize(None)

# Get current date as timezone-naive
current_date = pd.to_datetime('today')

# Calculate the difference (in years)
data['Android_App_Age'] = ((current_date - data['Released']).dt.days // 365).fillna

# Show the result
data['Android_App_Age'].head()
```

Out[11]:
```
0    5
1    4
2    5
3    6
4    5
Name: Android_App_Age, dtype: int32
```

In [12]:
```
# Check for duplicate values
data.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
# Check for missing values
data.isnull().sum()
```

Out[13]:
```
App Name                 5
App Id                   0
Category                 0
Rating               22883
Rating Count         22883
Installs               107
Minimum Installs       107
Maximum Installs         0
Free                     0
Price                    0
Currency               135
Size                   196
Minimum Android       6530
Developer Id            33
Developer Website    760835
Developer Email          31
Released              71053
Last Updated              0
Content Rating            0
Privacy Policy       420953
Ad Supported              0
In App Purchases          0
Editors Choice            0
Scraped Time              0
Android_App_Age           0
dtype: int64
```

In [14]:
```
# Cleaning the App Name Column
def comprehensive_clean(text, app_id):
```

```
    # Ensure the value is a string
    if not isinstance(text, str):
        text = str(text)

    # Remove non-ASCII characters
    text = re.sub(r'[^\x00-\x7F]+', '', text)
    # Remove special characters (except spaces)
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Remove Chinese and Arabic characters
    text = re.sub(r'[\u4e00-\u9fff\u0600-\u06FF]', '', text)

    # If cleaned text is empty, extract the word after the first dot in App Id
    if not text.strip():  # Check if the cleaned text is empty
        if isinstance(app_id, str):
            # Extract the word after the first dot
            match = re.search(r'\.([a-zA-Z0-9]+)', app_id)
            if match:
                return match.group(1)  # Return the word after the first dot
    return text

# Apply the function to 'App Name' column, using 'App Id' as a fallback
data['Playstore_App'] = data.apply(lambda row: comprehensive_clean(row['App Name'],
data['Playstore_App'].head(1)
```

Out[14]:
```
0    Gakondo
Name: Playstore_App, dtype: object
```

In [15]:
```
# Addressing the missing values in Rating and Rating Count by Imputation
columns_to_impute = ['Rating', 'Rating Count']

for col in columns_to_impute:
    data[col] = pd.to_numeric(data[col], errors='coerce')  # Convert to numeric, fo

# Replace missing values with the column mean
data[columns_to_impute] = data[columns_to_impute].apply(lambda x: x.fillna(x.mean()

# Print the updated DataFrame
data[columns_to_impute].isnull().sum()
```

Out[15]:
```
Rating          0
Rating Count    0
dtype: int64
```

In [16]:
```
# Replacing the null values in Installs with Maximum Installs
data['Installs'] = data['Installs'].fillna(data['Maximum Installs'])

# Fill the blank values in Minimum Installs with 0
data['Minimum Installs'] = data['Minimum Installs'].fillna(0)

data[['Installs', 'Minimum Installs']].isnull().sum()
```

Out[16]:
```
Installs           0
Minimum Installs    0
dtype: int64
```

In [17]:
```python
#Categorizing the Installs Column
def categorize_installs(value):
    if isinstance(value, str) and '+' in value:
        return value  # Keep values that already have '+'

    try:
        value = int(value)  # Convert numeric values
        if value < 5:
            return '1+'
        elif value < 10:
            return '5+'
        elif value < 50:
            return '10+'
        elif value < 100:
            return '50+'
        elif value < 500:
            return '100+'
        elif value < 1_000:
            return '500+'
        elif value < 5_000:
            return '1,000+'
        elif value < 10_000:
            return '5,000+'
        elif value < 50_000:
            return '10,000+'
        elif value < 100_000:
            return '50,000+'
        elif value < 500_000:
            return '100,000+'
        elif value < 1_000_000:
            return '500,000+'
        elif value < 5_000_000:
            return '1,000,000+'
        elif value < 10_000_000:
            return '5,000,000+'
        elif value < 50_000_000:
            return '10,000,000+'
        elif value < 100_000_000:
            return '50,000,000+'
        elif value < 500_000_000:
            return '100,000,000+'
        elif value < 1_000_000_000:
            return '500,000,000+'
        elif value < 5_000_000_000:
            return '1,000,000,000+'
        else:
            return '5,000,000,000+'
    except:
        return 'Unknown'  # Catch any unexpected values

# Apply the function to the "Installs" column
data['Installs'] = data['Installs'].apply(categorize_installs)

# Print result
print(data['Installs'].value_counts())
```

```
Installs
100+                443369
1,000+              398212
10+                 300164
10,000+             256736
500+                189081
50+                 170466
5,000+              143601
100,000+            110263
50,000+              75367
5+                   73775
1+                   65385
1,000,000+           33652
500,000+             27012
0+                   11566
5,000,000+            6595
10,000,000+           6192
50,000,000+            824
100,000,000+           549
500,000,000+            65
1,000,000,000+          55
5,000,000,000+          14
10,000,000,000+          1
Name: count, dtype: int64
```

In [18]:
```
# Handling the Size Xolumn
def convert_size(size):
    if isinstance(size, str):  # Ensure it's a string before processing
        size = size.replace(",", "")  # Remove commas if present
        if 'G' in size:
            return float(size.replace('G', '')) * 1_000_000_000  # Convert GB to by
        elif 'M' in size:
            return float(size.replace('M', '')) * 1_000_000  # Convert MB to bytes
        elif 'K' in size:
            return float(size.replace('K', '')) * 1_000  # Convert KB to bytes
        else:
            return np.nan  # Return NaN for unexpected values like 'Varies with dev
    return np.nan  # Ensure NaN for missing values

# Apply conversion function
data["Size"] = data["Size"].apply(convert_size)
data['Size'].head(1)
```

Out[18]:
```
0    10000000.0
Name: Size, dtype: float64
```

In [19]:
```
#Replacing missing values with "Unknown" for the remaining columns
columns_to_replace = ['Currency', 'Size', 'Minimum Android', 'Developer Website', '

# Convert to string before replacing NaNs
data[columns_to_replace] = data[columns_to_replace].astype(str).replace('nan', 'Unk

# Alternatively, using fillna()
data[columns_to_replace] = data[columns_to_replace].fillna("Unknown")
```

In [20]:
```
data = data.drop(columns=["App Name", "Scraped Time", "App Id", "Developer Id", "De
```

FINAL RESULT

In [22]:
```
data.head(3)
```

Out[22]:

	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Currency	
0	Adventure	0.0	0.0	10+	10.0	15	True	0.0	USD	1000(
1	Tools	4.4	64.0	5,000+	5000.0	7662	True	0.0	USD	290(
2	Productivity	0.0	0.0	50+	50.0	58	True	0.0	USD	370(

CLEANING THE APPLE STORE DATASET

In [24]:
```
# Load the Apple App Store data
apple = pd.read_csv("appleAppData.csv")
```

- Initial Data Exploration

In [26]:
```
apple.head(3)
```

Out[26]:

	App_Id	App_Name	AppStore_Url	Primary_Genre	Content
0	com.hkbu.arc.apaper	A+ Paper Guide	https://apps.apple.com/us/app/a-paper-guide/id...	Education	
1	com.dmitriev.abooks	A-Books	https://apps.apple.com/us/app/a-books/id103157...	Book	
2	no.terp.abooks	A-books	https://apps.apple.com/us/app/a-books/id145702...	Book	

3 rows × 21 columns

In [27]:
```
# Check for blank values in the apple data
apple.isnull().sum()
```

Out[27]:
```
App_Id                       0
App_Name                     1
AppStore_Url                 0
Primary_Genre                0
Content_Rating               0
Size_Bytes                 224
Required_IOS_Version         0
Released                     3
Updated                      0
Version                      0
Price                      490
Currency                     0
Free                         0
DeveloperId                  0
Developer                    0
Developer_Url             1109
Developer_Website       643988
Average_User_Rating          0
Reviews                      0
Current_Version_Score        0
Current_Version_Reviews      0
dtype: int64
```

In [28]:
```
#Check for duplicates
apple.duplicated().sum()
```

Out[28]:
```
0
```

In [29]:
```
# Check the dataset shape
apple.shape
```

Out[29]:
```
(1230376, 21)
```

- Data Cleaning

In [31]:
```
# Handle missing values in Size_Bytes column with median
median_size = apple['Size_Bytes'].median()
apple.loc[:, 'Size_Bytes'] = apple['Size_Bytes'].fillna(median_size)

# Handle missing values in Price column based on the Free column
median_price = apple['Price'].median()
apple.loc[:, 'Price'] = apple.apply(
    lambda row: 0 if row['Free'] else (median_price if pd.isnull(row['Price']) else
    axis=1
)

# Check the result
print(apple[["Size_Bytes", "Price"]].isnull().sum())
```

```
Size_Bytes    0
Price         0
dtype: int64
```

In [32]:
```
# Mapping the rating columns from the two datasets
rating_mapping = {
    '4+': 'Everyone',
    '9+': 'Everyone 10+',
    '12+': 'Teen',
    '17+': 'Mature 17+',
    'Not yet rated': 'Unrated'
}

# Apply the mapping to the Apple dataset
apple["Content_Rating"] = apple["Content_Rating"].replace(rating_mapping)

# Now both datasets have the same rating categories
print(apple["Content_Rating"].unique())
print(data["Content Rating"].unique())
```

```
['Everyone' 'Mature 17+' 'Everyone 10+' 'Teen' 'Unrated']
['Everyone' 'Teen' 'Mature 17+' 'Everyone 10+' 'Adults only 18+' 'Unrated']
```

In [33]:
```
#Convert the release date and updated date in apple data to datetime
apple['Released'] = pd.to_datetime(apple['Released'])
apple['Updated'] = pd.to_datetime(apple['Updated'])

#Creating App Age Columns for the App Store Applications
apple['Released'] = apple['Released'].dt.tz_localize(None)

# Get current date as timezone-naive
current_date = pd.to_datetime('today')

# Calculate the difference (in years)
apple['Apple_App_Age'] = ((current_date - apple['Released']).dt.days // 365).fillna

# Show the result
apple['Apple_App_Age'].head(3)
```

Out[33]:
```
0    7
1    9
2    3
Name: Apple_App_Age, dtype: int32
```

In [34]:
```
apple.rename(columns={'Primary_Genre': 'Category'}, inplace=True)
```

In [35]:
```
#Dropping Unnecessary Columns
apple = apple.drop(columns=["App_Id", "AppStore_Url", "Required_IOS_Version", "Vers
                            "Current_Version_Reviews"])
```

FINAL RESULT

In [37]:
```
apple.head(4)
```

Out[37]:

	App_Name	Category	Content_Rating	Size_Bytes	Released	Updated	Price	Curr
0	A+ Paper Guide	Education	Everyone	21993472.0	2017-09-28 03:02:41	2018-12-21 21:30:36+00:00	0.00	
1	A-Books	Book	Everyone	13135872.0	2015-08-31 19:31:32	2019-07-23 20:31:09+00:00	0.00	
2	A-books	Book	Everyone	21943296.0	2021-04-14 07:00:00	2021-05-30 21:08:54+00:00	0.00	
3	A-F Book #1	Book	Everyone	81851392.0	2012-02-10 03:40:07	2019-10-29 12:40:37+00:00	2.99	

CLEANING THE REVIEWS DATASET

In [39]:
```
review1 = pd.read_csv("googleplaystore.csv")
```

In [40]:
```
review2 = pd.read_csv("googleplaystore_user_reviews.csv")
```

In [41]:
```
review = review1.merge(review2, on='App', how='inner')
```

In [42]:
```
review.head(2)
```

Out[42]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	C
0	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev

In [43]:
```
review['Category'].unique()
```

Out[43]:
```
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'],
      dtype=object)
```

In [44]:
```
# Display basic info
print(review.info())

# Check for missing values
print(review.isnull().sum())

# Check for duplicates
print("Duplicate rows:", review.duplicated().sum())

# Display first few rows
review.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122662 entries, 0 to 122661
Data columns (total 18 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   Unnamed: 0              122662 non-null  int64  
 1   App                     122662 non-null  object 
 2   Category                122662 non-null  object 
 3   Rating                  122622 non-null  float64
 4   Reviews                 122662 non-null  object 
 5   Size                    122662 non-null  object 
 6   Installs                122662 non-null  object 
 7   Type                    122662 non-null  object 
 8   Price                   122662 non-null  object 
 9   Content Rating          122662 non-null  object 
 10  Genres                  122662 non-null  object 
 11  Last Updated            122662 non-null  object 
 12  Current Ver             122662 non-null  object 
 13  Android Ver             122662 non-null  object 
 14  Translated_Review       72605 non-null   object 
 15  Sentiment               72615 non-null   object 
 16  Sentiment_Polarity      72615 non-null   float64
 17  Sentiment_Subjectivity  72615 non-null   float64
dtypes: float64(3), int64(1), object(14)
memory usage: 16.8+ MB
None
Unnamed: 0                    0
App                           0
Category                      0
Rating                       40
Reviews                       0
Size                          0
Installs                      0
Type                          0
Price                         0
Content Rating                0
Genres                        0
Last Updated                  0
Current Ver                   0
Android Ver                   0
Translated_Review         50057
Sentiment                 50047
Sentiment_Polarity        50047
Sentiment_Subjectivity    50047
dtype: int64
Duplicate rows: 74518
```

Out[44]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type	Price	C
0	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev
2	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev
3	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev
4	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Ev

In [45]:
```
review.drop_duplicates(inplace=True)
review.loc[:, "Rating"] = review["Rating"].fillna(review["Rating"].median())
review.loc[:, "Sentiment_Polarity"] = review["Sentiment_Polarity"].fillna(review["S
review.loc[:, "Sentiment_Subjectivity"] = review["Sentiment_Subjectivity"].fillna(r
review.loc[:, "Sentiment"] = review["Sentiment"].fillna(review["Sentiment"].mode()[
review.loc[:, "Translated_Review"] = review["Translated_Review"].fillna("No Review"
```

In [46]:
```
# Check for missing values
print(review.isnull().sum())

# Check for duplicates
print("Duplicate rows:", review.duplicated().sum())
```

```
Unnamed: 0                0
App                       0
Category                  0
Rating                    0
Reviews                   0
Size                      0
Installs                  0
Type                      0
Price                     0
Content Rating            0
Genres                    0
Last Updated              0
Current Ver               0
Android Ver               0
Translated_Review         0
Sentiment                 0
Sentiment_Polarity        0
Sentiment_Subjectivity    0
dtype: int64
Duplicate rows: 0
```

In [47]:
```
# Convert all category names to lowercase
data["Category"] = data["Category"].str.lower()
review["Category"] = review["Category"].str.lower()
apple["Category"] = apple["Category"].str.lower()
```

In [48]:
```
review["Category"] = review["Category"].str.replace("_", " ")
data["Category"] = data["Category"].str.replace("_", " ")

# Mapping dictionary remains the same
category_mapping = {
    "health_and_fitness": "health & fitness",
    "communication": "social networking",
    "education": "education",
    "sports": "sports",
    "lifestyle": "lifestyle",
    "maps_and_navigation": "navigation",
    "business": "business",
    "social": "social networking",
    "entertainment": "entertainment",
    "medical": "medical",
    "game": "games",
    "books_and_reference": "book",
    "food_and_drink": "food & drink",
    "weather": "weather",
    "news_and_magazines": "news",
    "music": "music",
    "travel_and_local": "travel",
    "shopping": "shopping",
    "art_and_design": "graphics & design",
    "libraries_and_demo": "graphics & design",
    "finance": "finance",
    "productivity": "productivity",
    "tools": "utilities",
    "photography": "photo & video",
    "video_players": "photo & video",
```

```
    "dating": "social networking",
    "house_and_home": "lifestyle",
    "parenting": "lifestyle",
    "events": "lifestyle",
    "beauty": "lifestyle"
}

# Apply mapping
data["Category"] = data["Category"].replace(category_mapping)
review["Category"] = review["Category"].replace(category_mapping)
apple["Category"] = apple["Category"].replace(category_mapping)

print(review["Category"].unique())
```

```
['art and design' 'auto and vehicles' 'lifestyle' 'books and reference'
 'business' 'comics' 'social networking' 'education' 'entertainment'
 'finance' 'food and drink' 'health and fitness' 'house and home'
 'libraries and demo' 'games' 'family' 'medical' 'shopping'
 'photo & video' 'sports' 'travel and local' 'utilities' 'personalization'
 'productivity' 'weather' 'video players' 'news and magazines'
 'maps and navigation']
```

In [49]:
```
review = review.drop(columns=['Unnamed: 0', 'App', 'Rating', 'Reviews', 'Size',
       'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated',
       'Current Ver', 'Android Ver'])
```

In [50]:
```
review.columns
```

Out[50]:
```
Index(['Category', 'Translated_Review', 'Sentiment', 'Sentiment_Polarity',
       'Sentiment_Subjectivity'],
      dtype='object')
```

In [51]:
```
apple.columns
```

Out[51]:
```
Index(['App_Name', 'Category', 'Content_Rating', 'Size_Bytes', 'Released',
       'Updated', 'Price', 'Currency', 'Free', 'Developer',
       'Average_User_Rating', 'Reviews', 'Apple_App_Age'],
      dtype='object')
```

In [52]:
```
apple = apple.drop(columns=["App_Name"])
print(apple.columns)
```

```
Index(['Category', 'Content_Rating', 'Size_Bytes', 'Released', 'Updated',
       'Price', 'Currency', 'Free', 'Developer', 'Average_User_Rating',
       'Reviews', 'Apple_App_Age'],
      dtype='object')
```

In [53]:
```
# Rename Google Play Store User Reviews columns with a prefix
review = review.add_prefix("GoogleReview_")
# Rename Apple App Store dataset columns with a prefix
apple = apple.add_prefix("Apple_")
# Ensure 'Category' remains the same in Apple dataset for merging
apple.rename(columns={"Apple_Category": "Category"}, inplace=True)
review.rename(columns={"GoogleReview_Category": "Category"}, inplace=True)
```

FINAL OUTPUT

In [55]:
```
review.head(2)
```

Out[55]:

	Category	GoogleReview_Translated_Review	GoogleReview_Sentiment	GoogleReview_Sen
0	art and design	A kid's excessive ads. The types ads allowed a...	Negative	
1	art and design	It bad >:(	Negative	

SAVING THE DATASETS

In [57]:
```
data.to_csv('cleaned_googleplaystore.csv', index=False)
apple.to_csv('cleaned_appstore.csv', index=False)
review.to_csv('playstorereviews.csv', index=False)
print('Successfully saved!')
```

```
Successfully saved!
```

In [105…
```
data.shape
```

Out[105…
```
(2312944, 19)
```

In [107…
```
apple.shape
```

Out[107…
```
(1230376, 12)
```

In [109…
```
review.shape
```

Out[109…
```
(48144, 5)
```

In [ ]:
```

```