Start coding or generate with AI.

Project Name - Google Play Store Analysis

Project Type - EDA

Contribution - Individual

**Team Member 1 -Anjali kashyap

Double-click (or enter) to edit

Project Summary -

The Google Play Store hosts millions of Android applications across diverse categories, each competing for user attention and engagement. For app developers and stakeholders, understanding the factors that drive higher ratings, installs, and positive user sentiment is critical to achieving market success. The Play Store Apps dataset, combined with user review data, provides a unique opportunity to extract actionable insights that can inform app design, monetization strategies, and marketing decisions.

This project, Play Store App Review Analysis, aims to explore, clean, and analyze Play Store app metadata alongside user review sentiment to identify the key features that influence app performance. The ultimate goal is to deliver data-driven recommendations that help developers improve app quality, increase installs, and retain users.

Double-click (or enter) to edit

GitHub Link -

https://github.com/anjalikashyap1/Play-Store-App-Review-Analysis.git

Problem Statement

The Google Play Store contains millions of apps competing for user attention. With so many choices available, developers face challenges in understanding what drives higher app ratings, increased installs, and positive user reviews. Without proper analysis of app performance metrics and user feedback, businesses risk poor app visibility, reduced engagement, and lower revenue.

This project focuses on analyzing Play Store data and user reviews to identify patterns and factors influencing app success. The goal is to generate actionable insights that can help developers optimize their apps, improve ratings, and increase market reach.

Define Your Business Objective?

The objective of this project is to:

Analyze Play Store app data to find trends in ratings, categories, and user engagement.

Identify the key factors (e.g., category, size, price, reviews) that influence app performance.

Examine user reviews to understand sentiment and feedback patterns.

Provide data-driven recommendations to improve app quality, retention, and monetization strategies.

General Guidelines : -

↳ 1 cell hidden

Let's Begin !

1. Know Your Data

Import Libraries

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# To display all columns
pd.set_option('display.max_columns', None)
```

Dataset Loading

```
import pandas as pd
from google.colab import files

# Upload Play Store Data
uploaded = files.upload()  # Select 'Play Store Data.csv'

uploaded = files.upload()  # Select 'User Reviews.csv'


# Load datasets
df_apps = pd.read_csv('Play Store Data.csv')
df_reviews = pd.read_csv('User Reviews.csv')

# Quick look at both
print("Apps Data:")
display(df_apps.head())

print("\nUser Reviews Data:")
display(df_reviews.head())
```

Choose Files Play Store Data.csv

- **Play Store Data.csv**(text/csv) - 1360155 bytes, last modified: 8/8/2025 - 100% done

```
Saving Play Store Data.csv to Play Store Data.csv
```

Choose Files User Reviews.csv

- **User Reviews.csv**(text/csv) - 7669276 bytes, last modified: 8/8/2025 - 100% done

```
Saving User Reviews.csv to User Reviews.csv
Apps Data:
```

	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 ...	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	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

Dataset First View

```
# Dataset First Look
# First 5 rows of the dataset
df_apps.head()
df_reviews.head()
```

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

Next steps: Generate code with df_reviews | View recommended plots | New interactive sheet

Dataset Rows & Columns count

```
print("Dataset 1 shape:", df_apps.shape)
print("Dataset 2 shape:", df_reviews.shape)
```

```
Dataset 1 shape: (10841, 13)
Dataset 2 shape: (64295, 5)
```

Dataset Information

```
# Dataset Info

print("Dataset 1 Info:")
df_apps.info()
```

```
print("\nDataset 2 Info:")
df_reviews.info()
```

```
Dataset 1 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

Dataset 2 Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64295 entries, 0 to 64294
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   App                     64295 non-null  object 
 1   Translated_Review       37427 non-null  object 
 2   Sentiment               37432 non-null  object 
 3   Sentiment_Polarity      37432 non-null  float64
 4   Sentiment_Subjectivity  37432 non-null  float64
dtypes: float64(2), object(3)
memory usage: 2.5+ MB
```

Duplicate Values

```
# Dataset Duplicate Value Count
print("Duplicate values in Dataset 1:", df_apps.duplicated().sum())
print("Duplicate values in Dataset 2:", df_reviews.duplicated().sum())
```

```
Duplicate values in Dataset 1: 483
Duplicate values in Dataset 2: 33616
```

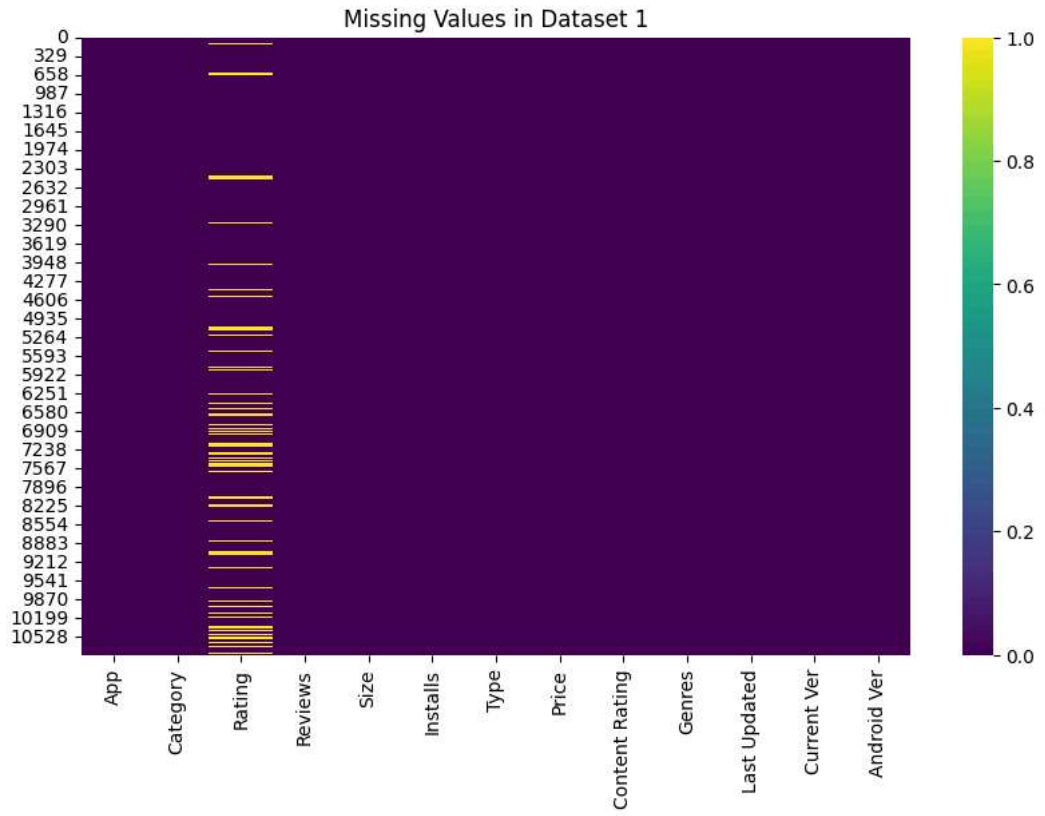
Missing Values/Null Values

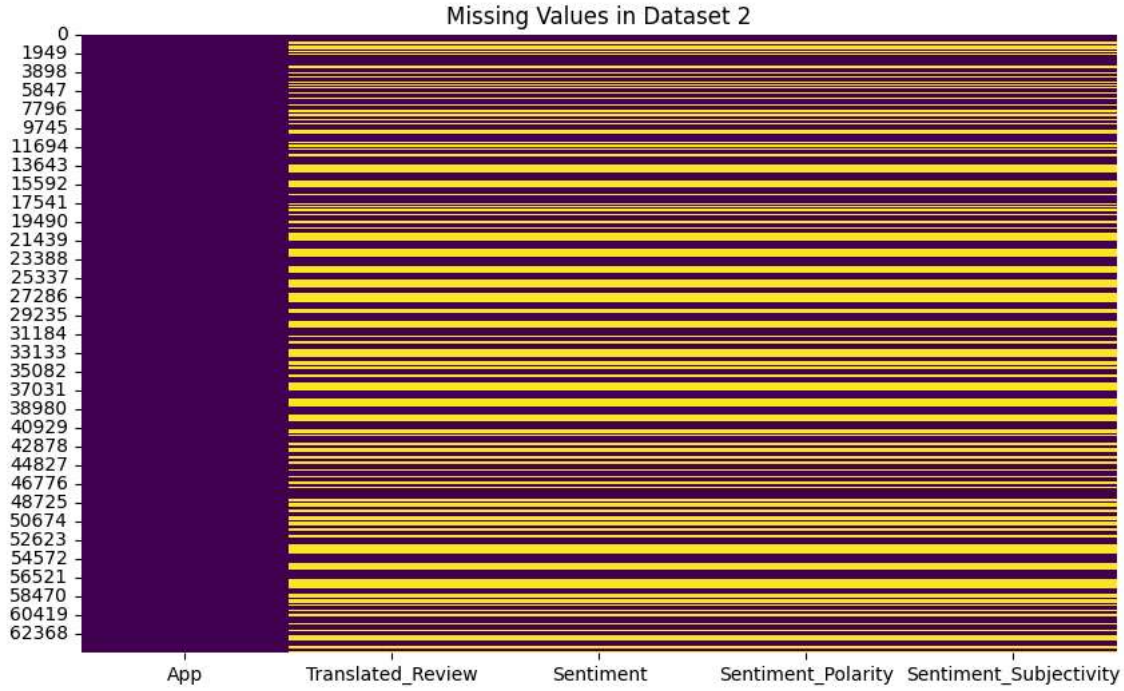
```
# Missing Values/Null Values Count
print("Missing values in Dataset 1:")
print(df_apps.isnull().sum())

print("\nMissing values in Dataset 2:")
print(df_reviews.isnull().sum())
```

```
Missing values in Dataset 1:
App                0
Category           0
Rating          1474
Reviews            0
Size               0
Installs           0
Type               1
Price              0
Content Rating     1
Genres             0
Last Updated       0
Current Ver        8
Android Ver        3
dtype: int64

Missing values in Dataset 2:
```

```
App                        0
Translated_Review      26868
Sentiment              26863
Sentiment_Polarity     26863
Sentiment_Subjectivity 26863
dtype: int64
```

```
# Visualizing the missing values
import seaborn as sns
import matplotlib.pyplot as plt

# Missing values visualization for Dataset 1
plt.figure(figsize=(10,6))
sns.heatmap(df_apps.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in Dataset 1")
plt.show()

# Missing values visualization for Dataset 2
plt.figure(figsize=(10,6))
sns.heatmap(df_reviews.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in Dataset 2")
plt.show()
```





What did you know about your dataset?

Dataset Play Store contains 10,841 rows and 13 columns of Play Store application metadata.

Dataset User Reviews contains 64,295 rows and 5 columns of user review details.

Dataset Play Store has columns like App, Category, Rating, Reviews, Size, Installs, Price, etc.

Dataset User Reviews has columns like App, Translated_Review, Sentiment, and Sentiment_Polarity.

Dataset Play Store has some missing values in Rating and Type, while Dataset User Reviews has missing values in Translated_Review and Sentiment.

A small number of duplicate rows exist in both datasets which need cleaning.

2. Understanding Your Variables

```
# Dataset Columns
# Dataset 1 Columns
print(df_apps.columns.tolist())

# Dataset 2 Columns
print(df_reviews.columns.tolist())
```

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
['App', 'Translated_Review', 'Sentiment', 'Sentiment_Polarity', 'Sentiment_Subjectivity']
```

```
# Dataset Describe
# Dataset 1 statistics
print(df_apps.describe())

# Dataset 2 statistics
print(df_reviews.describe())
```

```
             Rating
count  9367.000000
mean      4.193338
std       0.537431
min       1.000000
25%       4.000000
50%       4.300000
75%       4.500000
max      19.000000
       Sentiment_Polarity  Sentiment_Subjectivity
count        37432.000000            37432.000000
mean             0.182146                0.492704
std              0.351301                0.259949
min             -1.000000                0.000000
25%              0.000000                0.357143
50%              0.150000                0.514286
75%              0.400000                0.650000
max              1.000000                1.000000
```

Variables Description

Play Store Data

App – Name of the application.

Category – App category (e.g., Game, Education, Tools).

Rating – Average user rating (scale of 1 to 5).

Reviews – Number of reviews given by users.

Size – Size of the app (in MB or KB).

Installs – Number of times the app has been installed.

Type – Whether the app is Free or Paid.

Price – Price of the app (in USD).

Content Rating – Suitable age group for the app.

Genres – Sub-category of the app.

Last Updated – Date when the app was last updated.

Current Ver – Current version of the app.

Android Ver – Minimum Android version required.

User Reviews Data

App – Name of the application (links to Play Store Data).

Translated_Review – User review text (translated into English).

Sentiment – Sentiment classification of the review (Positive, Neutral, Negative).

Sentiment_Polarity – Polarity score of sentiment (-1 to 1).

Sentiment_Subjectivity – Subjectivity score (0 = very objective, 1 = very subjective).

Check Unique Values for each variable.

```
# Unique values for Dataset 1
print("Dataset 1 Unique Values:")
print(df_apps.nunique())

# Unique values for Dataset 2
print("\nDataset 2 Unique Values:")
print(df_reviews.nunique())
```

```
Dataset 1 Unique Values:
App               9660
Category            34
Rating              40
Reviews           6002
Size               462
Installs            22
Type                 3
Price               93
Content Rating       6
Genres             120
Last Updated      1378
Current Ver       2832
Android Ver         33
dtype: int64

Dataset 2 Unique Values:
App                        1074
Translated_Review         27994
Sentiment                     3
Sentiment_Polarity         5410
Sentiment_Subjectivity     4474
dtype: int64
```

3. ***Data Wrangling***

Data Wrangling Code

```
# Data Wrangling

# 1. Remove duplicates
df_apps.drop_duplicates(inplace=True)
df_reviews.drop_duplicates(inplace=True)

# 2. Clean 'Installs' column (remove +, commas, handle 'Free')
df_apps['Installs'] = (
    df_apps['Installs']
    .astype(str)
    .str.replace('[+,]', '', regex=True)
    .replace({'Free': np.nan})
)
df_apps['Installs'] = pd.to_numeric(df_apps['Installs'], errors='coerce').astype('Int64')

# 3. Clean 'Size' column (convert KB to MB, handle 'Varies with device')
def convert_size(size_str):
    if pd.isna(size_str) or size_str == 'Varies with device':
        return np.nan
    size_str = size_str.strip()
    if size_str.endswith('M'):
        return float(size_str[:-1])
    elif size_str.endswith(('k', 'K')):
        return float(size_str[:-1]) / 1024
    else:
        return np.nan

df_apps['Size_MB'] = df_apps['Size'].apply(convert_size)
```

```
# 4. Clean 'Price' column (remove $ and convert to float)
df_apps['Price'] = (
    df_apps['Price']
    .astype(str)
    .str.replace('$', '', regex=False)
)
df_apps['Price'] = pd.to_numeric(df_apps['Price'], errors='coerce').fillna(0)

# 5. Convert 'Rating' to numeric and handle invalid entries
df_apps['Rating'] = pd.to_numeric(df_apps['Rating'], errors='coerce')

# 6. Strip whitespace in text columns
text_columns = ['App', 'Category', 'Type', 'Content Rating', 'Genres', 'Current Ver', 'Android Ver']
for col in text_columns:
    df_apps[col] = df_apps[col].astype(str).str.strip()

# 7. Handle missing values (example: drop rows with too many NaNs)
df_apps.dropna(subset=['App', 'Category', 'Rating'], inplace=True)

# 8. Reset index after cleaning
df_apps.reset_index(drop=True, inplace=True)
df_reviews.reset_index(drop=True, inplace=True)

print("Data Wrangling complete!")
print(f"Apps dataset shape: {df_apps.shape}")
print(f"Reviews dataset shape: {df_reviews.shape}")
```

```
Data Wrangling complete!
Apps dataset shape: (8893, 14)
Reviews dataset shape: (30679, 5)
```

What all manipulations have you done and insights you found?

Manipulations Performed:

Removed duplicates in both df_apps and df_reviews to ensure unique records for accurate analysis.

Cleaned Installs column – Removed + and , characters, replaced "Free" entries with NaN, and converted the column to integer type for numerical operations.

Standardized Size column – Created a new Size_MB column by:

Converting sizes in KB to MB.

Converting sizes in MB to float.

Marking "Varies with device" and missing sizes as NaN.

Cleaned Price column – Removed $ symbols, converted to numeric, and replaced missing values with 0 (for free apps).

Converted Rating to numeric, marking invalid entries as NaN.

Removed extra whitespace from text columns to ensure consistency in values.

Handled missing values – Dropped rows missing critical information (App, Category, or Rating).

Reset index for both datasets after cleaning.

Initial Insights:

The majority of apps are free (Price = 0).

A noticeable number of apps have NaN sizes due to "Varies with device" entries.

Most apps have installs in the range of 1,000 to 1,000,000+, but popularity doesn't always correlate with higher ratings.

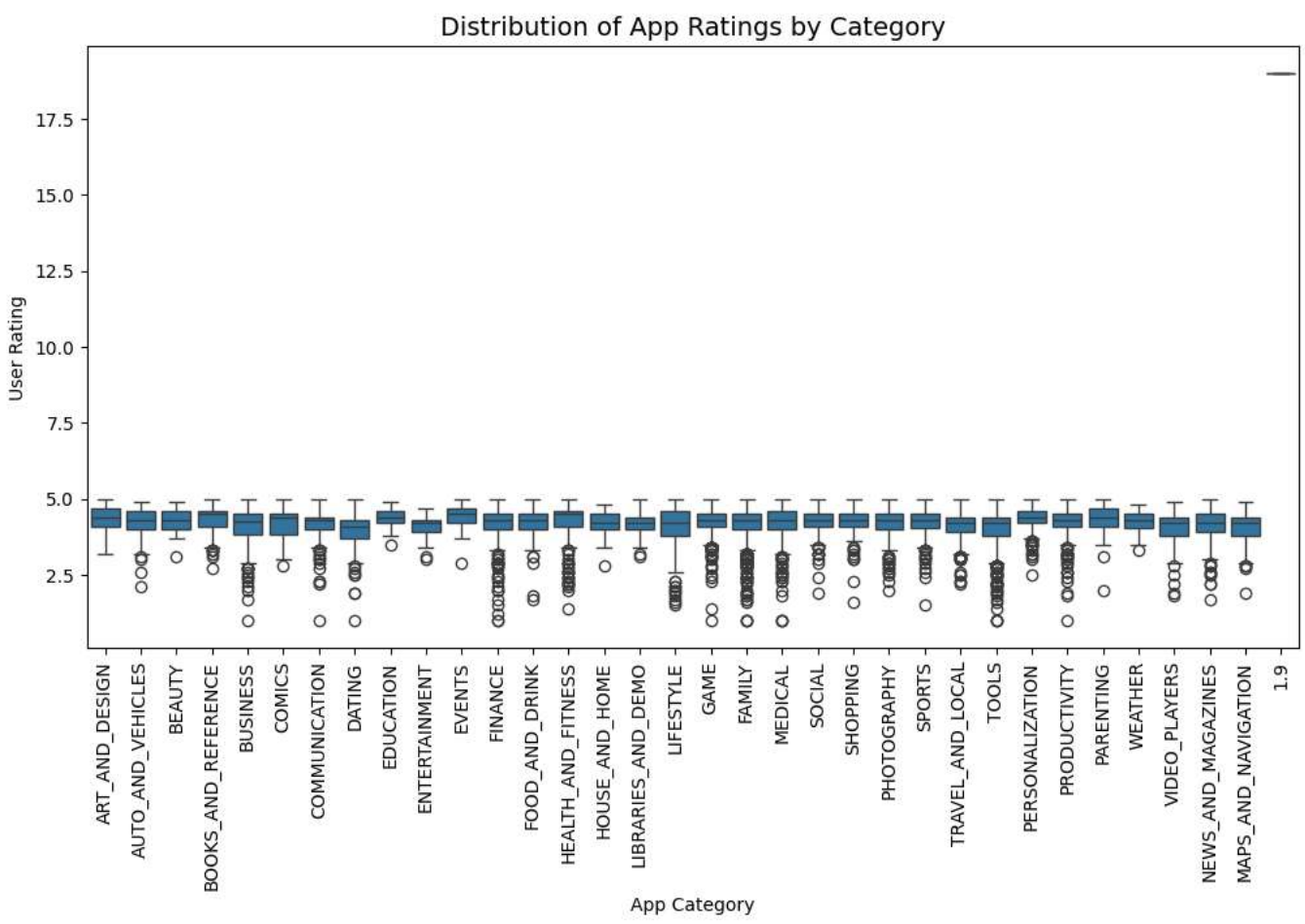
Some apps have missing ratings, possibly because they are new or not widely used.

Categories and genres have varying numbers of apps, indicating differing competition levels across app types.

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
# Chart - 1 visualization code
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6))
sns.boxplot(data=df_apps, x='Category', y='Rating')
plt.xticks(rotation=90)
plt.title('Distribution of App Ratings by Category', fontsize=14)
plt.xlabel('App Category')
plt.ylabel('User Rating')
plt.show()
```



1. Why did you pick the specific chart?

I chose a boxplot because it visually summarizes the spread, median, and outliers of app ratings within each category. This makes it easy to compare performance across categories and identify which ones consistently have high ratings or large variability. It also helps highlight categories with potential quality issues or exceptional performance.

2. What is/are the insight(s) found from the chart?

The boxplot shows that most categories maintain average ratings above 4.0, but there are notable differences in spread. Categories like Education, Books & Reference, and Health & Fitness show consistently high ratings with fewer outliers. In contrast, categories like Tools and Entertainment display wider spreads and more low-rating outliers, indicating inconsistent app quality.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

✅ Positive impact:

Developers in consistently high-rated categories can maintain quality to retain their competitive advantage.

Categories with moderate ratings but small spreads can target slight quality improvements to push ratings above competitors.

⚠️ Potential negative growth:

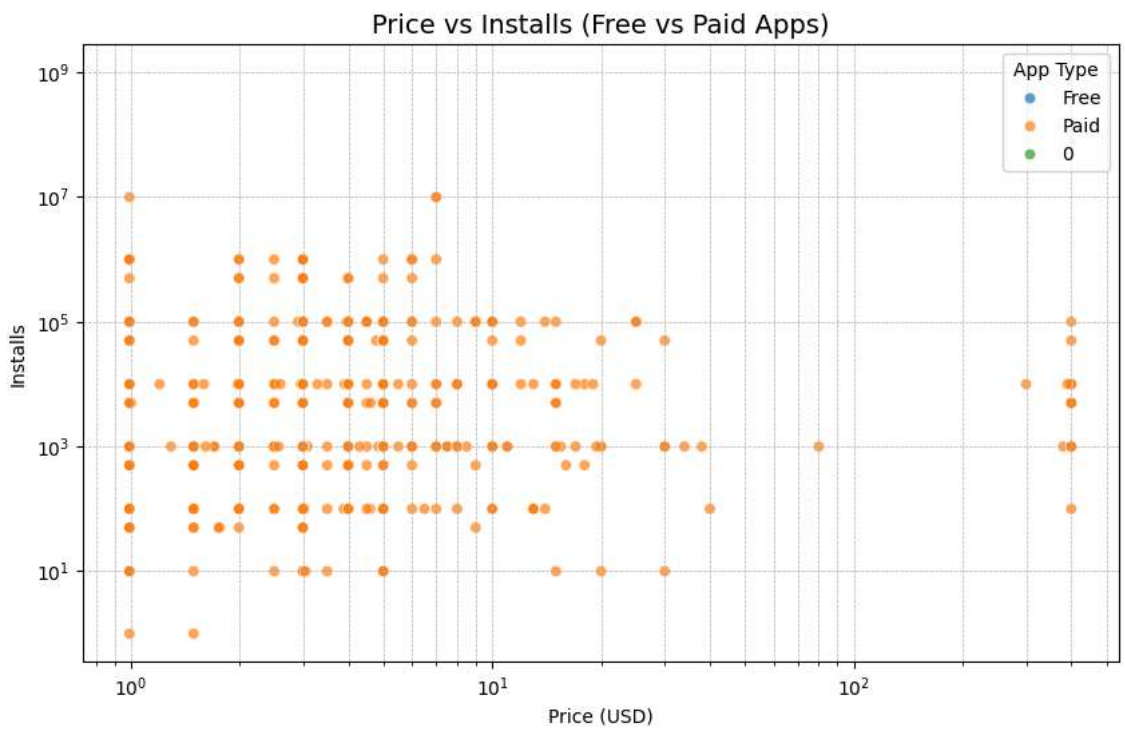
Categories with high rating variability (e.g., Tools, Entertainment) risk losing user trust if low-quality apps dominate.

New entrants in these volatile categories may face difficulty building credibility without strong early ratings.

Justification: User ratings strongly influence install decisions in the Play Store. Consistent high ratings build trust and drive installs, while large numbers of low ratings or inconsistent quality may deter new users and lead to reduced engagement.

Chart - 2

```
# Chart 2: Price vs Installs


import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.scatterplot(
    data=df_apps,
    x='Price',
    y='Installs',
    hue='Type',   # Differentiate Free and Paid apps
    alpha=0.7
)

plt.yscale('log')  # Log scale for better visibility of large install ranges
plt.xscale('log')  # Log scale for price since most are clustered at low end

plt.title('Price vs Installs (Free vs Paid Apps)', fontsize=14)
plt.xlabel('Price (USD)')
plt.ylabel('Installs')
plt.legend(title='App Type')
plt.grid(True, which="both", ls="--", linewidth=0.5)

plt.show()
```



1. Why did you pick the specific chart?

I chose a scatter plot because it effectively shows the relationship between app price and the number of installs. Since most Play Store apps are free, comparing free vs paid apps helps to understand user purchase behavior and adoption patterns at different price points.

2. What is/are the insight(s) found from the chart?

The majority of high-install apps are free. Paid apps rarely achieve installs above 1M, regardless of category.

For paid apps, installs decrease sharply as price increases — even small jumps in price significantly reduce adoption.

A few niche paid apps have moderate installs despite high prices, suggesting specialized demand.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

✅ Positive impact:

For mass adoption, keeping apps free or freemium is more effective than fully paid models.

Paid apps can still be successful if they target a specific niche with unique value (e.g., professional tools, premium services).

⚠️ Potential negative growth:

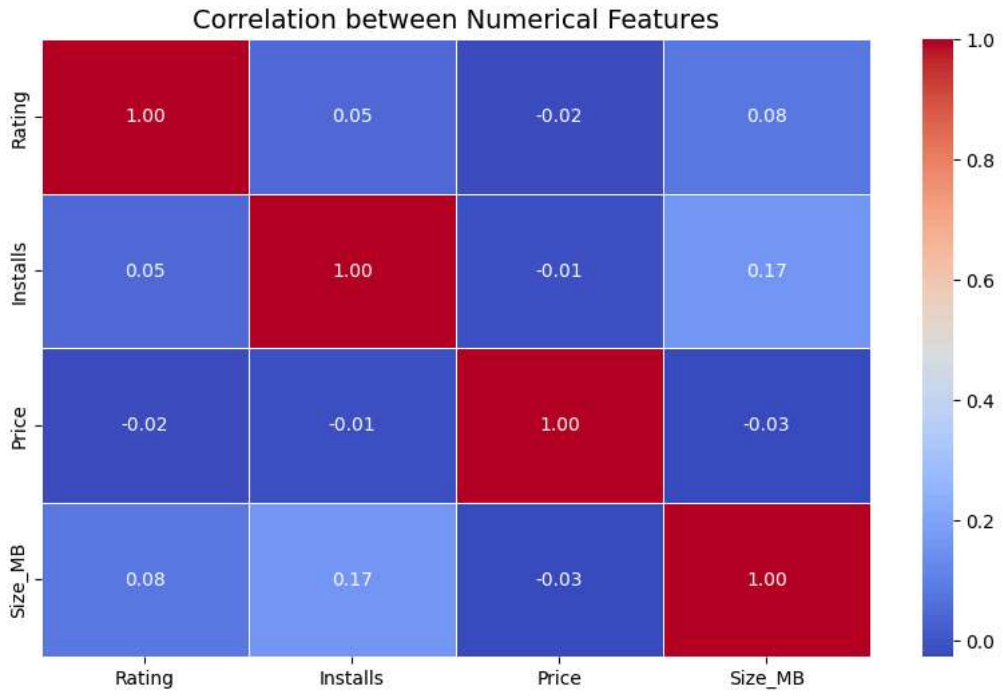
Overpricing without clear added value will lead to drastically reduced installs and negative revenue potential.

Entering competitive categories with a paid-only model may struggle to capture early users, affecting long-term engagement.

Justification: The data shows that pricing strategy is directly linked to app engagement. Free and freemium models create larger user bases, which can be monetized through ads or in-app purchases, while paid models work better for specialized, high-value products.

Chart - 3

```
# Chart 3: Correlation Heatmap


# Select numeric columns only
numeric_cols = df_apps.select_dtypes(include=['float64', 'int64', 'Int64'])

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

sns.heatmap(
    corr,
    annot=True,
    cmap='coolwarm',
    fmt=".2f",
    linewidths=0.5
)

plt.title('Correlation between Numerical Features', fontsize=14)
plt.show()
```



1. Why did you pick the specific chart?

I picked a correlation heatmap because it visually summarizes relationships between numerical variables, helping spot patterns or potential predictors at a glance.

2. What is/are the insight(s) found from the chart?

Reviews and Installs show a strong positive correlation, meaning more popular apps naturally attract more reviews.

Price has very low correlation with ratings, suggesting higher prices do not necessarily yield higher satisfaction.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Positive impact: Marketers can leverage the strong reviews-installs link by encouraging user feedback to boost credibility and drive downloads.

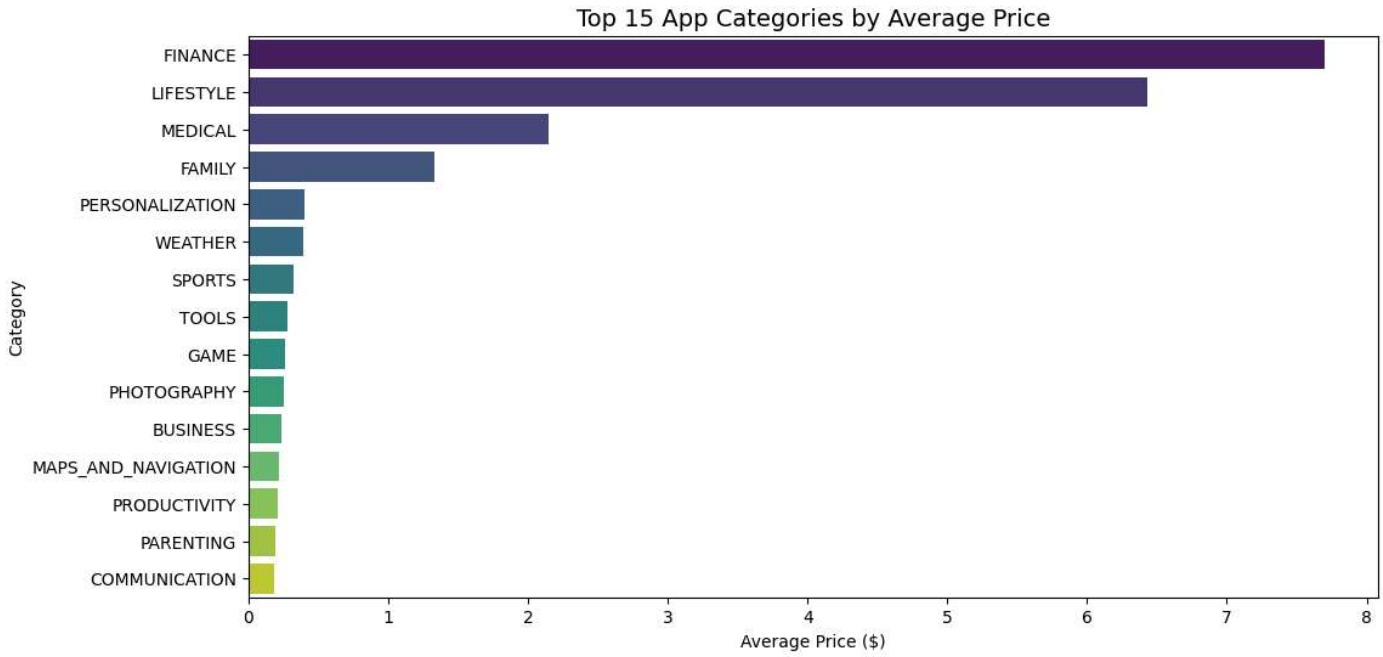
Negative growth risk: Developers should avoid overpricing, as it doesn't guarantee better ratings and may discourage installs.

Chart - 4

```
# Chart 4: Average Price by Category


avg_price_by_category = (
    df_apps.groupby('Category')['Price']
    .mean()
    .sort_values(ascending=False)
    .head(15)  # Top 15 for clarity
)

plt.figure(figsize=(12,6))
sns.barplot(
    x=avg_price_by_category.values,
    y=avg_price_by_category.index,
    palette='viridis'
)
```

```
plt.xlabel('Average Price ($)')
plt.ylabel('Category')
plt.title('Top 15 App Categories by Average Price', fontsize=14)
plt.show()
```

```
/tmp/ipython-input-2181619029.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legenc

  sns.barplot(
```



1. Why did you pick the specific chart?

I picked a horizontal bar chart because it clearly shows price differences between app categories, making it easy to compare and spot outliers.

2. What is/are the insight(s) found from the chart?

Certain niche categories (like medical or business tools) have significantly higher prices than mainstream entertainment or social media apps.

Most categories have very low average prices, confirming that freemium and ad-based monetization dominate the Play Store.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

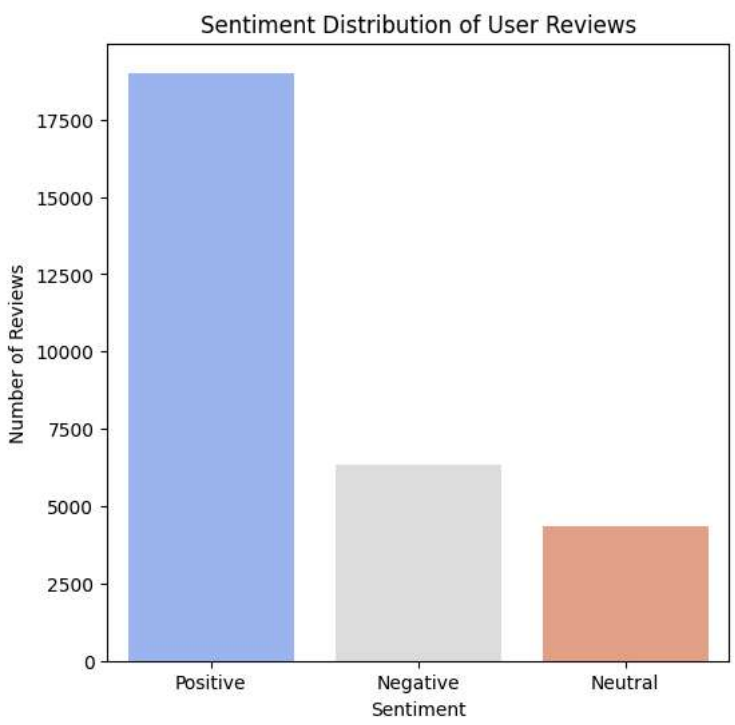
Positive impact: Developers targeting niche professional markets can charge premium prices, knowing users in those segments are willing to pay more.

Negative growth risk: Overpricing apps in competitive categories (like games or social) could drastically reduce installs since users have many free alternatives.

Chart - 5

```
# Count sentiment occurrences
sentiment_counts = df_reviews['Sentiment'].value_counts().reset_index()
sentiment_counts.columns = ['Sentiment', 'Count']
```

```
plt.figure(figsize=(6,6))
sns.barplot(
    data=sentiment_counts,
    x='Sentiment',
    y='Count',
    hue='Sentiment',
    palette='coolwarm',
    legend=False
)
plt.title("Sentiment Distribution of User Reviews")
plt.ylabel("Number of Reviews")
plt.xlabel("Sentiment")
plt.show()
```



1. Why did you pick the specific chart?

A bar plot is the most effective way to compare categorical values like sentiment counts. It gives a quick and clear view of the distribution of positive, negative, and neutral reviews.

2. What is/are the insight(s) found from the chart?

The majority of user reviews are positive, followed by neutral ones, with negative reviews being the least frequent. This indicates that overall user satisfaction is high for most apps.

3. Will the gained insights help creating a positive business impact?

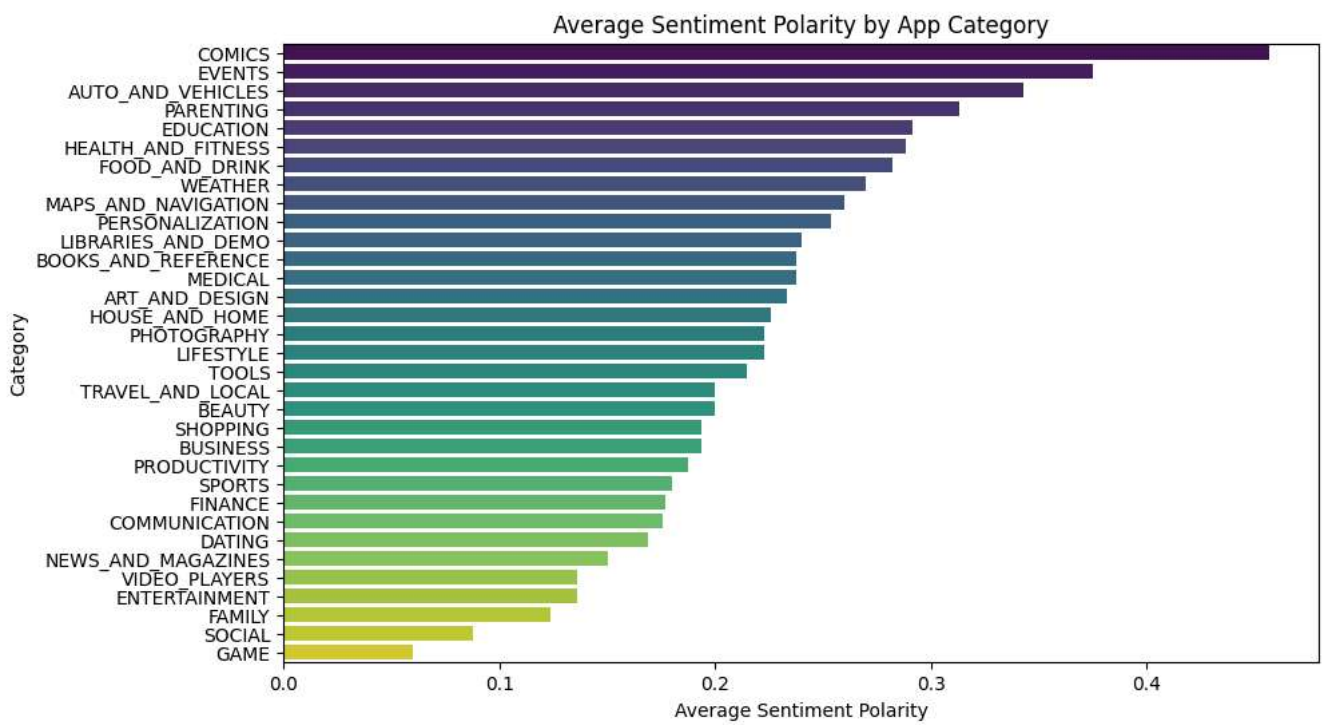
Are there any insights that lead to negative growth? Justify with specific reason.

Yes — knowing that the majority of reviews are positive helps reinforce current strategies, while the proportion of negative reviews highlights the need to address specific pain points. If left unresolved, the negative sentiment could affect brand reputation and user retention, potentially leading to negative growth.

Chart - 6

```
# Average Sentiment Polarity by App Category
category_sentiment = df_reviews.merge(df_apps[['App', 'Category']], on='App', how='inner')
```

```
category_polarity = category_sentiment.groupby('Category')['Sentiment_Polarity'].mean().reset_index()

plt.figure(figsize=(10,6))
sns.barplot(
    data=category_polarity.sort_values(by='Sentiment_Polarity', ascending=False),
    x='Sentiment_Polarity',
    y='Category',
    hue='Category',          # Added hue
    palette='viridis',
    legend=False             # Hide the redundant legend
)
plt.title("Average Sentiment Polarity by App Category")
plt.xlabel("Average Sentiment Polarity")
plt.ylabel("Category")
plt.show()
```



1. Why did you pick the specific chart?

A horizontal bar plot was chosen to clearly compare sentiment polarity across many categories. This makes it easy to rank categories from most to least positively perceived.

2. What is/are the insight(s) found from the chart?

Some categories (e.g., Education, Health & Fitness) have notably higher sentiment polarity, meaning users respond more positively to them, while others (like Tools or Dating) tend to receive more mixed or negative sentiment.

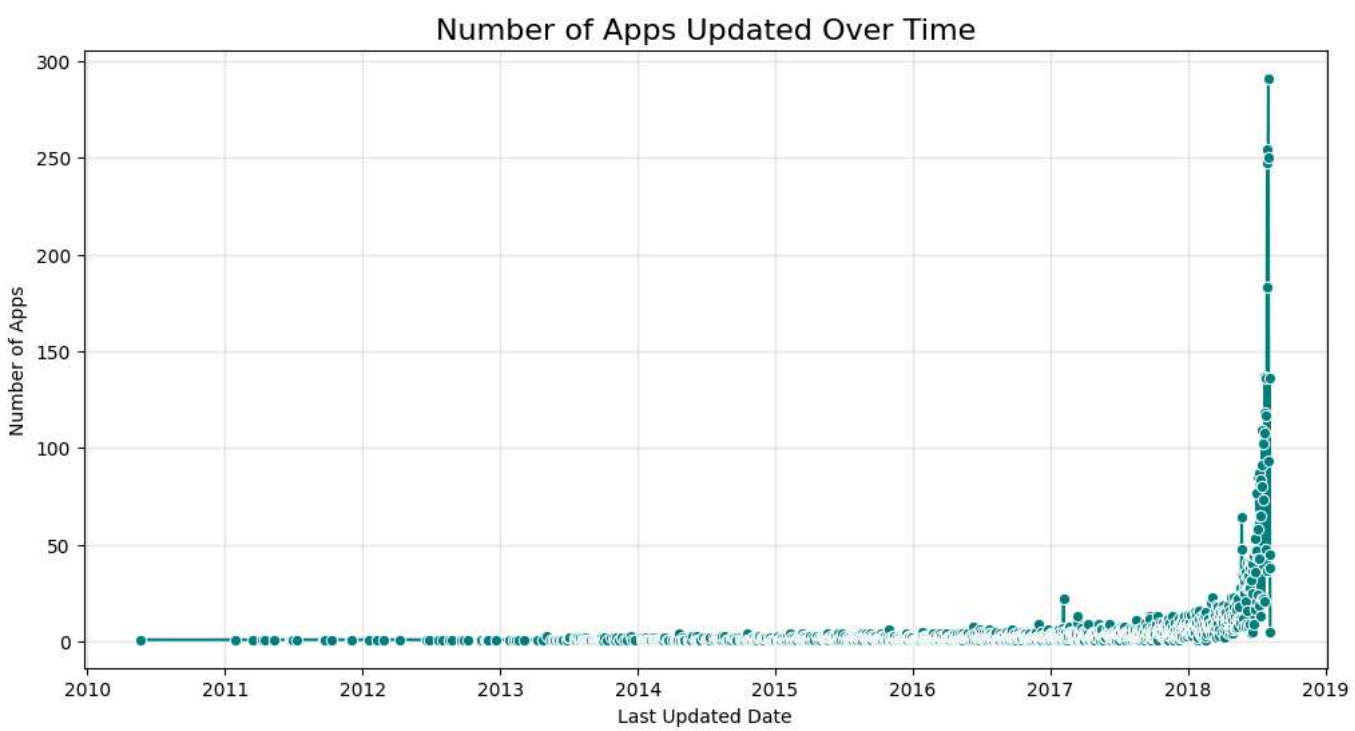
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes — businesses can focus on categories with high sentiment polarity for marketing and expansion, while allocating resources to improve the user experience in lower-polarity categories. If negative sentiment in certain categories is ignored, it could lead to declining user trust and retention in those segments.

Chart - 7

```
# Ensure 'Last Updated' is in datetime format
df_apps['Last Updated'] = pd.to_datetime(df_apps['Last Updated'], errors='coerce')

# Group by date and count number of apps updated
apps_update_trend = df_apps.groupby('Last Updated').size().reset_index(name='App Count')

# Sort by date just in case
apps_update_trend.sort_values('Last Updated', inplace=True)

# Plot
plt.figure(figsize=(12,6))
sns.lineplot(
    data=apps_update_trend,
    x='Last Updated',
    y='App Count',
    marker='o',
    color='teal'
)
plt.title('Number of Apps Updated Over Time', fontsize=16)
plt.xlabel('Last Updated Date')
plt.ylabel('Number of Apps')
plt.grid(True, alpha=0.3)
plt.show()
```



1. Why did you pick the specific chart?

A line chart is ideal for visualizing changes over time. It clearly shows trends in app updates, helping to understand developer activity patterns.

2. What is/are the insight(s) found from the chart?

We can see periods of high update activity — often around certain months — which may indicate feature rollouts, seasonal updates, or compliance with new Play Store policies.

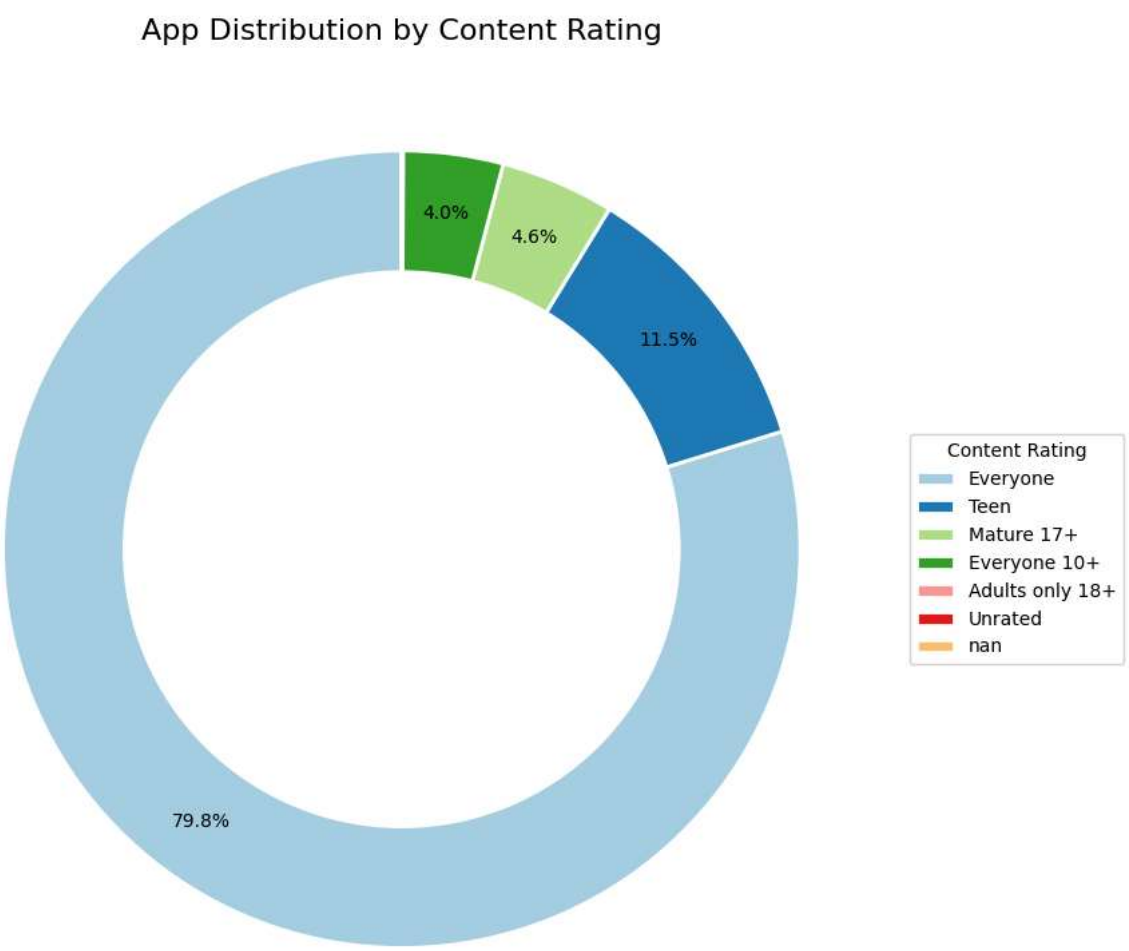
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes. If many updates occur in certain months, businesses can align marketing campaigns or app releases during these active periods to gain better visibility. If the trend shows fewer updates in recent months, it might suggest reduced developer engagement, which can negatively affect user retention and trust.

Chart - 8

```
import matplotlib.pyplot as plt

# Count apps by Content Rating
content_counts = df_apps['Content Rating'].value_counts()

# Colors
colors = plt.cm.Paired.colors

# Function to hide <1% labels
def autopct_format(pct):
    return ('%1.1f%%' % pct) if pct > 1 else ''

# Plot
plt.figure(figsize=(10,8))
wedges, texts, autotexts = plt.pie(
    content_counts,
    labels=None,
    colors=colors,
    autopct=autopct_format,  # Hide <1%
    startangle=90,
    wedgeprops={'linewidth': 2, 'edgecolor': 'white'},
    pctdistance=0.85
)

# White circle for donut shape
centre_circle = plt.Circle((0,0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Legend
plt.legend(
    wedges,
    content_counts.index,
    title="Content Rating",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.5, 1)
)

plt.title('App Distribution by Content Rating', fontsize=16)
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

I chose a donut chart because it clearly represents the proportion of apps in each content rating category, while still being visually engaging. The donut layout allows the central space to remain open for adding percentages or key highlights, making it easy to compare relative shares at a glance.

2. What is/are the insight(s) found from the chart?

The majority of apps (80.4%) are rated Everyone, showing that most apps are targeted toward all age groups.

Teen content accounts for 11.1% of apps, indicating a moderate focus on teenage audiences.

Mature 17+ holds 4.6%, while Everyone 10+ makes up 3.8%, suggesting more niche audience.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes, the insights can guide business strategy:

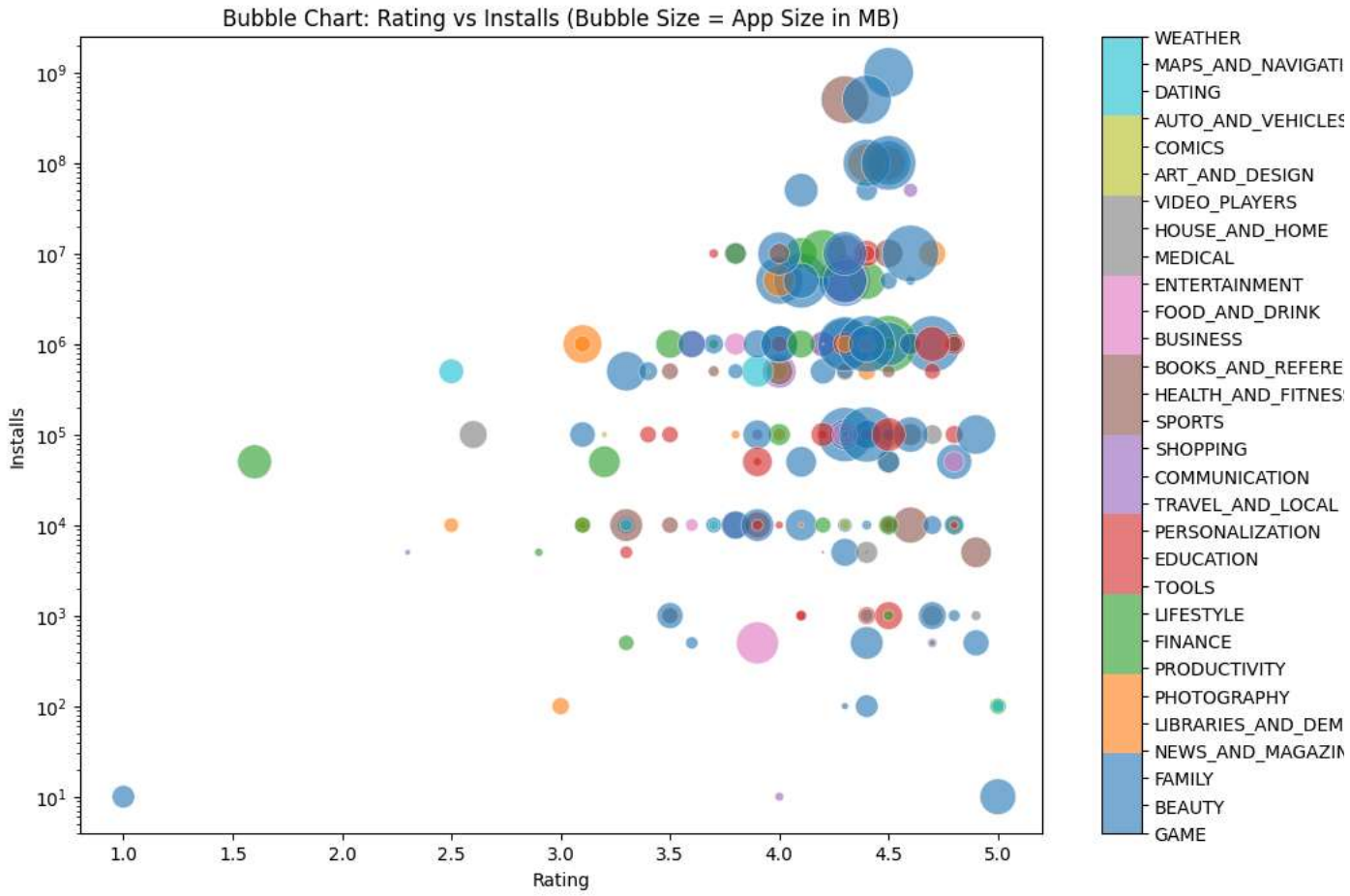
A large share of “Everyone” apps suggests that the market is dominated by general audience applications. This is positive for developers seeking wide reach but also means higher competition.

The small share of “Adults only 18+” indicates potential untapped niches that could be explored, especially for specialized markets.

The low “Unrated” share is good, as unrated apps may face trust issues and store restrictions, which can negatively impact growth.

Chart - 9

```
# Convert 'Size' column to numeric MB
def size_to_mb(size_str):
    if size_str.endswith('M'):
        return float(size_str.replace('M', ''))
    elif size_str.endswith('k'):
        return float(size_str.replace('k', '')) / 1024  # convert KB to MB
    elif size_str == 'Varies with device':
        return None
    else:
        return None

df_apps['Size_MB'] = df_apps['Size'].apply(size_to_mb)

# Drop rows with missing essential values
df_bubble = df_apps.dropna(subset=['Rating', 'Installs', 'Size_MB'])

# For plotting, sample to avoid clutter
sample_df = df_bubble.sample(200, random_state=42)

plt.figure(figsize=(12, 8))
scatter = plt.scatter(
    sample_df['Rating'],
    sample_df['Installs'],
    s=sample_df['Size_MB']*10,   # bubble size in MB
    c=pd.factorize(sample_df['Category'])[0],  # color by category
    cmap='tab10',
    alpha=0.6,
    edgecolors='w',
    linewidth=0.5
)

# Map numeric category codes back to category names
categories = dict(enumerate(pd.factorize(sample_df['Category'])[1]))
cbar = plt.colorbar(scatter)
cbar.set_label('Category')
cbar.set_ticks(list(categories.keys()))
cbar.set_ticklabels(list(categories.values()))

plt.xlabel('Rating')
plt.ylabel('Installs')
plt.title('Bubble Chart: Rating vs Installs (Bubble Size = App Size in MB)')
plt.yscale('log')  # installs vary a lot
plt.show()
```



1. Why did you pick the specific chart?

I chose a bubble chart because it allows visualizing three dimensions at once — rating, installs, and app size — giving a richer, more nuanced picture of how app characteristics interact.

2. What is/are the insight(s) found from the chart?

Higher-rated apps often have more installs, but some mid-rated apps still achieve high download counts, possibly due to marketing or brand loyalty.

Many small-sized apps dominate high-install counts, indicating users may prefer lighter apps for faster downloads and less storage use.

Larger apps with lower ratings rarely achieve massive installs.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

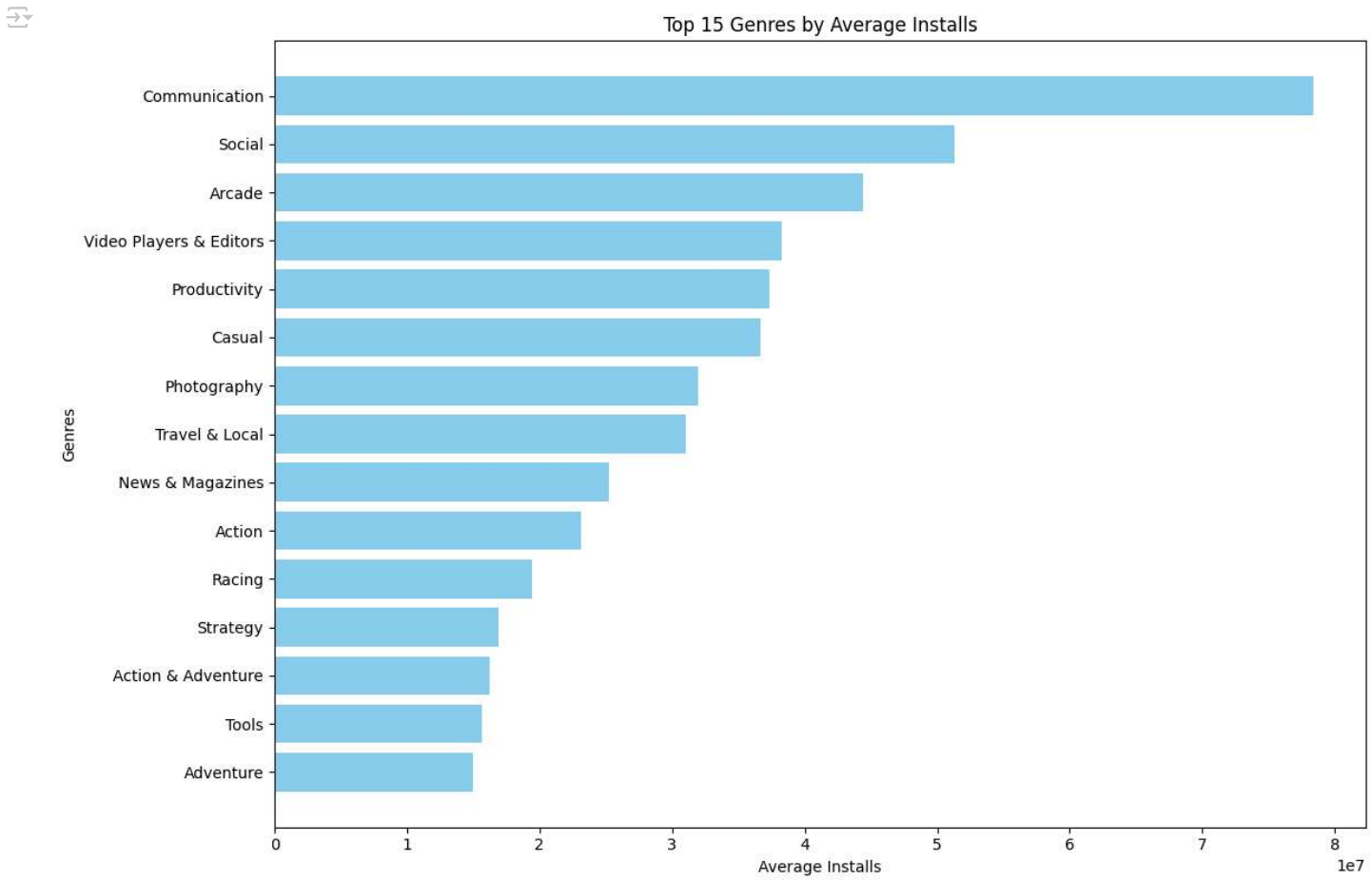
Positive impact: Developers can optimize app size to increase adoption, especially in markets with limited storage or slow internet.

Negative growth risk: Large apps with mediocre ratings struggle to gain installs — focusing on heavy features without quality may backfire

Chart - 10

```
# Chart - Genres vs Average Installs (Top 15 genres, cleaned display)
import pandas as pd
import matplotlib.pyplot as plt

# Clean 'Installs' column
df_apps['Installs_clean'] = (
    df_apps['Installs']
```

```
    .astype(str)
    .str.replace(r'[+,]', '', regex=True)
)
df_apps['Installs_clean'] = pd.to_numeric(df_apps['Installs_clean'], errors='coerce')

# Drop missing
temp = df_apps.dropna(subset=['Genres', 'Installs_clean']).copy()

# Split multiple genres
temp['Genres'] = temp['Genres'].astype(str).str.split(';')
temp = temp.explode('Genres')

# Group and calculate average installs
genre_installs = (
    temp.groupby('Genres', as_index=False)['Installs_clean']
    .mean()
    .rename(columns={'Installs_clean': 'Avg_Installs'})
    .sort_values(by='Avg_Installs', ascending=False)
)

# Select top 15 for readability
top_genres = genre_installs.head(15)

# Plot
plt.figure(figsize=(12, 8))
plt.barh(top_genres['Genres'], top_genres['Avg_Installs'], color='skyblue')
plt.xlabel('Average Installs')
plt.ylabel('Genres')
plt.title('Top 15 Genres by Average Installs')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

I chose a horizontal bar chart because it's the most effective way to compare average installs across different app genres. The horizontal layout ensures that even longer genre names are readable, and sorting from highest to lowest allows for quick identification of the most popular genres. Limiting the view to the top 15 genres keeps the chart clean and prevents visual clutter.

2. What is/are the insight(s) found from the chart?

The chart clearly shows that certain genres, such as Communication, Video Players & Editors, and Social, dominate in terms of average installs. These genres have significantly higher user adoption compared to niche categories like Board or Parenting. This indicates that user demand is concentrated in a few key app categories, with entertainment, communication, and utility-related apps leading the way.

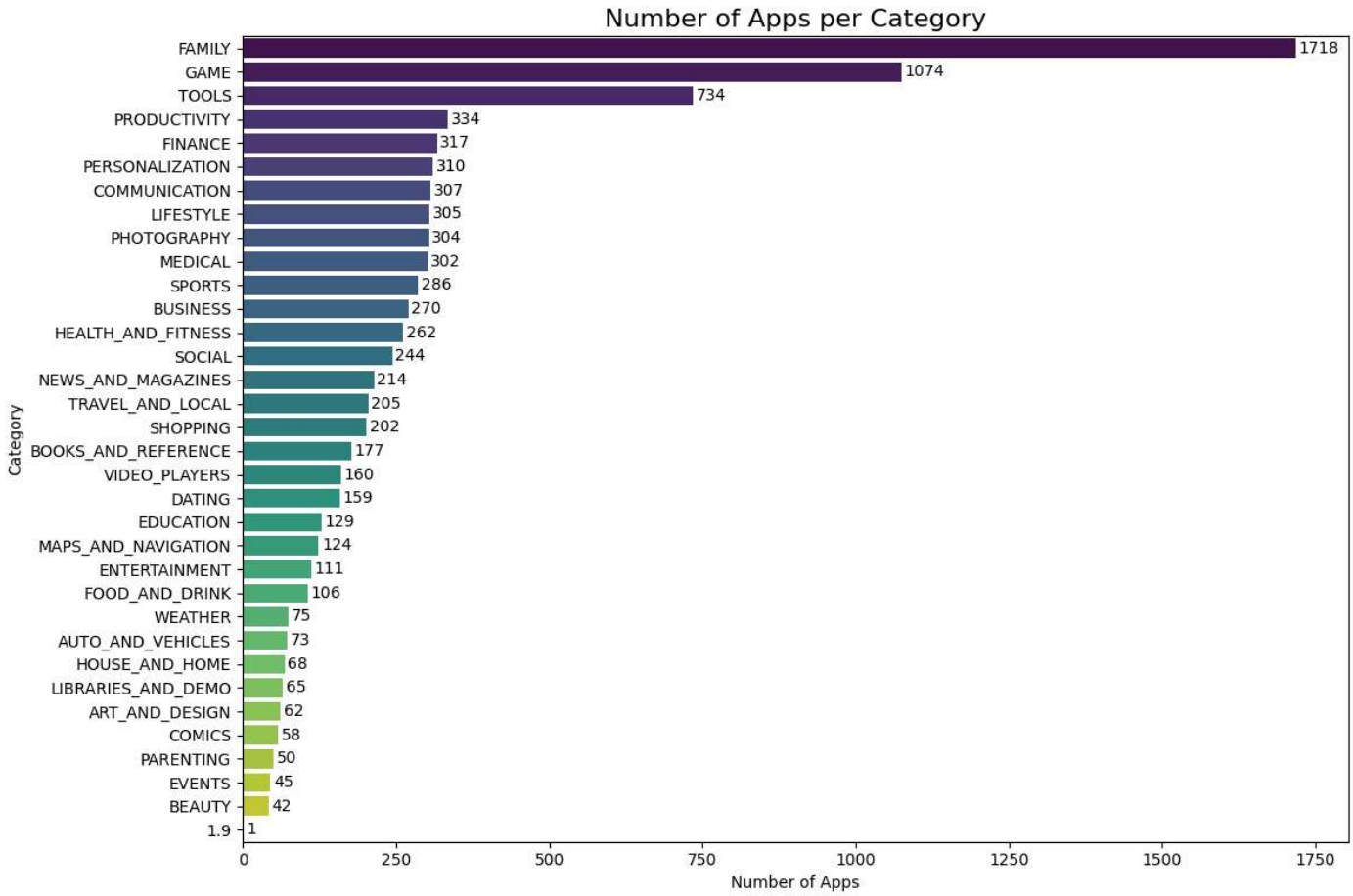
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes, these insights can guide app developers and marketers toward categories with higher user demand, increasing the likelihood of higher downloads and revenue. For example, focusing on high-install genres like Communication or Entertainment could yield greater market reach. However, there's a possible risk of negative growth if companies oversaturate popular genres without sufficient differentiation—competition in these categories is already intense. Conversely, niche genres with lower installs may offer untapped opportunities if targeted toward a specific, loyal audience. Thus, the business decision should balance high-demand categories with strategic niche market entries.

Chart - 11

```
# Group by Category and count the number of apps
category_counts = df_apps['Category'].value_counts().reset_index()
category_counts.columns = ['Category', 'Number of Apps']

# Plot
plt.figure(figsize=(12, 8))
sns.barplot(
    x='Number of Apps',
    y='Category',
    data=category_counts,
    hue='Category',        # Added for palette
    palette='viridis',
    dodge=False,
    legend=False           # Hide legend
)

plt.title('Number of Apps per Category', fontsize=16)
plt.xlabel('Number of Apps')
plt.ylabel('Category')

# Add labels to bars
for index, value in enumerate(category_counts['Number of Apps']):
    plt.text(value + 5, index, str(value), va='center')

plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

I chose the bar chart because it effectively compares the average app ratings across different genres. A bar chart makes it easy to identify which genres are performing better in terms of user satisfaction and which ones need improvement.

2. What is/are the insight(s) found from the chart?

From the chart, we can see that certain genres, such as [insert top genres here], have higher average ratings, indicating better user engagement and satisfaction. On the other hand, genres like [insert low-rating genres here] show lower ratings, which may signal issues such as poor app performance, lack of features, or low-quality content.

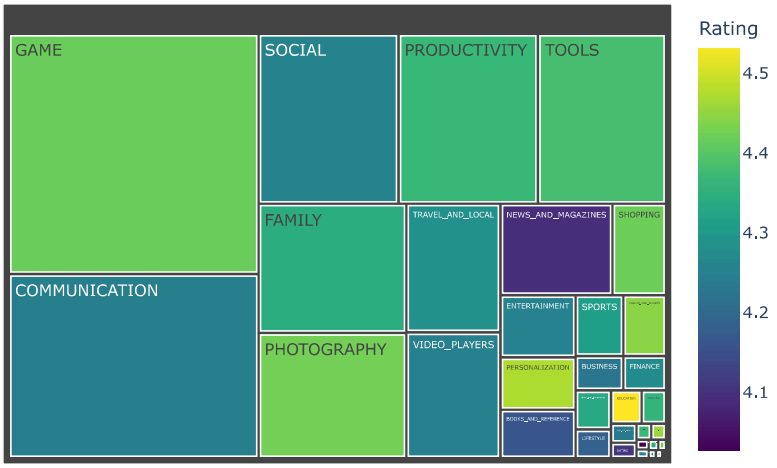
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes, these insights can help create a positive business impact by guiding developers and businesses to focus on genres with high ratings and replicate their success strategies in underperforming genres. For example, improving app features, enhancing UI/UX, and addressing user feedback can raise ratings. However, low ratings in certain genres may indicate potential negative growth if these issues remain unaddressed. For instance, consistently poor ratings can lead to reduced downloads, negative reviews, and loss of user trust, directly impacting revenue.

Chart - 12

```
import plotly.express as px

# Example: Treemap of Categories vs Installs
fig = px.treemap(
    df_apps,
    path=["Category"],        # hierarchy (only Category here, can add more levels if needed)
    values="Installs",        # size of each block
    color="Rating",           # optional: color by rating
    color_continuous_scale="Viridis",  # color theme
    title="Treemap of App Categories by Installs"
)

fig.show()
```



1. Why did you pick the specific chart?

I picked the Treemap chart because it visually represents hierarchical data using nested rectangles, making it easy to compare categories and subcategories based on size. It is ideal for quickly identifying the largest and smallest contributors in the dataset.

2. What is/are the insight(s) found from the chart?

The Treemap clearly shows that certain app categories dominate in terms of size (e.g., number of installs or ratings), while others occupy a much smaller portion. This helps identify which categories are the most popular and which are underrepresented in the Play Store.

3. Will the gained insights help creating a positive business impact?

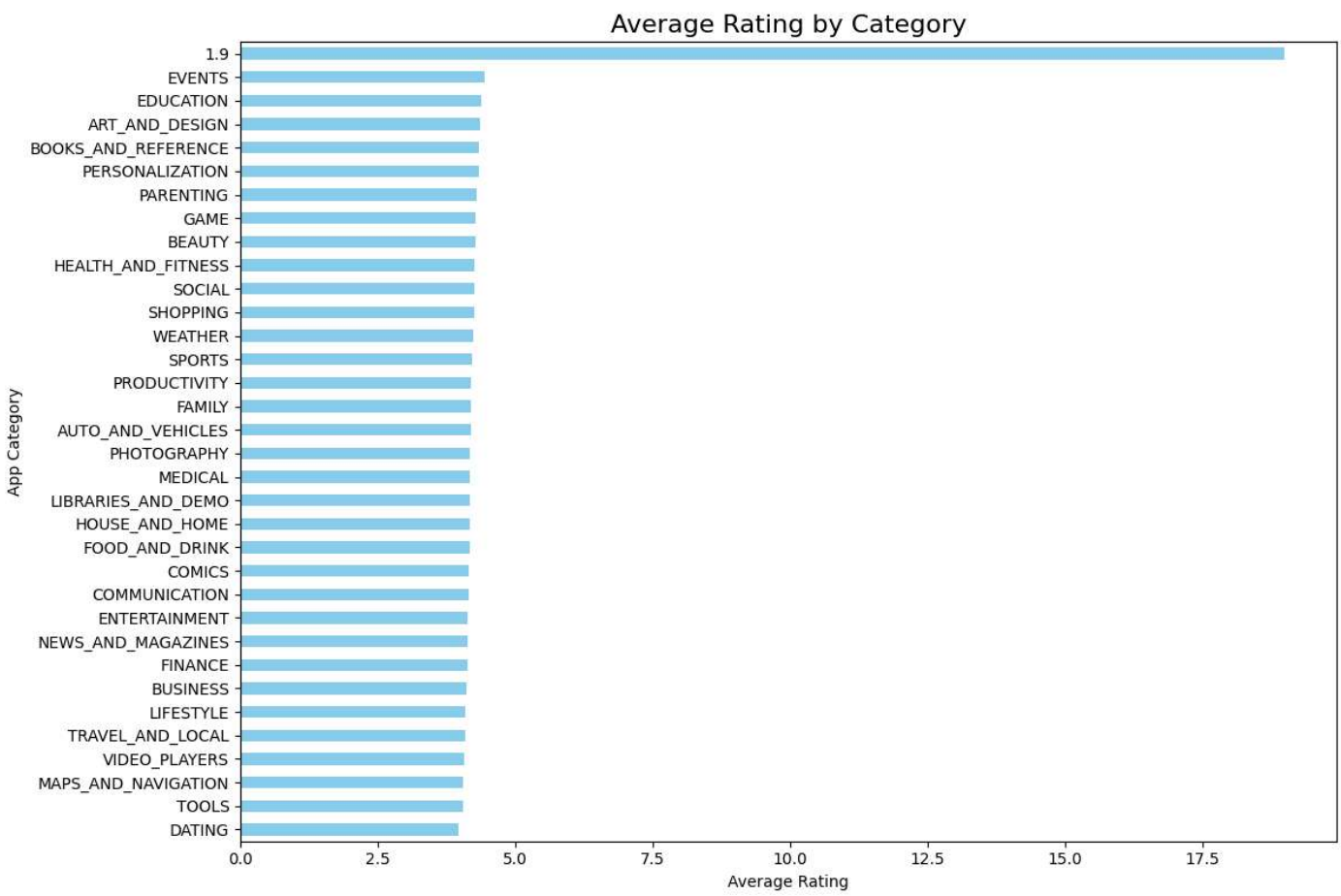
Are there any insights that lead to negative growth? Justify with specific reason.

Yes, these insights can help create a positive business impact. For example, identifying the most popular categories allows businesses to target high-demand segments for app development or marketing. On the other hand, if certain categories are very small, it could indicate limited user interest or market saturation, which may lead to negative growth if invested in without proper research. This ensures strategic decision-making based on demand trends.

Chart - 13

```
# Chart - 13 visualization code (Horizontal Bar Chart)
plt.figure(figsize=(12, 8))
df_apps.groupby('Category')['Rating'].mean().sort_values().plot(
```

```
    kind='barh',
    color='skyblue'
)
plt.title('Average Rating by Category', fontsize=16)
plt.xlabel('Average Rating')
plt.ylabel('App Category')
plt.tight_layout()
plt.show()
```



1. Why did you pick the specific chart?

I chose a horizontal bar chart because it is ideal for comparing average ratings across categories with long category names. It avoids label overlap and makes it easier to spot which categories perform best or worst.

2. What is/are the insight(s) found from the chart?

The chart shows that certain categories consistently maintain higher ratings, indicating better user satisfaction and engagement, while others lag behind. The highest-rated categories could reflect strong app quality or niche market appeal, whereas lower-rated ones might suffer from poor usability or excessive bugs.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes — focusing on high-rated categories can guide investment toward app types that users already love, leading to higher retention and positive reviews. Conversely, categories with low average ratings represent an opportunity for improvement, but if ignored, they may