|  |  |
| --- | --- |
| Project Title | **Google Play Store Apps**  (regulatory affairs) |
| Tools | Python, ML, SQL, Excel |
| Domain | Data Analyst & Data scientist |
| Project Difficulties level | intermediate |

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

[Click](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing) [here](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing) [to](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing) [download](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing) [data](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing) [set](https://drive.google.com/drive/folders/13RYJ7YfjwlavX3Twg5KTR1DYg2_PVhca?usp=sharing)

**About Dataset**

# Context

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

**Content**

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

# Acknowledgements

**This information is scraped from the Google Play Store. This app information would not be available**

**without it.**

# Inspiration

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

**NOTE :**

1. **this project is only for your guidance, not exactly the same you have to create. Here I am trying to show the way or idea of what steps you can follow and how your projects look. Some projects are very advanced (because it will be made with the help of flask, nlp, advance ai, advance DL and some advanced things ) which you can not understand .**
2. **You can make or analyze your project with yourself, with your idea, make it more creative from where we can get some information and understand about our business. make sure what overall things you have created all things you understand very well.**

**Example: You can get the basic idea how you can create a project from here** what steps you should have to follow

Here's a beginner-friendly guide to start a data analytics project using the "Google Play Store Apps" dataset with the specified columns. I'll walk you through the key steps, including code snippets and expected outputs.

**Project Title:**

**Exploratory Data Analysis of Google Play Store Apps**

1. **Objective**

The goal of this project is to analyze the characteristics of apps on the Google Play Store, including their ratings, reviews, sizes, installation counts, and more. The analysis will help identify trends, outliers, and patterns in the app market.

1. **Steps to Follow**

**Step 1: Import Libraries**

|  |
| --- |
| You'll need to import the necessary Python libraries for data manipulation and visualization.  import pandas as pd import matplotlib.pyplot as plt import seaborn as sns  **Step 2: Load the Dataset**  Assuming your dataset is in a CSV file, you can load it using Pandas.  python  Copy code df = pd.read\_csv('google\_play\_store\_apps.csv')  **Step 3: Basic Data Exploration**  Start by exploring the dataset to understand its structure.  # Display the first few rows  print(df.head())  # Get basic information about the dataset |

|  |
| --- |
| print(df.info())  # Summary statistics of numerical columns print(df.describe())  **Expected Output:**   * The first few rows of the dataset will display columns like App, Category,   Rating, etc.   * The info() method will show the data types and any missing values. * describe() will provide summary statistics for numerical columns like Rating, Reviews, Size, etc.   **Step 4: Data Cleaning**  You may need to clean the data by handling missing values, converting data types, and removing duplicates.  # Check for missing values print(df.isnull().sum())  # Handle missing values (e.g., filling or dropping) df['Rating'].fillna(df['Rating'].mean(), inplace=True) |

|  |
| --- |
| df.dropna(subset=['App', 'Category'], inplace=True)  # Convert columns to appropriate data types df['Reviews'] = df['Reviews'].astype(int)  df['Installs'] = df['Installs'].str.replace(',',  '').str.replace('+', '').astype(int) df['Price'] = df['Price'].str.replace('$', '').astype(float)  **Expected Output:**   * The output will show the number of missing values in each column. * The dataset will be cleaned with missing values handled and data types converted as needed.   **Step 5: Data Visualization**  Visualizing the data helps to understand the distribution and relationships between variables.  # Distribution of Ratings plt.figure(figsize=(10, 6)) sns.histplot(df['Rating'], bins=20, kde=True) |

|  |
| --- |
| plt.title('Distribution of App Ratings') plt.show()  # Count of Apps by Category plt.figure(figsize=(12, 8))  sns.countplot(y='Category', data=df,  order=df['Category'].value\_counts().index) plt.title('Count of Apps by Category') plt.show()  # Relationship between Installs and Rating plt.figure(figsize=(10, 6))  sns.scatterplot(x='Rating', y='Installs', hue='Category', data=df) plt.title('Relationship between Installs and Ratings') plt.show()  **Expected Output:**  ● A histogram showing the distribution of app ratings. |

|  |
| --- |
| * A bar chart showing the count of apps by category. * A scatter plot showing the relationship between the number of installs and app ratings, with colors representing different categories.   **Step 6: Analyzing Key Metrics**  You can perform further analysis to extract insights.  # Average rating by category  avg\_rating\_by\_category = df.groupby('Category')['Rating'].mean().sort\_values(ascending=F alse) print(avg\_rating\_by\_category)  # Most popular apps (by installs)  most\_installed\_apps = df[['App',  'Installs']].sort\_values(by='Installs', ascending=False).head(10) print(most\_installed\_apps)  # Top 5 genres  top\_genres = df['Genres'].value\_counts().head(5) |

|  |
| --- |
| print(top\_genres)  **Expected Output:**   * A list of average ratings by app category. * A list of the top 10 most installed apps. * The top 5 most common genres.  1. **Conclusion**   Summarize the findings from your analysis, discussing any trends, patterns, or anomalies observed. For example, you might find that certain categories have higher average ratings or that specific genres dominate the market.   1. **Next Steps**   Consider exploring further:   * + Sentiment analysis of user reviews.   + Time series analysis of app updates and their impact on ratings.   + Predictive modeling to forecast app ratings based on features.   This project provides a foundational understanding of exploratory data analysis using real-world data from the Google Play Store. |

**Example: You can get the basic idea how you can create a project from here**

# Sample code and output

Importing

Libraries

[¶](https://www.kaggle.com/code/hasnainkhan0123/complete-end-to-end-eda-of-google-play-store#Importing-Libraries)

In

[1]:

import

pandas

as

pd

import

numpy

as

np

import

seaborn

as

sns

import

matplotlib.pyplot

as

plt

%

matplotlib

inline

import

warnings

warnings

.

filterwarnings(

'ignore'

)

2.

Data

Loading

and

exploration

and

cleaning

c

Load

the

csv

file

with

the

pandas

↪

creating

the

dataframe

and

understanding

the

data

present

in

the

dataset

using

pandas

↪

Dealing

with

the

missing

data,

outliers

and

the

incorrect

records

In

[2]:

df

=

pd

.

read\_csv(

'/kaggle/input/google-play-store-apps/googleplaystore.csv'

)

df

.

head(

4

)

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | |
|  |  | App | Category | Rati ng | Revie ws | Siz e | Installs | Ty pe | Pri ce | Conte  nt  Rating | Genres | Last Updat ed | Curr ent Ver | Andr oid Ver |
| 0 | Photo  Editor & Candy  Camera  & Grid  &  ScrapB ook | ART\_AND\_DE  SIGN | 4.1 | 159 | 19  M | 10,000+ | Fre e | 0 | Everyo ne | Art &  Design | Janua ry 7, 2018 | 1.0.0 | 4.0.3 and up |
| 1 | Colorin g book moana | ART\_AND\_DE  SIGN | 3.9 | 967 | 14  M | 500,000  + | Fre e | 0 | Everyo ne | Art &  Design;Pret end Play | Janua ry 15, 2018 | 2.0.0 | 4.0.3 and up |
| 2 | U  Launch er Lite –  FREE  Live  Cool  Themes  , Hide  ... | ART\_AND\_DE  SIGN | 4.7 | 8751  0 | 8.7  M | 5,000,00  0+ | Fre e | 0 | Everyo ne | Art &  Design | Augu st 1, 2018 | 1.2.4 | 4.0.3 and up |
| 3 | Sketch  - Draw  & Paint | ART\_AND\_DE  SIGN | 4.5 | 2156  44 | 25  M | 50,000,0  00+ | Fre e | 0 | Teen | Art &  Design | June  8,  2018 | Varie s with devic e | 4.2 and up |
| In [3]: df.iloc[10474: 10494] | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: | | | | | | | | | | | | | | |
|  |  | App | Category | Rati ng | Revi ews | Si ze | Installs | Ty pe | Pri ce | Conte  nt Ratin g | Genres | Last Update d | Curre nt Ver | Andr oid Ver |
| 104  74 | Sat-Fi  Voice | COMMUNICATI  ON | 3.4 | 37 | 14  M | 1,000+ | Fr ee | 0 | Every one | Communic  ation | Novem ber 21, 2014 | 2.2.1.  5 | 2.2 and up |
| 104  75 | Wi-Fi Visuali zer | TOOLS | 3.9 | 132 | 2.  6  M | 50,000  + | Fr ee | 0 | Every one | Tools | May  17,  2017 | 0.0.9 | 2.3 and up |
| 104  76 | Lennox iComfo rt Wi-Fi | LIFESTYLE | 3.0 | 552 | 7.  6  M | 50,000  + | Fr ee | 0 | Every one | Lifestyle | March  22,  2017 | 2.0.1  5 | 2.3.3 and up |
| 104  77 | Sci-Fi  Sound s and  Ringto nes | PERSONALIZAT  ION | 3.6 | 128 | 11  M | 10,000  + | Fr ee | 0 | Every one | Personaliz ation | Septe mber  27,  2017 | 4.0 | 4.0 and up |
| 104  78 | Sci Fi Sound s | FAMILY | 3.2 | 4 | 8.  0  M | 1,000+ | Fr ee | 0 | Every one | Entertain ment | Novem ber 2, 2017 | 1.0 | 4.0 and up |
| 104  79 | Free  Wi-fi  Hotspo  T | COMMUNICATI  ON | 4.1 | 382 | 2.  3  M | 50,000  + | Fr ee | 0 | Every one | Communic  ation | July  20,  2018 | 2.5 | 4.0 and up |
|  | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | |
|  | 104  80 | FJ 4x4  Cruiser  Offroad  Driving | FAMILY | 4.1 | 3543 | 49  M | 500,00  0+ | Fr ee | 0 | Every one | Simulation | Januar y 4, 2017 | 1.1 | 2.3 and up |
| 104  81 | FJ 4x4  Cruiser  Snow  Driving | FAMILY | 4.2 | 1619 | 43  M | 500,00  0+ | Fr ee | 0 | Every one | Simulation | June 4,  2018 | 1.3 | 4.0 and up |
| 104  82 | Wallpa pers  Toyota  FJ  Cruiser | PERSONALIZAT  ION | 4.2 | 78 | 10  M | 10,000  + | Fr ee | 0 | Every one | Personaliz ation | June  20,  2016 | 1.0 | 2.3.3 and up |
| 104  83 | New  Wallpa pers  Toyota  FJ  Cruiser  Theme | PERSONALIZAT  ION | Na  N | 1 | 16  M | 100+ | Fr ee | 0 | Teen | Personaliz ation | Februa ry 23, 2018 | 1.0 | 4.1 and up |
| 104  84 | FJ  Final  Join ,  Circles  Game | GAME | 4.7 | 32 | 24  M | 1,000+ | Fr ee | 0 | Teen | Arcade | July  11,  2018 | 0.24 | 4.3 and up |
| 104  85 | HD  Wallpa per Toyota  FJ  Cruiser | TOOLS | Na  N | 2 | 6.  2  M | 100+ | Fr ee | 0 | Every one | Tools | Novem ber 10, 2017 | 1.0 | 4.0 and up |
|  | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | |
|  | 104  86 | FJ  Drive: Merce des-Be nz Lease | AUTO\_AND\_VE  HICLES | 4.6 | 107 | 27  M | 10,000  + | Fr ee | 0 | Every one | Auto &  Vehicles | Novem ber 6, 2017 | 2.0.0 | 4.1 and up |
| 104  87 | Driving n  Parkin g School  2017 | FAMILY | 4.5 | 15 | 46  M | 1,000+ | Fr ee | 0 | Every one | Simulation | May  31,  2017 | 1.0 | 2.3 and up |
| 104  88 | FJ  WiFi  HDD | TOOLS | Na  N | 40 | 2.  4  M | 5,000+ | Fr ee | 0 | Every one | Tools | Octobe r 31, 2017 | 1.0.5 | 2.1 and up |
| 104  89 | Offroad  Cruiser | FAMILY | 4.3 | 4243  2 | 36  M | 1,000,0  00+ | Fr ee | 0 | Every one | Simulation | July  13,  2016 | 1.3 | 2.3.3 and up |
| 104  90 | HD  Theme s Toyota  Cruiser  70 | PERSONALIZAT  ION | 4.5 | 86 | 17  M | 10,000  + | Fr ee | 0 | Teen | Personaliz ation | Octobe r 2, 2016 | 1.0 | 2.3.3 and up |
| 104  91 | Toyota Cruiser  s &  Trucks  Mag | TRAVEL\_AND\_L  OCAL | 4.5 | 10 | 8.  0  M | 500+ | Fr ee | 0 | Every one | Travel &  Local | March  14,  2018 | 3.0.0 | 4.4 and up |
|  | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | |
|  | 104  92 | 4 x4  Offroad  SUV  3D  Truck Simula  tor Driving  2017 | FAMILY | 4.4 | 32 | 37  M | 1,000+ | Fr ee | 0 | Every one | Simulation | Decem ber 6, 2017 | 1.0 | 2.3 and up |
| 104  93 | Cake  Shop -  Kids Cookin g | FAMILY | 4.3 | 3066  8 | 33  M | 5,000,0  00+ | Fr ee | 0 | Every one | Casual;Pr etend Play | July  16,  2018 | 2.1.3  181 | 4.0.3 and up |
|  | | |  |  | |  |  |  |  | | In [4]: | | |
| df.sample(10)  AppCategory  67  BT Share ItBUSINESS  81  27Groupon -  16ShopSHOPPING  Deals,  Discounts | | | | Rat ing  4.7  4.6 | ReviSiz  ewse  13  12  M  Var  1370ies  749wit  h de vic | | |  |  | | --- | --- | |  | Installs | |  | 500+ | |  | 50,000,  000+ | | Ty p e  Fr e e  Fr e e | Pr  ic  e 0  0 | Cont  ent  Genres  Ratin g  Every  Business one  TeenShopping | | Out[4]:  LastCurrAnd Updentroid atedVerVer  May  4.4  16,3.4.  and  2012  up  8  AugVariVari  usteses  3,withwith  201devidevi | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | & Coupons |  |  |  | e |  |  |  |  |  | 8 | ce | ce | | 20  73 | Super School: Educationa l Kids  Games &  Rhymes | FAMILY | 4.5 | 1791 | 56  M | 500,000  + | Fr e e | 0 | Every one | Education;E ducation | Jun e 2, 201  8 | 5.3.  11 | 5.0 and up | | 71  67 | CD - Teach me ABC  English L1 | FAMILY | Na  N | 2 | 63  M | 500+ | Fr e e | 0 | Every one | Education | Jun e 18,  201  7 | 1.0.  0 | 4.0 and up | | 36  92 | OnePlus  Gallery | VIDEO\_PLAYE  RS | 3.8 | 5555 | 64  M | 1,000,0  00+ | Fr e e | 0 | Every one | Video  Players &  Editors | July  12,  201  8 | 2.6.  71 | 7.1 and up | | 76  49 | Krypton by krypt.co | PRODUCTIVIT  Y | 4.6 | 38 | 13  M | 1,000+ | Fr e e | 0 | Every one | Productivity | July  17,  201  8 | 2.4.  0 | 6.0 and up | | 15  17 | Lamp  detector | LIBRARIES\_AN  D\_DEMO | Na  N | 5 | 1.8  M | 1,000+ | Fr e e | 0 | Every one | Libraries &  Demo | April  23,  201  8 | 4.4.  2 | 2.3.  3 and up | | 28  65 | Cymera  Camera-  Photo  Editor, Filter,Colla ge,La... | PHOTOGRAPH  Y | 4.4 | 2418  135 | Var ies wit  h de vic  e | 100,000  ,000+ | Fr e e | 0 | Every one | Photography | July  12,  201  8 | Vari es with devi ce | Vari es with devi ce | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 70  67 | ePN  Cashback  AliExpress | SHOPPING | 4.4 | 1921  2 | 6.9  M | 500,000  + | Fr e e | 0 | Every one | Shopping | Aug  ust 3,  201  8 | 0.2. 9.17 | 4.1 and up | | 47  96 | YouTube  Studio | VIDEO\_PLAYE  RS | 4.3 | 4361  70 | Var ies wit  h de vic  e | 10,000,  000+ | Fr e e | 0 | Teen | Video  Players &  Editors | Jun e 28,  201  8 | Vari es with devi ce | Vari es with devi ce |   Checking the tail of the column  In [5]:  df.tail()  Out[5]:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | App | Category | Rati ng | Revi ews | Siz e | Installs | Ty pe | Pri ce | Conte  nt Ratin g | Genre s | Last Upda  ted | Curr ent Ver | Andr oid Ver | | 108  36 | Sya9a  Maroc -  FR | FAMILY | 4.5 | 38 | 53  M | 5,000+ | Fr ee | 0 | Every one | Educa  tion | July  25,  2017 | 1.48 | 4.1 and up | | 108  37 | Fr.  Mike Schmit z Audio Teachin | FAMILY | 5.0 | 4 | 3.6  M | 100+ | Fr ee | 0 | Every one | Educa  tion | July  6,  2018 | 1.0 | 4.1 and up | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | gs |  |  |  |  |  |  |  |  |  |  |  |  | | 108  38 | Parkins on  Exercic  es FR | MEDICAL | Na  N | 3 | 9.5  M | 1,000+ | Fr ee | 0 | Every one | Medic  al | Janu ary 20,  2017 | 1.0 | 2.2 and up | | 108  39 | The  SCP  Founda tion DB fr nn5n | BOOKS\_AND\_REF  ERENCE | 4.5 | 114 | Vari es with dev ice | 1,000+ | Fr ee | 0 | Matur e 17+ | Books  &  Refere nce | Janu ary 19,  2015 | Vari es with devi ce | Vari es with devi ce | | 108  40 | iHorosc ope 2018  Daily Horosc ope &  Astrolo gy | LIFESTYLE | 4.5 | 3983  07 | 19  M | 10,000,  000+ | Fr ee | 0 | Every one | Lifesty le | July  25,  2018 | Vari es with devi ce | Vari es with devi ce |   Set the option maximum of rows and column  In [6]:  pd.set\_option('display.max\_columns', None)  In [7]: pd.set\_option('display.max\_rows', None)  Checking the shape of the columns  In [8]: |

print

(

f

'The

number

of

Rows

are

"

**{**

df

.

shape[

0

]

**}**

",

and

the

number

of

columns

are

"

**{**

df

.

shape[

1

]

**}**

"'

)

The

number

of

Rows

are

"10841",

and

the

number

of

columns

are

"13"

In

[9]:

print

(

f

'The

name

of

the

columns

are:

**{**

df

.

columns

**}**

'

)

The

name

of

the

columns

are:

Index(['App',

'Category',

'Rating',

'Reviews',

'Size',

'Installs',

'Type',

'Price',

'Content

Rating',

'Genres',

'Last

Updated',

'Current

Ver',

'Android

Ver'],

dtype='object')

Checking

the

info

of

the

dataset

In

[10]:

df

.

info()

<

class

'pandas.core.frame.DataFrame'>

RangeIndex:

10841

entries,

0

to

10840

Data

columns

(

total

13

columns):

#

Column

Non-Null

Count

Dtype

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | count | 9367.000000 | | mean | 4.193338 | | std | 0.537431 | | min | 1.000000 | | 25% | 4.000000 | | 50% | 4.300000 | | 75% | 4.500000 | | max | 19.000000 |   Removing this row from the data because this is causing some problem 10472  In [12]:  df.drop(10472, axis=0, inplace=True)  In [13]: df.info() |

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Index: 10840 entries, 0 to 10840 Data columns (total 13 columns):  # Column Non-Null Count Dtype  --- ------ -------------- ----0 App 10840 non-null object   1. Category 10840 non-null object 2. Rating 9366 non-null float64 3. Reviews 10840 non-null object 4. Size 10840 non-null object 5. Installs 10840 non-null object 6. Type 10839 non-null object 7. Price 10840 non-null object 8. Content Rating 10840 non-null object 9. Genres 10840 non-null object 10. Last Updated 10840 non-null object 11. Current Ver 10832 non-null object 12. Android Ver 10838 non-null object   dtypes: float64(1), object(12)  memory usage: 1.2+ MB  In [14]: df['Reviews'] = df['Reviews'].astype('int') |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| In [15]:  df.describe()  Out[15]:   |  |  |  | | --- | --- | --- | |  | Rating | Reviews | | count | 9366.000000 | 1.084000e+04 | | mean | 4.191757 | 4.441529e+05 | | std | 0.515219 | 2.927761e+06 | | min | 1.000000 | 0.000000e+00 | | 25% | 4.000000 | 3.800000e+01 | | 50% | 4.300000 | 2.094000e+03 | | 75% | 4.500000 | 5.477550e+04 | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | max | 5.000000 | 7.815831e+07 |   Taking size column and make it numeric  In [16]:  df['Size'].value\_counts()  Out[16]:  Size  Varies with device 1695  11M 198  12M 196  14M 194  13M 191  15M 184  17M 160  19M 154  26M 149  16M 149  25M 143  20M 139  21M 138  10M 136  24M 136 |

|  |
| --- |
| df['Size'].isnull().sum()  Out[17]:  0  There is no missing values in the size column  Checking the number of values in three different categories in Size  In [18]:  print("Number of M in Size Column",  df['Size'].loc[df['Size'].str.contains('M')].value\_counts().sum())  print("Number of k in Size Column",  df['Size'].loc[df['Size'].str.contains('k')].value\_counts().sum())  print("Number of Varies with device in Size Column",  df['Size'].loc[df['Size'].str.contains('Varies with device')].value\_counts().sum())  Number of M in Size Column 8829  Number of k in Size Column 316  Number of Varies with device in Size Column 1695  Convert the whole size of the column into bytes  In [19]: |

*###*

*Defining*

*a*

*Function*

def

convert\_into\_bytes(column\_name):

if

isinstance

column\_name,

(

str

):

if

'k'

**in**

column\_name:

return

float

(

column\_name

.

replace(

"k"

,

""

))

\*

1024

elif

'M'

**in**

column\_name:

return

float

(

column\_name

.

replace(

"M"

,

""

))

\*

1024

\*

1024

elif

'Varies

with

device'

**in**

column\_name:

return

np

.

nan

return

column\_name

In

[20]:

df[

'Size'

]

=

df[

'Size'

]

.

apply(convert\_into\_bytes)

In

[21]:

df[

'Size'

]

Out[21]:

0

19922944.0

1

14680064.0

2

9122611.2

|  |
| --- |
| 1. 26214400.0 2. 2936012.8 3. 5872025.6 4. 19922944.0 5. 30408704.0 6. 34603008.0 7. 3250585.6   Observations[¶](https://www.kaggle.com/code/hasnainkhan0123/complete-end-to-end-eda-of-google-play-store#Observations)   * + Remove + sign   + Remove , from the values   + Convert the column in to integers   In [26]:  *## Define a function to deal with installs column*  def installs(install):  if isinstance(install, str):  if '+' **in** install: return install.replace("+", "")  return int(install)  In [27]: |

df[

'Installs'

]

=

df[

'Installs'

]

.

apply(installs)

In

[28]:

df[

'Installs'

]

=

df[

'Installs'

]

.

apply(

lambda

x:

x

.

replace(

','

,

''

)

if

','

**in**

str

)

(

x

else

x)

In

[29]:

df[

'Installs'

]

=

df[

'Installs'

]

.

astype(

'int'

)

In

[30]:

df[

'Installs'

]

.

value\_counts()

Out[30]:

Installs

1000000

1579

10000000

1252

100000

1169

10000

1054

1000

907

5000000

752

100

719

500000

539

50000

479

5000 477

100000000 409

10 386

500 330

50000000 289

50 205

5 82

500000000 72

1 67

1000000000 58

0 15

Name: count, dtype: int64

In [31]: *# making a new column called 'Installs\_category' which will have the category of the*

*installs*

bins = [-1, 0, 10, 1000, 10000, 100000, 1000000, 10000000, 10000000000] labels=['no', 'Very low', 'Low', 'Moderate', 'More than moderate', 'High', 'Very High', 'Top Notch']

df['Installs\_category'] = pd.cut(df['Installs'], bins=bins, labels=labels)

In [32]:

df['Installs\_category'].value\_counts()

Out[32]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Installs\_category  Low 2161  High 2118  Very High 2004  More than moderate 1648  Moderate 1531  Top Notch 828 Very low 535  no 15  Name: count, dtype: int64 | | | | | |  |  |  |  |  |  |  |  |  |  | In [33]: |
| df.head(4) | | | | | |  |  |  |  |  |  |  |  |  |  | Out[33]: |
|  |  | App | Category | Ra  tin  g | Rev iew s | Size\_in  \_bytes | Instal ls | T y p e | Pr  ic  e | Cont ent Rati ng | Genres | Last  Upd ated | Cur ren  t  Ver | An  droi d Ver | Size  \_MB | Installs\_c ategory |
| 0 | Photo Editor  & Cand y  Came ra &  Grid  &  Scrap  Book | ART\_AND\_  DESIGN | 4.1 | 159 | 199229  44.0 | 1000  0 | Fr e e | 0 | Ever yone | Art &  Design | Jan uary  7,  201  8 | 1.0  .0 | 4.0.  3 and up | 19.0 | Moderate |
|  | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | |
|  | 1 | Colori ng book moan a | ART\_AND\_  DESIGN | 3.9 | 967 | 146800  64.0 | 5000  00 | Fr e e | 0 | Ever yone | Art &  Design;  Pretend  Play | Jan uary 15,  201  8 | 2.0  .0 | 4.0.  3 and up | 14.0 | High |
| 2 | U  Launc her Lite –  FREE  Live  Cool Them  es, Hide  ... | ART\_AND\_  DESIGN | 4.7 | 875  10 | 912261  1.2 | 5000  000 | Fr e e | 0 | Ever yone | Art &  Design | Aug  ust 1,  201  8 | 1.2  .4 | 4.0.  3 and up | 8.7 | Very  High |
| 3 | Sketc h -  Draw  &  Paint | ART\_AND\_  DESIGN | 4.5 | 215  644 | 262144  00.0 | 5000  0000 | Fr e e | 0 | Teen | Art &  Design | Jun e 8, 201  8 | Var ies wit  h dev ice | 4.2 and up | 25.0 | Top  Notch |
| Taking Price column | | | | | | | | | | | | | | | In [34]: |
| df['Price'].unique()  array(['0', '$4.99', '$3.99', '$6.99', '$1.49', '$2.99', '$7.99', '$5.99',  '$3.49', '$1.99', '$9.99', '$7.49', '$0.99', '$9.00', '$5.49',  '$10.00', '$24.99', '$11.99', '$79.99', '$16.99', '$14.99',  '$1.00', '$29.99', '$12.99', '$2.49', '$10.99', '$1.50', '$19.99', | | | | | | | | | | | | | | | | Out[34]: |

|  |
| --- |
| '$15.99', '$33.99', '$74.99', '$39.99', '$3.95', '$4.49', '$1.70',  '$8.99', '$2.00', '$3.88', '$25.99', '$399.99', '$17.99',  '$400.00', '$3.02', '$1.76', '$4.84', '$4.77', '$1.61', '$2.50',  '$1.59', '$6.49', '$1.29', '$5.00', '$13.99', '$299.99', '$379.99',  '$37.99', '$18.99', '$389.99', '$19.90', '$8.49', '$1.75',  '$14.00', '$4.85', '$46.99', '$109.99', '$154.99', '$3.08',  '$2.59', '$4.80', '$1.96', '$19.40', '$3.90', '$4.59', '$15.46',  '$3.04', '$4.29', '$2.60', '$3.28', '$4.60', '$28.99', '$2.95',  '$2.90', '$1.97', '$200.00', '$89.99', '$2.56', '$30.99', '$3.61',  '$394.99', '$1.26', '$1.20', '$1.04'], dtype=object)  In [35]: def adjust\_price(price):  if isinstance(price, str):  if '$' **in** price:  return price.replace("$", "")  return price  In [36]:  df['Price'] = df['Price'].apply(adjust\_price)  In [37]: df['Price'].unique() |

|  |
| --- |
| Out[37]: array(['0', '4.99', '3.99', '6.99', '1.49', '2.99', '7.99', '5.99',  '3.49', '1.99', '9.99', '7.49', '0.99', '9.00', '5.49', '10.00',  '24.99', '11.99', '79.99', '16.99', '14.99', '1.00', '29.99',  '12.99', '2.49', '10.99', '1.50', '19.99', '15.99', '33.99',  '74.99', '39.99', '3.95', '4.49', '1.70', '8.99', '2.00', '3.88',  '25.99', '399.99', '17.99', '400.00', '3.02', '1.76', '4.84',  '4.77', '1.61', '2.50', '1.59', '6.49', '1.29', '5.00', '13.99',  '299.99', '379.99', '37.99', '18.99', '389.99', '19.90', '8.49',  '1.75', '14.00', '4.85', '46.99', '109.99', '154.99', '3.08',  '2.59', '4.80', '1.96', '19.40', '3.90', '4.59', '15.46', '3.04',  '4.29', '2.60', '3.28', '4.60', '28.99', '2.95', '2.90', '1.97',  '200.00', '89.99', '2.56', '30.99', '3.61', '394.99', '1.26',  '1.20', '1.04'], dtype=object)  In [38]:  df['Price'].dtype  Out[38]:  dtype('O')  In [39]:  df['Price'] = df['Price'].astype('float')  In [40]: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| df.describe()  Out[40]:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | Rating | Reviews | Size\_in\_bytes | Installs | Price | Size\_MB | | count | 9366.000000 | 1.084000e+04 | 9.145000e+03 | 1.084000e+04 | 10840.000000 | 9145.000000 | | mean | 4.191757 | 4.441529e+05 | 2.256133e+07 | 1.546434e+07 | 1.027368 | 21.516165 | | std | 0.515219 | 2.927761e+06 | 2.368637e+07 | 8.502936e+07 | 15.949703 | 22.589084 | | min | 1.000000 | 0.000000e+00 | 8.704000e+03 | 0.000000e+00 | 0.000000 | 0.008301 | | 25% | 4.000000 | 3.800000e+01 | 5.138022e+06 | 1.000000e+03 | 0.000000 | 4.900000 | | 50% | 4.300000 | 2.094000e+03 | 1.363149e+07 | 1.000000e+05 | 0.000000 | 13.000000 | | 75% | 4.500000 | 5.477550e+04 | 3.145728e+07 | 5.000000e+06 | 0.000000 | 30.000000 | | max | 5.000000 | 7.815831e+07 | 1.048576e+08 | 1.000000e+09 | 400.000000 | 100.000000 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Observations:   * Now, we have only 6 columns as numeric data type. * We can observe their descriptive statistics. and make tons of observations as per our hypotheses. * We can see that the Rating column has a minimum value of 1 and a maximum value of 5, which is the range of rating, and the mean is 4.19 which is a good rating. On an average people give this rating. * We can see that the Reviews column has a minimum value of 0 and a maximum value of 78,158,306 78+ Millions, which is the range of reviews, and the mean is 444,111.93 which is a good number of reviews. On an average people give this number of reviews to the apps. But it does not make sense to us, as we have different categories of apps. * Similarly, we can observe the other columns as well.   Therefore, the most important thing is to classify as app based on the correlation matrix and then observe the descriptive statistics of the app category and number of installs, reviews, ratings, etc.  But even before that we have to think about the missing values in the dataset.  In [41]:  df.head()  Out[41]:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | App | Category | Ra  tin  g | Rev iew s | Size\_in  \_bytes | Insta  lls | T y p e | Pr  ic  e | Cont ent Rati ng | Genres | Last Upd  ate d | Cur ren  t  Ver | An  droi d Ver | Size  \_MB | Installs\_c ategory | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | Photo Editor  & Cand y  Cam era &  Grid  &  Scrap  Book | ART\_AND\_  DESIGN | 4.  1 | 159 | 199229  44.0 | 1000  0 | F  re e | 0.  0 | Ever yone | Art &  Design | Jan uar y 7, 201  8 | 1.0  .0 | 4.0.  3 and up | 19.0 | Moderate | | 1 | Colori ng book moan a | ART\_AND\_  DESIGN | 3.  9 | 967 | 146800  64.0 | 5000  00 | F  re e | 0.  0 | Ever yone | Art &  Design;Pr etend Play | Jan uar y 15,  201  8 | 2.0  .0 | 4.0.  3 and up | 14.0 | High | | 2 | U  Laun cher Lite –  FRE  E  Live  Cool Them  es, Hide  ... | ART\_AND\_  DESIGN | 4.  7 | 875  10 | 912261  1.2 | 5000  000 | F  re e | 0.  0 | Ever yone | Art &  Design | Aug  ust 1,  201  8 | 1.2  .4 | 4.0.  3 and up | 8.7 | Very  High | | 3 | Sketc h -  Draw  &  Paint | ART\_AND\_  DESIGN | 4.  5 | 215  644 | 262144  00.0 | 5000  0000 | F  re e | 0.  0 | Teen | Art &  Design | Jun e 8, 201  8 | Var ies wit  h dev ice | 4.2 and up | 25.0 | Top  Notch | | 4 | Pixel  Draw  -  Num | ART\_AND\_  DESIGN | 4.  3 | 967 | 293601  2.8 | 1000  00 | F  re | 0.  0 | Ever yone | Art &  Design;C | Jun e 20,  201 | 1.1 | 4.4 and | 2.8 | More than | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | ber Art Colori ng Book |  |  |  |  |  | e |  |  | reativity | 8 |  | up |  | moderate |   Missing Values  In [42]:  df.isnull().sum().sort\_values(ascending=False)  Out[42]:  Size\_in\_bytes 1695  Size\_MB 1695  Rating 1474  Current Ver 8  Android Ver 2  Type 1  App 0  Category 0  Reviews 0  Installs 0  Price 0  Content Rating 0  Genres 0  Last Updated 0  Installs\_category 0  dtype: int64 |

In

[43]:

*###*

*Plot*

*Missing*

*Values*

sns

.

heatmap(df

.

isnull(),

yticklabels

=

False

,

cbar

=

False

,

cmap

=

'viridis'

)

Out[43]:

<

Axes

:

>

In

[44]:

*#*

*make*

*figure*

*size*

plt

.

figure(figsize

=

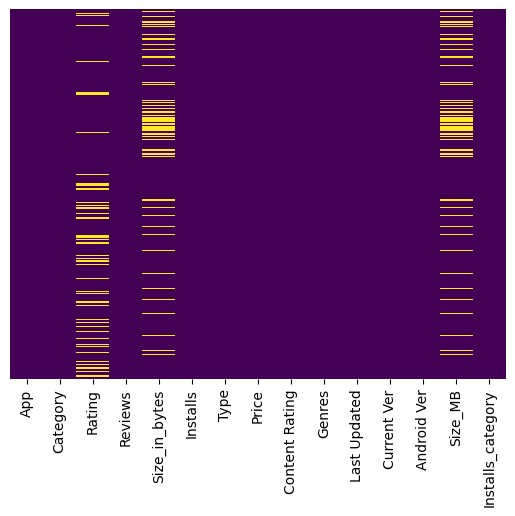
(

16

,

6

))



*#*

*plot*

*the*

*null*

*values*

*by*

*their*

*percentage*

*in*

*each*

*column*

missing\_percentage

=

df

.

isnull()

.

sum()

/

len

(

df

)

\*

100

missing\_percentage

.

plot(kind

=

'bar'

)

*#*

*add*

*the*

*labels*

plt

.

xlabel(

'Columns'

)

plt

.

ylabel(

'Percentage'

)

plt

.

title(

'Percentage

of

Missing

Values

in

each

Column'

)

Out[44]:

Text(0.5,

1.0

,

'Percentage

of

Missing

Values

in

each

Column')

In

[45]:

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

missing\_percentage[missing\_percentage

<

1

]

.

plot(kind

=

'bar'

)

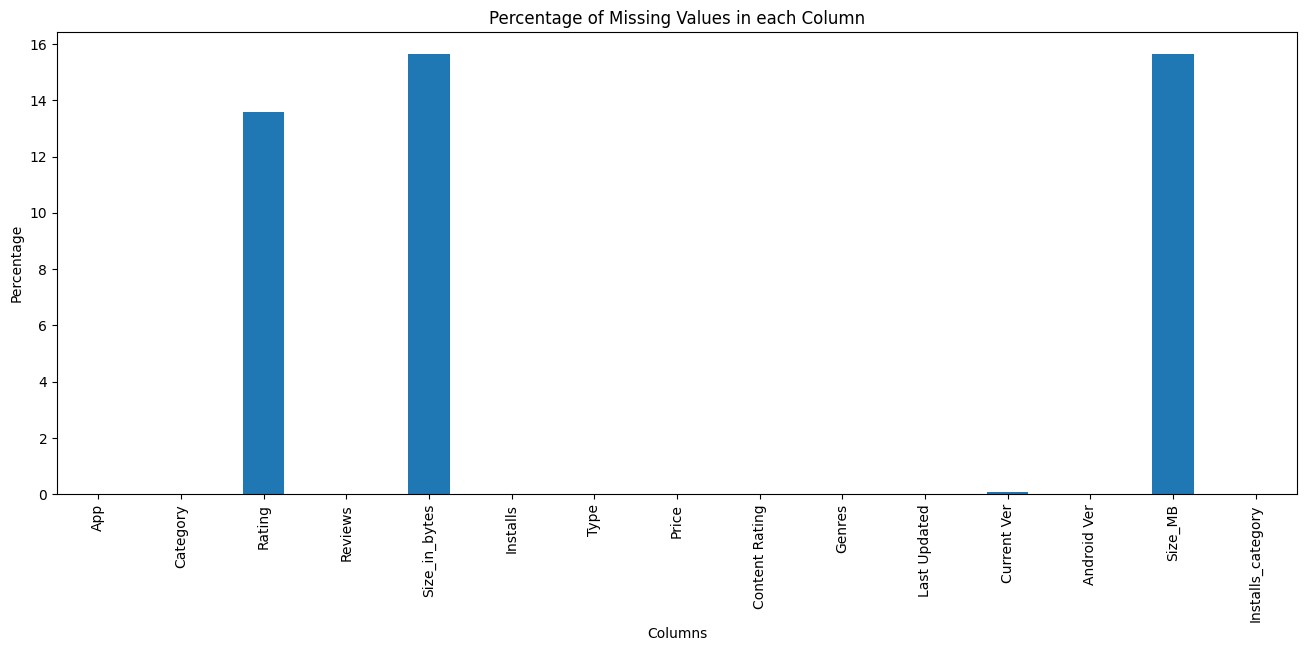
*#*

*plot*

*the*

*null*

*values*



*by*

*their*

*percentage*

*in*

*each*

*column*

plt

.

xlabel(

'Columns'

)

*#*

*add*

*the*

*x-axis*

*labels*

plt

.

ylabel(

'Percentage'

)

*#*

*add*

*the*

*labels*

*for*

*y-axis*

plt

.

title(

'Percentage

of

Missing

Values

in

each

Column

without

Rating

column'

)

*#*

*add*

*the*

*title*

*for*

*the*

*plot*

Out[45]:

Text(0.5,

1.0

,

'Percentage

of

Missing

Values

in

each

Column

without

Rating

column')

Observations:

●

We

have

1695

missing

values

in

the

'Size\_in\_bytes'

and

'Size\_in\_Mb'

columns,

which

is

15.6

%

of

the

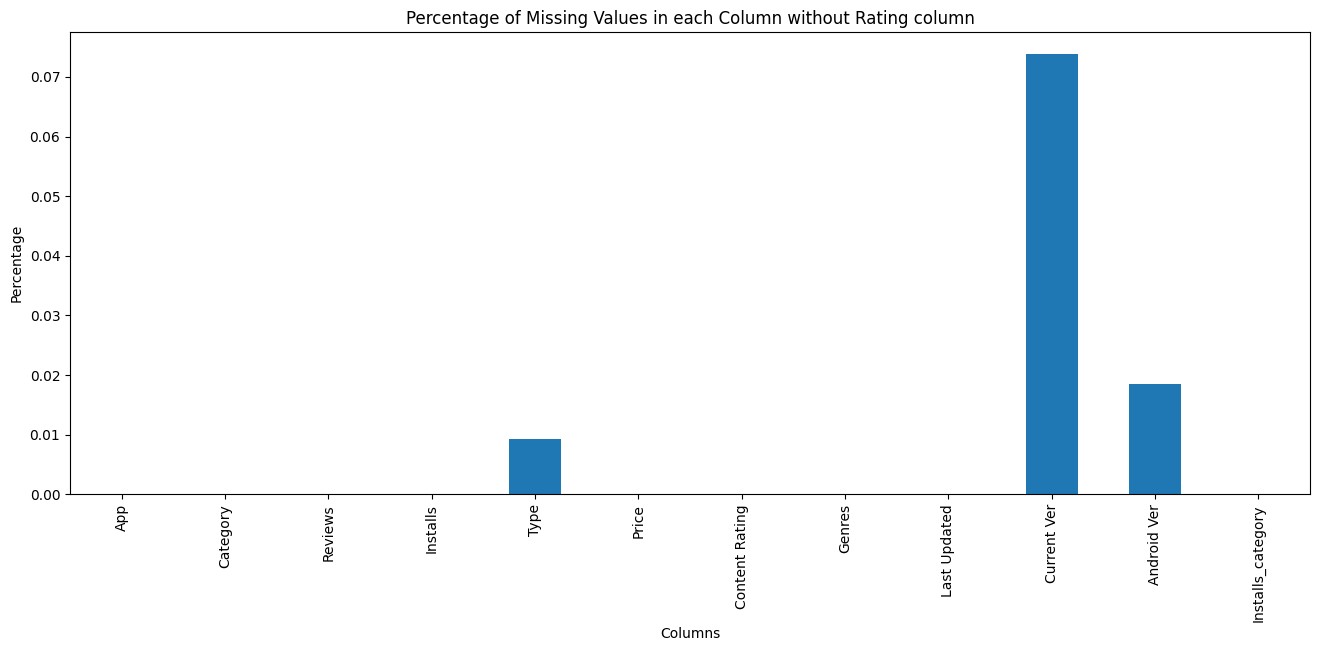
total

values

in

the

column.



●

We

have

1474

missing

values

in

the

'Rating'

column,

which

is

13.6

%

of

the

total

values

in

the

column.

●

We

have

8

missing

value

in

the

'Current

Ver'

column,

which

is

0.07

%

of

the

total

values

in

the

column.

●

We

have

2

missing

values

in

the

'Android

Ver'

column,

which

is

0.01

%

of

the

total

values

in

the

column.

●

We

have

only

1

missing

value

in

Category

,

Type

and

Genres

columns,

which

is

0.009

%

of

the

total

values

in

the

column.

**2.3.**

**Dealing**

**with**

**the**

**missing**

**values**

-

We

can

not

impute

the

Rating

column

as

is

is

directly

linked

with

the

installation

column.

To

test

this

Hypothesis

we

need

to

plot

the

Rating

column

with

the

Installs

and

Size

columns

and

statistically

test

it

using

pearson

correlation

test

.

In

[46]:

df

.

columns

Out[46]:

Index(['App',

'Category',

'Rating',

'Reviews',

'Size\_in\_bytes',

'Installs',

'Type',

'Price',

'Content

Rating',

'Genres',

'Last

Updated',

'Current

Ver',

'Android

Ver',

'Size\_MB',

'Installs\_category'],

dtype='object')

In

[47]:

numeric\_cols

=

[

i

for

i

**in**

df

.

columns

if

df[i]

.

dtype

!=

'object'

]

*#*

*make*

*a*

*list*

*of*

*numeric*

*columns*

In

[48]:

numeric\_cols

.

remove(

"Installs\_category"

)

In

[49]:

numeric\_cols

Out[49]:

[

'Rating',

'Reviews',

'Size\_in\_bytes',

'Installs',

'Price',

'Size\_MB']

In

[50]:

corr

=

df[numeric\_cols]

.

corr()

In

[51]:

corr

Out[51]:

Rating

Reviews

Size\_in\_bytes

Installs

Price

Size\_MB

Rating

1.000000

0.068141

0.083737

0.051355

-0.021903

0.083737

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Reviews | 0.068141 | 1.000000 | 0.238214 | 0.643122 | -0.009667 | 0.238214 | | Size\_in\_bytes | 0.083737 | 0.238214 | 1.000000 | 0.164787 | -0.023007 | 1.000000 | | Installs | 0.051355 | 0.643122 | 0.164787 | 1.000000 | -0.011689 | 0.164787 | | Price | -0.021903 | -0.009667 | -0.023007 | -0.011689 | 1.000000 | -0.023007 | | Size\_MB | 0.083737 | 0.238214 | 1.000000 | 0.164787 | -0.023007 | 1.000000 |   In [52]: plt.figure(figsize=(10, 10)) sns.heatmap(corr, cmap=sns.diverging\_palette(220, 20, as\_cmap=True)) plt.show() |

In

[53]:

*#*

*we*

*can*

*calculate*

*the*

*pearson*

*correlation*

*coefficient*

*using*

*scipy*

*as*

*well*

*as*

*follows*

*#*

*this*

*is*

*to*

*install*

*scipy*

*if*

*you*

*have*

*not*

*done*

*it*

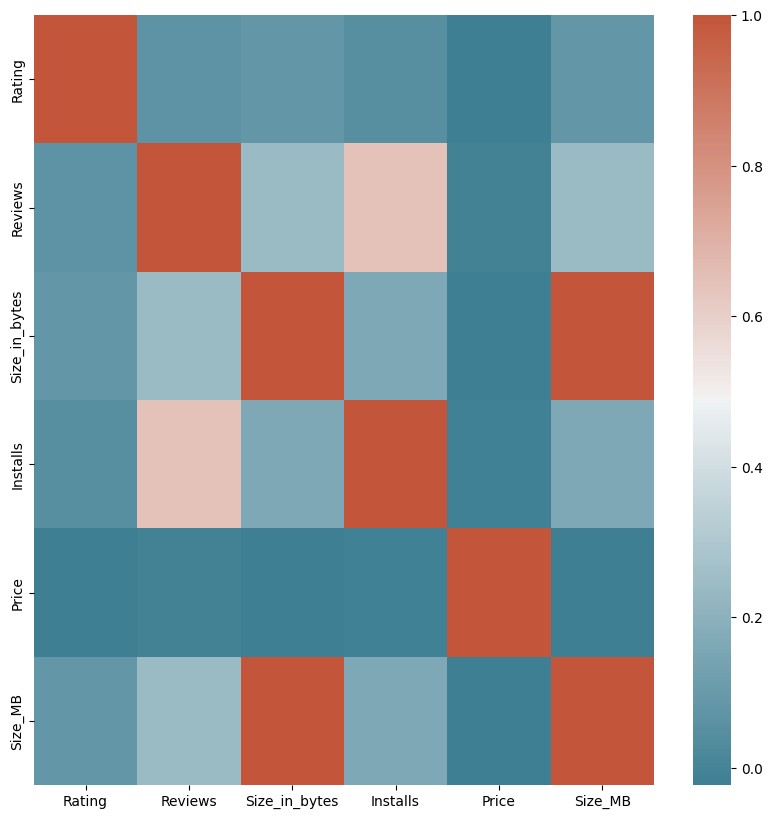
*before*

*#*

*pip*

*install*

*scipy*



|  |
| --- |
| from scipy import stats  *# remove rows containing NaN or infinite values (Important to calculate Pearson's R)* df\_clean = df.dropna()  *# calculate Pearson's R between Rating and Installs* pearson\_r, \_ = stats.pearsonr(df\_clean['Reviews'], df\_clean['Installs']) print(f"Pearson's R between Reviews and Installs: **{**pearson\_r**:**.4f**}**")  Pearson's R between Reviews and Installs: 0.6262  **Observations**   * Lighter color shows the high correlation and darker color shows the low correlation * We can see that the Reviews column has a high correlation with the Installs column, which is 0.64 according to corr(). Which is quite good.   ■ This shows that the more the reviews the more the installs are for one app. If in any case we need to impute reviews we have to think of number of install.  ○ If we have an ap with 2 installs and we imputer the reviews with 1000 or via average reviews then it will be wrong.   * Installs is slightly correlated with Size\_in\_Mb or Size\_in\_bytes , which is 0.16, this also shows us the importance of size and Installs. But we can not depend on it as the Peason correlation is very low. |

|  |
| --- |
| ● Before going ahead, let's remove the rows with missing values in the Current Ver, Android Ver,  Category, Type and Genres columns, as they are very less in number and will not affect our analysis.  In [54]:  *# remove the rows having null values in the 'Current Ver', 'Android Ver', 'Category',*  *'Type' and 'Genres' column* df.dropna(subset=['Current Ver', 'Android Ver', 'Category', 'Type', 'Genres'],  inplace=True)  In [55]:  *# length after removing null values* print(f"Length of the dataframe after removing null values: **{**len(df)**}**")  Length of the dataframe after removing null values: 10829  **Observations**  ● Only Rating and Size\_in\_bytes or Size\_in\_Mb columns are left with missing values.  ■ We know that we have to be carefull while deadling with Rating column, as it is directly linked with the Installs column.  ■ In Size columns we already know about Varies with device values, which we have |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| converted into null values, we do not need to impute at the moment, as every app has different size and nobody can predict that as nearly as possible.  In [56]:  *# use groupby function to find the trend of Rating in each Installs\_category*  df.groupby('Installs\_category')['Rating'].describe()  Out[56]:   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | count | mean | std | min | 25% | 50% | 75% | max | | Installs\_category |  |  |  |  |  |  |  |  | | no | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | | Very low | 81.0 | 4.637037 | 0.845199 | 1.0 | 4.8 | 5.0 | 5.0 | 5.0 | | Low | 1278.0 | 4.170970 | 0.825605 | 1.0 | 3.8 | 4.4 | 4.8 | 5.0 | | Moderate | 1440.0 | 4.035417 | 0.604428 | 1.4 | 3.8 | 4.2 | 4.5 | 5.0 | | More than moderate | 1616.0 | 4.093255 | 0.505619 | 1.6 | 3.9 | 4.2 | 4.5 | 4.9 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | High | 2113.0 | 4.207525 | 0.376594 | 1.8 | 4.0 | 4.3 | 4.5 | 4.9 | | Very High | 2004.0 | 4.287076 | 0.294902 | 2.0 | 4.1 | 4.3 | 4.5 | 4.9 | | Top Notch | 828.0 | 4.374396 | 0.193726 | 3.1 | 4.3 | 4.4 | 4.5 | 4.8 |   In [57]:  df['Rating'].isnull().sum()  Out[57]:  1469  In [58]:  *# in which Install\_category the Rating has NaN values*  df['Installs\_category'].loc[df['Rating'].isnull()].value\_counts()  Out[58]: Installs\_category  Low 880  Very low 453  Moderate 88  More than moderate 31  no 14 |

|  |
| --- |
| High 3  Very High 0  Top Notch 0  Name: count, dtype: int64  In [59]:  *# plot the boxplot of Rating in each Installs\_category* plt.figure(figsize=(16, 6)) *# make figure size* sns.boxplot(x='Installs\_category', y='Rating', hue='Installs\_category', data=df) *# plot the boxplot*  *# add the text of number of null values in each category* plt.text(0, 3.5, 'Null values: 14') plt.text(1, 3.5, 'Null values: 874') plt.text(2, 3.5, 'Null values: 86') plt.text(3, 3.5, 'Null values: 31') plt.text(4, 3.5, 'Null values: 3') plt.text(5, 3.5, 'Null values: 0') plt.text(6, 3.5, 'Null values: 0')  plt.text(7, 3.5, 'Null values: 0')  Out[59]:  Text(7, 3.5, 'Null values: 0') |

In

[60]:

def

fill\_missing\_ratings(df,

category,

fill\_value):

*"""Fills*

*missing*

*rating*

*values*

*in*

*a*

*specified*

*category*

*with*

*a*

*given*

*value.*

*Args:*

*df:*

*The*

*pandas*

*DataFrame*

*containing*

*the*

*data.*

*category:*

*The*

*category*

*to*

*fill*

*missing*

*values*

*for.*

*fill\_value:*

*The*

*value*

*to*

*fill*

*missing*

*ratings*

*with.*

*Returns:*

*The*

*modified*

*DataFrame*

*with*

*filled*

*missing*

*values.*

*"""*

*#*

*Filter*

*the*

*DataFrame*

*for*

*rows*

*where*

*the*

*category*

*matches*

*and*

*rating*

*is*

*missing*

filtered\_df

=

df[(df[

'Installs\_category'

]

==

category)

&

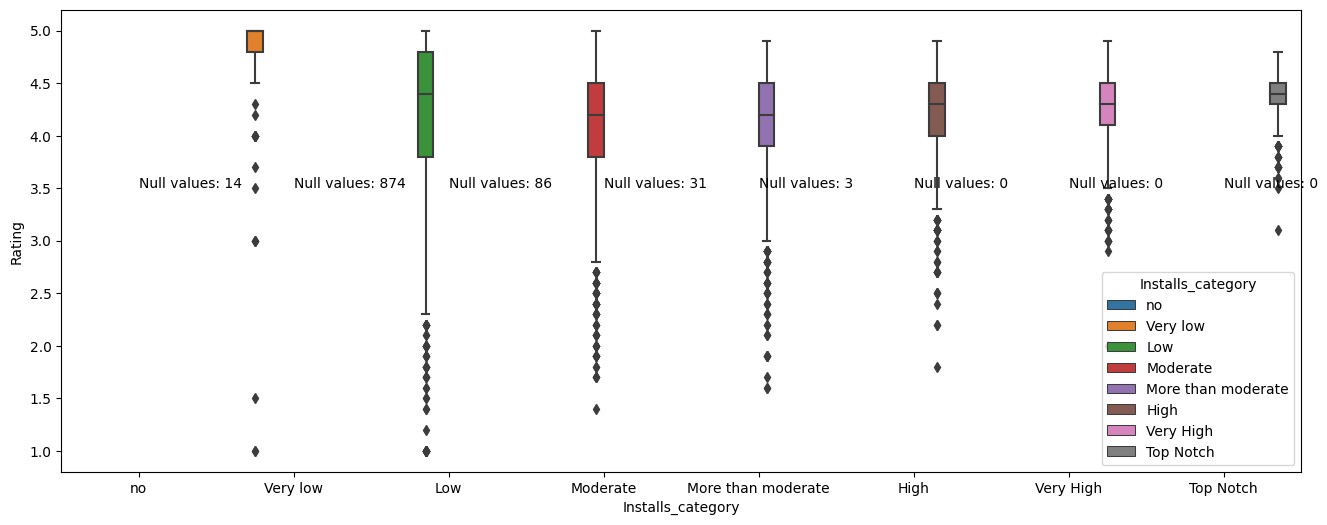
df[

'Rating'

]

.

isnull()]



*#*

*Fill*

*the*

*missing*

*values*

*with*

*the*

*specified*

*value*

df

.

loc[filtered\_df

.

index,

'Rating'

]

=

fill\_value

return

df

In

[61]:

df

=

fill\_missing\_ratings(df,

'Low'

,

4.170970

)

In

[62]:

df

=

fill\_missing\_ratings(df,

'Very

low'

,

4.637037

)

df

=

fill\_missing\_ratings(df,

'Moderate'

,

4.035417

)

df

=

fill\_missing\_ratings(df,

'More

than

moderate'

,

4.093255

)

df

=

fill\_missing\_ratings(df,

'High'

,

4.207525

)

In

[63]:

df

=

fill\_missing\_ratings(df,

'no'

,

0

)

In

[64]:

*#*

*in*

*which*

*Install\_category*

*the*

*Rating*

*has*

*NaN*

*values*

df[

'Installs\_category'

]

.

loc[df[

'Rating'

]

.

isnull()]

.

value\_counts()

Out[64]:

Installs\_category

no

0

Very

low

0

Low

0

Moderate

0

More

than

moderate

0

High

0

Very

High

0

Top

Notch

0

Name:

count,

dtype:

int64

In

[65]:

df[

'Rating'

]

.

isnull()

.

sum()

Out[65]:

0

In

[66]:

*#*

*let's*

*plot*

*the*

*same*

*plots*

*for*

*Reviews*

*column*

*as*

*well*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

boxplot(x

=

'Installs\_category'

,

y

=

'Reviews'

,

data

=

df)

*#*

*plot*

*the*

*boxplot*

Out[66]:

<

Axes

:

xlabel='Installs\_category',

ylabel='Reviews'>

In

[67]:

*#*

*let's*

*plot*

*the*

*same*

*plots*

*for*

*Reviews*

*column*

*as*

*well*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

boxplot(x

=

'Installs\_category'

,

y

=

np

.

log10(df[

'Reviews'

])

,

data

=

df)

*#*

*plot*

*the*

*boxplot*

Out[67]:

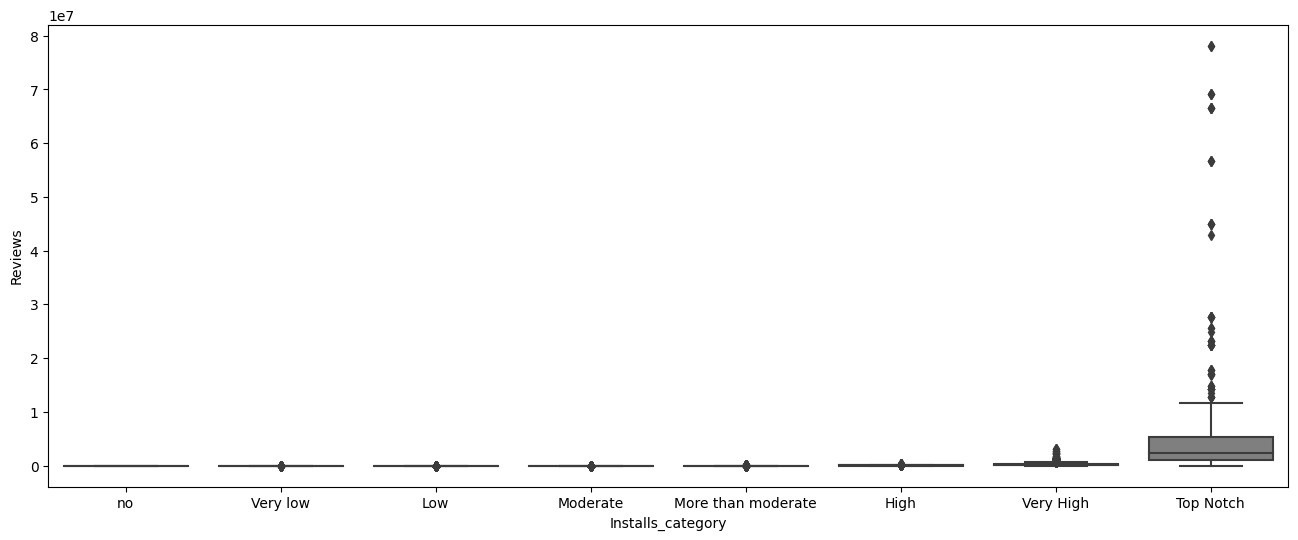
<

Axes

:

xlabel='Installs\_category',

ylabel='Reviews'>



●

We

also

draw

the

scatter

plot

of

the

Rating

and

Review

columns

with

the

Installs

column

In

[68]:

*#*

*Draw*

*a*

*scatter*

*plot*

*between*

*Rating,*

*Reviews*

*and*

*Installs*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

scatterplot(x

=

'Rating'

,

y

=

'Reviews'

,

hue

=

'Installs\_category'

,

data

=

df)

*#*

*plot*

*the*

*scatter*

*plot*

Out[68]:

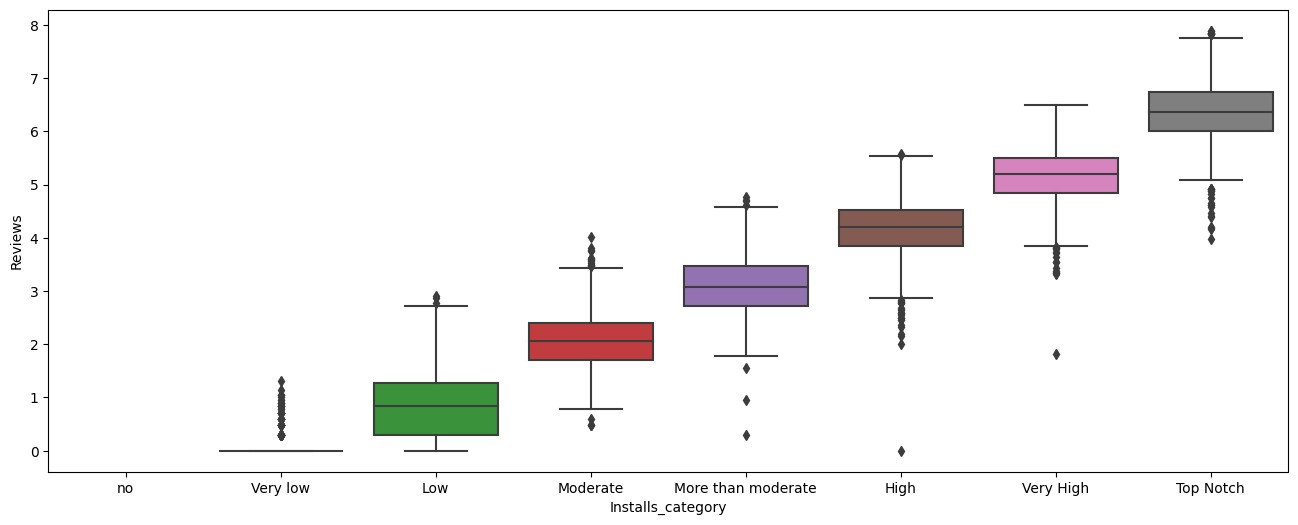
<

Axes

:

xlabel='Rating',

ylabel='Reviews'>



●

It

doesn't

show

any

trend,

because,

you

should

know

that

Rating

is

a

categorical

variable

)

(

Ordinal

and

Reviews

is

a

continuous

variable,

therefore,

we

can

not

plot

them

together.

●

Let's

try

with

Reviews

and

Installs

In

[69]:

*#*

*plot*

*reviews*

*and*

*installs*

*in*

*a*

*scatter*

*plot*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

scatterplot(x

=

'Reviews'

,

y

=

'Installs'

,

data

=

df)

*#*

*plot*

*the*

*scatter*

*plot*

Out[69]:

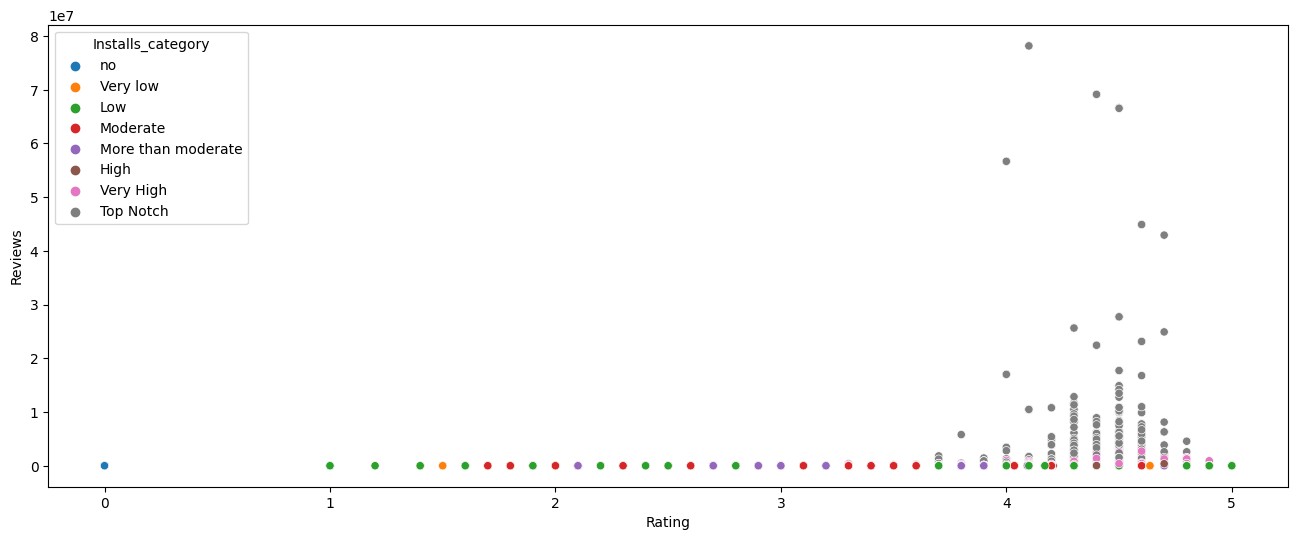
<

Axes

:

xlabel='Reviews',

ylabel='Installs'>



●

We

did

not

see

any

trend

and

the

issue

is

we

need

to

normalize

the

data

before

plotting

it,

let's

try

with

log

transformation

In

[70]:

*#*

*plot*

*reviews*

*and*

*installs*

*in*

*a*

*scatter*

*plot*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

scatterplot(x

=

np

.

log10(df[

'Reviews'

])

,

y

=

np

.

log10(df[

'Installs'

])

,

data

=

df)

*#*

*plot*

*the*

*scatter*

*plot*

Out[70]:

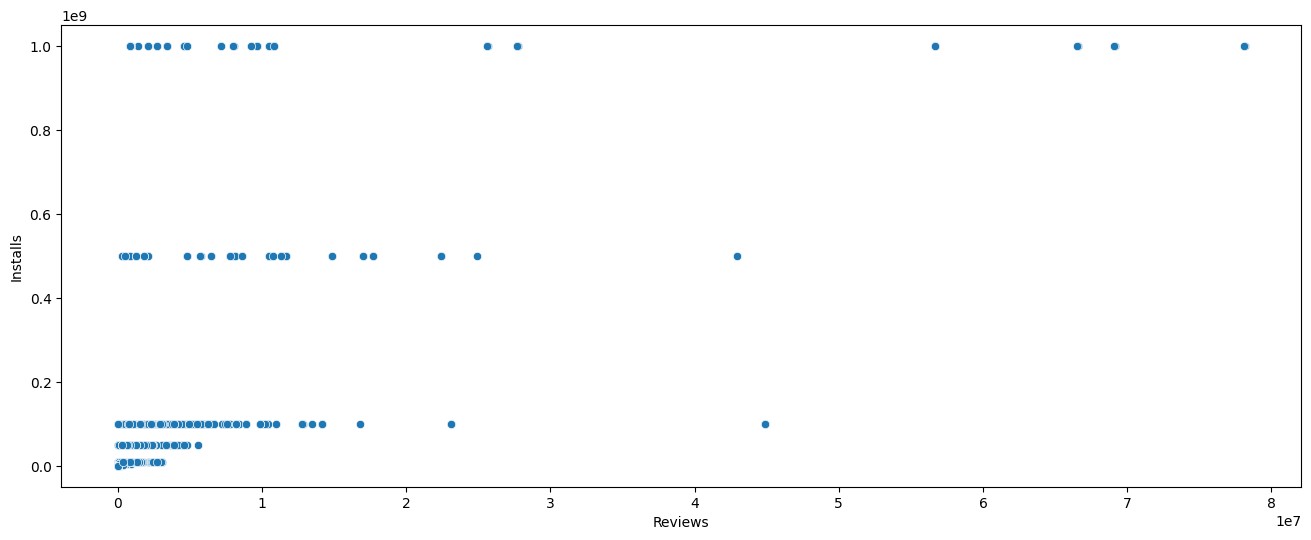
<

Axes

:

xlabel='Reviews',

ylabel='Installs'>



●

Now

we

see

a

slight

trend

but

still

the

issue

is

installs

were

given

in

a

factorial

manner,

as

10+

,

20+

,

1000+

etc,

and

these

are

not

continuous

number

but

Discreet

one,

therefore,

we

can

only

see

a

slight

trends

here.

Let's

plot

a

line

plot

to

see

the

trend.

In

[71]:

*#*

*plot*

*reviews*

*and*

*installs*

*in*

*a*

*scatter*

*plot*

*with*

*trend*

*line*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

lmplot(x

=

'Reviews'

,

y

=

'Installs'

,

data

=

df)

*#*

*plot*

*the*

*scatter*

*plot*

*with*

*trend*

*line*

Out[71]:

<

seaborn.axisgrid.FacetGrid

at

0

x7cfd283b47f

0>

<

Figure

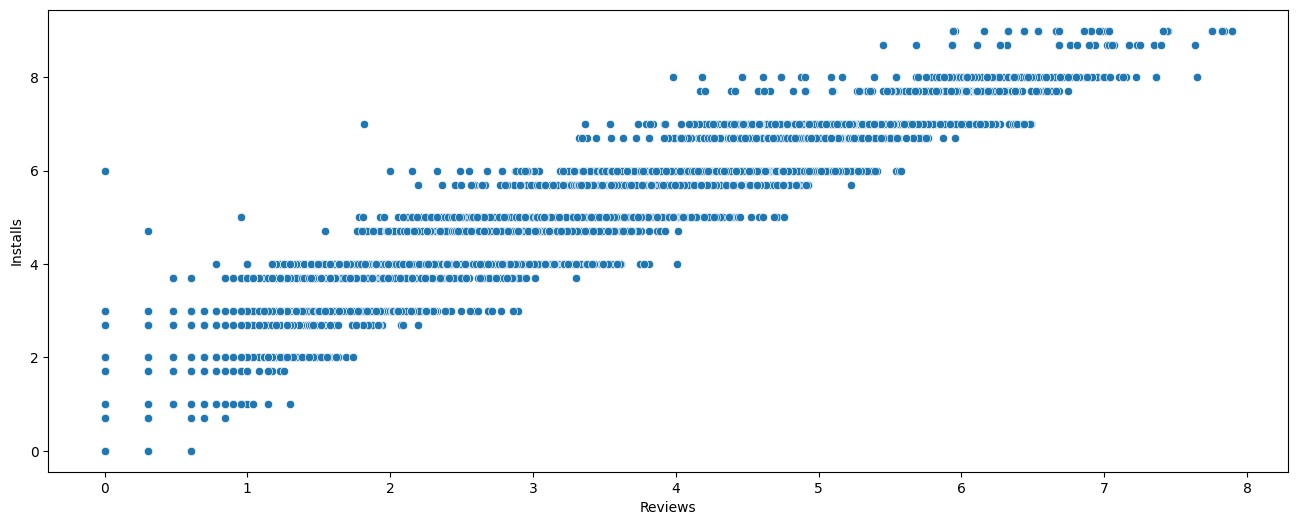
size

1600x600

with

0

Axes>



●

Here,

we

can

see

a

nice

trend,

which

shows

that

number

of

Reviews

increases

with

the

number

of

Installs,

which

is

quite

obvious.

**Observation**

-

We

can

see

that

most

of

the

null

values

from

Rating

column

are

no

-

Moderate

Installation

apps,

which

make

sense

that

if

the

app

has

less

installations,

it

has

less

Rating

and

review.

●

But

wait,

we

have

to

check

for

the

duplicates

as

well,

as

they

can

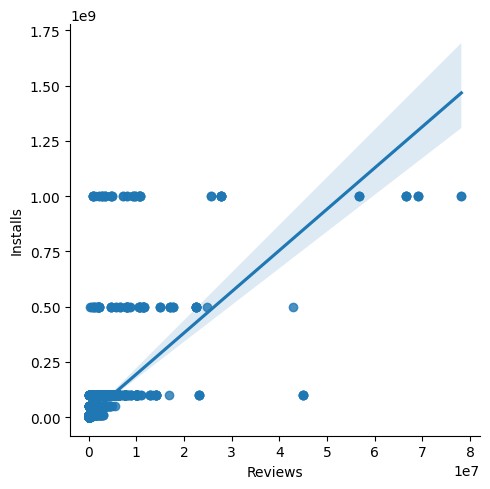
affect

our

analysis.

2.3.

**Duplicates**



|  |
| --- |
| * Removing duplicates is one of the most important part of the data wrangling process, we must remove the duplicates in order to get the correct insights from the data. * If you do not remove duplicates from a dataset, it can lead to incorrect insights and analysis. * Duplicates can skew statistical measures such as mean, median, and standard deviation, and can also lead to over-representation of certain data points. * It is important to remove duplicates to ensure the accuracy and reliability of your data analysis.   In [72]:  *# find duplicate if any*  df.duplicated().sum()  Out[72]:  483  In [73]:  *# let's check for number of duplicates*  for col **in** df.columns:  print(f"Number of duplicates in **{**col**}** column are: **{**df[col].duplicated().sum()**}**")  Number of duplicates in App column are: 1181  Number of duplicates in Category column are: 10796  Number of duplicates in Rating column are: 10784  Number of duplicates in Reviews column are: 4830  Number of duplicates in Size\_in\_bytes column are: 10373  Number of duplicates in Installs column are: 10809 |

|  |
| --- |
| Number of duplicates in Type column are: 10827  Number of duplicates in Price column are: 10737  Number of duplicates in Content Rating column are: 10823  Number of duplicates in Genres column are: 10710  Number of duplicates in Last Updated column are: 9453  Number of duplicates in Current Ver column are: 7998  Number of duplicates in Android Ver column are: 10796  Number of duplicates in Size\_MB column are: 10373  Number of duplicates in Installs\_category column are: 10821  In [74]:  *# print the number of duplicates in df* print(f"Number of duplicates in df are: **{**df.duplicated().sum()**}**")  Number of duplicates in df are: 483  In [75]:  *# remove the duplicates* df.drop\_duplicates(inplace=True)  ● Now we have removed 483 duplicates from the dataset. and have 10346 rows left. |

|  |
| --- |
| 1. **Insights from Data**   **3.1. Which category has the highest number of apps?**  In [76]:  *# which category has highest number of apps* df['Category'].value\_counts().head(10) *# this will show the top 10 categories with*  *highest number of apps*  Out[76]:  Category  FAMILY 1939  GAME 1121  TOOLS 841  BUSINESS 427  MEDICAL 408  PRODUCTIVITY 407  PERSONALIZATION 386  LIFESTYLE 373  COMMUNICATION 366  FINANCE 360  Name: count, dtype: int64  **3.2. Which category has the highest number of installs?**  In [77]:  *# category with highest number of Installs* |

|  |
| --- |
| df.groupby('Category')['Installs'].sum().sort\_values(ascending=False).head(10)  Out[77]:  Category  GAME 31544024415  COMMUNICATION 24152276251  SOCIAL 12513867902  PRODUCTIVITY 12463091369  TOOLS 11452271905  FAMILY 10041632405  PHOTOGRAPHY 9721247655  TRAVEL\_AND\_LOCAL 6361887146  VIDEO\_PLAYERS 6222002720  NEWS\_AND\_MAGAZINES 5393217760  Name: Installs, dtype: int64  **3.3. Which category has the highest number of reviews?**  In [78]:  *# Category with highest number of Reviews*  df.groupby('Category')['Reviews'].sum().sort\_values(ascending=False).head(10)  Out[78]:  Category  GAME 1415536650 |

|  |
| --- |
| COMMUNICATION 601273552  SOCIAL 533576829  FAMILY 396771746  TOOLS 273181033  PHOTOGRAPHY 204297410  VIDEO\_PLAYERS 110380188  PRODUCTIVITY 102554498  SHOPPING 94931162  PERSONALIZATION 75192744  Name: Reviews, dtype: int64  **3.4. Which category has the highest rating?**  In [79]:  *# Category with highest average Rating*  df.groupby('Category')['Rating'].mean().sort\_values(ascending=False).head(10)  Out[79]:  Category  EVENTS 4.394346  EDUCATION 4.373794  BOOKS\_AND\_REFERENCE 4.358435  PERSONALIZATION 4.322099  ART\_AND\_DESIGN 4.298885  GAME 4.281926  HEALTH\_AND\_FITNESS 4.273890 |

PARENTING

4.259759

SHOPPING

4.253376

SPORTS

4.253041

Name:

Rating,

dtype:

float64

In

[80]:

*#*

*plot*

*the*

*rating*

*distribution*

plt

.

figure(figsize

=

(

16

,

6

))

*#*

*make*

*figure*

*size*

sns

.

kdeplot(df[

'Rating'

]

,

color

=

"blue"

,

shade

=

True

)

*#*

*plot*

*the*

*distribution*

*plot*

Out[80]:

<

Axes

:

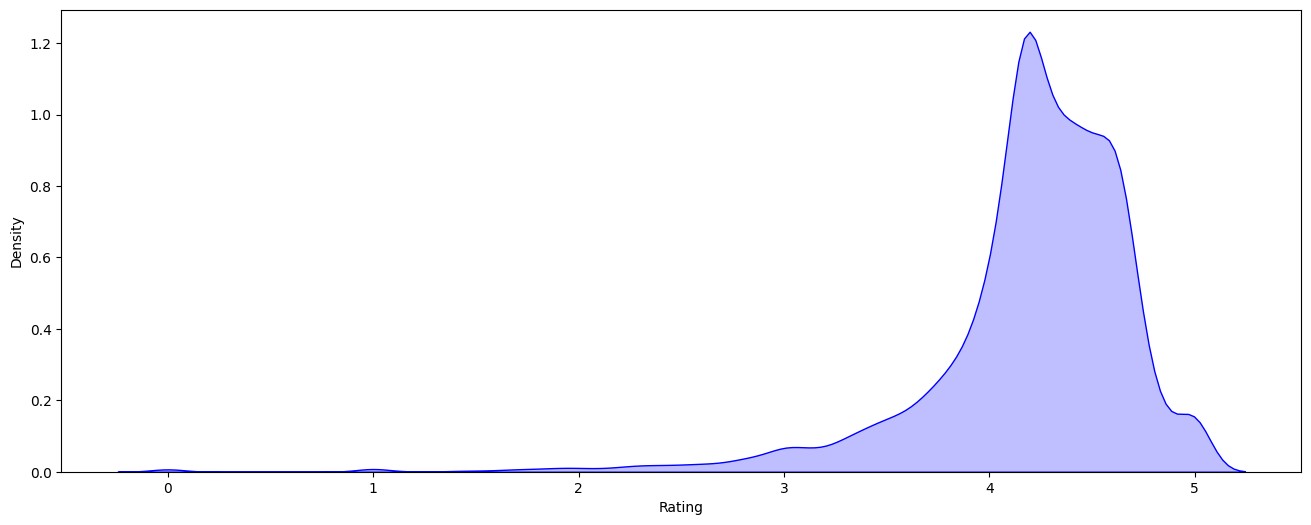
xlabel='Rating',

ylabel='Density'>

In

[

]:



[Reference](https://github.com/Datawithabhishek/Google-Play-Store-Data-Analysis) [link](https://github.com/Datawithabhishek/Google-Play-Store-Data-Analysis)