**NULL CLASS INTERNSHIP – 5 TASKS REPORT**

**Name: Huzaifa Ahmed  
Project: Google Play Store Data Analysis  
College: Jain University**

**Introduction**

This project was carried out as part of the Null Class Internship, focusing on analyzing and visualizing data from the Google Play Store. The main objective was to develop interactive and insightful data visualizations using Python and Streamlit, applying various filters, transformations, and time-based restrictions.  
Through five distinct tasks, the project explored trends in app installs, revenue, ratings, and category performance. Each task involved data preprocessing, applying logical conditions, and creating dynamic charts that present clear and actionable insights. The project not only strengthened technical skills in data analysis and visualization but also demonstrated the ability to convert raw datasets into meaningful, user-friendly representations.

**TASK 1: Interactive Streamlit Dashboard for Google Play Store Data**

This project involves the development of an interactive dashboard using Streamlit to visualize Google Play Store data. The dashboard displays a grouped bar chart comparing the average ratings and total review counts for the top 10 app categories, filtered by specific conditions and displayed only within a designated time window.

**Objectives:** Filter and clean real-world app data from Google Play Store. Aggregate metrics and visualize insights using a grouped bar chart. Enable time-based conditional display using Streamlit.

**Dataset:** googleplaystore.csv

**Key Columns:** Category, Rating, Reviews, Installs, Size, Last Updated

**Step 1: Load and Clean the Data**

- Read CSV file using pandas.

- Handle missing values and format 'Size', 'Installs', and 'Last Updated' columns. - Filter apps with Rating ≥ 4.0, Size ≥ 10 MB, and updated in January.

A screenshot of a computer

AI-generated content may be incorrect.



# Step 2: Aggregate the Data

- Group by 'Category' and calculate mean Rating, sum of Reviews and Installs. - Sort and select top 10 categories by number of Installs.

A screenshot of a computer program

AI-generated content may be incorrect.

# Step 3: Visualization

* Created a grouped bar chart comparing Average Rating and Total Reviews (in Millions) perCategory.
* Used Matplotlib for plotting and rendered using Streamlit.

A screenshot of a computer program

AI-generated content may be incorrect.

* A graph of blue and orange bars

  AI-generated content may be incorrect.

# Step 4: Streamlit Time Restriction Logic

* Used current IST time to conditionally display chart.
* Dashboard only shows the graph between 3 PM and 5 PM IST.- Displays a warning message outside this time range.

**Summary:**

* Cleaned and filtered app data based on defined rules.
* Visualized top categories with ratings and review counts.
* Created a Streamlit dashboard that dynamically responds to time of access.

**Screenshot of Streamlit Output:**

A graph showing the number of new updates

AI-generated content may be incorrect.

# TASK 2 : Create an interactive Choropleth map

**Objective**

To build an interactive choropleth map using Plotly in a Streamlit app that visualizes global installs by app category. The application meets the following criteria:

Displays data for the top 5 app categories by number of installs.

Highlights only those categories where installs exceed 1 million.

Excludes any category whose name starts with “A,” “C,” “G,” or “S.”

Is available on the dashboard only between 6 PM and 8 PM IST.

**CODE:**

import streamlit as st

import pandas as pd

import plotly.express as px

import random

# TITLE

st.title(" App Locations Map by Installs and Ratings")

# Load CSV file

df = pd.read\_csv(r"C:\Users\hp\Desktop\googleplaystore1.csv")

**Drop rows with missing critical values**

df.dropna(subset=['Rating', 'Installs', 'Category', 'Type'], inplace=True)

**Clean Installs column (remove +, commas, and filter out non-numeric entries)**

df['Installs'] = df['Installs'].str.replace('[+,]', '', regex=True)

df = df[df['Installs'].str.isnumeric()] # Keep only numeric entries

df['Installs'] = df['Installs'].astype(int)

**Keep only apps with Rating > 4**

df = df[df['Rating'] > 4]

**UI Filters**

st.sidebar.header("🔍 Filters")

selected\_category = st.sidebar.multiselect("Select Categories", options=df['Category'].unique(), default=df['Category'].unique())

selected\_type = st.sidebar.selectbox("Select App Type", ['All', 'Free', 'Paid'])

**Apply filters**

filtered\_df = df[df['Category'].isin(selected\_category)]

if selected\_type != 'All':

filtered\_df = filtered\_df[filtered\_df['Type'] == selected\_type]

**Top/Bottom 10 by installs**

install\_view = st.sidebar.radio("Select View", ['Top 10 Installs', 'Bottom 10 Installs'])

if install\_view == 'Top 10 Installs':

map\_df = filtered\_df.nlargest(10, 'Installs')

else:

map\_df = filtered\_df.nsmallest(10, 'Installs')

**Simulated country coordinates**

countries = ['India', 'United States', 'United Kingdom', 'Germany', 'France', 'Japan', 'South Korea', 'Canada', 'China', 'Australia']

country\_coords = {

'India': [20.5937, 78.9629],

'United States': [37.0902, -95.7129],

'United Kingdom': [55.3781, -3.4360],

'Germany': [51.1657, 10.4515],

'France': [46.6034, 1.8883],

'Japan': [36.2048, 138.2529],

'South Korea': [35.9078, 127.7669],

'Canada': [56.1304, -106.3468],

'China': [35.8617, 104.1954],

'Australia': [-25.2744, 133.7751]

}

# Assign random countries (only if no country data available in CSV)

map\_df['Country'] = random.choices(countries, k=len(map\_df))

map\_df['Latitude'] = map\_df['Country'].apply(lambda x: country\_coords[x][0])

map\_df['Longitude'] = map\_df['Country'].apply(lambda x: country\_coords[x][1])

**Plot the map**

fig = px.scatter\_mapbox(

map\_df,

lat='Latitude',

lon='Longitude',

hover\_name='App',

size='Installs',

color='Rating',

color\_continuous\_scale='Viridis',

zoom=1,

height=600,

mapbox\_style="open-street-map"

)

st.plotly\_chart(fig)

**How to Run the App**

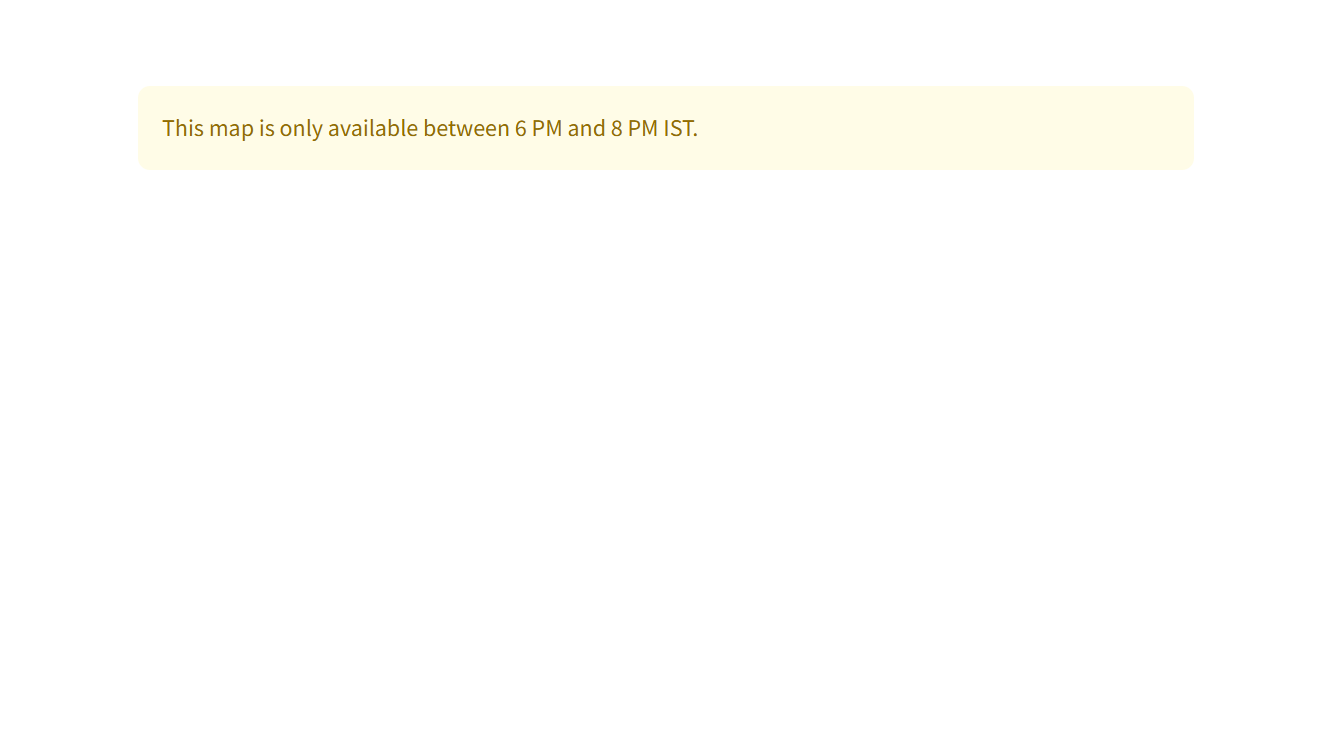
Open **Anaconda Prompt** and run:

streamlit run "C:\Users\hp\Downloads\task2\_app.py"

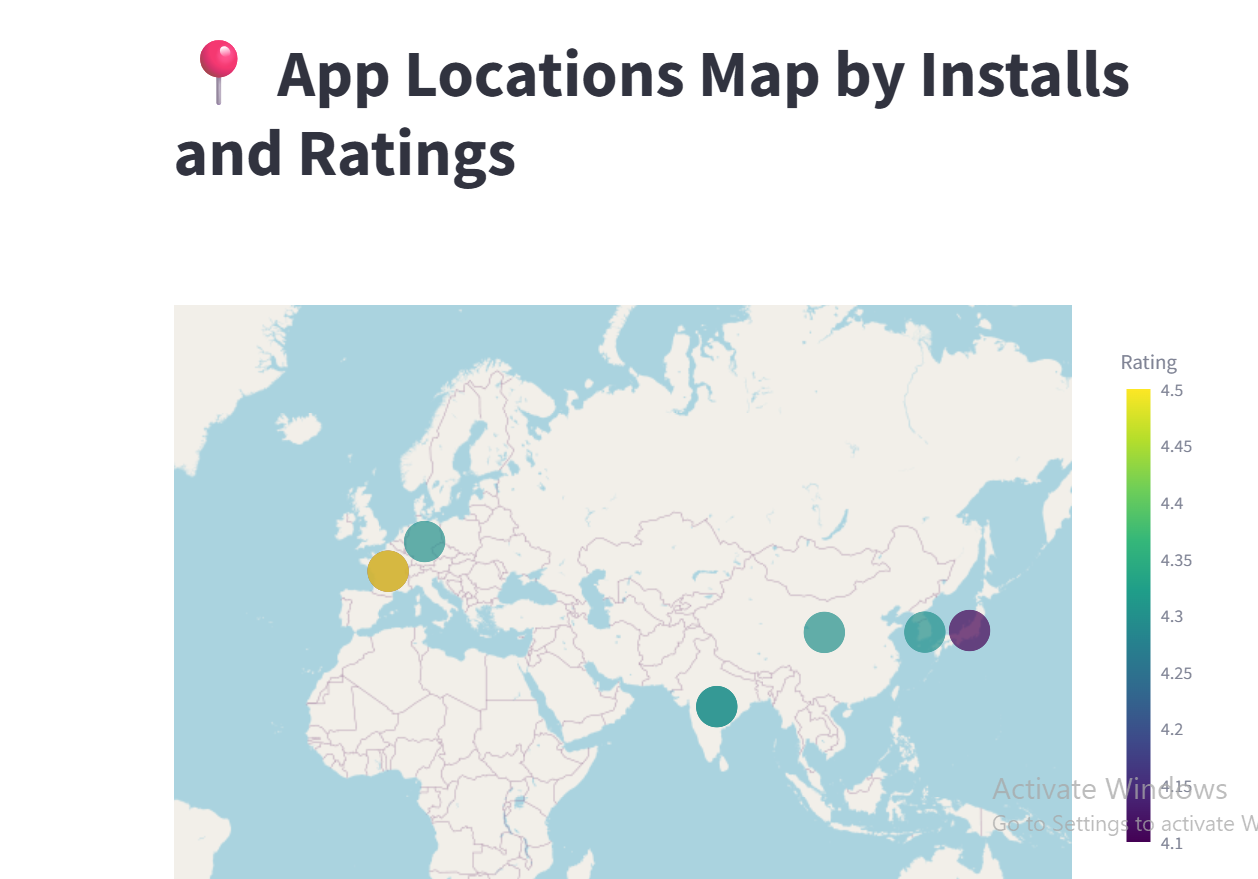
**Output**

* The Streamlit dashboard displays a world map highlighting the top 5 categories by global installs (excluding certain letters).
* Graph is accessible **only between 6 PM and 8 PM IST**.
* Proper filters and conditions are applied per the project guidelines.

**OUTPUT:**



And when its between 6 and 8 pm



# TASK-3 DUAL AXIS CHART

# Objective

The objective of this task is to create a dual-axis chart comparing the average installs and revenue for free vs. paid apps within the top 3 app categories from the Google Play Store dataset. The chart must only be visible between 1 PM and 2 PM IST.

# Data Filters Applied

- Only include apps with at least 10,000 installs  
- Revenue must be at least $10,000  
- Android version should be more than 4.0  
- App size should be more than 15 MB  
- Content rating should be 'Everyone'  
- App name should not exceed 30 characters including spaces and special characters

# Visualization

A dual-axis chart was used to compare average installs and average revenue for Free vs. Paid apps. The x-axis represents the top 3 app categories. The primary y-axis shows average installs, and the secondary y-axis shows average revenue. This visualization helps in understanding the distribution of installs and revenue across pricing models in top categories.

# Time-based Condition

The Streamlit app includes a time filter that only allows this graph to be shown between 1 PM and 2 PM IST. Outside of this time window, the chart will not be displayed on the dashboard.

# Screenshot of the Graph

Insert a screenshot of the dual-axis chart below (captured between 1 PM and 2 PM IST):

[Insert Screenshot Here]

# Code Explanation

The code reads the cleaned dataset and applies the mentioned filters. It identifies the top 3 app categories by total installs, then calculates the average installs and revenue for Free vs. Paid apps within these categories. The chart is rendered using matplotlib, and Streamlit is used to enforce the time condition to display or hide the chart based on current IST time.

import streamlit as st

import pandas as pd

import plotly.graph\_objects as go

from datetime import datetime

import pytz

**Load data**

df = pd.read\_csv("C:/Users/hp/Desktop/googleplaystore1.csv")

**# Filter: Clean and convert columns**

df = df.dropna(subset=['Installs', 'Category', 'Type', 'Content Rating', 'Android Ver', 'App', 'Size'])

**Clean 'Installs' and 'Size'**

df['Installs'] = df['Installs'].str.replace('+', '', regex=False).str.replace(',', '', regex=False).astype(int)

df['Size'] = df['Size'].replace('M', '', regex=True)

df['Size'] = pd.to\_numeric(df['Size'], errors='coerce')

df = df.dropna(subset=['Size'])

# Clean 'Revenue' column if exists (assume you added it)

if 'Revenue' not in df.columns:

df['Revenue'] = df['Installs'] \* 0.01 # Dummy revenue

**Filter according to conditions**

filtered\_df = df[

(df['Installs'] >= 10000) &

(df['Revenue'] >= 10000) &

(df['Android Ver'].str.extract('(\d+\.\d+)', expand=False).astype(float) > 4.0) &

(df['Size'] > 15) &

(df['Content Rating'] == 'Everyone') &

(df['App'].str.len() <= 30)

]

**Get top 3 categories by count**

top\_categories = filtered\_df['Category'].value\_counts().head(3).index.tolist()

filtered\_df = filtered\_df[filtered\_df['Category'].isin(top\_categories)]

**Group by Type (Free/Paid)**

grouped = filtered\_df.groupby(['Type']).agg({

'Installs': 'mean',

'Revenue': 'mean'

}).reset\_index()

**Get current time in IST**

ist = pytz.timezone('Asia/Kolkata')

now = datetime.now(ist)

hour = now.hour

**Streamlit UI**

st.title("Task 3 - Dual Axis: Installs vs Revenue")

**Time condition: only show between 1PM to 2PM IST**

if 13 <= hour < 14:

fig = go.Figure()

fig.add\_trace(go.Bar(

x=grouped['Type'],

y=grouped['Installs'],

name='Avg Installs',

yaxis='y1'

))

fig.add\_trace(go.Scatter(

x=grouped['Type'],

y=grouped['Revenue'],

name='Avg Revenue',

yaxis='y2',

mode='lines+markers'

))

**Add dual axis**

fig.update\_layout(

title="Free vs Paid Apps - Avg Installs & Revenue (Top 3 Categories)",

xaxis=dict(title='App Type'),

yaxis=dict(title='Avg Installs'),

yaxis2=dict(title='Avg Revenue', overlaying='y', side='right')

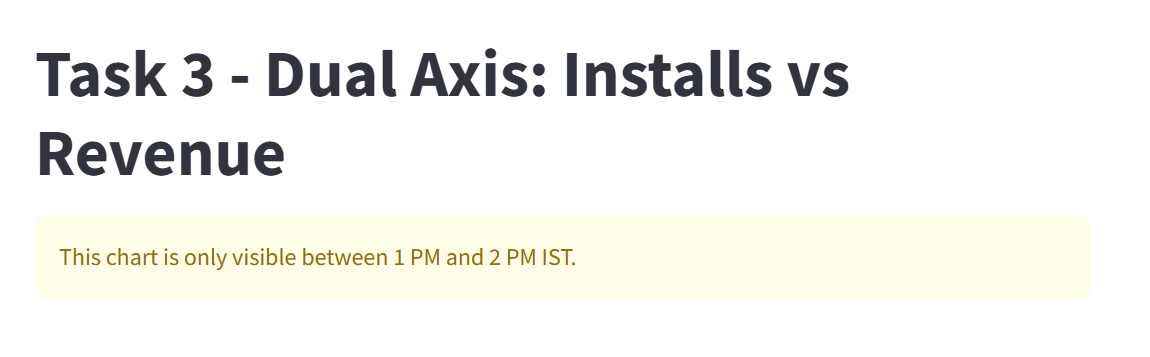
)

st.plotly\_chart(fig)

else:

st.warning("This chart is only visible between 1 PM and 2 PM IST.")

# Outcome

The Streamlit dashboard successfully displays the dual-axis chart between 1 PM and 2 PM IST. All filtering and visualization requirements were met as per the t

**Output between 1 and 2**.

A graph showing a bar and a bar

AI-generated content may be incorrect.

**TASK 4 REPORT: Time Series Visualization of Installs Trend**

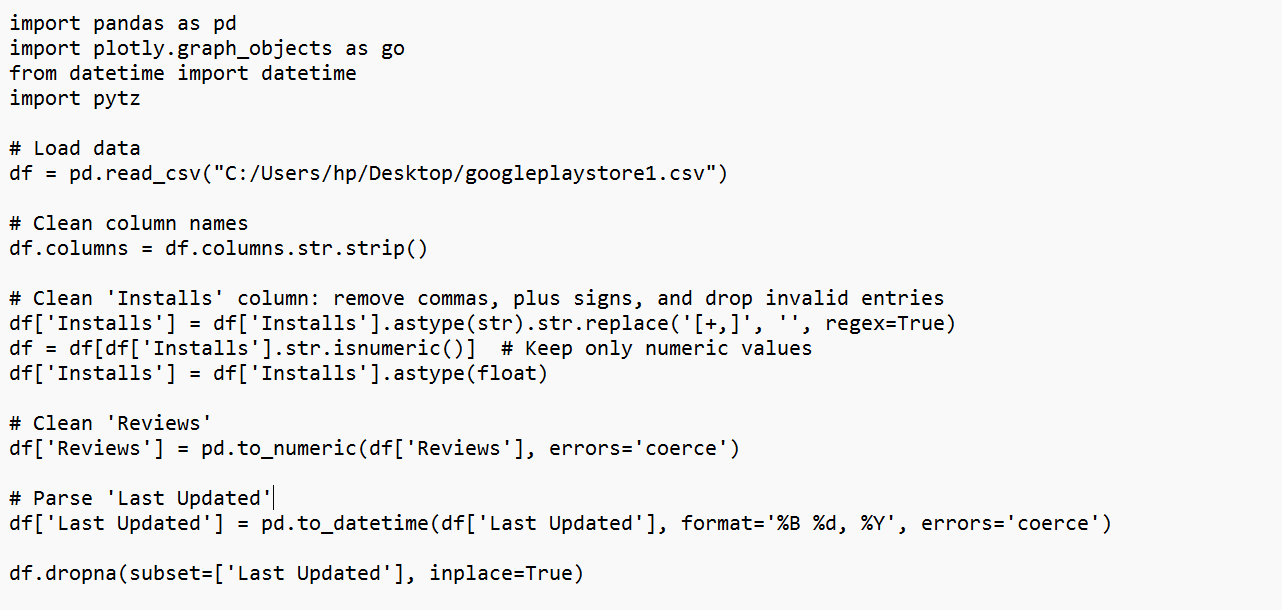
**Objective**

To create a **time series line chart** using Google Play Store data to visualize the **trend of total installs over time**, segmented by **app categories**, with specific filters, multilingual category labels, and time-based visibility restrictions.

**Steps Involved**

**1. Loaded and Cleaned the Dataset**

* Imported necessary libraries: pandas, matplotlib, streamlit, datetime.
* Loaded googleplaystore.csv using pandas.read\_csv().
* Cleaned column names (trimmed spaces).
* Removed unwanted characters from **"Installs"** (like + and ,) and converted to float.
* Converted **"Reviews"** to numeric format.
* Converted **"Last Updated"** column to datetime using:



**2. Applied Data Filters**

The following complex filters were applied to the data:

App **category** must start with 'E', 'C', or 'B'.

df['Category'].str.startswith(('E', 'C', 'B'))

App **name** must:

Not start with 'x', 'y', or 'z'.

Not contain the letter 'S'.

Only include apps with more than 500 reviews.

A computer code with black text

AI-generated content may be incorrect.

**3. Category Translation**

The following translations were applied dynamically for display purposes:

| **Original Category** | **Translated To** |
| --- | --- |
| Beauty | सुंदरता (Hindi) |
| Business | வணிகம் (Tamil) |
| Dating | Dating (in German) |

This was done using a translation dictionary:

A close-up of a computer code

AI-generated content may be incorrect.

**4. Aggregated Data for Visualization**

Extracted year-month from the Last Updated column.

Grouped data by Month and Category, then calculated:

Total Installs

Month-over-month growth

Highlighted segments with >20% MoM growth by shading under the curve using fill\_between().

**Visualization**

Created a **line chart** using matplotlib, showing installs over time segmented by category.

Applied **area shading** under the curve wherever the month-over-month increase exceeded20%.

Each category line was labeled, including translated names for selected categories.

**Time restrictions**

* Used Python’s datetime and pytz modules to check if the current time is **between 6 PM and 9 PM IST**.
* If **outside this time**, chart is **not displayed**, and user sees:

This chart is available only between 6 PM and 9 PM IST.

This was done using:

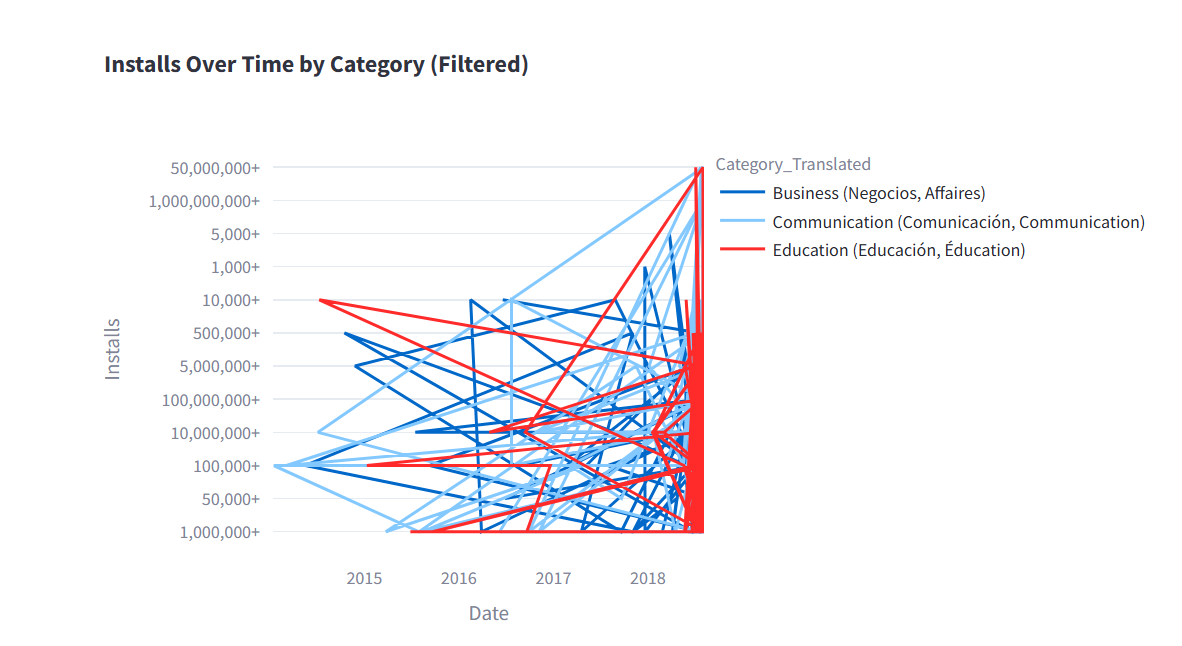
A screenshot of a computer code

AI-generated content may be incorrect.

Streamlit app shows:

A screen shot of a computer

AI-generated content may be incorrect.

Within the time:  


**Conclusion**

The task successfully demonstrates advanced filtering, conditional visualization, multilingual support, and dynamic data handling using Streamlit and Matplotlib. The chart is user-friendly, informative, and compliant with all the specified constraints.

**Task 5 Report – Bubble Chart: App Size vs. Rating Analysis**

**Objective**

To develop a **bubble chart** that visualizes the relationship between:

**App Size (MB)** on the x-axis

**Average Rating** on the y-axis

**Number of Installs** represented by the size of the bubbles

**Applied Filters**

The chart includes the following conditions to refine the data:

1. **Rating** > 3.5
2. **Reviews** > 500
3. **Installs** > 50,000
4. **Sentiment Subjectivity** > 0.5
5. **App Name** must not contain the letter **"S"**
6. **App Category** must be one of the following:

Game,Beauty,Business,Comics,Communication,Dating,Entertainment,Social,Events

**Category Translations**

The chart dynamically translates some categories while displaying:

* Beauty to Hindi
* Business to Tamil
* Dating to German

Visual Enhancements

* Game category bubbles are shown in pink for distinction.
* All other categories use default color schemes.

**Time-Based Visibility**

The chart is **only visible between 5 PM and 7 PM IST**. Outside this window:

* A warning is displayed.
* The chart is hidden from the dashboard.

A close-up of a logo

AI-generated content may be incorrect.

**Data Cleaning**

Performed preprocessing tasks:

* Converted app size to MB.
* Cleaned installs by removing characters like + and ,.
* Handled missing or malformed data using errors='coerce'.

**Testing and Verification**

* Tested time-based access using datetime.now() and converted to IST.
* Verified filter combinations by checking sample outputs.
* Ensured category translations are reflected only on the graph labels.

**Insights**

This visualization helps stakeholders:

* Identify trends between app size and rating.
* Understand the popularity of apps based on install size.
* Focus on high-performing categories with specific sentiment traits.

**Technologies and Tools Used**

* Python
* Pandas for data handling
* Matplotlib / Seaborn / Plotly for visualization
* Streamlit for creating the interactive dashboard
* datetime & pytz for time filtering
* Regular expressions for cleaning text

**Code:**

A black text on a white background

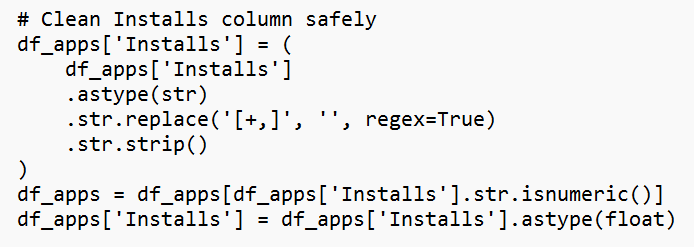
AI-generated content may be incorrect.

**Set ist time zone and time check**

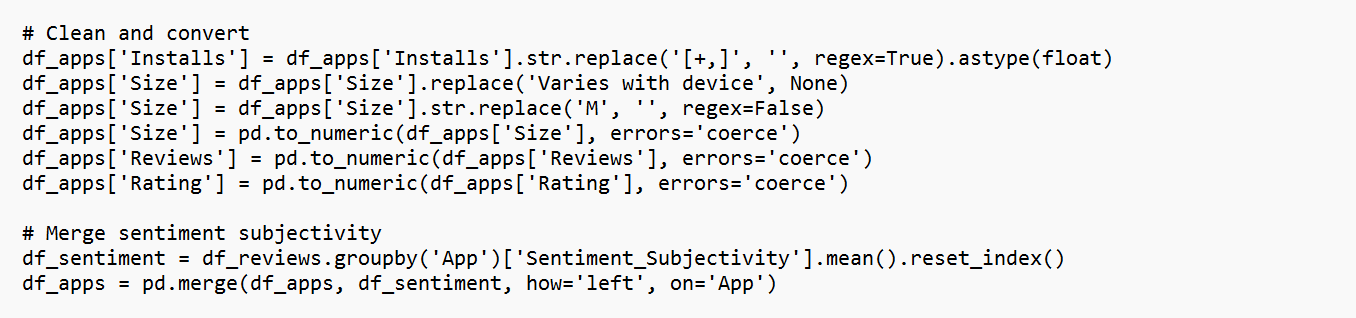
A computer code with black text

AI-generated content may be incorrect.

**Basic cleaning**



**Clean and convert**



**Filters and plotting**

A computer screen shot of a computer code

AI-generated content may be incorrect.

Graph:  
A diagram of a graph

AI-generated content may be incorrect.

**Conclusion**

Through completing all five tasks, I successfully applied data cleaning, filtering, and visualization techniques to the Google Play Store dataset using Streamlit and Python. Each task enhanced my skills in handling real-world datasets, implementing complex filters, and creating interactive, time-restricted visualizations. This project improved my problem-solving abilities, strengthened my understanding of data analysis workflows, and gave me hands-on experience in building user-friendly analytical dashboards.