NULL CLASS INTERNSHIP REPORT

REAL TIME GOOGLE PLAY STORE DATA ANALYTICS - PYTHON

19/12/2024 – 19/2/2025

Live Project URL: https://elaborate-marzipan-5b573c.netlify.app/

GitHub Link: https://github.com/sowmiyaM-1/GooglePlayStoreDataAnalysis

Introduction:

During my internship, I had the opportunity to dive into data visualization and analysis using Python. The project focused on creating meaningful and interactive visualizations for the Google Play Store dataset. Each task challenged me to apply advanced filtering techniques and dynamic logic to extract insights and present them effectively through dashboards. This hands-on experience helped me enhance both my technical and analytical skills.

Background:

The Google Play Store dataset contains detailed information about apps, such as their categories, installs, revenue, size, and ratings. My internship revolved around understanding this data, identifying patterns, and visualizing insights that could assist decision-making. By working on this project, I explored how data can drive impactful storytelling when presented effectively.

Learning Objectives:

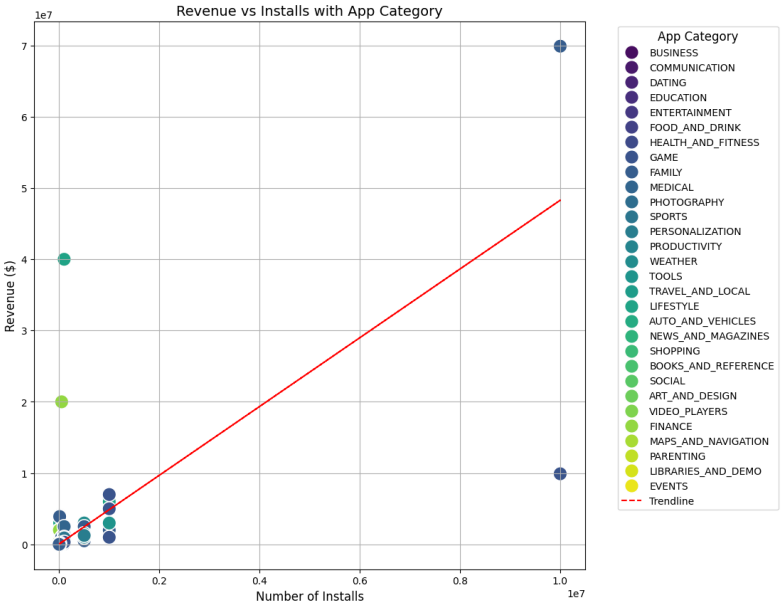
- To develop skills in data visualization and filtering using Python.
- To create time-based and dynamic dashboards.
- To learn how to work with real-world datasets and apply business logic to extract insights.
- To enhance creativity and problem-solving when faced with complex conditions in data analysis.

Activities and Tasks:

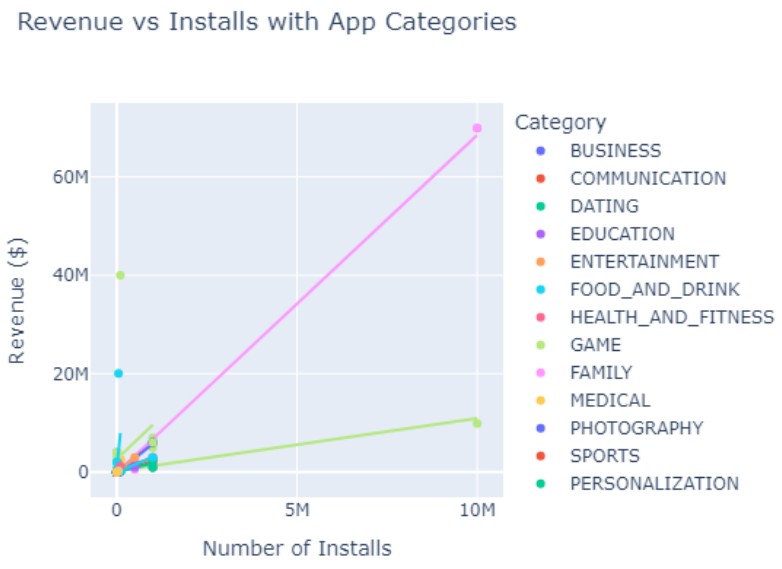
Here's a summary of the main tasks I worked on:

Task 1 Create a scatter plot to visualize the relationship between revenue and the number of installs for paid apps only. Add a trendline to show the correlation and color-coded the points based on app categories.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


data = pd.read_csv('Play Store Data.csv')


# 1. Filtered only paid apps
data = data[data['Type'] == 'Paid']

# 2. Handling of strings like $ in the price column
data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)

# 3. Handling of the characters like + into int
data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)

# 4. calculate revenue= Price* Installs to show the scatterplot
data['Revenue'] = data['Price'] * data['Installs']

# 5. Change the data frame based on our need
data = data[['Category', 'Price', 'Installs', 'Revenue']]

# 6. Scatterplot of Revenue vs Installs
plt.figure(figsize=(10, 10))
sns.scatterplot(
    data=data,
    x='Installs',
    y='Revenue',
    hue='Category',
    palette='viridis',
    s=100
)

# 7. Trendline for correlation
z = np.polyfit(data['Installs'], data['Revenue'], 1)  # Linear fit
p = np.poly1d(z)
plt.plot(data['Installs'], p(data['Installs']), color='red', linestyle='--', label='Trendline')



plt.title('Revenue vs Installs with App Category', fontsize=14)
plt.xlabel('Number of Installs', fontsize=12)
plt.ylabel('Revenue ($)', fontsize=12)

plt.legend(
    title='App Category',
    fontsize=12,
    title_fontsize=12,
    bbox_to_anchor=(1.05, 1),
    loc='upper left'
)

plt.grid(True)
plt.show()
```



```python
import pandas as pd
import plotly.express as px

data = pd.read_csv('Play Store Data.csv')


data = data[data['Type'] == 'Paid']

data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)

data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)


data['Revenue'] = data['Price'] * data['Installs']


data = data[['Category', 'Price', 'Installs', 'Revenue']]


fig = px.scatter(
    data,
    x='Installs',
    y='Revenue',
    color='Category',
    title='Revenue vs Installs with App Categy',
    labels={'Installs': 'Number of Installs', 'Revenue': 'Revenue ($)'},
    hover_data=['Price'],
    template='plotly'
)


fig = px.scatter(
    data,
    x='Installs',
    y='Revenue',
    color='Category',
    trendline='ols',
    title='Revenue vs Installs with App Category',
    labels={'Installs': 'Number of Installs', 'Revenue': 'Revenue ($)'},
    hover_data=['Price'],
    template='plotly'
)


fig.show()

```



Task 2 Create a dual-axis chart comparing the average installs and revenue for free vs. paid apps within the top 3 app categories.

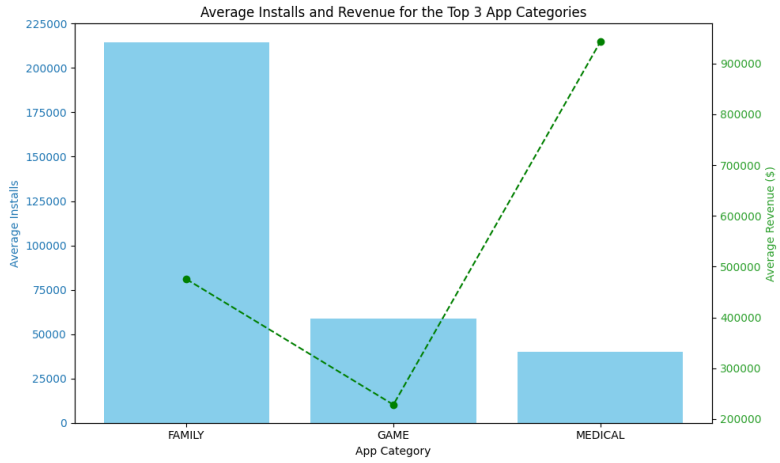
Apply filters - Exclude apps with fewer than 10,000 installs and revenue below $10,000, android version should be more than 4.0 as well as size should be more than 15M and content rating should be Everyone and app name should not have more than 30 characters including space and special character.

This graph should work only between 1 PM IST to 2 PM IST apart from that time we should not show this graph in dashboard itself.

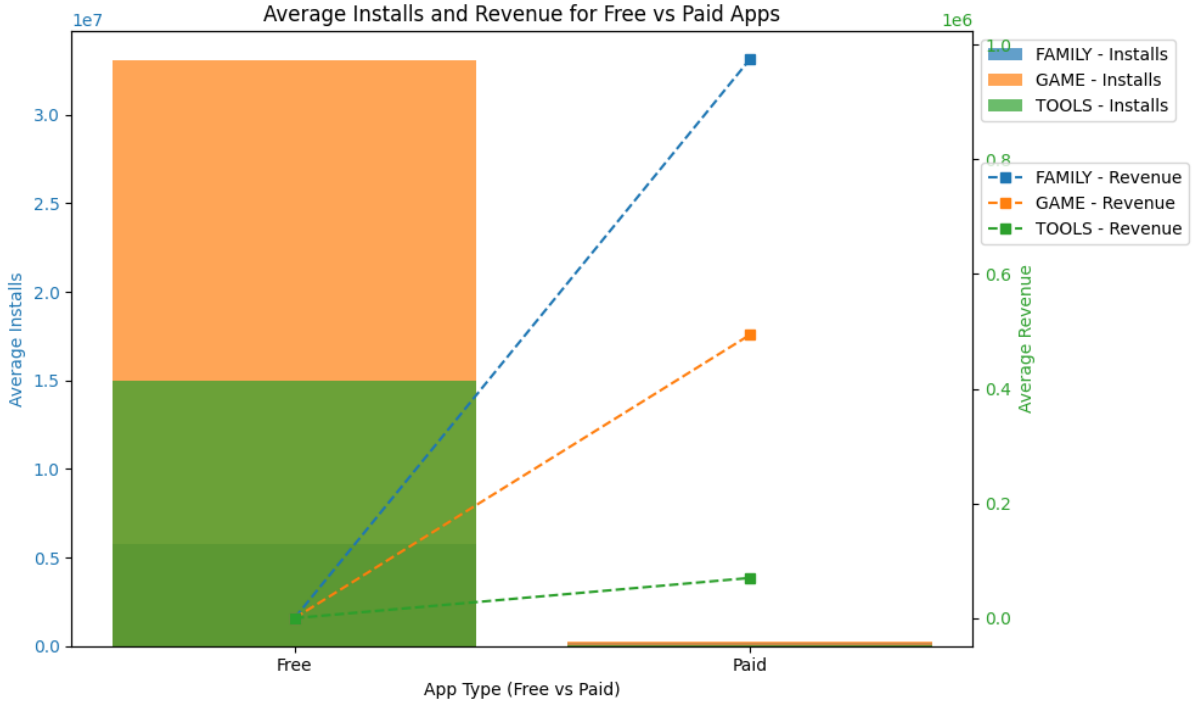
1.Filters and cleaning

```
data = pd.read_csv('Play Store Data.csv', encoding='utf-8', on_bad_lines='skip')

        data = data[data['Type'].isin(['Free', 'Paid'])]

    # 2. Handling of string in price value to float
        data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)

    # 3. Handling of the characters like + into int
        data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)

    # 4. Calculate revenue = Price * Installs
        data['Revenue'] = data['Price'] * data['Installs']
        data['Revenue'] = data['Revenue'].apply(lambda x: int(round(x)))

    # Filter out apps with fewer than 10,000 installs and revenue below 10,000
        data = data[(data['Installs'] >= 10000) & (data['Revenue'] >= 10000)]

    # 5. Exclude all except M
        data = data[~data['Size'].isin(['Varies with device'])]
        data = data[~data['Size'].str.contains('k')]
        data['Size'] = data['Size'].str.replace('M', '').astype(float)

    # 6. Filter size greater than 15M
        data = data[data['Size'] > 15]

        # 7. Filter Content Rating = 'Everyone'
        data = data[data['Content Rating'] == 'Everyone']

        # 8. Define a function to extract the number
        def extract_version(version_str):
            try:
                version = version_str.split(' ')[0]
                return float(version) if '.' in version else int(version)
            except:
                return None

        # 9. Apply the function to the 'Android Ver' column
        data['Android Ver'] = data['Android Ver'].apply(extract_version)

        # Filter for Android version >= 4.0
        data = data[data['Android Ver'] >= 4.0]

        # Trimming to get app names < 30 chars
        data['App'] = data['App'].apply(lambda x: x[:30] if len(x) > 30 else x)

        # 10. Select needed columns for the final output
        data = data[['Category', 'Type', 'Installs', 'Revenue', 'Size', 'App']]

        # Remove duplicates
        data = data.drop_duplicates()

        # Print the cleaned data
        print(data)
```

1.Task2.py



2.Task21.py

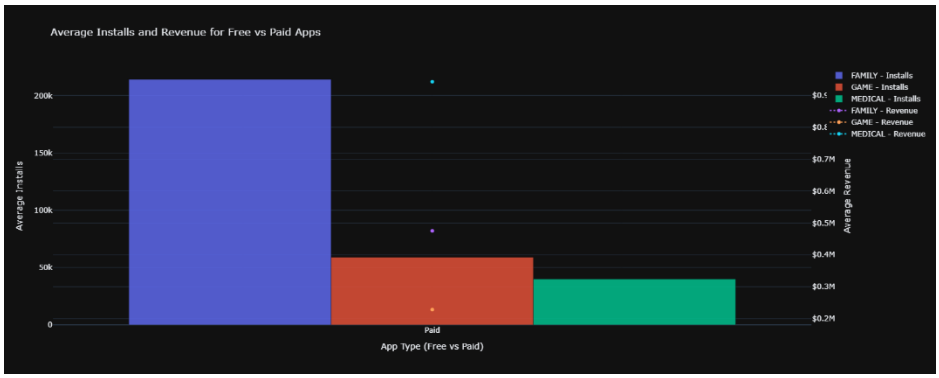


3.Task21Dashboard.py

Dashboard is visible during the timeframe

4.SaveHTMLFile.py → Final code

The plot will be saved as dashboard.html if within the mentioned time frame, else plot is not saved.

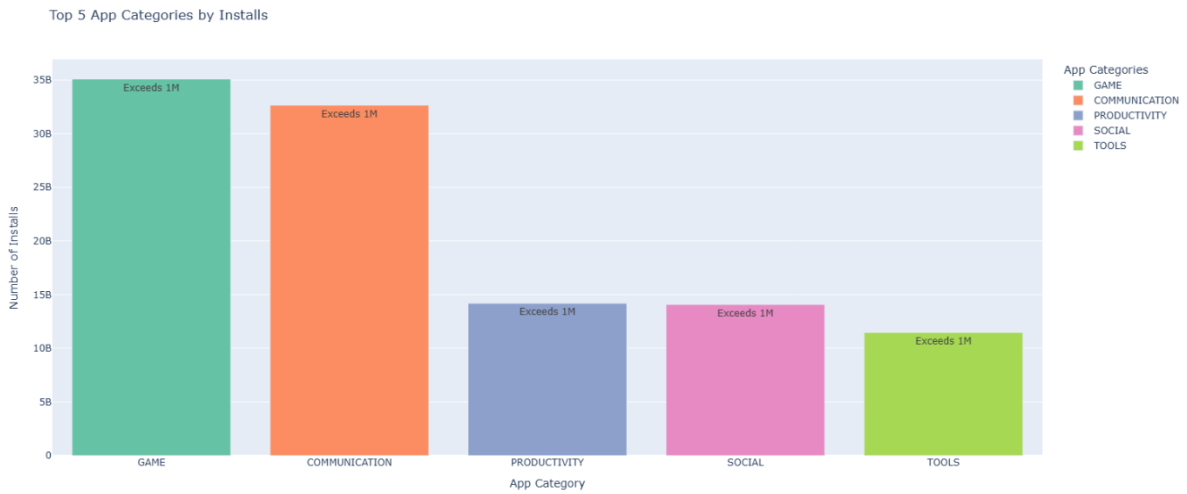


```python
def within_time_window():
    ist = pytz.timezone('Asia/Kolkata')
    current_time = datetime.now(ist)
    return 1 <= current_time.hour < 24

if within_time_window():

    fig = go.Figure()

    # bar chart for average installs
    for category in top_categories:
        category_data = filtered_data[filtered_data['Category'] == category]
        fig.add_trace(go.Bar(
            x=category_data['Type'],
            y=category_data['Installs'],
            name=f'{category} - Installs',
            opacity=0.8,
            yaxis='y1'
        ))

    # line chart for average revenue
    for category in top_categories:
        category_data = filtered_data[filtered_data['Category'] == category]
        fig.add_trace(go.Scatter(
            x=category_data['Type'],
            y=category_data['Revenue'],
            mode='lines+markers',
            name=f'{category} - Revenue',
            line=dict(dash='dot'),
            yaxis='y2'
        ))

    fig.update_layout(
        title='Average Installs and Revenue for Free vs Paid Apps',
        xaxis_title='App Type (Free vs Paid)',
        yaxis_title='Average Installs',
        yaxis2=dict(
            title='Average Revenue',
            overlaying='y',
            side='right',
            tickprefix='$',
        ),
        barmode='group',
        template='plotly_dark',
        height=600,
        showlegend=True,
    )

    fig.write_html('dashboard.html')

    print("Graph generated and saved as 'dashboard.html'.")

else:
    print("This graph is only available between 1 PM and 2 PM IST.")

```

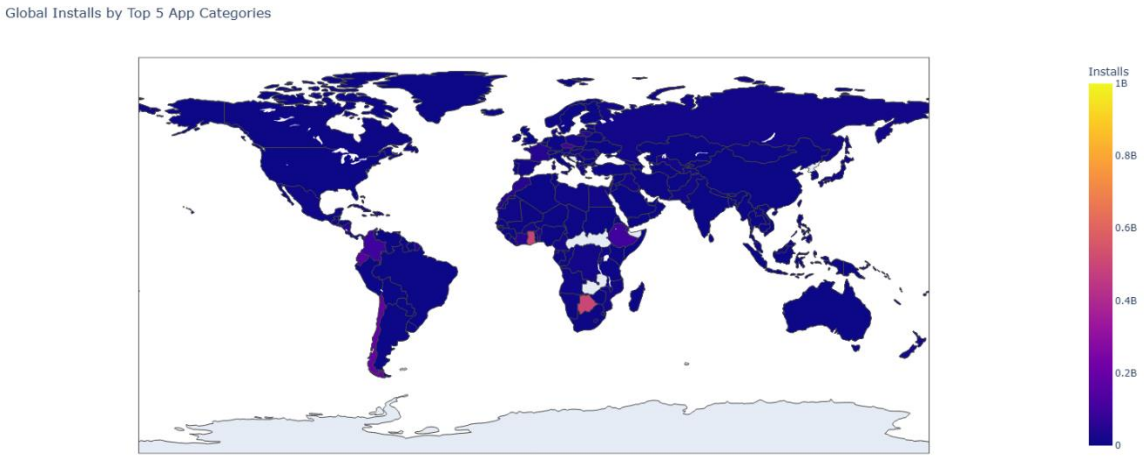
Task 3 Create an interactive choropleth map using Plotly to visualize global installs by categories . Apply filters to show data for only the top 5 app categories and highlight categories where the number of installs exceeds 1 million and App category should not start with character “A” , “C” , “G” and “S” . this graph should work only between 6PM IST to 8 PM IST apart from that time we should not show this graph in dashboard itself.

task3Barchart.py

```python
import pandas as pd
import plotly.express as px

data = pd.read_csv('C:/Users/sowmi/OneDrive/Documents/AI/Nullclass/Play Store Data.csv', encoding='utf-8', on_bad_lines='skip')

data = data[data['Price'] != "Everyone"]

data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)

data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)

category_installs = data.groupby('Category')['Installs'].sum().reset_index()

top_5_categories = category_installs.nlargest(5, 'Installs')

filtered_data = data[data['Category'].isin(top_5_categories['Category'])]

top_5_categories['Highlight'] = top_5_categories['Installs'].apply(lambda x: 'Exceeds 1M' if x > 1_000_000 else 'Below 1M')

fig = px.bar(
    top_5_categories,
    x='Category',
    y='Installs',
    color='Category',
    text='Highlight',
    title='Top 5 App Categories by Installs',
    color_discrete_sequence=px.colors.qualitative.Set2
)

fig.update_layout(
    legend_title_text='App Categories',
    xaxis_title='App Category',
    yaxis_title='Number of Installs',
    showlegend=True
)

fig.write_html('top_5_categories.html')
print("Bar chart generated and saved as 'top_5_categories.html'.")
```

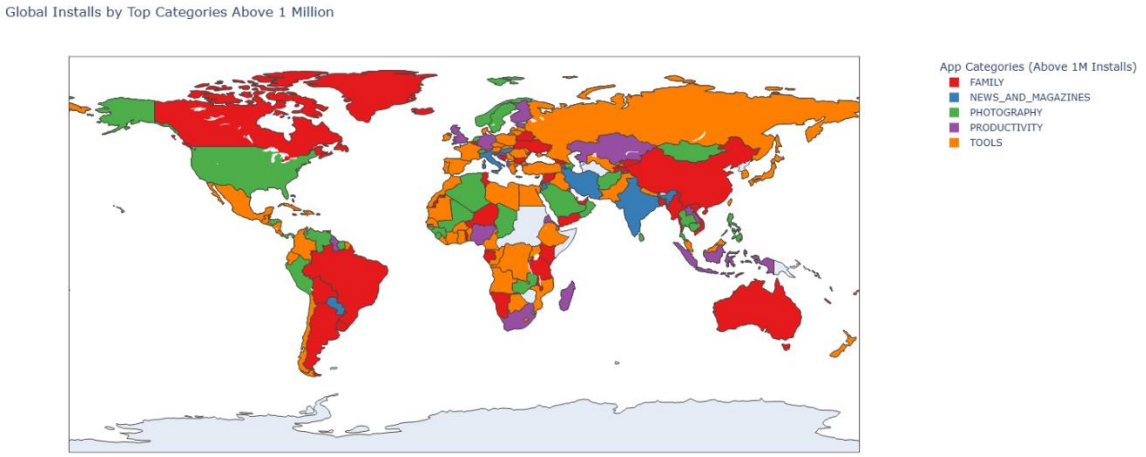


```python
import pandas as pd
import plotly.express as px
from datetime import datetime
import pytz

data = pd.read_csv('C:/Users/sowmi/OneDrive/Documents/AI/Nullclass/Play Store Data.csv', encoding='utf-8', on_bad_lines='skip')

data = data[data['Price'] != "Everyone"]

data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)


data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)

data['Revenue'] = data['Price'] * data['Installs']
data['Revenue'] = data['Revenue'].apply(lambda x: int(round(x)))

excluded_categories = ['A', 'C', 'G', 'S']
data = data[~data['Category'].str.startswith(tuple(excluded_categories))]

data = data[~data['Size'].isin(['Varies with device'])]
data = data[~data['Size'].str.contains('k')]
data['Size'] = data['Size'].str.replace('M', '').astype(float)

data = data[data['Size'] > 1]

top_5_categories = data.nlargest(5, 'Installs')

filtered_data = data[data['Category'].isin(top_5_categories['Category'])]

final_grouped_data = filtered_data.groupby(['Category', 'Country'])['Installs'].sum().reset_index()

def within_time_window():
    ist = pytz.timezone('Asia/Kolkata')
    current_time = datetime.now(ist)
    return 1 <= current_time.hour < 24

if within_time_window():

    fig = px.choropleth(
        final_grouped_data,
        locations='Country',
        locationmode='country names',
        color='Installs',
        hover_name='Category',
        title='Global Installs by Top 5 App Categories',
        color_continuous_scale=px.colors.sequential.Plasma
    )


    fig.write_html('dashboard3.html')
    print("Graph generated and saved as 'dashboard3.html'.")
else:
    print("This graph is only available between 1 PM and 2 PM IST.")

```

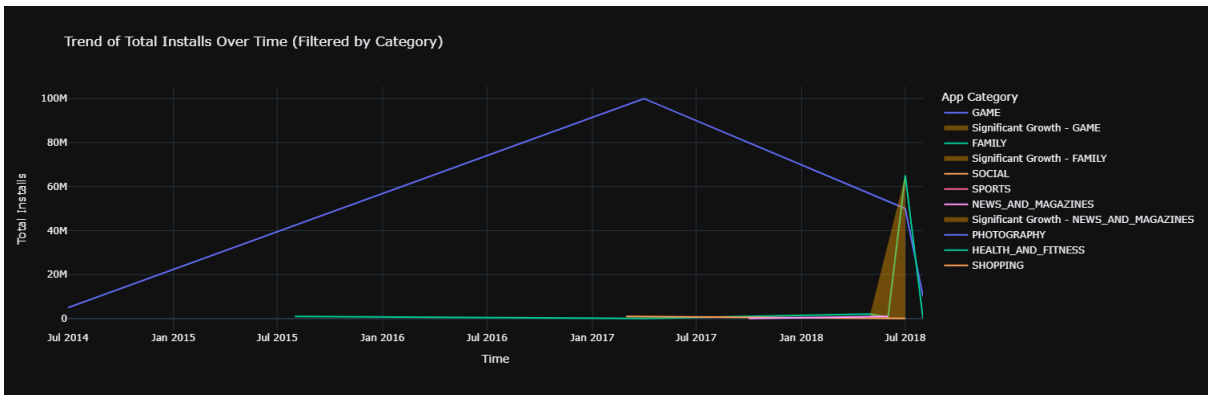


task3.py

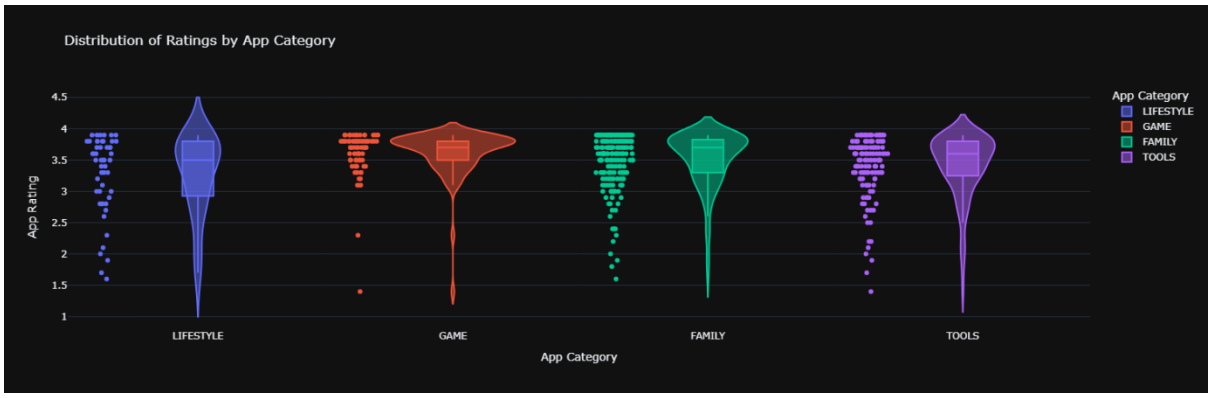
```python
import pandas as pd
import plotly.express as px
from datetime import datetime
import pytz

# Load the data
data = pd.read_csv('C:/Users/sowmi/OneDrive/Documents/AI/Nullclass/Play Store Data.csv', encoding='utf-8', on_bad_lines='skip')

# Data Cleaning
data = data[data['Price'] != "Everyone"]

data['Price'] = data['Price'].apply(lambda x: str(x).replace('$', '') if isinstance(x, str) else x).astype(float)


data['Installs'] = data['Installs'].str.replace(r'[^\d]', '', regex=True).astype(int)


data['Revenue'] = data['Price'] * data['Installs']
data['Revenue'] = data['Revenue'].apply(lambda x: int(round(x)))


ex_categories = ['A', 'C', 'G', 'S']
data = data[~data['Category'].str.startswith(tuple(ex_categories))]

data = data[~data['Size'].isin(['Varies with device'])]
data = data[~data['Size'].str.contains('k')]
data['Size'] = data['Size'].str.replace('M', '').astype(float)


data = data[data['Size'] > 15]


top_5_categories = data.groupby('Category')['Installs'].sum().nlargest(5).reset_index()


data = data[data['Category'].isin(top_5_categories['Category'])]


data['Highlight'] = data['Installs'].apply(lambda x: 'Above 1M Installs' if x > 1_000_000 else 'Other')


data1 = data[data['Highlight'] == 'Above 1M Installs']


final_data = data1.groupby(['Category', 'Country'])['Installs'].sum().reset_index()

def within_time_window():
    ist = pytz.timezone('Asia/Kolkata')
    current_time = datetime.now(ist)
    return 11 <= current_time.hour < 20

if within_time_window():

    fig = px.choropleth(
        final_data,
        locations='Country',
        locationmode='country names',
        color='Category',
        hover_name='Category',
        hover_data={
            'Category': True,
            'Installs': True
        },
        title='Global Installs by Top Categories Above 1 Million',
        color_discrete_sequence=px.colors.qualitative.Set1
    )

    # Add a legend and save the graph
    fig.update_layout(
        legend_title_text='App Categories (Above 1M Installs)',
        showlegend=True
    )

    # Save the graph as an HTML file
    fig.write_html('dashboard31.html')
    print("Graph generated and saved as 'dashboard31.html'.")
else:
    print("This graph is only available between 1 PM and 2 PM IST.")

```



Task 4 Plot a time series line chart to show the trend of total installs over time, segmented by app category. Highlight periods of significant growth by shading the areas under the curve where the increase in installs exceeds 20% month-over-month and content rating should be teen and app name should start with letter 'E' and installs should be more than 10k as well as this graph should work only between 6 PM IST to 9 PM IST apart from that time we should not show this graph in dashboard itself.



Task 5 Create a violin plot to visualize the distribution of ratings for each app category, but only include categories with more than 50 apps and app name should contain letter "C" and exclude apps with fewer than 10 reviews and rating should be less 4.0. this graph should work only between 4 PM IST to 6 PM IST apart from that time we should not show this graph in dashboard itself.



Challenges and Solutions:

1: Filtering data based on multiple conditions.

- I used pandas and advanced filtering techniques, combining multiple logical conditions seamlessly.

2: Restricting graphs to specific time windows.

- By integrating Python's datetime module, I ensured that graphs were displayed only during specified hours.

Outcomes and Impact:

This internship gave me a deeper understanding of how data visualization can drive meaningful insights. I successfully delivered interactive and visually engaging dashboards that met all business requirements. Beyond the technical skills, I learned to think critically and creatively when solving data-related challenges.

Conclusion:

Overall, this internship was a learning experience. It pushed me to explore new tools, apply advanced concepts, and find solutions to complex problems. I leave this internship feeling more confident in my ability to analyse and visualize data effectively, skills that I am excited to carry forward in future projects.