**Data Visualization and Exploration : A User-Friendly Tool Using Streamlit and Plotly**

**Introduction**

PhonePe has become one of the most popular digital payment platforms in India, with millions of users relying on it for their day-to-day transactions. The app is known for its simplicity, user-friendly interface, and fast and secure payment processing. It has also won several awards and accolades for its innovative features and contributions to the digital payments industry.

We create a web app to analyse the Phonepe transaction and users depending on various Years, Quarters, States, and Types of transaction and give a Geographical and Geo visualization output based on given requirements.

PhonePe Pulse website:

https://www.phonepe.com/pulse/explore/transaction/2022/4/

GitHub Link:

<https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson>

1.Developer Guide

**Libraries/Modules needed for the project!**

**1.Tools install**

 1. [Plotly](https://plotly.com/python/) - (To plot and visualize the data)

 2. [Pandas](https://pandas.pydata.org/docs/) - (To Create a DataFrame with the scraped data)

 3. mysql.connector - (To store and retrieve the data)

 4. [Streamlit](https://docs.streamlit.io/library/api-reference) - (To Create Graphical user Interface)

 5. json - (To load the json files)

 6. git.repo.base - (To clone the GitHub repository)

**2.Requirement Libraries to Install**

pip install pandas, numpy, os, json, requests, matplotlib.pyplot, mysql.connector , PIL, git, streamlit, plotly.express

# 2.WORKFLOW

**Importing the Libraries:**

    Importing the libraries. As I have already mentioned above the list of libraries/modules needed for the project. First we have to import all those libraries. If the libraries are not installed already use the below piece of code to install.

**Step 1:**

#pip install ["Name of the library"]

       If the libraries are already installed then we have to import those into our script by mentioning the below codes.

    import pandas as pd

        import mysql.connector as sql

        import streamlit as st

        import plotly.express as px

        import os

        import json

        from streamlit\_option\_menu import option\_menu

        from PIL import Image

        from git.repo.base import Repo

**Step 2:**

**ETL Process**

**Data extraction:**

   Clone the Github using scripting to fetch the data from the Phonepe pulse Github repository and store it in a suitable format such as JSON.

Initially, we Clone the data from the Phonepe GitHub repository by using Python libraries.

"<https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42ebb9b765c0a36/india_states.geojson>”

**Step 3:**

**Data transformation:**

   In this step the JSON files that are available in the folders are converted into the readeable and understandable DataFrame format by using the for loop and iterating file by file and then finally the DataFrame is created. In order to perform this step I've used **\*\*os\*\***, **\*\*json\*\*** and **\*\*pandas\*\*** packages. And finally converted the dataframe into CSV file and storing in the local drive.

    #This converts a given string into a dictionary which allows you to access your JSON data easily within my code

    # The JSON data and print that data same as we access the keys and values of a dictionary.

    # After making the get request to an API we store the JSON data in a variable “API\_Data” using the response.json() method

    data = json.loads(response.content)

    geo\_state = [i['properties']['ST\_NM'] for i in data['features']]

    geo\_state.sort(reverse=False)

    return geo\_state

**Load data**

# function for getting the state list

def original\_state\_list():

    mycursor.execute("""select distinct state

                            from aggregated\_transaction

                            order by state asc;""")

    s = mycursor.fetchall()

    original\_state = [i[0] for i in s]

    return original\_state

# fuction for making a dictionary from states in geojson to our states

def state\_dict():

    original = original\_state\_list()

    geo = geo\_state\_list()

    data = {}

    for i in range(0, len(original)):

        data[original[i]] = geo[i]

    return data

# taking the key value of that dictionary as it needs to be passed in the chloropeth map locations

def state\_list\_val(data):

    dat = []

    for key, val in data.items():

        dat.append(val)

    return dat

# function for getting original state list

def state\_list():

    mycursor.execute(f"""select distinct state

                            from aggregated\_transaction

                            order by state asc;""")

    data = mycursor.fetchall()

    original\_state = [i[0] for i in data]

    return original\_state

# function for getting the year list

def year\_list():

    mycursor.execute(

        "SELECT distinct year FROM phonepe.aggregated\_transaction order by year asc;")

    data = mycursor.fetchall()

    data = [i[0] for i in data]

    return data

# function for getting the quarter list

def quarter\_list():

    mycursor.execute(

        "SELECT distinct quarter FROM phonepe.aggregated\_transaction order by quarter asc;")

    data = mycursor.fetchall()

    data = [i[0] for i in data]

    return data

# function for getting the transaction type

def get\_transaction\_type():

    mycursor.execute(

        "SELECT distinct transaction\_type FROM phonepe.aggregated\_transaction;")

    data = mycursor.fetchall()

    data = [i[0] for i in data]

    return data

# function for getting average transaction value

def agg\_trans\_avg(agg\_trans):

    data = []

    for i in range(0, len(agg\_trans)):

        avg = agg\_trans.iloc[i]["Transaction\_amount"] / \

            agg\_trans.iloc[i]["Transaction\_count"]

        data.append(avg)

    return data

# getting transaction data

def get\_map\_transaction():

    mycursor.execute("SELECT \* FROM phonepe.map\_transaction;")

    data = mycursor.fetchall()

    d = pd.DataFrame(data, columns=mycursor.column\_names)

    return d

# creating index for tables

def new\_frame(v):

    i = [i for i in range(1, len(v)+1)]

    data = pd.DataFrame(v.values, columns=v.columns, index=i)

    return data

# function for getting aggregrated users

def get\_agg\_users():

    mycursor.execute("SELECT \* FROM phonepe.aggregated\_users;")

    data = mycursor.fetchall()

    d = pd.DataFrame(data, columns=mycursor.column\_names)

    return d

# getting map users data

def get\_map\_users():

    mycursor.execute("SELECT \* FROM phonepe.map\_users;")

    data = mycursor.fetchall()

    d = pd.DataFrame(data, columns=mycursor.column\_names)

    return d

# finding average users

def users\_trans\_avg(agg\_trans):

    data = []

    for i in range(0, len(agg\_trans)):

        avg = agg\_trans.iloc[i]["App\_opens"] / \

            agg\_trans.iloc[i]["Registered\_user"]

        data.append(avg)

    return data

Covert into dataframe:

            df\_agg\_total = get\_aggregated\_user()

            df\_agg\_total = df\_agg\_total.groupby(["State", "Year", "Transaction\_type"])[

                ["Transaction\_count", "Transaction\_amount"]].sum().reset\_index()

            df\_agg\_avg = agg\_trans\_avg(df\_agg\_total)

            df\_agg\_avg = pd.DataFrame(df\_agg\_avg, columns=["Avg\_value"])

            df\_agg\_total = pd.concat([df\_agg\_total, df\_agg\_avg], axis=1)

            q = df\_agg\_total[(df\_agg\_total["Year"] == year\_df) & (

                df\_agg\_total["Transaction\_type"] == transaction\_type)]

**Step 4:**

**Database insertion:**

   To insert the datadrame into SQL first I've created a new database and tables using **\*\*"mysql-connector-python"\*\*** library in Python to connect to a MySQL database and insert the transformed data using SQL commands.

**Creating the connection between python and mysql:**

        mydb = sql.connect(host=" 127.0.0.1", user="root", password="ka\*\*\*\*1",database= "phonepe")

        mycursor = mydb.cursor(buffered=True)

**### Step 5:**

**Dashboard creation**

THE MAIN COMPONENTS OF DASHBOARD ARE

1 GEO-VISUALIZATION

2 TRANSACTIONS ANALYSIS

3 USERS ANALYSIS

4 TOP STATES DATA

1 Geo-Visualization:

The India map shows the Total Transactions of PhonePe in both state wide and District wide.It comes with zoom

option and on hover displays the content related to that particular state or district.The main

functions I have used to create this map are (User can give year and quarter input to show how the data changed over time)

1 Plotlys scatter\_geo for plotting districts along with the conent

2 Plotlys coropleth for drawing the states in India map

2 Transactions Analysis:

The Transactions data mainly contains the total Transactions count and total amount in each state and

district, I have used different graphs available in plotly to represent this data

1 State-wise study

The above bar graph shows the increasing order of PhonePe Transactions according to the states of India,

Here we can observe the top states with the highest Transaction by looking at graph

2 District-wise study

User can observe how transactions are happening in districts of a selected state.We can observe the

leading distric in a state

3 Year-wise study

We can observe the states with total transactions in particular mode in the selected year

4 Overall Analysis

To show how the transactions drastically increased with time

3 User Data Analysis:

The Users data mainly contains the Registered Users count and App openings via different

mobile brands in each state and district,I have used different graphs available in plotly

to represent this data

1 State-wise study

User can observe how the App Openings are growing and how Registered users are growing in a state

2 District-wise study

User can observe how App Openings are happening in districts of a selected state

3 Year-wise study

User can observe the top leading brands in a particular state in given year

4 Overall Analysis

We can see that the Registered Users and App openings are increasing year by year

4 Top States Data:

1 States with top Registered users

2 States with top Total Amount Transacted

3 States with highest Trabsactions count

4 States with top app openings

**Results**

- The result of this project is a live geo visualization dashboard that displays information and insights from the Phonepe Pulse GitHub repository in an interactive and visually appealing manner.

- The dashboard provides at least 10 different dropdown options for users to select various facts and figures for display.

- The data is efficiently stored in a MySQL database and is dynamically updated to reflect the latest data.

- Users can access the dashboard from a web browser and easily navigate the different visualizations and facts and figures displayed.

- The dashboard provides valuable insights and information about the data in the Phonepe Pulse GitHub repository, making it a valuable tool for data analysis and decision-making.

**Links:**

Github Link: <https://github.com/kanis11/Phonepe-Pulse-Data-Visualization-and-Exploration->

**Want to see demo video of my project? - [Click here]**

**(** <https://www.linkedin.com/feed/update/urn:li:ugcPost:7232638745777217536/> )

**References:**

* [**https://www.phonepe.com/pulse/explore/transaction/2022/4/**](https://www.phonepe.com/pulse/explore/transaction/2022/4/)
* [**https://www.geeksforgeeks.org/data-cleansing-introduction/**](https://www.geeksforgeeks.org/data-cleansing-introduction/)
* [**https://blog.streamlit.io/crafting-a-dashboard-app-in-python-using-streamlit/**](https://blog.streamlit.io/crafting-a-dashboard-app-in-python-using-streamlit/)
* [**https://python-graph-gallery.com/bubble-plot-with-seaborn/**](https://python-graph-gallery.com/bubble-plot-with-seaborn/)

**Author:** Kanimozhi S,

Linkedin: https://www.linkedin.com/in/ [View Profile]

Mail-Id: kanimozhis3010@gmail.com

Happy Analyzing!