Project Report

# 1. Title:

Data Analysis Using PhonePe Pulse Data

# 2. Objective:

To analyze financial transaction data from the PhonePe Pulse GitHub repository and create an interactive dashboard showing insights based on state, district, year, and quarter.

# 3. Tools & Technologies Used:

* Python
* Streamlit
* PostgreSQL
* Plotly
* GitHub API
* Pandas

# 4. Project Description:

This project involves extracting PhonePe transaction data from their open-source GitHub repository, transforming the data, storing it in a PostgreSQL database, and visualizing it using a dashboard built with Streamlit. The dashboard includes maps, graphs, and filters to explore various insights.

# 5. Features:

* Total transactions and amounts by year and quarter
* State-wise and district-wise data visualization
* Top categories and user data
* Interactive and user-friendly dashboard interface

# 7.main.py

**tep 1: Connect to PostgreSQL Database**

* The script connects to a PostgreSQL database using credentials like host, database name, user, and password.
* This connection allows the script to save data into database tables.

**Step 2: Clean State Names**

* Some state names in the data have hyphens (-), like "andhra-pradesh".
* The script replaces hyphens with spaces and converts names to title case (like "Andhra Pradesh") for consistency.

**Step 3: Define Functions to Insert Data**

* There are different insert functions for each kind of data:
  + Aggregated user data (users by device brand)
  + Aggregated transaction data (transaction type, count, amount)
  + Aggregated insurance data (insurance type, count, amount)
  + Map user data (registered users and app opens by district)
  + Map transaction data (transaction count and amount by district)
  + Top users (district-wise registered users)
  + Top transactions (district, pincode-wise transaction counts and amounts)
* These functions take data records and insert them into appropriate database tables.
* They also handle errors during insertion.

**Step 4: Process Files in Folder Structure**

* The data is organized in folders by:
  + State → Year → Quarter (each quarter as a JSON file)
* The script loops through these folders and files.
* For each JSON file, it reads the data and calls the right insert function to save data into the database.

**Step 5: Process National Level Map Data**

* Some data (map user and transaction data) is at the national level, not specific to states.
* Separate functions handle these files by reading data and inserting into the database.

**Step 6: Run the Full ETL Pipeline**

* The run\_etl() function calls all the above processes in order:
  + Load aggregated user data
  + Load aggregated transaction data
  + Load aggregated insurance data
  + Load map user data
  + Load map transaction data
  + Load top user data
  + Load top transaction data
* This ensures the entire dataset is processed and loaded.

**Step 7: Commit Changes and Handle Errors**

* After each file’s data is inserted, the script commits the changes to the database.
* If any error occurs, it rolls back changes and prints the error for debugging.

**Step 8: Start Script Execution**

* When you run the script, it begins processing data from a base folder (../data).
* The script loads all PhonePe Pulse data into the PostgreSQL database, ready for further analysis or visualization.

# 6.phonepe\_dashboard.py

**1. Background Styling**

* Adds a full-page blue background image using custom CSS injected via st.markdown.
* Ensures the image covers the entire page and stays fixed.

**2. Database Configuration & Connection**

* Sets PostgreSQL credentials (host, db, user, password).
* Uses @st.cache\_resource decorator to cache the database connection for efficiency.

**3. Data Loading Functions**

* load\_data(query, params) runs SQL queries and caches results for 10 minutes to reduce DB load.
* load\_geojson() fetches India state boundaries GeoJSON from a GitHub URL and caches it.

**4. UI Setup**

* Page title and layout set with st.set\_page\_config.
* Displays a red heading for the dashboard.

**5. Filters**

* Two dropdowns for users to select **Year** and **Quarter** (latest years first).

**6. Tabs for Different Data Views**

* Three tabs:
  + Aggregated Transactions
  + Top Transactions
  + Insurance

**7. Tab 1: Aggregated Transactions**

* Queries aggregated transaction data by state, summing amounts and counts.
* Displays a choropleth map of transaction amounts by state using Plotly.
* Maps some state names to match GeoJSON property names for correct mapping.
* Shows the underlying data in a table below the map.

**8. Tab 2: Top Transactions**

* Queries top 20 districts by transaction amount for the selected period.
* Displays a bar chart colored by state and a data table.

**9. Tab 3: Insurance**

* Queries insurance amounts and counts by insurance type.
* Shows a pie chart of insurance distribution and a data table.

# 6. Conclusion:

The project gave hands-on experience in data extraction, cleaning, storage, and visualization. It also improved understanding of how to create dashboards using real-world data.

## OUTPUT:





