```
#PhonePe Transaction Insights - Steamlit view coe

import streamlit as st
import streamlit as st
import pandas as pd
import mysql.connector


# ------------------------------------------
st.set_page_config(page_title="🧮 PhonePe Transaction Insights",
                   page_icon="💰",
                   layout="wide")

# Custom CSS styling
st.markdown("""
    <style>
    .stApp { background-color: #0F0326; color: #FFFFFF; }
    .table-card { background-color: #1E0A47; border-radius: 8px; padding: 16px;
margin-bottom: 16px; }
    .table-card h3 { color: #FFD700; margin: 0; padding-bottom: 8px; }
    .sidebar .sidebar-content { background-color: #12143A; }
    </style>
""", unsafe_allow_html=True)

st.title("💳 PhonePe Transaction Insights")

mydb = mysql.connector.connect(
    host="gateway01.ap-southeast-1.prod.aws.tidbcloud.com",
    user="r39R7LHcFCC4RPb.root",
    password="kgl0zVHcTUhPeylz",
    database="phone_pay",
    autocommit=True
)

@st.cache_data
def run_query(query: str):
    cursor = mydb.cursor(dictionary=True)
    cursor.execute(query)
    rows = cursor.fetchall()
    cursor.close()
    return pd.DataFrame(rows)

st.sidebar.header("📊 Dashboard Options")
mode = st.sidebar.radio("Choose View Mode", ["🗂 Table View", "📈 Business Case
Study"])

if mode == "🗂 Table View":
    st.sidebar.header("🗂 Select Table")
    table_list = [
        "Aggregated_Transaction_data",
```

```
        "Aggregated_Insurance_data",
        "Aggregated_User_data",
        "Map_Insurance_data",
        "Map_Transaction_data",
        "Map_User_data",
        "Top_Insurance_data",
        "Top_Transaction_data",
        "Top_users_data"
    ]
    selected_table = st.sidebar.selectbox("Choose a table", table_list)
    st.markdown(f"### Displaying data from: **{selected_table}**")

    df = run_query(f"SELECT * FROM `{selected_table}`;")

    if df.empty:
        st.warning("⚠ No data found.")
    else:
        st.sidebar.markdown("---")
        st.sidebar.header("🔍 Filter Options")

        filtered_df = df.copy()

        if "State" in df.columns:
            states = st.sidebar.multiselect("Select State(s)",
sorted(df["State"].dropna().unique()))
            if states:
                filtered_df = filtered_df[filtered_df["State"].isin(states)]

        if "Year" in df.columns:
            years = st.sidebar.multiselect("Select Year(s)",
sorted(df["Year"].dropna().unique()))
            if years:
                filtered_df = filtered_df[filtered_df["Year"].isin(years)]

        if "Quater" in df.columns:
            quarters = st.sidebar.multiselect("Select Quarter(s)",
sorted(df["Quater"].dropna().unique()))
            if quarters:
                filtered_df = filtered_df[filtered_df["Quater"].isin(quarters)]

        st.subheader("📋 Filtered Table Data")
        st.write(f"Showing {len(filtered_df)} records after applying filters")
        st.dataframe(filtered_df, use_container_width=True)

        if {"Year", "Transacion_count", "Transacion_amount"}.issubset(df.columns):
            summary = filtered_df.groupby("Year")[["Transacion_count",
"Transacion_amount"]].sum().reset_index()
            st.write("#### Transaction Count and Amount by Year")
            st.dataframe(summary)
            st.bar_chart(summary.set_index("Year"))
```

```
        elif {"Year", "Transaction_count",
"Transaction_amount"}.issubset(df.columns):
            summary = filtered_df.groupby("Year")[["Transaction_count",
"Transaction_amount"]].sum().reset_index()
            st.write("#### Transaction Count and Amount by Year")
            st.dataframe(summary)
            st.bar_chart(summary.set_index("Year"))
        else:
            st.info("ℹ️ No numeric transaction data available for chart
visualization.")

        st.success("☑️ Table data loaded successfully!")

else:
    st.sidebar.header("💼 Select Business Case Study")
    case_study = st.sidebar.selectbox("Choose Scenario", [
        "1️⃣ Decoding Transaction Dynamics",
        "2️⃣ Device Dominance & User Engagement",
        "3️⃣ Insurance Penetration & Growth Potential",
        "4️⃣ User Engagement & Growth Strategy",
        "5️⃣ Transaction Analysis Across States & Districts"
    ])

    def show_top_bottom(df, value_col, state_col="State", year_col="Year",
quarter_col="Quater"):
        if not {state_col, year_col, quarter_col, value_col}.issubset(df.columns):
            st.warning("⚠️ Insufficient columns to determine top and bottom
states.")
            return
        summary = (
            df.groupby([state_col, year_col, quarter_col])[value_col]
            .sum()
            .reset_index()
        )
        top_row = summary.loc[summary[value_col].idxmax()]
        low_row = summary.loc[summary[value_col].idxmin()]

        st.markdown("### ⏳ Top Performing State")
        st.info(f"**{top_row[state_col]}** — Year: {top_row[year_col]}, Quarter:
{top_row[quarter_col]}, Value: {top_row[value_col]:,.2f}")

        st.markdown("### ⚠️ Lowest Performing State")
        st.error(f"**{low_row[state_col]}** — Year: {low_row[year_col]}, Quarter:
{low_row[quarter_col]}, Value: {low_row[value_col]:,.2f}")

    # Business Case 1
    if case_study.startswith("1️⃣"):
        st.subheader("📊 Decoding Transaction Dynamics on PhonePe")
        query = """
        SELECT
```

```
            State, Year, Quater, Transacion_type,
            SUM(Transacion_count) AS Total_Transactions,
            SUM(Transacion_amount) AS Total_Value
        FROM Aggregated_Transaction_data
        GROUP BY State, Year, Quater, Transacion_type
        ORDER BY Year, Quater;
        """
        df = run_query(query)
        st.dataframe(df)
        st.bar_chart(df.groupby("Year")[["Total_Transactions",
"Total_Value"]].sum())
        show_top_bottom(df, "Total_Value")

    # Business Case 1️⃣ – Decoding Transaction Dynamics
    if case_study.startswith("1️"):
        import plotly.express as px

        st.subheader("📊 Decoding Transaction Dynamics on PhonePe")

        # Sidebar selection for analysis type
        st.sidebar.markdown("---")
        st.sidebar.header("📈 Choose Analysis Type")
        analysis_option = st.sidebar.selectbox(
            "Select Query / Analysis View",
            [
                "1️⃣ Total Transactions & Value by State (Top Performing States)",
                "2️⃣ Yearly Growth Trend Across India",
                "3️⃣ Quarterly Performance by State (Seasonal Insights)",
                "4️⃣ Transaction Share by Transaction Type",
                "5️⃣ Identify Low-Performing States (Bottom 10)"
            ]
        )

        # 1️⃣ Total Transactions & Value by State (Top Performing States)
        if analysis_option.startswith("1️"):
            query = """
            SELECT
                State,
                SUM(Transacion_count) AS Total_Transactions,
                SUM(Transacion_amount) AS Total_Value
            FROM Aggregated_Transaction_data
            GROUP BY State
            ORDER BY Total_Value DESC
            LIMIT 10;
            """
            df = run_query(query)
            st.subheader("🏆 Total Transactions & Value by State (Top 10
Performing States)")
            st.dataframe(df)
            fig = px.bar(df, x="State", y="Total_Value", color="Total_Value",
```

```
                        title="💰 Top 10 States by Transaction Value",
text_auto=True)
            st.plotly_chart(fig, use_container_width=True)

        # 2️⃣ Yearly Growth Trend Across India
        elif analysis_option.startswith("2️"):
            query = """
            SELECT
                Year,
                SUM(Transacion_count) AS Total_Transactions,
                SUM(Transacion_amount) AS Total_Value
            FROM Aggregated_Transaction_data
            GROUP BY Year
            ORDER BY Year;
            """
            df = run_query(query)
            st.subheader("📈 Yearly Growth Trend Across India")
            st.dataframe(df)
            fig = px.line(df, x="Year", y="Total_Value", markers=True,
                          title="📊 Yearly Growth in Transaction Value")
            st.plotly_chart(fig, use_container_width=True)

        # 3️⃣ Quarterly Performance by State (Seasonal Insights)
        elif analysis_option.startswith("3️"):
            query = """
            SELECT
                State,
                Year,
                Quater,
                SUM(Transacion_count) AS Total_Transactions,
                SUM(Transacion_amount) AS Total_Value
            FROM Aggregated_Transaction_data
            GROUP BY State, Year, Quater
            ORDER BY Year, Quater;
            """
            df = run_query(query)
            st.subheader("📅 Quarterly Performance by State (Seasonal Insights)")
            st.dataframe(df)
            fig = px.bar(df, x="Quater", y="Total_Value", color="State",
                         title="🗓️ Quarterly Transaction Value by State",
barmode="group")
            st.plotly_chart(fig, use_container_width=True)

        # 4️⃣ Transaction Share by Transaction Type
        elif analysis_option.startswith("4️"):
            query = """
            SELECT
                Transacion_type,
                SUM(Transacion_count) AS Total_Transactions,
                SUM(Transacion_amount) AS Total_Value,
```

```
                ROUND(SUM(Transacion_amount) * 100 /
                      (SELECT SUM(Transacion_amount) FROM
Aggregated_Transaction_data), 2) AS Percentage_Share
            FROM Aggregated_Transaction_data
            GROUP BY Transacion_type
            ORDER BY Total_Value DESC;
            """
            df = run_query(query)
            st.subheader("🍩 Transaction Share by Transaction Type")
            st.dataframe(df)
            fig = px.pie(df, names="Transacion_type", values="Total_Value",
                         title="💡 Transaction Value Share by Type", hole=0.4)
            st.plotly_chart(fig, use_container_width=True)

        # 5️⃣ Identify Low-Performing States (Bottom 10)
        elif analysis_option.startswith("5️"):
            query = """
            SELECT
                State,
                SUM(Transacion_count) AS Total_Transactions,
                SUM(Transacion_amount) AS Total_Value
            FROM Aggregated_Transaction_data
            GROUP BY State
            ORDER BY Total_Value ASC
            LIMIT 10;
            """
            df = run_query(query)
            st.subheader("⚠️ Identify Low-Performing States (Bottom 10)")
            st.dataframe(df)
            fig = px.bar(df, x="State", y="Total_Value", color="Total_Value",
                         title="🚨 Bottom 10 States by Transaction Value",
text_auto=True)
            st.plotly_chart(fig, use_container_width=True)

        st.success(f"☑️ Analysis Loaded: {analysis_option}")


    # ================================
    # 📱 BUSINESS CASE 2 - Device Dominance & User Engagement
    # ================================
    elif case_study.startswith("2️"):
        import plotly.express as px

        st.subheader("📱 Device Dominance & User Engagement Analysis")

        # --- Load main overview data ---
        overview_query = """
        SELECT
            a.Brand_type AS Device_Brand,
            m.State,
```

```
            m.Year,
            m.Quater,
            SUM(m.RegisteredUsers) AS Total_Registered,
            SUM(m.AppOpens) AS Total_App_Opens,
            ROUND(SUM(m.AppOpens)/NULLIF(SUM(m.RegisteredUsers),0),2) AS
AppOpens_per_User
        FROM Aggregated_User_data AS a
        JOIN Map_User_data AS m
          ON a.State = m.State AND a.Year = m.Year AND a.Quater = m.Quater
        GROUP BY a.Brand_type, m.State, m.Year, m.Quater
        ORDER BY Total_Registered DESC;
        """
        overview_df = run_query(overview_query)

        # --- Show main business case data ---
        st.markdown("### 📊 Device-Level Overview Data")
        st.dataframe(overview_df)
        st.bar_chart(overview_df.groupby("Device_Brand")[["Total_Registered",
"Total_App_Opens"]].sum())
        show_top_bottom(overview_df, "Total_Registered")

        # --- Sidebar for 5 analysis queries ---
        st.sidebar.markdown("---")
        st.sidebar.header("📈 Choose Analysis Type")
        analysis_option = st.sidebar.selectbox(
            "Select Query / Analysis View",
            [
                "1️⃣ Top 5 Performing Brands by Total Registered Users",
                "2️⃣ App Engagement Efficiency (App Opens per User by Brand)",
                "3️⃣ Top 5 States with Highest App Usage",
                "4️⃣ Underperforming States (Lowest Engagement per User)",
                "5️⃣ Yearly Trend of App Usage Across India"
            ]
        )

        # 1️⃣ Top 5 Performing Brands
        if analysis_option.startswith("1️⃣"):
            q = """
            SELECT a.Brand_type AS Device_Brand, SUM(m.RegisteredUsers) AS
Total_Registered_Users
            FROM Aggregated_User_data AS a
            JOIN Map_User_data AS m
              ON a.State=m.State AND a.Year=m.Year AND a.Quater=m.Quater
            GROUP BY a.Brand_type
            ORDER BY Total_Registered_Users DESC LIMIT 5;
            """
            df = run_query(q)
            st.subheader("⏳ Top 5 Performing Brands by Total Registered Users")
            st.dataframe(df)
            st.plotly_chart(px.bar(df, x="Device_Brand",
```

```
y="Total_Registered_Users", color="Total_Registered_Users",
                                    text_auto=True, title="Top 5 Performing
Brands"))

        # 2️⃣ App Engagement Efficiency
        elif analysis_option.startswith("2️"):
            q = """
            SELECT a.Brand_type AS Device_Brand,
                   ROUND(SUM(m.AppOpens)/NULLIF(SUM(m.RegisteredUsers),0),2) AS
AppOpens_per_User
            FROM Aggregated_User_data AS a
            JOIN Map_User_data AS m
              ON a.State=m.State AND a.Year=m.Year AND a.Quater=m.Quater
            GROUP BY a.Brand_type ORDER BY AppOpens_per_User DESC LIMIT 5;
            """
            df = run_query(q)
            st.subheader("⚡ App Engagement Efficiency (App Opens per User by
Brand)")
            st.dataframe(df)
            st.plotly_chart(px.bar(df, x="Device_Brand", y="AppOpens_per_User",
color="AppOpens_per_User",
                                    text_auto=True, title="App Opens per User by
Brand"))

        # 3️⃣ Top 5 States with Highest App Usage
        elif analysis_option.startswith("3️"):
            q = """
            SELECT m.State, SUM(m.AppOpens) AS Total_App_Opens
            FROM Aggregated_User_data AS a
            JOIN Map_User_data AS m
              ON a.State=m.State AND a.Year=m.Year AND a.Quater=m.Quater
            GROUP BY m.State ORDER BY Total_App_Opens DESC LIMIT 5;
            """
            df = run_query(q)
            st.subheader("🌍 Top 5 States with Highest App Usage")
            st.dataframe(df)
            st.plotly_chart(px.pie(df, names="State", values="Total_App_Opens",
                                    title="Top 5 States by App Opens", hole=0.3))

        # 4️⃣ Underperforming States
        elif analysis_option.startswith("4️"):
            q = """
            SELECT m.State,
                   ROUND(SUM(m.AppOpens)/NULLIF(SUM(m.RegisteredUsers),0),2) AS
AppOpens_per_User
            FROM Aggregated_User_data AS a
            JOIN Map_User_data AS m
              ON a.State=m.State AND a.Year=m.Year AND a.Quater=m.Quater
            GROUP BY m.State ORDER BY AppOpens_per_User ASC LIMIT 5;
            """
```

```
            df = run_query(q)
            st.subheader("⚠ Underperforming States (Lowest Engagement per User)")
            st.dataframe(df)
            st.plotly_chart(px.bar(df, x="State", y="AppOpens_per_User",
color="AppOpens_per_User",
                                   text_auto=True, title="Lowest Engagement
States"))


        # 5️⃣ Yearly Trend of App Usage
        elif analysis_option.startswith("5️"):
            q = """
            SELECT m.Year,
                   SUM(m.RegisteredUsers) AS Total_Registered,
                   SUM(m.AppOpens) AS Total_App_Opens,
                   ROUND(SUM(m.AppOpens)/NULLIF(SUM(m.RegisteredUsers),0),2) AS
Avg_AppOpens_per_User
            FROM Aggregated_User_data AS a
            JOIN Map_User_data AS m
              ON a.State=m.State AND a.Year=m.Year AND a.Quater=m.Quater
            GROUP BY m.Year ORDER BY m.Year;
            """
            df = run_query(q)
            st.subheader("📈 Yearly Trend of App Usage Across India")
            st.dataframe(df)
            st.plotly_chart(px.line(df, x="Year", y="Avg_AppOpens_per_User",
markers=True,
                                    title="Average App Opens per User Over Years"))

        st.success(f"☑ Analysis Loaded: {analysis_option}")

    # ================================
    # 🌓 BUSINESS CASE 3 - Insurance Penetration & Growth Potential
    # ================================
    elif case_study.startswith("3️"):
        import plotly.express as px

        st.subheader("🌓 Insurance Penetration & Growth Potential Analysis")

        # --- Overview data ---
        overview_query = """
        SELECT
            State, Year, Quater,
            SUM(Insurance_count) AS Insurance_Transactions,
            SUM(Insurance_amount) AS Insurance_Value
        FROM Aggregated_Insurance_data
        GROUP BY State, Year, Quater
        ORDER BY Year, Quater;
        """
        overview_df = run_query(overview_query)
```

```
        st.markdown("### 📊 Insurance Overview Data")
        st.dataframe(overview_df)
        st.bar_chart(overview_df.groupby("Year")[["Insurance_Transactions",
"Insurance_Value"]].sum())
        show_top_bottom(overview_df, "Insurance_Value")

        # --- Sidebar analysis queries ---
        st.sidebar.markdown("---")
        st.sidebar.header("📈 Choose Analysis Type")
        insurance_option = st.sidebar.selectbox(
            "Select Query / Analysis View",
            [
                "1️⃣ Total Insurance Transactions & Value by State (Top Performing
States)",
                "2️⃣ Yearly Growth Trend of Insurance Transactions",
                "3️⃣ Quarter-wise Distribution of Insurance Value",
                "4️⃣ Top 5 High-Value States in the Latest Year",
                "5️⃣ YoY Growth Rate of Insurance Value"
            ]
        )

        # 1️⃣
        if insurance_option.startswith("1️⃣"):
            q = """
            SELECT State, SUM(Insurance_count) AS Total_Transactions,
SUM(Insurance_amount) AS Total_Value
            FROM Aggregated_Insurance_data GROUP BY State ORDER BY Total_Value DESC
LIMIT 10;
            """
            df = run_query(q)
            st.subheader("🏆 Top 10 States by Total Insurance Value")
            st.dataframe(df)
            st.plotly_chart(px.bar(df, x="State", y="Total_Value",
color="Total_Value",
                                   title="Top Performing States by Insurance
Value", text_auto=True))

        # 2️⃣
        elif insurance_option.startswith("2️⃣"):
            q = """
            SELECT Year, SUM(Insurance_count) AS Total_Transactions,
SUM(Insurance_amount) AS Total_Value
            FROM Aggregated_Insurance_data GROUP BY Year ORDER BY Year;
            """
            df = run_query(q)
            st.subheader("📈 Yearly Growth Trend of Insurance Transactions")
            st.dataframe(df)
            st.plotly_chart(px.line(df, x="Year", y="Total_Value", markers=True,
                                    title="Yearly Growth in Insurance Value"))
```

```
        # 3️⃣
        elif insurance_option.startswith("3️⃣"):
            q = """
            SELECT Year, Quater, SUM(Insurance_amount) AS Total_Value
            FROM Aggregated_Insurance_data GROUP BY Year, Quater ORDER BY Year,
Quater;
            """
            df = run_query(q)
            st.subheader("📅 Quarter-wise Distribution of Insurance Value")
            st.dataframe(df)
            st.plotly_chart(px.bar(df, x="Quater", y="Total_Value", color="Year",
                                   title="Quarterly Insurance Value Distribution",
barmode="group"))

        # 4️⃣
        elif insurance_option.startswith("4️⃣"):
            q = """
            SELECT State, SUM(Insurance_count) AS Total_Transactions,
SUM(Insurance_amount) AS Total_Value
            FROM Aggregated_Insurance_data
            WHERE Year = (SELECT MAX(Year) FROM Aggregated_Insurance_data)
            GROUP BY State ORDER BY Total_Value DESC LIMIT 5;
            """
            df = run_query(q)
            st.subheader("🌟 Top 5 High-Value States in the Latest Year")
            st.dataframe(df)
            st.plotly_chart(px.pie(df, names="State", values="Total_Value",
                                   title="Insurance Value Share – Top 5 States",
hole=0.4))

        # 5️⃣
        elif insurance_option.startswith("5️⃣"):
            q = """
            SELECT a.State, a.Year,
                   SUM(a.Insurance_amount) AS Total_Value,
                   ROUND(
                       (SUM(a.Insurance_amount) - LAG(SUM(a.Insurance_amount)) OVER
(PARTITION BY a.State ORDER BY a.Year))
                       / NULLIF(LAG(SUM(a.Insurance_amount)) OVER (PARTITION BY
a.State ORDER BY a.Year), 0) * 100, 2
                   ) AS YoY_Growth_Percentage
            FROM Aggregated_Insurance_data a
            GROUP BY a.State, a.Year
            ORDER BY a.State, a.Year;
            """
            df = run_query(q)
            st.subheader("📊 YoY Growth Rate of Insurance Value by State")
            st.dataframe(df)
            st.plotly_chart(px.line(df, x="Year", y="YoY_Growth_Percentage",
```

```
color="State",
                                    title="Year-over-Year Growth Rate by State",
markers=True))

        st.success(f"☑ Analysis Loaded: {insurance_option}")



        # ================================
    # 👥 BUSINESS CASE 4 - User Engagement & Growth Strategy
    # ================================
    elif case_study.startswith("4 "):
        import plotly.express as px

        st.subheader("👥 User Engagement & Growth Strategy")

        # --- Overview Query ---
        overview_query = """
        SELECT
            State, Year, Quater,
            SUM(RegisteredUsers) AS Total_Users,
            SUM(AppOpens) AS Total_App_Opens
        FROM Map_User_data
        GROUP BY State, Year, Quater
        ORDER BY Year, Quater;
        """
        overview_df = run_query(overview_query)

        st.markdown("### 📊 User Engagement Overview Data")
        st.dataframe(overview_df)
        st.bar_chart(overview_df.groupby("Year")[["Total_Users",
"Total_App_Opens"]].sum())
        show_top_bottom(overview_df, "Total_Users")

        # --- Sidebar options for 5 analysis queries ---
        st.sidebar.markdown("---")
        st.sidebar.header("📈 Choose Analysis Type")
        engagement_option = st.sidebar.selectbox(
            "Select Query / Analysis View",
            [
                "1️⃣ Top 10 States by Total App Engagement",
                "2️⃣ Yearly Growth of User Base & Engagement",
                "3️⃣ Quarterly Engagement Pattern",
                "4️⃣ Engagement Efficiency (App Opens per User)",
                "5️⃣ YoY Growth in App Engagement by State"
            ]
        )

        # 1️⃣ Top 10 States by Total App Engagement
        if engagement_option.startswith("1️⃣"):
```

```
    q = """
    SELECT
        State,
        SUM(AppOpens) AS Total_App_Opens,
        SUM(RegisteredUsers) AS Total_Users
    FROM Map_User_data
    GROUP BY State
    ORDER BY Total_App_Opens DESC
    LIMIT 10;
    """
    df = run_query(q)
    st.subheader("⏳ Top 10 States by Total App Engagement")
    st.dataframe(df)
    st.plotly_chart(
        px.bar(df, x="State", y="Total_App_Opens", color="Total_App_Opens",
               text_auto=True, title="Top 10 States by Total App Opens"),
        use_container_width=True
    )

# 2️⃣ Yearly Growth of User Base & Engagement
elif engagement_option.startswith("2️"):
    q = """
    SELECT
        Year,
        SUM(RegisteredUsers) AS Total_Users,
        SUM(AppOpens) AS Total_App_Opens
    FROM Map_User_data
    GROUP BY Year
    ORDER BY Year;
    """
    df = run_query(q)
    st.subheader("📈 Yearly Growth of User Base & Engagement")
    st.dataframe(df)
    st.plotly_chart(
        px.line(df, x="Year", y="Total_App_Opens", markers=True,
                title="Yearly Growth in App Engagement"),
        use_container_width=True
    )

# 3️⃣ Quarterly Engagement Pattern
elif engagement_option.startswith("3️"):
    q = """
    SELECT
        Year,
        Quater,
        SUM(AppOpens) AS Total_App_Opens
    FROM Map_User_data
    GROUP BY Year, Quater
    ORDER BY Year, Quater;
    """
```

```
            df = run_query(q)
            st.subheader("📅 Quarterly Engagement Pattern")
            st.dataframe(df)
            st.plotly_chart(
                px.bar(df, x="Quater", y="Total_App_Opens", color="Year",
                       title="Quarterly Engagement Pattern", barmode="group"),
                use_container_width=True
            )

        # 4️⃣ Engagement Efficiency (App Opens per User)
        elif engagement_option.startswith("4️"):
            q = """
            SELECT
                State,
                ROUND(SUM(AppOpens) / NULLIF(SUM(RegisteredUsers), 0), 2) AS
AppOpens_per_User
            FROM Map_User_data
            GROUP BY State
            ORDER BY AppOpens_per_User DESC
            LIMIT 10;
            """
            df = run_query(q)
            st.subheader("⚡ Engagement Efficiency (App Opens per User by State)")
            st.dataframe(df)
            st.plotly_chart(
                px.bar(df, x="AppOpens_per_User", y="State", orientation='h',
color="AppOpens_per_User",
                       text_auto=True, title="Engagement Efficiency by State"),
                use_container_width=True
            )

        # 5️⃣ YoY Growth in App Engagement by State
        elif engagement_option.startswith("5️"):
            q = """
            SELECT
                State,
                Year,
                SUM(AppOpens) AS Total_App_Opens,
                ROUND(
                    (SUM(AppOpens) - LAG(SUM(AppOpens)) OVER (PARTITION BY State
ORDER BY Year))
                    / NULLIF(LAG(SUM(AppOpens)) OVER (PARTITION BY State ORDER BY
Year), 0) * 100, 2
                ) AS YoY_Growth_Percentage
            FROM Map_User_data
            GROUP BY State, Year
            ORDER BY State, Year;
            """
            df = run_query(q)
            st.subheader("📊 YoY Growth in App Engagement by State")
```

```
            st.dataframe(df)
            st.plotly_chart(
                px.line(df, x="Year", y="YoY_Growth_Percentage", color="State",
markers=True,
                        title="YoY Growth in App Engagement by State"),
                use_container_width=True
            )

        st.success(f"☑ Analysis Loaded: {engagement_option}")



    # ============================================
    # 🌍 BUSINESS CASE 5 - Transaction Analysis Across States and Districts
    # ============================================
    elif case_study.startswith("5"):
        import plotly.express as px

        st.subheader("🌍 Transaction Analysis Across States and Districts")

        # --- Overview Query ---
        overview_query = """
        SELECT
            State, Year, Quater,
            SUM(Transaction_count) AS Total_Transactions,
            SUM(Transaction_amount) AS Total_Value
        FROM Map_Transaction_data
        GROUP BY State, Year, Quater
        ORDER BY Year, Quater, Total_Value DESC;
        """
        overview_df = run_query(overview_query)

        st.markdown("### 📊 Overview Data")
        st.dataframe(overview_df)
        st.bar_chart(overview_df.groupby("Year")[["Total_Transactions",
"Total_Value"]].sum())
        show_top_bottom(overview_df, "Total_Value")

        # --- Sidebar Analysis Options ---
        st.sidebar.markdown("---")
        st.sidebar.header("📈 Choose Analysis Type")
        transaction_option = st.sidebar.selectbox(
            "Select Query / Analysis View",
            [
                "1️⃣ Top 10 States by Total Transaction Value",
                "2️⃣ Yearly Growth Trend of Transactions Across India",
                "3️⃣ Quarter-wise Performance by State",
                "4️⃣ District-wise Transaction Insights (Top 10 Districts)",
                "5️⃣ Identify Low-Performing States (Bottom 10)"
            ]
```

```
)

# 1️⃣ Top 10 States by Transaction Value
if transaction_option.startswith("1️⃣"):
    q = """
    SELECT
        State,
        SUM(Transaction_count) AS Total_Transactions,
        SUM(Transaction_amount) AS Total_Value
    FROM Map_Transaction_data
    GROUP BY State
    ORDER BY Total_Value DESC
    LIMIT 10;
    """
    df = run_query(q)
    st.subheader("⏳ Top 10 States by Total Transaction Value")
    st.dataframe(df)
    st.plotly_chart(
        px.bar(df, x="State", y="Total_Value", color="Total_Value",
               text_auto=True, title="Top 10 States by Transaction Value"),
        use_container_width=True
    )

# 2️⃣ Yearly Growth Trend
elif transaction_option.startswith("2️⃣"):
    q = """
    SELECT
        Year,
        SUM(Transaction_count) AS Total_Transactions,
        SUM(Transaction_amount) AS Total_Value
    FROM Map_Transaction_data
    GROUP BY Year
    ORDER BY Year;
    """
    df = run_query(q)
    st.subheader("📈 Yearly Growth Trend of Transactions Across India")
    st.dataframe(df)
    st.plotly_chart(
        px.line(df, x="Year", y="Total_Value", markers=True,
                title="Yearly Transaction Value Growth"),
        use_container_width=True
    )

# 3️⃣ Quarter-wise Performance by State
elif transaction_option.startswith("3️⃣"):
    q = """
    SELECT
        State, Year, Quater,
        SUM(Transaction_amount) AS Total_Value
    FROM Map_Transaction_data
```

```
            GROUP BY State, Year, Quater
            ORDER BY Year, Quater;
            """
            df = run_query(q)
            st.subheader("📅 Quarter-wise Performance by State")
            st.dataframe(df)
            st.plotly_chart(
                px.bar(df, x="Quater", y="Total_Value", color="State",
                       title="Quarter-wise Transaction Value by State",
barmode="group"),
                use_container_width=True
            )

        # 4️⃣ District-wise Insights (Top 10 Districts)
        elif transaction_option.startswith("4️"):
            q = """
            SELECT
                District,
                SUM(Transaction_count) AS Total_Transactions,
                SUM(Transaction_amount) AS Total_Value
            FROM Top_Transaction_data
            GROUP BY District
            ORDER BY Total_Value DESC
            LIMIT 10;
            """
            df = run_query(q)
            st.subheader("🏙️ Top 10 Districts by Transaction Value")
            st.dataframe(df)
            st.plotly_chart(
                px.bar(df, x="District", y="Total_Value", color="Total_Value",
                       text_auto=True, title="Top 10 Districts by Transaction
Value"),
                use_container_width=True
            )

        # 5️⃣ Identify Low-Performing States
        elif transaction_option.startswith("5️"):
            q = """
            SELECT
                State,
                SUM(Transaction_count) AS Total_Transactions,
                SUM(Transaction_amount) AS Total_Value
            FROM Map_Transaction_data
            GROUP BY State
            ORDER BY Total_Value ASC
            LIMIT 10;
            """
            df = run_query(q)
            st.subheader("⚠️ Identify Low-Performing States (Bottom 10)")
            st.dataframe(df)
```

```
            st.plotly_chart(
                px.bar(df, x="State", y="Total_Value", color="Total_Value",
                       text_auto=True, title="Bottom 10 States by Transaction
Value"),
                use_container_width=True
            )

        st.success(f"☑ Analysis Loaded: {transaction_option}")
```