

EXPT NO: 8	MINI PROJECT – Comprehensive E-Commerce Sales Dashboards: Real-Time Insights and Data-Driven Growth

## AIM

To design and develop an interactive e-commerce sales dashboard that provides real-time insights into sales performance, customer trends, and revenue growth using data visualization and analytics techniques.

## ALGORITHM

1. Start
2. Import Dataset: Load the e-commerce sales dataset (CSV or Excel file).
3. Data Cleaning:
  - Remove missing and duplicate records.
  - Convert date and numeric fields into proper formats.
4. Data Preprocessing:
  - Calculate total sales = Quantity × Price.
  - Group data by category, region, and time.
5. Visualization:
  - Create bar charts for category-wise sales.
  - Use line graphs for sales over time.
  - Use pie charts for regional or product share.
  - Add KPI cards for total revenue, profit, and orders.
6. Dashboard Design:
  - Combine all visualizations in one dashboard using Power BI / Tableau / Python.
  - Enable filters for category, region, and date.
7. Generate Insights:
  - Identify top-selling products.
  - Observe seasonal sales trends.
8. Display Final Dashboard showing real-time updates and insights.
9. Stop

## CODING

```
# dashboard.py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import streamlit as st
import plotly.express as px

# Page config
st.set_page_config(page_title="E-commerce Sales Dashboard", layout="wide")
st.title("E-commerce Sales Dashboard (Interactive & Real-time)")
```

```

# ----- Load Data -----
df = pd.read_csv("sales.csv")

df["Sales"] = pd.to_numeric(df["Sales"], errors="coerce")
df["Profit"] = pd.to_numeric(df.get("Profit", 0), errors="coerce")
df["Order Date"] = pd.to_datetime(df["Order Date"], errors="coerce")
df["Month"] = df["Order Date"].dt.to_period("M").astype(str)
df["Year"] = df["Order Date"].dt.year
df["Day"] = df["Order Date"].dt.day

# ----- Widgets for Interactivity -----
st.sidebar.header("🔍 Filters (Real-time)")
category_filter = st.sidebar.multiselect("Select Category", df["Category"].unique(),
default=df["Category"].unique())
product_filter = st.sidebar.text_input("Search Product (Type any name)", "")
date_range = st.sidebar.date_input("Select Date Range", [df["Order Date"].min(), df["Order Date"].max()])
top_n_products = st.sidebar.number_input("Top N Products", min_value=1, max_value=20,
value=10)
profit_target = st.sidebar.slider("Profit Margin Target", 0, 100, 20)

# Apply filters
filtered_df = df[df["Category"].isin(category_filter)]
if product_filter:
    filtered_df = filtered_df[filtered_df["Product Name"].str.contains(product_filter,
case=False)]
filtered_df = filtered_df[(filtered_df["Order Date"] >= pd.to_datetime(date_range[0])) &
(filtered_df["Order Date"] <= pd.to_datetime(date_range[1]))]

# ----- Feature 1: Total Sales -----
st.subheader(" Total Sales") st.metric("📈 Total Sales",
f"${filtered_df['Sales'].sum():.2f}")

# ----- Feature 2: Monthly Sales Trend -----
st.subheader(" Monthly Sales Trend")
monthly_sales = filtered_df.groupby("Month")["Sales"].sum().reset_index()
fig, ax = plt.subplots(figsize=(10,5))
sns.lineplot(x="Month", y="Sales", data=monthly_sales, marker="o", ax=ax)
plt.xticks(rotation=45)
st.pyplot(fig)

# ----- Feature 3: Daily Sales Trend -----
st.subheader(" Daily Sales Trend")
daily_sales = filtered_df.groupby("Order Date")["Sales"].sum().reset_index()
fig = px.line(daily_sales, x="Order Date", y="Sales", title="Daily Sales")
st.plotly_chart(fig, use_container_width=True)

```

```

# ----- Feature 4: Sales by Category -----
st.subheader(" Sales by Category")
category_sales = filtered_df.groupby("Category")["Sales"].sum().reset_index()
fig = px.bar(category_sales, x="Category", y="Sales", title="Category-wise Sales",
text_auto=True)
st.plotly_chart(fig, use_container_width=True)

# ----- Feature 5: Sub-category Sales -----
st.subheader(" Sales by Sub-Category")
sub_sales = filtered_df.groupby("Sub-
Category")["Sales"].sum().reset_index().sort_values("Sales", ascending=False)
fig = px.bar(sub_sales, x="Sub-Category", y="Sales", title="Sub-Category-wise Sales")
st.plotly_chart(fig, use_container_width=True)

# ----- Feature 6: Region-wise Sales -----
st.subheader(" Sales by Region")
region_sales = filtered_df.groupby("Region")["Sales"].sum().reset_index()
fig = px.pie(region_sales, names="Region", values="Sales", title="Sales by Region")
st.plotly_chart(fig, use_container_width=True)

# ----- Feature 7: Sales by Country -----
if "Country" in filtered_df.columns:
    st.subheader(" SalesbyCountry")
    fig = px.choropleth(filtered_df, locations="Country", locationmode="country names",
        color="Sales", title="Salesby Country")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 8: Top N Products -----
st.subheader(" Top Selling Products")
top_products = filtered_df.groupby("Product
Name")["Sales"].sum().nlargest(top_n_products).reset_index()
fig = px.bar(top_products, x="Sales", y="Product Name", orientation="h", title=f"Top
{top_n_products} Products")
st.plotly_chart(fig, use_container_width=True)

# ----- Feature 9: Customer Segment Sales -----
if "Segment" in filtered_df.columns:
    st.subheader(" Sales by Customer Segment")
    segment_sales = filtered_df.groupby("Segment")["Sales"].sum().reset_index()
    fig = px.pie(segment_sales, names="Segment", values="Sales", title="Segment-wise
Sales")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 10: Yearly Sales -----
st.subheader(" YearlySales")
yearly_sales = filtered_df.groupby("Year")["Sales"].sum().reset_index()

```

```

fig = px.bar(yearly_sales, x="Year", y="Sales", title="Yearly Sales")
st.plotly_chart(fig, use_container_width=True)

# ----- Feature 11: Profit by Category -----
if "Profit" in filtered_df.columns:
    st.subheader(" Profit by Category")
    profit_category = filtered_df.groupby("Category")["Profit"].sum().reset_index()
    fig = px.bar(profit_category, x="Category", y="Profit", title="Profit by Category")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 12: Profit vs Sales Scatterplot -----
if "Profit" in filtered_df.columns:
    st.subheader(" Profit vs Sales")
    fig = px.scatter(filtered_df, x="Sales", y="Profit", color="Category", size="Sales",
title="Profit vs Sales")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 13: Average Order Value -----

- st.subheader(" Average OrderValue") avg_order =

filtered_df.groupby("Order ID")["Sales"].sum().mean()

st.metric("AvgOrderValue", f"${avg_order:,.2f}")
# ----- Feature 14: Sales Heatmap -----
st.subheader(" Sales Heatmap (Day vs Month)")
heatmap_data = filtered_df.pivot_table(index="Day", columns="Month", values="Sales",
aggfunc="sum")
fig, ax = plt.subplots(figsize=(12,6)) sns.heatmap(heatmap_data, cmap="Blues", ax=ax)
st.pyplot(fig)

# ----- Feature 15: Sales by Payment Mode -----
if "Payment Mode" in filtered_df.columns:
    st.subheader(" Sales by Payment Mode")
    payment_sales = filtered_df.groupby("Payment Mode")["Sales"].sum().reset_index()
    fig = px.pie(payment_sales, names="Payment Mode", values="Sales", title="Payment
Methods")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 16: State-wise Sales -----
if "State" in filtered_df.columns:
    st.subheader(" State-wise Sales")
    state_sales =
filtered_df.groupby("State")["Sales"].sum().reset_index().sort_values("Sales",
ascending=False).head(10)
    fig = px.bar(state_sales, x="State", y="Sales", title="Top 10 States by Sales")
    st.plotly_chart(fig, use_container_width=True)

```

```

# ----- Feature 17: Ship Mode Analysis -----
if "Ship Mode" in filtered_df.columns:
    st.subheader(" Sales by Shipping Mode")
    ship_sales = filtered_df.groupby("Ship Mode")["Sales"].sum().reset_index()
    fig = px.bar(ship_sales, x="Ship Mode", y="Sales", title="Sales by Ship Mode")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 18: Profit Margin -----
if "Profit" in filtered_df.columns:
    st.subheader(" Profit Margin")
    profit_margin = (filtered_df["Profit"].sum() / filtered_df["Sales"].sum()) * 100
    st.metric(" ProfitMargin", f"{profit_margin:.2f}%")

# ----- Feature19: SalesvsDiscount -----
if "Discount" in filtered_df.columns:
    st.subheader(" Sales vs Discount")
    fig = px.scatter(filtered_df, x="Discount", y="Sales", color="Category", title="Sales vs
Discount")
    st.plotly_chart(fig, use_container_width=True)

# ----- Feature 20: Correlation Heatmap -----
----
st.subheader(" Correlation Heatmap")
corr = filtered_df.corr(numeric_only=True)
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", ax=ax)
st.pyplot(fig)
# ----- Extra Feature 21: Product Filter -----
product_sales = filtered_df.groupby("Product Name")["Sales"].sum().reset_index()
fig = px.bar(product_sales, x="Product Name", y="Sales", title=f"Sales for
st.subheader(" Filtered Product Sales")
if product_filter:
    '{product_filter}')
    st.plotly_chart(fig, use_container_width=True)
# ----- Extra Feature 22: Date Range Sales -----
st.subheader(" Sales in Selected Date Range")
date_filtered_sales = filtered_df.groupby("Order Date")["Sales"].sum().reset_index()
fig = px.line(date_filtered_sales, x="Order Date", y="Sales", title="Sales Over Selected
Dates")
st.plotly_chart(fig, use_container_width=True)

# ----- Extra Feature 23: Segment Filter -----
if "Segment" in filtered_df.columns:
    st.subheader(" SegmentFilter")

```

```

segments = filtered_df.groupby("Segment")["Sales"].sum().reset_index() fig
= px.bar(segments, x="Segment", y="Sales", title="Sales by Segment")
st.plotly_chart(fig, use_container_width=True)

```

# ----- Extra Feature 24: Profit Target Highlight -----

```

st.subheader(" Products Below Profit Target")
low_profit_products = filtered_df[filtered_df["Profit"] < profit_target]
fig = px.bar(low_profit_products.groupby("Product Name")["Profit"].sum().reset_index(),
             x="Product Name", y="Profit", title="Products with Profit < {profit_target}")
st.plotly_chart(fig, use_container_width=True)

```

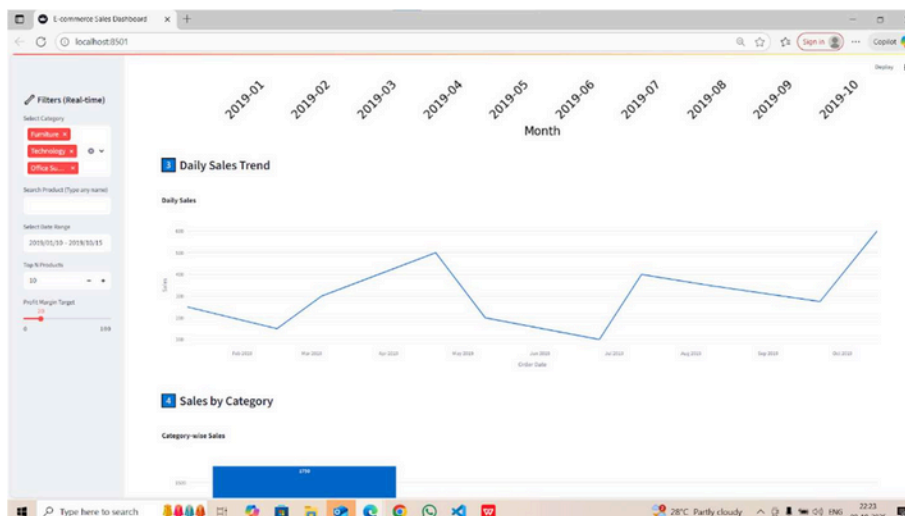
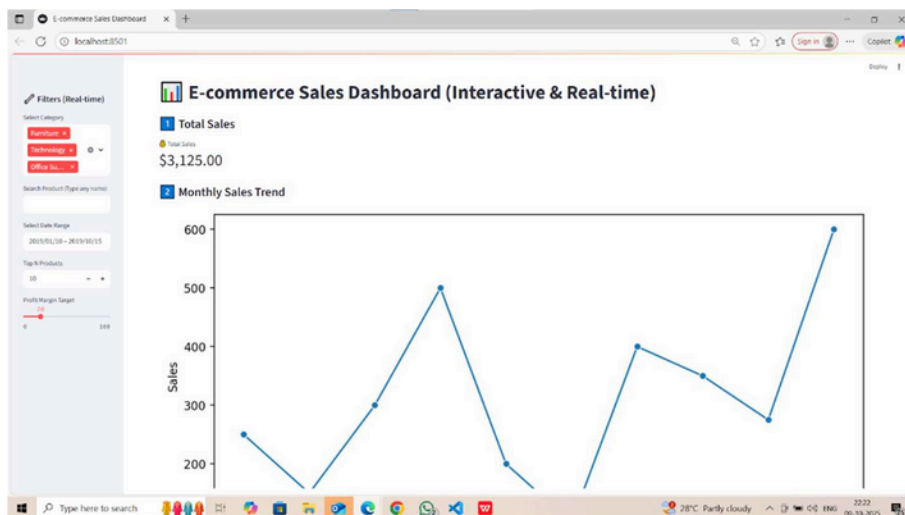
# ----- Extra Feature 25: Top N Products (Interactive) -----

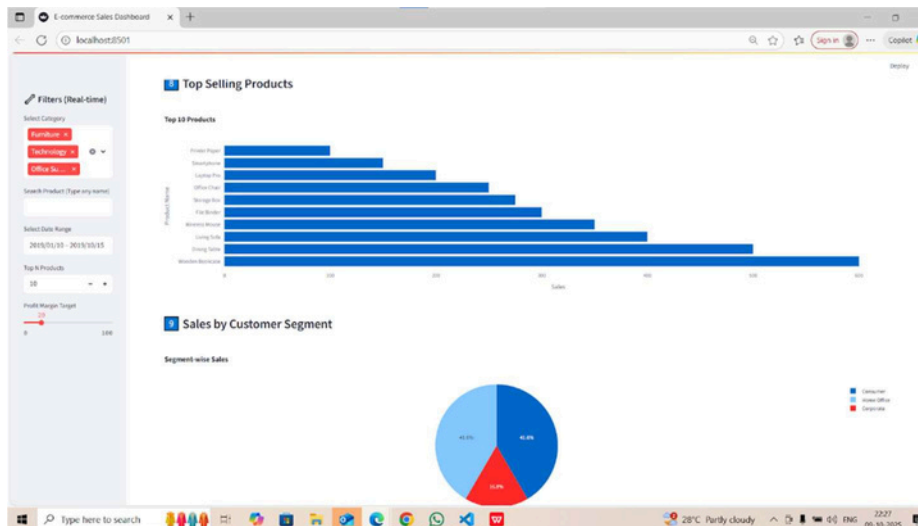
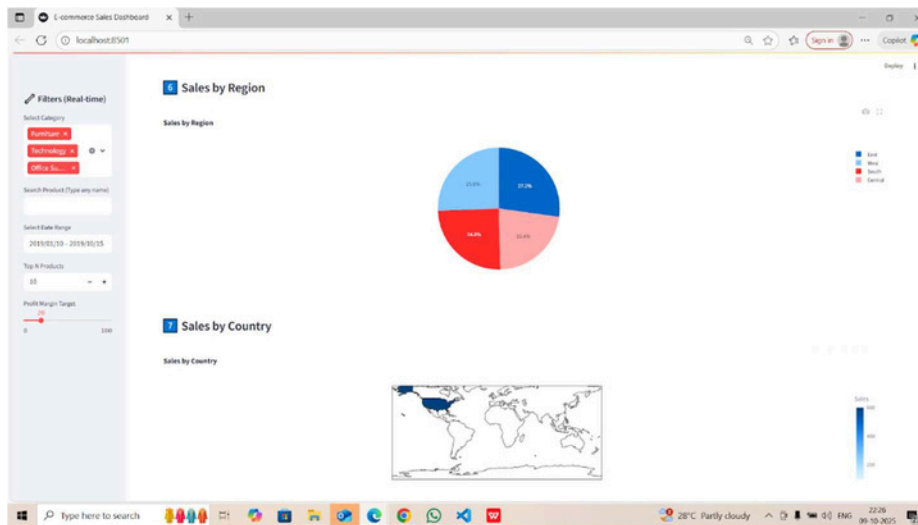
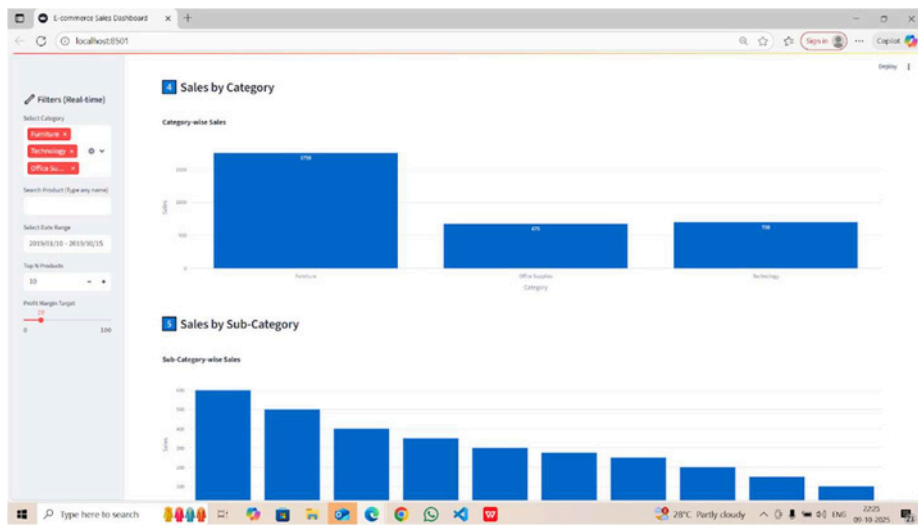
```

st.subheader(" Top N Products Selector")
# Already done above using top_n_products in sidebar

```

## OUTPUT :





**RESULT:**

The project successfully visualized the e-commerce sales data using real-time dashboards. The system helps in monitoring sales performance, identifying top-selling products, and supporting data-driven decisions for business growth.