# NAME: PRTHAPJ

# TOPIC: Retail Sales Dashboard Report

## **Introduction**

**This document provides an overview of a Python-based retail sales analysis dashboard. The program processes sales data, generates visualizations, and automates PowerPoint generation. This approach helps businesses analyze sales trends and category performance efficiently.**

Functions & Their Roles:

* **load\_sales\_data():** Loads and preprocesses the sales data.
* **aggregate\_sales\_data()**: Computes sales trends by month and category.
* **create\_sales\_figures():** Generates visualizations using Plotly.
* **save\_figures()**: Saves the generated images.

Benefits:

* Code Readability: Well-structured functions improve readability.
* Reusability: The functions can be reused in different projects
* Maintainability: Changes can be made easily without affecting other parts of the code.

## Advantages

1. Automates sales data analysis and visualization.

2. Generates PowerPoint slides with insights automatically.

3. Uses Python for data processing and visualization.

4. Provides interactive graphs and insights.

## Solution Explanation

The solution is divided into the following modules:  
1. Data Processing: Cleans and aggregates sales data.  
2. Visualization: Generates charts using Plotly.  
3. Report Generation: Saves visualizations and creates a PowerPoint file.

**CODE IMPLEMENTATION**

import pandas as pd

import plotly.express as px

import dash

from dash import dcc, html

from dash.dependencies import Input, Output

# Load Sales Data

df = pd.read\_csv('/home/asu/sales\_data1.csv')

# Strip spaces from column names

df.columns = df.columns.str.strip()

# Convert Date column to datetime

df['Date'] = pd.to\_datetime(df['Date'])

# Group sales by month

sales\_trend = df.groupby(df['Date'].dt.to\_period('M')).agg({'Total\_Sales': 'sum'}).reset\_index()

sales\_trend['Date'] = sales\_trend['Date'].astype(str) # Convert period to string

# Group sales by category

category\_sales = df.groupby('Category').agg({'Total\_Sales': 'sum'}).reset\_index()

# Initialize Dash app

app = dash.Dash(\_\_name\_\_)

# App Layout

app.layout = html.Div([

html.Div([

html.H2("Retail Sales Dashboard", style={'textAlign': 'center', 'color': 'white'}),

html.Hr(),

html.Label("Select Month:", style={'fontWeight': 'bold', 'color': 'white'}),

dcc.Dropdown(

id='month-dropdown',

options=[{'label': month, 'value': month} for month in sales\_trend['Date']],

value=sales\_trend['Date'].iloc[-1], # Default to latest month

clearable=False,

style={'width': '90%', 'margin': 'auto'}

),

], style={'width': '20%', 'backgroundColor': '#003366', 'padding': '20px', 'position': 'fixed', 'height': '100vh', 'color': 'black'}),

html.Div([

html.Div([

html.H3("Sales Trend Over Time", style={'color': '#006699'}),

dcc.Graph(id='sales-trend')

], style={'backgroundColor': '#f8f9fa', 'padding': '20px', 'borderRadius': '10px', 'marginBottom': '20px'}),

html.Div([

html.H3("Sales by Product Category", style={'color': '#006699'}),

dcc.Graph(id='category-sales')

], style={'backgroundColor': '#f8f9fa', 'padding': '20px', 'borderRadius': '10px'}),

], style={'marginLeft': '22%', 'padding': '20px'})

], style={'fontFamily': 'Arial', 'backgroundColor': '#e9ecef'})

# Callbacks to update charts

@app.callback(

Output('sales-trend', 'figure'),

Output('category-sales', 'figure'),

Input('month-dropdown', 'value')

)

def update\_graphs(selected\_month):

filtered\_df = df[df['Date'].dt.to\_period('M').astype(str) == selected\_month]

sales\_trend\_filtered = filtered\_df.groupby(filtered\_df['Date'].dt.date).agg({'Total\_Sales': 'sum'}).reset\_index()

sales\_trend\_filtered['Date'] = sales\_trend\_filtered['Date'].astype(str) # Convert date to string

trend\_fig = px.line(

sales\_trend\_filtered,

x='Date', y='Total\_Sales',

title=f"Sales Trend for {selected\_month}",

markers=True,

color\_discrete\_sequence=['#FF5733']

)

category\_sales\_filtered = filtered\_df.groupby('Category').agg({'Total\_Sales': 'sum'}).reset\_index()

category\_fig = px.bar(

category\_sales\_filtered, x='Category', y='Total\_Sales',

title=f"Product Sales by Category for {selected\_month}",

color='Category',

color\_discrete\_sequence=px.colors.qualitative.Pastel

)

return trend\_fig, category\_fig

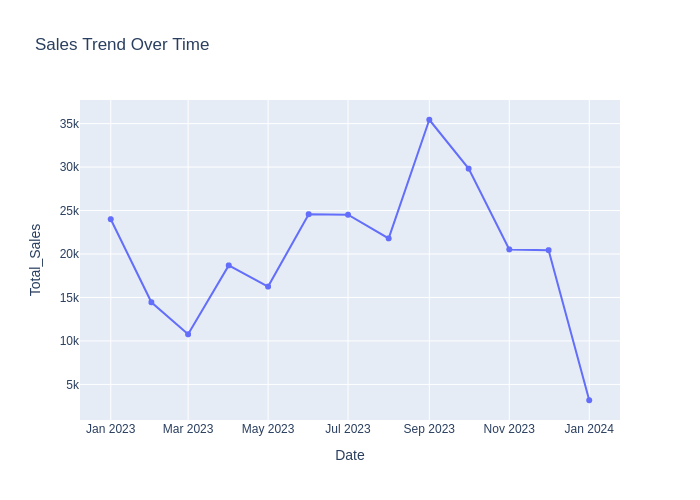
# Run server

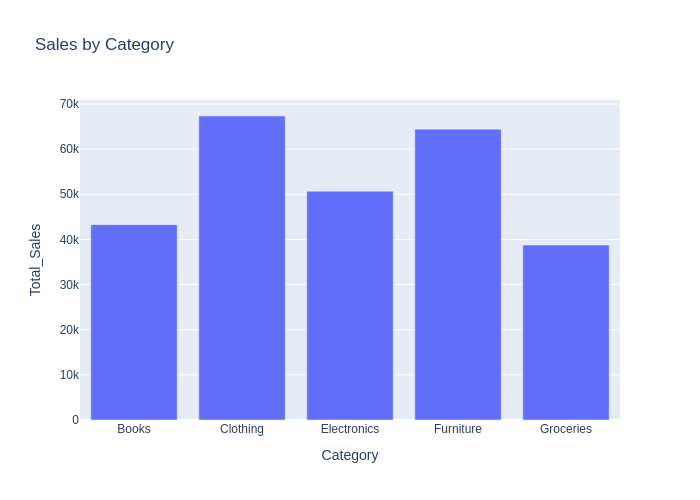
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

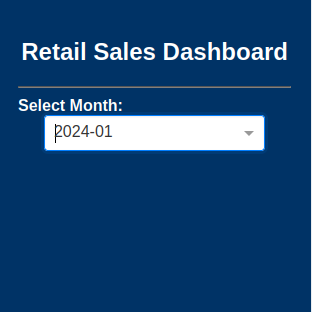
## OUTPUT:

## Generated Visualizations





**CATEGORY\_SALES**

****

