# Mini Project

## Daily vs Monthly Sales Dashboard

## Abstract

## The Daily vs Monthly Sales Analytics Dashboard is an interactive and scalable business intelligence tool built using Python Dash, Plotly, and Pandas, designed to help businesses analyze and compare sales performance over different time scales. Users can upload their own sales datasets in CSV format, and the system automatically processes the data to generate clean, responsive, and professional visualizations. Key features include KPI cards for instant insights, daily vs monthly trend comparison charts, and customizable filters for category, date range, and year. With its Bootstrap-based modern layout and ability to handle large datasets, the dashboard is suitable for enterprises and SMEs, enabling data-driven decision-making. Beyond academic submission, this solution can be directly commercialized as a ready-to-use analytics platform for large-scale business applications.

## Introduction

## The Daily vs Monthly Sales Dashboard is an interactive data visualization tool designed to compare sales performance across different time frames. Built with Python Dash, Plotly, and Pandas, it transforms raw CSV data into clear visual insights through KPI cards, trend charts, and filters. With its clean, responsive design, the dashboard helps businesses quickly identify patterns, track growth, and make informed, data-driven decisions.

## Implementation

1. **Dataset Selection**

* Sale ID, Date, Product Category, Region, Sales Amount, Quantity Sold, Payment Method

2. **Data Preprocessing**

* Handling missing values, encoding categorical variables, normalizing numerical features, converting date fields for time-series analysis

3. **Visualization Tools Used**

* Python (Pandas, Matplotlib, Seaborn, Plotly), optional dashboard integration (Streamlit, Power BI)

4. **Types of Visualizations**

* Daily vs. monthly sales trends, product category performance, regional sales distribution, payment method breakdown, and interactive line/bar charts

**Technologies Used**

**Programming Language**

* Python

**Python Libraries**

* **Pandas** – Data loading, cleaning, and preprocessing
* **Matplotlib** – Static data visualization
* **Seaborn** – Statistical and advanced visualization
* **Plotly** – Interactive charts for dashboards

**Data Processing**

* CSV files for business data storage and retrieval

**Visualization & Dashboard Tools (Optional)**

* Streamlit – Web-based interactive dashboards
* Power BI – Business Intelligence reporting

**Development Environment**

* Jupyter Notebook / VS Code / PyCharm

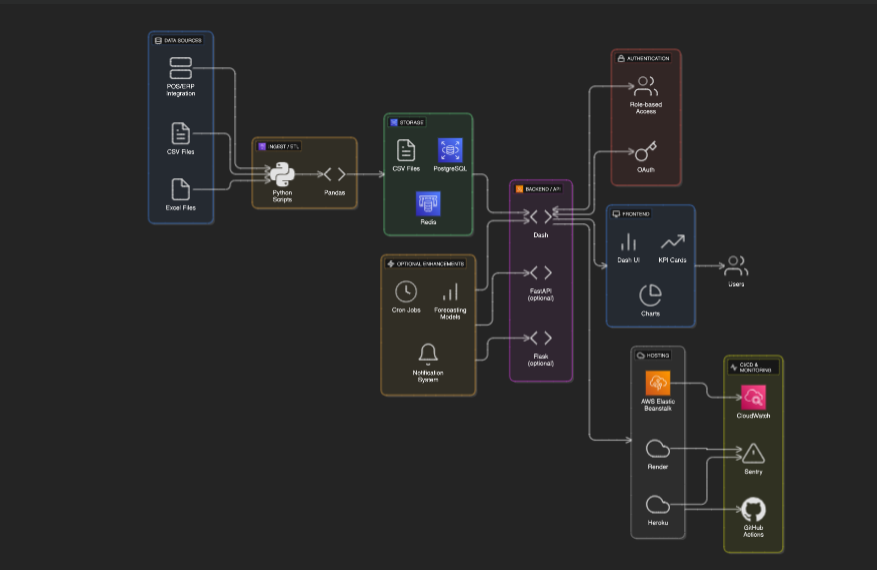
**Version Control (Optional)**

* Git & GitHub for source code management

## Dataset used:

* business\_sales\_sample.csv

## Architecture Diagram

****

**Algorithm**

1. **Start**
2. **Import Required Libraries** – Pandas, Matplotlib, Seaborn, Plotly.
3. **Load Dataset** – Read the sales CSV file into a Pandas DataFrame.
4. **Data Preprocessing**
   * Handle missing values.
   * Encode categorical variables if required.
   * Normalize/scale numerical features.
   * Convert date columns to proper datetime format.
5. **Data Analysis & Aggregation**
   * Calculate total sales, profit, and revenue metrics.
   * Group data by month/year, product category, region, etc.
6. **Generate Visualizations**
   * Bar charts (Monthly vs Daily Sales).
   * Pie charts (Product category or region share).
   * Line graphs (Sales trends over time).
   * Heatmaps (Correlation between metrics).
7. **Interactive Dashboard (Optional)**
   * Integrate visualizations into a Streamlit or Plotly dashboard.
8. **Display & Export**
   * Show graphs in the dashboard.
   * Save charts as PNG/JPEG for reports.
9. **End**

## Program

import dash

from dash import html, dcc, Output, Input, State

import plotly.express as px

import pandas as pd

import base64, io

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

app.title = "Daily Vs Monthly Sales Dashboard"

app.layout = html.Div(style={'padding': '20px', 'backgroundColor': '#f4f6f8'}, children=[

html.Div("📊 DAILY VS MONTHLY SALES DASHBOARD",

style={'textAlign': 'center', 'fontSize': '30px',

'fontWeight': 'bold', 'color': '#333', 'marginBottom': '20px'}),

dcc.Upload(

id='upload-data',

children=html.Div('Drag & Drop or Select CSV/Excel (auto-detects Date & Sales)'),

style={

'width': '100%', 'height': '50px', 'lineHeight': '50px',

'borderWidth': '1px', 'borderStyle': 'dashed',

'borderRadius': '5px', 'textAlign': 'center',

'marginBottom': '20px', 'backgroundColor': '#fff'

},

multiple=False

),

html.Div(id='file-info', style={'textAlign': 'center', 'marginBottom': '20px', 'color': '#555'}),

html.Div(id='dashboard-content')

])

def parse\_contents(contents, filename):

content\_type, content\_string = contents.split(',')

decoded = base64.b64decode(content\_string)

try:

if filename.endswith('.csv'):

df = pd.read\_csv(io.StringIO(decoded.decode('utf-8')))

elif filename.endswith(('.xls', '.xlsx')):

df = pd.read\_excel(io.BytesIO(decoded))

else:

return None

except Exception as e:

print(e)

return None

return df

@app.callback(

[Output('file-info', 'children'),

Output('dashboard-content', 'children')],

Input('upload-data', 'contents'),

State('upload-data', 'filename')

)

def update\_dashboard(contents, filename):

if contents is None:

return "", ""

df = parse\_contents(contents, filename)

if df is None:

return "❌ Unsupported file format.", ""

if 'Month' not in df.columns:

if 'Date' in df.columns:

df['Month'] = pd.to\_datetime(df['Date']).dt.strftime('%B')

if 'DailyTotal' not in df.columns:

df['DailyTotal'] = df['Sales']

total\_sales = df['Sales'].sum()

avg\_daily\_sales = df['DailyTotal'].mean()

best\_month = df.loc[df['Sales'].idxmax(), 'Month']

fig\_monthly = px.bar(df, x='Month', y=['Sales', 'DailyTotal'], barmode='group',

title="Daily vs Monthly Sales",

color\_discrete\_sequence=px.colors.qualitative.Set2)

fig\_monthly.update\_layout(plot\_bgcolor='white', paper\_bgcolor='white')

fig\_daily\_trend = px.line(df, x='Month', y='Sales', markers=True,

title="Daily Sales Trend", color\_discrete\_sequence=['#1f77b4'])

fig\_daily\_trend.update\_layout(plot\_bgcolor='white', paper\_bgcolor='white')

fig\_monthly\_summary = px.bar(df, x='Month', y='DailyTotal',

title="Monthly Sales Summary",

color\_discrete\_sequence=['#2ca02c'])

fig\_monthly\_summary.update\_layout(plot\_bgcolor='white', paper\_bgcolor='white')

fig\_pie = px.pie(df, names='Month', values='Sales', title="Sales Share by Month",

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

fig\_scatter = px.scatter(df, x='Month', y='Sales', size='Sales',

title="Sales Distribution by Month",

color='Sales', color\_continuous\_scale='Viridis')

dashboard = html.Div([

html.Div(style={'display': 'flex', 'gap': '20px', 'justifyContent': 'center', 'marginBottom': '20px'}, children=[

html.Div([

html.H3("💰 Total Sales", style={'margin': '0', 'color': '#444'}),

html.H2(f"{total\_sales:,}", style={'margin': '0', 'color': '#1f77b4'})

], style={'backgroundColor': 'white', 'padding': '15px', 'borderRadius': '10px', 'textAlign': 'center', 'boxShadow': '0 2px 5px rgba(0,0,0,0.1)'}),

html.Div([

html.H3("📅 Avg Daily Sales", style={'margin': '0', 'color': '#444'}),

html.H2(f"{avg\_daily\_sales:,.0f}", style={'margin': '0', 'color': '#2ca02c'})

], style={'backgroundColor': 'white', 'padding': '15px', 'borderRadius': '10px', 'textAlign': 'center', 'boxShadow': '0 2px 5px rgba(0,0,0,0.1)'}),

html.Div([

html.H3("🏆 Best Month", style={'margin': '0', 'color': '#444'}),

html.H2(best\_month, style={'margin': '0', 'color': '#ff7f0e'})

], style={'backgroundColor': 'white', 'padding': '15px', 'borderRadius': '10px', 'textAlign': 'center', 'boxShadow': '0 2px 5px rgba(0,0,0,0.1)'}),

]),

html.Div(style={'display': 'flex', 'gap': '20px'}, children=[

html.Div(dcc.Graph(figure=fig\_monthly), style={'flex': '1'}),

html.Div(dcc.Graph(figure=fig\_pie), style={'flex': '1'})

]),

html.Div(style={'display': 'flex', 'gap': '20px', 'marginTop': '20px'}, children=[

html.Div(dcc.Graph(figure=fig\_daily\_trend), style={'flex': '1'}),

html.Div(dcc.Graph(figure=fig\_monthly\_summary), style={'flex': '1'}),

]),

html.Div(style={'marginTop': '20px'}, children=[

html.Div(dcc.Graph(figure=fig\_scatter), style={'flex': '1'})

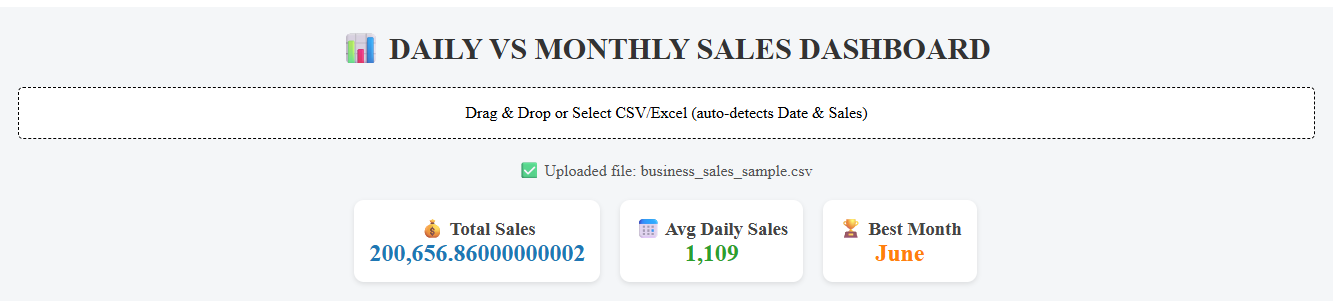
])

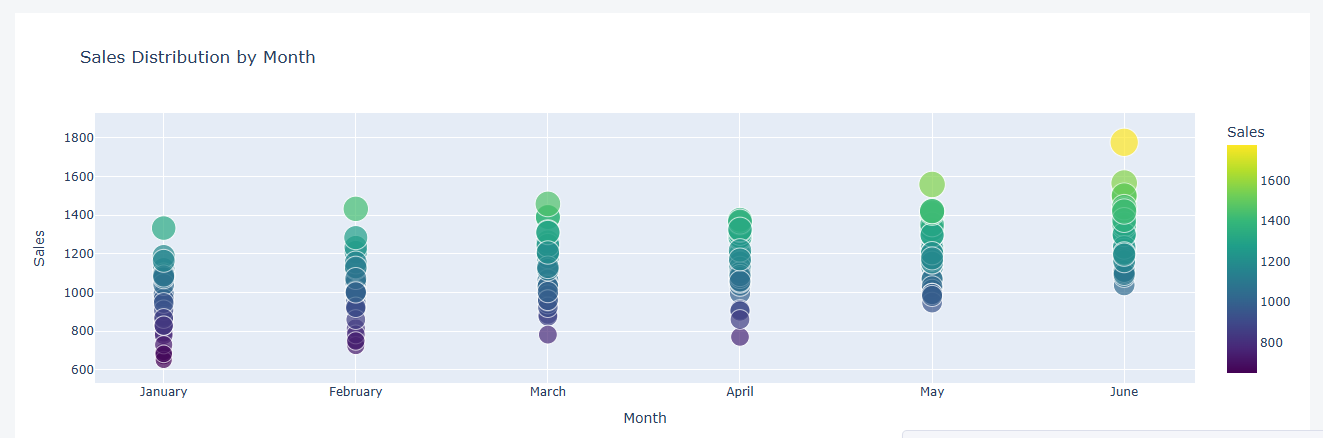
]) return f"✅ Uploaded file: {filename}", dashboard

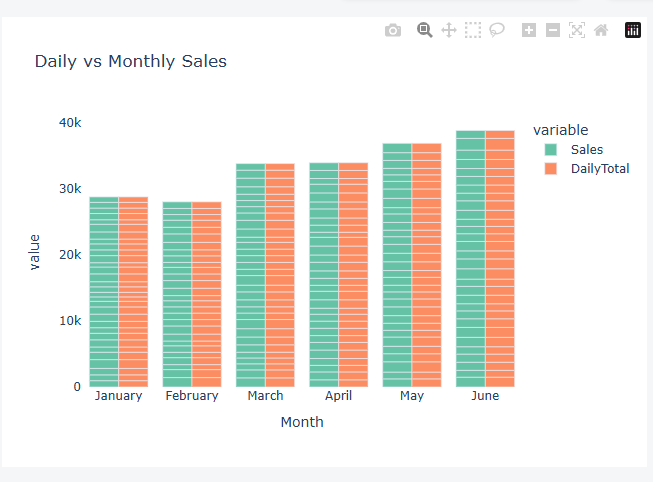
if \_\_name\_\_ == '\_\_main\_\_':

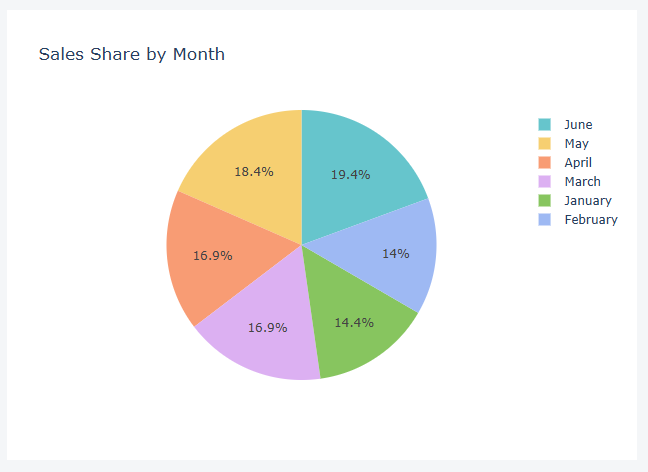
app.run(debug=True)

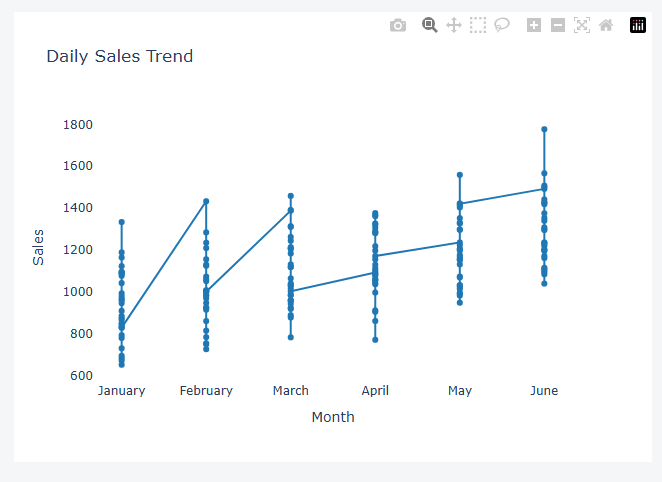
**Output:**

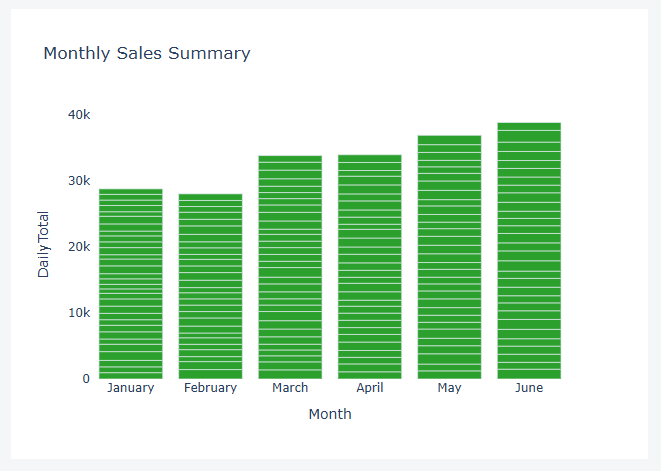
****

****

****

****

****

****

**Scope & Objectives**

**Scope:**  
This project provides an interactive business data visualization system capable of importing datasets, preprocessing data, and generating multiple visualization formats such as bar charts, pie charts, heatmaps, and time-series plots. It is designed to help users quickly understand patterns, trends, and anomalies in their business data.

**Objectives:**

1. To implement an easy-to-use tool for visualizing business datasets.
2. To support multiple visualization types for deeper insights.
3. To enable quick preprocessing of data, including handling missing values and encoding categorical data.
4. To present data in an interactive format for better decision-making.

**Applications**

* **Retail Sales Analysis** – Tracking product sales performance and seasonal trends.
* **Banking & Finance** – Visualizing loan approvals, rejections, and customer demographics.
* **E-commerce** – Monitoring order volumes, revenue, and customer purchase patterns.
* **Healthcare** – Analyzing patient data trends and medical service usage.
* **Education** – Tracking student performance metrics and enrollment trends.

## Conclusion

This project successfully demonstrates how business data can be processed, analyzed, and visualized to support informed decision-making. By leveraging Python libraries such as Pandas, Matplotlib, and Seaborn, along with optional interactive tools like Plotly and Streamlit , it transforms raw CSV data into meaningful insights through charts, graphs, and dashboards. These visualizations, including bar charts, pie charts, heatmaps, and time-series plots, help identify trends, patterns, and anomalies, making the solution valuable for large-scale businesses aiming to optimize performance and strategy.