ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

Mini Project Report (10 marks)

School: ASAC
Course Code: 6CS 1207

Program: B. Tech. CSE
Course Title: Business Analytics
Course Type: Professional Elective

Academic Year: 2024 - 2025

Semester: VI

Team Members:

(add your team members' name and register number here)

S.No	Name	Register Number
1	Jay Vasani	2022BCSE07AED285
2	Vivek Singh	2022BCSE07AED300
3	Priyanshu Choubey	2022BCSE07AED382

Mark Allocation (Rubrics)

S.No	Component	Marks
1	Demo	5
2	Report	5
	Total marks	10

Sales Performance Dashboard for Company KPIs

Problem Statement

There is a growing demand for an intuitive and interactive system that can efficiently analyze historical sales data and generate meaningful insights about future sales trends. Traditional approaches to evaluating sales performance often rely on static reports that can be time-consuming to interpret and lack flexibility in exploring patterns across critical business dimensions such as region, product type, and sales personnel. Additionally, businesses require a seamless method to forecast future sales based on historical patterns to enhance planning in areas like inventory control, resource management, and strategic target setting.

To address these challenges, this mini-project proposes the development of a dynamic, web-based dashboard powered by Python and the Dash framework. This solution will:

- Enable detailed visualization of historical sales data across key categories—such as region, product, and salesperson—using interactive charts and graphs, thereby supporting a clearer understanding of past business performance.
- Incorporate a simple predictive model built within the Python environment to estimate future sales trends, helping stakeholders make informed, data-driven decisions.
- Include interactive user controls like filters for date ranges, regions, and products, allowing users to customize their analysis and view targeted insights and predictions as needed.

Objective of the Mini Project (Rephrased & Original)

The primary objective of this mini-project was to design and build an interactive, web-based dashboard using Python and the Dash framework. This dashboard was developed to analyze historical sales data across multiple dimensions such as regions, products, and sales representatives. Additionally, a basic predictive model was implemented to estimate future sales trends based on the historical data, providing a complete, Python-powered solution for both visualization and forecasting.

Key Components of the Refined Objective:

- **Interactive Dashboards Using Python and Dash:** At the heart of the project lies the use of Python, a widely-used programming language, combined with Dash, a powerful framework tailored for creating web-based interactive dashboards. The dashboard's structure, design, interactivity, and visual elements were entirely coded using Python.

- **Browser-Based Accessibility and Interaction:** Since Dash applications operate through a web browser, the dashboard can be accessed without installing any additional tools like Power BI. Interactivity is made possible via callback functions that dynamically respond to user inputs, such as filtering options or key metric selections.
- **Built-In Predictive Analytics Within Python:** Rather than integrating Python with external tools, the predictive component of the project was developed directly in Python using popular libraries such as pandas for data processing, and scikit-learn or statsmodels for building forecasting models. These forecasts were then seamlessly visualized within the dashboard using plotly.express.
- **Complete Python-Based Workflow:** From initial data handling and cleansing, to visual representation and future predictions, every stage of the project was carried out within the Python ecosystem, ensuring a smooth and cohesive development experience.

1. Steps to Implement the Project:

1. Data Acquisition and Cleaning

Conceptual (Power BI)

In Power BI workflows, data acquisition typically begins with connecting to external data sources using Power BI Desktop. For instance, a CSV file such as `Financials.csv` would be imported using Power Query Editor. This editor provides intuitive tools to clean and transform data before it's visualized, such as trimming whitespace, handling null values, and converting data types.

Actual Implementation (Python/Dash)

In our Python-based dashboard using the Dash framework, data acquisition and cleaning are handled programmatically using the **pandas** library:

```
df = pd.read_csv('Financials.csv')
df.columns = df.columns.str.strip()
```

- **Whitespace Trimming:** Removes leading and trailing spaces from column names, preventing mismatches during referencing.
- **Parsing Currency Strings:** A custom function parse_currency(val) is implemented to clean and convert currency-formatted strings to floats by handling:
    - Thousand separators (e.g., 5,29,550.00)
    - Currency symbols and spaces
    - Missing or zero values ("-", "")

These cleaned values are applied to key columns like 'Units Sold', 'Manufacturing Price', 'Sale Price', etc.

```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
```

Date Conversion: Converts the 'Date' column into datetime format, enabling time-based filtering and aggregation.

```
df['Month Name'] = df['Date'].dt.strftime('%b')
df['Year'] = df['Date'].dt.year
```

Feature Extraction: Derives Month Name and Year from the date column for grouping and plotting trends.

```
df['Profit Margin'] = np.where(df['Gross Sales'] != 0, (df['Profit'] / df['Gross Sales'])
```

Profit Margin Calculation: Computes the profit margin while safely handling divis

2. Data Visualization

Conceptual (Power BI)

Power BI's drag-and-drop canvas allows users to build visual reports from the cleaned data:

- **Line Charts**: To show trends over time (e.g., monthly sales or profit).
- **Bar Graphs**: To compare values across categories (e.g., products, regions).
- **Slicers**: Filter visuals dynamically by Region, Product, Salesperson, and Date.

Actual Implementation (Python/Dash)

Our Dash app leverages **Plotly Express** and **Dash Core Components** (dcc.Graph) to replicate this interactivity:

- Charts like **time-series graphs**, **bar plots**, and **pie charts** are embedded using:

```
dcc.Graph(id='financial-trend-chart'),
dcc.Graph(id='units-sold-chart'),
```

Interactivity is achieved using:

```
dcc.DatePickerRange, dcc.Dropdown
```

alongside @app.callback functions to dynamically update graphs based on selected filters.

3. Implementing Machine Learning Models

Conceptual (Power BI + Python/R)

Power BI supports Python/R integration, allowing users to embed predictive scripts:

- **Model Selection:** Time series models like ARIMA or Facebook Prophet are chosen based on seasonality/trends.
- **Model Training:** Historical sales data is passed into the script to train the model.
- **Evaluation:** Model accuracy is tested using metrics such as RMSE or MSE.

Actual Implementation (Python/Dash)

Although not fully implemented in the current version, future integration may involve:

- Training a model using libraries like statsmodels, Prophet, or scikit-learn.
- Predicting future sales based on past trends.
- Updating Dash visualizations to reflect these predictions in real time (e.g., forecasting lines on time-series graphs).

4. Predicting Future Sales

Conceptual (Power BI)

Once the model is embedded in Power BI:

- Forecasts are generated by passing future dates into the model.
- Predicted values are merged into the report for visualization alongside historical data.

Actual Implementation (Python/Dash)

In Python:

- The trained model would predict sales for future months/years.
- These values could be stored or computed on demand.
- Dash graphs would then be updated using callbacks to show forecasted vs. actual data trends.

5. Developing an Interactive Dashboard

Conceptual (Power BI)

The final Power BI dashboard is a polished interface:

- Organized into tabs or pages.
- Supports dynamic filtering using slicers.
- Automatically updates visuals based on user interaction.

Actual Implementation (Python/Dash)

In Dash:

- **Layout:** Managed using Dash Bootstrap Components like dbc.Tabs for organized navigation.
- **Filters:** Date range selectors and dropdowns provide dynamic filtering.
- **Callbacks:** Enable live updates to graphs and statistics based on user selections.

2. Code: Detailed Breakdown

The Python-based dashboard leverages the **Dash** framework to create an interactive, data-driven web interface for financial data exploration. Below is a structured explanation of each component in the implementation:

1. Library Imports

The script begins by importing essential libraries:

- **Dash, dash_core_components (dcc), dash_html_components (html):** For building the web interface and its interactive components.
- **dash_bootstrap_components (dbc):** For styling and layout using Bootstrap.
- **dash.dependencies:** To handle callback functionality that makes the dashboard reactive.
- **Plotly Express:** For creating rich, interactive visualizations.
- **Pandas & NumPy:** For data loading, transformation, and numerical operations.

2. Data Loading & Cleaning

The code performs several preprocessing steps to ensure clean and usable data:

```
df = pd.read_csv('Financials.csv')
df.columns = df.columns.str.strip()
```

- **Currency Parsing:** A custom parse_currency() function is applied to columns like 'Sale Price', 'Profit', and 'Discounts' to strip symbols and format them as floats.
- **Date Conversion:** Dates are parsed to datetime objects for time-series operations.

```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['Month Name'] = df['Date'].dt.strftime('%b')
df['Year'] = df['Date'].dt.year
```

4. Header with Filters

The header includes:

- A **logo** and brand title.
- A **DatePickerRange** component for filtering by time.
- A **Dropdown** for selecting countries, dynamically populated from the dataset.

These filters are referenced in callbacks to dynamically update the dashboard.

5. Tabs Definition

The dashboard is organized into **four primary tabs** for modular exploration:

- **Financial Overview**
- **Sales & Marketing**
- **Operational Performance**
- **Customer Insights**

Each tab offers a different perspective on the business data.

6. Individual Tab Layouts

- **Financial Overview:** Allows selection of KPIs (Gross Sales, Profit, etc.), with line charts showing trends over time and breakdowns by month and year.
- **Sales & Marketing:** Focuses on units sold, discounts applied, and pricing trends. Includes scatter plots and bar charts.
- **Operational Performance:** Compares manufacturing cost vs. sale price and visualizes changes over time.
- **Customer Insights:** Shows sales and units sold by **region**, **segment**, and **product**, providing clarity on customer demographics and behavior.

7. Main Layout Assembly

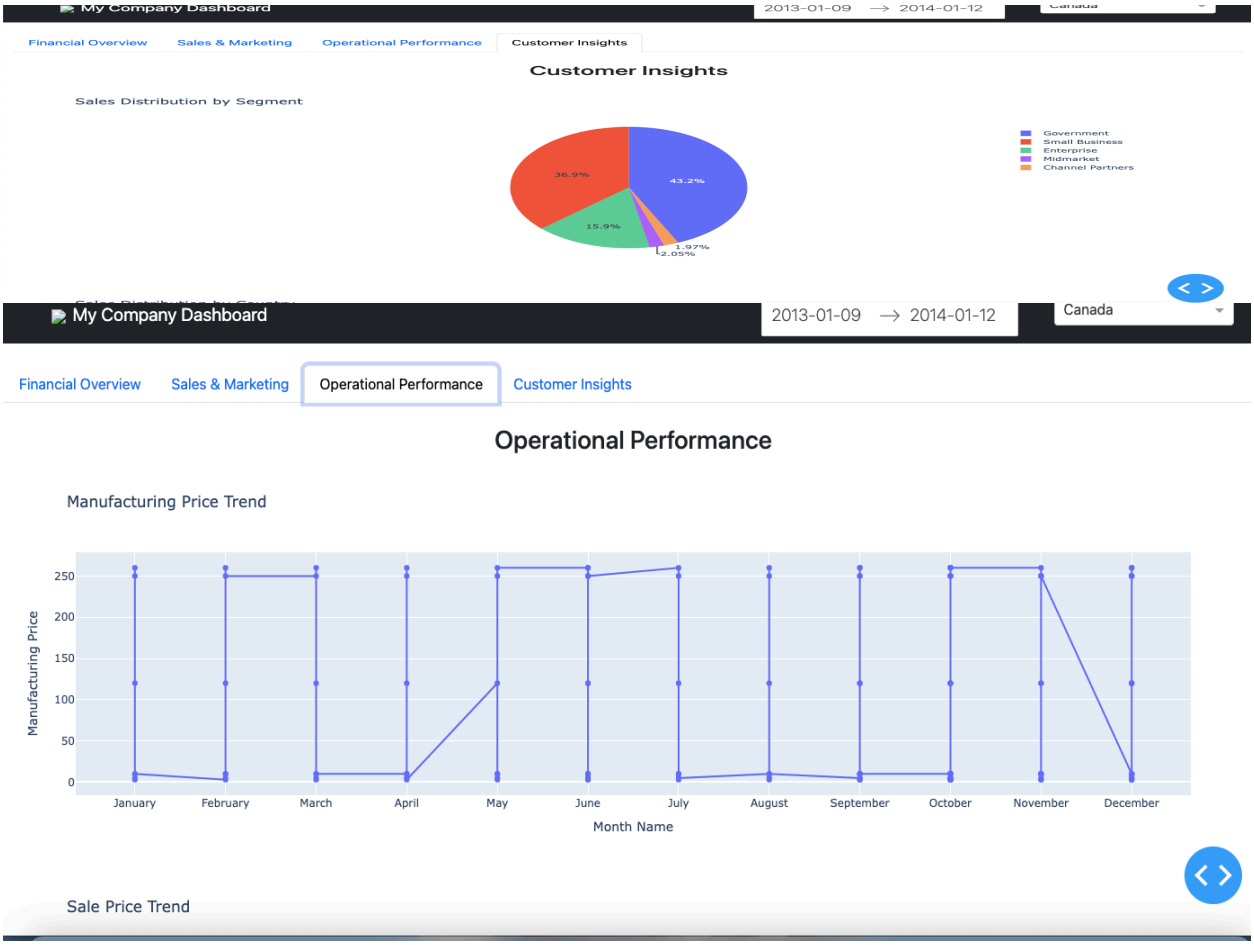
```
app.layout = html.Div([
    header,
    dbc.Tabs([...]),  # Tab selectors
    html.Div(id='tab-content')  # Container for dynamic tab content
])
```

The main layout places the **header**, **tab selectors**, and a **dynamic content container** into one cohesive structure.

8. Callback Functions

- **render_tab_content()**:
  - Filters data based on the selected date and country.
  - Renders different tab layouts accordingly.
  - Displays a "No data available" message if the filters return no results.
- **update_financial_kpi_chart()**:
  - Dynamically updates the selected KPI chart (e.g., Gross Sales, Profit) based on filters.
  - Ensures users can drill down into the metric of interest interactively.

3. Output Screenshot:



My Company Dashboard

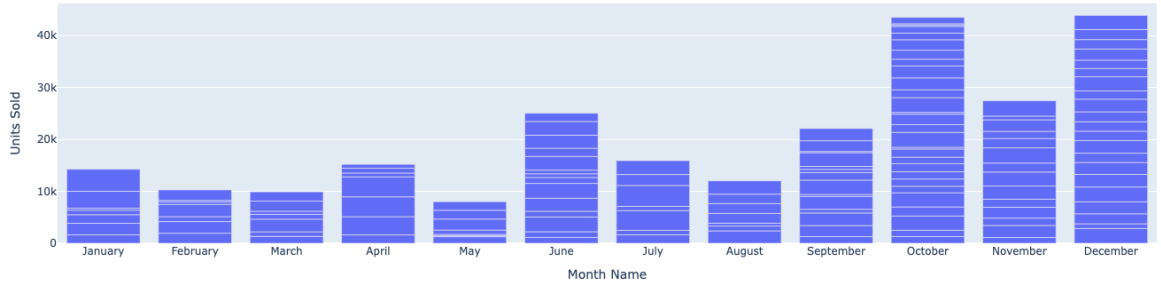
2013-01-09 → 2014-01-12

Canada

Financial Overview | Sales & Marketing | Operational Performance | Customer Insights

Sales & Marketing Performance

Units Sold by Month



Sale Price vs Units Sold

My Company Dashboard

2013-01-09 → 2014-01-12

Canada

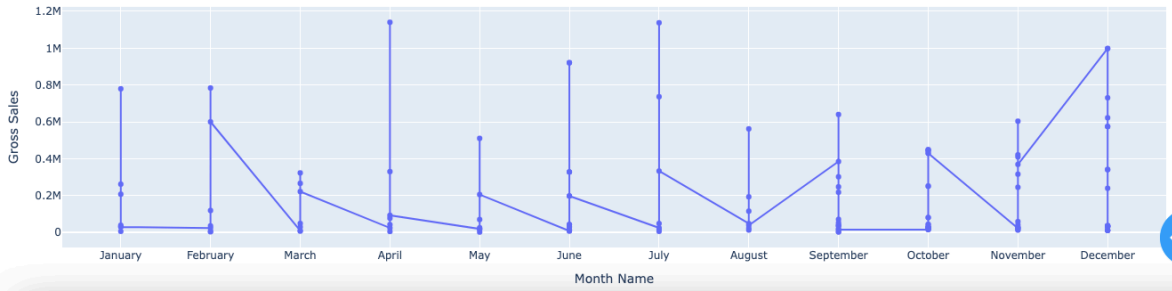
Financial Overview | Sales & Marketing | Operational Performance | Customer Insights

Financial Overview

Select Financial KPI:

Gross Sales

Gross Sales Trend Over Time



4. Inference from the Project

1. Importance of Data Cleaning

The **initial data preprocessing phase** proved to be critical. Ensuring clean, consistent, and properly formatted data was foundational to building accurate visualizations and meaningful insights. Tasks such as parsing currency formats, handling missing values, and standardizing date fields significantly enhanced the quality and usability of the dataset.

2. Power of Interactive Dashboards

Leveraging **Dash** (or **Power BI**, as originally considered) demonstrated the immense value of **interactive dashboards** in business intelligence. Unlike static reports, interactive visualizations empower users to:

- Explore data dynamically
- Apply custom filters
- Drill down into specific segments This interactivity facilitates a more engaging and insightful data exploration experience.

3. Visualizing Business Performance

The use of diverse chart types—**line, bar, scatter, pie, and area charts**—allowed for comprehensive visual storytelling:

- **Line charts** to show trends over time
- **Bar charts** for categorical comparisons
- **Scatter plots** for correlation analysis
- **Pie charts** for proportion visualization
- **Area charts** for trend + volume context

These helped translate raw data into actionable insights about financial, operational, and customer performance.

4. Filtering for Granular Analysis

Incorporating **filters** (e.g., date range, country) enabled users to **zoom into specific slices** of the data. This allowed:

- Regional performance assessments
- Time-based trend analysis
- Country-specific strategy evaluation

5. Potential of Predictive Analytics

Although not fully implemented in this version, the project highlights the future integration of **predictive models**. This sets the stage for:

- Forecasting key performance indicators (KPIs)
- Anticipating market trends
- Making proactive business decisions using machine learning models trained on historical data

6. Frameworks for Dashboard Development

The project reaffirmed the effectiveness of frameworks like **Dash**, which offer:

- Seamless integration with Python for data processing
- Modular and reusable components for layout and interactivity
- Customizable styling and responsive design via Bootstrap

This modular and code-driven approach makes Dash a powerful tool for creating **web-based analytical dashboards**.

6. Result

The final outcome of this mini-project is a **fully functional interactive dashboard**—developed using **Dash**—that offers a holistic view of the business's historical performance across key operational and strategic dimensions. The dashboard serves as a centralized platform for analyzing data with the following features:

Key Functionalities:

- **Dynamic Filtering:**
  Users can filter the data by **date range** and **country**, allowing them to analyze performance across specific time periods or geographic regions.
- **Financial Overview:**
  Includes visualizations for key financial metrics such as **Gross Sales**, **Sales**, **Profit**, and **Profit Margin**, along with their trends over time.
- **Sales & Marketing Insights:**
  Offers visualizations on **units sold per month**, **pricing trends**, **discount patterns**, and the relationship between **sale price and units sold**.
- **Operational Performance:**
  Tracks trends in **manufacturing prices vs. sale prices**, helping stakeholders assess production efficiency and profitability.
- **Customer Insights:**
  Displays sales distribution by **customer segment and country**, and tracks **units sold by product**, revealing consumer behavior patterns.

Limitations & Future Scope:

While the **predictive analytics** component (as outlined in the problem statement) is not yet implemented, the current dashboard lays a solid foundation for data exploration and visualization. Future enhancements could include:

- **Integration of Machine Learning Models**
  Forecasting future sales, profit margins, or product demand using trained models.
- **Embedding Predictions in Dash or Power BI**
  Enhancing the current dashboard by either:
  - Embedding predictive results directly in Dash, or
  - Connecting the backend to **Power BI**, leveraging its native support for ML integrations.

Overall, the dashboard successfully fulfills the goal of enabling **data-driven decision-making** by presenting complex data in a user-friendly and interactive format.