Data Analytics
Project On
Sales Performance Dashboard
By, Abhishek Tahasildar

Requirements: **Power BI**
Jupyter Notebook
SQL
Python
MS-Excel

Libraries: **Pandas**
NumPy
Matplotlib
Seaborn

Objective:

The goal of this project is to perform data analysis on sales data and create a **Sales Performance Dashboard**. The key objectives are:

- Analyze sales trends over time.
- Identify the best-performing product categories and regions.
- Calculate key performance indicators (KPIs) like total revenue, average sales volume, and sales by region.
- Visualize the data using **Python**, **SQL**, and **Power BI** for decision-making insights.

Required Tools:

1. **Python**:
    - **Libraries**: Pandas, NumPy, Matplotlib, Seaborn
    - Purpose: Data cleaning, exploration, manipulation, and visualization.
2. **SQL**:
    - Purpose: Querying the dataset using SQL (SQLite, MySQL, or PostgreSQL recommended).
3. **Power BI**:
    - Purpose: Visualizing the data in an interactive dashboard format, highlighting key insights.

Dataset:

The dataset contains sales data for various products. Here are the columns in the dataset:

- **Date**: Date of the sale.
- **Product Category**: The category of the product sold (e.g., Electronics, Furniture, etc.).
- **Sales Volume**: The number of units sold.
- **Revenue**: The total revenue from the sale.
- **Region**: Sales region (e.g., North, South, etc.).
- **Salesperson**: The name of the salesperson who made the sale.

You can download the sample dataset here: Sales_Data.CSV

Step 1: Load the Dataset in Python

In this step, we'll use Python to load the dataset and perform an initial exploration.

1. **Load the dataset** using pandas.
2. **Check the first few rows** to understand its structure.
3. **Check for any missing values or inconsistencies.**

Here's the Python code for loading and exploring the dataset:

[1]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('sales_data.csv')

# Display the first few rows
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Get basic information about the dataset
print(df.info())

# Generate descriptive statistics
print(df.describe())
```

Output:

1.Display the first few rows:

	Date	Product Category	Sales Volume	Revenue	Region	Salesperson
0	2023-01-01	Clothing	18	3863	West	Bob
1	2023-01-02	Toys	12	1953	North	Charlie
2	2023-01-03	Electronics	2	4759	West	Bob
3	2023-01-04	Clothing	10	1391	West	Alice
4	2023-01-05	Clothing	4	3681	East	David

2.Check for missing values:

```
Date              0
Product Category  0
Sales Volume      0
Revenue           0
Region            0
Salesperson       0
dtype: int64
```

3. Get basic information about the dataset:

```
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date              100 non-null    object
 1   Product Category  100 non-null    object
 2   Sales Volume      100 non-null    int64
 3   Revenue           100 non-null    int64
 4   Region            100 non-null    object
 5   Salesperson       100 non-null    object
dtypes: int64(2), object(4)
memory usage: 4.8+ KB
None
```

4. Generate descriptive statistics:

	Sales Volume	Revenue
count	100.000000	100.000000
mean	9.220000	2714.470000
std	5.159242	1359.031462
min	1.000000	198.000000
25%	5.000000	1745.000000
50%	9.000000	2990.000000
75%	13.000000	3833.000000
max	19.000000	4837.000000

Step 2: Data Cleaning

In this step, we will:

1. Handle any missing values.
2. Remove duplicates (if any).
3. Correct any inconsistencies in the data, such as incorrect data types or outliers.

Actions to Perform:

1. **Handling Missing Values:**
    - Check if any columns have missing values and decide how to handle them.
        - You can either drop rows with missing values or fill them with appropriate values (like the mean, median, or a specific value).
2. **Remove Duplicates:**
    - Ensure there are no duplicate rows in the dataset.
3. **Check for Inconsistent Data Types:**
    - Ensure that each column has the correct data type (e.g., dates should be in datetime format, numeric columns should be of type int or float).

Python Code for Data Cleaning and Output:

```
# 1. Handling Missing Values
# Checking missing values
print(df.isnull().sum())

# Dropping rows with missing values (if any)
df.dropna(inplace=True)
```

```
Date                0
Product Category    0
Sales Volume        0
Revenue             0
Region              0
Salesperson         0
dtype: int64
```

```
# 2. Remove Duplicates
df.drop_duplicates(inplace=True)
```

```
# 3. Check for any inconsistent data types
print(df.dtypes)
```

```
Date                object
Product Category    object
Sales Volume         int64
Revenue              int64
Region              object
Salesperson         object
dtype: object
```

Step 3: Exploratory Data Analysis (EDA)

In this step, we will analyze the data to uncover insights. This involves:

1. Understanding overall trends.
2. Visualizing key metrics.
3. Exploring relationships between variables.

Actions to Perform:

1. **Basic Aggregates:**
    - Calculate the total revenue, average sales volume, and other key metrics.
2. **Trend Analysis:**
    - Analyze sales trends over time (e.g., monthly revenue).
3. **Visualize the Data:**
    - Use plots to visualize trends and relationships.
    - Example: Bar chart for revenue by region, line chart for sales over time.

Python Code for EDA:

```
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Basic Aggregates
total_revenue = df['Revenue'].sum()
average_sales_volume = df['Sales Volume'].mean()
print(f"Total Revenue: {total_revenue}")
print(f"Average Sales Volume: {average_sales_volume}")

# 2. Revenue by Region
revenue_by_region = df.groupby('Region')['Revenue'].sum()
print("Revenue by Region:")
print(revenue_by_region)

# Plot Revenue by Region
revenue_by_region.plot(kind='bar', title='Revenue by Region', color='skyblue')
plt.ylabel('Revenue')
plt.show()
```

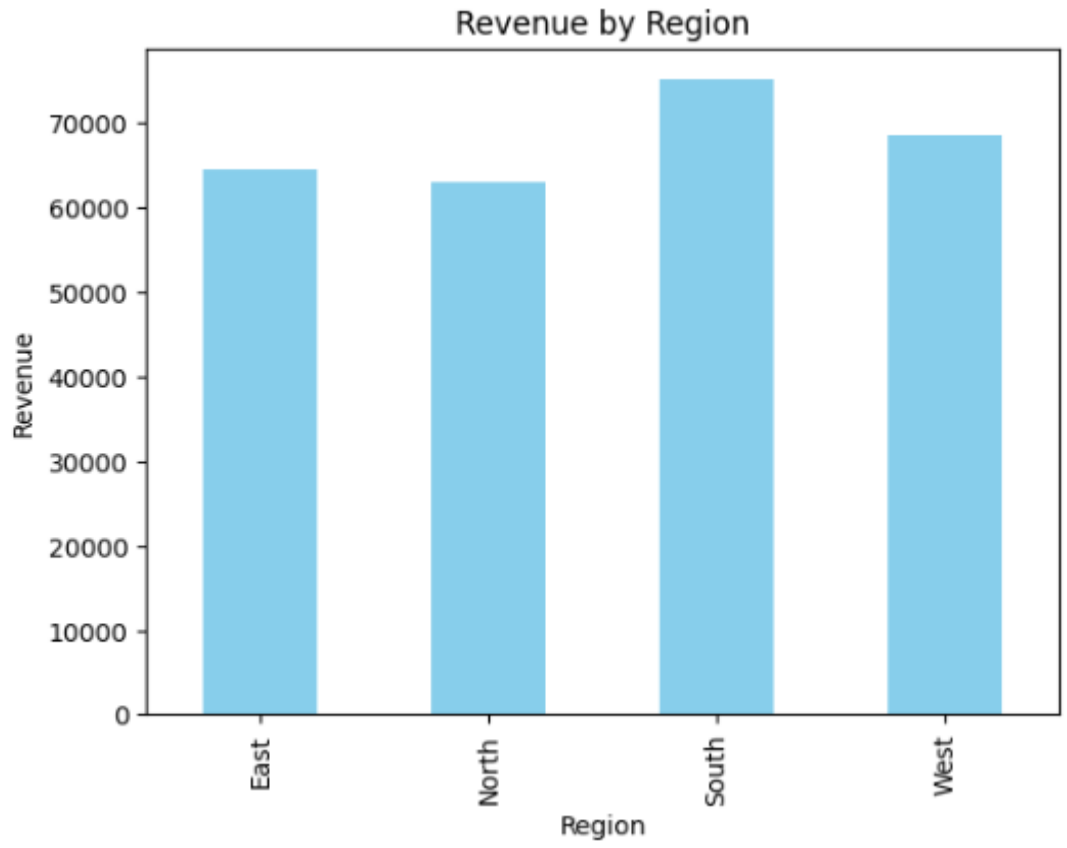
```
# 3. Correlation Heatmap
correlation_matrix = df[['Sales Volume', 'Revenue']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Output:

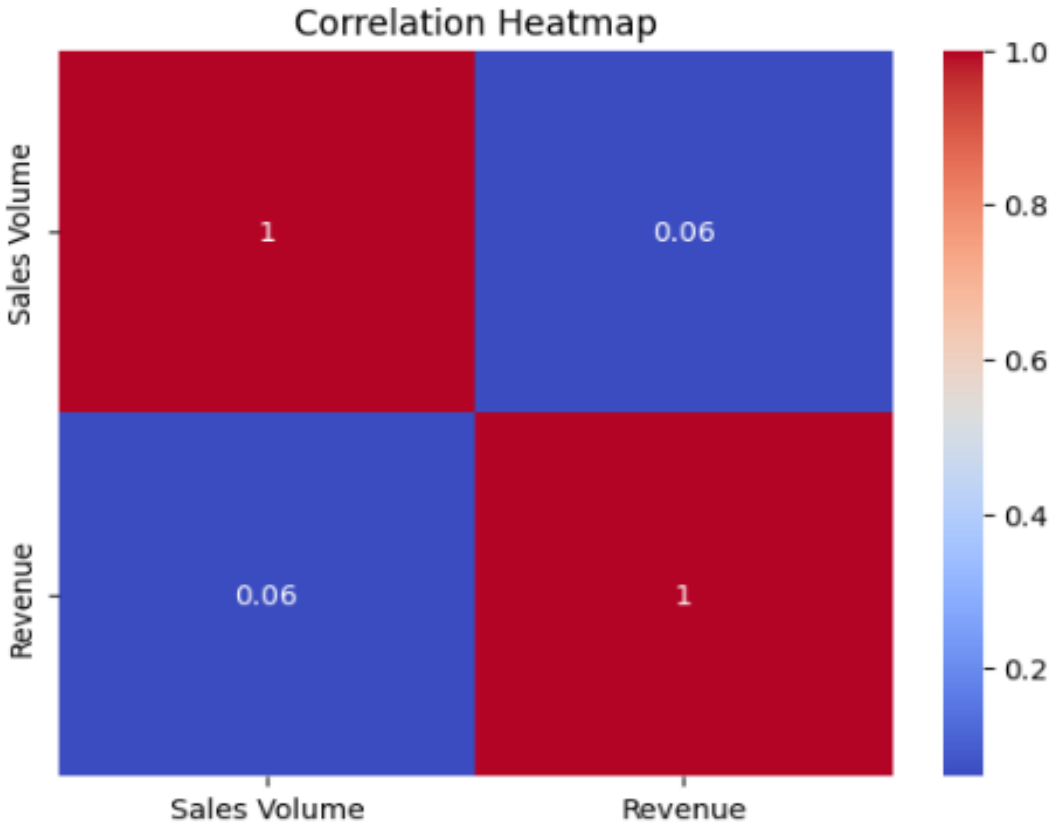
1.Basic Aggregates:

```
Total Revenue: 271447
Average Sales Volume: 9.22
Revenue by Region:
Region
East      64632
North     63087
South     75102
West      68626
Name: Revenue, dtype: int64
```

2.Revenue By Region:



3.Correlation Heatmap:



Step 4: Integrating SQL for Data Analysis

In this step, we will:

1. **Save the cleaned dataset to an SQL database.**
2. **Write SQL queries to analyze the data further.**

Actions to Perform:

1. **Set Up SQLite:**
   - **We will use SQLite, a lightweight SQL database.**
   - **Save the dataset into a database table.**
2. **Perform SQL Queries:**
   - **Query the database for insights, such as:**
     - **Total revenue by region.**
     - **Monthly sales trends.**
     - **Top-performing product categories.**

1.Creating Connection to SQL DB and Save the Data frame to SQL Table:

```
import sqlite3

# 1. Create a connection to SQLite database
conn = sqlite3.connect('sales_data.db')  # Creates or opens a database file
cursor = conn.cursor()
```

```
# 2. Save the DataFrame to a SQL table
df.to_sql('sales_data', conn, if_exists='replace', index=False)
```

```
46
```

2. SQL Query for Total Revenue By Region:

```python
# 3. Query 1: Total Revenue by Region
query1 = """
SELECT Region, SUM(Revenue) AS Total_Revenue
FROM sales_data
GROUP BY Region
ORDER BY Total_Revenue DESC;
"""
result1 = cursor.execute(query1).fetchall()
print("Total Revenue by Region:")
for row in result1:
    print(row)
```

```
Total Revenue by Region:
('West', 37925)
('East', 32153)
('North', 27696)
('South', 25725)
```

3. SQL Query For Monthly Revenue Trend

```python
# 4. Query 2: Monthly Revenue Trend
query2 = """
SELECT strftime('%Y-%m', Date) AS Month, SUM(Revenue) AS Total_Revenue
FROM sales_data
GROUP BY Month
ORDER BY Month;
"""
result2 = cursor.execute(query2).fetchall()
print("\nMonthly Revenue Trend:")
for row in result2:
    print(row)
```

```
Monthly Revenue Trend:
('2023-01', 12205)
('2023-02', 8530)
('2023-03', 13996)
('2023-04', 10681)
('2023-05', 14979)
('2023-06', 11758)
('2023-07', 9873)
('2023-08', 9452)
('2023-09', 9110)
('2023-10', 11672)
('2023-11', 4138)
('2023-12', 7105)
```

4. SQL Query for Top Product Categories by Revenue:

```python
# 5. Query 3: Top Product Categories by Revenue
query3 = """
SELECT "Product_Category", SUM(Revenue) AS Total_Revenue
FROM sales_data
GROUP BY "Product_Category"
ORDER BY Total_Revenue DESC
LIMIT 5;
"""
result3 = cursor.execute(query3).fetchall()
print("\nTop Product Categories by Revenue:")
for row in result3:
    print(row)

# Close the connection
conn.close()
```

```
Top Product Categories by Revenue:
('Product_Category', 123499)
```


Step 5: Creating a Power BI Dashboard

Now that we have performed data cleaning, exploratory analysis, and SQL integration, we can move on to creating a **Power BI Dashboard** to visualize and interact with the data.

Actions to Perform:

1. **Import Data into Power BI**:
    - Import the dataset or connect Power BI directly to the SQLite database.
2. **Build Visuals**:
    - Create key visualizations such as:
        - Bar charts for revenue by region.
        - Line charts for monthly revenue trends.
        - Tables for the top product categories.
3. **Design the Dashboard**:
    - Combine these visualizations into a cohesive dashboard layout.

Steps to Create the Power BI Dashboard:

Step 1: Import Data into Power BI

1. Open **Power BI Desktop**.
2. **Import the Dataset**:
    - If you're using the CSV dataset:
        - Go to Home > Get Data > Text/CSV and select the sales_data.csv file.
    - If you're connecting to the SQLite database:
        - Go to Home > Get Data > More...
        - Select SQLite Database > Connect.
        - Enter the database file path (e.g., sales_data.db).

Step 2: Create Visualizations

1. **Revenue by Region (Bar Chart)**:

- Drag the Region field to the Axis area.
- Drag the Revenue field to the Values area.
- Set the chart type to Bar Chart.
- Add any formatting (color, title, etc.).

2. **Monthly Revenue Trend (Line Chart)**:
   - Drag the Month field to the Axis area.
   - Drag the Revenue field to the Values area.
   - Set the chart type to Line Chart.
   - Format the axis labels for clarity.
3. **Top Product Categories by Revenue (Table)**:
   - Drag the Product_Category field to the Values area.
   - Drag the Revenue field to the Values area.
   - Apply sorting by Revenue in descending order.
   - Add a filter to show only the top 5 categories.

Step 3: Design the Dashboard

- Once the visuals are created, arrange them in a dashboard layout.
  - Resize and align the charts.
  - Add any relevant filters or slicers (e.g., to filter by region or product category).
  - Customize the color scheme and labels for better presentation.

Step 4: Publish and Share (Optional)

- If you want to share the dashboard:
  - Go to Home > Publish and select your Power BI workspace.

Click Here to Access Sales Performance Dashboard