Topic: **Retail business sales performance & probability analysis**

Introduction:

- Retail performance analysis is the practice of examining and assessing a retail business's condition and efficiency. Retail performance analysis includes gathering information and data from multiple sources and then processing and evaluating it for various aspects of retail stores.

- On the other hand, retail KPI is a well-defined and specific metric used to evaluate a retail business and indicate whether it performs effectively. These indicators can be used to evaluate a particular retail store's performance in various aspects. For instance, customer loyalty, fulfillment of demands, sales rate, and inventory management are some examples.

- A performance analysis of the retail industry closely follows this process. It is the process of measuring different factors and indicators to attain the level of health, productivity, and efficacy of retail outlets.

The Impact of Effective Retail Performance Analysis on Business Growth

- Data-driven Decision Making
- Enhanced operational efficiency
- Customer-centric strategies
- Strategic growth planning
- Competitive advantage

Objectives:

Analyze transactional retail data to uncover profit-draining categories, optimize inventory turnover, and identify seasonal product behavior.

Abstract:

Probability analysis informs retail business performance by helping to quantify uncertainty in areas like sales forecasting, market trends, and investment decisions, enabling data-driven strategies for product placement, inventory management, and marketing campaigns. By applying probabilistic models to track Key Performance Indicators (KPIs) such as conversion rate, average transaction value, and customer retention, retailers can identify risks, optimize store layouts, and forecast future results with greater accuracy

How Probability Informs Retail Performance

- Forecasting & Planning
- Marketing & Customer Behavior
- Risk Management
- Investment Decisions

Sales Forecasting (Probability Distribution)

- Time Series Models (ARIMA, Prophet, LSTM)
- Probabilistic Forecasting: Instead of a single number, predict a confidence interval (e.g., “Next month’s sales are expected between $85K and $95K with 90% confidence”).

Inventory Stock-out Probability

- Calculate probability of stock running out based on historical demand patterns.

Formula:

- P(Stockout)=P(Demand>Inventory Level)

Tools used:

Pandas library:

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Examples

```
import pandas as pd

# Sample data
data = {'Category': ['A', 'B', 'A', 'C', 'B'],
        'Sales': [100, 200, 150, 300, 250]}

df = pd.DataFrame(data)

# Group by category and sum sales
result = df.groupby('Category')['Sales'].sum()
print(result)
```

Output:

```
Category
A    250
B    450
C    300
Name: Sales, dtype: int64
```

Data visualization:

Matplotlib.pyplot

- Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

- import matplotlib.pyplot as plt

- Now the Pyplot package can be referred to as plt.

Matplotlib is a powerful and versatile open-source plotting library for Python, designed to help users visualize data in a variety of formats. Developed by John D. Hunter in 2003, it enables users to graphically represent data, facilitating easier analysis and understanding

Examples:

```
import matplotlib.pyplot as plt


x = [0, 1, 2, 3, 4]
y = [0, 1, 4, 9, 16]


plt.plot(x, y)
plt.show()
```

Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs so that we can switch between different visual representations for the same variables for a better understanding of the dataset.

Installation of Seaborn Library

- For Python environment :

```
pip install seaborn
```

- For conda environment :

```
conda install seaborn
```

Examples:

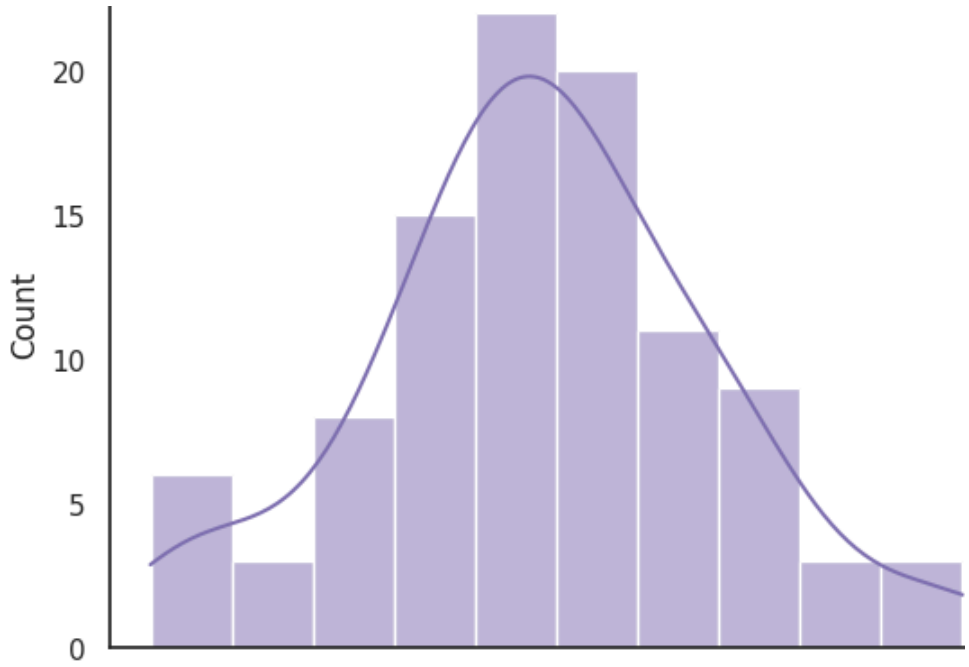
```
import numpy as np
import seaborn as sns

sns.set(style="white")

# Generate a random univariate dataset
rs = np.random.RandomState(10)
d = rs.normal(size=100)

# Plot a simple histogram and kde
sns.histplot(d, kde=True, color="m")
```

Output:



Steps involved:

Step 1: Import libraries

- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns

Why:

pandas: for loading and manipulating the dataset
matplotlib.pyplot & seaborn: for creating different visualizations

Step 2: Load data

- df = pd.read_csv("retail_sales_dataset.csv")
- df

Why: Reads the retail sales dataset into a DataFrame for analysis.

Step 3: Explore First Rows

- df.head(5)

Why:

- Quickly checks the first few records to understand the dataset structure.

Step 4: Dataset Information

- df.info()

Why: Provides details on columns, data types, and missing values.

Step 5: Statistical Summary

- df.describe()

Why: Summarizes numerical columns (mean, median, standard deviation, etc.) to detect ranges and outliers.

Step 6: Convert Date & Extract Month

- df['Date'] = pd.to_datetime(df['Date'])
- df['Month'] = df['Date'].dt.to_period('M')

Why: Prepares the dataset for time-series analysis (monthly sales trend).

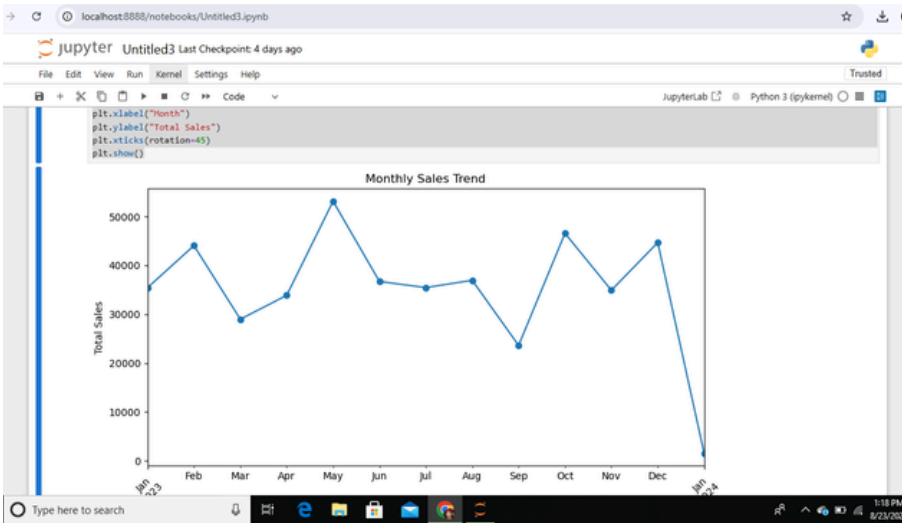
Data visualization:

Step 7: Monthly Sales Trend

- plt.figure(figsize=(10,5))
- monthly_sales = df.groupby('Month')['Total Amount'].sum()
- monthly_sales.plot(marker='o')
- plt.title("Monthly Sales Trend")
- plt.xlabel("Month")
- plt.ylabel("Total Sales")
- plt.xticks(rotation=45)
- plt.show()

Why: Identifies sales patterns over time (seasonality, growth, or decline).

Output:

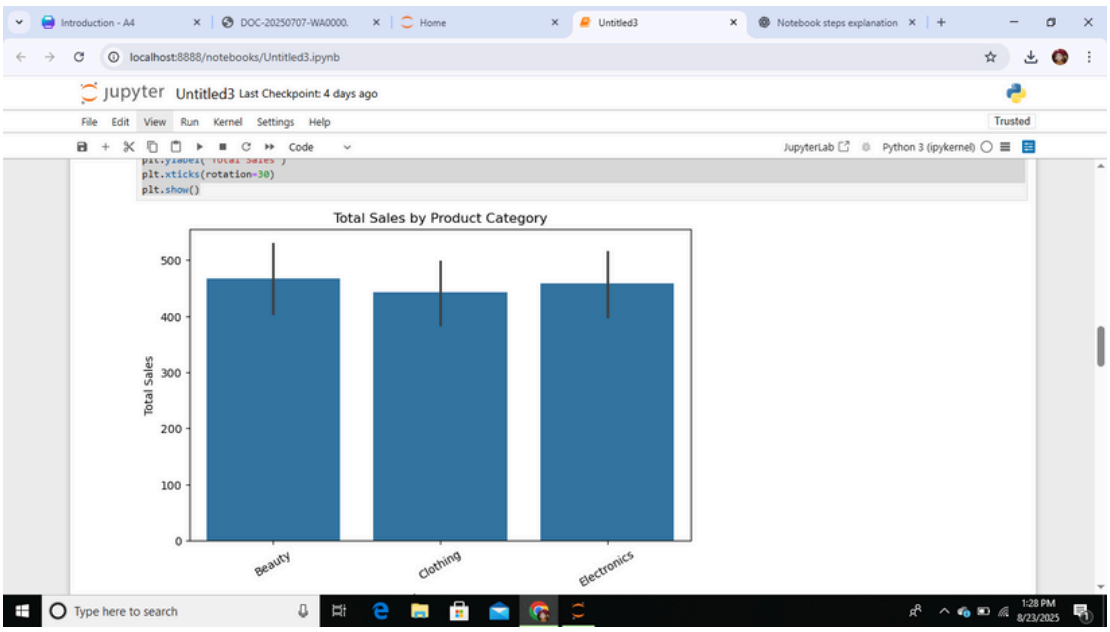


Step 8: Sales by Product Category

- plt.figure(figsize=(8,5))
- sns.barplot(x="Product Category", y="Total Amount", data=df)
- plt.title("Total Sales by Product Category")
- plt.ylabel("Total Sales")
- plt.xticks(rotation=30)
- plt.show()

Why: Shows which product categories generate the most revenue.

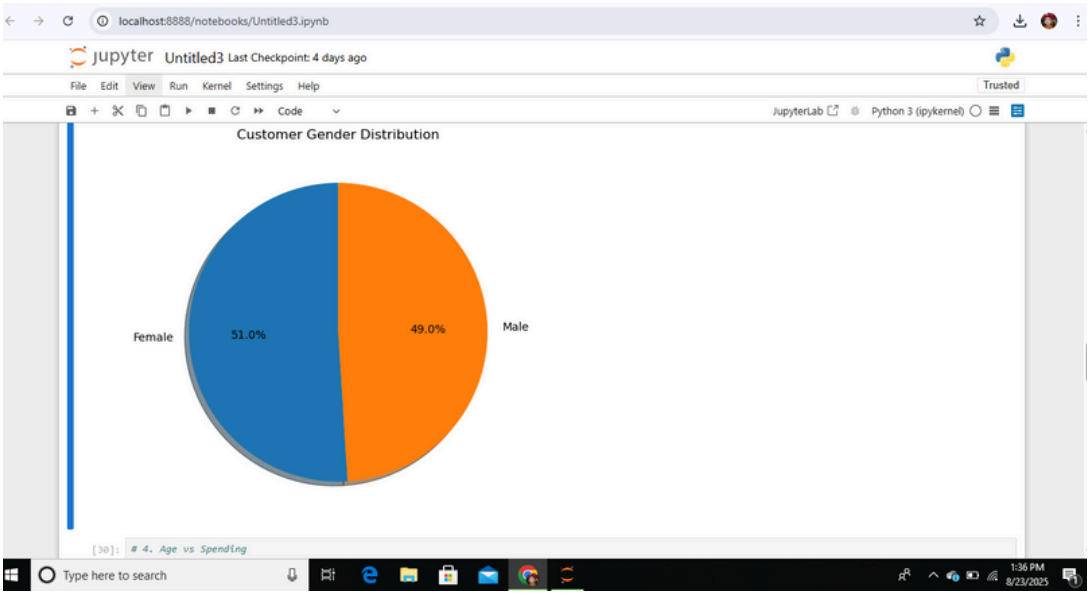
Output:



Step 9: Customer Gender Distribution

- plt.figure(figsize=(6,6))
- df['Gender'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, shadow=True)
- plt.title("Customer Gender Distribution")
- plt.ylabel("")
- plt.show()

Output:

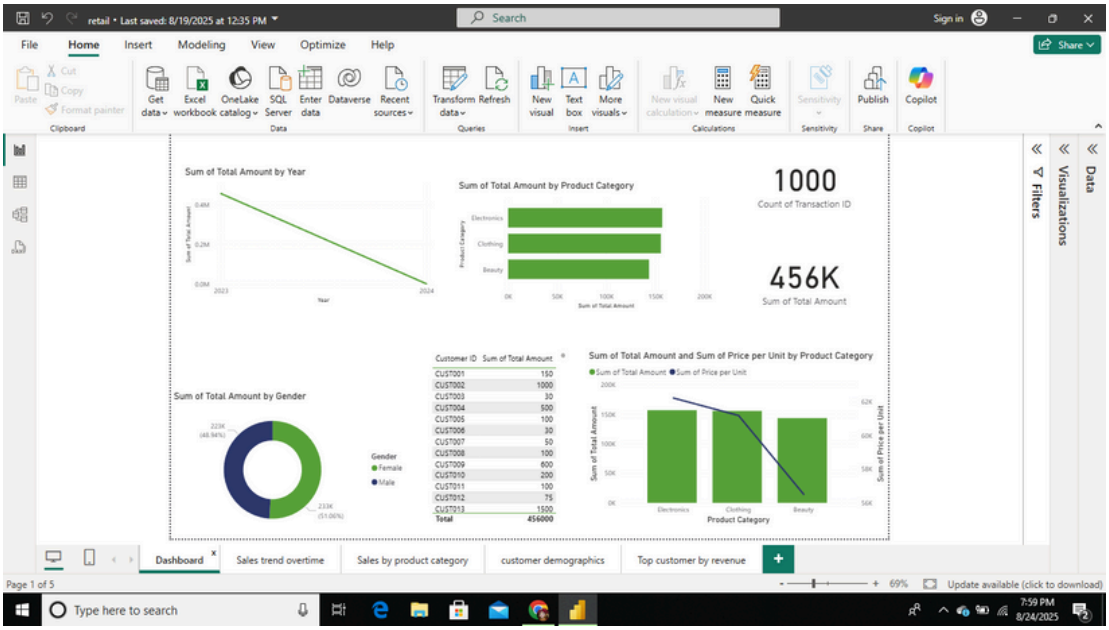


Step 10: Age vs Spending

- plt.figure(figsize=(8,5))
- sns.scatterplot(x="Age", y="Total Amount", hue="Gender", data=df, alpha=0.7)
- plt.title("Customer Age vs Spending")
- plt.xlabel("Age")
- plt.ylabel("Total Amount")
- plt.show()

Output:

Interactive Dashboard result



Thank you