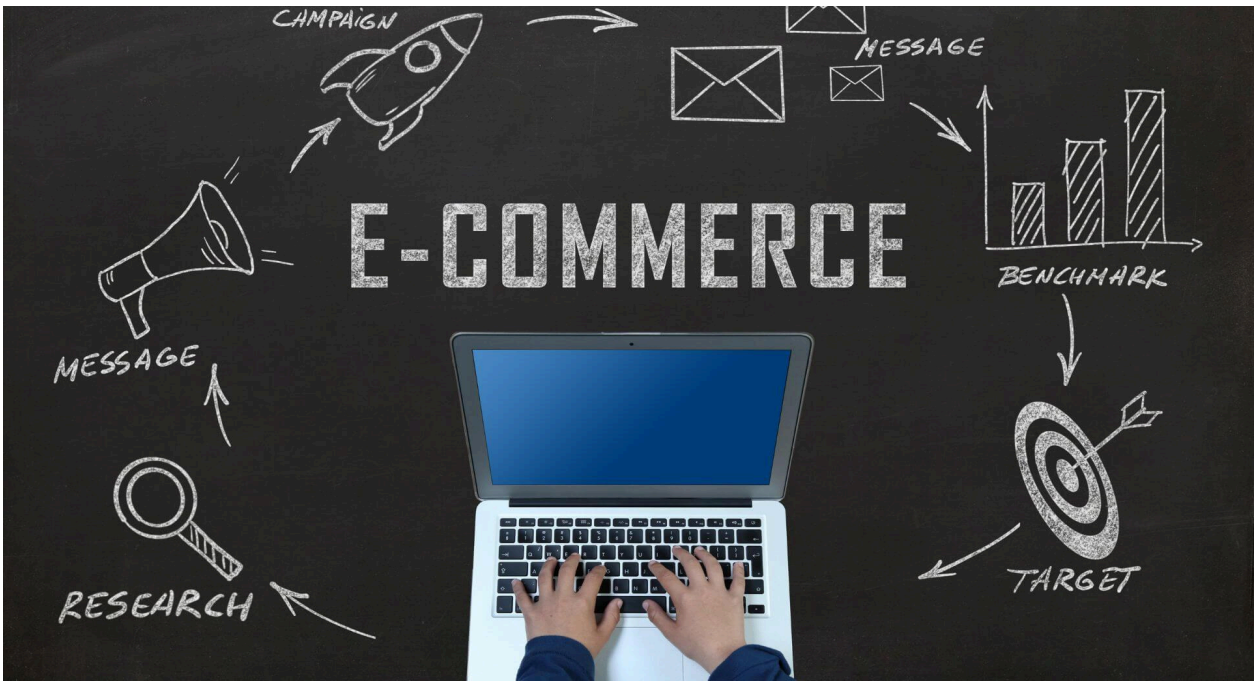


E-Commerce Project

3.02.2025

Sneha Jha

Link: ***Ecommerce_recoded.ipynb***

Project Overview

This project analyzes an E-commerce dataset to gain insights into sales, customer behavior, product performance, and shipping trends using Python. The goal is to clean, explore, visualize, and derive actionable insights to enhance business decision-making.

Objective:
Analyze an e-commerce dataset to uncover trends in sales, customer behavior, product performance, and delivery efficiency.
Dataset:

- Contains fields like order_date, customer_region, category_name, sales_per_order, profit_per_order, etc.
- Focuses on transactional data, shipping logistics, and customer demographics.

Business Relevance:
Helps businesses optimize inventory, improve delivery processes, and target high-value customer segments.

Project Goals

- Deliver actionable insights to refine pricing, inventory, and logistics strategies.Analyze sales trends and profitability to identify top-performing products, peak periods, and profit margins.
- Segment customers by purchase behavior and demographics for targeted marketing.
- Optimize shipping efficiency by evaluating delivery times, costs, and delays.
- Detects outliers and trends through statistical analysis and interactive visualizations.
- Deliver actionable insights to refine pricing, inventory, and logistics strategies.

Tools & Specification

Tools Used:
- Python Libraries: Pandas, NumPy, Matplotlib, Seaborn, Plotly.
- Dataset: Ecommerce_data.csv (cleaned to Cleaned_Ecommerce_data.csv).

Key Metrics Analyzed:
- Sales, profit, order quantity, delivery status.
- Customer demographics (region, city, segment).
- Shipping performance.

Methodology:
- Data Cleaning: Remove missing values, handle duplicates, and standardize formats.

- **Exploratory Analysis:** Use descriptive statistics and correlation analysis.
- **Visualization:** Create meaningful charts to uncover trends and insights.

Analysis & Insights

1.Data Cleaning - Handling Missing Values

```
data_cleaned = data.dropna()
```

Explanation:

"I used data.dropna() to remove rows with missing values, ensuring a clean dataset for accurate analysis."

Output : *"Now, the dataset is free of null values, improving data integrity and reliability."*

2.Saving the Cleaned Dataset

```
data_cleaned.to_csv("Cleaned_Ecommerce_data.csv",
index=False)
```

Explanation: Saves the cleaned data as a CSV file for easy access and future analysis.
Output: A refined dataset stored for further use.

3.Checking Shipping Delays

```
data['shipping_delay'] = data['days_for_shipment_real'] -
data['days_for_shipment_scheduled']
# Check if delays exist
```

```
print(data[['days_for_shipment_scheduled', 'days_for_shipment_real',
'shipping_delay']].head(10))
```

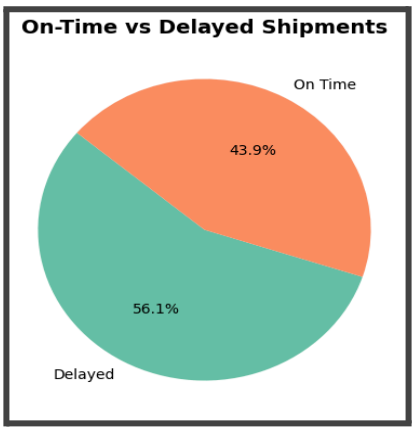
Explanation: Calculates shipping delays by subtracting the scheduled shipment days from the actual shipment days.
A positive value indicates a delay, while zero or negative means on-time delivery.

It helps businesses:

- *Monitor delivery performance* to track how often shipments are delayed.
- *Identify logistics inefficiencies a*nd work with carriers to improve shipping timelines.
- *Enhance customer satisfaction* by reducing delivery delays and improving service levels.
- *Optimize supply chain strategies* by analyzing patterns in late shipments.

Output: Displays a table with scheduled shipment days, actual shipment days, and calculated delays for the first 10 orders.

	days_for_shipment_scheduled	days_for_shipment_real	shipping_delay
0	2.0	2.0	0.0
1	2.0	3.0	1.0
2	4.0	5.0	1.0
3	2.0	4.0	2.0
4	1.0	2.0	1.0
5	4.0	4.0	0.0
6	4.0	3.0	-1.0
7	4.0	4.0	0.0
8	4.0	4.0	0.0
9	4.0	6.0	2.0



4.Top 5 customer by sales

```
# Group sales by customer and sort in descending order
top_customers =
data.groupby('customer_id')['sales_per_order'].sum().sort_values(ascend
ing=False).head(5)
print("Top 5 Customers by Sales:\n", top_customers)
```

Explanation: Groups data by customer_id and sums the sales_per_order to calculate total sales per customer.
Sorts the customers in descending order based on total sales and selects the top 5 highest-spending customers.

It helps businesses:

- Identify high-value customers for personalized marketing and loyalty programs.
- Optimize sales strategies by focusing on repeat or high-spending buyers.
- Improve customer retention by offering exclusive deals to top customers.
- Forecast revenue trends by analyzing purchasing behavior of top buyers.

5.Top products generating the most revenue

```
top_products =
data.groupby('product_name')['sales_per_order'].sum().sort_values(ascen
ding=False).head(10)
top_products_df = pd.DataFrame({
    'Product Name': top_products.index,
    'Total Sales': top_products.values})
top_products_df
```

Explanation: Groups data by product name and sums total sales to identify the highest-grossing products.

Sorts them in descending order and selects the top 10.

It helps businesses:

- Focus on best-sellers.
- Optimize inventory management.
- Plan targeted promotions or marketing efforts.

Output: Displays the top 10 products based on total sales in a structured DataFrame.

	Product Name	Total Sales
0	Staple envelope	22072.620408
1	Staples	19614.364288
2	Easy-staple paper	17681.284324
3	Nu-Dell Executive Frame	8929.090114
4	Avery Non-Stick Binders	8224.268172
5	KI Adjustable-Height Table	7950.578135
6	Staple-based wall hangings	6959.338089
7	Situations Contoured Folding Chairs, 4/Set	6905.798098
8	Storex Dura Pro Binders	6801.510114
9	SanDisk Ultra 32 GB MicroSDHC Class 10 Memory ...	6744.638134

6.Average profit by shipping type

```
avg_profit_by_shipping =
data.groupby('shipping_type')['profit_per_order'].mean()

plt.figure(figsize=(8, 4))

avg_profit_by_shipping.plot(kind='bar', color='teal')

plt.title("Average Profit by Shipping Type")

plt.xlabel("Shipping Type")

plt.ylabel("Average Profit")

plt.xticks(rotation=45)

plt.show()
```

Explanation:

The code calculates the average profit per order for each shipping type, grouping data by shipping_type. It then visualizes the results in a bar chart.

How it helps in business:

- Identifies which shipping types are most profitable.
- Helps optimize shipping strategies and cost management.
- Informs decisions on offering promotions or adjusting shipping options for better profitability.

Output:

7.Region-wise sales performance

```
import matplotlib.pyplot as plt

# Group and sort sales data

region_sales =
data.groupby('customer_region')['sales_per_order'].sum().sort_values(as
cending=False)

# Plot the pie chart

plt.figure(figsize=(5, 5))

plt.pie(region_sales, labels=region_sales.index, autopct="%1.1f%%",
colors=sns.color_palette("coolwarm", len(region_sales)),
startangle=140)

plt.title("Sales Distribution by Region", fontsize=14,
fontweight='bold')

plt.show()
```

Explanation:

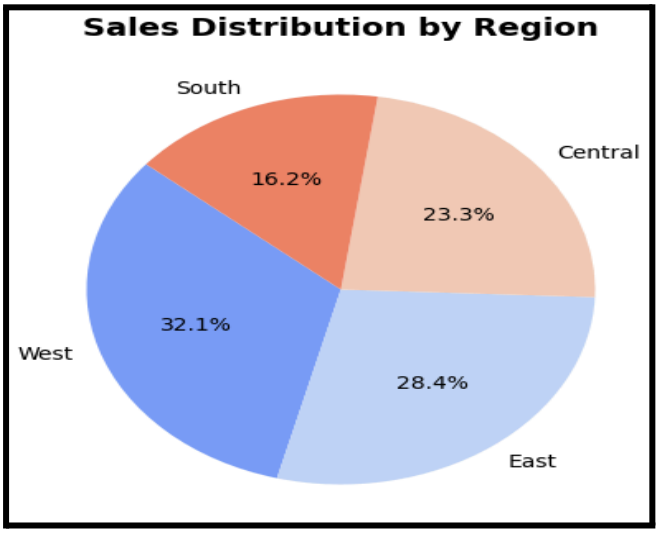
Groups data by customer_region and sums the sales_per_order to calculate total sales for each region.

Sorts the regions in descending order to identify top-performing markets.
Visualizes the sales distribution across regions using a pie chart.

How it helps in business:

- Identifies the regions with the highest sales, helping to target key markets.
- Optimizes marketing and sales strategies for top-performing regions.
- Assists in inventory planning and resource allocation based on regional demand.

Output:A pie chart displaying sales distribution across different regions.Regions with the highest sales are represented with larger slices, making it easy to compare performance.



8.Monthly Sales Trend

```
monthly_sales = data.groupby('month_year')['sales_per_order'].sum()
# Plot sales trend over time
plt.figure(figsize=(10, 5))
monthly_sales.plot(kind='line', marker='o', color='b')
plt.title("Monthly Sales Trend")
plt.xlabel("Month-Year")
plt.ylabel("Total Sales")
```

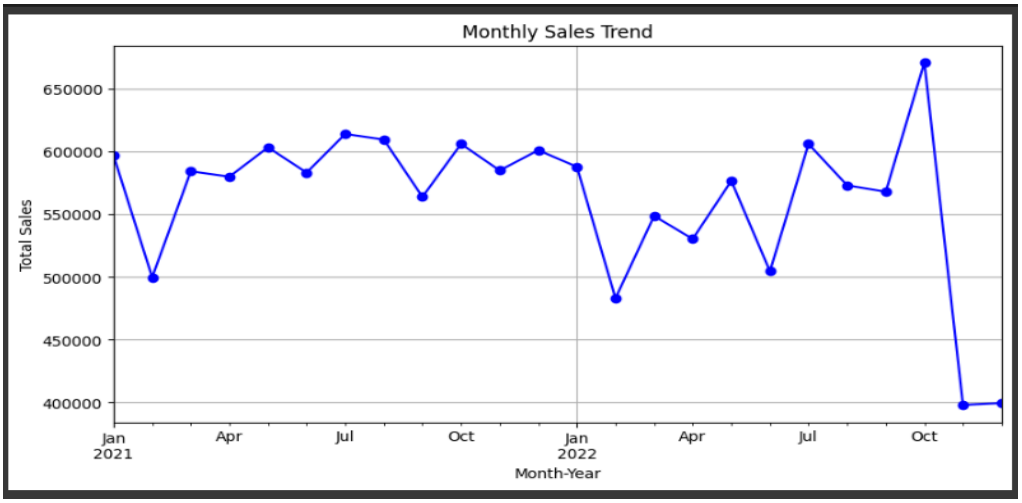
```
plt.grid()

plt.show()
```

Explanation: Groups data by month_year and sums the sales_per_order to calculate total monthly sales.
Plots a line graph to visualize sales trends over time, identifying peaks and dips.

It helps businesses:

- Identify seasonal trends to optimize inventory and marketing campaigns.
- Forecast future sales based on past trends for better financial planning.
- Evaluate business performance by tracking revenue growth over time.
- Improve demand planning by adjusting stock levels based on high and low sales periods.

Output:A line graph displaying the monthly sales trend, highlighting fluctuations and patterns over time.
Would you like to add trendlines or moving averages for deeper insights?



9.Top 10 cities and their sales in Rs

```
top_cities =
data.groupby('customer_city')['sales_per_order'].sum().sort_values(asce
nding=False).head(10)

top_cities_df = pd.DataFrame({

    'City': top_cities.index,
```

```
    'Total Sales': top_cities.values})

top_cities_df
```

Explanation:

Groups data by customer_city and sums sales_per_order to calculate total sales for each city. Sorts the cities by total sales in descending order and selects the top 10.

How it helps in business:

- Identifies cities with the highest revenue, aiding in targeted sales strategies.
- Helps allocate marketing budgets effectively to high-performing cities.
- Supports inventory planning based on demand in top cities.
- Assists in expansion decisions by identifying potential high-growth areas.

Output: Displays a structured DataFrame showing the top 10 cities and their total sales in rupees, ranked from highest to lowest.

	City	Total Sales
0	New York City	348370.304340
1	Los Angeles	303490.266803
2	Philadelphia	222108.965859
3	San Francisco	206803.389634
4	Seattle	169708.128644
5	Houston	157639.092547
6	Chicago	123750.104200
7	Columbus	85702.177577
8	San Diego	68963.849063
9	Springfield	66349.853376

10.Delivery statuses and their respective counts.

```
delivery_status_count = data['delivery_status'].value_counts()
```

```
delivery_status_df = pd.DataFrame({

    'Delivery Status': delivery_status_count.index,

    'Count': delivery_status_count.values})

delivery_status_df
```

Explanation:

Counts occurrences of each delivery_status category in the dataset. Creates a structured DataFrame displaying the count of each delivery status type.

How it helps in business:

- Monitors delivery performance to ensure timely order fulfillment.
- Identifies issues like delays or cancellations for process improvements.
- Enhances customer satisfaction by addressing delivery bottlenecks.
- Helps optimize logistics and courier partnerships for better efficiency.

Output: Displays a structured DataFrame showing different delivery statuses and their respective counts.

	Delivery Status	Count
0	Advance shipping	9451
1	Shipping on time	5933
2	Late delivery	1576
3	Shipping canceled	1563

Milestones

- **Data Cleaning:** Removed missing values and ensured data consistency.
- **Exploratory Data Analysis (EDA):** Examined dataset structure, summary statistics, and missing data.
- **Key Insights:** Identified top products, order quantities, revenue by region, and customer segments.
- **Correlation Analysis:** Explored relationships between order quantity, profit, and sales trends.

- **Trend Analysis:** Analyzed monthly sales patterns to identify seasonal trends.
- **Performance Metrics:** Assessed sales and profit by product, customer, region, and shipping type.
- **Customer Segmentation:** Identified top customers and cities contributing to revenue.
- **Visualization:** Created Sunburst and correlation heatmaps for deeper insights.

Conclusion

This E-commerce analysis provided valuable insights into sales trends, customer behavior, product performance, and shipping efficiency. By leveraging Python and data visualization techniques, we identified top-selling products, high-value customers, regional sales performance, and delivery inefficiencies. These findings enable businesses to optimize inventory based on demand patterns, enhance logistics by addressing shipment delays, improve marketing strategies by targeting high-revenue regions and customers, and increase profitability by refining pricing and sales strategies. This project serves as a data-driven roadmap for business growth and operational efficiency in the E-commerce sector.

Google Colab

Link: ***Ecommerce_recoded.ipynb***