



**Internship Report**  
**On**  
**Data science**  
(Using Python libraries)

**Submitted by**

Udaya Nath Gour  
Email – udayanathgour@gmail.com  
Aspirant, Data Science Batch

**Submitted to**

Mallika Srivastava  
Head, Training Delivery  
EI Systems Service  
&  
Mayur Dev Sewak  
Head, Internships & Trainings  
EI Systems Services

## Table of Contents

Serial No	Title	Page No
1	Overview of Organization	3
2	Project Summary	4
3	Data Flow Diagram / Process Flow	5-6
4	Objectives	7
5	Data Processing	8-12
6	Data Visualisation	12-16
7	References	17
8	Weekly Progress Report	18

# Overview of Organization

## Introduction of the Organization

EI Systems Services, widely known as EISys, is a premier Indian technology organization with extensive operations across the country. The company is a leader in delivering training services in cutting-edge fields such as Cybersecurity, Machine Learning, Automobiles, Internet of Things (IoT), Robotics, and social media. EISys is dedicated to empowering both enterprises and students by providing them with the skills necessary to thrive in a rapidly evolving technological landscape. To date, EISys has trained approximately 50,000 students and has reached over 200,000 through various outreach initiatives, significantly impacting the student community across India since its inception.

## Vision:

EISys aims to be a driving force in shaping the future of technology education in India by equipping the next generation of professionals and enterprises with cutting-edge skills in innovative fields.

## Mission:

The organization's mission is to deliver high-quality, industry-relevant training programs that foster creativity, technical expertise, and lifelong learning. EISys is committed to making technology education accessible and impactful for students and professionals alike.

## Values:

EISys is guided by core values of **innovation, accessibility, integrity, and community impact**. The organization believes in empowering learners through practical, hands-on training and fostering an inclusive environment where technology is a tool for positive change.

## Policy of the Organization in Relation to the Intern Role:

As part of its commitment to nurturing talent, EISys emphasizes real-world applications in its internship programs. Interns are trained to use practical tools and technologies, such as Python and its associated libraries (NumPy, Pandas, Matplotlib, Seaborn), to analyse data and solve problems related to emerging fields like data science and machine learning. EISys's internship policy encourages hands-on experience, critical thinking, and innovation, ensuring that interns not only gain technical proficiency but also align with the organization's vision of fostering growth and excellence in technology.

## Project Summary

During my Data Science internship at **Ei Systems**, I undertook a comprehensive project focusing on data preprocessing, analysis, and visualization using Python and various data science libraries, including **NumPy**, **Pandas**, **Matplotlib**, and **Seaborn**. The project involved a dataset containing multiple columns such as TransactionID, CustomerID, TransactionDate, ProductID, ProductCategory, Quantity, PricePerUnit, TotalAmount, TrustPointsUsed, PaymentMethod, and DiscountApplied. My objective was to clean and analyse the dataset and provide meaningful insights, while ensuring data integrity and quality.

### Data Preprocessing

**Loading the Data:** I used the panda's library to import the dataset into a Jupyter Notebook for manipulation and analysis.

**Handling Missing Data:** During the initial exploration, I identified that some columns like PricePerUnit and TotalAmount had missing or blank values. Using pandas, I filled these missing values with zeros to ensure completeness.

### Data Validation

To ensure accurate preprocessing, I employed data validation techniques:

**Type Checking:** Ensured that all columns had the correct data types (e.g., int for numerical columns and datetime for transaction dates).

**Duplicate Removal:** Checked for duplicate entries to avoid skewed analysis results.

**Edge Case Consideration:** Considered edge cases like negative values, empty strings, or outliers and handled them accordingly.

### Data Aggregation and Grouping

I performed data aggregation to gain insights from the dataset:

**Grouping by Product Category:** I aggregated data to see the total sales and average PricePerUnit per product category.

### Exploratory Data Analysis (EDA)

Using **Matplotlib** and **Seaborn**, I visualized the data to explore relationships and trends:

**Sales Trend Analysis:** I plotted the total sales over time to see which days had the highest number of transactions.

### Results

<https://github.com/Udayagour14/EI-Systems-Report>

Through this project, I was able to demonstrate my ability to work with real-world data, clean and preprocess it, and draw actionable insights using Python's powerful data libraries. The results provided meaningful insights for potential business decisions and showed the importance of data validation, cleaning, and proper visualization in understanding customer behaviour and product trends.

## Data Flow Diagram / Process Flow

### 1. Input Data (CSV File)

- The raw dataset is loaded into the system using **Pandas** in a **Jupyter Notebook**.
- 

### 2. Data Preprocessing

- **Step 1: Handling Missing Data**
  - NaN and empty values in columns like PricePerUnit and TotalAmount are identified.
  - Fill missing values with 0.
- **Step 2: Handling Negative Values**
  - Convert negative values in PricePerUnit and TotalAmount to their positive equivalents using NumPy.
- **Step 3: Data Type Correction**
  - Ensure columns like TransactionDate are in datetime format, and numerical columns have the correct data type.

### 3. Data Validation

- Check for **duplicate entries** and remove them.
- Validate the data types to ensure the correctness of each field (e.g., dates, numbers).
- Handle **edge cases** such as outliers or improper data entries.

### 4. Data Aggregation and Grouping

- **Group by Product Category:**
  - Calculate total sales and average price per product category.
- **Group by Customer:**
  - Aggregate data based on customer purchase frequency and trust points usage.

### 5. Exploratory Data Analysis (EDA)

- **Step 1: Sales Trend Analysis:**
  - Use **Matplotlib** to plot total sales over time.
- **Step 2: Category-wise Sales Distribution:**
  - Visualize the sales across product categories using **Seaborn** bar plots.
- **Step 3: Payment Method Analysis:**
  - Analyse which payment methods are used most frequently.

## 6. Data Visualization and Insights

- Insights drawn from **EDA** are visualized to uncover trends:
  - Popular product categories, customer behaviour, and sales trends.
- Visualizations are used to present the data to decision-makers or stakeholders.

## 7. Output and Reporting

- The final cleaned dataset, analysis results, and visualizations are presented.
- **Project summary** and insights are documented, providing clear business recommendations.

**This flow diagram can be represented graphically as:**

```
graph LR; A["[Input Data (CSV)]"] --> B["[Data Preprocessing]"]; B --> C["[Data Validation]"]; C --> D["[Data Aggregation & Grouping]"]; D --> E["[Exploratory Data Analysis]"]; E --> F["[Data Visualization & Insights]"]; F --> G["[Final Output & Reporting]"]
```

[Input Data (CSV)] --> [Data Preprocessing] -->  
[Data Validation] --> [Data Aggregation &  
Grouping] --> [Exploratory Data Analysis] --> [Data  
Visualization & Insights] --> [Final Output &  
Reporting]

# Objectives

## Data Cleaning and Preprocessing:

- Identify and resolve missing or erroneous data points in the dataset (e.g., empty values in PricePerUnit and TotalAmount).
- Convert negative values in the dataset to positive ones to ensure data consistency.
- Correct data types for numerical, categorical, and date-time fields.

## Data Validation:

- Ensure data integrity through validation checks such as handling duplicates, data type correction, and identifying outliers or edge cases.

## Exploratory Data Analysis (EDA):

- Analyze the dataset to uncover trends and insights, including customer purchase behaviors, product category performance, and sales trends over time.
- Use Pandas for data manipulation and aggregation, followed by visualization using Matplotlib and Seaborn.

## Data Aggregation:

- Perform data grouping and aggregation to gain insights into key business metrics such as total sales, average price per unit, customer usage of trust points, and product performance.

## Data Visualization:

- Present key findings through visualizations, using graphs and charts to clearly communicate insights.
- Visualize trends in sales, product categories, customer behavior, and payment methods to assist in decision-making.

## Reporting:

- Summarize the project findings, including data analysis and insights, in a comprehensive report for stakeholders on my GitHub <https://github.com/Udayagour14/El-Systems-Report>

## Data Preprocessing

In the given CSV file, do the preprocessing.

### Import data from my GitHub link

[https://github.com/Udayagour14/El-Systems-Report/blob/main/Data%20Source%20\(sales\\_transactions\).csv](https://github.com/Udayagour14/El-Systems-Report/blob/main/Data%20Source%20(sales_transactions).csv)

**Below is a Python script that demonstrates preprocessing on the given data:**

```
!pip install pandas matplotlib seaborn numpy
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
df = pd.read_csv(r'C:\Users\PC-Udaya\Downloads\Data Source (sales_transactions).csv')  
df.head()
```

```
pd.set_option('display.max_rows',None)
```

```
df_original = df.copy()
```

```
df.isnull().sum()
```

```
df.info()
```

### Data Cleaning

```
df['PaymentMethod'].fillna(df['PaymentMethod'].mode()[0],inplace=True)
```

```
df['DiscountApplied'].fillna(0,inplace=True)
```

**There is no fixed or consistent value of ProductCategory as compare to PricePerUnit so i will fill 0 to NaN in PricePerUnit and TotalAmount.**

```
df['PricePerUnit'].fillna(0,inplace=True)
```

```
df['TotalAmount'].fillna(0,inplace=True)
```

```
df['TransactionDate'] = pd.to_datetime(df['TransactionDate'])
```



**Note-Ignoring NaN in CustomerID focusing on product sales or payment methods. This can provide insights into how many transactions occur without customer identification.**

**Otherwise, i will remove or replace NaN values with "unknown"**

```
df.info() df['Quantity'].replace(to_replace=[-1],value=1,inplace=True)
```

```
df['TotalAmount'] = df['TotalAmount'].abs()
```

## Data Analysis

```
df.describe()
```

```
sales_by_category = df.groupby('ProductCategory')['TotalAmount'].sum()
```

```
price_per_category = df.groupby('ProductCategory')['PricePerUnit'].mean()
```

```
df.info()
```

## Pivot tables

**Understand which customers contribute the most to sales**

```
sale_by_customer =
```

```
pd.pivot_table(df,values='TotalAmount',index='CustomerID',aggfunc=np.sum)
```

```
sale_by_customer
```

[ 54 ] :

TotalAmount	
CustomerID	
1001.0	870.0
1002.0	440.0
1003.0	2570.0
1004.0	1170.0
1005.0	80.0

**Payment methods receive the highest average discounts.**

```
Discount_by_payment = pd.pivot_table(df, values='DiscountApplied',  
index='PaymentMethod', aggfunc=np.mean)
```

```
Discount_by_payment
```

	DiscountApplied
PaymentMethod	
Cash	23.076923
Credit Card	19.000000
Trust Points	20.357143

Track sales trends over time for different product categories.

```
sales_by_date_category = pd.pivot_table(df, values='TotalAmount',
index='TransactionDate', columns='ProductCategory', aggfunc=np.sum)
```

sales\_by\_date\_category

[62]:

ProductCategory	Electronics	Fashion	Grocery	Home Decor	Toys
TransactionDate					
2024-01-08 04:00:00	NaN	NaN	NaN	NaN	0.0
2024-01-08 08:00:00	60.0	NaN	NaN	NaN	NaN
2024-01-08 13:00:00	NaN	NaN	NaN	30.0	NaN
2024-01-08 14:00:00	NaN	1000.0	NaN	NaN	10.0
2024-01-08 23:00:00	NaN	NaN	NaN	NaN	50.0
2024-02-08 03:00:00	NaN	NaN	NaN	0.0	NaN
2024-02-08 08:00:00	NaN	1000.0	NaN	NaN	NaN
2024-02-08 15:00:00	NaN	NaN	NaN	NaN	30.0
2024-02-08 16:00:00	NaN	50.0	NaN	20.0	NaN
2024-02-08 19:00:00	NaN	NaN	90.0	NaN	NaN
2024-02-08 20:00:00	NaN	NaN	200.0	NaN	NaN
2024-02-08 23:00:00	NaN	0.0	NaN	NaN	NaN
2024-03-08 04:00:00	NaN	0.0	NaN	NaN	NaN
2024-03-08 14:00:00	NaN	NaN	NaN	NaN	100.0
2024-03-08 16:00:00	20.0	NaN	NaN	NaN	NaN
2024-03-08 22:00:00	NaN	NaN	NaN	500.0	NaN
2024-04-08 08:00:00	0.0	NaN	NaN	NaN	NaN

Trust points are being used across different product categories.

```
Trustpoint_by_category =  
pd.pivot_table(df, values='TrustPointsUsed', index='ProductCategory', aggfunc=np.sum)
```

Trustpoint\_by\_category

[63]:

TrustPointsUsed	
ProductCategory	
Electronics	610
Fashion	90
Grocery	190
Home Decor	320
Toys	230

Identify which product categories are most frequently purchased.

```
Quantity_by_category = pd.pivot_table(df, values='Quantity', index='ProductCategory',  
aggfunc=np.sum)
```

Quantity\_by\_category

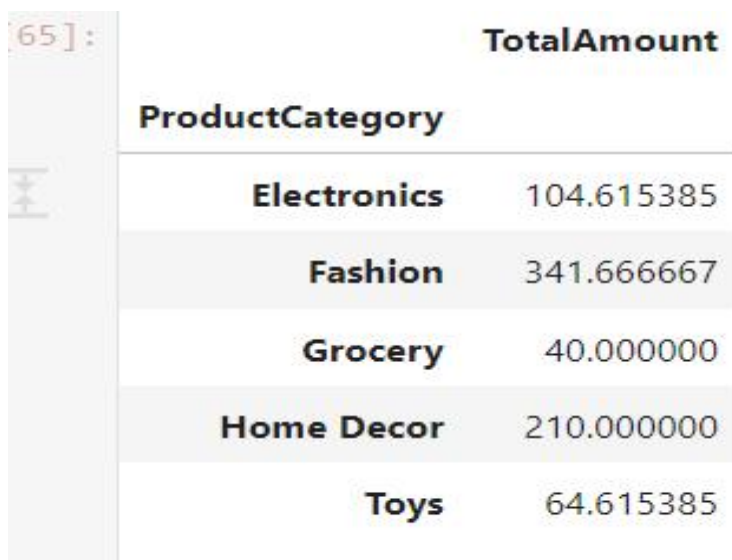
64]:

Quantity	
ProductCategory	
Electronics	16
Fashion	9
Grocery	19
Home Decor	14
Toys	21

Average transaction value for each product category.

```
Avg_sales_by_category = pd.pivot_table(df, values='TotalAmount',  
index='ProductCategory', aggfunc=np.mean)
```

Avg\_sales\_by\_category



The image shows a Jupyter Notebook cell with the index [65]:. The output is a pivot table with 'ProductCategory' as the index and 'TotalAmount' as the values. The table has five rows corresponding to the product categories: Electronics, Fashion, Grocery, Home Decor, and Toys. The average transaction values are 104.615385, 341.666667, 40.000000, 210.000000, and 64.615385 respectively.

ProductCategory	TotalAmount
Electronics	104.615385
Fashion	341.666667
Grocery	40.000000
Home Decor	210.000000
Toys	64.615385

## Data Visualization

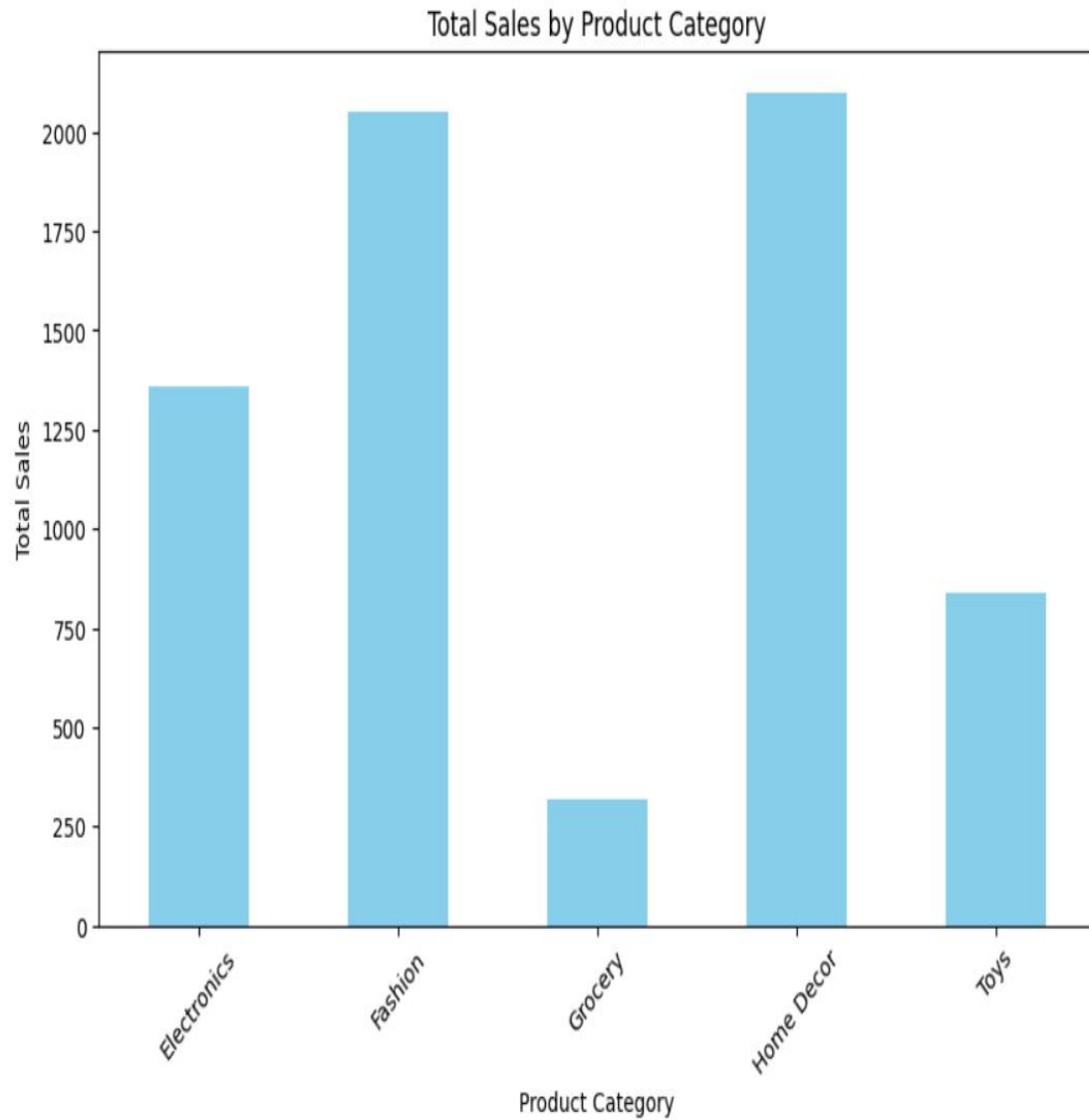
**Total Sales by Product Category**

```
category_sales = df.groupby('ProductCategory')['TotalAmount'].sum()  
plt.figure(figsize=(10,6))  
category_sales.plot(kind='bar', color='skyblue')  
plt.title('Total Sales by Product Category')  
plt.xlabel('Product Category')
```

```
plt.ylabel('Total Sales')
```

```
plt.xticks(rotation=45)
```

```
plt.show()
```



### Distribution of Payment Methods

```
payment_counts = df['PaymentMethod'].value_counts()
```

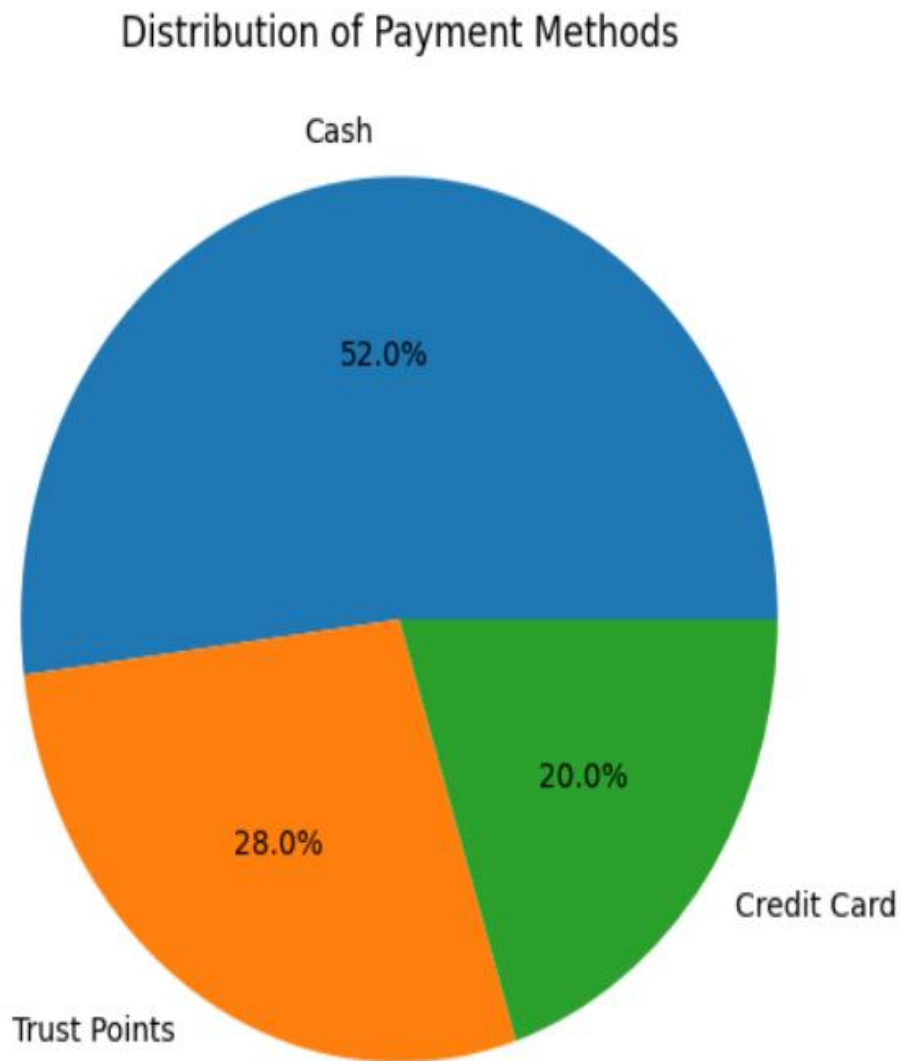
```
plt.figure(figsize=(10,6))
```

```
payment_counts.plot(kind='pie',autopct='%1.1f%%')
```

```
plt.title('Distribution of Payment Methods')
```

```
plt.ylabel('')
```

```
plt.show()
```



### **Visualize trends in sales over time**

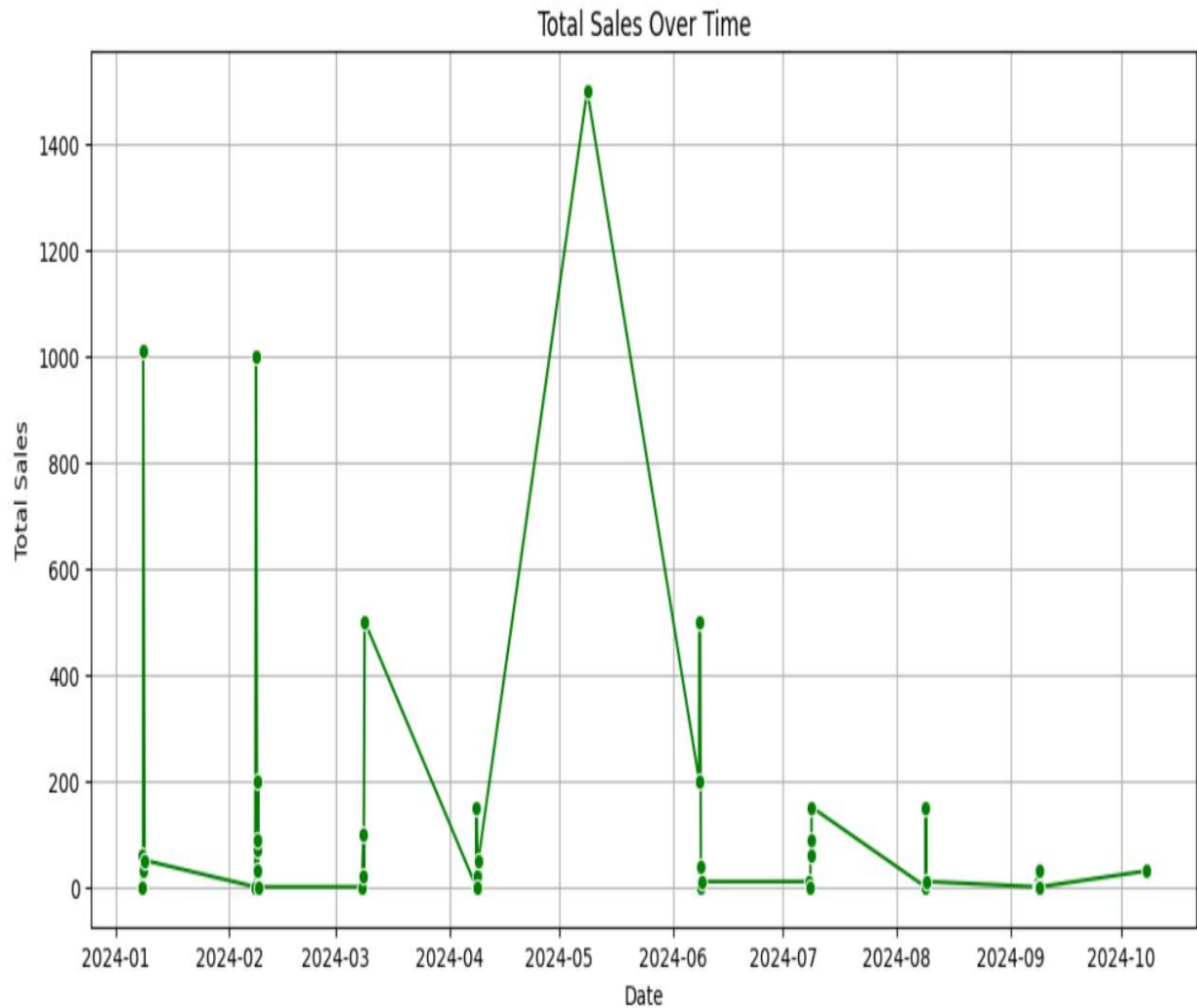
```
sales_over_time = df.groupby('TransactionDate')['TotalAmount'].sum()
sales_over_time = df.groupby('TransactionDate')['TotalAmount'].sum().reset_index()
plt.figure(figsize=(12, 6))

sns.lineplot(x='TransactionDate', y='TotalAmount', data=sales_over_time, marker='o',
color='green')

plt.title('Total Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
```

```
plt.grid()
```

```
plt.show()
```



**Shows the relationship between the quantity of items sold and the total amount, highlighting trends such as bulk purchases.**

```
plt.figure(figsize=(10, 6))
```

```
plt.scatter(df['Quantity'], df['TotalAmount'], color='purple')
```

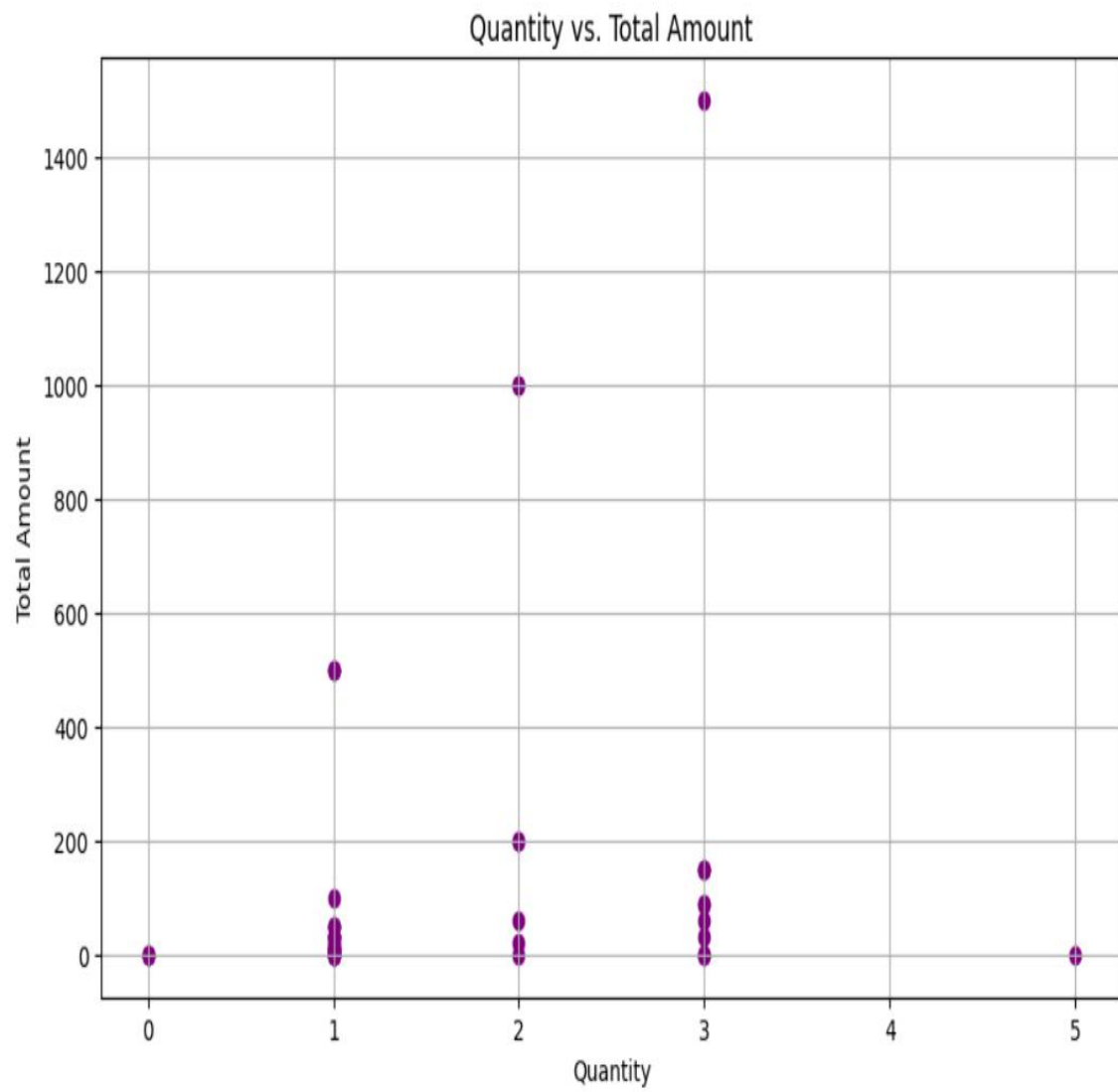
```
plt.title('Quantity vs. Total Amount')
```

```
plt.xlabel('Quantity')
```

```
plt.ylabel('Total Amount')
```

```
plt.grid(True)
```

```
plt.show()
```





## References

1. **My GitHub:** For importing CSV file

- [https://github.com/Udayagour14/El-Systems-Report/blob/main/Data%20Source%20\(sales\\_transactions\).csv](https://github.com/Udayagour14/El-Systems-Report/blob/main/Data%20Source%20(sales_transactions).csv)

2. **Pandas Documentation:** For data manipulation, cleaning, and aggregation operations.

- URL: <https://pandas.pydata.org/>

3. **NumPy Documentation:** For handling numeric data and applying mathematical functions.

- URL: <https://numpy.org/doc/>

4. **Matplotlib Documentation:** For creating static, animated, and interactive visualizations in Python.

- URL: <https://matplotlib.org/stable/contents.html>

5. **Seaborn Documentation:** For making statistical visualizations based on data analysis.

- URL: <https://seaborn.pydata.org/>

6. **Jupyter Notebook Documentation:** For running interactive data analysis and developing data science workflows.

- URL: <https://jupyter.org/documentation>

## Weekly Progress Report

Week(s)	Summary of Weekly Activity
Week 1	Introduction to Data Science & Machine Learning concepts.
Week 2	Revisiting Python basics, focusing on syntax and conditional statements.
Week 3	Introduction to Python programming
Week 4	Worked on Jupyter IDE and Python's conditional statements.
Week 5	Learned about different data types: List, Tuple, and Dictionary.
Week 6	Explored loops in Python: for, while, and nested loops.
Week 7	Studied functions in Python and explored function usage and packages.
Week 8	Installed and worked with various Python packages, including NumPy.
Week 9	Introduction to NumPy arrays, their creation, and indexing techniques.
Week 10	Performed data processing using NumPy arrays and different dimensions.
Week 11	Learned about Pandas for data analysis, including data types and DataFrames.
Week 12	Imported datasets and performed various operations on data using Pandas.
Week 13	Introduced to data visualization using Matplotlib, learning graph types.
Week 14	Implemented different types of graphs with Matplotlib tools.
Week 15	Explored machine learning algorithms using Scikit-learn and data processing and make project.
Week 16	Applied regression analysis and classification algorithms to datasets.

