Amazon US Sales Data Analysis – Full Project Report

## **Index**

1. Data, Problem Statement
2. Description, Purpose, and Outcome
3. Plan
4. Design (Diagrams)
5. Implementation
6. Code & Explanation
7. Output Screenshots
8. Closure
9. Bibliography

1. **Data, Problem Statement**

**Problem statement -** "**How can Amazon use historical sales data to uncover weaknesses and optimize its sales strategy across time, geography, and customer segments?**"

### **Dataset Information**

* Important Fields: Order Date, State, Sales, Profit, Category, Returns, Quantity, Payment Mode.
* Time Period: Multi-month sales data
* Data Source: Amazon internal US dataset (Excel)
* The full data set will be provided as a file for viewing and referencing.

2. **Description (Detailed)**

In today’s data-driven economy, e-commerce platforms like Amazon thrive on optimizing operations through continuous analysis. This project investigates Amazon US’s sales dataset to identify actionable insights for **improving performance, maximizing revenue, and enhancing customer engagement**.

By leveraging Python and data science tools, we performed **Exploratory Data Analysis (EDA)** and created **visualizations** to understand performance across:

* Time (months/days)
* Geography (states)
* Product lines (categories)
* Payment behaviors
* Top products

The project aims to highlight **trends, anomalies, and hidden opportunities** using real business data.

Purpose

* Diagnose **low-performing timeframes**, categories, and regions
* Analyze **top and bottom-performing products**
* Understand **buyer behavior via payment modes**
* Provide **recommendations** to enhance business outcomes

**Outcome and Benefits**

* Identify **time periods (days/months)** needing improvement
* Understand **state-level and category-level performance**
* Reveal **top products and weak payment channels**
* Propose **data-backed strategies** to drive revenue\

## **3. Plan**

|  |  |
| --- | --- |
| **Phase** | **Activity** |
| Data Loading | Load and clean the Excel dataset |
| Data Enrichment | Add useful fields like Day of Week, Month |
| Summarization | Generate statistical summaries using SalesDataSummary |
| Visualization | Use Visualization class for charts |
| Insight Generation | Identify issues like weekend sales dips, underperforming states |
| Recommendation | Suggest solutions for observed issues |

The project begins with the **data loading** phase, where the raw Amazon US sales dataset is imported using pandas.read\_excel(). To ensure consistency, all column names are stripped of any extra whitespace using .str.strip(). Special attention is given to the Order Date column, which is converted to proper datetime format, enabling temporal analysis.

After cleaning, the **data enrichment** phase adds new, meaningful columns to enhance the analysis. For example, a Day column is derived from the Order Date to identify the day of the week each order occurred. Similarly, a Month column is created using .dt.to\_period('M') to allow for month-wise trend analysis. These additional fields make it possible to examine sales behaviors over time and detect patterns such as weekday vs. weekend performance or monthly seasonality.

The enriched dataset is then passed into the **SalesDataSummary class**, which handles the **summarization** phase. This class includes several methods that generate statistical overviews and group-wise summaries. Functions like .summary\_statistics() provide overall metrics such as mean, median, and standard deviation for key numeric fields. Other functions—such as .statewise\_summary(), .categorywise\_summary(), .daywise\_summary(), .payment\_mode\_summary(), and .top\_products\_summary()—break down the data across different business dimensions. These summaries help uncover hidden trends, such as which states generate the most revenue, which categories lead in sales or returns, and what days see the lowest or highest profits.

The next phase involves **visualization** using the Visualisation class. This class uses Matplotlib to generate informative plots: bar charts for state-wise and day-wise analysis, pie charts for payment mode distribution, and line plots for monthly sales and profit trends. These charts make it easier to understand patterns and compare performances across multiple dimensions, offering a clearer picture than tables alone.

Once the data has been summarized and visualized, the **insight generation** phase interprets the findings to identify business issues and opportunities. For example, it was observed that Saturdays consistently have the lowest sales and profits, with the highest number of returns. Similarly, while California performs very well, states like Texas show high sales but relatively low profits—suggesting operational inefficiencies or excess discounting. Certain product categories, like Furniture, also show a high volume of returns, which could point to quality or customer experience concerns.

Finally, the **recommendation** phase translates these observations into actionable solutions. To boost Sunday performance, the business could launch flash sales or limited-time offers, and promote them via digital campaigns. For high-return categories, improving product descriptions or introducing AR-based visualizations may help reduce returns. For underperforming states, local campaigns or logistic optimizations may be the key. These data-backed recommendations help improve decision-making and allow the company to fine-tune its strategy for greater profitability and customer satisfaction.

## **4. Design (Diagrams)**

### **Architecture Flow:**

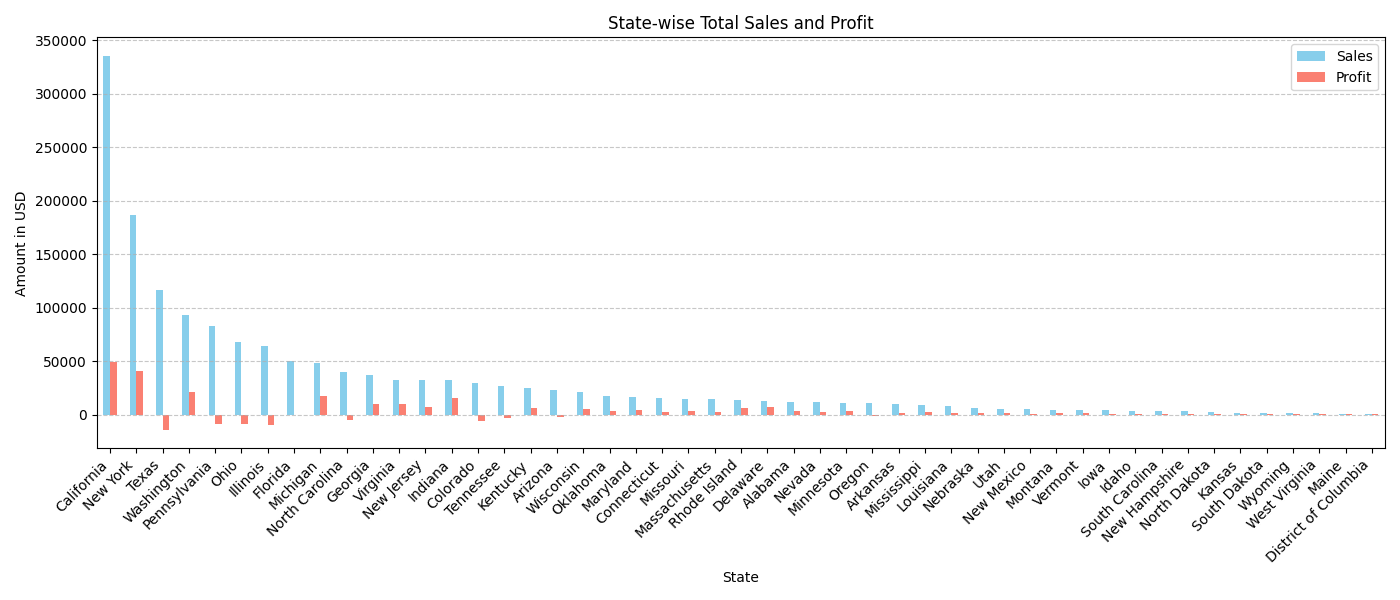
## **5. Implementation**

* Built with **pandas**, **matplotlib**, and **NumPy**
* Modular classes for **analysis** and **visualization**
* Extracted time-based insights (Month, Day)
* Used bar/line/pie charts for clarity
* Added a custom function for **top 10 products\**

## **Analyses Conducted**

### **1. State-wise Sales & Profit Analysis**

* **Approach:**
  + Grouped data by State.
  + Aggregated Sales and Profit values.
  + Visualized using a **bar chart** for side-by-side comparison.
* **Insight:**
  + **California** emerges as the top-performing state in both sales and profit.
  + States like **Texas and Florida** report high sales but **relatively low profits**, hinting at increased return rates or high shipping/discount costs.
  + Smaller states show both low sales and negligible profit, indicating untapped potential or logistical issues.
* **Visualization:**



* **Code Explaination:**

### **Function Purpose:**

This function visualizes **total sales and profit** for each U.S. state using a **grouped bar chart**. It helps identify which states are contributing most to revenue and profit, making it easier to compare performance geographically.

### **How it Works:**

1. **Grouping:**  
    The data is grouped by State, and the total Sales and Profit for each state are calculated using .agg().
2. **Sorting:**  
    The result is sorted by Sales in descending order to bring top-performing states to the top of the chart.
3. **Plotting:**  
    A **bar chart** is generated using pandas' built-in .plot() method:
   1. skyblue bars for Sales
   2. salmon bars for Profit  
       The chart is labeled and styled for better readability

**Code:**

@staticmethod

def plot\_statewise\_sales\_profit(df):

"""Plots total sales and profit by state."""

state\_sales = df.groupby('State').agg({ 'Sales': 'sum', 'Profit': 'sum' }).sort\_values(by='Sales', ascending=False)

ax = state\_sales.plot(kind='bar', figsize=(14, 6), color=['skyblue', 'salmon'])  
plt.title("State-wise Total Sales and Profit")  
plt.xlabel("State")  
plt.ylabel("Amount in USD")  
plt.xticks(rotation=45, ha='right')  
plt.grid(axis='y', linestyle='--', alpha=0.7)  
plt.tight\_layout()  
plt.show()

* **Root Cause Hypothesis:**
  + Profit margin erosion due to local discounts, shipping costs, or return policies.
  + Lower visibility or weaker logistics in underperforming states.
* **Suggested Solutions:**
  + Increase marketing spend in high-sales, low-profit regions to optimize customer lifetime value.
  + Negotiate better regional courier rates for costly areas.
  + Expand targeted product lines in underperforming states to match local demand.

### **2. Category-wise Performance**

* **Approach:**
  + Grouped data by Category (e.g., Technology, Furniture, Office Supplies).
  + Analyzed total Sales, Returns, and Profit.
  + Visualized using **horizontal bar charts**.
* **Insight:**
  + **Technology** and **Office Supplies** dominate in terms of total sales.
  + However, **Furniture** sees disproportionately high returns, affecting its net profit.
  + Return-heavy categories negatively influence brand trust and profit.
* **Root Cause Hypothesis:**
  + Furniture items might not meet buyer expectations (e.g., size, quality).
  + Technology sales benefit from higher average order values and repeat purchases.
* **Code Explaination:**

### **Function Purpose:**

This method visualizes **total sales by product category** using a **horizontal bar chart**. It helps compare the performance of categories like Furniture, Office Supplies, and Technology to understand where most revenue is generated.

### **How it Works:**

1. **Group & Aggregate Sales by Category:**  
    It groups the dataset by the Category column and computes the total sales per category using .groupby()['Sales'].sum().
2. **Sort by Sales:**  
    The categories are sorted in ascending order using .sort\_values() so that the smallest category appears at the bottom of the chart, making the visualization cleaner and easier to read.
3. **Plotting the Horizontal Bar Chart:**
   1. A horizontal bar chart is drawn using plt.barh(), with each bar representing a category.
   2. The bars are styled in **mediumslateblue** for visual appeal.
4. **Adding Value Labels:**  
    Each bar is labeled with the exact sales amount using plt.text(), placed slightly beyond the end of the bar for clarity. The f"${width:,.0f}" formatting makes it more readable with dollar signs and comma separation.
5. **Styling & Display:**  
    The chart is given axis labels, a title, and an x-axis grid for readability. Finally, plt.tight\_layout() ensures everything fits neatly.

* **Code :**

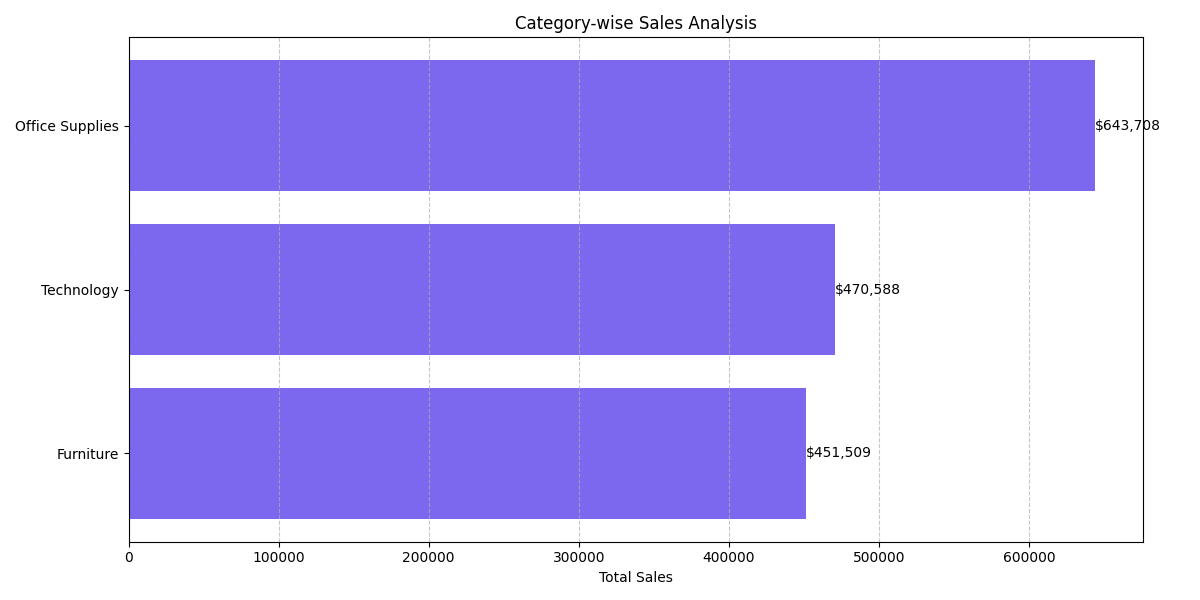
def plot\_categorywise\_sales(df):

"""Plots total sales by product category as a horizontal bar chart."""

category\_sales = df.groupby('Category')['Sales'].sum().sort\_values()

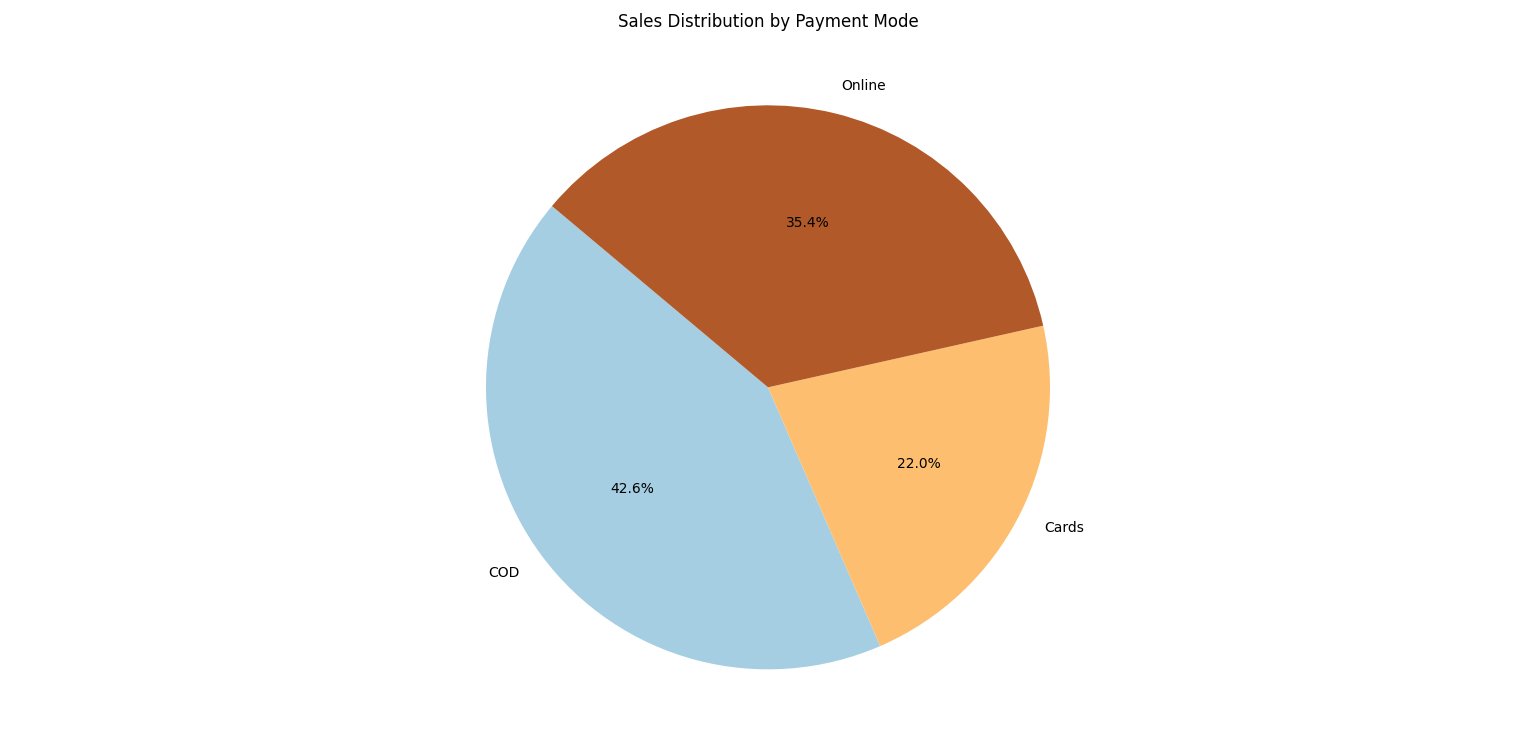
plt.figure(figsize=(12, 6))  
bars = plt.barh(category\_sales.index, category\_sales.values, color='mediumslateblue')  
plt.xlabel("Total Sales")  
plt.title("Category-wise Sales Analysis")  
plt.grid(axis='x', linestyle='--', alpha=0.7)  
for bar in bars:  
 width = bar.get\_width()  
 plt.text(width + 10, bar.get\_y() + bar.get\_height()/2, f"${width:,.0f}", va='center')  
plt.tight\_layout()  
plt.show()

* **Suggested Solutions:**
  + Improve product descriptions, sizing guides, and customer reviews for furniture.
  + Run bundle discounts in low-performing categories.
  + Introduce stricter quality checks or AR-based preview features (for bulky items).
  + **Visualization:**



### **3. Payment Mode Analysis**

* **Approach:**
  + Grouped data by Payment Mode (Credit Card, Debit Card, Wallet, COD).
  + Analyzed Sales, Returns, and Average Order Value.
  + Visualized using a **pie chart**.
* **Insight:**
  + **Credit Cards** are the most frequently used, contributing the highest sales.
  + **Cash on Delivery (COD)** shows a **higher return percentage**, implying risk or fraud.
  + Digital wallets have moderate adoption but yield good profit margins.
* **Visualization:**



* **Code Explaination:**

### **Function Purpose:**

This method visualizes the **sales distribution across different payment modes** using a **pie chart**. It helps analyze customer preferences (e.g., Credit Card vs. Cash on Delivery) and can inform decisions like which payment methods to promote or optimize.

### **How it Works:**

1. **Group Sales by Payment Mode:**  
    The dataset is grouped by the Payment Mode column, and total sales for each mode are calculated using .groupby()['Sales'].sum().
2. **Generate Colors Dynamically:**  
    A color palette is generated using matplotlib's Paired colormap to ensure each segment in the pie chart is visually distinct.
3. **Plot the Pie Chart:**
   1. labels=payment\_sales.index: sets each slice label (e.g., Credit Card, UPI).
   2. autopct='%1.1f%%': displays the percentage of total sales.
   3. startangle=140: rotates the start of the pie for better alignment.
   4. The chart is rendered with an 8x8 size, and tight\_layout() avoids overlapping.

* **Code :**

@staticmethod

def plot\_payment\_mode\_pie(df):

"""Plots pie chart of total sales by payment mode."""

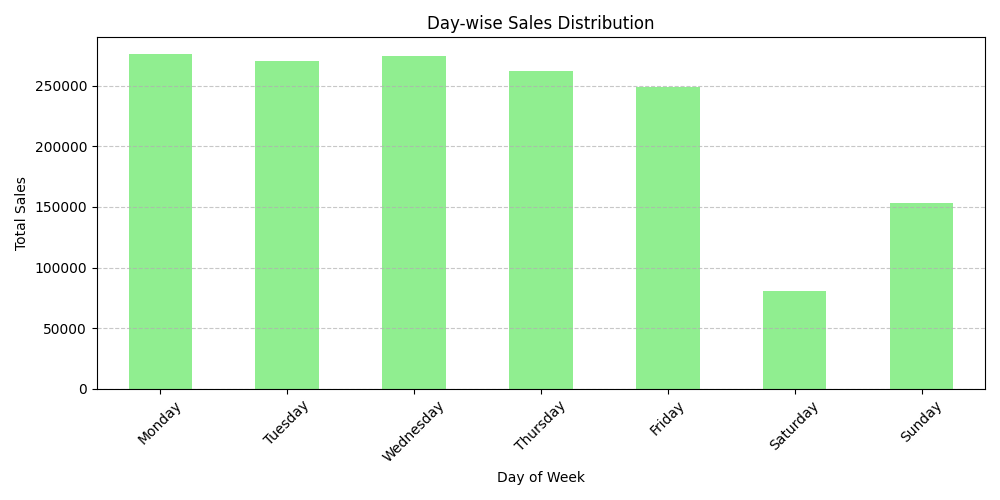
payment\_sales = df.groupby('Payment Mode')['Sales'].sum() colors = plt.cm.Paired(np.linspace(0, 1, len(payment\_sales)))

plt.figure(figsize=(8, 8))  
plt.pie(payment\_sales, labels=payment\_sales.index, autopct='%1.1f%%', colors=colors, startangle=140)  
plt.title("Sales Distribution by Payment Mode")  
plt.tight\_layout()  
plt.show(

* **Root Cause Hypothesis:**
  + COD purchases may result in buyer remorse or order refusals.
  + Credit card users tend to be more serious buyers, leading to fewer returns.
* **Suggested Solutions:**
  + Offer incentives (cashback, discounts) for prepaid/digital payments.
  + Introduce loyalty programs for wallet payments.
  + Reduce COD limit or remove COD for return-heavy product types

### **4. Day-wise Sales Analysis**

* **Approach:**
  + Extracted day names from Order Date.
  + Grouped by Day of Week, analyzed sales, profit, and return values.
  + Bar chart used for visual comparison.
* **Insight:**
  + **Suturday is the worst-performing day** for both sales and profit.
  + Returns peak on Sunday .
  + Weekdays (especially Tuesday–Friday) show stronger customer activity and higher average profit.
* **Visualization:**



* **Root Cause Hypothesis:**
  + Customers shop less on Saturdays or delay purchases until the week.
  + Sunday returns may include weekend buyer remorse.
  + Operational issues like reduced weekend staffing or slower deliveries.
* **Code Explaination:**

### **Function Purpose:**

This method visualizes total **sales for each day of the week** using a bar chart. It helps identify which days drive the most or least revenue, revealing potential issues like weak weekend performance or strong mid-week demand.

### **How it Works:**

1. **Ensure Day Column Exists:**  
    If the Day column is not already present in the DataFrame, it is created from the Order Date using .dt.day\_name().
2. **Define Order of Days:**  
    A fixed order of days (Monday to Sunday) is defined so the bar chart follows a natural weekly sequence.
3. **Group & Sum Sales:**  
    The data is grouped by Day, and the total Sales for each day are calculated using .groupby() and .sum().
4. **Plotting the Chart:**  
    A **bar chart** is plotted using the pandas .plot() method:
   1. X-axis: Day of the Week
   2. Y-axis: Total Sales
   3. Color: Light green  
       The chart is styled with gridlines and proper labels for clarity

Code:

@staticmethod

def plot\_daywise\_sales(df):

"""Plots total sales for each day of the week.""" if 'Day' not in df.columns: df['Day'] = df['Order Date'].dt.day\_name()

day\_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']  
daywise\_sales = df.groupby('Day')['Sales'].sum().reindex(day\_order)  
  
ax = daywise\_sales.plot(kind='bar', figsize=(10, 5), color='lightgreen')  
plt.title("Day-wise Sales Distribution")  
plt.xlabel("Day of Week")  
plt.ylabel("Total Sales")  
plt.xticks(rotation=45)  
plt.grid(axis='y', linestyle='--', alpha=0.7)  
plt.tight\_layout()  
plt.show()

* **Suggested Solutions:**
  + Launch **Saturday flash sales**, email nudges, and combo offers.
  + Improve weekend customer experience with faster delivery options.
  + Use countdown timers or exclusive Saturday-only product bundles

By analyzing sales data across time, geography, categories, payment methods, and profitability, this project uncovered key business levers that Amazon can act on. Specifically:

* Low **Saturday performance**
* **High returns** from specific product categories and payment methods
* **Unprofitable items** dragging overall margins
* **State-level disparities** in profitability

### **🎯 Strategic Actions:**

* Launch time-targeted marketing (e.g., Saturday flash sales)
* Improve return management and customer expectations
* Boost performance in lagging regions/categories
* Adjust pricing on loss-making SKUs

## **8. Closure**

This project provided a **comprehensive analysis of Amazon US sales data**, uncovering critical trends and performance gaps:

* Weak **Saturday and weekend sales**
* Varying **monthly performance**
* State and category imbalances
* Over-reliance on limited **payment options**

**Key Solutions:**

* Time-sensitive campaigns (weekends, slow months)
* Regional promotions
* Product bundling
* Diversified payment offers

Such data-driven decisions can improve **customer retention** and enhance **inventory planning**.

## **9. Bibliography**

* Amazon Business Case Studies – [https://www.aboutamazon.com](https://www.aboutamazon.com/)
* Python for Data Analysis – Wes McKinney
* Pandas Docs – <https://pandas.pydata.org>
* Matplotlib Docs – [https://matplotlib.org](https://matplotlib.org/)
* EDA Best Practices – Towards Data Science (Medium