INTRODUCTION

- This project's goal is to perform a thorough investigation of Amazon sales transactions using an actual dataset. This dataset includes detailed information such as order ID, purchase date, product type, amount purchased, order status, fulfillment method, sales channel, and transaction value.
- The major purpose is to gain real business insights that can help with strategic decision-making in areas such as sales performance, product trends, consumer behavior, and operational efficiency.

Libraries

1. Pandas - Used for Data manipulation & analysis
2. Numpy - Used for Numerical computing, arrays, and matrix operations
3. Seaborn - Used for Statistical data visualization (built on top of Matplotlib)
4. Matplotlib.pyplot - Used for Basic plotting (line, bar, scatter, etc.)

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading the dataset

In [2]:
```
dt = pd.read_csv('Amazon Sale Report.csv', encoding='ISO-8859-1')
```

dt.head()

To check the 1st 5 rows of the dataset and chechk what all columns are present in the dataset.

In [3]:
```
dt.head()
```

Out[3]:

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Category	S
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	T-shirt	
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	Shirt	3
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	Shirt	
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	Blazzer	
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	Trousers	3

5 rows × 21 columns

dt.info

To get the information about the columns and their data types as well as non-values are present in which columns.

In [4]:
```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128976 entries, 0 to 128975
Data columns (total 21 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   index               128976 non-null  int64  
 1   Order ID            128976 non-null  object 
 2   Date                128976 non-null  object 
 3   Status              128976 non-null  object 
 4   Fulfilment          128976 non-null  object 
 5   Sales Channel       128976 non-null  object 
 6   ship-service-level  128976 non-null  object 
 7   Category            128976 non-null  object 
 8   Size                128976 non-null  object 
 9   Courier Status      128976 non-null  object 
 10  Qty                 128976 non-null  int64  
 11  currency            121176 non-null  object 
 12  Amount              121176 non-null  float64
 13  ship-city           128941 non-null  object 
 14  ship-state          128941 non-null  object 
 15  ship-postal-code    128941 non-null  float64
 16  ship-country        128941 non-null  object 
 17  B2B                 128976 non-null  bool   
 18  fulfilled-by        39263 non-null   object 
 19  New                 0 non-null       float64
 20  PendingS            0 non-null       float64
dtypes: bool(1), float64(4), int64(2), object(14)
memory usage: 19.8+ MB
```

dt.shape

To determine the dimensions of a DataFrame. It returns a tuple representing the number of rows and columns in the DataFrame, respectively.

In [5]:
```
dt.shape
```

Out[5]:
```
(128976, 21)
```

dt.describe

To generate descriptive statistics of a DataFrame. It summarizes the central tendency, dispersion, and shape of the dataset's distribution.

For numerical data, the output includes: Count: The number of non-null values. Mean: The average value. std: The standard deviation. min: The minimum value. 25%: The 25th percentile. 50%: The median (50th percentile). 75%: The 75th percentile. max: The maximum value.

For object data (e.g., strings), the output includes: Count: The number of non-null values. Unique: The number of unique values. Top: The most frequent value. Freq: The frequency of the most frequent value.

The method analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. By default, it only describes numeric columns, but this behavior can be modified using the include and exclude parameters.

In [6]:
```
dt.describe()
```

Out[6]:

	index	Qty	Amount	ship-postal-code	New	Pendi
count	128976.000000	128976.000000	121176.000000	128941.000000	0.0	
mean	64486.130427	0.904401	648.562176	463945.677744	NaN	N
std	37232.897832	0.313368	281.185041	191458.488954	NaN	N
min	0.000000	0.000000	0.000000	110001.000000	NaN	N
25%	32242.750000	1.000000	449.000000	382421.000000	NaN	N
50%	64486.500000	1.000000	605.000000	500033.000000	NaN	N
75%	96730.250000	1.000000	788.000000	600024.000000	NaN	N
max	128974.000000	15.000000	5584.000000	989898.000000	NaN	N

dt.isnull().sum()

Returns the number of missing values in the dataset.

In [7]:
```
dt.isnull().sum()
```

Out[7]:
```
index                      0
Order ID                   0
Date                       0
Status                     0
Fulfilment                 0
Sales Channel              0
ship-service-level         0
Category                   0
Size                       0
Courier Status             0
Qty                        0
currency                7800
Amount                  7800
ship-city                 35
ship-state                35
ship-postal-code          35
ship-country              35
B2B                        0
fulfilled-by           89713
New                   128976
PendingS              128976
dtype: int64
```

Data Cleaning

In [8]:
```
# Drop irrelevant and empty columns from the dataset
dt_clean = dt.drop(columns = ['index', 'New', 'PendingS'])
```

In [9]:
```
# Convert 'Date' column to datetime format
dt_clean['Date'] = pd.to_datetime(dt_clean['Date'], errors='coerce', form
```

In [10]:
```
# Drop rows with missing Date or Amount (necessary for time-based and rev
dt_clean = dt_clean.dropna(subset=['Date', 'Amount'])
```

In [11]:
```
# Review the cleaned data
dt_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69416 entries, 0 to 128975
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Order ID            69416 non-null  object
 1   Date                69416 non-null  datetime64[ns]
 2   Status              69416 non-null  object
 3   Fulfilment          69416 non-null  object
 4   Sales Channel       69416 non-null  object
 5   ship-service-level  69416 non-null  object
 6   Category            69416 non-null  object
 7   Size                69416 non-null  object
 8   Courier Status      69416 non-null  object
 9   Qty                 69416 non-null  int64
 10  currency            69416 non-null  object
 11  Amount              69416 non-null  float64
 12  ship-city           69401 non-null  object
 13  ship-state          69401 non-null  object
 14  ship-postal-code    69401 non-null  float64
 15  ship-country        69401 non-null  object
 16  B2B                 69416 non-null  bool
 17  fulfilled-by        23266 non-null  object
dtypes: bool(1), datetime64[ns](1), float64(2), int64(1), object(13)
memory usage: 9.6+ MB
```

In [12]:
```
dt_clean.shape
```

Out[12]:
```
(69416, 18)
```

1) Sales Overview

Sort the data by Order Date (daily or monthly).

Visualize total sales and quantity sold over time.

Examine for seasonality or increases (for example, festival months and promotions).

In [13]:
```
# 'Date' to datetime format
dt_clean['Date'] = pd.to_datetime(dt_clean['Date'], errors='coerce', form
```

In [14]:
```
# Drop rows with missing values in 'Date' or 'Amount'
dt_clean = dt_clean.dropna(subset=['Date', 'Amount'])
```

In [15]:
```
# Add a Month-Year column
dt_clean['Month'] = dt_clean['Date'].dt.to_period('M')
```

In [16]:
```
# Group by month to get sales trends
monthly_sales = dt_clean.groupby('Month').agg({'Amount': 'sum', 'Order ID
monthly_sales['Month'] = monthly_sales['Month'].astype(str)
```

In [18]:
```
# Total Unique Orders
total_orders = dt_clean['Order ID'].nunique()

# Total Revenue (only where status indicates shipped/delivered)
shipped_orders = dt_clean[dt_clean['Status'].str.contains('Shipped', case
total_revenue = shipped_orders['Amount'].sum()

# Cancelled Orders (where status indicates cancellation)
cancelled_orders = dt_clean[dt_clean['Status'].str.contains('Cancelled',

# Display the results
print("Total Orders:", total_orders)
print("Total Revenue (INR):", total_revenue)
print("Cancelled Orders:", cancelled_orders)
```

```
Total Orders: 64579
Total Revenue (INR): 40538727.0
Cancelled Orders: 5679
```

In [ ]:
```
# Plotting Sales Overview
plt.figure(figsize=(14, 6))

# Line plot for sales amount
sns.lineplot(data=monthly_sales, x='Month', y='Amount', marker='o', label

# Bar plot for quantity sold
ax2 = plt.twinx()  # Create a secondary y-axis
sns.barplot(data=monthly_sales, x='Month', y='Qty', alpha=0.4, label='Tot

# Chart titles and labels
plt.title('📈 Monthly Sales Performance Overview')
plt.xlabel('Month')
ax2.set_ylabel('Quantity Sold', color='skyblue')
plt.ylabel('Sales Amount (₹)', color='darkblue')

# Rotate x-axis labels
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

Conclusions

Sales rises from March to April but from April to June there is quite decline and for high revenue months indicate effective marketing or seasonal buying. Also, high revenue months require advance inventory stocking. Months with low sales have to be checked for proper marketing and promotion of the item.

2) Product Analysis

In [ ]:
```
# Set consistent style
sns.set(style="whitegrid")
```

a) Top Product Categories by Quantity Sold

In [ ]:
```
category_sales = dt_clean.groupby('Category')['Qty'].sum().sort_values(as
```

In [ ]:
```
plt.figure(figsize=(12, 6))
sns.barplot(data=category_sales, x='Category', y='Qty', palette='Blues_d'
plt.title('Top Product Categories by Quantity Sold')
plt.xlabel('Product Category')
plt.ylabel('Total Quantity Sold')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

The Graph above shows the top product categories sold in terms of quantity sold which infers that T-shirt are the most sold products and after T-shirts, Shirts are mostly sold and so on. The graph also clearly shows that T-shirts generate greater money than other products.

b) Size Distribution

In [ ]:
```
size_sales = dt_clean.groupby('Size')['Qty'].sum().sort_values(ascending=
```

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.barplot(data=size_sales, x='Size', y='Qty', palette='Purples_d')
plt.title('Sales Distribution by Product Size')
plt.xlabel('Product Size')
plt.ylabel('Total Quantity Sold')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

RESULT: From the graph it is clear that sizes M,L,XL are the top selling sizes whereas sizes larger than 3XL are sold in much smaller quantities.

CONCLUSION: Most customers prefer L size and high demand of medium to large sizes so inventory of these sizes must be maintained.

In [ ]:
```
dt_clean
```

3) Fulfillment Analysis

a) Fulfillment Method

In [ ]:
```
# Count of orders by Fulfilment
fulfillment_counts = dt_clean['Fulfilment'].value_counts().reset_index()
fulfillment_counts.columns = ['Fulfilment Method', 'Number of Orders']
```

In [ ]:
```
# Bar Chart
plt.figure(figsize=(8, 5))
sns.barplot(data=fulfillment_counts, x='Fulfilment Method', y='Number of
plt.title('Distribution of Fulfillment Methods')
plt.ylabel('Number of Orders')
plt.xlabel('Fulfillment Method')
plt.tight_layout()
plt.show()
```

The graph above shows that Amazon has fulfilled more orders than the orders fulfilled by the merchant so we can say that amazon dominates with more orders fullfilled.

b) Effectiveness by Order Status

In [ ]:
```
# Crosstab between Fulfilment and Order Status
fulfillment_status = pd.crosstab(dt_clean['Fulfilment'], dt_clean['Status
```

In [ ]:
```
# Stacked bar chart
fulfillment_status.plot(kind='bar', stacked=True, figsize=(10, 6), colorm
plt.title('Order Status by Fulfillment Method')
plt.xlabel('Fulfillment Method')
plt.ylabel('Number of Orders')
plt.xticks(rotation=0)

# Place legend outside the plot (right side)
plt.legend(title='Order Status', bbox_to_anchor=(1.05, 1), loc='upper lef
plt.tight_layout()
plt.show()
```

In [ ]:
```
# Clean and filter data (ensure essential fields are not missing)
dt_clean = dt.dropna(subset=['Order ID', 'Fulfilment'])

# Count the number of orders by fulfilment method
fulfilment_counts = dt_clean['Fulfilment'].value_counts()

# Display the result
print(fulfilment_counts)
```

Insights and Conclusions

Amazon has fulfilled more orders and higher revenue whereas higher return and cancelled orders when fulfilled by the merchant so most customers prefered that the order should be fulfilled by the amazon as it has higher completion rate. Other than this merchant performance must be monitor regularly.

4) Customer Segmentation

a) Based on Buying Behaviour

In [ ]:
```
# Group by customer (based on 'ship-city' or equivalent column)
customer_behavior = dt_clean.groupby('ship-service-level').agg({'Order ID
```

In [ ]:
```
# Rename columns for clarity
customer_behavior.columns = ['ship-service-level', 'Total Orders', 'Total

# Sort by top spenders
top_customers = customer_behavior.sort_values(by='Total Spend (₹)', ascen
```

In [ ]:
```
# Visualize
plt.figure(figsize=(12, 6))
sns.barplot(data=top_customers, x='ship-service-level', y='Total Spend (₹
plt.title('Top 10 Customers by Ship service')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

b) Based on Location Segment

In [ ]:
```
# Group by State and aggregate metrics
state_segmentation = dt_clean.groupby('ship-state').agg({'Order ID': 'nun
```

In [ ]:
```
state_segmentation.columns = ['State', 'Total Orders', 'Total Spend (₹)',
state_segmentation = state_segmentation.sort_values(by='Total Spend (₹)',
```

In [ ]:
```
# Visualize top states
plt.figure(figsize=(12, 6))
sns.barplot(data=state_segmentation.head(10), x='State', y='Total Spend (
plt.title('Top 10 States by Customer Spend')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Conclusion

From the above graph it is clear that most of the customers are from Maharastra and Karnataka whereas less number of customers are from West Bengal, Andhra Pradesh and Haryana.

5) Geographical Analysis

a) Grouping by States

In [ ]:
```
# Group by State and aggregate metrics
state_sales = dt_clean.groupby('ship-state').agg({'Order ID': 'nunique',
```

In [ ]:
```
# Plotting top 10 states by sales
plt.figure(figsize=(12, 6))
sns.barplot(data=state_sales.head(10), x='ship-state', y='Amount', palett
plt.title('Top 10 States by Total Sales Amount')
plt.xlabel('State')
```

```
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Insights and Interpretation

Maharashtra, Karnataka, and Telengana are often top contributors to revenue.

These states may benefit from localized marketing, warehouse stocking, or promotions.

b) Grouping by Cities

In [ ]:
```
# Group by City and aggregate metrics
city_sales = dt_clean.groupby('ship-city').agg({'Order ID': 'nunique', 'A
```

In [ ]:
```
# Plotting top 10 cities by sales
plt.figure(figsize=(12, 6))
sns.barplot(data=city_sales.head(10), x='ship-city', y='Amount', palette=
plt.title('Top 10 Cities by Total Sales Amount')
plt.xlabel('City')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Insights and Interpretation

Major metros like Bengaluru, Hyderabad, Mumbai and Delhi often dominate.

Consider running geo-targeted ads or setting up fulfillment hubs in high-sales zones.

Conclusion

Geographic insights help optimize inventory, shipping, and regional sales campaigns.

Focus efforts in high-performing regions while also identifying low-performing areas for growth potential.

Business Insights

Based on the data analysis, here are the key insights and actionable recommendations:

Sales Performance Overview

- **Total Orders:** 64,579
- **Total Revenue (Shipped Orders):** ₹4.05 Crores
- **Cancelled Orders:** 5,679 (approx. 8.8%)

📌 **Recommendation:** Investigate reasons behind cancellations. Focus on improving **order accuracy**, **payment success**, and **last-mile delivery** especially in high-cancel areas.

Product Insights

- **Top Selling Category:** T-shirts
- Likely due to high demand in fashion/apparel, especially low-cost items.

📌 **Recommendation:** Promote and stock more SKUs in this category. Use customer reviews and ratings to identify the best-performing T-shirt variants.

Fulfilment Effectiveness

- **Amazon Fulfilment:** 89,713 orders
- **Merchant Fulfilment:** 39,263 orders

📌 **Recommendation:** Amazon-fulfilled orders form the majority, suggesting better logistics or buyer preference. Encourage more sellers to shift to **Amazon FBA (Fulfilment by Amazon)** to reduce cancellation rates and delivery delays.

Geographical Demand

- **Top City:** Bengaluru
- **Top State:** Maharashtra

📌 **Recommendation:** Focus marketing efforts and logistics optimizations in these regions. They represent core demand hubs—ideal for **next-day delivery**, **targeted discounts**, and **region-specific inventory planning**.

Here is the **completed list of strategic recommendations** based on your Amazon sales data analysis:

Strategic Recommendations

1. Inventory Optimization

- **What to do:** Prioritize inventory stocking of high-demand products like T-shirts, especially in top-performing states (e.g., Maharashtra) and cities (e.g., Bengaluru).
- **Why:** Ensures better product availability, reduces delivery time, and avoids stockouts.

---

2. **Fulfilment Efficiency Enhancement**

- **What to do:** Encourage more merchants to adopt **Fulfilled by Amazon (FBA)** services.
- **Why:** Amazon-fulfilled orders show better performance, fewer cancellations, and faster delivery—enhancing customer satisfaction.

---

3. **Order Cancellation Reduction**

- **What to do:** Analyze top reasons for cancellations (e.g., delayed shipping, COD issues). Improve delivery promise accuracy and communication. Limit COD for risky regions or new users.
- **Why:** Reducing cancellation rate (currently ~8.8%) directly improves revenue and operational efficiency.

---

4. **Targeted Marketing Campaigns**

- **What to do:** Run localized and seasonal campaigns in top regions. Use email/SMS for cart abandonment recovery and re-engagement.
- **Why:** Increases conversion rate and builds brand loyalty in regions with established demand.

---

5. **Customer Segmentation & Personalization**

- **What to do:** Group customers into segments (e.g., frequent buyers, high spenders, inactive users). Use these segments to offer tailored discounts or loyalty rewards.
- **Why:** Personalization increases repeat purchases and customer lifetime value (CLTV).

---

6. **Geographic Expansion Planning**

- **What to do:** Identify underserved states/cities with growth potential and run pilot marketing or fulfillment tests there.
- **Why:** Expanding into high-potential regions with low competition can drive new revenue streams.

---

7. **Return & Refund Policy Optimization**

- **What to do:** Analyze refund/return reasons and streamline the process with customer-friendly automation.
- **Why:** Enhances trust and reduces support load, while also preserving revenue through fewer disputes.

---

8. **Merchant Training & Standardization**

- **What to do:** Provide onboarding, packaging, and shipping training to new or underperforming sellers.
- **Why:** Improves delivery times and reduces bad reviews, benefiting the entire marketplace.