AMAZON SALES REPORT ANALYSIS

DONE BY:

DINESH S

PANIMALAR INSTITUTE OF TECHNOLOGY

1. Introduction

The Amazon Sales Report Data Analysis project aims to provide insights into sales trends, revenue generation, and customer behavior by analyzing a comprehensive dataset of Amazon sales. Using Python and its data analysis libraries, this project covers various aspects of sales data to uncover patterns and trends.

2. Data Overview

2.1 Loading and Inspecting the Data

The analysis begins with loading the Amazon Sales Report dataset using the Pandas library. The first few rows of the dataset are displayed to get a glimpse of the data structure. Summary statistics and data types are also inspected using the info() and describe() functions. This initial step helps in understanding the dataset's composition, including the types of data available, missing values, and basic statistical information.

Python code

```
import pandas as pd
df = pd.read_csv('Amazon Sale Report.csv')
print(df.head())
print(df.info())
print(df.describe())
```

3. Sales Analysis

3.1 Sales Trends Over Time

To analyze sales trends, the 'Date' column is converted to a datetime format. The data is then grouped by date to calculate the total sales amount for each day. A time series plot is created to visualize the daily sales trend, which helps in identifying any patterns, spikes, or declines in sales over time.

Python code

```
df['Date'] = pd.to_datetime(df['Date'])
daily_sales = df.groupby('Date')['Amount'].sum()
daily_sales.plot()
```

3.2 Revenue Calculation

The total revenue generated from all non-canceled orders is calculated by filtering out canceled orders from the dataset. This provides a clear picture of the actual revenue earned from successful transactions. The total revenue is a critical metric for assessing the business's financial performance.

Python code

```
non_cancelled_orders = df[df['Status'] != 'Cancelled']
total_revenue = non_cancelled_orders['Amount'].sum()
```

4. Customer and Product Analysis

4.1 City with the Highest Number of Orders

By grouping the data by city, we can determine which city had the highest number of orders. The city with the maximum orders is identified, providing insights into the geographical distribution of sales. This information can be useful for targeted marketing and resource allocation.

Python code

```
city_order_counts = df.groupby('ship-city')['Order ID'].count()
city_with_max_orders = city_order_counts.idxmax()
```

4.2 Product Category Analysis

The dataset is grouped by product category to identify which category had the highest number of orders. Additionally, a bar plot is created to visualize sales by category, helping to identify which products are most popular and contribute the most to overall revenue.

Python code

```
category_order_counts = df.groupby('Category')['Order ID'].count()
category_sales =
df.groupby('Category')['Amount'].sum().sort_values(ascending=False)
category_sales.plot(kind='bar')
```

4.3 Average Order Amount for Delivered Orders

For orders that were successfully delivered to buyers, the average order amount is calculated. This metric helps in understanding the typical spending behavior of customers who receive their orders.

Python code

```
delivered_orders = df[df['Status'] == 'Shipped - Delivered to Buyer']
average_amount = delivered_orders['Amount'].mean()
```

5. Regional Sales Analysis

5.1 State with the Highest Revenue

To determine which state generated the highest total revenue, the data is grouped by state. The state with the maximum revenue is identified, providing insights into regional sales performance. This information can guide decisions regarding supply chain management and regional marketing strategies.

Python code

```
state_revenue = df.groupby('ship-state')['Amount'].sum()
highest_revenue_state = state_revenue.idxmax()
```

5.2 Sales by Shipping Service Level

The analysis also includes evaluating sales by different shipping service levels, such as expedited or standard shipping. Grouping the data by service level and calculating the total sales amount reveals how shipping choices impact sales.

Python code

```
shipping_method_sales = df.groupby('ship-service-
level')['Amount'].sum().sort_values(ascending=False)
shipping_method_sales.plot(kind='bar')
```

5.3 Sales by Country

Sales are also analyzed on a country-by-country basis by grouping the data by shipping country. This analysis helps in understanding the international reach of the sales and identifying key markets outside the primary region.

Python code

```
country_sales = df.groupby('ship-
country')['Amount'].sum().sort_values(ascending=False)
country_sales.plot(kind='bar')
```

6. Fulfillment and Order Distribution

6.1 Fulfillment by Amazon vs. Merchant

The dataset includes information on whether orders were fulfilled by Amazon or by a merchant. The distribution of fulfillment methods is visualized using a pie chart, highlighting the proportion of orders handled by Amazon compared to third-party merchants.

Python code

```
fulfilment_counts = df['Fulfilment'].value_counts()
plt.pie(fulfilment_counts, labels=fulfilment_counts.index,
autopct='%1.1f%%')
```

6.2 Sales by Sales Channel

Sales are analyzed based on the channel through which they were made (e.g., Amazon vs. non-Amazon channels). Grouping the data by sales channel and visualizing the results provides insights into the performance of different sales channels.

Python code

```
channel_sales = df.groupby('Sales
Channel')['Amount'].sum().sort_values(ascending=False)
channel_sales.plot(kind='bar')
```

7. Additional Insights

7.1 Business-to-Business Product Ordering

The project also explores back-to-back product ordering by analyzing sales data grouped by the B2B flag. A pie chart is used to visualize the distribution of B2B orders, providing insights into business-to-business sales.

Python code

```
b2b_sales = df.groupby('B2B')['Amount'].sum().sort_values(ascending=True)
b2b_sales.plot(kind='pie')
```

7.2 Most Common Size Ordered

The most common product sizes ordered by customers are identified by grouping the data by product category and size. This analysis is useful for inventory management and understanding customer preferences.

Python code

```
most_common_size = df.groupby('Category')['Size'].agg(lambda x:
x.mode()[0])
```

7.3 Orders Delivered by Easy Ship

The project also identifies the number of orders delivered using the "Easy Ship" method, offering insights into the popularity and efficiency of this shipping option.

Python code

```
easy_ship_orders = df[df['fulfilled-by'] == 'Easy Ship']
easy_ship_order_count = easy_ship_orders.shape[0]
```

8. Conclusion

This project successfully analyzed the Amazon Sales Report dataset, uncovering key insights into sales trends, customer behavior, and regional performance. Through data visualization and statistical analysis, we identified top-performing cities, product categories, and states, as well as significant revenue drivers. The findings provide actionable insights that can help optimize sales strategies, improve inventory management, and enhance customer satisfaction. Overall, this analysis serves as a valuable tool for understanding and leveraging the sales data to drive business growth.