Project Code Document

Author	**Shubham Gaikwad**
Company	**iNeuron.ai**
Project Name	**Amazon Sales Data Analysis**

Introduction:

The goal of this project is to analyse Amazon sales data using BI approaches and to do Exploratory Data Analysis (EDA) in a Jupyter Notebook. The dataset comprises Amazon sales data, including sales indicators, goods, and profitability. The analysis will be carried out in Python utilising a variety of packages such as pandas, matplotlib, and seaborn.

1. Data Pre-processing:

In this section, we will perform the Data Pre-processing on the dataset for this analysis:

- Unwanted Columns

- Duplicate Values

- Converting Invoice Date column in Date Format and Extracting Year, Month and Month_name from 'Invoice Date' column

-Handling Null Values

```
#Dropping Unwanted Columns
data.drop(['Unnamed: 21','@dropdown'],axis=1,inplace=True)
```

```
#Checking Duplicate Values
data[data.duplicated()]
```

```
# Converting Invoice Date column in Date Format and Extracting Year,Month and Month_name from 'Invoice Date' column
data['Invoice Date'] = pd.to_datetime(data['Invoice Date'])
data['Year'] = data['Invoice Date'].dt.year
data['Month'] = data['Invoice Date'].dt.month
data['month_name'] = data['Invoice Date'].dt.month_name()
```

```
#Checking Null Values
data.isnull().sum()
```

```
#Handling Null Values
data['Discount Amount']=data['Discount Amount'].fillna(0)
data['Item Class']=data['Item Class'].fillna('NA')
data['Item Number']=data['Item Number'].fillna(0)
data['Sales Price']=data['Sales Price'].fillna(0)
sns.heatmap(data.isnull())
```

2. Sales Analysis by Month and Year:

In this section, we will perform the following tasks:

- Analyse sales trends by month and year.

- Visualize the monthly and yearly sales using line charts or bar charts. - Identify peak sales months and years with the highest revenue.

```
Monthly_sales = data.groupby([ 'month_name'])['Sales Amount'].sum()
print(Monthly_sales)
```

```
Yearly_sales = data.groupby(['Year', ])['Sales Amount'].sum()
print(Yearly_sales)
```

```
Yearly_monthly_sales = data.groupby(['Year', 'month_name'])['Sales Amount'].sum()
print(Yearly_monthly_sales)
```

```
# Line chart for Sales Trend Month Wise
plt.figure(figsize=(15, 3))
sns.lineplot(x='month_name', y='Sales Amount', data=data, ci=None,color='green',linestyle='dashed',marker='o')
plt.xlabel('Months',color='black',size=15)
plt.ylabel('Sales',color='black',size=15)
plt.title('SALES TREND MONTH WISE',color='black',size=20)
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
sns.set_style(style="darkgrid")
plt.show()
```

```
# Catplot for Sales Trend Year Wise
plt.figure(figsize=(5, 5))
sns.catplot(x='Year', y='Sales Amount', data=data)
plt.xlabel('year',color='black',size=15)
plt.ylabel('Sales',color='black',size=15)
plt.title('SALES TREND YEAR WISE',color='black',size=20)
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
sns.set_style(style="darkgrid")
plt.show()
```

```
# Line chart for Sales Trend Yearly Month Wise
plt.figure(figsize=(15, 5))
sns.lineplot(x='month_name', y='Sales Amount', hue='Year', data=data, ci=None, linestyle='dashed',
             marker='o',palette=['green', 'orange', 'blue'])
plt.xlabel('Month', color='black', size=15)
plt.ylabel('Sales Amount', color='black', size=15)
plt.title('SALES TREND YEARLY MONTH WISE', color='black', size=20)
plt.legend(title='Year', title_fontsize=15, fontsize=10)
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
plt.tight_layout()
plt.show()
```

3. Analysis on Sales Metrics and Items:

In this section, we will perform the following tasks:

- Calculating and analysing sales metrics, such as average sales, total sales, and profit margin.

- Identify the top and least selling items

- Visualizations, such as column charts or bar charts, to represent the sales distribution of different items.

```
Metrics = ['Sales Amount', 'Sales Amount Based on List Price', 'Sales Cost Amount',
                'Sales Margin Amount', 'Sales Price']
Sales_metrics = data[sales_metrics].mean().sort_values(ascending=False)
Sales_metrics
```

```
Most10_item=data['Item'].value_counts().index[0:10]
Most10_item
```

```
Least10_item=data['Item'].value_counts().sort_values().index[0:10]
Least10_item
```

```
# Column chart for Sales Comparison

#Data
Metrics = ['Sales Amount', 'Sales Amount Based on List Price', 'Sales Cost Amount',
                'Sales Margin Amount', 'Sales Price']
Sales_metrics = data[Metrics].mean().sort_values(ascending=False)
#Plot
plt.figure(figsize=(12, 3))
color = ['green', 'yellow','orange', 'blue','red']
plt.bar(Sales_metrics.index,Sales_metrics.values,color =color)
plt.xlabel('Sales Metrics',size =15,color='black')
plt.ylabel('Average Value',size =15,color='black')
plt.title('COMPARISON OF SALES METRICS',size =20,color='black')
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
plt.tight_layout()
plt.show()
```

```
# Bar chart for Items
plt.figure(figsize=(6,3))
ax = sns.countplot(y="Item", data=data, order=data['Item'].value_counts().index[0:10])
plt.title("MOST 10 ITEMS SOLD",size=20,color='black')
plt.ylabel("Items",size =15,color='black')
plt.xlabel("Count",size =15,color='black')
sns.set_style('darkgrid')
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
for patch in ax.patches:
    height = patch.get_height()
    width = patch.get_width()
    ax.text(width + 5, patch.get_y() + height / 2, f'{int(width):,}', ha='right', va='center', fontsize=10, color='black')
plt.show()
```

```
# Bar chart for Items
plt.figure(figsize=(6,3))
ax = sns.countplot(y="Item", data=data, order=data['Item'].value_counts().sort_values().index[0:10])
plt.title("LEAST 10 ITEMS SOLD",size=20,color='black')
plt.ylabel("Items",size =15,color='black')
plt.xlabel("Count",size =15,color='black')
plt.xticks(size=12,color='black')
plt.yticks(size=12,color='black')
sns.set_style('darkgrid')
plt.show()
```

4. Sales and Profitability Analysis:

In this section, we will perform the following tasks:

- Conduct a comprehensive analysis of sales and profitability.

- Explore the relationship between sales and profitability.

- Visualize the sales and profitability

```
#Total sales amount
total_sales_amount = data['Sales Amount'].sum()
print("Total Sales Amount:", total_sales_amount)
```

```
#Total sales quantity
total_sales_quantity = data['Sales Quantity'].sum()
print("Total Sales Quantity:", total_sales_quantity)
```

```
# Average sales amount
average_sales_amount = data['Sales Amount'].mean()
print("Average Sales Amount:", average_sales_amount)
```

```
# Average sales quantity
average_sales_quantity = data['Sales Quantity'].mean()
print("Average Sales Quantity:", average_sales_quantity)
```

```
# Total sales margin amount
total_sales_margin_amount = data['Sales Margin Amount'].sum()
print("Total Sales Margin Amount:", total_sales_margin_amount)
```

```
# Average sales margin amount
average_sales_margin_amount = data['Sales Margin Amount'].mean()
print("Average Sales Margin Amount:", average_sales_margin_amount)
```

```
# Total discount amount
total_discount_amount = sum(data['Discount Amount'])
print("Total Discount Amount:", total_discount_amount)
```

```
# Total cost amount
total_cost_amount = sum(data['Sales Cost Amount'])
print("Total Cost Amount:", total_cost_amount)
```

```
#Total list price amount
total_list_price_amount = data['List Price'].sum()
print("Total List Price Amount:", total_list_price_amount)
```

```
#Total profit amount
total_profit_amount = total_sales_margin_amount - total_discount_amount
print("Total Profit Amount:", total_profit_amount)
```

```
# Pie Chart for Total Sales Amount and Total Sales Quantity

#Data
labels = ['Total Sales Amount', 'Total Sales Quantity']
values = [total_sales_amount, total_sales_quantity]
#Plot
plt.figure(figsize=(5, 5))
plt.pie(values, labels=labels, autopct='%1.1f%%',textprops={'fontsize': 12, 'color': 'black'})
plt.title('Total Sales Amount vs. Total Sales Quantity',size=15,color='black')
plt.show()
```

```python
# Donut Chart for Average Sales Amount and Average Sales Quantity

#Data
categories = ['Average Sales Amount', 'Average Sales Quantity']
values = [average_sales_amount, average_sales_quantity]
#Plot
plt.figure(figsize=(5, 5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=120, wedgeprops=dict(width=0.4),
        textprops=dict(size=12, color='black'))
centre_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Average Sales Amount vs. Average Sales Quantity',size=15,color='black')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

```python
# Donut Chart for Total Sales Margin and Average Sales Margin

#Data
categories = ['Total Sales Margin', 'Average Sales Margin']
values = [total_sales_margin_amount, average_sales_margin_amount]
#Plot
plt.pie(values, labels=categories, autopct='%1.1f%%', colors=['blue', 'green'], startangle=0,
        textprops=dict(size=12, color='black'))
centre_circle = plt.Circle((0, 0), 0.7, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Total and Average Sales Margin Amounts',size=15,color='black')
plt.axis('equal')
plt.show()
```

```python
# Donut Chart for Discount Amount and Cost Amount

#Data
categories = ['Discount Amount', 'Cost Amount']
values = [total_discount_amount, total_cost_amount]
#Plot
plt.figure(figsize=(5, 5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=90, wedgeprops=dict(width=0.4),
        textprops=dict(size=12, color='black'))
centre_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Total Discount Amount vs. Total Cost Amount',size=15,color='black')
plt.show()
```

```python
# Pie chart for Total List Price Amount and Total Profit Amount

#Data
categories = ['Total List Price Amount', 'Total Profit Amount']
values = [abs(total_list_price_amount), abs(total_profit_amount)]
#for negative values
if any(val < 0 for val in values):
    raise ValueError("Values must be non-negative for a pie chart.")
#Plot
plt.figure(figsize=(5, 5))
plt.pie(values, labels=categories, autopct='%1.1f%%', colors=['blue', 'green'], startangle=90,
        textprops={'fontsize': 12, 'color': 'black'})
plt.title('Total List Price vs. Total Profit', size=15, color='black')
plt.show()
```

Notebooks and Code:

Jupyter Notebook with the EDA code for analysing Amazon sales data. Include clear, fully-commented code, as well as any relevant explanations.

Conclusion:

Based on the data analysis results, summarise the findings, important insights, and actionable suggestions.