Slash Data Analysis Task

1- EDA

- Loading data

Code :

```
import pandas as pd
file_path = 'CSV file path'
df = pd.read_csv('file_path', encoding='latin-1')
print(df.head())
print(df.info())
print(df.describe())
print(df.describe(include='object'))
```

Results :

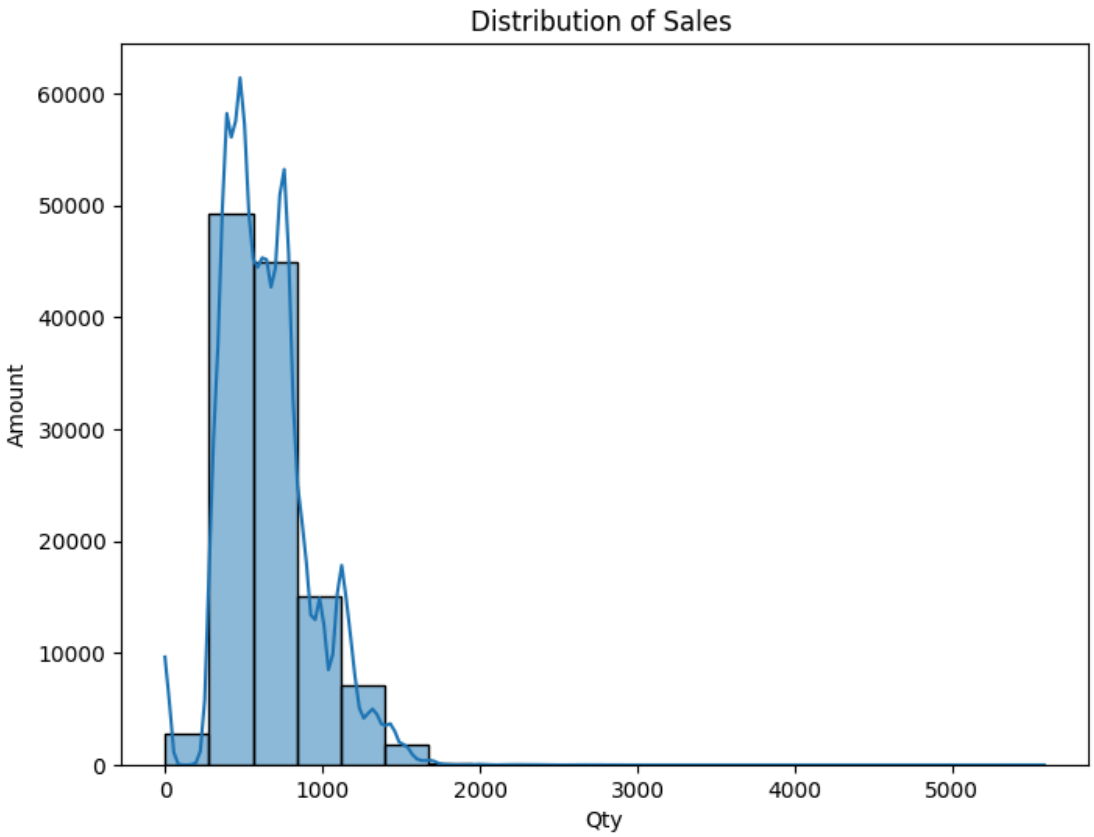
```
2    2.0  404-0687676-7273146  04-30-22                    Shipped  ...       IN Core Free Shipping 2015/04/08 23-48-5-108   True          NaN         NaN
3    3.0  403-9615377-8133951  04-30-22                  Cancelled  ...                                                    NaN  False    Easy Ship         NaN
4    4.0  407-1069790-7240320  04-30-22                    Shipped  ...                                                    NaN  False          NaN         NaN

[5 rows x 24 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128977 entries, 0 to 128976
Data columns (total 24 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   index               128975 non-null  float64
 1   Order ID            128977 non-null  object
 2   Date                128977 non-null  object
 3   Status              128977 non-null  object
 4   Fulfilment          128977 non-null  object
 5   Sales Channel       128977 non-null  object
 6   ship-service-level  128977 non-null  object
 7   Style               128977 non-null  object
 8   SKU                 128975 non-null  object
 9   Category            128975 non-null  object
 10  Size                128975 non-null  object
 11  ASIN                128975 non-null  object
 12  Courier Status      122103 non-null  object
 13  Qty                 128975 non-null  float64
 14  currency            121180 non-null  object
 15  Amount              121180 non-null  float64
 16  ship-city           128942 non-null  object
 17  ship-state          128940 non-null  object
 18  ship-postal-code    128940 non-null  float64
 19  ship-country        128940 non-null  object
 20  promotion-ids       79820 non-null   object
 21  B2B                 128973 non-null  object
 22  fulfilled-by        39275 non-null   object
 23  Unnamed: 22         79923 non-null   object
dtypes: float64(4), object(20)
memory usage: 23.6+ MB
None
```

- Statistics (standard deviation, mean, max, and min)

```
dtypes: float64(4), object(20)
memory usage: 23.6+ MB
None
              index            Qty         Amount  ship-postal-code
count  128975.000000  128975.000000  121180.000000     128940.000000
mean    64487.000000       0.904431     648.561465     463969.927594
std     37232.019822       0.313354     281.211687     191475.956294
min         0.000000       0.000000       0.000000     110001.000000
25%     32243.500000       1.000000     449.000000     382421.000000
50%     64487.000000       1.000000     605.000000     500033.000000
75%     96730.500000       1.000000     788.000000     600024.000000
max    128974.000000      15.000000    5584.000000     989898.000000
                Order ID    Date  Status Fulfilment Sales Channel   ... ship-country                                promotion-ids     B2B fulfilled-by Unnamed:
 22
count               128977  128977  128977     128977         128977  ...       128940                                         79820  128973        39275       79
923
unique              120379      92      14          4              3  ...            1                                          5787       2            1
  1
top     171-5057375-2831560  05-03-22  Shipped     Amazon      Amazon.in  ...           IN  IN Core Free Shipping 2015/04/08 23-48-5-108   False    Easy Ship       Fa
lse
freq                    12    2085   77804      89698         128851  ...       128940                                         46100  128102        39275       79
923
```

- According to the data provided, I made a plot between Qty and the amount that reflects the frequency of sales using Seaborn and Matplotlib

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Amount'], bins=20, kde=True)
plt.title('Distribution of Sales ')
plt.xlabel('Qty')
plt.ylabel('Amount')
plt.show()
```



2- Data Preprocessing

- Handling Missing Values: I chose to drop specific rows to prevent losing important data, so choosing specific rows like Amount and Qty instead of using a function and loops

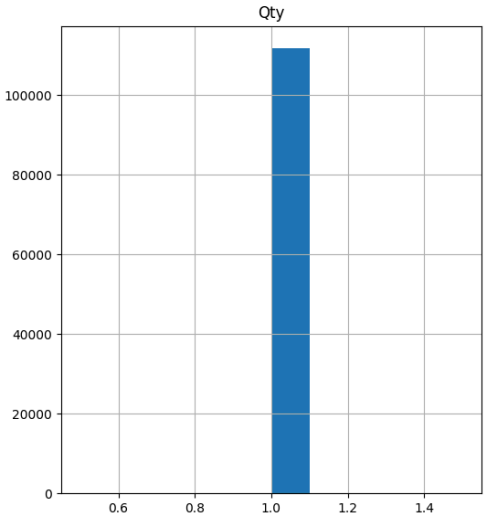
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])

numerical_columns = df.select_dtypes(include=['float64',
'int64']).columns
categorical_columns =
df.select_dtypes(include=['object']).columns

numerical_columns_with_nan = [col for col in numerical_columns
if df[col].isnull().sum() > 0]
print(f"Numerical columns with missing values:
{numerical_columns_with_nan}")

categorical_columns_with_nan = [col for col in
categorical_columns if df[col].isnull().sum() > 0]
print(f"Categorical columns with missing values:
{categorical_columns_with_nan}")

for column in numerical_columns_with_nan:
    df[column] = df[column].fillna(df[column].mean())

for column in categorical_columns_with_nan:
    df[column] = df[column].fillna(df[column].mode()[0])
```

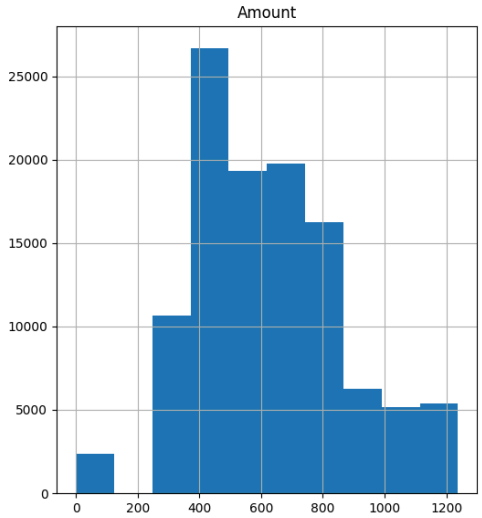
```
df = df.infer_objects(copy=False)


for column in numerical_columns:
    plt.figure(figsize=(10, 5))
    sns.boxplot(data=df, x=column)
    plt.show()

Q1 = df[numerical_columns].quantile(0.25)
Q3 = df[numerical_columns].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[numerical_columns] < (Q1 - 1.5 * IQR)) |
(df[numerical_columns] > (Q3 + 1.5 * IQR))).any(axis=1)]

import os
#making a new directory to save the cleaned file
os.makedirs('New directory for cleaned file ', exist_ok=True)

cleaned_file_path = 'path'
df.to_csv(cleaned_file_path, index=False)

print(f"Cleaned dataset saved to {cleaned_file_path}")
```

3- Data Visualization:

```
# Histograms
numerical_columns = ['Qty', 'Amount']
df[numerical_columns].hist(figsize=(10, 10))
plt.show()

# Heatmap
plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()

df['Month'] = df['Date'].dt.to_period('M')
monthly_sales = df.groupby('Month')['Amount'].sum()
monthly_sales.plot(figsize=(12, 6))
plt.title('Monthly Sales Trends')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.show()

top_products = df.groupby('SKU')['Amount'].sum()
top_products.plot(kind='bar', figsize=(12, 6))
plt.title('Top-Selling Products')
plt.xlabel('Product')
plt.ylabel('Sales')
plt.show()

category_sales = df.groupby('Category')['Amount'].sum()
category_sales.plot(kind='bar', figsize=(12, 6))
plt.title('Sales by Category')
plt.xlabel('Category')
plt.ylabel('Sales')
plt.show()
```

Output:





4- Predictive Modeling:

Building Models to predict order status

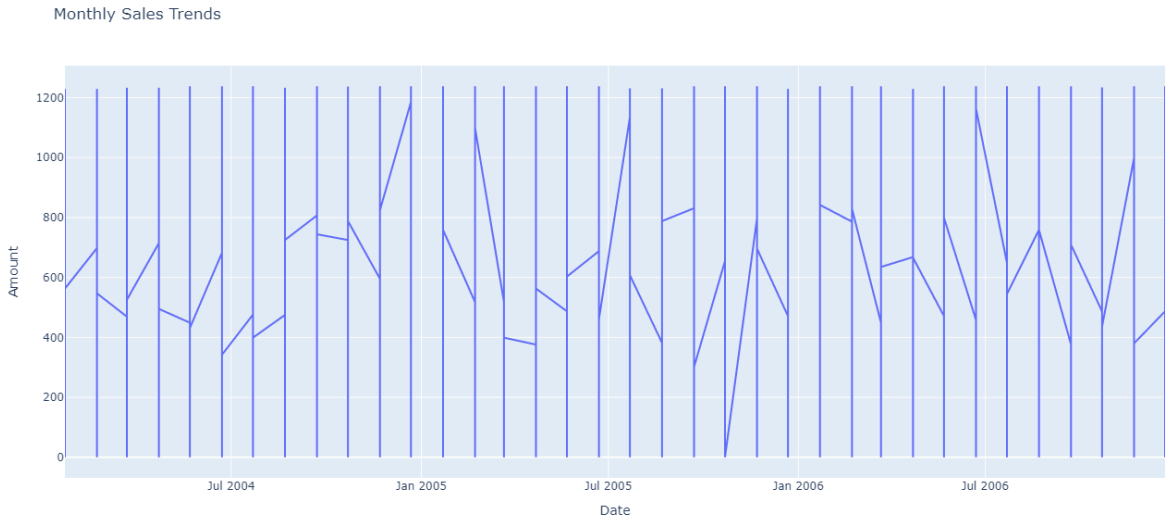
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, confusion_matrix


X = df[['Qty', 'Amount', 'Sales Channel', 'Category']]
y = df['Status']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

lr = LogisticRegression()
```

```
lr.fit(X_train, y_train)

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

rf = RandomForestClassifier()
rf.fit(X_train, y_train)

y_pred_lr = lr.predict(X_test)
y_pred_dt = dt.predict(X_test)
y_pred_rf = rf.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test,
y_pred_lr))
print("Decision Tree Accuracy:", accuracy_score(y_test,
y_pred_dt))
print("Random Forest Accuracy:", accuracy_score(y_test,
y_pred_rf))
```

Model evaluation :

```
# Evaluation
def evaluate_model(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred,
average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    cm = confusion_matrix(y_test, y_pred)
    return accuracy, precision, recall, cm

#Logistic Regression
accuracy_lr, precision_lr, recall_lr, cm_lr =
evaluate_model(y_test, y_pred_lr)
print(f'Logistic Regression - Accuracy: {accuracy_lr},
Precision: {precision_lr}, Recall: {recall_lr}')
print('Confusion Matrix:\n', cm_lr)
```

```
#Decision Tree
accuracy_dt, precision_dt, recall_dt, cm_dt =
evaluate_model(y_test, y_pred_dt)
print(f'Decision Tree - Accuracy: {accuracy_dt}, Precision:
{precision_dt}, Recall: {recall_dt}')
print('Confusion Matrix:\n', cm_dt)

#Random Forest
accuracy_rf, precision_rf, recall_rf, cm_rf =
evaluate_model(y_test, y_pred_rf)
print(f'Random Forest - Accuracy: {accuracy_rf}, Precision:
{precision_rf}, Recall: {recall_rf}')
print('Confusion Matrix:\n', cm_rf)
```



5- dashboard

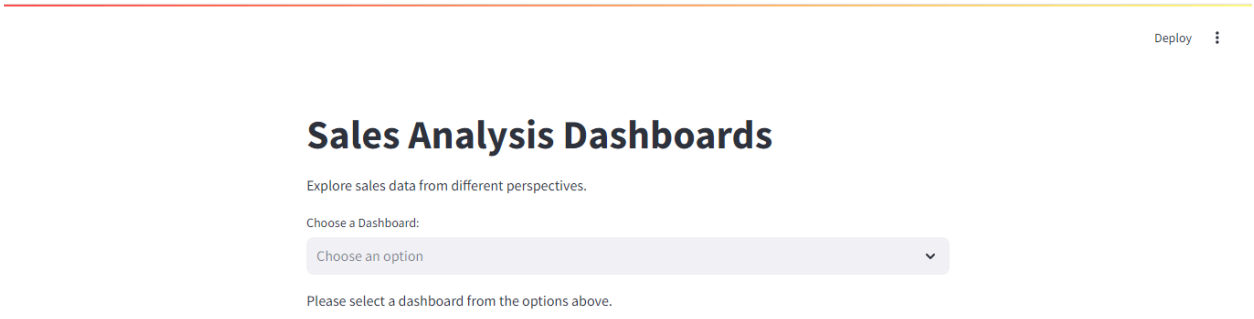
```
import streamlit as st

st.title("Sales Analysis Dashboards")
st.write("Explore sales data from different perspectives.")

selected_dashboard = st.multiselect("Choose a Dashboard:",
("Order Status", "Sales Performance"))

if "Order Status" in selected_dashboard:

  st.header("Order Status Distribution")
  order_status_counts = df['Status'].value_counts()
  st.bar_chart(order_status_counts)


elif "Sales Performance" in selected_dashboard:
  st.header("Sales by Channel")
  sales_by_channel = df.groupby('Sales
Channel')['Amount'].sum().reset_index()
  st.bar_chart(sales_by_channel, x='Sales Channel',
y='Amount')

  st.header("Top Selling Categories")
  top_categories =
df.groupby('Category')['Amount'].sum().nlargest(10).reset_inde
x()
  st.bar_chart(top_categories, x='Category', y='Amount')

if len(selected_dashboard) == 0:
  st.write("Please select a dashboard from the options
above.")
```

Then by entering this command in Powershell:

“Streamlight run code.py “

Note: code.py is the Python file

Dashboard :



Deploy

Choose a Dashboard:

Order Status ×

Order Status Distribution