1. Data Cleaning and Preparation

Python Code

import pandas as pd

# Load the dataset

data = {

'CustomerID': [12345, 12346, 12347, 12348, 12349, 12350, 12351, 12352, 12353, 12354],

'OrderID': [1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010],

'Product': ['T-shirt', 'Bicycle', 'Water bottle', 'Backpack', 'Sunglasses',

'Notebook', 'Tablet', 'Smartphone', 'Camera', 'Headphones'],

'Quantity': [2, 1, 5, 2, 1, 4, 1, 1, 1, 2],

'UnitPrice': [15.00, 200.00, 8.00, 45.00, 25.00, 12.00, 350.00, 700.00, 250.00, 75.00],

'PurchaseDate': [

'2022-07-15 08:30:00', '2022-07-15 09:00:00', '2022-07-16 10:00:00',

'2022-07-16 11:30:00', '2022-07-17 12:45:00', '2022-07-18 13:00:00',

'2022-07-19 14:25:00', '2022-07-20 15:45:00', '2022-07-21 16:00:00',

'2022-07-22 17:35:00'

],

'Country': ['USA', 'Canada', 'UK', 'USA', 'Australia', 'India',

'Germany', 'France', 'Spain', 'Italy']

}

df = pd.DataFrame(data)

# Check for missing or inconsistent data

print("Missing Values:\n", df.isnull().sum())

print("\nDuplicates:\n", df.duplicated().sum())

# Convert PurchaseDate to datetime

df['PurchaseDate'] = pd.to\_datetime(df['PurchaseDate'])

# Ensure proper date formatting

df['PurchaseDate'] = df['PurchaseDate'].dt.strftime('%Y-%m-%d %H:%M:%S')

print("\nCleaned Data:\n", df.head())

. Filtering Data

2.1 Customer and Order Filters

1. Filter transactions from a specific country (e.g., 'USA')

usa\_transactions = df[df['Country'] == 'USA']

print("USA Transactions:\n", usa\_transactions)

1. Extract orders where total spend exceeds $500

df['TotalSpend'] = df['Quantity'] \* df['UnitPrice']

high\_spend\_orders = df[df['TotalSpend'] > 500]

print("High Spend Orders:\n", high\_spend\_orders)

1. Identify customers who purchased more than 3 different products

customer\_product\_count = df.groupby('CustomerID')['Product'].nunique()

customers\_more\_than\_3 = customer\_product\_count[customer\_product\_count > 3].index

print("Customers with >3 Products:\n", customers\_more\_than\_3)

2.2 Time-Based Filters

1. Filter transactions in July 2022

july\_transactions = df[df['PurchaseDate'].str.startswith('2022-07')]

print("July 2022 Transactions:\n", july\_transactions)

1. Extract orders placed during weekends

df['PurchaseDate'] = pd.to\_datetime(df['PurchaseDate'])

weekend\_transactions = df[df['PurchaseDate'].dt.dayofweek >= 5]

print("Weekend Transactions:\n", weekend\_transactions)

1. Identify transactions during specific sales events (e.g., Black Friday or Cyber Monday)

# Assuming Black Friday is 2022-11-25 and Cyber Monday is 2022-11-28

sales\_events = df[df['PurchaseDate'].isin(['2022-11-25', '2022-11-28'])]

print("Sales Event Transactions:\n", sales\_events)

3. Sorting Data

1. Sort transactions by total spend (descending), purchase date (ascending), and product name (alphabetically)

sorted\_df = df.sort\_values(by=['TotalSpend', 'PurchaseDate', 'Product'], ascending=[False, True, True])

print("Sorted Data:\n", sorted\_df)

1. Rank customers based on total spending

customer\_spending = df.groupby('CustomerID')['TotalSpend'].sum().reset\_index()

customer\_spending['Rank'] = customer\_spending['TotalSpend'].rank(ascending=False)

print("Customer Spending Rank:\n", customer\_spending)

4. Aggregated Analysis

4.1 Customer Insights

1. Total amount spent by each customer and top 10 spenders

top\_spenders = df.groupby('CustomerID')['TotalSpend'].sum().nlargest(10)

print("Top 10 Spenders:\n", top\_spenders)

1. Total revenue per country

revenue\_by\_country = df.groupby('Country')['TotalSpend'].sum()

print("Revenue by Country:\n", revenue\_by\_country)

1. Country with the highest average transaction value

avg\_transaction\_by\_country = df.groupby('Country')['TotalSpend'].mean()

highest\_avg\_country = avg\_transaction\_by\_country.idxmax()

print("Country with Highest Avg Transaction Value:\n", highest\_avg\_country)

4.2 Product Analysis

1. Most purchased product and total quantity sold

most\_purchased\_product = df.groupby('Product')['Quantity'].sum().idxmax()

print("Most Purchased Product:\n", most\_purchased\_product)

1. Product that generated the highest revenue

highest\_revenue\_product = df.groupby('Product')['TotalSpend'].sum().idxmax()

print("Highest Revenue Product:\n", highest\_revenue\_product)

1. Top 3 least popular products based on sales quantity

least\_popular\_products = df.groupby('Product')['Quantity'].sum().nsmallest(3)

print("Least Popular Products:\n", least\_popular\_products)

4.3 Time-Based Insights

1. Total revenue per day in July 2022

daily\_revenue = df.groupby(df['PurchaseDate'].dt.date)['TotalSpend'].sum()

print("Daily Revenue in July 2022:\n", daily\_revenue)

1. Peak shopping hours based on the number of transactions

peak\_hours = df['PurchaseDate'].dt.hour.value\_counts().idxmax()

print("Peak Shopping Hour:\n", peak\_hours)

1. Day with the highest total revenue

highest\_revenue\_day = daily\_revenue.idxmax()

print("Day with Highest Revenue:\n", highest\_revenue\_day)

5. Advanced Insights

1. Create a column for total spend per transaction

df['TotalSpend'] = df['Quantity'] \* df['UnitPrice']

1. Identify the top 10% of transactions based on total spend

top\_10\_percent = df[df['TotalSpend'] > df['TotalSpend'].quantile(0.9)]

print("Top 10% Transactions:\n", top\_10\_percent)

1. Analyze purchasing trends by country

# Most popular product in each country

most\_popular\_by\_country = df.groupby(['Country', 'Product'])['Quantity'].sum().reset\_index()

most\_popular\_by\_country = most\_popular\_by\_country.loc[most\_popular\_by\_country.groupby('Country')['Quantity'].idxmax()]

print("Most Popular Product by Country:\n", most\_popular\_by\_country)

# Average order value per country

avg\_order\_value\_by\_country = df.groupby('Country')['TotalSpend'].mean()

print("Average Order Value by Country:\n", avg\_order\_value\_by\_country)

6. Visualization

Python Code

import matplotlib.pyplot as plt

# 1. Total revenue per country (bar chart)

revenue\_by\_country.plot(kind='bar')

plt.title('Total Revenue by Country')

plt.xlabel('Country')

plt.ylabel('Revenue')

plt.show()

# 2. Daily revenue trends (line chart)

daily\_revenue.plot(kind='line')

plt.title('Daily Revenue Trends in July 2022')

plt.xlabel('Date')

plt.ylabel('Revenue')

plt.show()

# 3. Distribution of products sold (pie chart)

product\_distribution = df.groupby('Product')['Quantity'].sum()

product\_distribution.plot(kind='pie', autopct='%1.1f%%')

plt.title('Distribution of Products Sold')

plt.show()