You have been given the role of a business analyst for an E-Commerce company and have been asked to prepare a basic report on the data. Follow the steps below for preparation of the report.

Before you start analysing the data, it is always a good practice to see the size of the data, its features and feature types. If the data set is big, it is not possible to print out all the records.

Note: Use the markdown feature of Python to explain your answer.

Load the necessary libraries. Import and load the dataset with a name ECom_Data .

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
pd.options.display.max_rows = 10

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Get the Data

ECom_Data=pd.read_csv("E-Commerce.csv")
```

We have read the data and stored the data in "ECom_Data" variable

Q 1. To get familiar with the data

a) Print out the first 10 and the last 10 records of the data. (2 marks)

b) How many rows and columns are present in the dataset? Use any two different methods to extract this information. (2 + 2 + 2 marks)

c) How many object data types are there? (1 mark)

d) Is there any Boolean data type? (1 mark)

Note: Use the markdown feature of Python to explain your answer.

Ans 1 a)

To print the first 10 records, we can use head(10) method and for printing last 10 records, we can use tail(10) method. By default head() & tail() methods print the first & last 5 records repectively.

In [3]:
```
# First 10 records
print("First 10 records of ECom_Data:")
print("---------------------------------\n")
ECom_Data.head(10)
```

```
First 10 records of ECom_Data:
---------------------------------
```

Out[3]:

	Customer_uniq_id	Region	Order_Date	Expected_Delivery_Date	Delive
0	e71017e224688489edfe856f2308806d	East	24-10-2021	25-10-2021	25
1	6286847ee2da18f587503db49511c539	East	24-10-2021	25-10-2021	25
2	0686fec9b70e5039583a38119ca0c835	West	24-10-2021	25-10-2021	25
3	ea2406dc597bee2abb6b867fa668501f	West	24-10-2021	25-10-2021	25
4	5935ed077915347dc695744df68c565c	East	03-09-2021	04-09-2021	04
5	89fcdddaad50084e395d0928a7426afe	East	03-09-2021	04-09-2021	04
6	b9b183aa18d3a721d2ac23e7184525b0	East	03-09-2021	04-09-2021	04
7	dee64864c0419bec80fbbb94d19bc40d	East	03-09-2021	04-09-2021	04
8	595f55f2c1293f07ea9ec9fa2bb39f46	East	03-09-2021	04-09-2021	04
9	547b6585272473ae006bcfbdb47b6ae0	West	26-08-2021	27-08-2021	27

In [4]:
```
# Last 10 records
print("\nLast 10 records of ECom_Data:")
print("---------------------------------\n")
ECom_Data.tail(10)
```

```
Last 10 records of ECom_Data:
---------------------------------
```

Out[4]:

	Customer_uniq_id	Region	Order_Date	Expected_Delivery_Date	De
8896	4db03bc4ccbe216cf151b2f2b904ba3f	East	01-12-2020	04-12-2020	
8897	4d4d6aeb13fa253499d0dd45a5abd87e	West	01-12-2020	04-12-2020	
8898	ea19e7ef703293d3f6c799ca9db4642d	West	01-12-2020	04-12-2020	
8899	19a53a958992fb575acffb5d41e7ef9e	East	01-12-2020	04-12-2020	
8900	09abab80c8dfdc6f268e0a6f05a0be11	North	01-12-2020	04-12-2020	
8901	90d30478255e23621e8929ed15c2f6e4	South	01-12-2020	04-12-2020	
8902	20a73e3f41490a73ceeba5f17658db8f	West	01-12-2020	04-12-2020	
8903	5c1554cd45f9d538c2c6947dbdd59c75	East	01-12-2020	04-12-2020	
8904	6b737a4deca1ed0e56c179e66036e994	West	01-12-2020	04-12-2020	
8905	a5235ac28d3d5487f54025f9d6b57433	North	01-12-2020	04-12-2020	

Ans 1 b)

There are **8906** rows and **17** columns are present in the ECom_Data.

Method 1 - We can use the **shape** attribute to give us the dimensions of the dataset.

In [5]:
```
# Method 1 - Using shape attribute
shape = ECom_Data.shape

print(f"Number of rows - {shape[0]} \nNumber of columns - {shape[1]}")
```

```
Number of rows - 8906
Number of columns - 17
```

Method 2 - We can use **len()** method to calculate the length of rows which is the number of rows. And for number of columns, we pass ECom_Data.columns inside the len() method.

In [6]:
```
# Method 2 - Using len() method
num_rows = len(ECom_Data)

columns = ECom_Data.columns
num_columns = len(columns)

print(f"Number of rows - {num_rows} \nNumber of columns - {num_columns}")
```

```
Number of rows - 8906
Number of columns - 17
```

Additionally we can use **info()** method which gives us the number of rows and columns

In [7]:
```
# Additional Method - Using info() method
ECom_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8906 entries, 0 to 8905
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Customer_uniq_id             8906 non-null   object
 1   Region                       8906 non-null   object
 2   Order_Date                   8906 non-null   object
 3   Expected_Delivery_Date       8906 non-null   object
 4   Delivered_Date               8906 non-null   object
 5   product_name                 8906 non-null   object
 6   product_main_category        8906 non-null   object
 7   product_subcategory          8906 non-null   object
 8   product_category_filter      8906 non-null   object
 9   product_category_subfilter   8906 non-null   object
 10  product_unique ID            8906 non-null   object
 11  retail_price                 8906 non-null   int64
 12  discounted_price             8906 non-null   int64
 13  product_rating               8906 non-null   float64
 14  Brand                        8906 non-null   object
 15  product_specifications       8906 non-null   object
 16  description                  8906 non-null   object
dtypes: float64(1), int64(2), object(14)
memory usage: 1.2+ MB
```

Ans 1 c)

There are 14 columns of object data type. We can find this from info() method or we can use the below method to find the number of object data types.

In [8]:
```
object_type = ECom_Data.select_dtypes(include='object')
print(f'Number of Object Data Type columns - {len(object_type.columns)}')
```

```
Number of Object Data Type columns - 14
```

Ans 1 d)

There is No boolean data type in the dataframe. We can verify that from info() method or we can use the below method.

In [9]:
```
print('Is Boolean Datatype present in the dataframe? ','bool' in ECom_Data.dtypes.v
```

```
Is Boolean Datatype present in the dataframe?  False
```

Once you are familiar with the data, you may decide that not all features are of use to you and you may want to delete the non-informative features (columns)

Q 2. Eliminating the non-informative columns.

a) Drop the columns product_specifications and description. (2 marks)

b) Which method or function is used to permanently delete the columns mentioned in part (a)? Write the code explicitly (2 marks)

Note: Use the markdown feature of Python to explain your answer.

Ans 2 a)

We can use **drop()** method to drop the columns and axis=1 as we are dropping along column wise. By default, axis=0 indicating row wise

In [10]:
```
# Dropping product_specifications and description
before_dropping = len(ECom_Data.columns)
new_df = ECom_Data.drop(['product_specifications','description'],axis=1)
after_dropping = len(new_df.columns)

print(f'No. of columns before dropping - {before_dropping} \nNo. of columns after d
```

```
No. of columns before dropping - 17
No. of columns after dropping - 15
```

In [11]:
```
ECom_Data.columns
```

Out[11]:
```
Index(['Customer_uniq_id', 'Region', 'Order_Date', 'Expected_Delivery_Date',
       'Delivered_Date', 'product_name', 'product_main_category',
       'product_subcategory', 'product_category_filter',
       'product_category_subfilter', 'product_unique ID', 'retail_price',
       'discounted_price', 'product_rating', 'Brand', 'product_specifications',
       'description'],
      dtype='object')
```

Ans 2 b)

From 2 a) we can clearly see that the dropping of product_specifications' & 'description' columns from **ECom_Data** are not permanent. To make permanent changes, we need to use '**inplace=True**'

In [12]:
```
ECom_Data.drop(['product_specifications','description'],axis=1,inplace = True)
```

The next steps in this project involves summarization of data at various levels and visualization. Apparently,

such simple steps are very useful to get an overall sense of the data.

Q 3. Here we summarize the data at Brand level.

a) How many unique Brand are there? (2 marks)

b) Note that each Brand contains multiple products. Show the average product_rating within each Brand (2 marks)

Ans 3 a)

We can display the number of unique brands as shown below using **unique** and **nunique** methods

Number of unique Brands - **2484**

In [13]:
```
# Unique Method
unique_brand = len(ECom_Data['Brand'].unique())
print(f'Number of unique Brands (unique method) - {unique_brand}')

# NUnique Method
nunique_brand = ECom_Data['Brand'].nunique()
print(f'Number of unique Brands (nunique method) - {nunique_brand}')
```

```
Number of unique Brands (unique method) - 2484
Number of unique Brands (nunique method) - 2484
```

In [14]:
```
ECom_Data.head(2)
```

Out[14]:

	Customer_uniq_id	Region	Order_Date	Expected_Delivery_Date	Deliver
0	e71017e224688489edfe856f2308806d	East	24-10-2021	25-10-2021	25
1	6286847ee2da18f587503db49511c539	East	24-10-2021	25-10-2021	25

Ans 3 b)

We can use **groupby()** method to group the **product_rating** by **Brand** column & using **mean()** method to find the average product rating

In [15]:
```
avg_rating_per_brand = ECom_Data.groupby(['Brand'])['product_rating'].mean().round(
avg_rating_per_brand
```

Out[15]:
```
Brand
1OAK           1.50
3A AUTOCARE    3.27
3D MAT         3.00
3KFACTORY      2.00
4D             3.60
               ...
ZORDEN         4.00
ZOSIGN         3.40
ZRESTHA        1.00
ZYXEL          3.33
 TARKAN        5.00
Name: product_rating, Length: 2484, dtype: float64
```

Q 4. Next we study the main categories of the products.

a) Create an appropriate plot to show the count of items ordered for each product_main_category. (6 marks).

Hint: Create a bar chart titled "Product Category type" where product_main_category are on x-axis and counts are on y-axis.

Note: Both axis labels, i.e. the names of the product_main_category and counts must be clearly legible.

b) From the plot identify for which two product_main_category(s) maximum and minimum orders were placed. (2 marks)

c) Write code to print out the top 5 product_main_category(s) in descending order? (3 marks

Ans 4 a)

we can use **countplot** from seaborn library to show the count of items ordered for each product_main_category.

In [16]:
```
#Setting the Figure size
plt.figure(figsize=(12,5))
```

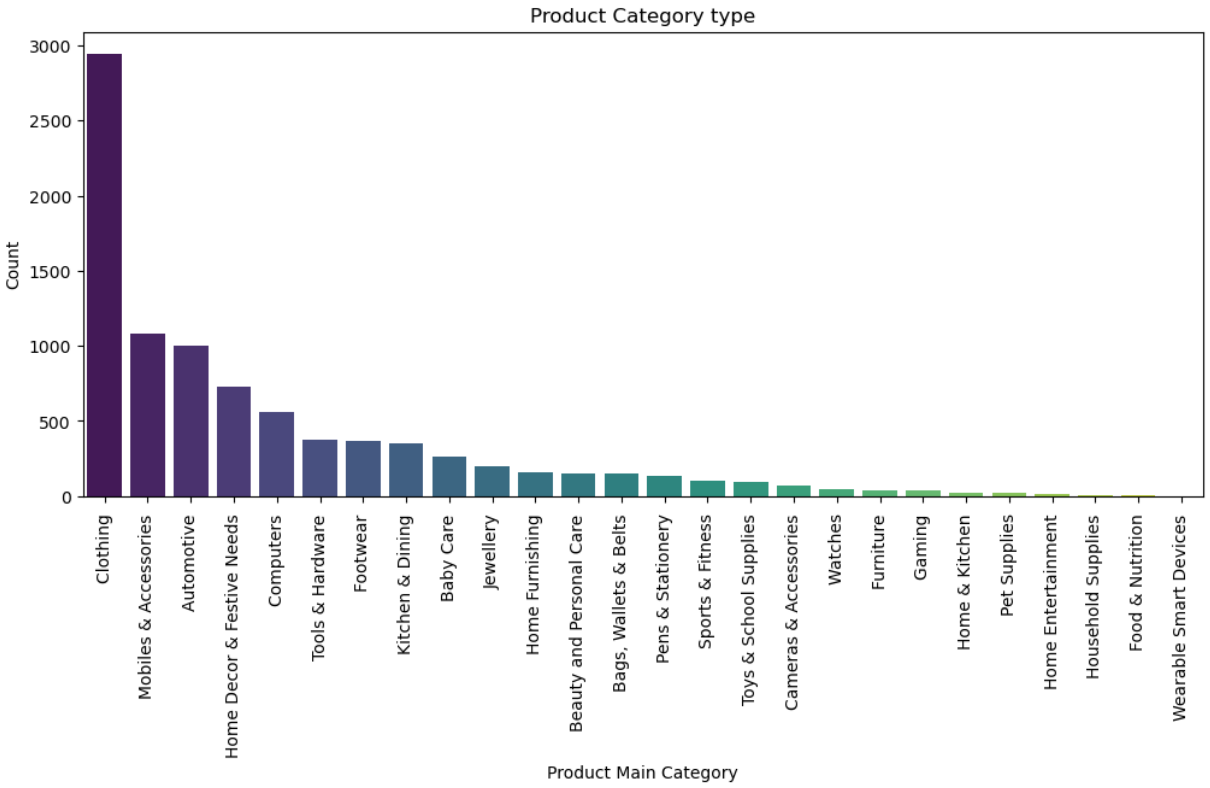
```
# Rotating the x ticks by 90 deg for clarity
plt.xticks(rotation=90)

# Setting the title of chart
plt.title('Product Category type')

# Ordering the product main category and taking the index
order_desc = ECom_Data['product_main_category'].value_counts().index

# Plotting
sns.countplot(data=ECom_Data,x='product_main_category',palette='viridis',order=orde

# Setting x & y Label
plt.xlabel('Product Main Category');
plt.ylabel('Count');
```



Ans 4 b)

From the above count(bar) plot, we can clearly see that the **Maximum** orders were placed for **Clothing** category and **Minimum** orders were placed for **Wearlable Smart Devices** category

Ans 4 c)

We can use the **value_counts()** method to show top product categories

In [17]:
```
order_desc = ECom_Data['product_main_category'].value_counts()
order_desc[:5]
```

Out[17]:
```
product_main_category
Clothing                        2943
Mobiles & Accessories           1084
Automotive                      1001
Home Decor & Festive Needs       727
Computers                        558
Name: count, dtype: int64
```

In E-commerce, both the retailers (here brands) and the company have to make profit to sustain in the business. The E-Commerce company has the following rule for computing their own revenue:

The company charges

(i) 25% on the orders having final price (discounted price) greater than 600

(ii) 15% on the orders having final price (discounted price) greater than 350 but less than or equal to 600

(iii) 10% on the orders having final price (discounted price) greater than 100 but less than or equal to 350

(iv) Otherwise, 5% on the final price (discounted price)

Q 5. Find the Total Revenue generated by the E-Commerce company over all orders placed. (6 marks)

Hint: Calculate revenue of E-commerce company using the conditions mentioned above and then do the total (sum) of all to get total Revenue.

Ans 5

- We created the **calculate_revenue** function to calculate the revenue for the company after applying the percentages

- Then we applied the calculate_revenue function to the **discounted_price** column and saving the values to the new **Revenue** column

- Then applied **sum()** method to the **Revenue** column to find the sum of the column which is the total revenue to the company

The Total Revenue to the company is **2217486.85**

In [18]:
```
ECom_Data['discounted_price']
```

Out[18]:
```
0         699
1         275
2         275
3         837
4         699
        ...
8901     1490
8902     1300
8903    24995
8904     1000
8905     1350
Name: discounted_price, Length: 8906, dtype: int64
```

In [19]:
```python
# Creating the calculate_revenue function to calculate the revenue for the company
def calculate_revenue(price):
    sum = 0
    if (price > 600):
        sum += price * 0.25
    elif (price > 350) and (price <= 600):
        sum += price * 0.15
    elif (price > 100) and (price <= 350):
        sum += price * 0.10
    else:
        sum += price * 0.05

    return sum
```

In [20]:
```python
ECom_Data['Revenue'] = ECom_Data['discounted_price'].apply(calculate_revenue)
ECom_Data[['discounted_price','Revenue']].head()
```

Out[20]:

	discounted_price	Revenue
0	699	174.75
1	275	27.50
2	275	27.50
3	837	209.25
4	699	174.75

In [21]:
```python
total_revenue = ECom_Data['Revenue'].sum()
print(f'The Total Revenue to the company is {total_revenue}')
```

```
The Total Revenue to the company is 2217486.85
```

Now you need to find the revenue for each retailer (Brand)

Q6.Calculate the total BrandRevenue and list the top 10 Brand having maximum revenue in

descending order (6 marks)

Hint: Total BrandRevenue is BrandRevenue which is generated after all the deductions. Also, BrandRevenue is different from the E-Commerce company's revenue.

Brand Revenue is the discounted price minus the Revenue (generated using the conditions given above Q5)

Ans 6

- We created the **calculate_brand_revenue** function to calculate the brand revenue

- Then we applied the **calculate_brand_revenue** function to the **discounted_price & Revenue** columns and saving the values to the new **Brand_Revenue** column

- Then applied **sum()** method to the **Brand_Revenue** column to find the sum of the column which is the total brand revenue

- We then grouped by Brands to show the top 10 brands having high revenue

The Total Brand Revenue is **7522992.15**

In [22]:
```
ECom_Data.columns
```

Out[22]:
```
Index(['Customer_uniq_id', 'Region', 'Order_Date', 'Expected_Delivery_Date',
       'Delivered_Date', 'product_name', 'product_main_category',
       'product_subcategory', 'product_category_filter',
       'product_category_subfilter', 'product_unique ID', 'retail_price',
       'discounted_price', 'product_rating', 'Brand', 'Revenue'],
      dtype='object')
```

In [23]:
```
def calculate_brand_revenue(col_name):
    discount_price = col_name['discounted_price']
    company_revenue = col_name['Revenue']
    brand_revenue = discount_price - company_revenue
    return brand_revenue
```

In [24]:
```
ECom_Data['Brand_Revenue'] = ECom_Data[['discounted_price', 'Revenue']].apply(calcu
```

In [25]:
```
total_brand_revenue = ECom_Data['Brand_Revenue'].sum()
print(f'The Total Brand Revenue is {total_brand_revenue}')
```

```
The Total Brand Revenue is 7522992.15
```

In [26]:
```
ECom_Data.groupby(['Brand'])[['discounted_price','Revenue','Brand_Revenue']].sum().
```

Out[26]:

Brand	discounted_price	Revenue	Brand_Revenue
ALLURE AUTO	664619	166154.75	498464.25
GAGA	316520	79130.00	237390.00
SLIM	272177	60114.40	212062.60
DAILYOBJECTS	242640	60660.00	181980.00
DIVINITI	190820	47705.00	143115.00
THELOSTPUPPY	149750	22462.50	127287.50
REGULAR	152979	26442.50	126536.50
ENTHOPIA	164195	41048.75	123146.25
ASUS	132322	33080.50	99241.50
SPRINGWEL	118638	29659.50	88978.50

Let us now investigate multiple features for each product to determine any pattern.

Q 7. Compare prices for each product.

a) Draw boxplots of retail_price & discounted_price. (3 marks)

b) Are there any outliers? (Yes/No) (1mark)

c) Create a scatterplot of retail_price (x-axis) and discounted_price (y-axis) (3 marks)

Ans 7 a)

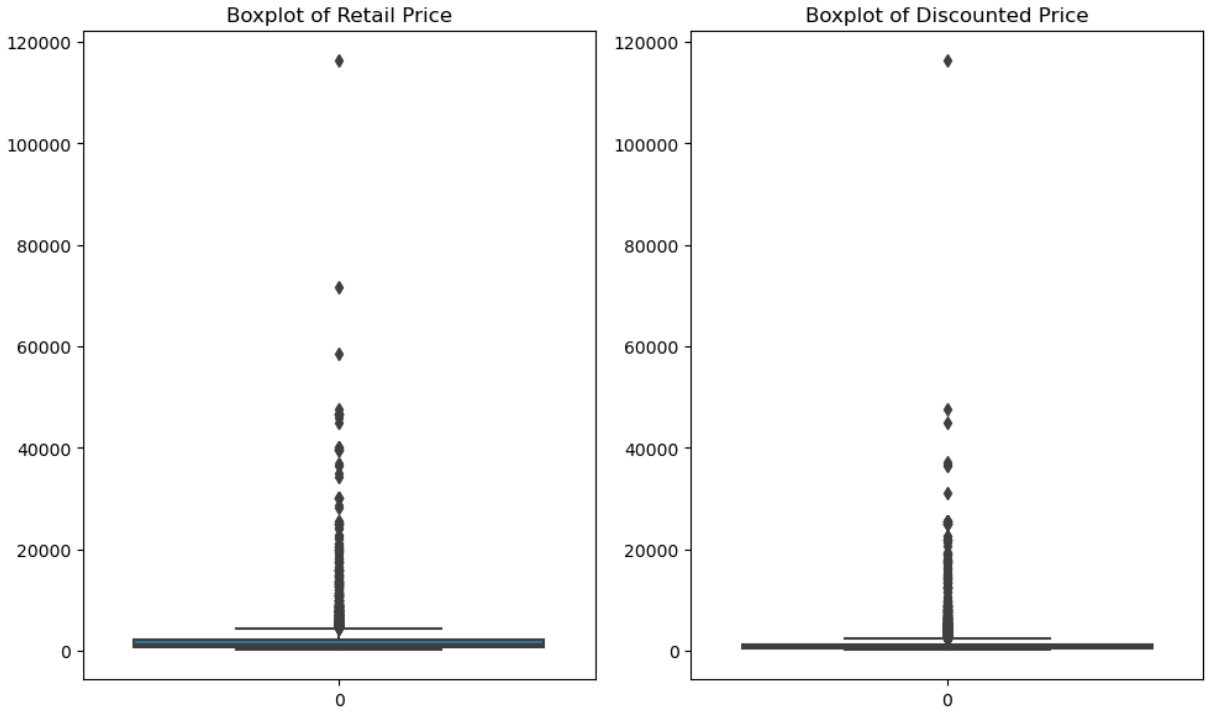
We used seaborn boxplot for the retail price and discounted price. Used subplots to show them in the same row

In [40]:
```
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
sns.boxplot(ECom_Data['retail_price'])
plt.title('Boxplot of Retail Price')

plt.subplot(1, 2, 2)
sns.boxplot(ECom_Data['discounted_price'])
```

```
plt.title('Boxplot of Discounted Price')

plt.tight_layout()
```



Ans 7 b)

Yes, There are Outliers. We can clearly see them from the above plots

Ans 7 c)

We used searborn scatterplot to show **Retail Price vs Discounted Price**

In [28]:
```
plt.figure(figsize=(5, 5))
plt.title('Retail Price vs Discounted Price')
plt.xlabel('Retail Price')
plt.ylabel('Discounted Price')
sns.scatterplot(data=ECom_Data,x='retail_price',y='discounted_price',color='red');
```

We can see that there is a positive correlation between Retail Price and Discounted Price

The next steps will enable to study brand-level information.

Q 8. Create a new dataframe to include the Brand specific information as stated:

i. total number of orders placed per Brand

ii. total retail_price per Brand

iii. total discounted_price per Brand, and

iv. total BrandRevenue generated per Brand.

Also, draw a pairplot using these four features. (6 marks)

Ans 8

We performed groupby operation on the 'Brand' column of the E-commerce data and then aggregated the data based on the following metrics:

- Count of unique customers ('Customer_uniq_id')

- Sum of retail prices ('retail_price')

- Sum of discounted prices ('discounted_price')

- Sum of brand revenue ('Brand_Revenue')

The resulting dataframe is then assigned new column names to reflect the aggregated metrics:

- 'Total_Orders' for the count of unique customers

- 'Total_Retail_Price' for the sum of retail prices

- 'Total_Discounted_Price' for the sum of discounted prices

- 'Total_Brand_Revenue' for the sum of ebrand revenu

In [29]:
```
brand_df = ECom_Data.groupby('Brand').agg({'Customer_uniq_id': 'count', 'retail_pri
brand_df.columns = ['Total_Orders', 'Total_Retail_Price', 'Total_Discounted_Price',

brand_df
```

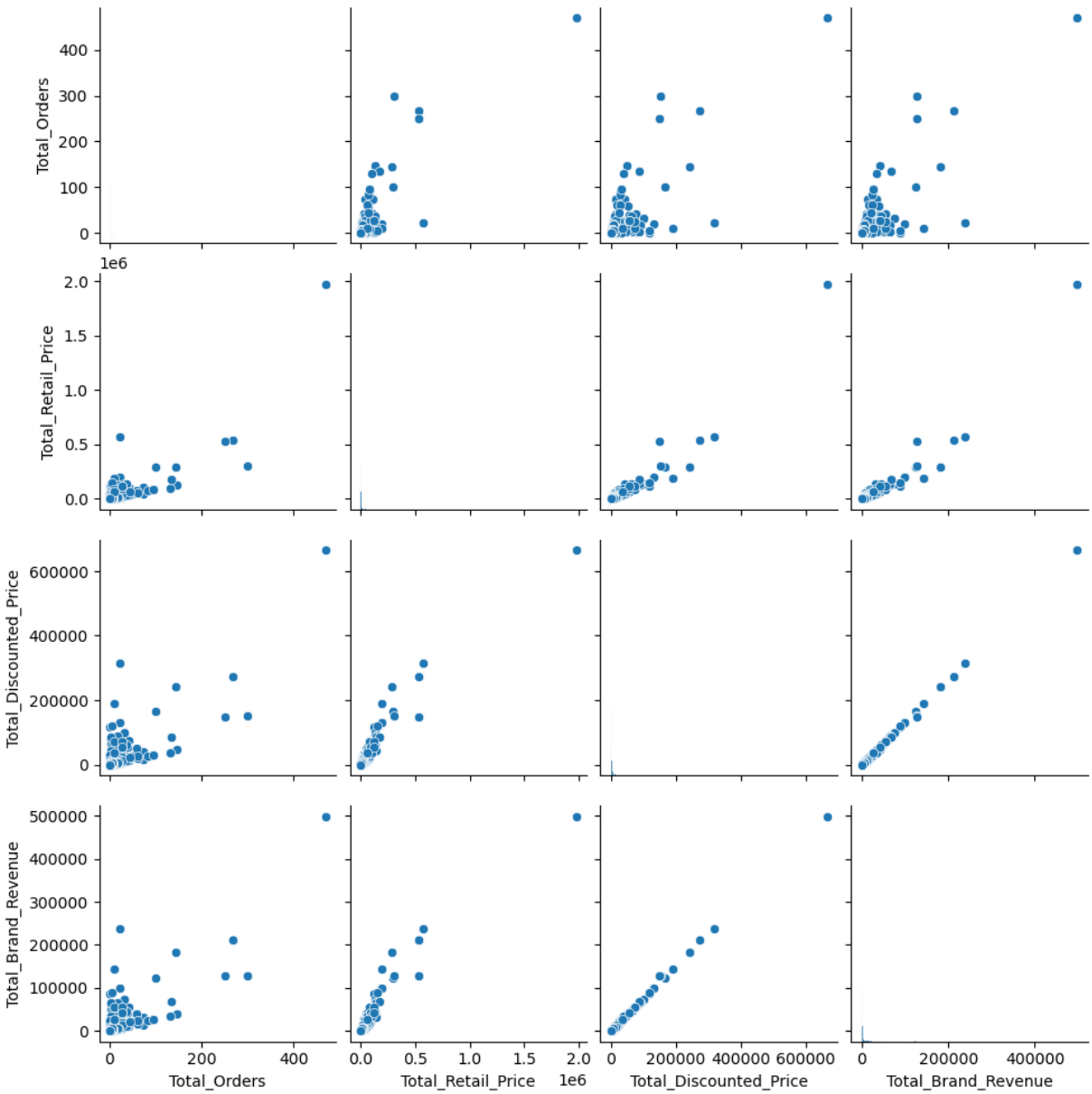
Out[29]:

Brand	Total_Orders	Total_Retail_Price	Total_Discounted_Price	Total_Brand_Revenue
1OAK	2	1698	1274	1015.40
3A AUTOCARE	41	107059	74134	55647.90
3D MAT	1	7250	6999	5249.25
3KFACTORY	1	399	174	156.60
4D	5	17500	7948	5961.00
...	...	...	...	...
ZORDEN	1	3999	2799	2099.25
ZOSIGN	5	6995	6995	5246.25
ZRESTHA	1	1999	899	674.25
ZYXEL	9	64742	35392	26544.00
TARKAN	1	1999	349	314.10

2484 rows × 4 columns

Then we used seaborn paiplot on the four columns from the above.

In [30]:
```
sns.pairplot(brand_df);
```



The E-Commerce company operate in multiple regions. It is important to understand its performance in each region.

Q 9.Compare performance regionwise

a) Draw a lineplot for the monthly Revenue of E-Commerce Company for each region separately. (4 marks)

b) Identify the best and the worst performing months for each region. (2 marks)

Note: Only those days with actual orders(Order_Date) placed are present in the dataset. Assuming there were no orders on other days. Also, show Month and Year on x-axis.

Ans 9 a)

The code first converts the 'Order_Date' column in the ECom_Data to datetime format using pd.to_datetime.

Then, it sorts the data based on the 'Order_Date' in ascending order.

Finally, it creates a new column 'Month_Year' by extracting the month and year from the 'Order_Date' and formatting it as 'Month Year'.

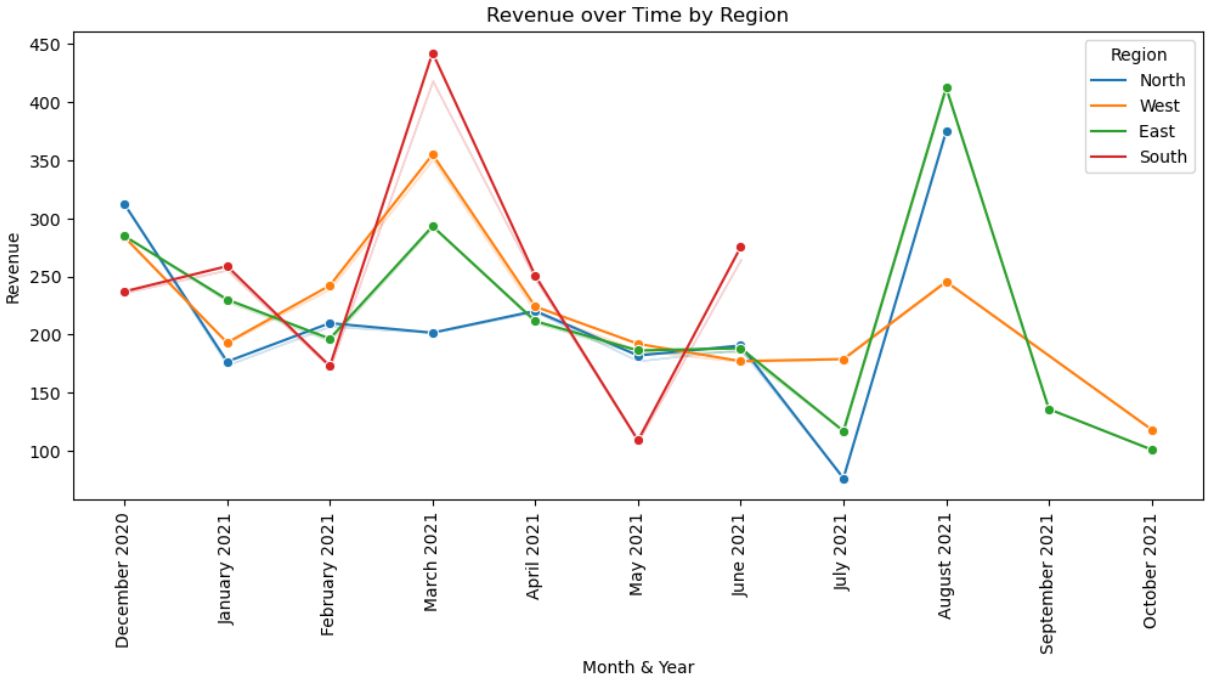
In [31]:
```
ECom_Data['Order_Date'] = pd.to_datetime(ECom_Data['Order_Date'], dayfirst=True, fo
ECom_Data = ECom_Data.sort_values('Order_Date')
ECom_Data['Month_Year'] = ECom_Data['Order_Date'].dt.strftime('%B %Y')
ECom_Data.head(1)
```

Out[31]:

	Customer_uniq_id	Region	Order_Date	Expected_Delivery_Date	Del
8905	a5235ac28d3d5487f54025f9d6b57433	North	2020-12-01	04-12-2020	

We used seaborn lineplot to plot the company revenue over time with region as hue

In [32]:
```
plt.figure(figsize=(12,5))
plt.xticks(rotation=90)
plt.xlabel('Month & Year')
plt.ylabel('Revenue')
plt.title('Revenue over Time by Region')
sns.lineplot(data=ECom_Data,x='Month_Year',y='Revenue',ci=False,hue='Region',marker
```



Ans 9 b)

Best Performing Months for each region:

---> North - August 2021

---> East - August 2021

---> West - March 2021

---> South - March 2021

Worst Performing Months for each region:

---> North - July 2021

---> East - October 2021

---> West - October 2021

---> South - May 2021

Congratulations! You have learnt how to approach a complex data and extract information out of it.