E-COMMERCE TRANSACTION ANALYSIS

Data Analyst Project by Farah Dibah

01
BACKGROUND AND OBJECTIVES

02
DATA SOURCE AND DEFINITION

03
DATA CLEANSING

04
EXPLORATORY DATA ANALYSIS

05
RFM ANALYSIS

Objective Statement

The goals of this project is to **find out** the sales performance based on revenue and total number of sales and to **identify** customer segmentation to increase purchases and customer loyalty.

Data Source and Data Definition

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.
(Source: UCI Machine Learning Online Retail)

InvoiceNo: a 6-digit integral number uniquely assigned to each transaction
StockCode: a 5-digit integral number uniquely assigned to each distinct product
Description: product name
Quantity: the quantities of each product (item) per transaction
InvoiceDate: the day and time when each transaction was generated
UnitPrice: product price per unit
CustomerID: a 5-digit integral number uniquely assigned to each customer
Country: the name of the country where each customer resides

Data Cleansing

Display The Data Type

```
[3] # data type
    df.dtypes

InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

The data type in "InvoiceDate" has wrong data type. It needs to be changed to date time type.

Change Data Type

```
[12] # change the data type of invoice date to date time
     df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

     # review the data type
     df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   541909 non-null  object
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 33.1+ MB
```

Data Cleansing

Display Missing Value

```
[7] # method to display the missing value in each columns of the data frame
    df.isna().sum()
```

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

It shows that there are missing values in "Description" and "CustomerID". It will be replaced with predetermined text.

Replace Missing Value

```
[11] # Function to replace NaN in CustomerID with custom text
     def fill_customer_id(row):
         if pd.isna(row['CustomerID']):
             return f"Customer of Invoice № {row['InvoiceNo']}"
         else:
             return row['CustomerID']

     # Apply the function to each row
     df['CustomerID'] = df.apply(fill_customer_id, axis=1)

     # Show how many missing values remain in each column of the DataFrame
     df.isna().sum()
```

```
InvoiceNo         0
StockCode         0
Description    1454
Quantity          0
InvoiceDate       0
UnitPrice         0
CustomerID        0
Country           0
dtype: int64
```

Replace the missing value in "CustomerID" using "Customer of Invoice No (Number of Invoice)"

```
[9] # replace missing value with predetermined text
    df['Description'].fillna('No description available')
```

```
0          WHITE HANGING HEART T-LIGHT HOLDER
1                         WHITE METAL LANTERN
2              CREAM CUPID HEARTS COAT HANGER
3         KNITTED UNION FLAG HOT WATER BOTTLE
4              RED WOOLLY HOTTIE WHITE HEART.
                          ...
541904            PACK OF 20 SPACEBOY NAPKINS
541905           CHILDREN'S APRON DOLLY GIRL
541906          CHILDRENS CUTLERY DOLLY GIRL
541907         CHILDRENS CUTLERY CIRCUS PARADE
541908           BAKING SET 9 PIECE RETROSPOT
Name: Description, Length: 541909, dtype: object
```

Replace the missing value in "Description" using "No Description Available"

Data Cleansing

Calculate Null Value

```
[5] # show column contains at least one value of zero
    df.columns[df.isin([0]).any()]

    Index(['UnitPrice'], dtype='object')
```

It shows that there are zero value in 'UniPrice' Column and will be replaced with mean of data.

Replace Null Value

```
[8] # calculate the mean of unit price
    unit_price_mean = sum(df['UnitPrice'])/len(df['UnitPrice'])

    # replace the 0 value with mean of data
    df['UnitPrice'] = df['UnitPrice'].replace(0, unit_price_mean)
```

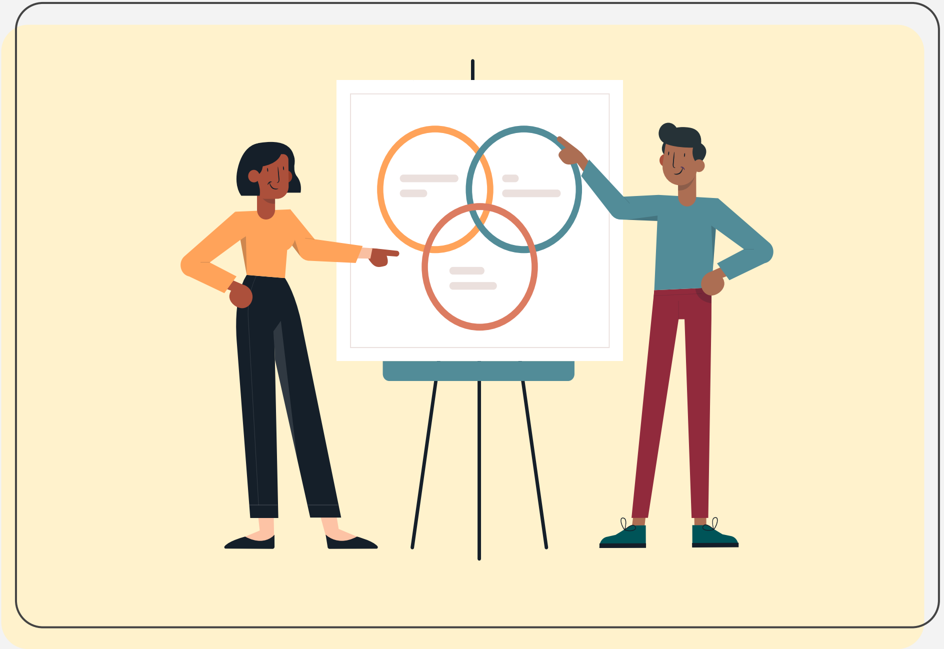
Remove Negative Value

```
[26] # Delete the negative values in quantity column
     df["Quantity"] = df["Quantity"].mask(df["Quantity"] < 0, np.nan)
```

```
[29] # Delete the negative values in unit price column
     df["UnitPrice"] = df["UnitPrice"].mask(df["UnitPrice"] < 0, np.nan)
```

```
[30] # Delete the negative values in total price column
     df["TotalPrice"] = df["TotalPrice"].mask(df["TotalPrice"] < 0, np.nan)
```

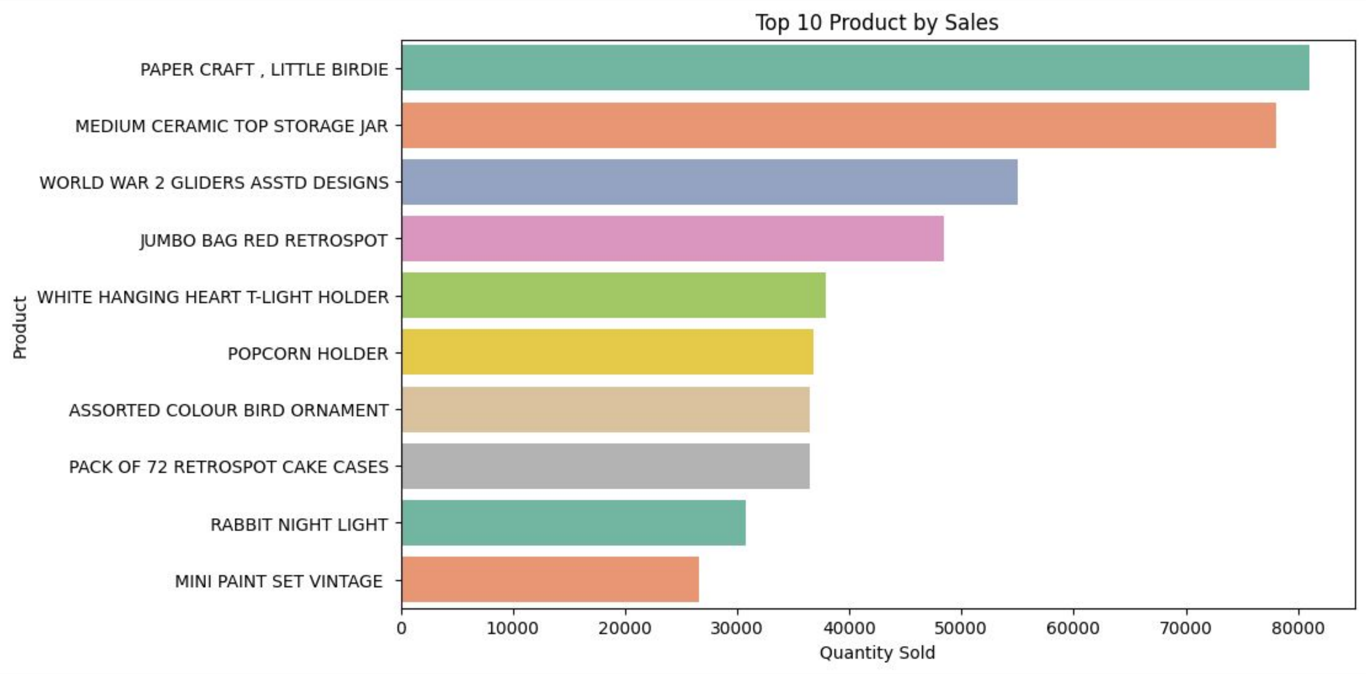
There are any negatives values in 'Quantity', 'Unit Price', 'Total Price'. It should not be negatives and will be removed from the data.



Exploratory Data Analysis

Best Product by Sales

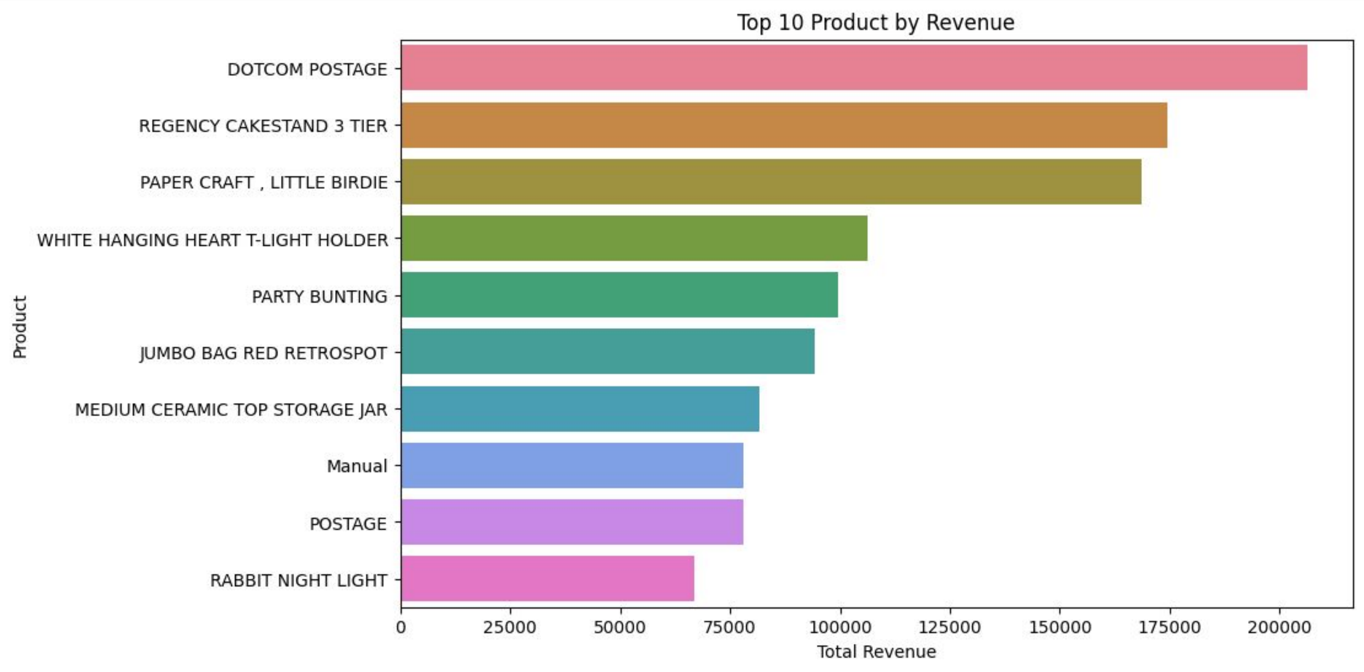
- **Top 10 Product by Sales**



The chart shows that the most sold product by volume of sales is "Paper Craft, Little Birdie" with total sales is 80.995 unit.

Best Product by Revenue

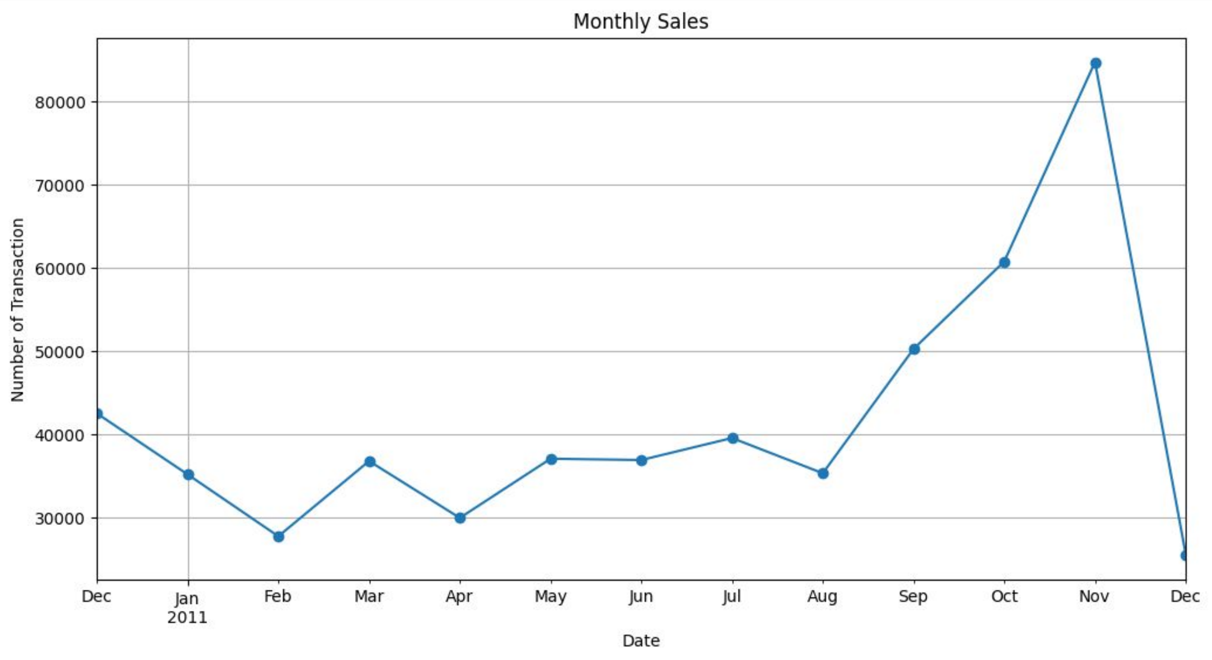
- **Top 10 Product Sales by Revenue**



The chart shows that the most sold product by revenue is "Dotcom Postage" with total revenue is 206.257 euro dollar.

Sales Performance by Month

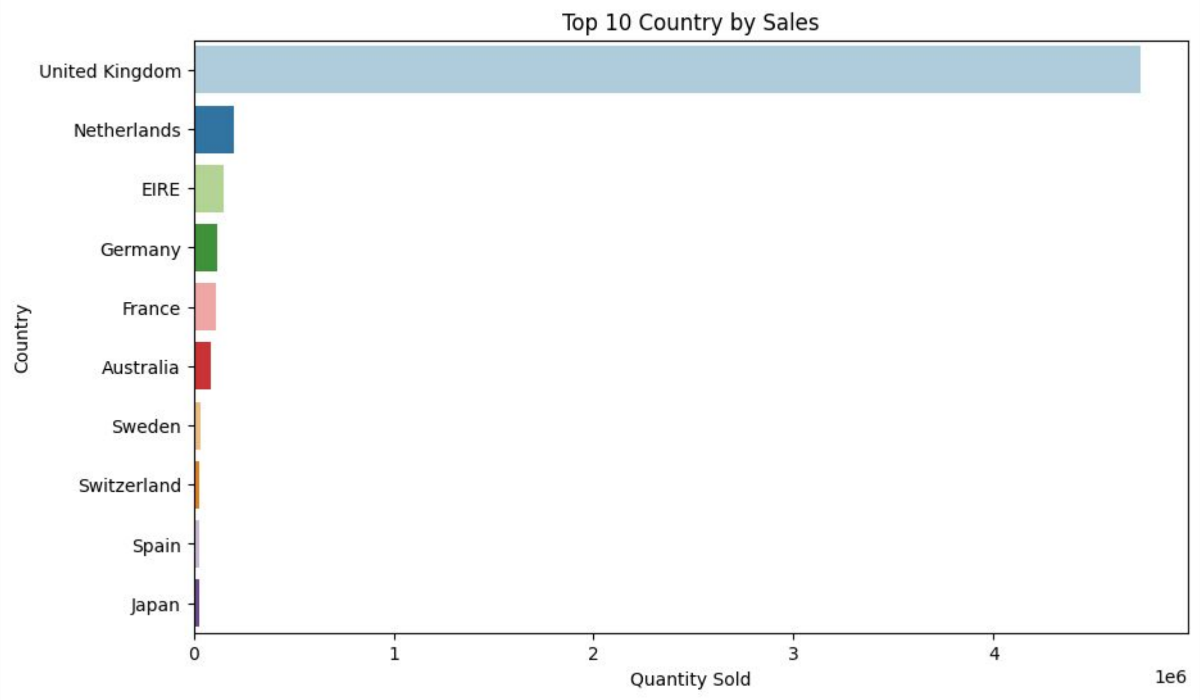
- **Monthly Sales Trend**



The highest number of sales was in November with 84.711 total number of order. The lowest number of sales was in December with 25.525 total number of order. There was significantly increasing in October to November by 23.969 number of order. Also, there was significantly decreasing in November to December by 59.186 number of order.

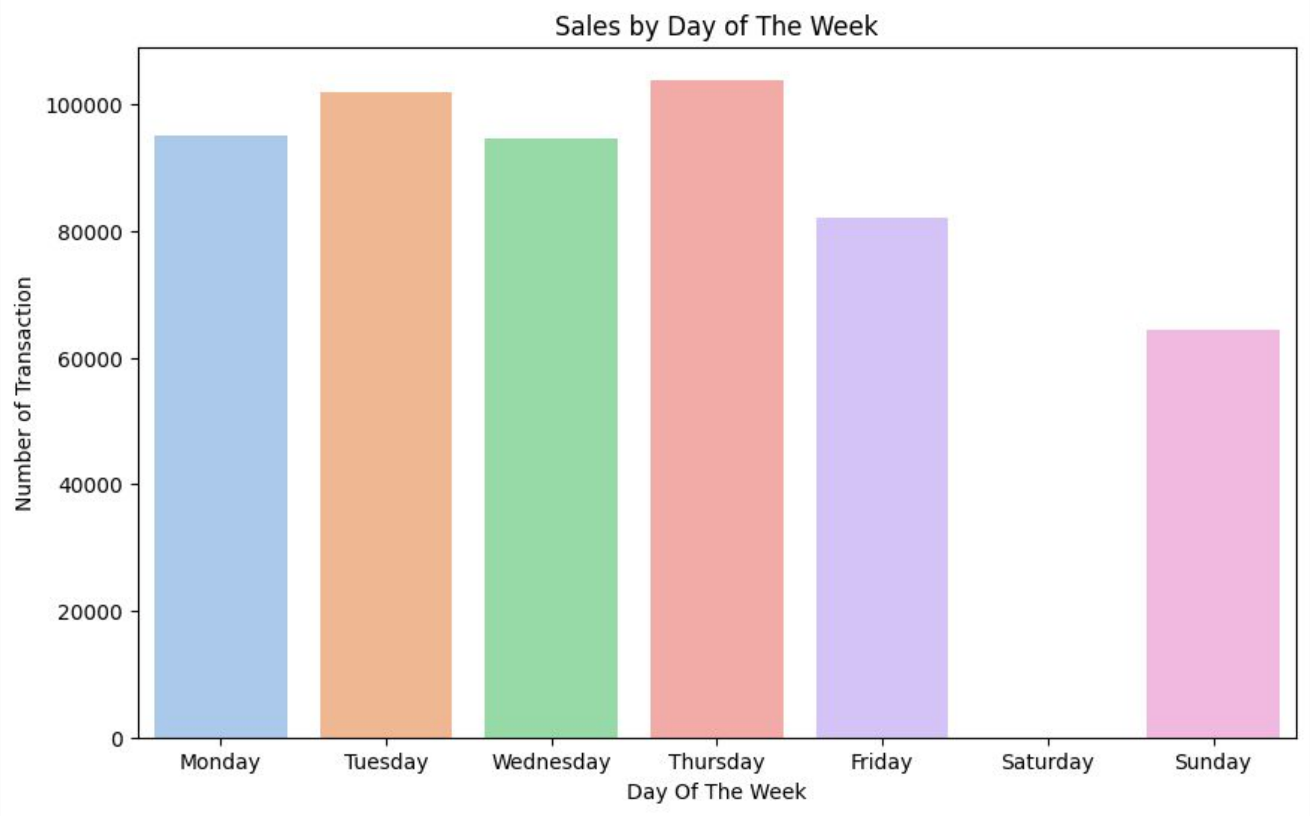
Customer Demographic

- **Most Customer Location**



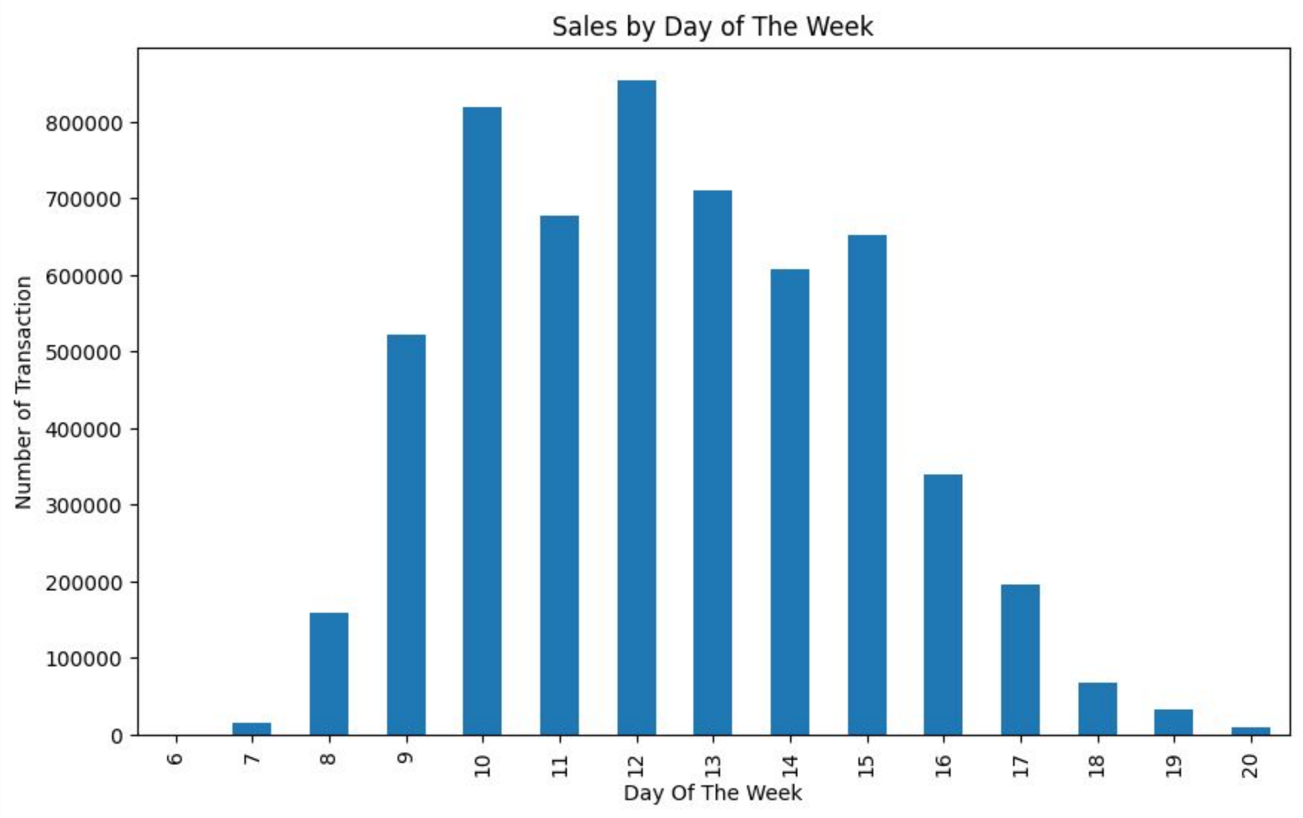
The chart shows that most customer's locations are based on United Kingdom. It shows that the local market is still the strongest among other country.

Sales Performance by Week

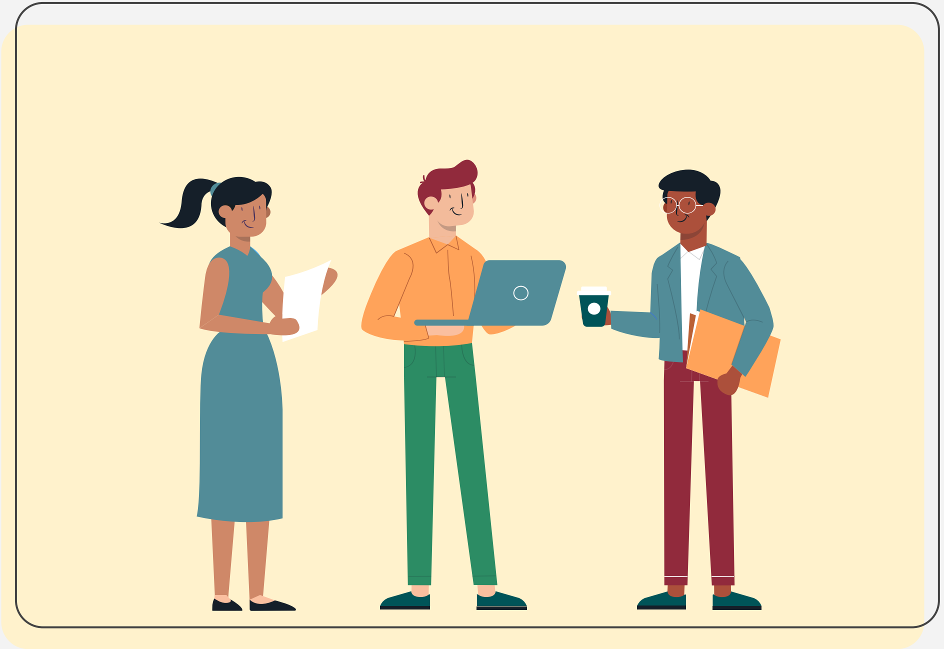


The chart shows that most sales by week were coming on Tuesday and Thursday. In these busy day, the company should add more work shift to avoid overload order.

Sales Performance by Day



The chart shows that the busiest order was at 10AM and 12PM with the peak of the order was at 12PM. In these busy hour, the company should add more employee on shift to avoid overload order.



RFM Analysis

RFM Analysis

RFM analysis is a marketing technique used to quantitatively rank and group customers based on the recency, frequency and monetary total of recent transactions to identify the best customers and perform targeted marketing campaigns.

1, **Recency**: How recent was the customer's last purchase? Customers who recently made a purchase will still have the product on their mind and are more likely to purchase the product again.
2. **Frequency**: How often did customer make a purchase in a given period? Customers who purchased once are often more likely to purchase again. Additionally, first time customers may be good targets for follow-up advertising to convert them into more frequent customers.
3. **Monetary**: How much money did customer spend in a given period? Customers who spend a lot of money are more likely to spend money in the future and have a high value to a business.

RFM Analysis

- Calculating RFM Value

```
# calculate the last transaction date
last_transaction = df["InvoiceDate"].max()

# calculate the recency, frequency, and monetary
rfm = df.groupby("CustomerID").agg({
    "InvoiceDate": lambda x: (last_transaction - x.max()).days,
    "InvoiceNo": "count",
    "TotalPrice": "sum"
}).reset_index()

# create column for RFM table
rfm.columns = ["CustomerID", "recency", "frequency", "monetary"]

# show rfm table
rfm.head()
```

	CustomerID	recency	frequency	monetary
0	12346.0	325	2	77183.60
1	12347.0	1	182	4310.00
2	12348.0	74	31	1797.24
3	12349.0	18	73	1757.55
4	12350.0	309	17	334.40

```
rfm.describe()
```

	recency	frequency	monetary
count	8082.000000	8082.000000	8082.000000
mean	132.116803	67.051349	1361.230540
std	114.035982	185.114186	6800.824868
min	0.000000	1.000000	0.000000
25%	28.000000	1.000000	9.537500
50%	96.000000	19.000000	326.400000
75%	227.000000	70.000000	1140.090000
max	373.000000	7983.000000	282862.021449

RFM Analysis

- **Specify the binning range to predetermine the customer segmentation**

```python
# specify the binning range
recency_bins = [0, 28, 96, 227, 373]
frequency_bins = [0, 1, 19, 70, 7983]
monetary_bins = [0, 10, 326, 1140, 282862]

# Apply binning range to the determine RFM Score
rfm["R"] = pd.cut(rfm["recency"], bins=recency_bins, labels=["1", "2", "3", "4"])
rfm["F"] = pd.cut(rfm["frequency"], bins=frequency_bins, labels=["1", "2", "3", "4"])
rfm["M"] = pd.cut(rfm["monetary"], bins=monetary_bins, labels=["1", "2", "3", "4"])

# Combining the RFM Score
rfm["RFM_Score"] = rfm["R"].astype(str) + rfm["F"].astype(str) + rfm["M"].astype(str)
rfm[["CustomerID", "recency", "frequency", "monetary", "RFM_Score"]].head()
```

The binning range is used to determine the customer segmentation. The binning range determined by observing the mean, minimum, maximum, and quartile on each RFM.

	CustomerID	recency	frequency	monetary	RFM_Score
0	12346.0	325	2	77183.60	424
1	12347.0	1	182	4310.00	144
2	12348.0	74	31	1797.24	234
3	12349.0	18	73	1757.55	144
4	12350.0	309	17	334.40	423

RFM Analysis

- **Create the customer segmentation**

```
# Create customer segmentation column
rfm["Customer_Segment"] = "Undefined"

# Determine the conditions based on RFM score
champion_condition = (rfm["RFM_Score"] == "111")
loyal_condition = (rfm["RFM_Score"] == "114")
big_spender_condition = (rfm["RFM_Score"] == "211")
at_risk_condition = (rfm["RFM_Score"] == "212")
lost_condition = (rfm["RFM_Score"] == "444")

# Named the column based on condition that already defined
rfm.loc[champion_condition, "Customer_Segment"] = "Champion Customers"
rfm.loc[loyal_condition, "Customer_Segment"] = "Loyal Customers"
rfm.loc[big_spender_condition, "Customer_Segment"] = "Big Spenders"
rfm.loc[at_risk_condition, "Customer_Segment"] = "At Risk Customers"
rfm.loc[lost_condition, "Customer_Segment"] = "Lost Customers"

rfm.head()
```

	CustomerID	recency	frequency	monetary	R	F	M	RFM_Segment	RFM_Score	Customer_Segment
0	12346.0	325	2	77183.60	4	2	4	424	424	Undefined
1	12347.0	1	182	4310.00	1	4	4	124	144	Undefined
2	12348.0	74	31	1797.24	2	3	4	224	234	Undefined
3	12349.0	18	73	1757.55	1	4	4	124	144	Undefined
4	12350.0	309	17	334.40	4	2	3	423	423	Undefined

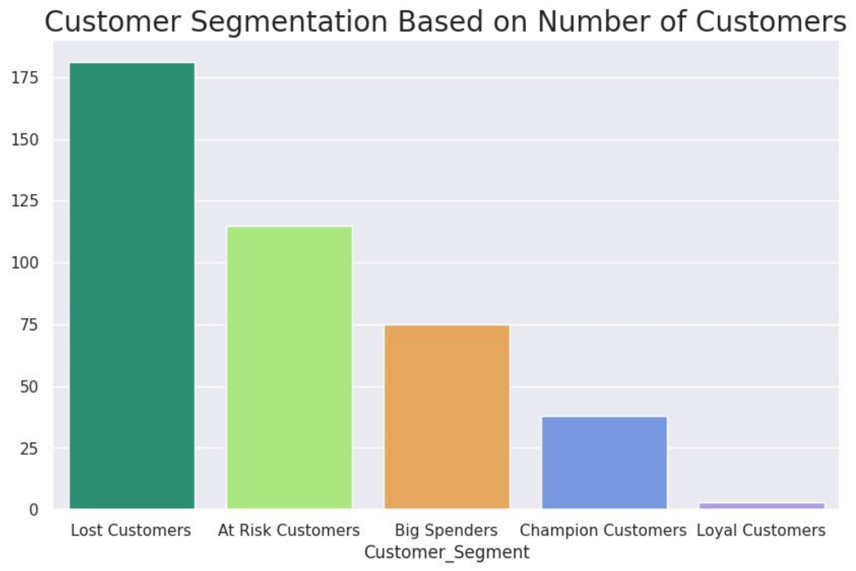
```
# select cuctomer segmentation that want to show in the chart
show_segments = ["Champion Customers", "Loyal Customers", "Big Spenders", "At Risk Customers", "Lost Customers"]
segments_df = rfm[rfm["Customer_Segment"].isin(show_segments)]

# Number of customers in each segment
segments_count = segments_df["Customer_Segment"].value_counts()

print(segments_count)
```

```
Customer_Segment
Lost Customers        181
At Risk Customers     115
Big Spenders           75
Champion Customers     38
Loyal Customers         3
Name: count, dtype: int64
```

After define the customer segmentation and count the number of each segments. We know that most of segment come from lost customer segmentation.

RFM Analysis

- **Visualize the customer segmentation.**



The chart shows that most of customers are on at risk and low segmentation. It should be the company concern to increase customer satisfaction and customer loyalty with giving any rewards or discount and treatment based on the segmentation. The description of each segments given below:

1. **Champion Customers**: Customers who are transacted recently, do so often and spend more than other customers.
2. **Loyal Customers**: Customers who bought most recently
3. **Big Spenders**: Customers who spend the most
4. **At Risk Customers**: Haven't purchased for some time, but purchased frequently and spend the most.
5. **Loyal Customers**: Last purchase long ago, purchased few and spend little.

Thank you!

If you have any questions, please contact:

farahdibah045@gmail.com
or
linkedin.com/in/farah-dibah-1436b6163/

Find more about the project code here:
github e-commerce analysis project