ECOMMERCE PROJECT:

Importing Libararies

In [4]:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
color = sns.color_palette()
import warnings
warnings.filterwarnings('ignore')
sns.set_style('whitegrid')
import gc
import datetime
```

Importing Dataset

In [5]:

```
df = pd.read_csv("C:/Users/yuvak/OneDrive/Desktop/Ecommerce - UK Retailer.csv",encoding='unicode_escape')
```

In [6]:

```
print(os.listdir())
```

```
['.conda', '.condarc', '.IBM', '.idlerc', '.ipynb_checkpoints', '.ipython',
'.jupyter', '.matplotlib', '.spss', '.VirtualBox', '3D Objects', 'anaconda3
', 'AppData', 'Application Data', 'Asgn - Playstore Analysis v0.1.pdf', 'Co
ntacts', 'Cookies', 'Documents', 'Downloads', 'E-commerce uk retailer proje
ct.ipynb', 'E-Commerce-EDA-Python Project 2.pdf', 'Ecommerce - UK Retailer.
csv', 'Favorites', 'Google playstore-analysis.ipynb', 'IntelGraphicsProfile
s', 'Links', 'Local Settings', 'Music', 'My Documents', 'NetHood', 'NTUSER.
DAT', 'ntuser.dat.LOG1', 'ntuser.dat.LOG2', 'NTUSER.DAT{bbdf736f-4479-11ec-
8f8b-002248474a9f}.TM.blf', 'NTUSER.DAT{bbdf736f-4479-11ec-8f8b-002248474a9
f}.TMContainer00000000000000000001.regtrans-ms', 'NTUSER.DAT{bbdf736f-4479-
11ec-8f8b-002248474a9f}.TMContainer00000000000000000002.regtrans-ms', 'ntus
er.ini', 'OneDrive', 'playstore-analysis (2) (1).csv', 'PrintHood', 'Recent
', 'Saved Games', 'Searches', 'SendTo', 'source', 'Start Menu', 'Templates'
, 'Videos', 'VirtualBox VMs']
```

In [7]:

```
df.head()
```

Out[7]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-01-2010 08:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-01-2010 08:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-01-2010 08:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-01-2010 08:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-01-2010 08:26	3.39	17850.0	United Kingdom

Basic information about the data-EDA

In [8]:

```
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  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

In [9]:

```
df.describe()
```

Out[9]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

Finding the duplicate values

In [10]:

```
df.duplicated().sum()
```

Out[10]:

```
5268
```

In [11]:

```
df.drop_duplicates()
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-01-2010 08:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12-01-2010 08:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-01-2010 08:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-01-2010 08:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-01-2010 08:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12-09-2011 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12-09-2011 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12-09-2011 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12-09-2011 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12-09-2011 12:50	4.95	12680.0	France

536641 rows × 8 columns

Identification of the unique values in the Represented columns.

In [12]:

```
df['InvoiceNo'].unique()
df['CustomerID'].unique()
df['InvoiceDate'].unique()
df['Country'].unique()
```

Out[12]:

```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

In [13]:

```
df.dtypes
```

Out[13]:

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

In [14]:

```
df.isnull().sum().sort_values(ascending=False)
```

Out[14]:

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

In [15]:

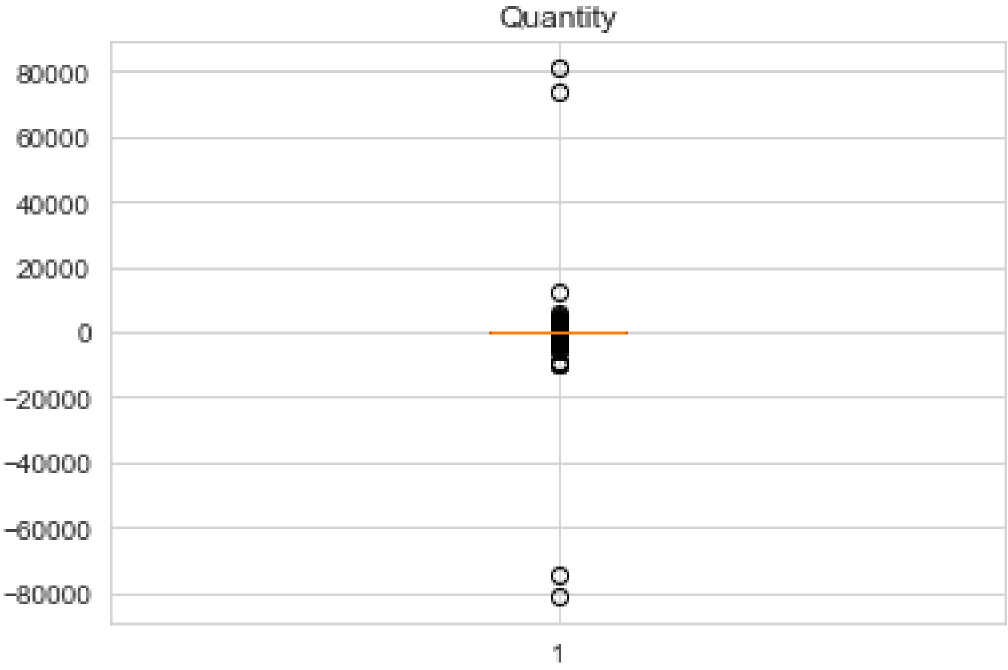
```
numeric_col =["Quantity","UnitPrice","CustomerID"]
categ_col = ["StockCode","Description","Country"]
```

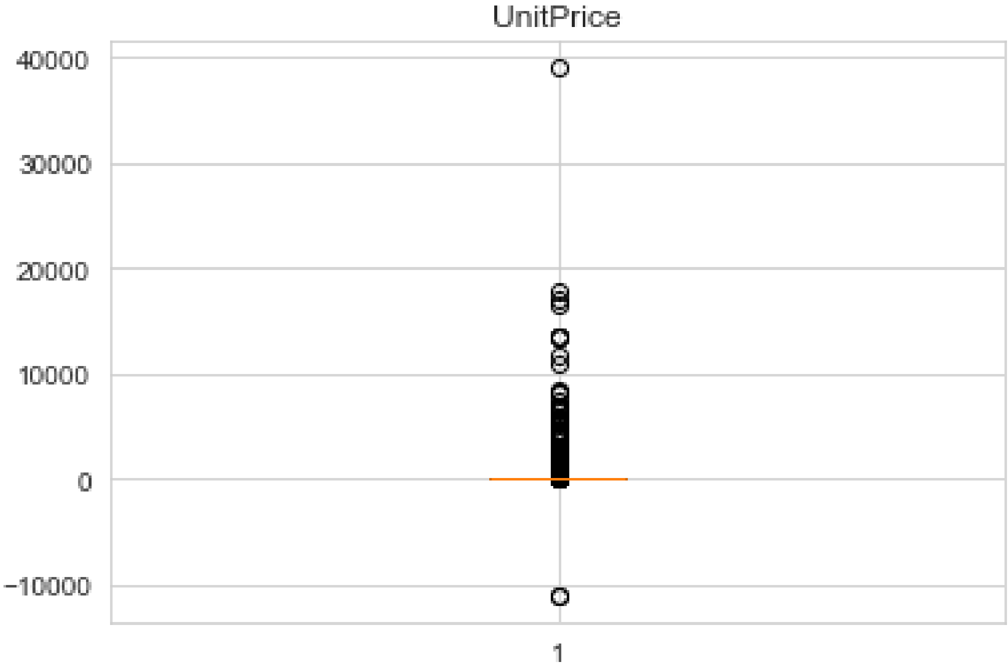
1. Perform Basic EDA

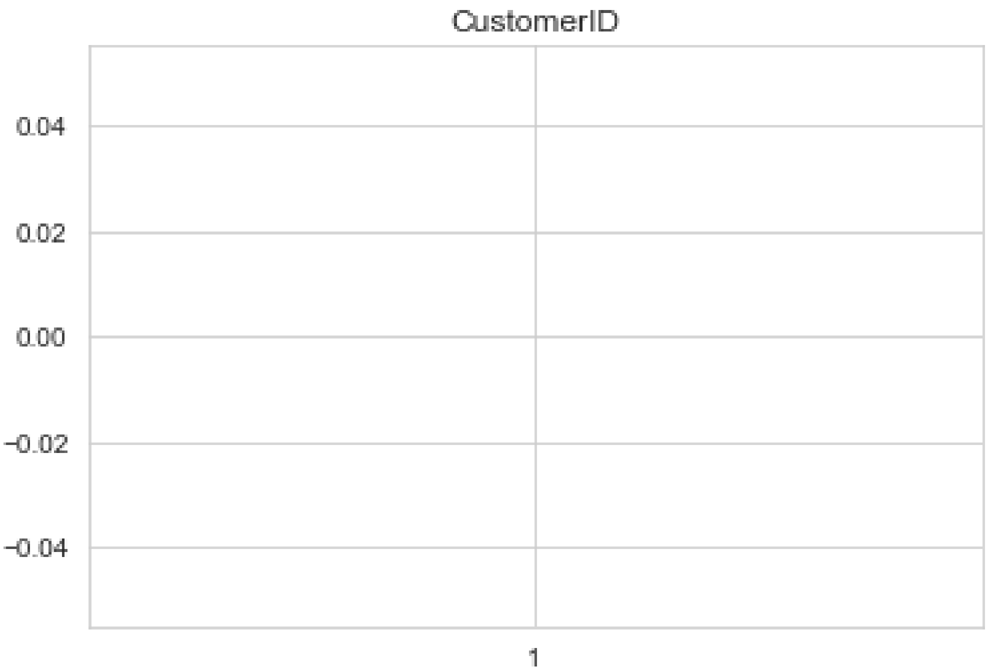
a. Boxplot – All Numeric Variables.

In [16]:

```
for i in numeric_col:
    plt.boxplot(df[i])
    plt.title(i)
    plt.show()
```



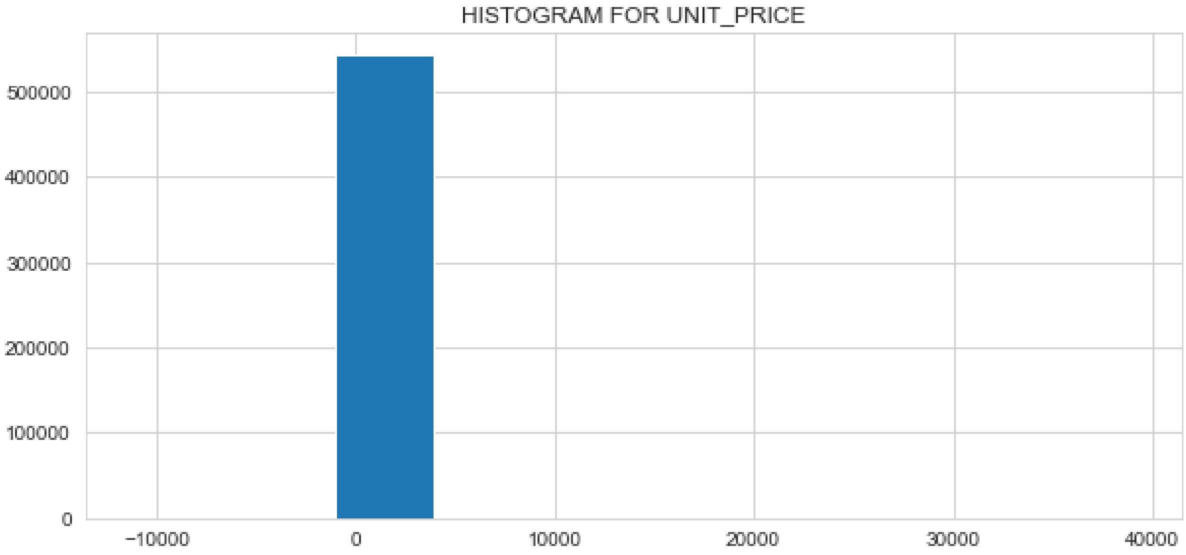


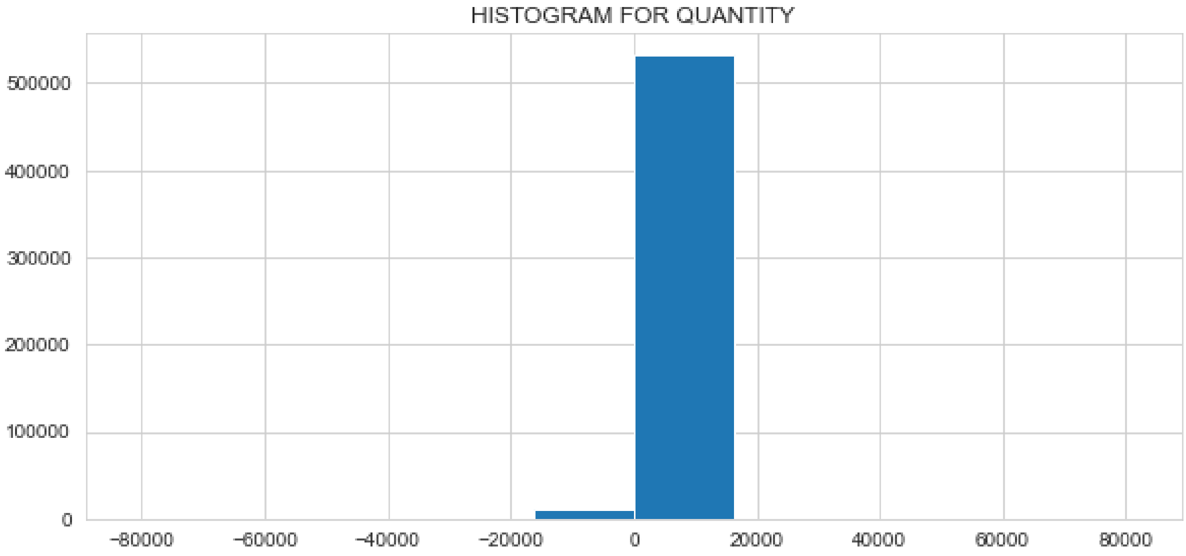


b. Histogram – All Numeric Variables.

In [17]:

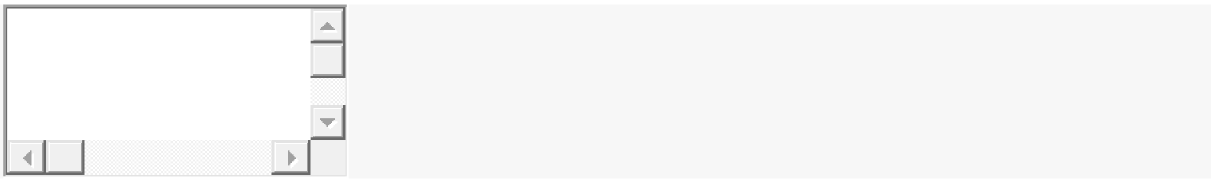
```
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.hist(df["UnitPrice"])
plt.title("HISTOGRAM FOR UNIT_PRICE")
plt.show()

plt.figure(figsize=(10,10))
plt.subplot(2,1,2)
plt.hist(df["Quantity"])
plt.title("HISTOGRAM FOR QUANTITY")
plt.show()
```



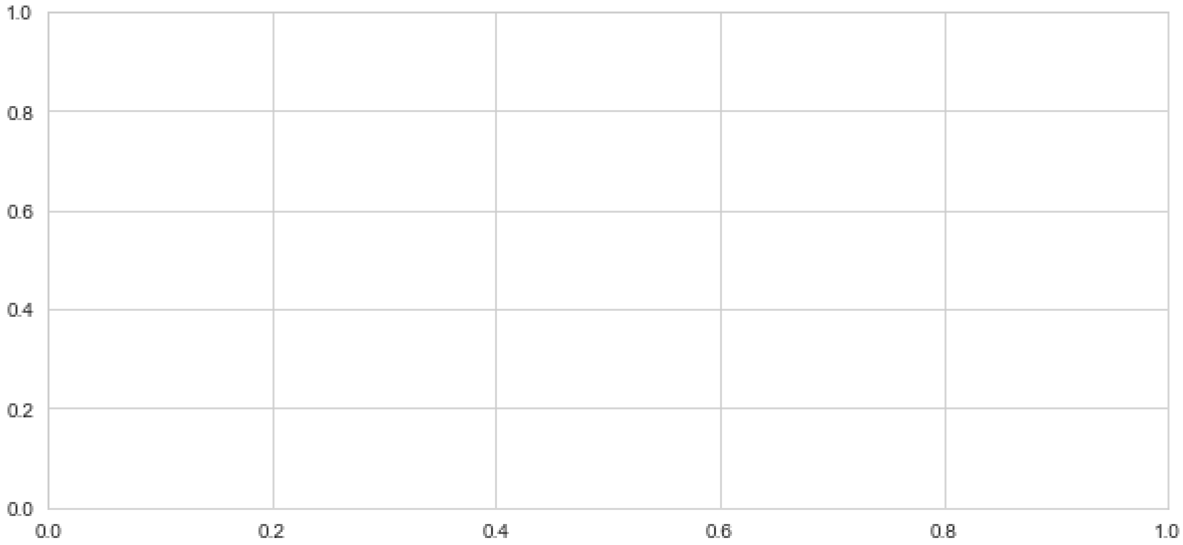


c. Distribution Plot – All Numeric Variables.

In [ ]:



```
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
sns.displot(df["UnitPrice"])
plt.title("DISTRIBUTION FOR UNIT_PRICE")
plt.show()

plt.figure(figsize=(10,10))
plt.subplot(2,1,2)
sns.displot(df["Quantity"])
plt.title("DISTRIBUTION FOR QUANTITY")
plt.show()
```



d. Aggregation for all numerical Columns.

In [18]:

```
for i in numeric_col:
    print(i,sum(df[i]))
```

```
Quantity 5176450
UnitPrice 2498803.9739972674
CustomerID nan
```

e. Unique Values across all columns.

In [19]:

```
col=df.columns
for i in col:
    print(i,df[i].nunique())
```

```
InvoiceNo 25900
StockCode 4070
Description 4223
Quantity 722
InvoiceDate 23260
UnitPrice 1630
CustomerID 4372
Country 38
```

f. Duplicate values across all columns.

In [20]:

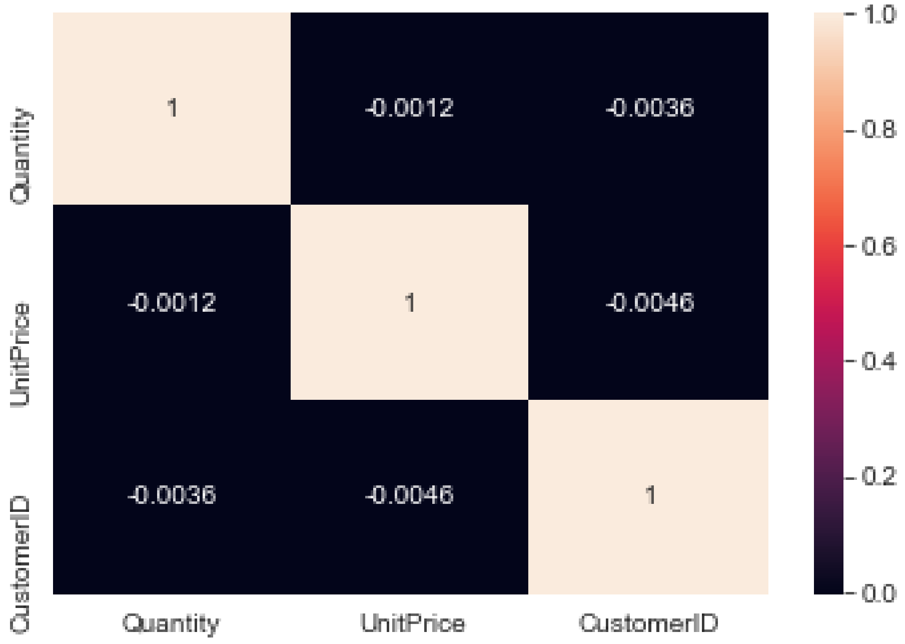
```
for i in col:
    print(i,df[i].duplicated().sum())
```

```
InvoiceNo 516009
StockCode 537839
Description 537685
Quantity 541187
InvoiceDate 518649
UnitPrice 540279
CustomerID 537536
Country 541871
```

g. Correlation – Heatmap - All Numeric Variables.

In [21]:

```
sns.heatmap(df.corr(), annot = True)
plt.show()
```



h. Regression Plot - All Numeric Variables.

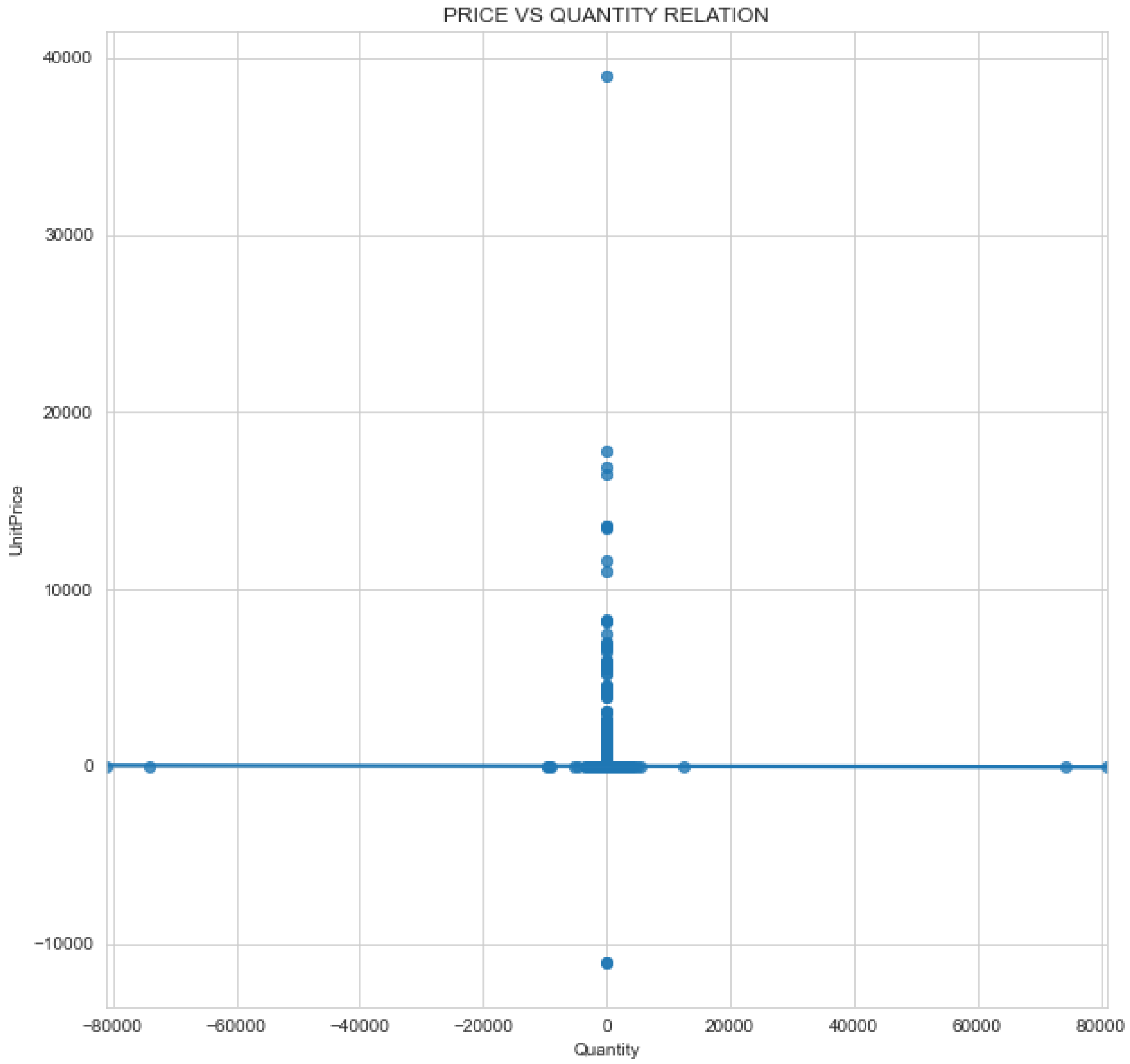
In [19]:

```
plt.figure(figsize=(10,10))
```

```
sns.regplot(data = df, x= "Quantity", y ="UnitPrice")
plt.title("PRICE VS QUANTITY RELATION")
```

Out[19]:

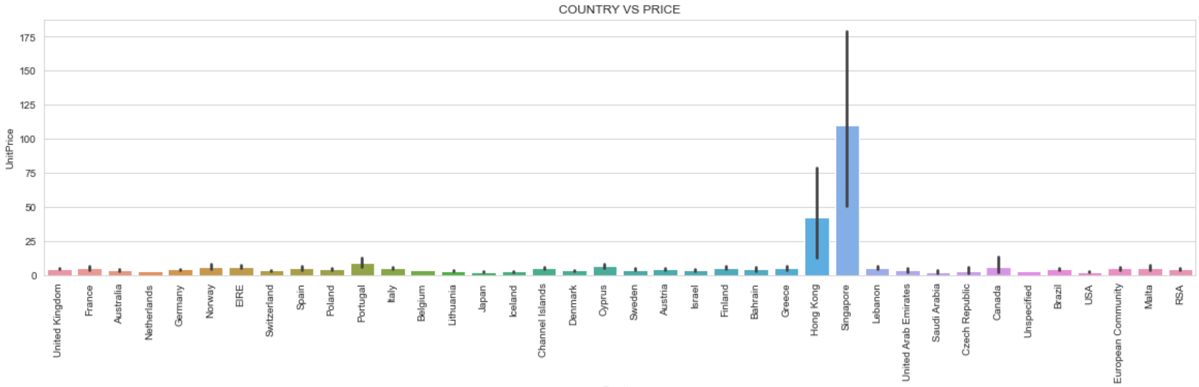
```
Text(0.5, 1.0, 'PRICE VS QUANTITY RELATION')
```



i. Bar Plot – Every Categorical Variable vs every Numerical Variable.

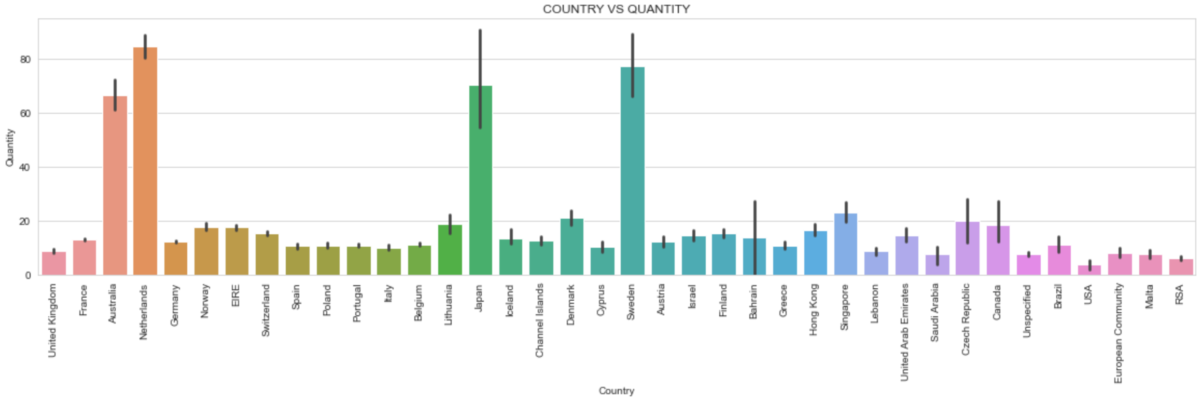
In [25]:

```
plt.figure(figsize=(20,10))
plt.subplot(2,1,1)
sns.barplot(data = df, x="Country", y="UnitPrice")
plt.xticks(rotation=90)
plt.title("COUNTRY VS PRICE")
plt.show()

plt.figure(figsize=(20,10))
plt.subplot(2,1,2)
sns.barplot(data = df, x="Country", y="Quantity")
plt.xticks(rotation=90)
plt.title("COUNTRY VS QUANTITY")
```



Out[25]:

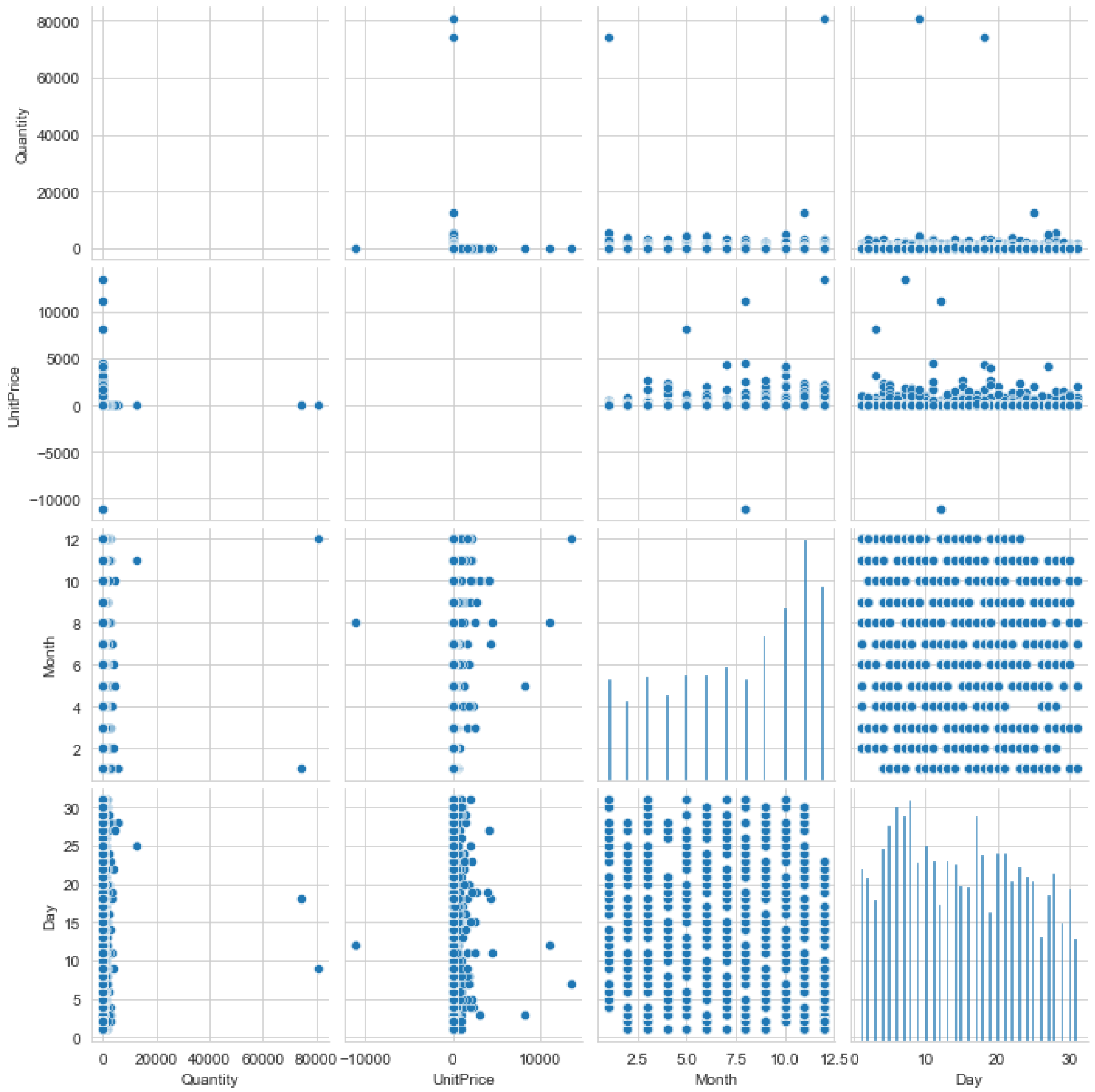
```
Text(0.5, 1.0, 'COUNTRY VS QUANTITY')
```



j. Pair plot - All Numeric Variables.

In [76]:

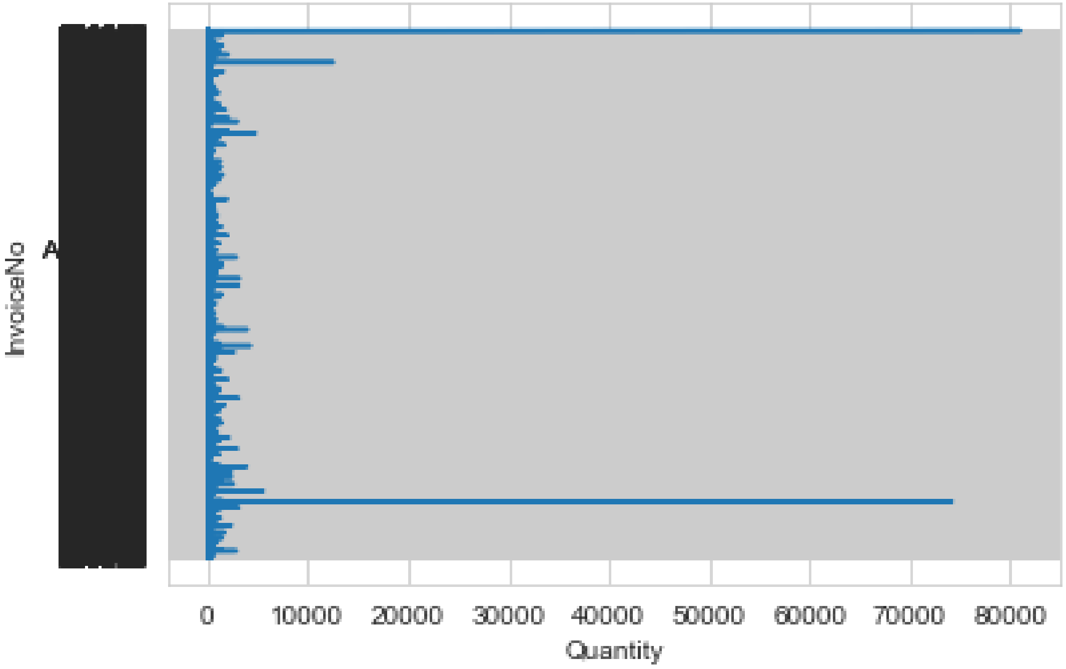
```
sns.pairplot(df,vars=["Quantity","UnitPrice","Month","Day"])
plt.show()
```



k. Line chart to show the trend of data - All Numeric/Date Variables.

In [85]:

```
plt.plot(df['Quantity'],df['InvoiceNo'])
plt.xlabel('Quantity')
plt.ylabel('InvoiceNo')
plt.show()
```



In [ ]:

```
sales_per_hour = df.groupby("hh")["price"].sum().reset_index()
sales_per_day = df.groupby("day")["price"].sum().reset_index()
sales_per_month = df.groupby("month")["price"].sum().reset_index()
```

In [ ]:

```
plt.figure(figsize=(40,8))
plt.subplot(1,3,1)
sns.lineplot(data=sales_per_month, x="month", y="price", marker = True, color = "red")
plt.title("SALES TRENDS ACROSS MONTH")

plt.figure(figsize=(40,8))
plt.subplot(1,3,2)
sns.lineplot(data=sales_per_day, x="day", y="price", marker = True, color = "red")
plt.title("SALES TRENDS ACROSS DAY")

plt.figure(figsize=(40,8))
plt.subplot(1,3,3)
sns.lineplot(data=sales_per_hour, x="hh", y="price", marker = True, color = "red")
plt.title("SALES TRENDS ACROSS HOUR");
```

In [ ]:

```
quan_per_hour = df.groupby("hh")["Quantity"].sum().reset_index()
quan_per_day = df.groupby("day")["Quantity"].sum().reset_index()
quan_per_month = df.groupby("month")["Quantity"].sum().reset_index()
```

In [ ]:

```
plt.figure(figsize=(40,8))
plt.subplot(1,3,1)
sns.lineplot(data=quan_per_month, x="month", y="Quantity", marker = True, color = "red")
plt.title("QUANTITY TRENDS ACROSS MONTH")

plt.figure(figsize=(40,8))
plt.subplot(1,3,2)
sns.lineplot(data=quan_per_day, x="day", y="Quantity", marker = True, color = "red")
plt.title("QUANTITY TRENDS ACROSS DAY")

plt.figure(figsize=(40,8))
plt.subplot(1,3,3)
sns.lineplot(data=quan_per_hour, x="hh", y="Quantity", marker = True, color = "red")
plt.title("QUANTITY TRENDS ACROSS HOUR");
```

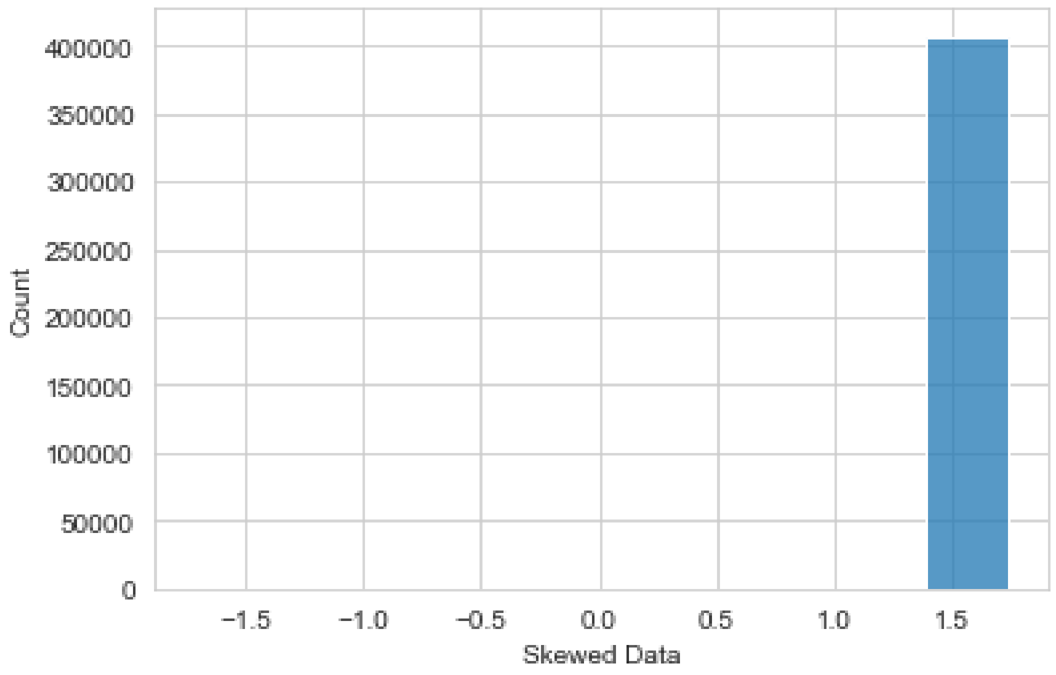
i. Plot the skewness - All Numeric Variables.

In [22]:

```
df['Skewed Data'] = pd.DataFrame(df.skew(axis=1,skipna=True))
```

In [23]:

```
sns.histplot(df['Skewed Data'],bins=10);
```

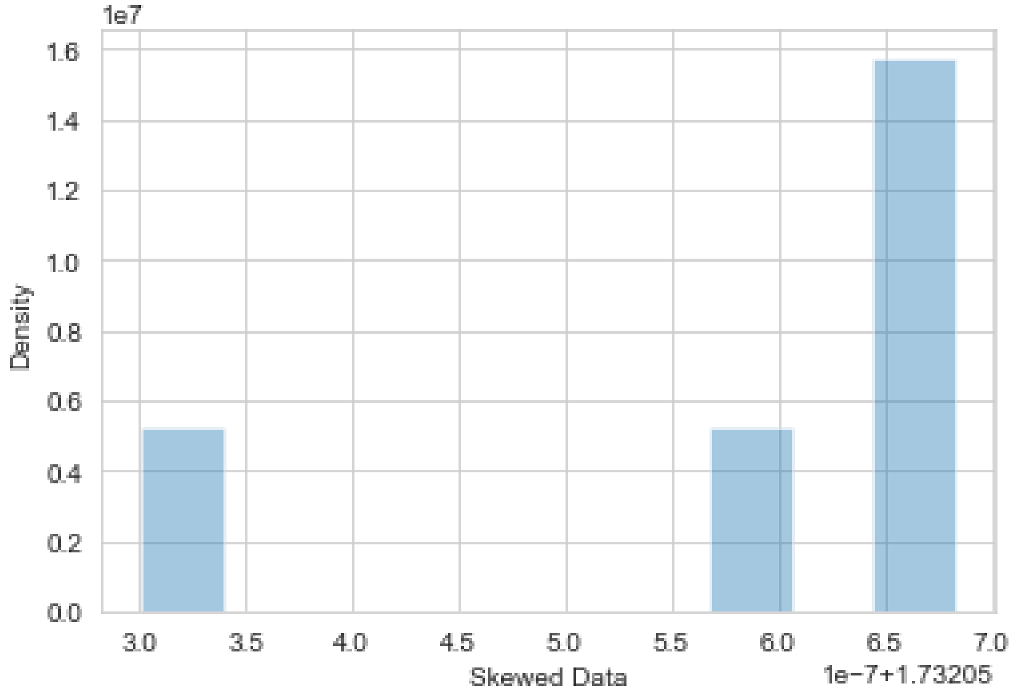


In [24]:

```
sns.distplot(df['Skewed Data'].head(), bins=10)
```

Out[24]:

```
<AxesSubplot:xlabel='Skewed Data', ylabel='Density'>
```



2. Check for missing values in all columns and replace them with the appropriate metric.

(Mean/Median/Mode)

In [25]:

```
df.isnull().sum().sort_values(ascending=False)
```

Out[25]:

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

In [26]:

```
df['CustomerID'].fillna(df['CustomerID'].mode()[0],inplace=True)
```

In [27]:

```
df['Description'].fillna(df['Description'].mode()[0],inplace=True)
```

In [28]:

```
df[df["Description"].isna()]
```

Out[28]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Skewed Data

In [29]:

```
df.isnull().sum().sort_values(ascending=False)
```

Out[29]:

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

3. Remove duplicate rows.

In [30]:

```
df.duplicated().sum()
```

Out[30]:

```
5268
```

In [31]:

```
df.drop_duplicates(inplace = True)
```

4. Remove rows which have negative values in Quantity column.

In [32]:

```
temp=df[df["Quantity"]<0].index
```

In [33]:

```
temp
```

Out[33]:

```
Int64Index([   141,    154,    235,    236,    237,    238,    239,    240,
               241,    939,
            ...
            540141, 540142, 540176, 540422, 540448, 540449, 541541, 541715,
            541716, 541717],
           dtype='int64', length=10587)
```

In [34]:

```
df.drop(labels = temp, inplace = True)
```

In [35]:

```
df[df["Quantity"]<0]
```

Out[35]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Skewed Data

5. Add the columns - Month, Day and Hour for the invoice.

In [36]:

```
df["Invoicedate"] = pd.to_datetime(df['InvoiceDate'])
```

In [37]:

```
df.dtypes
```

Out[37]:

```
InvoiceNo              object
```

```
StockCode                 object
Description               object
Quantity                   int64
InvoiceDate               object
UnitPrice                float64
CustomerID               float64
Country                   object
Skewed Data              float64
Invoicedate       datetime64[ns]
dtype: object
```

In [38]:

```
df["Day"]= pd.DatetimeIndex(df["InvoiceDate"]).day
df["Month"]= pd.DatetimeIndex(df["InvoiceDate"]).month
df["Year"]= pd.DatetimeIndex(df["InvoiceDate"]).year
```

In [39]:

```
df.head()
```

Out[39]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Skewed Data	Invoicedate	Day	Month	Year
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-01-2010 08:26	2.55	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010
1	536365	71053	WHITE METAL LANTERN	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Skewed Data	Invoicedate	Day	Month	Year
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-01-2010 08:26	2.75	17850.0	United Kingdom	1.732050	2010-12-01 08:26:00	1	12	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010

6. How many orders made by the customers?

In [40]:

```
df.groupby(by=["CustomerID","Country"])['InvoiceNo'].count()
```

Out[40]:

```
CustomerID  Country
12346.0     United Kingdom      1
12347.0     Iceland           182
12348.0     Finland            31
12349.0     Italy              73
12350.0     Norway             17
                              ...
18280.0     United Kingdom     10
18281.0     United Kingdom      7
```

```
18282.0     United Kingdom     12
18283.0     United Kingdom    721
18287.0     United Kingdom     70
Name: InvoiceNo, Length: 4355, dtype: int64
```

7. TOP 5 customers with higher number of orders.

In [41]:

```
df.groupby(by=["CustomerID","Country"])['InvoiceNo'].count().sort_values(ascending = False).head(5)
```

Out[41]:

```
CustomerID  Country
17841.0     United Kingdom    139592
14911.0     EIRE                5672
14096.0     United Kingdom      5111
12748.0     United Kingdom      4413
14606.0     United Kingdom      2677
Name: InvoiceNo, dtype: int64
```

8. How much money spent by the customers?

In [42]:

```
df["amount"] = df["Quantity"]*df["UnitPrice"]
df["amount"].astype(int)
```

Out[42]:

```
0         15
1         20
2         22
3         20
4         20
          ..
541904    10
541905    12
541906    16
541907    16
541908    14
Name: amount, Length: 526054, dtype: int32
```

In [43]:

```
total_spent = df.groupby(by=["CustomerID","Country"], as_index=False)['amount'].sum()
total_spent
```

Out[43]:

	CustomerID	Country	amount
0	12346.0	United Kingdom	77183.60
1	12347.0	Iceland	4310.00
2	12348.0	Finland	1797.24
3	12349.0	Italy	1757.55
4	12350.0	Norway	334.40
...	...	...	...
4350	18280.0	United Kingdom	180.60
4351	18281.0	United Kingdom	80.82
4352	18282.0	United Kingdom	178.05
4353	18283.0	United Kingdom	2045.53
4354	18287.0	United Kingdom	1837.28

4355 rows × 3 columns

In [44]:

```
df["amount"].sum()
```

Out[44]:

```
10619986.684
```

9. TOP 5 customers with highest money spent.

In [45]:

```
total_spent.sort_values(by='amount', ascending=False).head()
```

Out[45]:

	CustomerID	Country	amount
4026	17841.0	United Kingdom	1.735115e+06
1698	14646.0	Netherlands	2.802060e+05
4218	18102.0	United Kingdom	2.596573e+05
3737	17450.0	United Kingdom	1.943908e+05
3017	16446.0	United Kingdom	1.684725e+05

10. How many orders per month?

In [46]:

```
order_pr_month = df.groupby(by="Month")["InvoiceNo"].count()
order_pr_month
```

Out[46]:

```
Month
1      34104
2      26961
3      35609
4      28957
5      36044
6      35793
7      38466
8      34347
9      48962
10     58629
11     82133
12     66049
Name: InvoiceNo, dtype: int64
```

In [47]:

```
month=[x for x in range(1,13)]
data=order_pr_month.values
```

In [48]:

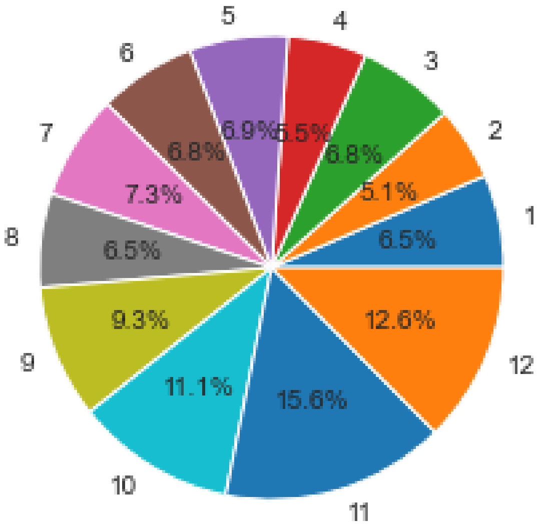
```
data
```

Out[48]:

```
array([34104, 26961, 35609, 28957, 36044, 35793, 38466, 34347, 48962,
       58629, 82133, 66049], dtype=int64)
```

In [49]:

```
plt.pie(data,labels=month,autopct = '%1.1f%%')
plt.show()
```



11. How many orders per day?

In [50]:

```
df["hour"]= pd.DatetimeIndex(df["InvoiceDate"]).hour
```

In [51]:

```
order_pr_day = df.groupby(by="Day")["InvoiceNo"].count()
order_pr_day
```

Out[51]:

```
Day
1     17035
2     16232
3     14049
4     19026
5     21467
6     23248
7     22350
8     23935
9     17658
10    19334
11    17891
12    13618
13    17845
14    17489
15    15344
16    15217
17    22361
18    18522
19    12757
20    18603
21    18602
22    15785
23    17306
24    16331
25    15797
26    10246
27    14448
28    16676
29    11663
30    15069
31    10150
Name: InvoiceNo, dtype: int64
```

13. How many orders for each country?

In [52]:

```
df
```

Out[52]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	SkewedData	Invoicedate	Day	Month	Year	amount	hour
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12-01-2010 08:26	2.55	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010	15.30	8
1	536365	71053	WHITE METAL LANTERN	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010	20.34	8
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12-01-2010 08:26	2.75	17850.0	United Kingdom	1.732050	2010-12-01 08:26:00	1	12	2010	22.00	8
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010	20.34	8
4	536365	84029E	RED WOOLLY HOTTIE	6	12-01-2010 08:26	3.39	17850.0	United Kingdom	1.732051	2010-12-01 08:26:00	1	12	2010	20.34	8

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	SkewedData	Invoicedate	Day	Month	Year	amount	hour
			WHITE HEART.												
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12-09-2011 12:50	0.85	12680.0	France	1.732046	2011-12-09 12:50:00	9	12	2011	10.20	12
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12-09-2011 12:50	2.10	12680.0	France	1.732050	2011-12-09 12:50:00	9	12	2011	12.60	12
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12-09-2011 12:50	4.15	12680.0	France	1.732051	2011-12-09 12:50:00	9	12	2011	16.60	12
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12-09-2011 12:50	4.15	12680.0	France	1.732051	2011-12-09 12:50:00	9	12	2011	16.60	12
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12-09-2011 12:50	4.95	12680.0	France	1.732051	2011-12-09 12:50:00	9	12	2011	14.85	12

526054 rows × 15 columns

In [53]:

```
orderbycountry=df.groupby(by="Country")['InvoiceNo'].count()
orderbycountry
```

Out[53]:

```
Country
Australia                 1184
Austria                    398
Bahrain                     18
Belgium                   2031
Brazil                      32
Canada                     151
Channel Islands            747
Cyprus                     603
Czech Republic              25
Denmark                    380
EIRE                      7883
European Community          60
Finland                    685
France                    8393
Germany                   9027
Greece                     145
Hong Kong                  280
Iceland                    182
Israel                     292
Italy                      758
Japan                      321
Lebanon                     45
Lithuania                   35
Malta                      112
Netherlands               2363
Norway                    1072
Poland                     330
Portugal                  1492
RSA                         58
Saudi Arabia                 9
Singapore                  222
Spain                     2480
Sweden                     450
Switzerland               1959
USA                        179
United Arab Emirates        68
United Kingdom          481143
```

```
Unspecified                   442
Name: InvoiceNo, dtype: int64
```

In [54]:

```
df["Country"].nunique()
```

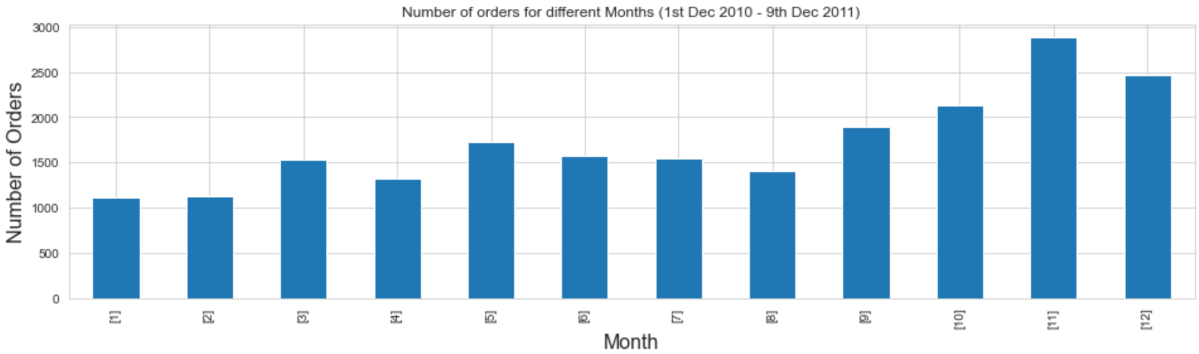
Out[54]:

```
38
```

14. Orders trend across months

In [55]:

```
ax = df.groupby('InvoiceNo')['Month'].unique().value_counts().sort_index().plot(kind='bar',figsize=(16,4))
ax.set_xlabel('Month',fontsize=16)
ax.set_ylabel('Number of Orders',fontsize=16)
ax.set_title('Number of orders for different Months (1st Dec 2010 - 9th Dec 2011)',fontsize=12)
plt.show()
```



15. How much money spent by each country?

In [56]:

```
spent_country = df.groupby(by="Country")['amount'].sum()
spent_country
```

Out[56]:

```
Country
Australia             1.384538e+05
Austria               1.019868e+04
Bahrain               7.541400e+02
Belgium               4.119634e+04
```

```
Brazil                  1.143600e+03
Canada                  3.666380e+03
Channel Islands         2.044054e+04
Cyprus                  1.350285e+04
Czech Republic          8.267400e+02
Denmark                 1.895534e+04
EIRE                    2.831405e+05
European Community      1.300250e+03
Finland                 2.254608e+04
France                  2.096254e+05
Germany                 2.286784e+05
Greece                  4.760520e+03
Hong Kong               1.548300e+04
Iceland                 4.310000e+03
Israel                  8.129410e+03
Italy                   1.748324e+04
Japan                   3.741637e+04
Lebanon                 1.693880e+03
Lithuania               1.661060e+03
Malta                   2.725590e+03
Netherlands             2.854463e+05
Norway                  3.616544e+04
Poland                  7.334650e+03
Portugal                3.368305e+04
RSA                     1.002310e+03
Saudi Arabia            1.459200e+02
Singapore               2.127929e+04
Spain                   6.155856e+04
Sweden                  3.836783e+04
Switzerland             5.706760e+04
USA                     3.580390e+03
United Arab Emirates    1.902280e+03
United Kingdom          8.979620e+06
Unspecified             4.740940e+03
Name: amount, dtype: float64
```

In [59]:

```
spent_country =df.groupby(by="Country")['amount'].sum().reset_index(drop=True)
spent_country
```

Out[59]:

```
0      1.384538e+05
1      1.019868e+04
2      7.541400e+02
3      4.119634e+04
```

```
4      1.143600e+03
5      3.666380e+03
6      2.044054e+04
7      1.350285e+04
8      8.267400e+02
9      1.895534e+04
10     2.831405e+05
11     1.300250e+03
12     2.254608e+04
13     2.096254e+05
14     2.286784e+05
15     4.760520e+03
16     1.548300e+04
17     4.310000e+03
18     8.129410e+03
19     1.748324e+04
20     3.741637e+04
21     1.693880e+03
22     1.661060e+03
23     2.725590e+03
24     2.854463e+05
25     3.616544e+04
26     7.334650e+03
27     3.368305e+04
28     1.002310e+03
29     1.459200e+02
30     2.127929e+04
31     6.155856e+04
32     3.836783e+04
33     5.706760e+04
34     3.580390e+03
35     1.902280e+03
36     8.979620e+06
37     4.740940e+03
Name: amount, dtype: float64
```

In [63]:

```
spent_country= spent_country.head()
spent_country
```

Out[63]:

```
0    138453.81
1     10198.68
2       754.14
3     41196.34
```

```
4      1143.60
Name: amount, dtype: float64
```

In [ ]:

```
plt.figure(figsize=(40,30))
plt.plot(df['amount'],df['Country'])
plt.xlabel('amount')
plt.ylabel('Country')
plt.show()
```

Result for Ecommerce-uk-Retailer.

Top 10 orders made by the customers are UK, FINLAND, ITALY, NORWAY. Top 5 customers with higher number of orders are from UK, EIRE, UK,UK,UK (i.e, 17841, 14911, 14096, 12748, 14606)@ total = 5. Total amount spent by the countries is $9747747.933999998. Top 5 customers with highest money spent NETHERLANDS@279489.02, UK@256438.49, UK@187482.17, EIRE@132572.62, AUSTRALIA@123725.45. Here 38 unique countries have same orders. AND finally, the orders were increased in the 11th month @15% i.e, upto 3500 orders.