In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```python
df=pd.read_excel('uk_retail_file.xlsx')
```

In [3]:
```python
df.head()
```

Out[3]:

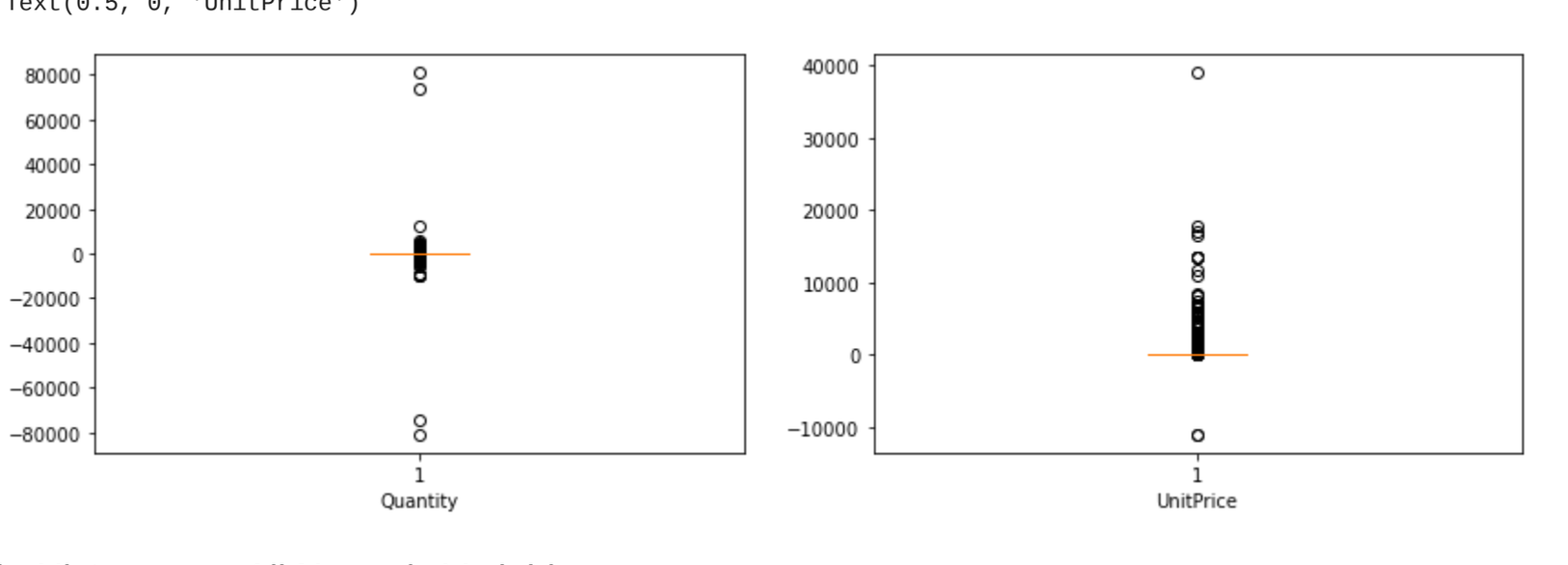
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-01-12 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-01-12 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS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

1. Perform Basic EDA

a. Boxplot – All Numeric Variables

In [4]:
```python
plt.figure(figsize=(14,4))
plt.subplot(1,2,1)
plt.boxplot(df['Quantity'])
plt.xlabel('Quantity')
plt.subplot(1,2,2)
plt.boxplot(df['UnitPrice'])
plt.xlabel('UnitPrice')
```

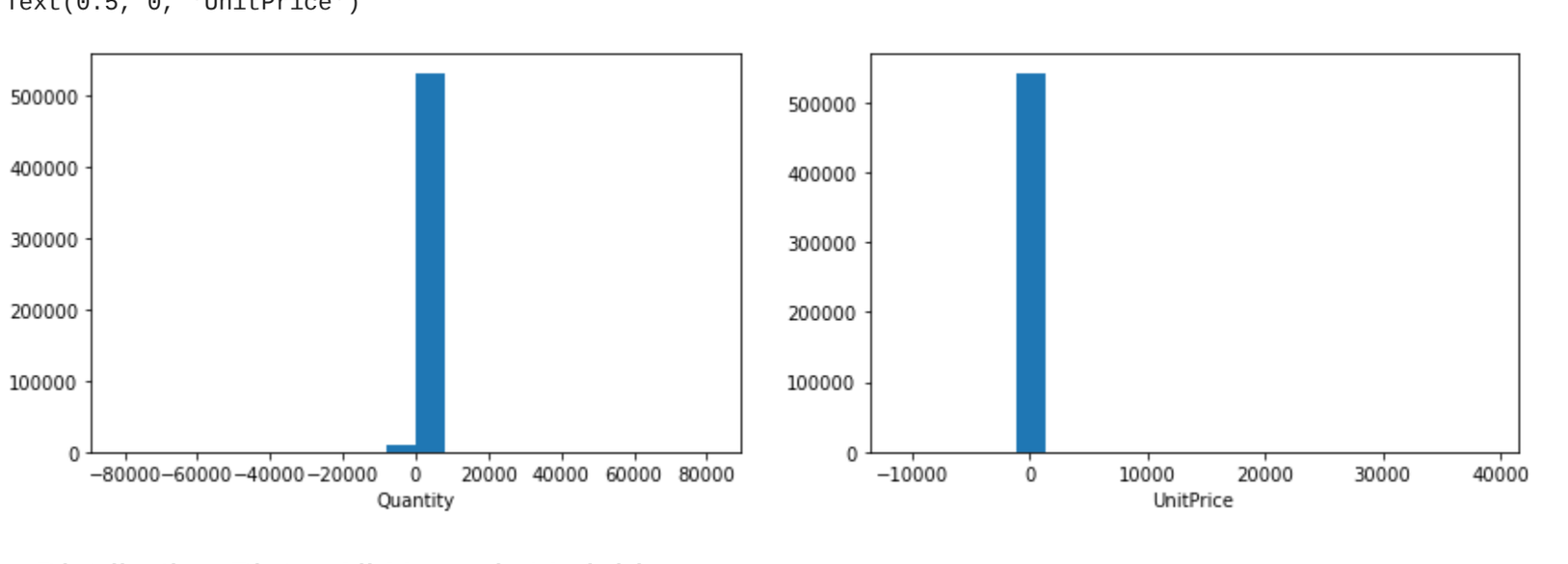
Out[4]:
```
Text(0.5, 0, 'UnitPrice')
```



b. Histogram – All Numeric Variables

In [5]:
```python
plt.figure(figsize=(14,4))
plt.subplot(1,2,1)
plt.hist(df['Quantity'],bins=20)
plt.xlabel('Quantity')
plt.subplot(1,2,2)
plt.hist(df['UnitPrice'],bins=20)
plt.xlabel('UnitPrice')
```

Out[5]:
```
Text(0.5, 0, 'UnitPrice')
```

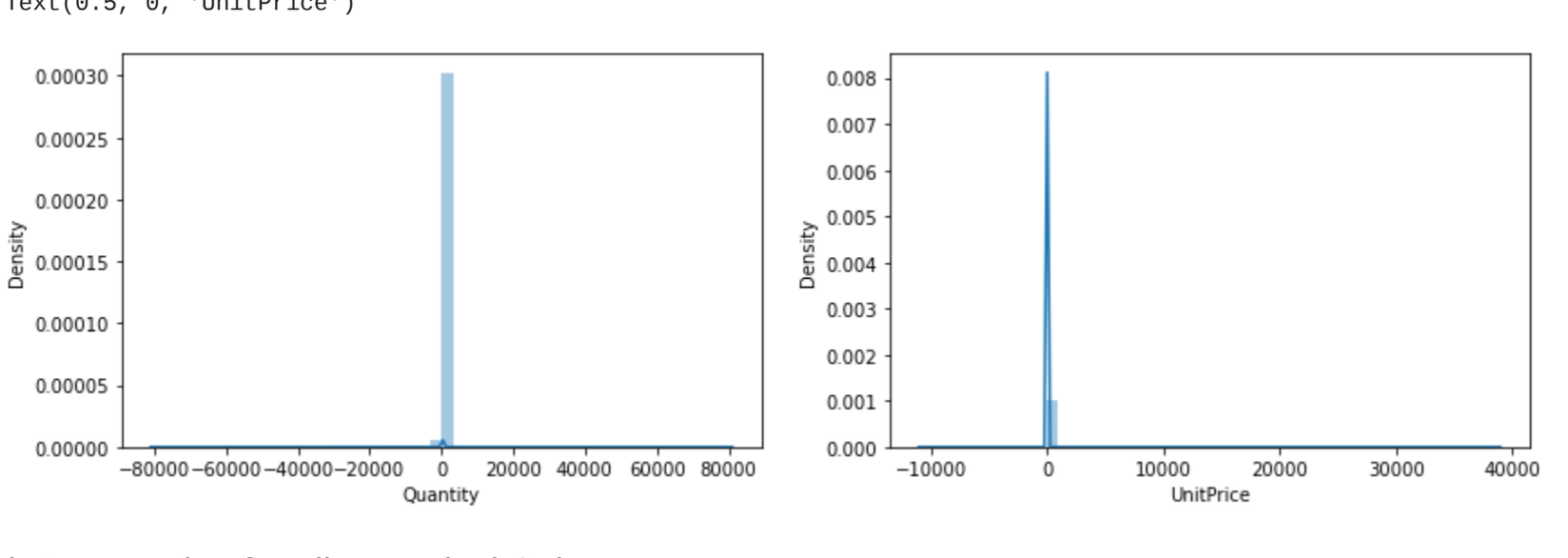


c. Distribution Plot – All Numeric Variables

In [6]:
```python
plt.figure(figsize=(14,4))
plt.subplot(1,2,1)
sns.distplot(df['Quantity'])
plt.xlabel('Quantity')
plt.subplot(1,2,2)
sns.distplot(df['UnitPrice'])
plt.xlabel('UnitPrice')
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\humza\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[6]:
```
Text(0.5, 0, 'UnitPrice')
```



d. Aggregation for all numerical Columns

In [7]:
```python
df.describe().round(1)
```

Out[7]:

	Quantity	UnitPrice	CustomerID
count	541909.0	541909.0	406829.0
mean	9.6	4.6	15287.7
std	218.1	96.8	1713.6
min	-80995.0	-11062.1	12346.0
25%	1.0	1.2	13953.0
50%	3.0	2.1	15152.0
75%	10.0	4.1	16791.0
max	80995.0	38970.0	18287.0

e. Unique Values across all columns

In [8]:
```python
df.nunique()
```

Out[8]:
```
InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate    23260
UnitPrice       1630
CustomerID      4372
Country           38
dtype: int64
```

f. Duplicate values across all columns

In [9]:
```python
df[df.duplicated()]
```

Out[9]:

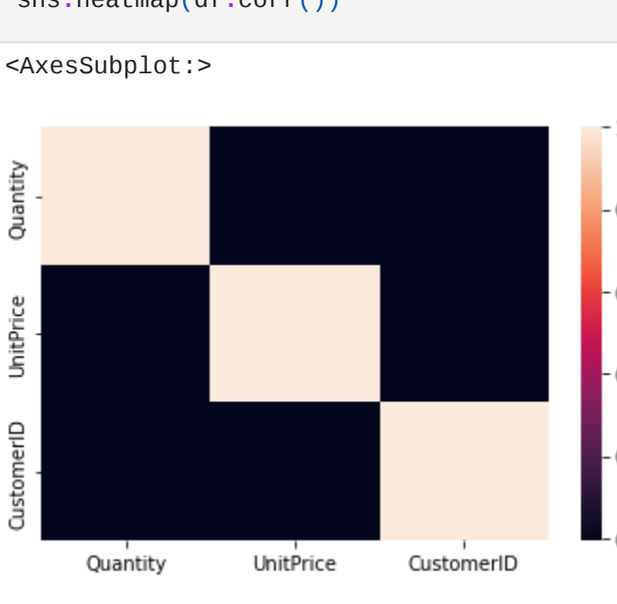
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
517	536409	21866	UNION JACK FLAG LUGGAGE TAG	1	2010-01-12 11:45:00	1.25	17908.0	United Kingdom
527	536409	22866	HAND WARMER SCOTTY DOG DESIGN	1	2010-01-12 11:45:00	2.10	17908.0	United Kingdom
537	536409	22900	SET 2 TEA TOWELS I LOVE LONDON	1	2010-01-12 11:45:00	2.95	17908.0	United Kingdom
539	536409	22111	SCOTTIE DOG HOT WATER BOTTLE	1	2010-01-12 11:45:00	4.95	17908.0	United Kingdom
555	536412	22327	ROUND SNACK BOXES SET OF 4 SKULLS	1	2010-01-12 11:49:00	2.95	17920.0	United Kingdom
...	...	...	...	...	...	...	...	...
541675	581538	22068	BLACK PIRATE TREASURE CHEST	1	2011-09-12 11:34:00	0.39	14446.0	United Kingdom
541689	581538	23318	BOX OF 6 MINI VINTAGE CRACKERS	1	2011-09-12 11:34:00	2.49	14446.0	United Kingdom
541692	581538	22992	REVOLVER WOODEN RULER	1	2011-09-12 11:34:00	1.95	14446.0	United Kingdom
541699	581538	22694	WICKER STAR	1	2011-09-12 11:34:00	2.10	14446.0	United Kingdom
541701	581538	23343	JUMBO BAG VINTAGE CHRISTMAS	1	2011-09-12 11:34:00	2.08	14446.0	United Kingdom

5268 rows × 8 columns

g. Correlation – Heatmap - All Numeric Variables

In [10]:
```python
sns.heatmap(df.corr())
```

Out[10]:
```
<AxesSubplot:>
```

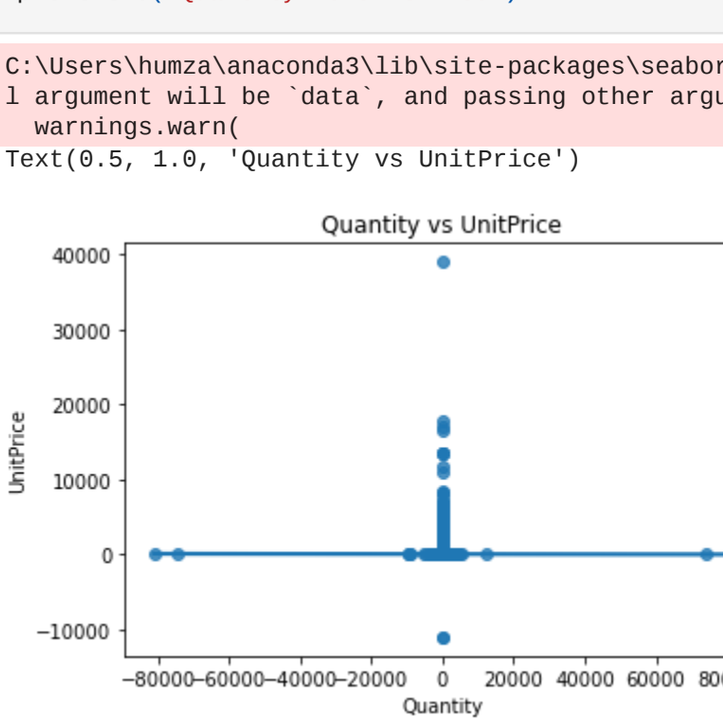


h. Regression Plot - All Numeric Variables

In [11]:
```python
sns.regplot(df['Quantity'],df['UnitPrice'])
plt.title('Quantity vs UnitPrice')
```

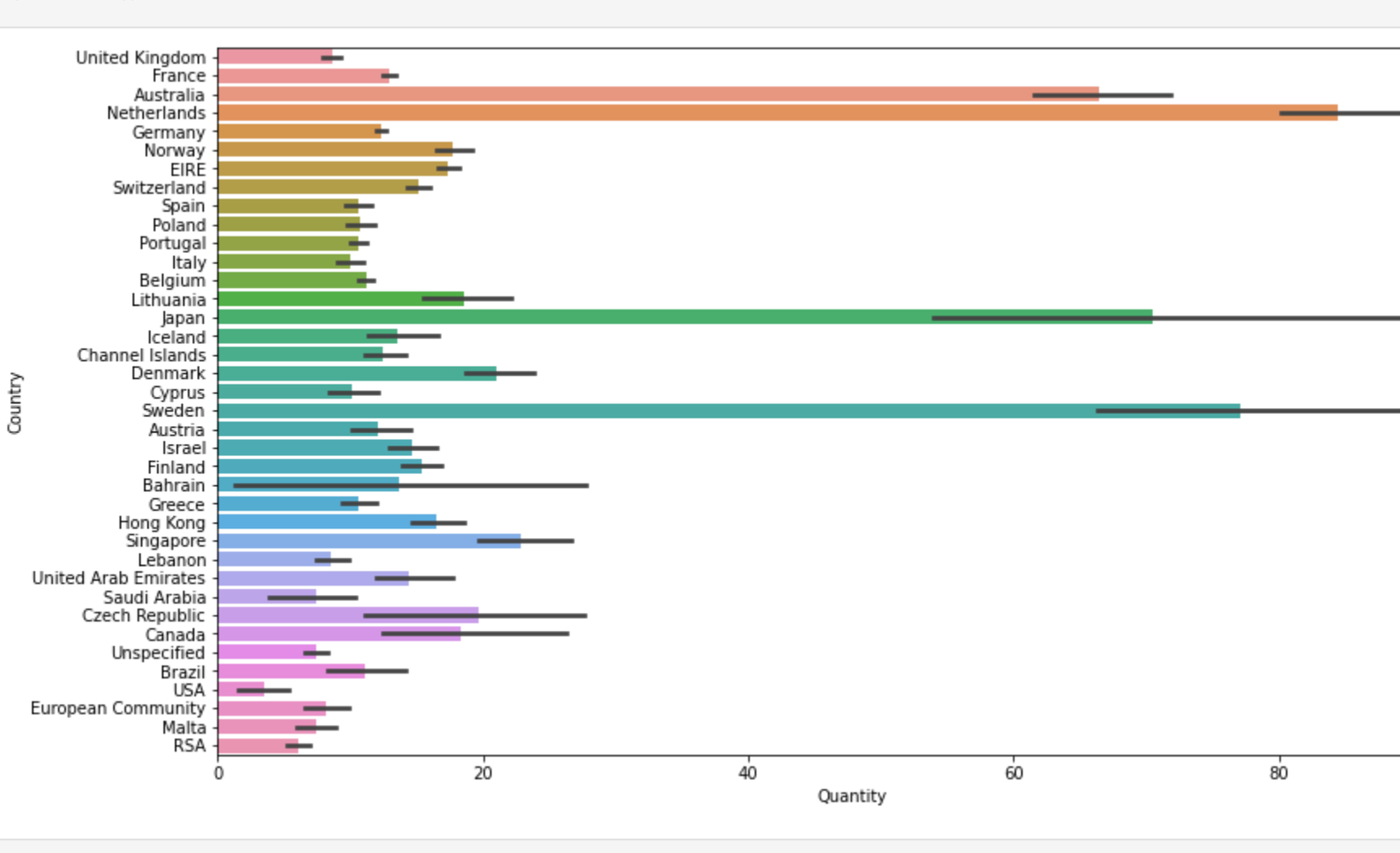
```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[11]:
```
Text(0.5, 1.0, 'Quantity vs UnitPrice')
```

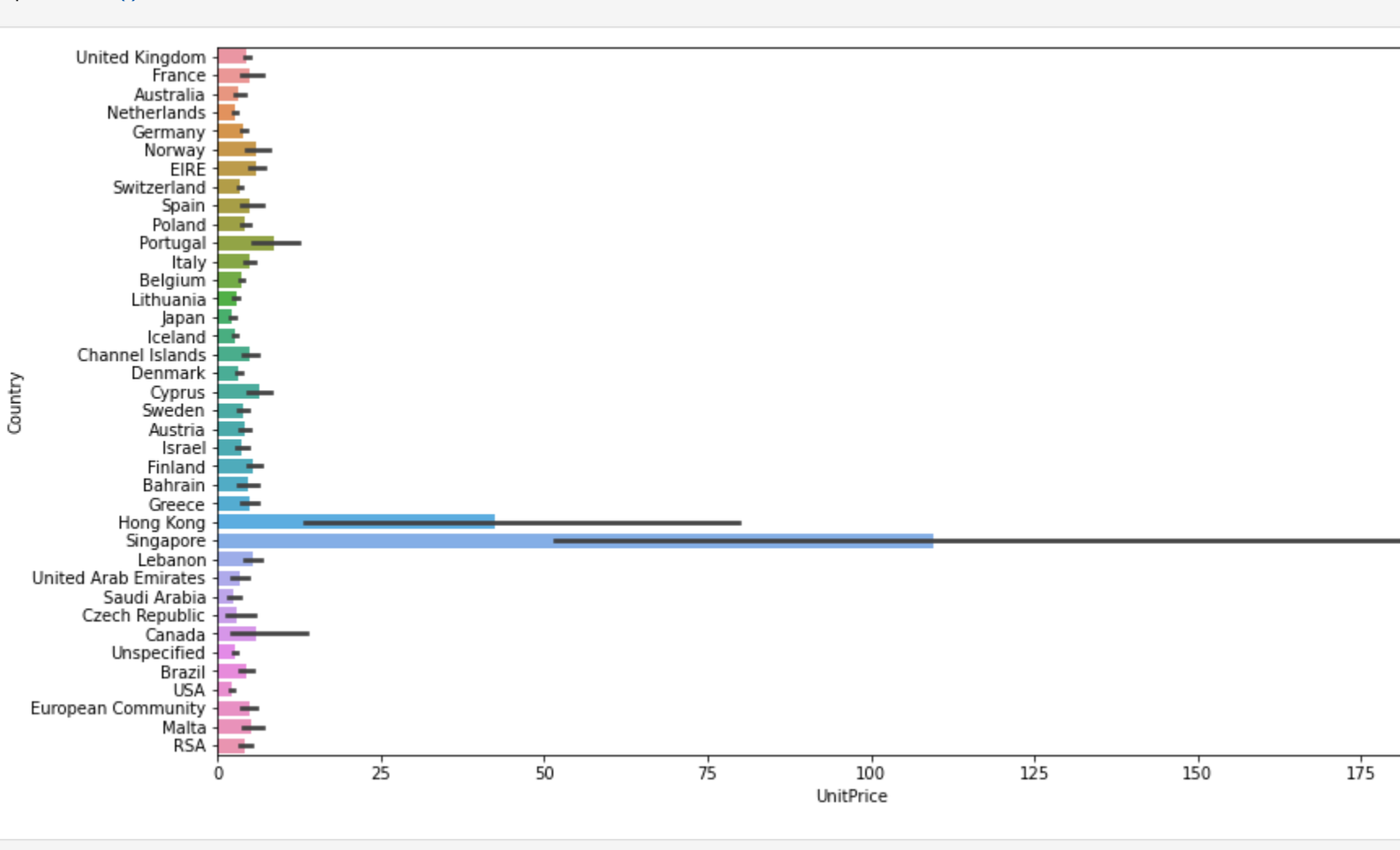


i. Bar Plot – Every Categorical Variable vs every Numerical Variable

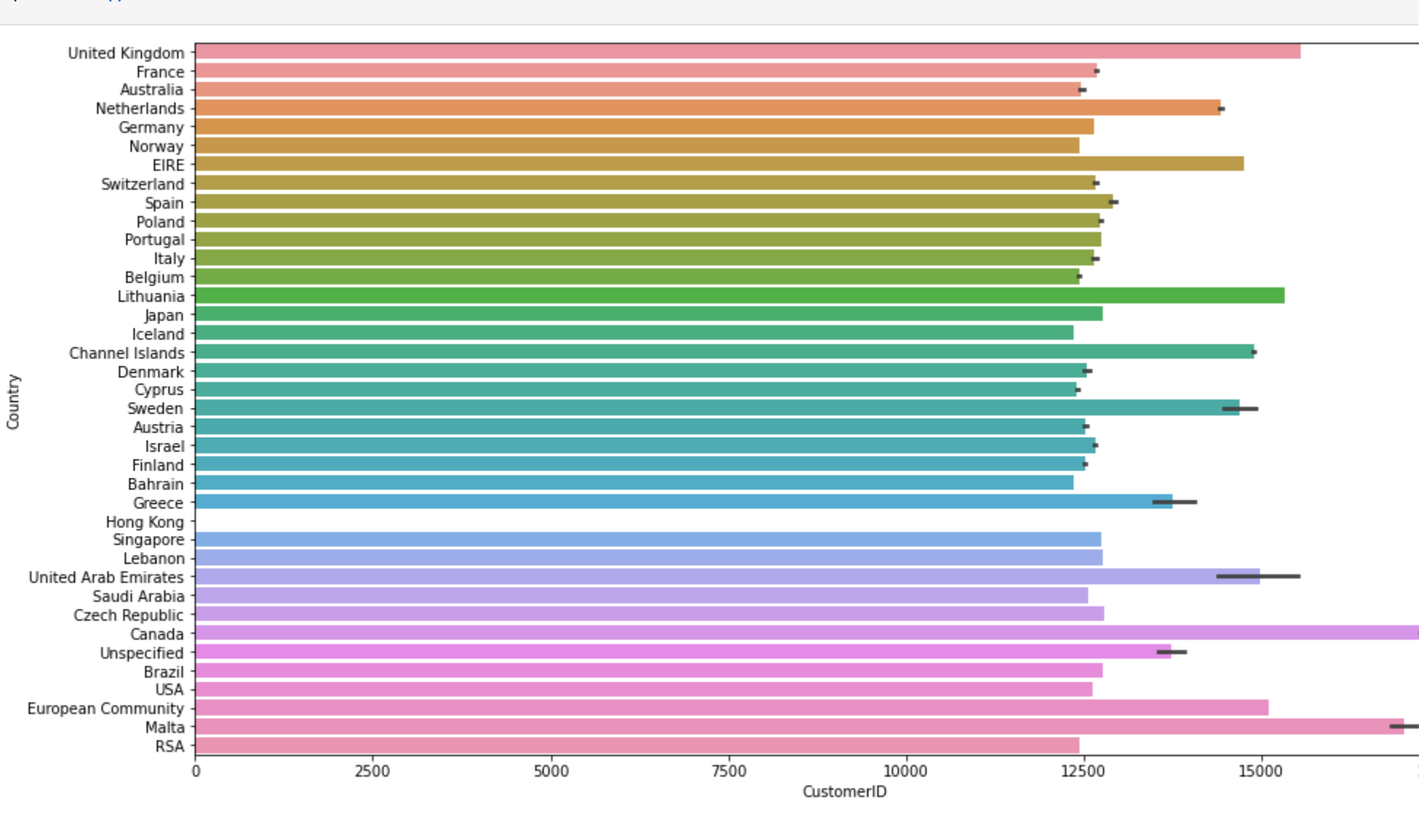
In [12]:
```python
plt.figure(figsize=(14,8))
sns.barplot(data=df,y='Country',x='Quantity',orient='h')
plt.show()
```



In [13]:
```python
plt.figure(figsize=(14,8))
sns.barplot(data=df,y='Country',x='UnitPrice',orient='h')
plt.show()
```



In [14]:
```python
plt.figure(figsize=(16,9))
sns.barplot(data=df,y='Country',x='CustomerID',orient='h')
plt.show()
```



j. Pair plot - All Numeric Variables

Not Working

In [15]:
```python
#sns.pairplot(df,vars=['Quantity','UnitPrice'])
#plt.show()
```

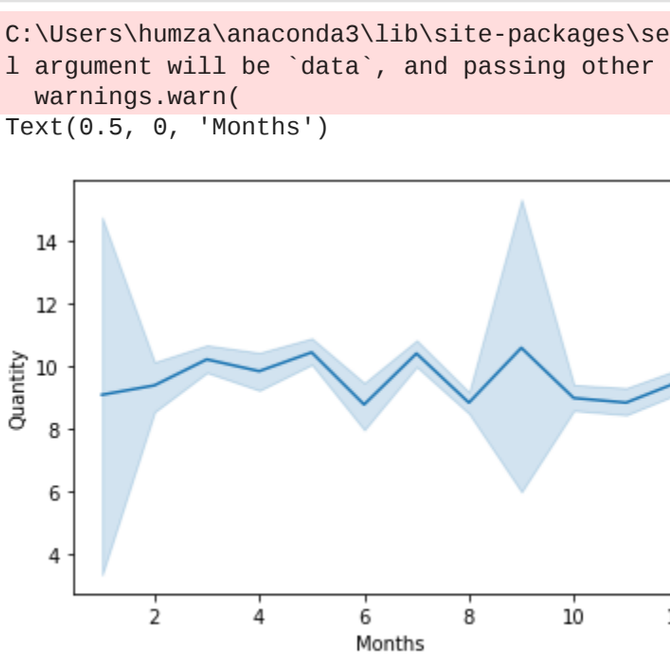
k. Line chart to show the trend of data - All Numeric/Date Variables

In [16]:
```python
df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
```

In [17]:
```python
sns.lineplot(df['InvoiceDate'].dt.month,df['Quantity'])
plt.xlabel('Months')
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

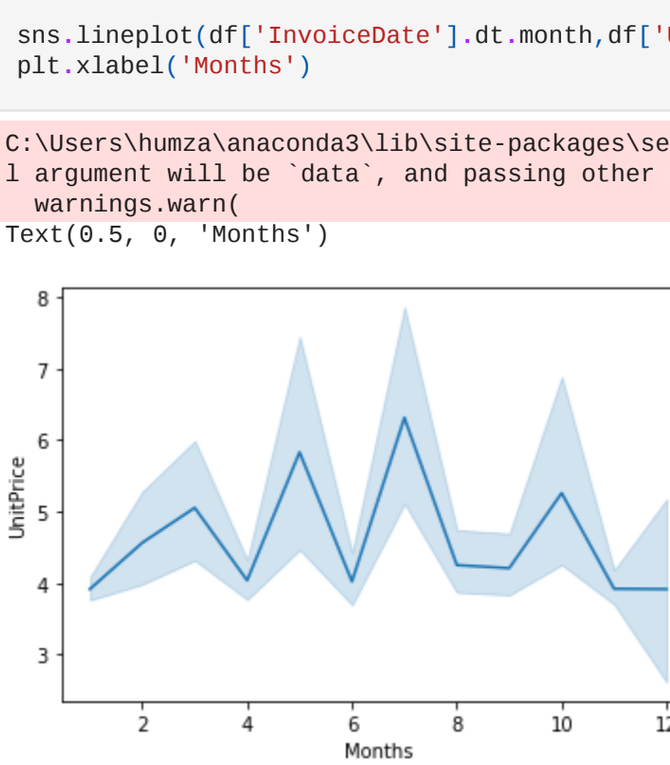
Out[17]:
```
Text(0.5, 0, 'Months')
```



In [18]:
```python
sns.lineplot(df['InvoiceDate'].dt.month,df['UnitPrice'])
plt.xlabel('Months')
```

```
C:\Users\humza\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[18]:
```
Text(0.5, 0, 'Months')
```



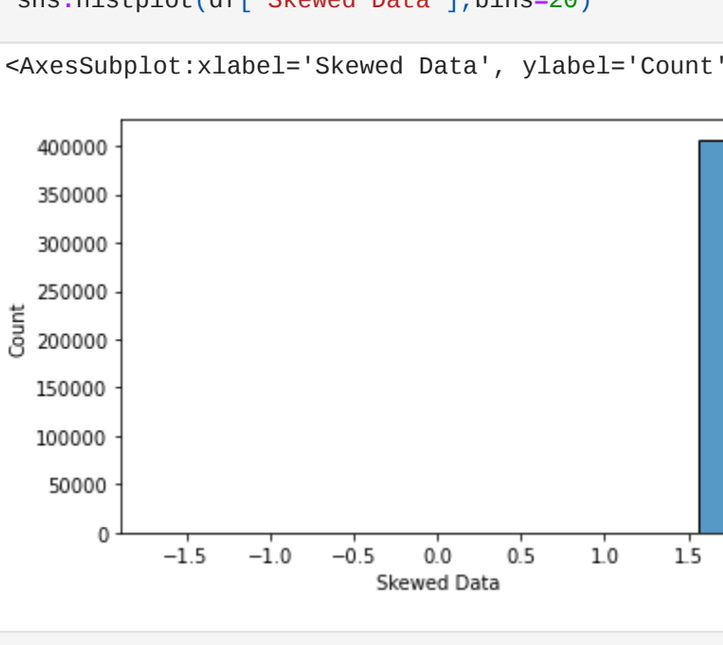
l. Plot the skewness - All Numeric Variables

In [19]:
```python
df['Skewed Data'] = pd.DataFrame(df.skew(axis=1,skipna=True))
```

```
C:\Users\humza\AppData\Local\Temp/ipykernel_13320/147960197.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df['Skewed Data'] = pd.DataFrame(df.skew(axis=1,skipna=True))
```

In [20]:
```python
sns.histplot(df['Skewed Data'],bins=20)
```

Out[20]:
```
<AxesSubplot:xlabel='Skewed Data', ylabel='Count'>
```



In [21]:
```python
df.drop('Skewed Data',axis=1,inplace=True)
```

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

(Mean/Median/Mode)

In [22]:
```python
df.isnull().sum()
```

Out[22]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [23]:
```python
df[df['Quantity']==0]
```

Out[23]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

In [24]:
```python
df[df['UnitPrice']==0]
```

Out[24]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	2010-01-12 11:52:00	0.0	NaN	United Kingdom
1970	536545	21134	NaN	1	2010-01-12 14:32:00	0.0	NaN	United Kingdom
1971	536546	22145	NaN	1	2010-01-12 14:33:00	0.0	NaN	United Kingdom
1972	536547	37509	NaN	1	2010-01-12 14:33:00	0.0	NaN	United Kingdom
1987	536549	85226A	NaN	1	2010-01-12 14:34:00	0.0	NaN	United Kingdom
...	...	...	...	...	...	...	...	...
536981	581234	72817	NaN	27	2011-08-12 10:33:00	0.0	NaN	United Kingdom
538504	581406	46000M	POLYESTER FILLER PAD 45x45cm	240	2011-08-12 13:58:00	0.0	NaN	United Kingdom
538505	581406	46000S	POLYESTER FILLER PAD 40x40cm	300	2011-08-12 13:58:00	0.0	NaN	United Kingdom
538554	581408	85175	NaN	20	2011-08-12 14:06:00	0.0	NaN	United Kingdom
538919	581422	23169	smashed	-235	2011-08-12 15:24:00	0.0	NaN	United Kingdom

2515 rows × 8 columns

Replacing the missing values of the 'Description' column with the Mode,since it is a Categorical Column:-

In [25]:
```python
df['Description'].fillna(df['Description'].mode()[0],inplace=True)
```

Replacing the missing values of the 'CustomerID' column with the Mode,since it is a Discrete Column:-

In [26]:
```python
df['CustomerID'].fillna(df['CustomerID'].mode()[0],inplace=True)
```

Since the 'UnitPrice' column has some 0.0 values which could not be detected with df.isnull().sum. So we need to replace such values with the Median of the column.

In [27]:
```python
df['UnitPrice'].replace(0.0,df['UnitPrice'].median(),inplace=True)
```

3. Remove duplicate rows

In [28]:
```python
df.drop(df[df.duplicated()].index,axis=0,inplace=True)
```

4. Remove rows which have negative values in Quantity column

In [29]:
```python
df.drop(df[df['Quantity']<0].index,axis=0,inplace=True)
```

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

In [30]:
```python
df['Month']=df['InvoiceDate'].dt.month
```

In [31]:
```python
df['Day']=df['InvoiceDate'].dt.day
```

In [32]:
```python
df['Hour']=df['InvoiceDate'].dt.hour
```

6. How many orders made by the customers?

In [33]:
```python
df.groupby('CustomerID')['InvoiceNo'].count().sort_values(ascending=False)
```

Out[33]:
```
CustomerID
17841.0    140998
14911.0      5672
14096.0      5111
12748.0      4413
14606.0      2677
            ...
15940.0         1
15823.0         1
15802.0         1
15753.0         1
12346.0         1
Name: InvoiceNo, Length: 4339, dtype: int64
```

7. TOP 5 customers with higher number of orders

In [34]:
```python
df.groupby('CustomerID')['InvoiceNo'].count().sort_values(ascending=False).head(5)
```

Out[34]:
```
CustomerID
17841.0    140998
14911.0      5672
14096.0      5111
12748.0      4413
14606.0      2677
Name: InvoiceNo, dtype: int64
```

8. How much money spent by the customers?

In [35]:
```python
df['Money_spent']=df['Quantity']*df['UnitPrice']
```

In [36]:
```python
df.groupby('CustomerID')['Money_spent'].sum()
```

Out[36]:
```
CustomerID
12346.0    77183.60
12347.0     4310.00
12348.0     1797.24
12349.0     1757.55
12350.0      334.40
             ...
18280.0      180.60
18281.0       80.82
18282.0      178.05
18283.0     2045.53
18287.0     1837.28
Name: Money_spent, Length: 4339, dtype: float64
```

9. TOP 5 customers with highest money spent

In [37]:
```python
df.groupby('CustomerID')['Money_spent'].sum().sort_values(ascending=False).head(5)
```

Out[37]:
```
CustomerID
17841.0    1895420.67
14646.0     281404.10
18102.0     259657.30
17450.0     194390.79
16446.0     168472.50
Name: Money_spent, dtype: float64
```

10. How many orders per month?

In [38]:
```python
df.groupby('Month')['InvoiceNo'].count()
```

Out[38]:
```
Month
1     38592
2     32968
3     37449
4     36107
5     41462
6     44532
7     45747
8     43608
9     49975
10    55238
11    71421
12    28955
Name: InvoiceNo, dtype: int64
```

11. How many orders per day?

In [39]:
```python
df.groupby('Day')['InvoiceNo'].count()
```

Out[39]:
```
Day
1     12547
2     10225
3     12209
4     11876
5     16049
6     14509
7     15069
8     14674
9     16645
10    22725
11    28603
12    50712
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
```

12. How many orders per hour?

In [40]:
```python
df.groupby('Hour')['InvoiceNo'].count().sort_values()
```

Out[40]:
```
Hour
6         1
7       379
20      778
19     3428
18     7696
8      8802
17    27498
9     33738
10    47670
16    53074
11    55592
14    65384
13    70067
15    75851
12    76096
Name: InvoiceNo, dtype: int64
```

13. How many orders for each country?

In [41]:
```python
df.groupby('Country')['InvoiceNo'].count().sort_values(ascending=False)
```

Out[41]:
```
Country
United Kingdom          481143
Germany                   9027
France                    8393
EIRE                      7883
Spain                     2480
Netherlands               2363
Belgium                   2031
Switzerland               1959
Portugal                  1492
Australia                 1184
Norway                    1072
Italy                      758
Channel Islands            747
Finland                    685
Cyprus                     603
Sweden                     450
Unspecified                442
Austria                    398
Denmark                    380
Poland                     330
Japan                      321
Israel                     292
Hong Kong                  280
Singapore                  222
Iceland                    182
USA                        179
Canada                     151
Greece                     145
Malta                      112
United Arab Emirates        68
European Community          60
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              25
Bahrain                     18
Saudi Arabia                 9
Name: InvoiceNo, dtype: int64
```

14. Orders trend across months

In [42]:
```python
plt.plot(df.groupby('Month')['InvoiceNo'].count())
plt.xlabel('Months')
plt.ylabel('No of Orders')
plt.title('Orders trend across months')
plt.show
```

Out[42]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

15. How much money spent by each country?

In [43]:
```python
df.groupby('Country')['Money_spent'].sum().sort_values(ascending=False)
```

Out[43]:
```
Country
United Kingdom          9128175.654
Netherlands              286644.420
EIRE                     283710.440
Germany                  228682.560
France                   209627.450
Australia                139094.450
Spain                     61581.440
Switzerland               57069.680
Belgium                   41196.340
Sweden                    38367.830
Japan                     37416.370
Norway                    36169.600
Portugal                  33683.050
Finland                   22546.080
Singapore                 21279.290
Channel Islands           20440.540
Denmark                   18955.340
Italy                     17483.240
Hong Kong                 15483.000
Cyprus                    13502.850
Austria                   10198.680
Israel                     8129.410
Poland                     7334.650
Greece                     4760.520
Unspecified                4740.940
Iceland                    4310.000
Canada                     3666.380
USA                        3580.390
Malta                      2725.590
United Arab Emirates       1902.280
Lebanon                    1693.880
Lithuania                  1661.060
European Community         1300.250
Brazil                     1143.600
RSA                        1004.390
Czech Republic              826.740
Bahrain                     754.140
Saudi Arabia                145.920
Name: Money_spent, dtype: float64
```