Problem Statement:-

The transactions made by a UK-based, registered, non-store online retailer between December 1, 2010, and December 9, 2011, are all included in the transnational data set known as online retail. The company primarily offers one-of-a-kind gifts for every occasion. The company has a large number of wholesalers as clients.Company ObjectiveUsing the global online retail dataset, we will design a clustering model and select the ideal group of clients for the business to target.

In [42]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore')
```

In [43]:
```
data =pd.read_csv(r"C:\Users\HP\OneDrive\Documents\online Retail.csv")
data
```

Out[43]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	09-12-2011 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	09-12-2011 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	09-12-2011 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	09-12-2011 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	09-12-2011 12:50	4.95	12680.0	France

541909 rows × 8 columns

In [44]:
```
data.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 [45]:
```
data.isnull().sum()
```

Out[45]:
```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
dtype: int64
```

In [46]:
```
data.fillna(method='ffill',inplace=True)
```

In [47]:
```
1 data.isnull().sum()
```

Out[47]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

In [48]:
```
1 data.columns
```

Out[48]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [49]:
```
1 data['Country'].value_counts()
```

Out[49]:
```
Country
United Kingdom          495478
Germany                   9495
France                    8557
EIRE                      8196
Spain                     2533
Netherlands               2371
Belgium                   2069
Switzerland               2002
Portugal                  1519
Australia                 1259
Norway                    1086
Italy                      803
Channel Islands            758
Finland                    695
Cyprus                     622
Sweden                     462
Unspecified                446
Austria                    401
Denmark                    389
Japan                      358
Poland                     341
Israel                     297
USA                        291
Hong Kong                  288
Singapore                  229
Iceland                    182
Canada                     151
Greece                     146
Malta                      127
United Arab Emirates        68
European Community          61
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              30
Bahrain                     19
Saudi Arabia                10
Name: count, dtype: int64
```

In [50]:
```
r=pd.crosstab(data['Quantity'],data['UnitPrice'])
print(r)
```

```
UnitPrice  -11062.060   0.000       0.001       0.010       0.030          
Quantity                                                                   
-80995              0           0           0           0           0  \
-74215              0           0           0           0           0   
-9600               0           2           0           0           0   
-9360               0           0           0           0           1   
-9058               0           1           0           0           0   
...               ...         ...         ...         ...         ...   
 4800               0           0           0           0           0   
 5568               0           1           0           0           0   
 12540              0           1           0           0           0   
 74215              0           0           0           0           0   
 80995              0           0           0           0           0   

UnitPrice   0.040       0.060       0.070       0.080       0.090       ...
Quantity                                                                ...
-80995              0           0           0           0           0  ...  \
-74215              0           0           0           0           0  ...   
-9600               0           0           0           0           0  ...   
-9360               0           0           0           0           0  ...   
-9058               0           0           0           0           0  ...   
...               ...         ...         ...         ...         ...  ...   
 4800               0           0           0           0           0  ...   
 5568               0           0           0           0           0  ...   
 12540              0           0           0           0           0  ...   
 74215              0           0           0           0           0  ...   
 80995              0           0           0           0           0  ...   

UnitPrice   8142.750    8286.220    11062.060   11586.500   13474.790          
Quantity                                                                   
-80995              0           0           0           0           0  \
-74215              0           0           0           0           0   
-9600               0           0           0           0           0   
-9360               0           0           0           0           0   
-9058               0           0           0           0           0   
...               ...         ...         ...         ...         ...   
 4800               0           0           0           0           0   
 5568               0           0           0           0           0   
 12540              0           0           0           0           0   
 74215              0           0           0           0           0   
 80995              0           0           0           0           0   

UnitPrice   13541.330   16453.710   16888.020   17836.460   38970.000          
Quantity                                                                   
-80995              0           0           0           0           0   
-74215              0           0           0           0           0   
-9600               0           0           0           0           0   
-9360               0           0           0           0           0   
-9058               0           0           0           0           0   
...               ...         ...         ...         ...         ...   
 4800               0           0           0           0           0   
 5568               0           0           0           0           0   
 12540              0           0           0           0           0   
 74215              0           0           0           0           0   
 80995              0           0           0           0           0   

[722 rows x 1630 columns]
```

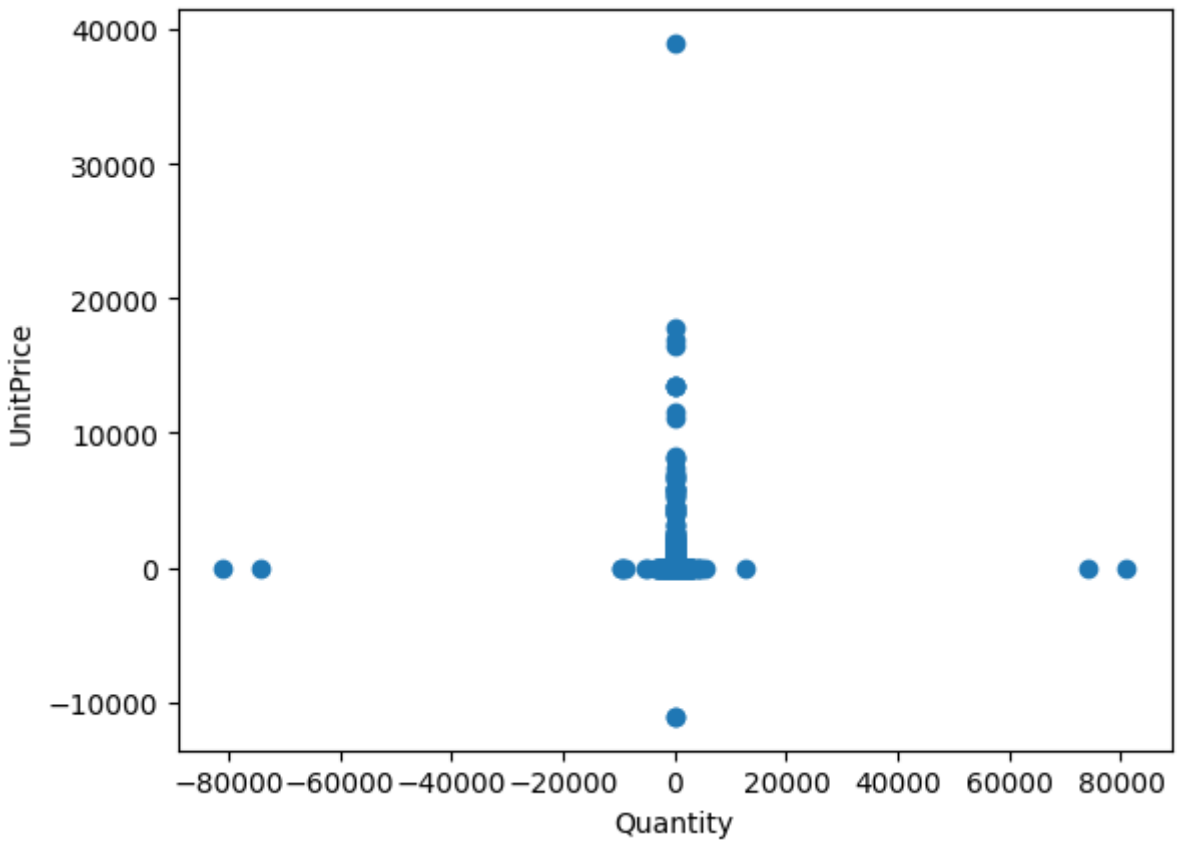
In [51]:
```
x=data[['Quantity','UnitPrice']]
y=data[['CustomerID']]
```

In [52]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [53]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
print(lr.score(x_train,y_train))
```

```
-2.2239827190961847e-06
7.963649773068404e-06
```

In [54]:
```
plt.scatter(data['Quantity'],data['UnitPrice'])
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.show()
```



In [55]:
```
from sklearn.cluster import KMeans
km=KMeans()
km
```

Out[55]: KMeans()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:
```
y_predicted  = km.fit_predict(data[['Quantity','UnitPrice']])
y_predicted
```

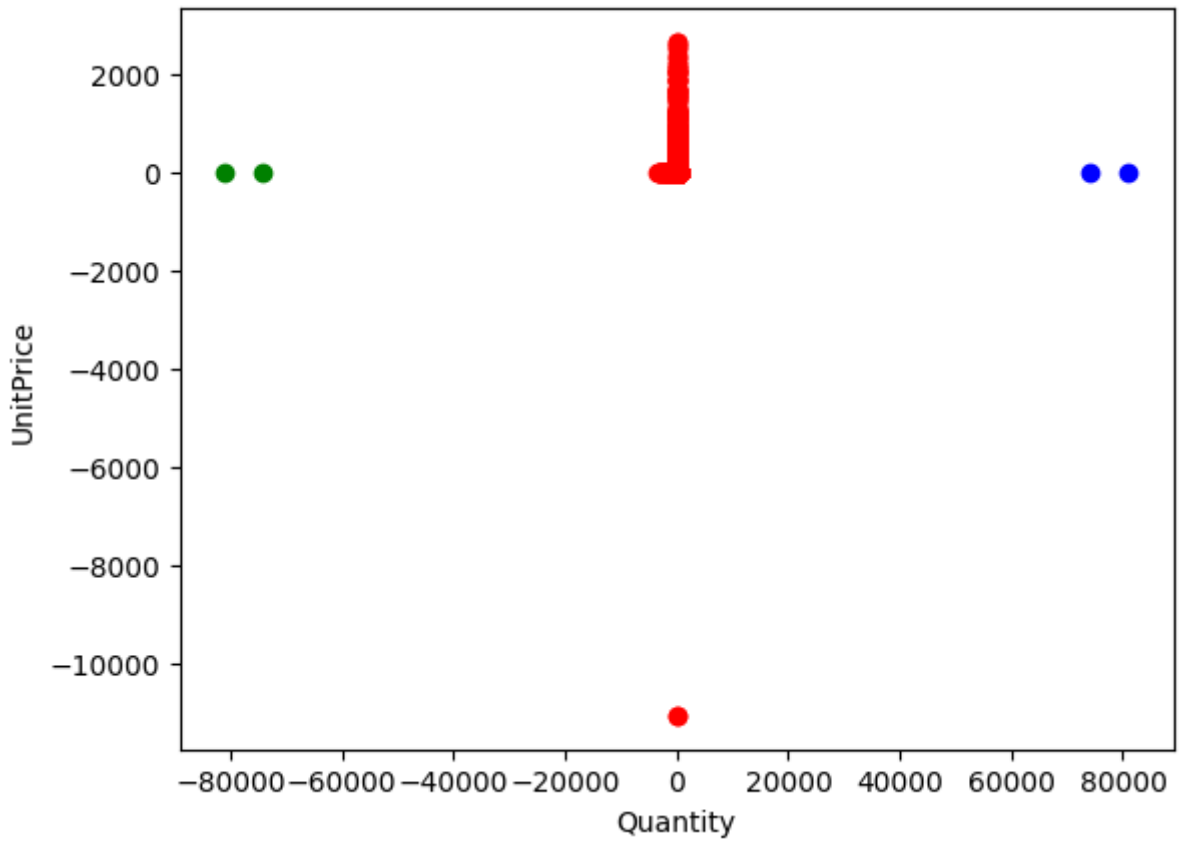
Out[56]: array([0, 0, 0, ..., 0, 0, 0])

In [57]:
```
data['Cluster']=y_predicted
data.head()
```

Out[57]:

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

In [58]:
```
data1 = data[data.Cluster == 0]
data2 = data[data.Cluster == 1]
data3 = data[data.Cluster == 2]
plt.scatter(data1['Quantity'],data1['UnitPrice'],color="red")
plt.scatter(data2['Quantity'],data2['UnitPrice'],color="green")
plt.scatter(data3['Quantity'],data3['UnitPrice'],color="blue")
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.show()
```



In [59]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [60]:
```
scaler=MinMaxScaler()
```

In [61]:
```
scaler.fit(data[["UnitPrice"]])
data["UnitPrice"]=scaler.transform(data[["UnitPrice"]])
data.head()
```

Out[61]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Cluster
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	0.221150	17850.0	United Kingdom	0
1	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	0.221154	17850.0	United Kingdom	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0

In [62]:
```
scaler.fit(data[["Quantity"]])
data["Quantity"]=scaler.transform(data[["Quantity"]])
data.head()
```

Out[62]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Cluster
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	0.500037	01-12-2010 08:26	0.221150	17850.0	United Kingdom	0
1	536365	71053	WHITE METAL LANTERN	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	0.500049	01-12-2010 08:26	0.221154	17850.0	United Kingdom	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0

In [63]:
```
km=KMeans()
```

In [64]:
```
y_predicted  = km.fit_predict(data[['Quantity','UnitPrice']])
y_predicted
```

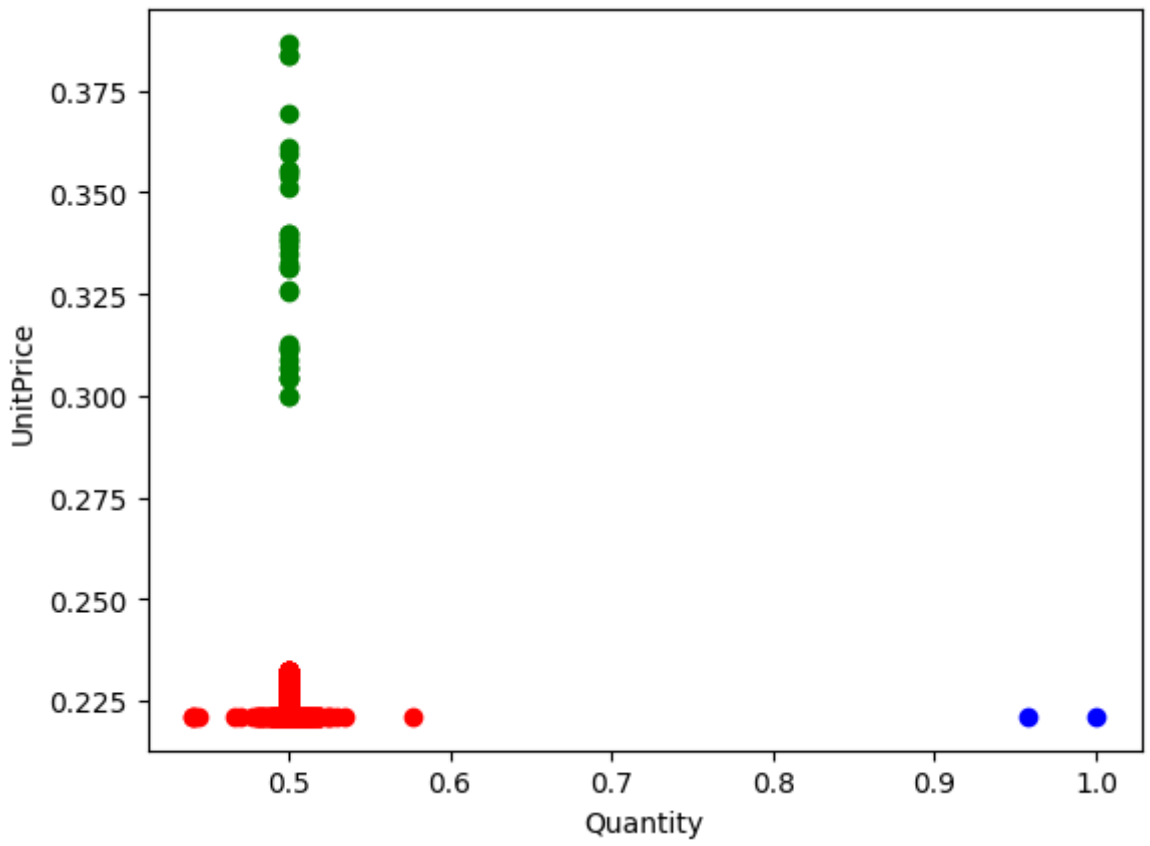
Out[64]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [65]:
```
data['New Cluster'] =y_predicted
data.head()
```

Out[65]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Cluster	New Cluster
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	0.500037	01-12-2010 08:26	0.221150	17850.0	United Kingdom	0	0
1	536365	71053	WHITE METAL LANTERN	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	0.500049	01-12-2010 08:26	0.221154	17850.0	United Kingdom	0	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	0.500037	01-12-2010 08:26	0.221167	17850.0	United Kingdom	0	0

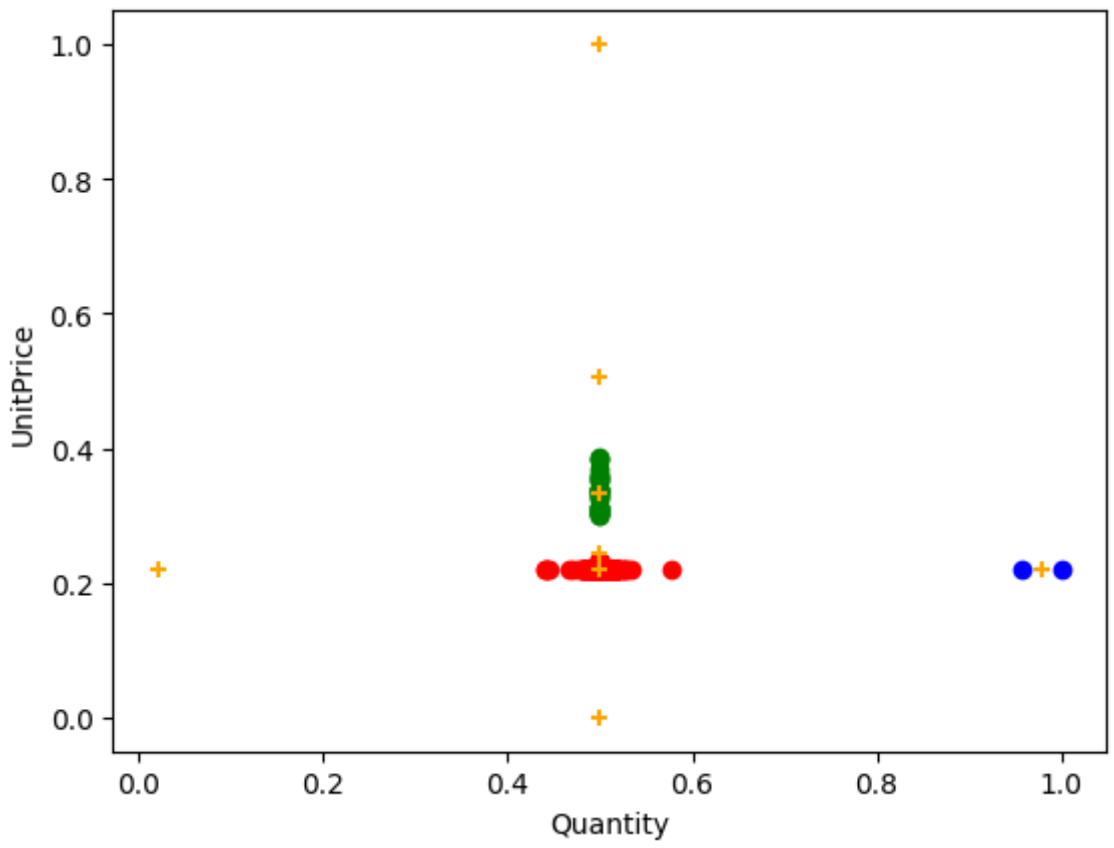
In [66]:
```
data1 = data[data['New Cluster'] == 0]
data2 = data[data['New Cluster'] == 1]
data3 = data[data['New Cluster'] == 2]
plt.scatter(data1['Quantity'],data1['UnitPrice'],color="red")
plt.scatter(data2['Quantity'],data2['UnitPrice'],color="green")
plt.scatter(data3['Quantity'],data3['UnitPrice'],color="blue")
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.show()
```



In [67]:
```
km.cluster_centers_
```

Out[67]:
```
array([[0.50005899, 0.22117195],
       [0.49999588, 0.33389462],
       [0.97907278, 0.22113061],
       [0.02092722, 0.22113061],
       [0.49999383, 1.        ],
       [0.49999657, 0.50519622],
       [0.50000432, 0.24394336],
       [0.50000617, 0.        ]])
```

In [68]:
```
data1 = data[data['New Cluster'] == 0]
data2 = data[data['New Cluster'] == 1]
data3 = data[data['New Cluster'] == 2]
plt.scatter(data1['Quantity'],data1['UnitPrice'],color="red")
plt.scatter(data2['Quantity'],data2['UnitPrice'],color="green")
plt.scatter(data3['Quantity'],data3['UnitPrice'],color="blue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="orange",marker = "+")
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.show()
```



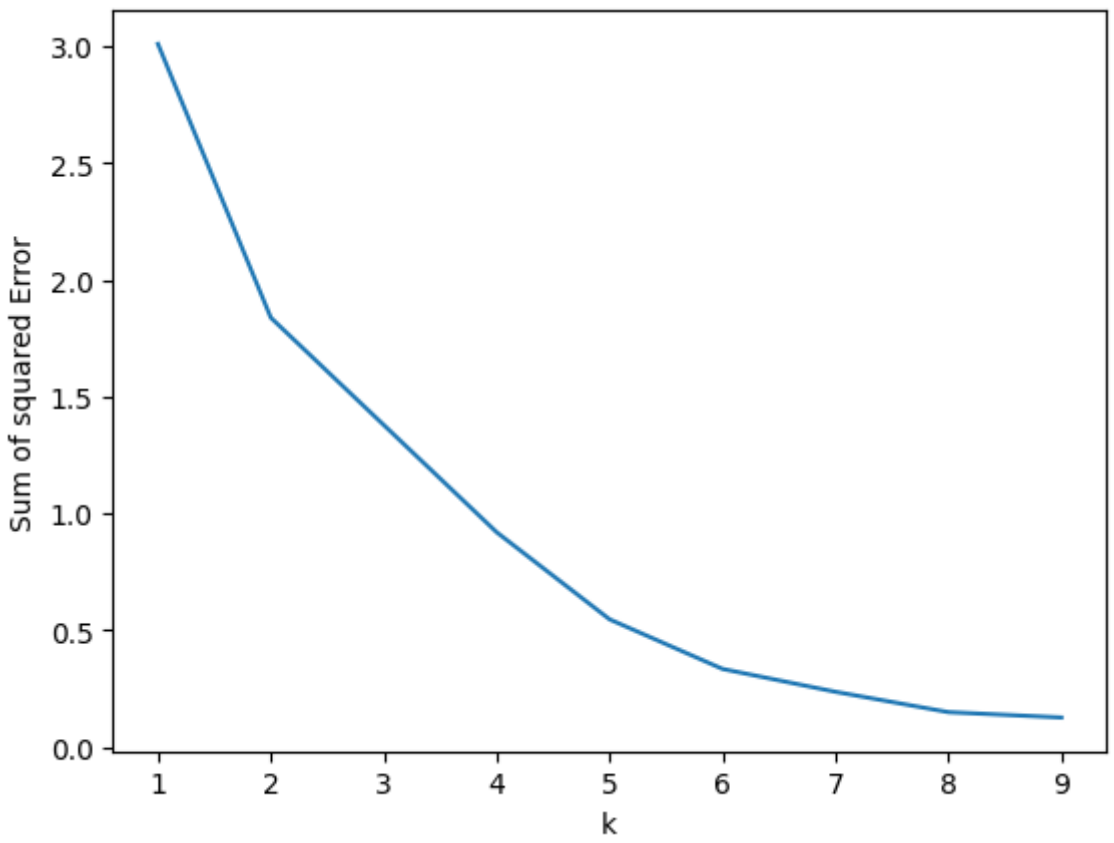
In [69]:
```
k_rng = range(1,10)
sse = []
for k in k_rng:
    km = KMeans(n_clusters = k)
    km.fit(data[["Quantity","UnitPrice"]])
    sse.append(km.inertia_)
sse

```

Out[69]:
```
[3.009005955427426,
 1.837504727956976,
 1.3783685528541396,
 0.9194617812522263,
 0.5467414071688541,
 0.33453989350844715,
 0.23669990686841533,
 0.14964381100039745,
 0.12673474424187725]
```

In [70]:
```
plt.plot(k_rng,sse)
plt.xlabel("k")
plt.ylabel("Sum of squared Error")
```

Out[70]: Text(0, 0.5, 'Sum of squared Error')



Conclusion:-

This Online Retail.csv DataFrame is done by using KMeans Clustering

In [ ]:
```
```