Importing Necessary packages

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

In [2]:
```
%matplotlib inline
```

Importing data set

In [3]:
```
src = pd.read_csv("Ecommerce - UK Retailer.csv", encoding="ISO-8859-1")
src.head()
```

Out[3]:

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

EDA

In [4]:
```
src.shape
```

Out[4]:
```
(541909, 8)
```

In [5]:
```
1 src.dtypes
```

Out[5]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

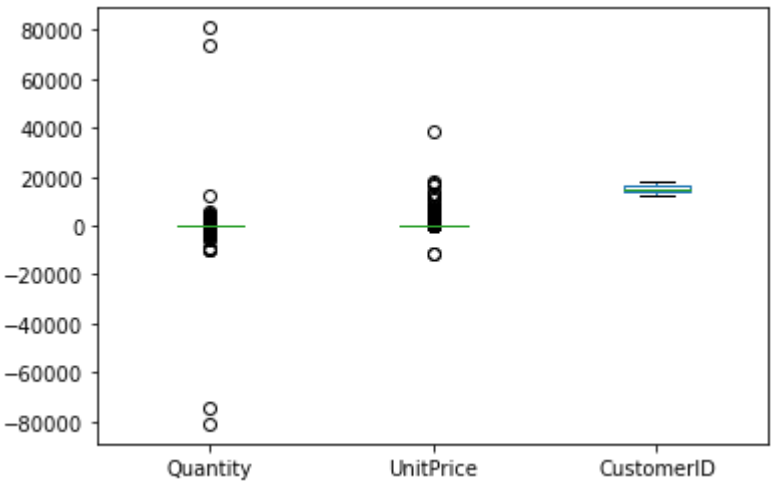
In [6]:
```
1 src['InvoiceNo'].unique()[:30]
```

Out[6]:
```
array(['536365', '536366', '536367', '536368', '536369', '536370',
       '536371', '536372', '536373', '536374', '536375', '536376',
       '536377', '536378', '536380', '536381', 'C536379', '536382',
       'C536383', '536384', '536385', '536386', '536387', '536388',
       '536389', '536390', 'C536391', '536392', '536393', '536394'],
      dtype=object)
```

1a Perform Basic EDA - Boxplot – All Numeric Variables

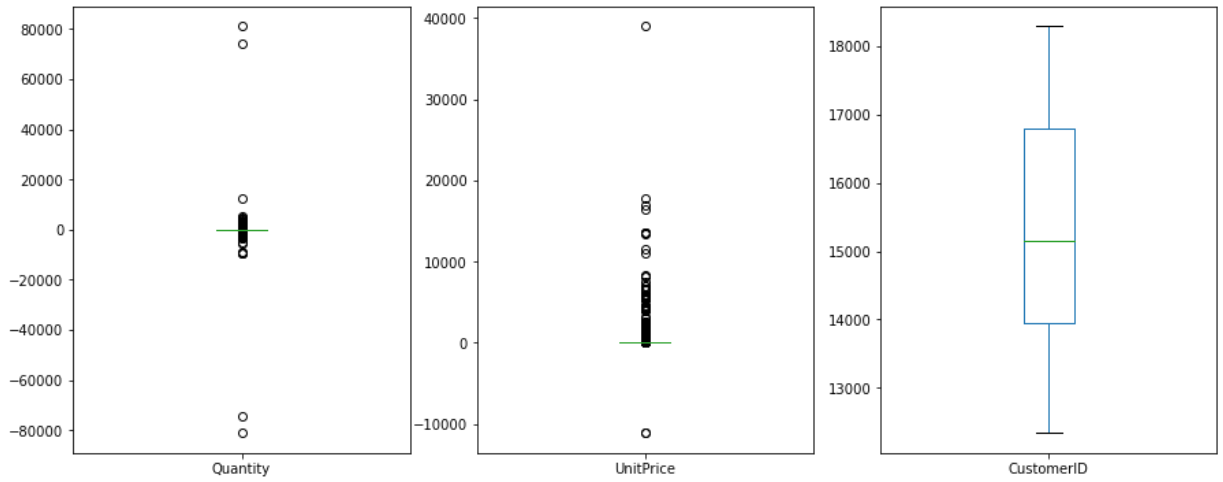
In [7]:
```
1 src[src.select_dtypes(np.number).columns].plot(kind = 'box')
```

Out[7]:
```
<AxesSubplot:>
```



In [8]:
```
plt.figure(figsize = (15,6))

plt.subplot(1, 3, 1)
src['Quantity'].plot(kind = 'box')

plt.subplot(1, 3, 2)
src['UnitPrice'].plot(kind = 'box')

plt.subplot(1, 3, 3)
src['CustomerID'].plot(kind = 'box')
```

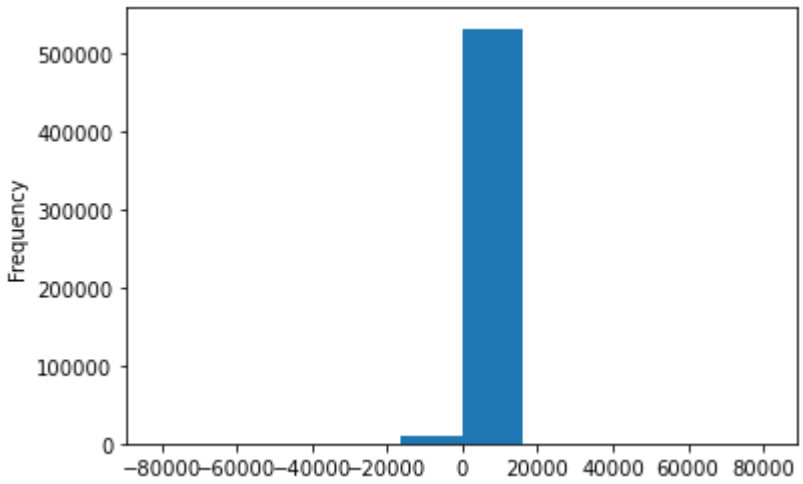
Out[8]: <AxesSubplot:>



1b Perform Basic EDA - Histplot – All Numeric Variables

In [9]:
```
1 src['Quantity'].plot(kind = 'hist')
```

Out[9]: <AxesSubplot:ylabel='Frequency'>

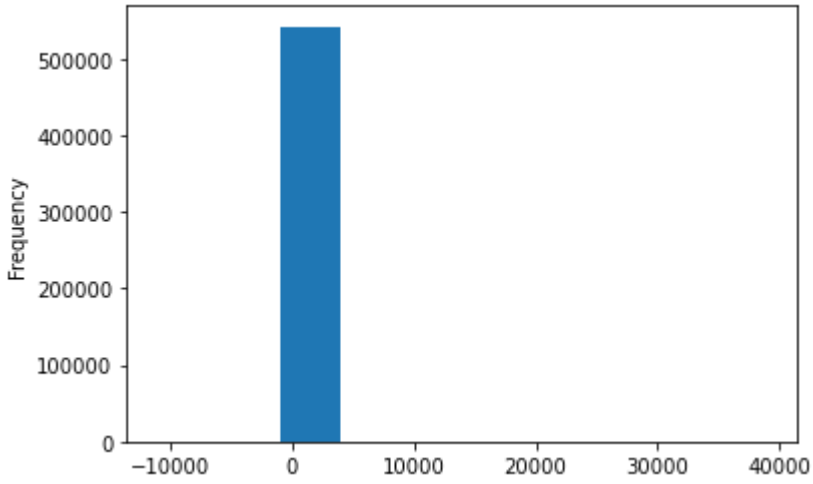


In [10]:
```
1 src[src['Quantity'] < 0].shape
```

Out[10]: (10624, 8)

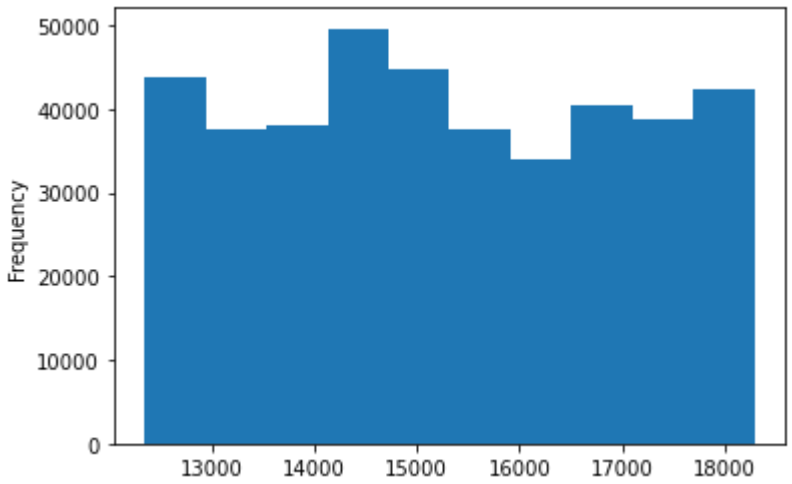
In [11]:
```
1 src['UnitPrice'].plot(kind = 'hist')
```

Out[11]: <AxesSubplot:ylabel='Frequency'>



In [12]:
```
1 src['CustomerID'].plot(kind = 'hist')
```

Out[12]:
```
<AxesSubplot:ylabel='Frequency'>
```



1c Perform Basic EDA - Distribution plot – All Numeric Variables

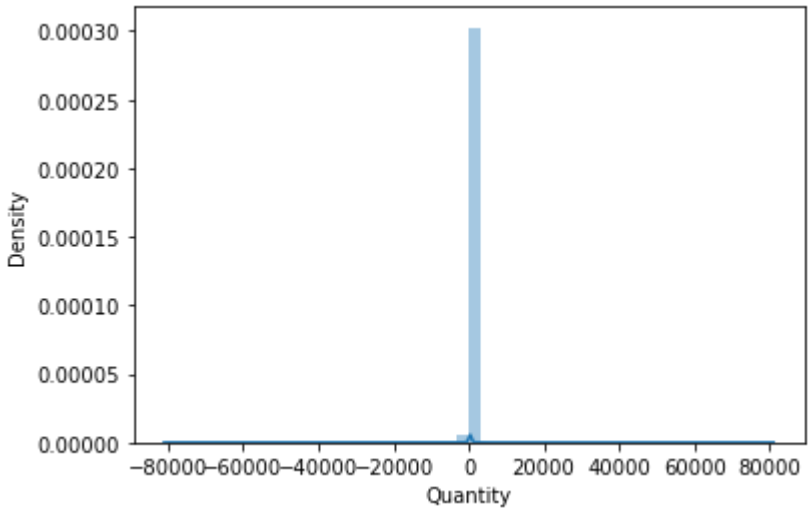
In [13]:
```
1 src.select_dtypes('number').columns
```

Out[13]:
```
Index(['Quantity', 'UnitPrice', 'CustomerID'], dtype='object')
```

In [14]:
```
1 sns.distplot(a = src['Quantity'])
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

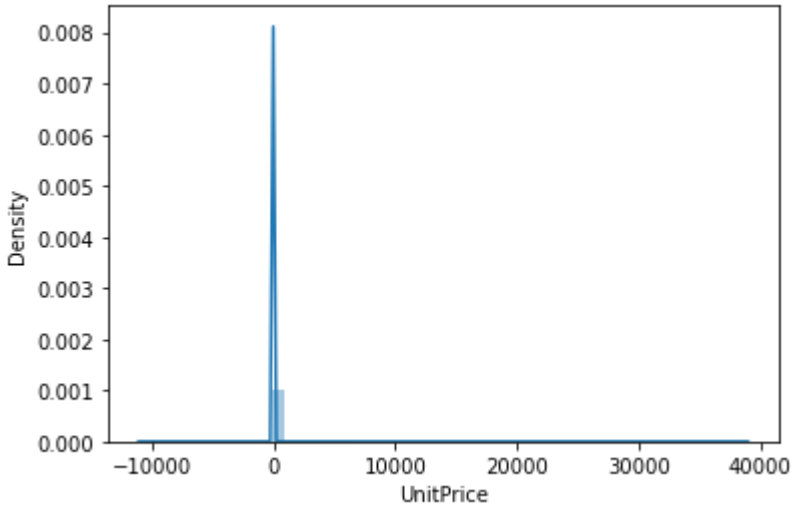
Out[14]:
```
<AxesSubplot:xlabel='Quantity', ylabel='Density'>
```



In [15]:
```
1 sns.distplot(a = src['UnitPrice'])
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

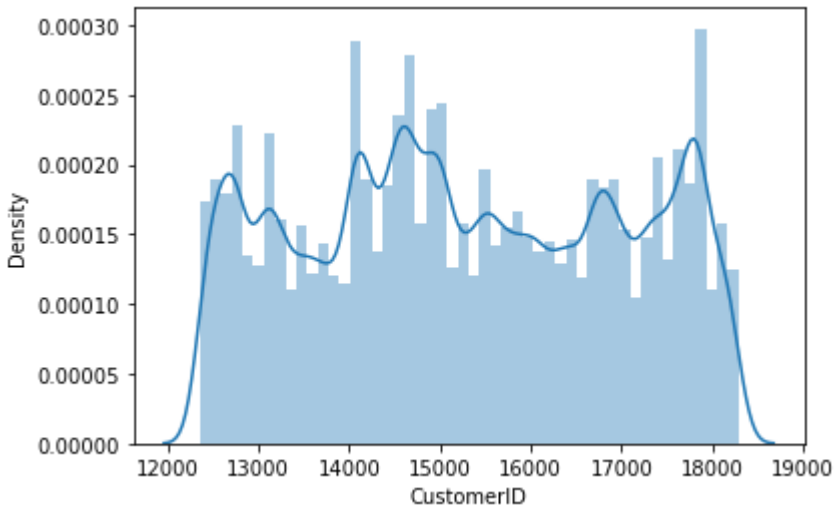
Out[15]: <AxesSubplot:xlabel='UnitPrice', ylabel='Density'>



In [16]:
```
1 sns.distplot(a = src['CustomerID'])
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[16]: <AxesSubplot:xlabel='CustomerID', ylabel='Density'>



1d Perform Basic EDA - Aggregation – All Numeric Variables

In [17]:
```
1 src.describe().round(2)
```

Out[17]:

	Quantity	UnitPrice	CustomerID
count	541909.00	541909.00	406829.00
mean	9.55	4.61	15287.69
std	218.08	96.76	1713.60
min	-80995.00	-11062.06	12346.00
25%	1.00	1.25	13953.00
50%	3.00	2.08	15152.00
75%	10.00	4.13	16791.00
max	80995.00	38970.00	18287.00

1e Perform Basic EDA - Unique Value – All Columns

In [18]:

```
for i in src.columns:
    print(i)
    print(src[i].unique())
    print('\n\n')
```

```
InvoiceNo
['536365' '536366' '536367' ... '581585' '581586' '581587']



StockCode
['85123A' '71053' '84406B' ... '90214U' '47591b' '23843']



Description
['WHITE HANGING HEART T-LIGHT HOLDER' 'WHITE METAL LANTERN'
 'CREAM CUPID HEARTS COAT HANGER' ... 'lost'
 'CREAM HANGING HEART T-LIGHT HOLDER' 'PAPER CRAFT , LITTLE BIRDIE']



Quantity
[     6      8      2     32      3      4     24     12     48     18
     20     36     80     64     10    120     96     23      5      1
     -1     50     40    100    192    432    144    288    -12    -24
     16      9    128     25     30     28      7     56     72    200
    600    480     -6     14     -2     11     33     13     -4     -5
     -7     -3     70    252     60    216    384    -10     27     15
     22     19     17     21     34     47    108     52  -9360    -38
     75    270     42    240     90    320   1824    204     69    -36
   -192   -144    160   2880   1400     39    -48    -50     26   1440
     31     82     78     97     98     35     57    -20    110    -22
    -30    -70   -130    -80   -120    -40    -25    -14    -15    -69
   -140   -320     -8    720    156    324     38     37     49     95
     -9    -11     29     41    -72    -35    -21    -43    -19    -18
    -44    402    378    150    300     54    104     67    258     66
     44     55     46     99     61    408    972    208   1008   1000
    -77   1488    250   1394    400   -223   -150    -13    -33   -723
   -177     79     84    -32   -100    -28    272   -145    -47    -96
    113     45    106     68    267    115     65   1728    -60    -16
     53   -240     76    460     71     43    213     58    576   2400
    500    180   -300   -500    -23    752    960   1296    210    172
    215    129    138    116    135    197   -106    -54    -17   -939
    147    168    256   -201    -53    -29  -2600   -990   -290    -45
    860   1010   1356   1284    186    114    360   1930   2000   3114
   1300    670    111    211     59   -310    -61    -41    176    648
     62  74215 -74215    -64    -84     89  -1400     73    -57    112
    456    -59    -31   5568   2560    136    900   -600    -42    -94
   -207    -52    130   -206   2592    420    800    101   1200    864
   -217     94  -1430   1287   -162   -230   -173   -390   -234    504
    123    118    -76   -200   1056   1500    280    407    141    124
    -99     51    -92   -741   3906   -400   -114    102   1152    -88
    198    117     86   -720    125    -86   -391    -87   -278    140
    228   -154  -3000     81   -675   -210   -345   -975  -1200  -1121
    -27   -541  -1277   -197   3000  -1100    -63  -5368    219   -259
```

```
   232    -82   -178    170   -110    768     88    700   -102   2160
  -323   -450   -232    -83   -153   -524  -2472    -49   -105    342
   304    167    -34    640    175    220     74     93    164    163
    63    637    122    158    165    350    -46    -75   -690    -39
   -66     83    312   -180    392   -624   -194    -26    -62    -37
   -91   -139   -158  -1479    -55   -576   -750    330    151    -93
  -432    -58  -1092   -670  -1300  -1930  -2000  -3114    462    -65
   -68    492   -620    273    -90   -170  -1512    -51     85    -56
  -160   -360    105   -960  -2376   1350    428  -1350    336   -786
  -590   -168   -101    -71    132   -413   -664    227   2700    222
   246    906    756    888    552    -97    224    738    608   -212
  4300    146    143    276   -125   -116   -108   4000   -304   -272
 -1206    -95   1600    323   -161   -472   -618   -204   1515  -1515
 -9058  -9600    660   -420   -126   -220   -271  -1440    264    188
   588    612    152   -324   -480   -164    -78   -118    430  -1681
    87    155    701    828    540    696    560   -250   -408   -179
   121   -124    512   -251  -3100   3100   -169    126    291   3186
 -2834   -109    109   -121   -530   -227    261   -346    352    142
   107   -188  -1060   -342   -288    348   1900    157   -343   -455
   425    968    684    824   -828   -701    196    248    410    236
   230   -156   -553    145    448    245   -252   -334   -318   -113
  -115    171   -242    840   -967   -203  -3167   -443  -1897    225
  -434    750   -682   -484    682    344   -635   -117  -3667    450
   310    494   -384     92   1788   -138    624    744    416    496
   396    306   1878   1944    666    708   1428    852   1412    528
  -756   -752   -152    -85   -312    -79   -147    -67   -131    183
  -209   -186   -231   -129   -458   -275  -2880   1540    672   -800
  -430   -380    -74   -840  -1296   -365   -104   -270    -73   -306
    91    255    468   -468   -111   -184   -103   -335   4800   -112
 -1000    912   1992    184    148   -657  -1671  -1158  -2618  -2003
  -674  -4830   -905  -1128    832    992    630   1020   2100    162
 -1560  -1284    -81    314    370    131    133    484    149    153
   257    139    137    628    179   1820    478    335    253    242
   375   -276   -256    281    193    181    404    244    207    199
  1130    326    654    688    268    249   -864    234  -1510   -550
  -244   -132   -327   -313    279   -398   -280    374   -504   -696
  -149   -224   -428    212   -267   -175  12540    760    -98    774
  -151   2040  -1131   -135   1120     77   -900    177    490    388
   620   1404   -155   -355   -337  -1050   -338    205   -235    698
 80995 -80995]
```

```
InvoiceDate
['12/1/2010 8:26' '12/1/2010 8:28' '12/1/2010 8:34' ... '12/9/2011 12:31'
 '12/9/2011 12:49' '12/9/2011 12:50']
```

```
UnitPrice
[   2.55    3.39    2.75 ...  933.17 1714.17  224.69]
```

```
CustomerID
[17850. 13047. 12583. ... 13298. 14569. 12713.]
```

```
Country
['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']
```

1f Perform Basic EDA - Duplicate Value – All Columns

In [19]:
```
1 src[src.duplicated()]
```

Out[19]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
517	536409	21866	UNION JACK FLAG LUGGAGE TAG	1	12/1/2010 11:45	1.25	17908.0	United Kingdom
527	536409	22866	HAND WARMER SCOTTY DOG DESIGN	1	12/1/2010 11:45	2.10	17908.0	United Kingdom
537	536409	22900	SET 2 TEA TOWELS I LOVE LONDON	1	12/1/2010 11:45	2.95	17908.0	United Kingdom
539	536409	22111	SCOTTIE DOG HOT WATER BOTTLE	1	12/1/2010 11:45	4.95	17908.0	United Kingdom
555	536412	22327	ROUND SNACK BOXES SET OF 4 SKULLS	1	12/1/2010 11:49	2.95	17920.0	United Kingdom
...	...	...	...	...	...	...	...	..
541675	581538	22068	BLACK PIRATE TREASURE CHEST	1	12/9/2011 11:34	0.39	14446.0	United Kingdom
541689	581538	23318	BOX OF 6 MINI VINTAGE CRACKERS	1	12/9/2011 11:34	2.49	14446.0	United Kingdom
541692	581538	22992	REVOLVER WOODEN RULER	1	12/9/2011 11:34	1.95	14446.0	United Kingdom
541699	581538	22694	WICKER STAR	1	12/9/2011 11:34	2.10	14446.0	United Kingdom
541701	581538	23343	JUMBO BAG VINTAGE CHRISTMAS	1	12/9/2011 11:34	2.08	14446.0	United Kingdom

5268 rows × 8 columns

1g Perform Basic EDA - Correlation: HeatMap – All Columns

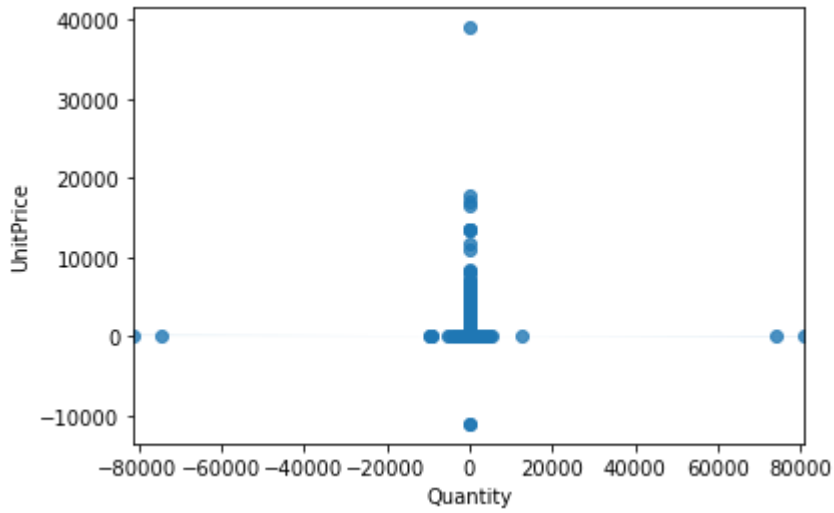
In [20]:
```
1 sns.heatmap(src.corr(), annot=True, cmap = 'Blues')
```

Out[20]: <AxesSubplot:>

1h Perform Basic EDA - Regression Plot - All Numeric Variables

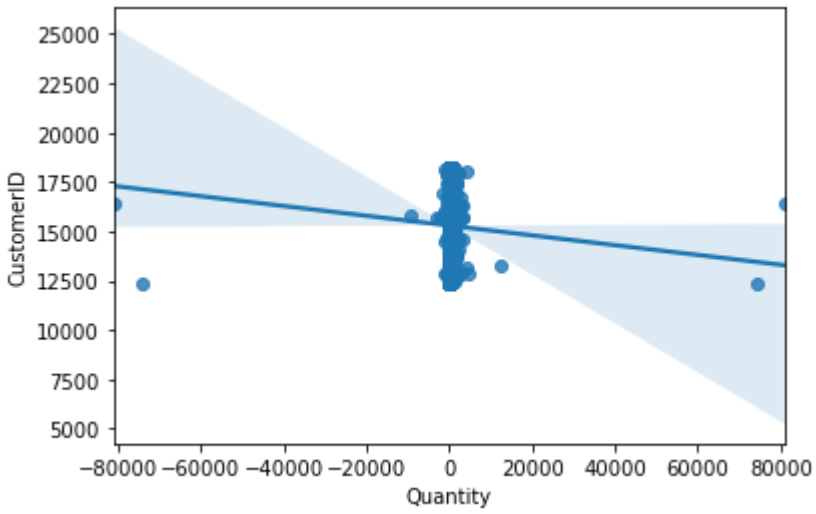
In [21]:
```
1 sns.regplot(x = 'Quantity', y = 'UnitPrice', data = src)
```

Out[21]: <AxesSubplot:xlabel='Quantity', ylabel='UnitPrice'>



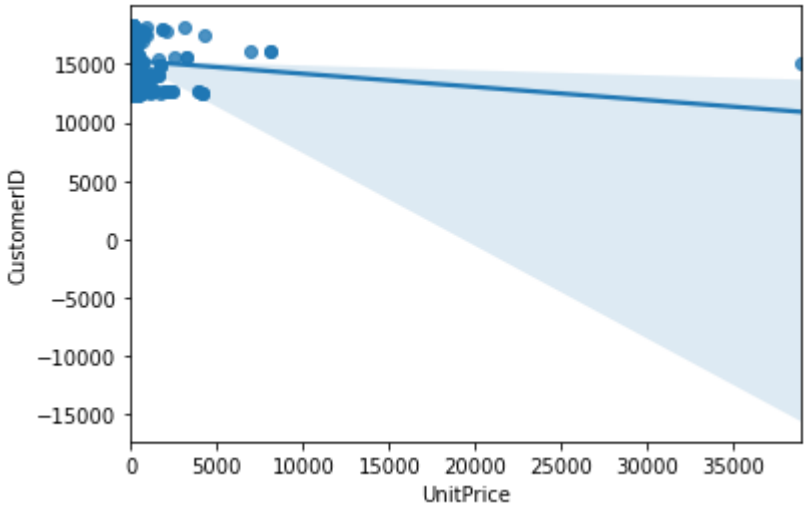
In [22]:
```
1 sns.regplot(x = 'Quantity', y = 'CustomerID', data = src)
```

Out[22]: <AxesSubplot:xlabel='Quantity', ylabel='CustomerID'>



In [23]:
```
1 sns.regplot(x = 'UnitPrice', y = 'CustomerID', data = src)
```

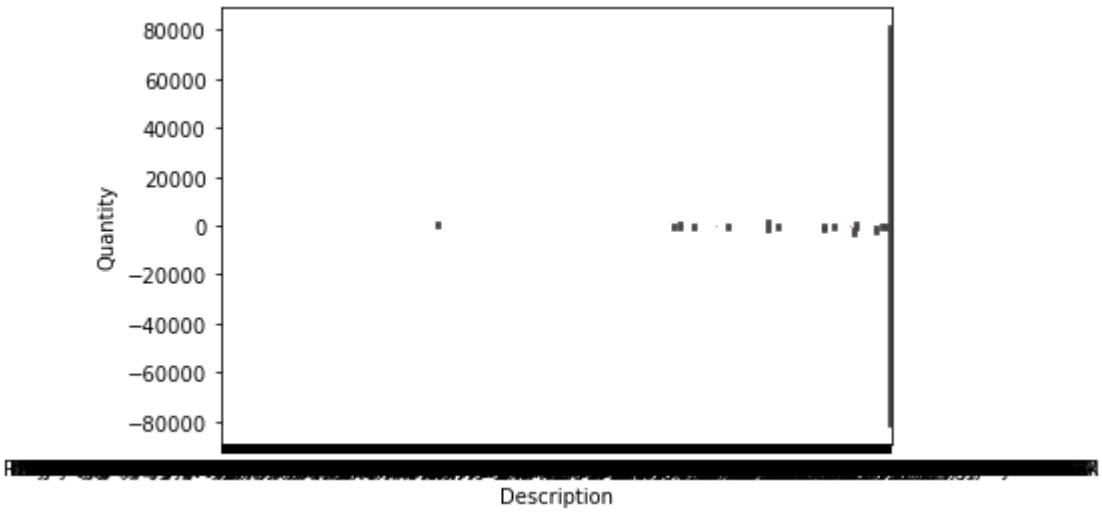
Out[23]: <AxesSubplot:xlabel='UnitPrice', ylabel='CustomerID'>



1i Perform Basic EDA - Bar Plot – Every Categorical Variable vs every Numerical Variable

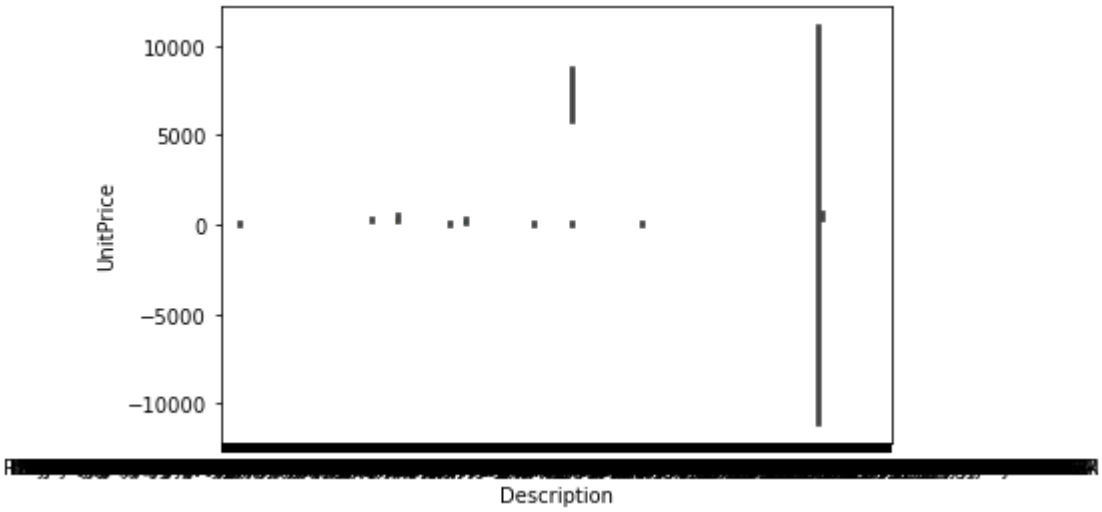
In [24]:
```
1 sns.barplot(x = 'Description', y = 'Quantity', data = src)
```

Out[24]: <AxesSubplot:xlabel='Description', ylabel='Quantity'>



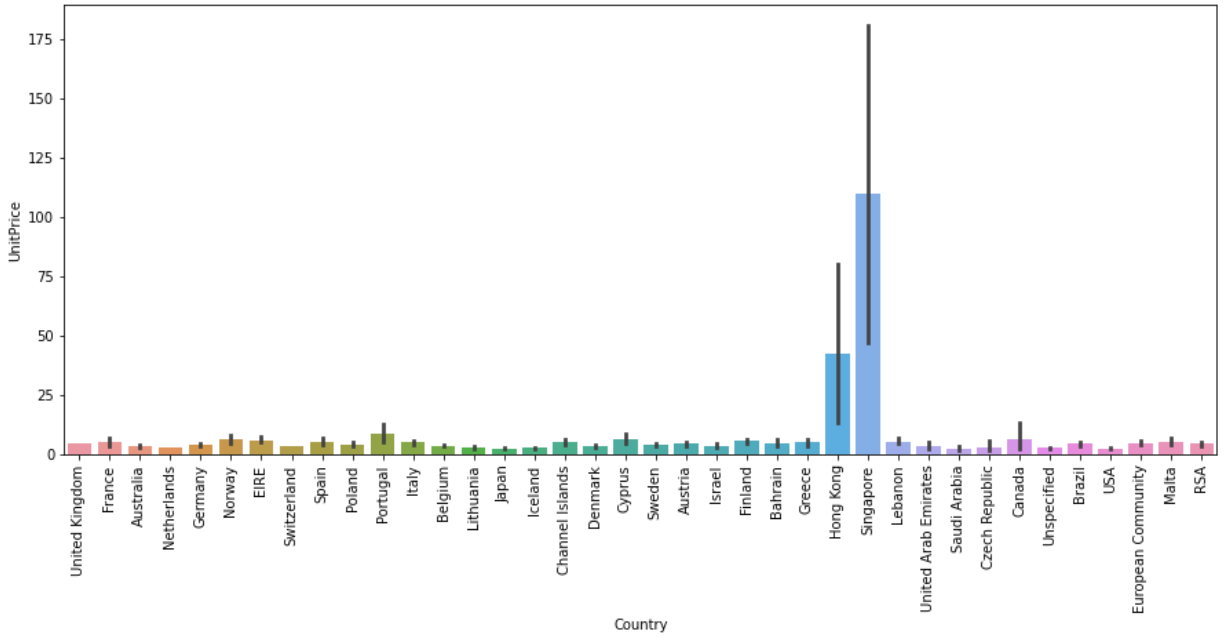
In [25]:
```
1 sns.barplot(x = 'Description', y = 'UnitPrice', data = src)
```

Out[25]: <AxesSubplot:xlabel='Description', ylabel='UnitPrice'>



In [27]:

```
plt.figure(figsize =(15, 6))
a = sns.barplot(x = 'Country', y = 'UnitPrice', data = src)
a.set_xticklabels(a.get_xticklabels(), rotation=90)
plt.show()
```



1j Perform Basic EDA - Pair Plot – Every Numerical Variable

In [56]:
```
1 sns.pairplot(src, vars = ['Quantity', 'UnitPrice'])
```

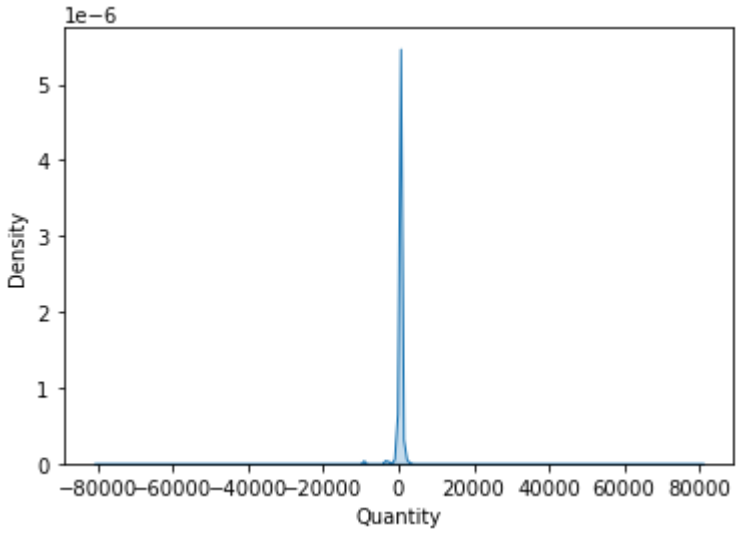
```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-56-007d79ba35f8> in <module>
----> 1 sns.pairplot(src, vars = ['Quantity', 'UnitPrice'])

~\Anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kwar
gs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)
})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~\Anaconda3\lib\site-packages\seaborn\axisgrid.py in pairplot(data, hue, hue_
order, palette, vars, x_vars, y_vars, kind, diag_kind, markers, height, aspec
t, corner, dropna, plot_kws, diag_kws, grid_kws, size)
   1959     diag_kws.setdefault("legend", False)
   1960     if diag_kind == "hist":
```

1k Perform Basic EDA - Plot the skewness – Every Numerical Variable

In [28]:
```
1 sns.kdeplot(src['Quantity'], shade = True)
```

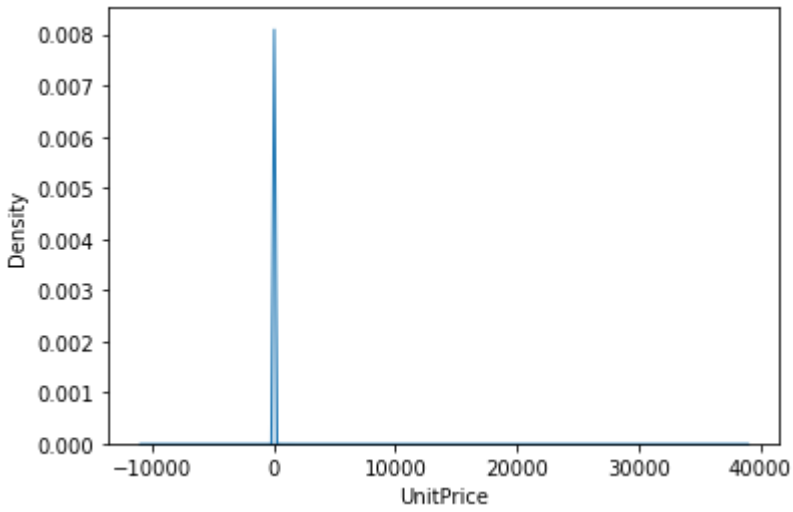
```
ERROR! Session/line number was not unique in database. History logging moved to
new session 1856
```

Out[28]:
```
<AxesSubplot:xlabel='Quantity', ylabel='Density'>
```



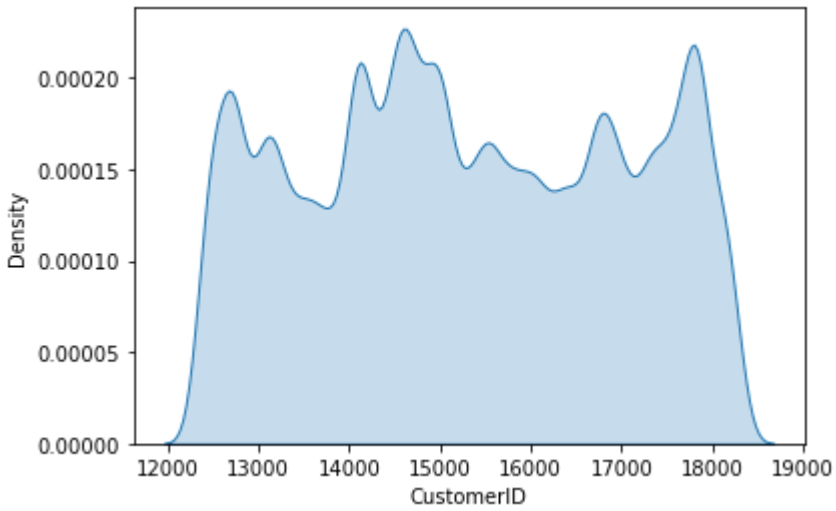
In [29]:
```
1 sns.kdeplot(src['UnitPrice'], shade = True)
```

Out[29]: <AxesSubplot:xlabel='UnitPrice', ylabel='Density'>



In [30]:
```
1 sns.kdeplot(src['CustomerID'], shade = True)
```

Out[30]: <AxesSubplot:xlabel='CustomerID', ylabel='Density'>



2. Check for missing values in all columns and replace them with the appropriate metric (Mean/Median/Mode)

In [31]:
```
(src.isna().mean() * 100).sort_values(ascending = False)
```

Out[31]:
```
CustomerID     24.926694
Description     0.268311
Country         0.000000
UnitPrice       0.000000
InvoiceDate     0.000000
Quantity        0.000000
StockCode       0.000000
InvoiceNo       0.000000
dtype: float64
```

In [32]:
```
# Since there are no outliers in CustomerID column, imputing with mean
src['CustomerID'].fillna(value = src['CustomerID'].mean(), inplace = True)
```

In [33]:
```
# Since Desciption is a categorical column, imputing can be done with mode
src['Description'].fillna(value = src['Description'].mode()[0], inplace = Tr
```

In [34]:
```
(src.isna().mean() * 100).sort_values(ascending = False)
```

Out[34]:
```
Country        0.0
CustomerID     0.0
UnitPrice      0.0
InvoiceDate    0.0
Quantity       0.0
Description    0.0
StockCode      0.0
InvoiceNo      0.0
dtype: float64
```

3. Remove duplicate rows

In [35]:
```
src.shape
```

Out[35]:
```
(541909, 8)
```

In [36]:
```
src.drop_duplicates(inplace = True)
```

In [37]:
```
src.shape
```

Out[37]:
```
(536641, 8)
```

4. Remove rows which have negative values in Quantity column

In [38]:
```
src = src.loc[src['Quantity'] > 0, :]
```

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

In [39]:
```
1 src['InvoiceDate'].dtypes
```

Out[39]:
```
dtype('O')
```

In [40]:
```
1 # Converting the datatype from 'object' to 'datetime' format
2 src['InvoiceDate'] = pd.to_datetime(src['InvoiceDate'])
3 src['InvoiceDate'].dtypes
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[40]:
```
dtype('<M8[ns]')
```

In [41]:
```
1 src['Month'] = src['InvoiceDate'].dt.month
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

In [42]:
```
1 src['Day'] = src['InvoiceDate'].dt.day
2 src.loc[50000, ['Day', 'InvoiceDate']]
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[42]:
```
Day                             9
InvoiceDate    2011-01-09 15:18:00
Name: 50000, dtype: object
```

In [43]:
```
src['Hour'] = src['InvoiceDate'].dt.hour
src.loc[50000, ['Hour', 'InvoiceDate']]
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[43]:
```
Hour                            15
InvoiceDate    2011-01-09 15:18:00
Name: 50000, dtype: object
```

6. How many orders made by the customers?

In [44]:
```
# Since each Invoice ID represents a customer
src['InvoiceNo'].value_counts()
```

Out[44]:
```
573585    1114
581219     749
581492     731
580729     721
558475     705
          ...
572293       1
543460       1
572072       1
551595       1
571916       1
Name: InvoiceNo, Length: 20728, dtype: int64
```

7. TOP 5 customers with higher number of orders

In [45]:
```
# Since each Invoice ID represents a customer
src['InvoiceNo'].value_counts().head()
```

Out[45]:
```
573585    1114
581219     749
581492     731
580729     721
558475     705
Name: InvoiceNo, dtype: int64
```

8. How much money spent by the customers?.

In [46]:
```
1 src['Total'] = src['UnitPrice'] * src['Quantity']
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

In [47]:
```
1 src.groupby('InvoiceNo').agg({'Total': 'sum'})
```

Out[47]:

InvoiceNo	Total
536365	139.12
536366	22.20
536367	278.73
536368	70.05
536369	17.85
...	...
581586	339.20
581587	249.45
A563185	11062.06
A563186	-11062.06
A563187	-11062.06

20728 rows × 1 columns

9. TOP 5 customers with highest money spent

In [48]:
```
1 src.groupby('InvoiceNo').agg({'Total': 'sum'}).sort_values(ascending = False
```

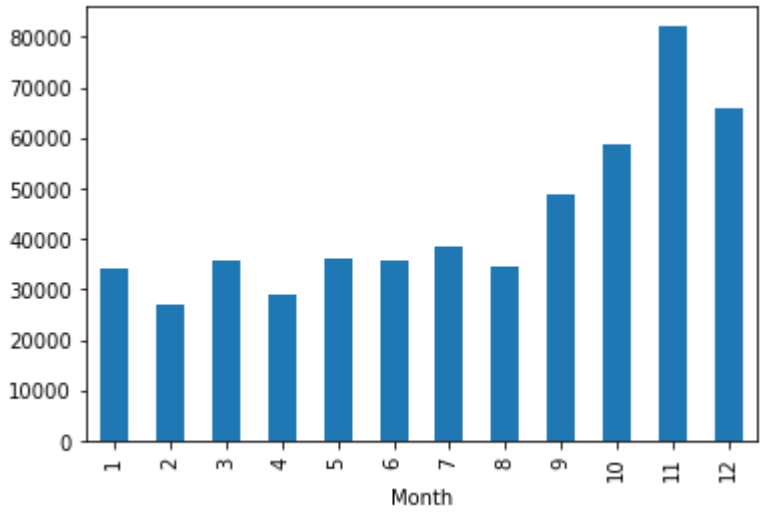
Out[48]:

InvoiceNo	Total
581483	168469.60
541431	77183.60
574941	52940.94
576365	50653.91
556444	38970.00

10. How many orders per month?

In [49]:
```
1 src.groupby('Month').count()['InvoiceNo'].plot(kind = 'bar')
```

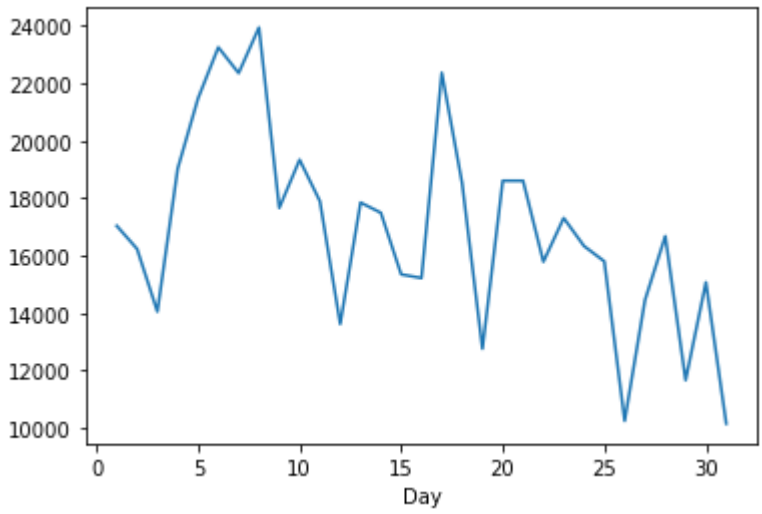
Out[49]: <AxesSubplot:xlabel='Month'>



11. How many orders per day?

In [50]:
```
1 src.groupby('Day').count()['InvoiceNo'].plot(kind = 'line')
```

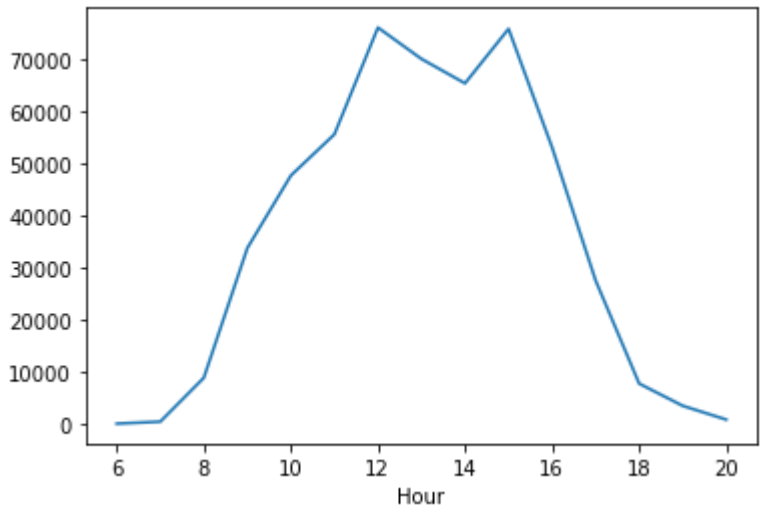
Out[50]: <AxesSubplot:xlabel='Day'>



12. How many orders per hour?

In [51]:
```
1 src.groupby('Hour').count()['InvoiceNo'].plot(kind = 'line')
```

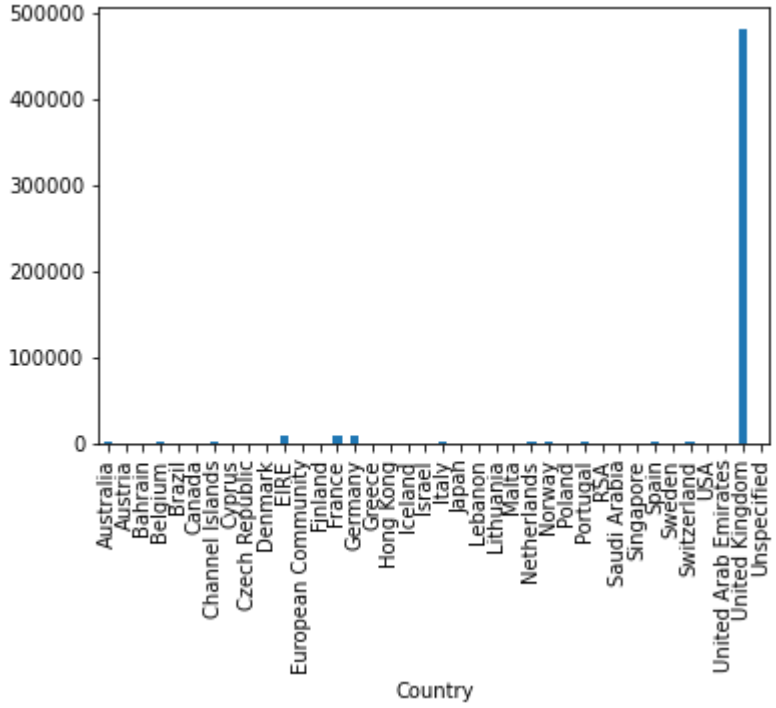
Out[51]: <AxesSubplot:xlabel='Hour'>



13. How many orders for each country?

In [52]:
```
1 src.groupby('Country').count()['InvoiceNo'].plot(kind = 'bar')
```

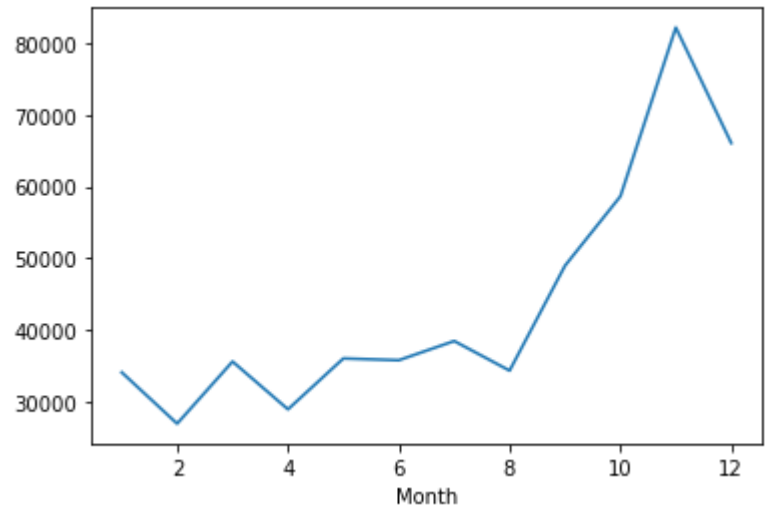
Out[52]: <AxesSubplot:xlabel='Country'>



14. Orders trend across months

In [53]:
```
1 src.groupby('Month').count()['InvoiceNo'].plot(kind = 'line')
```

Out[53]: <AxesSubplot:xlabel='Month'>



15. How much money spent by each country?

In [54]:
```
1 plt.figure(figsize=(15, 6))
2 src.groupby('Country').count()['InvoiceNo'].sort_values().plot(kind = 'barh'
```

Out[54]: <AxesSubplot:ylabel='Country'>