EDA of Online Retail

In [5]:
```
# import the neccessary libraries
import numpy as np
import pandas as pd

# import visuals Python Libraries
import seaborn as sns
import matplotlib.pyplot as plt
```

In [6]:
```
# Loading and reading a dataset
# For Excel you will do the following 'df = pd.read_excel(r'')'

df = pd.read_csv(r'C:\Users\hp\Desktop\Data Analysis Training\Python\DataSet\OnlineRetail.csv')
df
```

Out[6]:

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

541909 rows × 8 columns

In [7]:
```
#Check the first 5 rows

df = pd.read_csv(r'C:\Users\hp\Desktop\Data Analysis Training\Python\DataSet\OnlineRetail.csv')
df.head()
```

Out[7]:

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

Data Inspection and Manipulation

In [4]:
```
# in View of the shape of the data
df.shape
```

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

In [5]:
```
# we also want to check the tail
df.tail()
```

Out[5]:

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

In [6]:
```
# We also want to check the info
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 [7]:
```
# we aalso we want to check the columns

df.columns
```

Out[7]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [8]:
```
# we can also change the column case to lower or upper
[x.lower() for x in df.columns]
```

Out[8]:
```
['invoiceno',
 'stockcode',
 'description',
 'quantity',
 'invoicedate',
 'unitprice',
 'customerid',
 'country']
```

In [9]:
```
# we want to do summary statistics with describe()
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

Cleaning and Manipulation

- Cleaning of Data is done for the following purposes:
  - To prevent time wastage
  - To make your analysis run faster
  - To prevent bad conclusions
- How to clean:
  - You may choose to delete rows or modify content such as filling in or replacing empty values and so on
- Data cleaning is also called data wranggling and this process is 80% of the work of a data analyst.
- Empty values could be deleted, replaced with a values that makes sense
- Check for data consistency: This is important for string formating etc
- Handling Outliers
- Remove duplicates
- Validate correctness of entries; e.g age column shouldnt contain texts

In [10]:
```
# Check for mising Values
df.isna().sum()

# Below we have missing values in description and customer ID
```

Out[10]:
```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135080
Country            0
dtype: int64
```

Questions

- What does the records above with empty values mean?
- Does it mean the sale of the customer is not recorded?
- Answer: it is either the sales was not recorded or someonelse made the sale.

In [11]:
```
df[df.CustomerID.isna()]
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	NaN	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	NaN	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	NaN	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	NaN	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	NaN	United Kingdom
...	...	...	...	...	...	...	...	...
541536	581498	85099B	JUMBO BAG RED RETROSPOT	5	12/9/2011 10:26	4.13	NaN	United Kingdom
541537	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	12/9/2011 10:26	4.13	NaN	United Kingdom
541538	581498	85150	LADIES & GENTLEMEN METAL SIGN	1	12/9/2011 10:26	4.96	NaN	United Kingdom
541539	581498	85174	S/4 CACTI CANDLES	1	12/9/2011 10:26	10.79	NaN	United Kingdom
541540	581498	DOT	DOTCOM POSTAGE	1	12/9/2011 10:26	1714.17	NaN	United Kingdom

135080 rows × 8 columns

Question

- What is the result of CustomerID.isna as a mask?
- Is it boolean or a dataframe?

In [12]:
```
# As a mask mean we want to run it alone as seen below:
df.CustomerID.isna()
#'''With the output below we see its boolean'''
```

Out[12]:
```
0         False
1         False
2         False
3         False
4         False
          ...
541904    False
541905    False
541906    False
541907    False
541908    False
Name: CustomerID, Length: 541909, dtype: bool
```

In [13]:
```
# We also want to check record with empty discription

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

Out[13]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.0	NaN	United Kingdom
1970	536545	21134	NaN	1	12/1/2010 14:32	0.0	NaN	United Kingdom
1971	536546	22145	NaN	1	12/1/2010 14:33	0.0	NaN	United Kingdom
1972	536547	37509	NaN	1	12/1/2010 14:33	0.0	NaN	United Kingdom
1987	536549	85226A	NaN	1	12/1/2010 14:34	0.0	NaN	United Kingdom
...	...	...	...	...	...	...	...	...
535322	581199	84581	NaN	-2	12/7/2011 18:26	0.0	NaN	United Kingdom
535326	581203	23406	NaN	15	12/7/2011 18:31	0.0	NaN	United Kingdom
535332	581209	21620	NaN	6	12/7/2011 18:35	0.0	NaN	United Kingdom
536981	581234	72817	NaN	27	12/8/2011 10:33	0.0	NaN	United Kingdom
538554	581408	85175	NaN	20	12/8/2011 14:06	0.0	NaN	United Kingdom

1454 rows × 8 columns

In [14]:
```
df.Description.isna()
```

Out[14]:
```
0         False
1         False
2         False
3         False
4         False
          ...
541904    False
541905    False
541906    False
541907    False
541908    False
Name: Description, Length: 541909, dtype: bool
```

Question

- How many store codes have no Description?

In [15]:
```
# that means we will use the length to check or we use nunique

len(df[df.Description.isna()].StockCode.unique())
```

Out[15]:
```
960
```

In [16]:
```
# we want to use nunique

df[df.Description.isna()].StockCode.nunique()
```

Out[16]:
```
960
```

Question

- What countries have sales with no descriptions?

In [17]:
```
df[df.Description.isna()].Country.value_counts()
```

Out[17]:
```
United Kingdom    1454
Name: Country, dtype: int64
```

Question

- What country has sales with no Customer ID and how many records affected?

In [18]:
```
df[df.CustomerID.isna()].Country.value_counts()
```

Out[18]:
```
United Kingdom    133600
EIRE                 711
Hong Kong            288
Unspecified          202
Switzerland          125
France                66
Israel                47
Portugal              39
Bahrain                2
Name: Country, dtype: int64
```

Note:

- You can choose to delete rows with no records.
- Before deleting check number of records that will be affected
  - Use the following methods to determine that.

In [19]:
```
print('Total Number of Records: ', df.shape[0])
print('Total Number of Records with missing description: ', df[df.Description.isna()].shape[0])
print('Total Number of Records without missing description: ', df[df.Description.notna()].shape[0])
```

```
Total Number of Records:  541909
Total Number of Records with missing description:  1454
Total Number of Records without missing description:  540455
```

In [20]:
```
## Now we want to get the numbers of rows with missing descriptions
```

In [21]:
```
num_missing = df[df.Description.isna()].shape[0]

# percentage of rows in all datase
num_all = df.shape[0]

(num_missing / num_all) * 100
```

Out[21]:
```
0.2683107311375157
```

In [22]:
```
# numbers of rows with missing descriptions
num_missing = df[df.Description.isna()].shape[0]
num_missing
```

Out[22]:
```
1454
```

In [23]:
```
# numbers of rows in the entire dataset
num_all = df.shape[0]
num_all
```

Out[23]:
```
541909
```

In [24]:
```
# Now we want to check records that are not Nan - notna()
df = df[df.Description.notna()]
df
```

Out[24]:

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

540455 rows × 8 columns

In [25]:
```
# Now we want to check records that are not Nan - notna()
# We want to view the rows with NAN column by column we will add agregate function to it

df.isna().sum()
```

Out[25]:
```
InvoiceNo          0
StockCode          0
Description        0
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     133626
Country            0
dtype: int64
```

Other Cleaninning Activities we can do...

In [26]:
```
# We want to replace values E.G EIRE
df.Country.unique()
```

Out[26]:
```
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 [27]:
```
# We want to find record where country is represented as EIRE
df[df.Country == 'EIRE']
```

Out[27]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1404	536540	22968	ROSE COTTAGE KEEPSAKE BOX	4	12/1/2010 14:05	9.95	14911.0	EIRE
1405	536540	85071A	BLUE CHARLIE+LOLA PERSONAL DOORSIGN	6	12/1/2010 14:05	2.95	14911.0	EIRE
1406	536540	85071C	CHARLIE+LOLA"EXTREMELY BUSY" SIGN	6	12/1/2010 14:05	2.55	14911.0	EIRE
1407	536540	22355	CHARLOTTE BAG SUKI DESIGN	50	12/1/2010 14:05	0.85	14911.0	EIRE
1408	536540	21579	LOLITA DESIGN COTTON TOTE BAG	6	12/1/2010 14:05	2.25	14911.0	EIRE
...	...	...	...	...	...	...	...	...
539151	581433	22192	BLUE DINER WALL CLOCK	2	12/8/2011 15:54	8.50	14911.0	EIRE
539152	581433	48187	DOORMAT NEW ENGLAND	2	12/8/2011 15:54	8.25	14911.0	EIRE
539153	581433	48184	DOORMAT ENGLISH ROSE	2	12/8/2011 15:54	8.25	14911.0	EIRE
539154	581433	20685	DOORMAT RED RETROSPOT	2	12/8/2011 15:54	8.25	14911.0	EIRE
539155	581433	79302M	ART LIGHTS,FUNK MONKEY	6	12/8/2011 15:54	2.95	14911.0	EIRE

8196 rows × 8 columns

In [28]:
```
# Here we want to find the number of rows with EIRE
df[df.Country == 'EIRE'].Country
```

Out[28]:
```
1404      EIRE
1405      EIRE
1406      EIRE
1407      EIRE
1408      EIRE
          ...
539151    EIRE
539152    EIRE
539153    EIRE
539154    EIRE
539155    EIRE
Name: Country, Length: 8196, dtype: object
```

In [29]:
```
# Let us replace EIRE with Ireland .
# We want to use replace function country.replace

df[df.Country == 'EIRE'].Country.replace('EIRE', 'Ireland')
```

Out[29]:
```
1404      Ireland
1405      Ireland
1406      Ireland
1407      Ireland
1408      Ireland
           ...
539151    Ireland
539152    Ireland
539153    Ireland
539154    Ireland
539155    Ireland
Name: Country, Length: 8196, dtype: object
```

In [30]:
```
df.head()
```

Out[30]:

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

In [11]:
```
# We now want to applly the replace operation
# We are saying beliow to show us the column where the replacement is made in the dataset to be sure.

df['Country'] = df.Country.replace('EIRE', 'Ireland')
df['Country']
```

Out[11]:
```
0         United Kingdom
1         United Kingdom
2         United Kingdom
3         United Kingdom
4         United Kingdom
              ...
541904           France
541905           France
541906           France
541907           France
541908           France
Name: Country, Length: 541909, dtype: object
```

In [12]:
```
# We now want to find Values using loc()
# Loc - It is used to find a value along the rows it is the label based indexing which means we can specify rows and co
# iLoc - This is an integer based indexing. which means, we can specify rows and columns by their integer index.
df.loc[:, 'Country'] = df.Country.replace('EIRE', 'Ireland')
```

In [13]:
```
# We can check what we have done now.. if we have rows with Ireland not EIRE

df[df.Country == 'Ireland'].head()
```

Out[13]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1404	536540	22968	ROSE COTTAGE KEEPSAKE BOX	4	12/1/2010 14:05	9.95	14911.0	Ireland
1405	536540	85071A	BLUE CHARLIE+LOLA PERSONAL DOORSIGN	6	12/1/2010 14:05	2.95	14911.0	Ireland
1406	536540	85071C	CHARLIE+LOLA"EXTREMELY BUSY" SIGN	6	12/1/2010 14:05	2.55	14911.0	Ireland
1407	536540	22355	CHARLOTTE BAG SUKI DESIGN	50	12/1/2010 14:05	0.85	14911.0	Ireland
1408	536540	21579	LOLITA DESIGN COTTON TOTE BAG	6	12/1/2010 14:05	2.25	14911.0	Ireland

In [14]:
```
# We could also check to be sure if there are no EIRE in the dataset, we will do the following
df[df.Country == 'EIRE'].head()
```

Out[14]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

In [35]:
```
'''the return output shows no record with EIRE in the dataset as seen above'''
```

Out[35]:
```
'the return output shows no record with EIRE in the dataset as seen above'
```

Now we want to replace missing Customer IDs

In [15]:
```
df.CustomerID.value_counts()
```

Out[15]:
```
17841.0    7983
14911.0    5903
14096.0    5128
12748.0    4642
14606.0    2782
           ...
15070.0       1
15753.0       1
17065.0       1
16881.0       1
16995.0       1
Name: CustomerID, Length: 4372, dtype: int64
```

In [16]:
```
# We noticed here that the customer ids are floats having decimal, we want to convert to srings
# How to convert floats to string - first we convert to integers to block the decimal, before converting to string.
# This means we are converting the IDs to texts

df.CustomerID.astype('Int64').astype(str)
```

Out[16]:
```
0         17850
1         17850
2         17850
3         17850
4         17850
          ...
541904    12680
541905    12680
541906    12680
541907    12680
541908    12680
Name: CustomerID, Length: 541909, dtype: object
```

We Now want to replace NaN CustomerIDs to Unidentified

- Recall that the data type customer ID here was originally a float so we made it string so that we can replace the values that are NaN with 'Unidentified' which is a string or text

In [17]:
```
df['CustomerID'] = df.CustomerID.astype('Int64').astype(str)
```

In [18]:
```
df.dtypes
```

Out[18]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID      object
Country         object
dtype: object
```

In [19]:
```
# We now want to check for any missing records or customer IDs
df.isna().sum()
```

Out[19]:
```
InvoiceNo         0
StockCode         0
Description    1454
Quantity          0
InvoiceDate       0
UnitPrice         0
CustomerID        0
Country           0
dtype: int64
```

Note:

- If you notice that earlier the customer ID has null values or empty cells of about 133626. So when we convered it to a integer and we no longer have the empty cells but it as be replace to NaN. That was why we converted the ID from float to Integer.

In [20]:
```
# Npw we want to check for the NaN values becus the column is now an integer
df.CustomerID.value_counts()
```

Out[20]:
```
<NA>     135080
17841      7983
14911      5903
14096      5128
12748      4642
          ...
13270         1
17763         1
17291         1
15668         1
15562         1
Name: CustomerID, Length: 4373, dtype: int64
```

In [21]:
```
# Therefore we found out that we have the missing values are replace with NA not NaN
# We will now replace the NA with Unidentfy.. but then we want to check
df[df.CustomerID == '<NA>']
```

Out[21]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	<NA>	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	<NA>	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	<NA>	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	<NA>	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	<NA>	United Kingdom
...	...	...	...	...	...	...	...	...
541536	581498	85099B	JUMBO BAG RED RETROSPOT	5	12/9/2011 10:26	4.13	<NA>	United Kingdom
541537	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	12/9/2011 10:26	4.13	<NA>	United Kingdom
541538	581498	85150	LADIES & GENTLEMEN METAL SIGN	1	12/9/2011 10:26	4.96	<NA>	United Kingdom
541539	581498	85174	S/4 CACTI CANDLES	1	12/9/2011 10:26	10.79	<NA>	United Kingdom
541540	581498	DOT	DOTCOM POSTAGE	1	12/9/2011 10:26	1714.17	<NA>	United Kingdom

135080 rows × 8 columns

In [22]:
```
# We now want to replace '<NA>' with unidentify...using the syntax below
# Using a paramiter 'inplace' will apply the operation straight up
df.CustomerID.replace('<NA>', 'Unidentify', inplace = True)
```

In [23]:
```
# We have run it an it is now activated
# We now want to check if <NA> is still in the dataset
df[df.CustomerID == '<NA>']
```

Out[23]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

In [24]:
```
# As seen above we notice that there are no NA in the dataset
# We want to check the whole dataset
df[df.CustomerID == 'Unidentify']
```

Out[24]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	Unidentify	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	Unidentify	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	Unidentify	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	Unidentify	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	Unidentify	United Kingdom
...	...	...	...	...	...	...	...	...
541536	581498	85099B	JUMBO BAG RED RETROSPOT	5	12/9/2011 10:26	4.13	Unidentify	United Kingdom
541537	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	12/9/2011 10:26	4.13	Unidentify	United Kingdom
541538	581498	85150	LADIES & GENTLEMEN METAL SIGN	1	12/9/2011 10:26	4.96	Unidentify	United Kingdom
541539	581498	85174	S/4 CACTI CANDLES	1	12/9/2011 10:26	10.79	Unidentify	United Kingdom
541540	581498	DOT	DOTCOM POSTAGE	1	12/9/2011 10:26	1714.17	Unidentify	United Kingdom

135080 rows × 8 columns

In [25]:
```
# We could also check with value count

df.CustomerID.value_counts()
```

Out[25]:
```
Unidentify    135080
17841           7983
14911           5903
14096           5128
12748           4642
               ...
13270              1
17763              1
17291              1
15668              1
15562              1
Name: CustomerID, Length: 4373, dtype: int64
```

In [26]:
```
# We want to check again if there are any missing values
df.isna().sum()
```

Out[26]:
```
InvoiceNo         0
StockCode         0
Description    1454
Quantity          0
InvoiceDate       0
UnitPrice         0
CustomerID        0
Country           0
dtype: int64
```

In [27]:
```
# We now want to check the percentage of the unidentified customerids

df[df.CustomerID == 'Unidentify'].shape[0] / df.shape[0]
```

Out[27]:
```
0.249266943342886
```

Relationship and Insight

- We want to look at the relationships in the data and try to answer basic business questions so as to gain more insight

In [28]:
```
df.head()
```

Out[28]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 08:26	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 08:26	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 08:26	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 08:26	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 08:26	3.39	17850	United Kingdom

In [29]:
```
# We want to get the average sales
df['Sales'] = df.UnitPrice * df.Quantity
```

In [30]:
```
df.head()
```

Out[30]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 08:26	2.55	17850	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 08:26	2.75	17850	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34

If we noticed above a new column was added and that was sales column which give us a unique value(s)

In [31]:
```
# Now we want to check al the values in the sales
df['Sales'].head()
```

Out[31]:
```
0    15.30
1    20.34
2    22.00
3    20.34
4    20.34
Name: Sales, dtype: float64
```

In [32]:
```
# We equally want to check the overall average price and we will use the mean function to get that

df.Sales.mean()
```

Out[32]:
```
17.987794877005495
```

In [33]:
```
# We can also group the values by invoice number (invoice are unique number just like CustomerID)
# Average Sales by invoice number
df.groupby('InvoiceNo').Sales.mean()
```

Out[33]:
```
InvoiceNo
536365         19.874286
536366         11.100000
536367         23.227500
536368         17.512500
536369         17.850000
                 ...
C581484   -168469.600000
C581490       -16.265000
C581499      -224.690000
C581568       -54.750000
C581569        -3.750000
Name: Sales, Length: 25900, dtype: float64
```

We now want to generate insights

Question:

- What is the total sales value by country for the top 10 countries?

In [34]:
```
df.groupby('Country').Sales.sum().head(10)
```

Out[34]:
```
Country
Australia          137077.27
Austria             10154.32
Bahrain               548.40
Belgium             40910.96
Brazil               1143.60
Canada               3666.38
Channel Islands     20086.29
Cyprus              12946.29
Czech Republic        707.72
Denmark             18768.14
Name: Sales, dtype: float64
```

In [35]:
```
df.groupby('Country').Sales.sum().astype(int).head(10)
```

Out[35]:
```
Country
Australia          137077
Austria             10154
Bahrain               548
Belgium             40910
Brazil               1143
Canada               3666
Channel Islands     20086
Cyprus              12946
Czech Republic        707
Denmark             18768
Name: Sales, dtype: int32
```

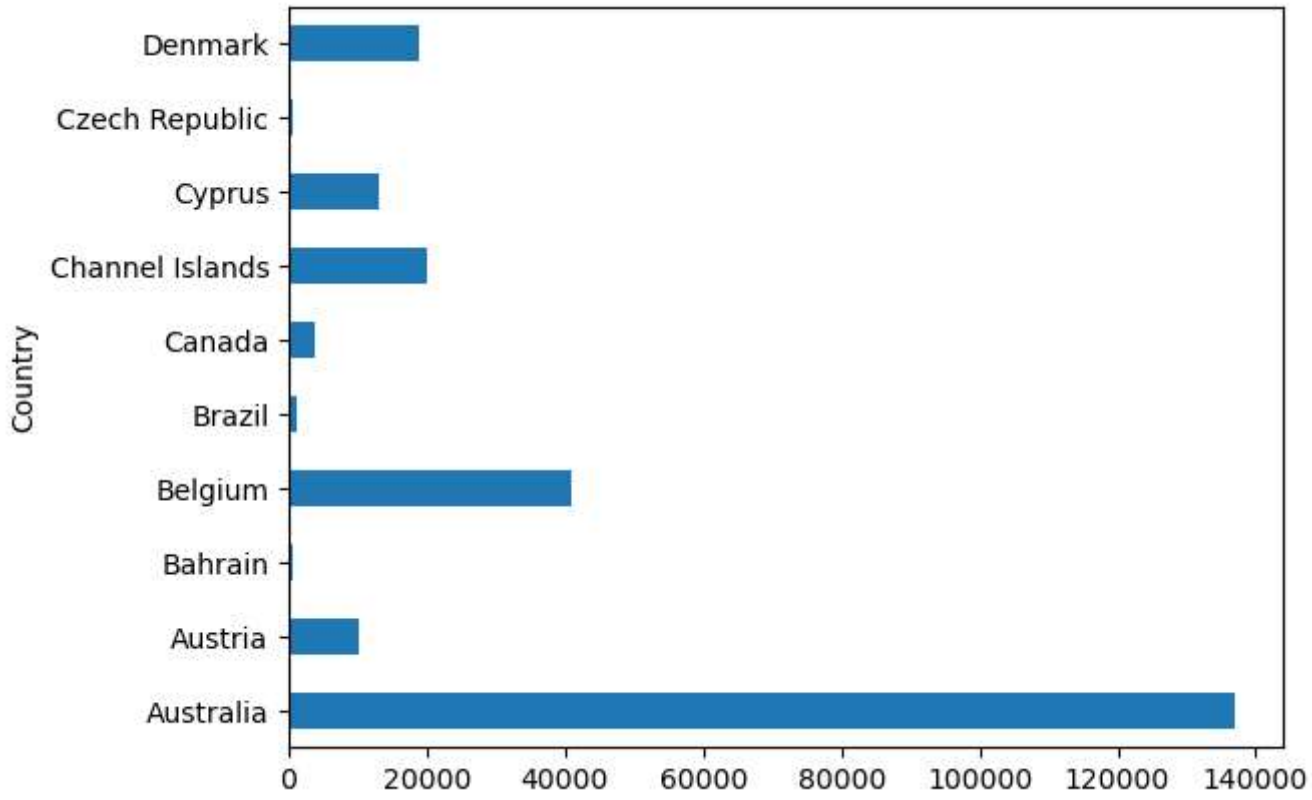
In [36]:
```
top10_sales = df.groupby('Country').Sales.sum().astype(int).head(10)
top10_sales
```

Out[36]:
```
Country
Australia          137077
Austria             10154
Bahrain               548
Belgium             40910
Brazil               1143
Canada               3666
Channel Islands     20086
Cyprus              12946
Czech Republic        707
Denmark             18768
Name: Sales, dtype: int32
```

Visualization

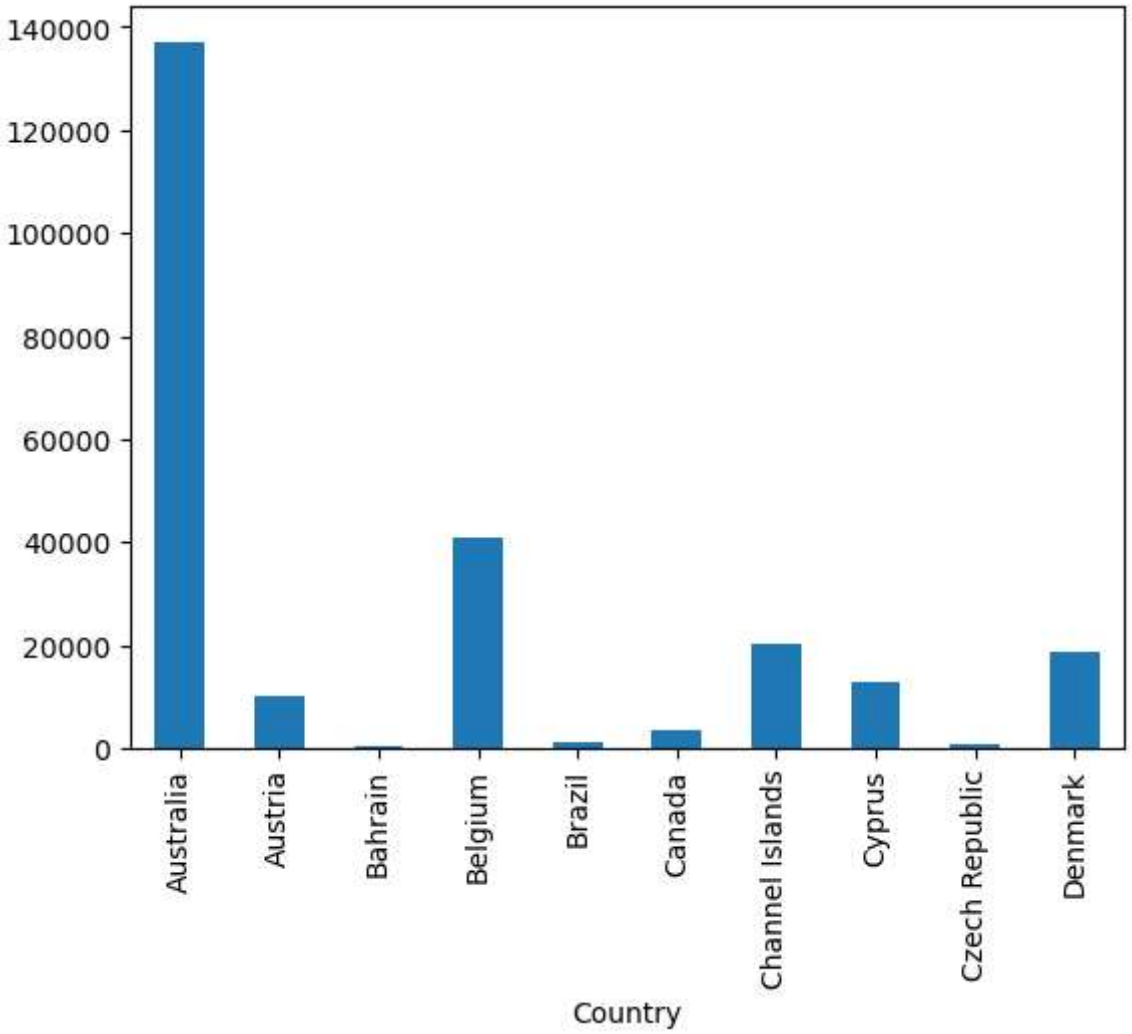
In [37]:
```
# We now want to use barchart to visualize the above

top10_sales.plot.barh()
```

Out[37]:
```
<AxesSubplot:ylabel='Country'>
```



In [38]:
```
# Vertical view

top10_sales.plot.bar()
```

Out[38]:
```
<AxesSubplot:xlabel='Country'>
```



In [39]:
```
# Total quantities by Country in ascending or deccending order

sales_qty = df.groupby('Country').Quantity.sum().sort_values(ascending = False)
sales_qty
```

Out[39]:
```
Country
United Kingdom          4263829
Netherlands              200128
Ireland                  142637
Germany                  117448
France                   110480
Australia                 83653
Sweden                    35637
Switzerland               30325
Spain                     26824
Japan                     25218
Belgium                   23152
Norway                    19247
Portugal                  16180
Finland                   10666
Channel Islands            9479
Denmark                    8188
Italy                      7999
Cyprus                     6317
Singapore                  5234
Austria                    4827
Hong Kong                  4769
Israel                     4353
Poland                     3653
Unspecified                3300
Canada                     2763
Iceland                    2458
Greece                     1556
USA                        1034
United Arab Emirates        982
Malta                       944
Lithuania                   652
Czech Republic              592
European Community          497
Lebanon                     386
Brazil                      356
RSA                         352
Bahrain                     260
Saudi Arabia                 75
Name: Quantity, dtype: int64
```

In [40]:
```
# Top 10 Total quantities by Country in ascending or deccending order

top10_qty = df.groupby('Country').Quantity.sum().sort_values(ascending = False)[:10]
top10_qty
```

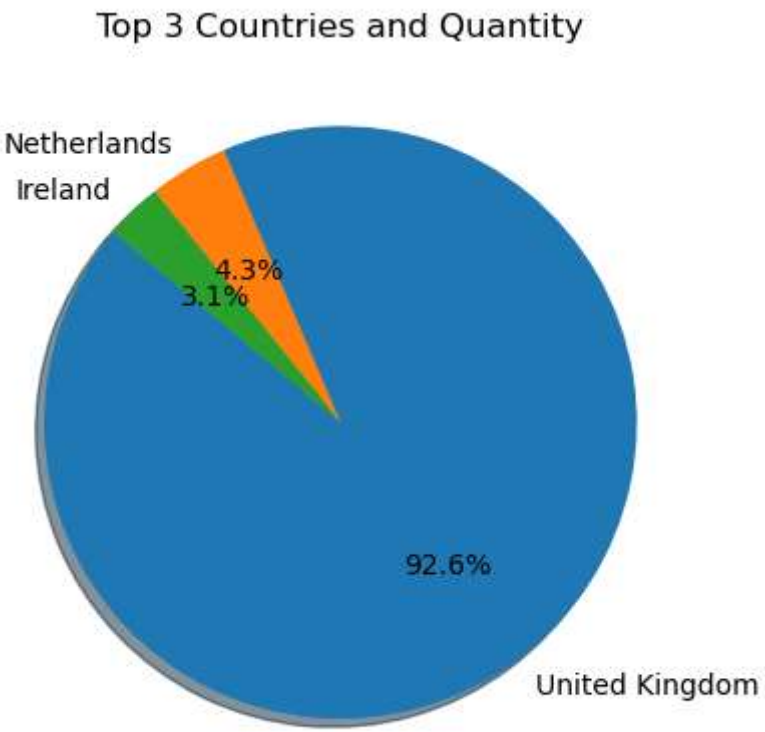
Out[40]:
```
Country
United Kingdom    4263829
Netherlands        200128
Ireland            142637
Germany            117448
France             110480
Australia           83653
Sweden              35637
Switzerland         30325
Spain               26824
Japan               25218
Name: Quantity, dtype: int64
```

In [41]:
```
# Top 3 Total quantities by Country in ascending or deccending order

top3_qty = df.groupby('Country').Quantity.sum().sort_values(ascending = False)[:3]
top3_qty
```

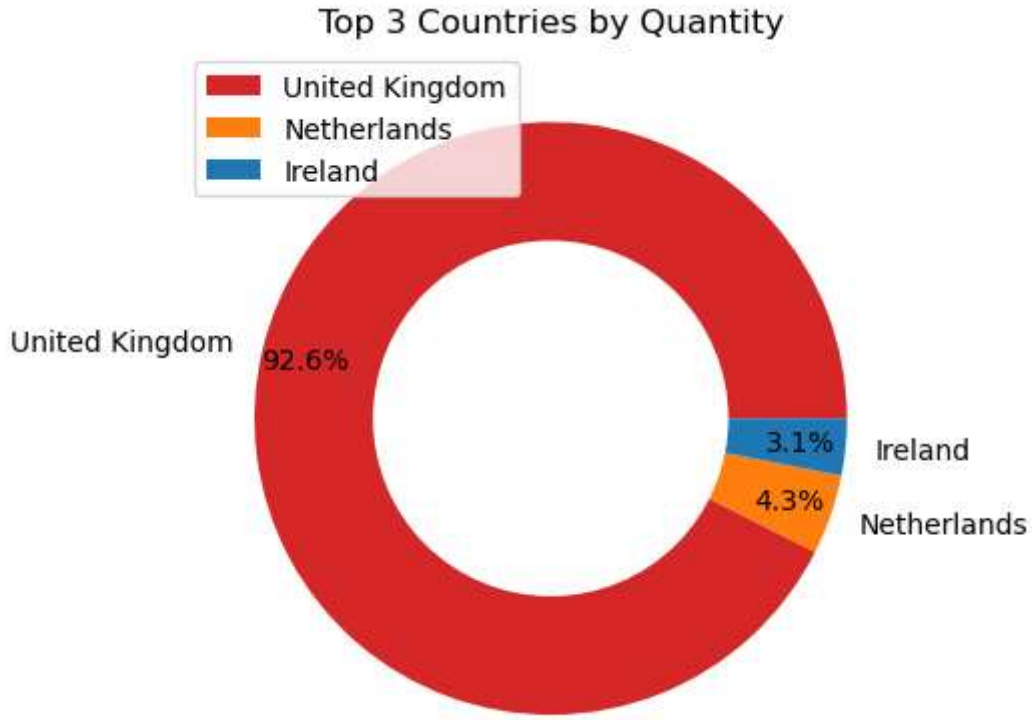
Out[41]:
```
Country
United Kingdom    4263829
Netherlands        200128
Ireland            142637
Name: Quantity, dtype: int64
```

Visualization

In [42]:
```
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#Bc564b']
explode = (0.1, 0, 0, 0, 0)
plt.pie(top3_qty, labels = top3_qty.index, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 140)
plt.title('Top 3 Countries and Quantity')
plt.show()
```



In [50]:
```
colors = ['#d62728', '#ff7f0e', '#1f77b4', '#2ca02c', '#Bc564b']
explode = (0.0, 0.0, 0.0)

plt.pie(top3_qty, labels = top3_qty.index, colors = colors, autopct = '%1.1f%%', pctdistance = 0.85, explode = explode)

centre_circle = plt.Circle((0, 0), 0.60, fc = 'white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.legend(top3_qty.index, loc = 'upper left')

plt.title('Top 3 Countries by Quantity')
plt.show()
```

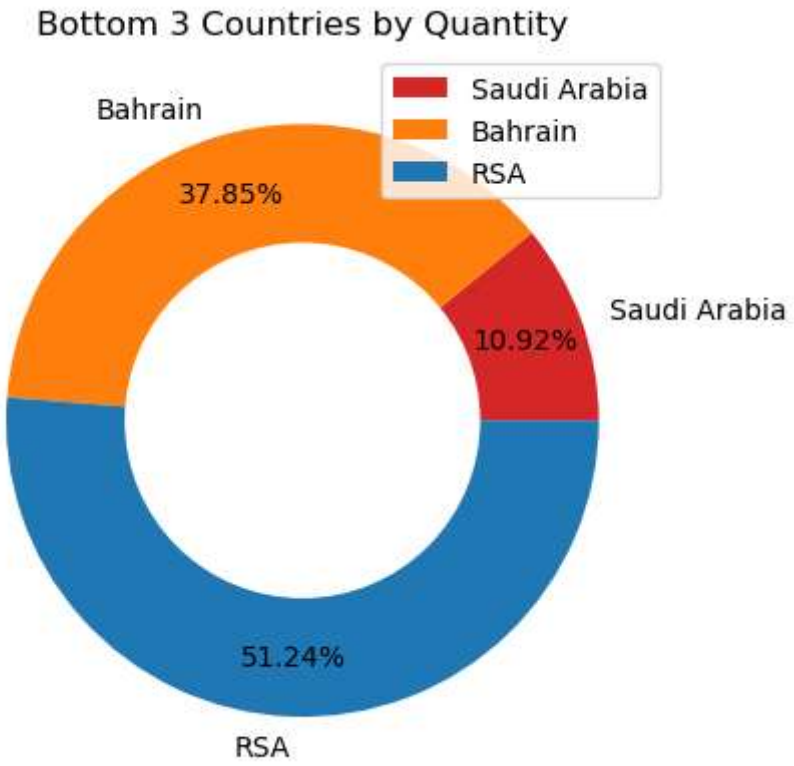


In [45]:
```
bot3_qty = df.groupby('Country').Quantity.sum().sort_values(ascending = True)[:3]
bot3_qty
```

Out[45]:
```
Country
Saudi Arabia     75
Bahrain         260
RSA             352
Name: Quantity, dtype: int64
```

In [52]:
```
colors = ['#d62728', '#ff7f0e', '#1f77b4', '#2ca02c', '#Bc564b']
explode = (0.0, 0.0, 0.0)

plt.pie(bot3_qty, labels = bot3_qty.index, colors = colors, autopct = '%1.2f%%', pctdistance = 0.80, explode = explode)

centre_circle = plt.Circle((0, 0), 0.60, fc = 'white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.legend(bot3_qty.index, loc = 'upper right')
```

```
plt.title('Bottom 3 Countries by Quantity')
plt.show()
```

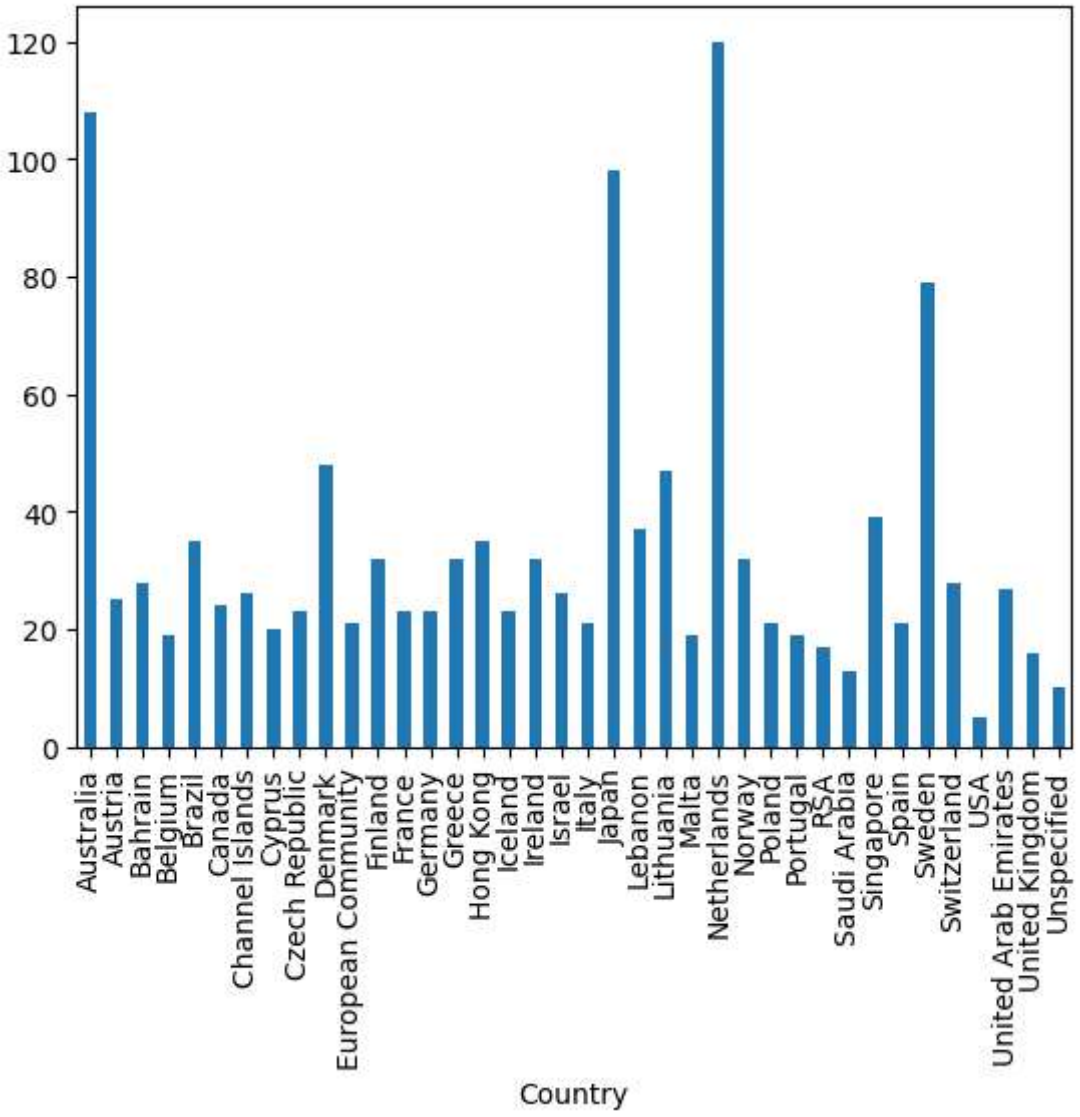


In [67]:
```
# Average Sales by Country
avg_sales = df.groupby('Country').Sales.mean().astype(int)
avg_sales
```

Out[67]:
```
Country
Australia               108
Austria                  25
Bahrain                  28
Belgium                  19
Brazil                   35
Canada                   24
Channel Islands          26
Cyprus                   20
Czech Republic           23
Denmark                  48
European Community       21
Finland                  32
France                   23
Germany                  23
Greece                   32
Hong Kong                35
Iceland                  23
Ireland                  32
Israel                   26
Italy                    21
Japan                    98
Lebanon                  37
Lithuania                47
Malta                    19
Netherlands             120
Norway                   32
Poland                   21
Portugal                 19
RSA                      17
Saudi Arabia             13
Singapore                39
Spain                    21
Sweden                   79
Switzerland              28
USA                       5
United Arab Emirates     27
United Kingdom           16
Unspecified              10
Name: Sales, dtype: int32
```

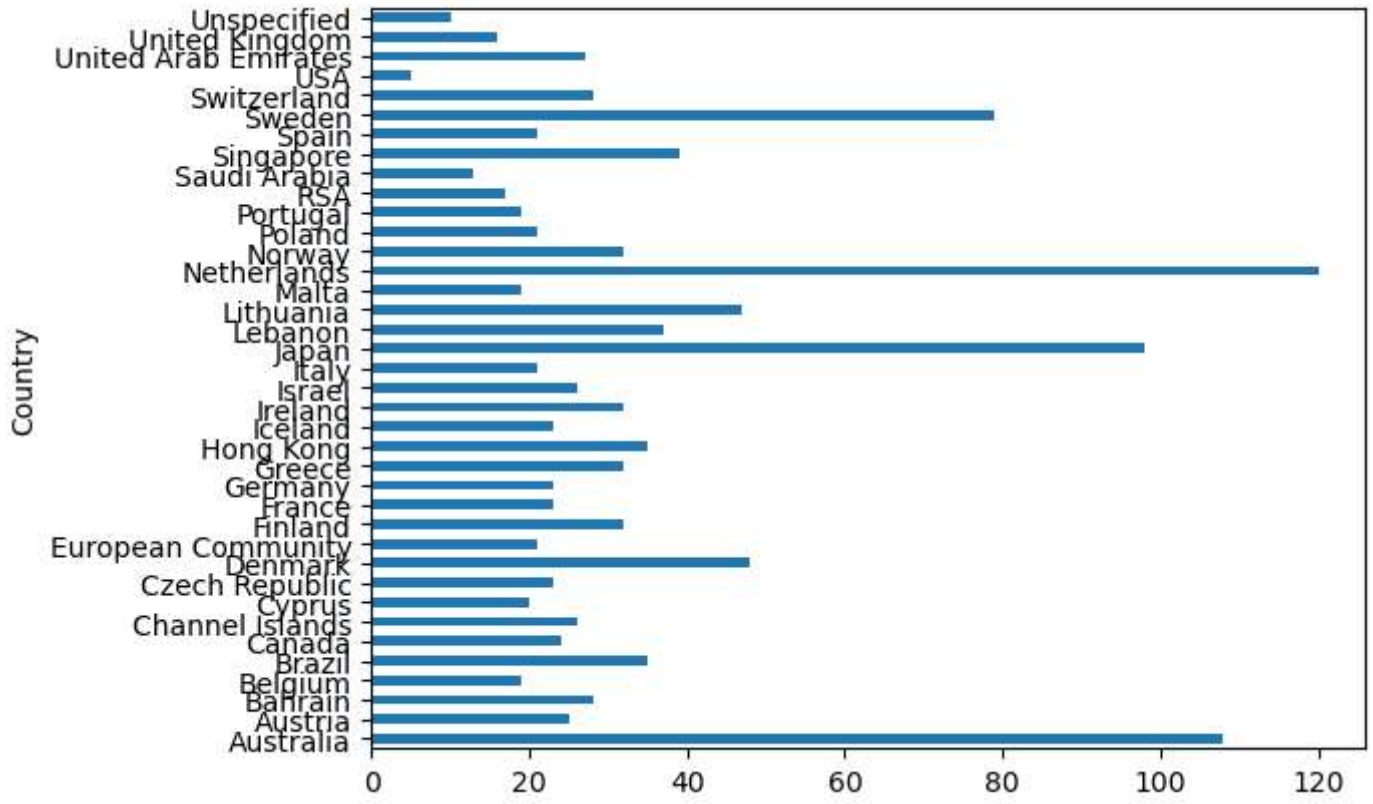
In [68]:
```
avg_sales.plot.bar()
```

Out[68]:
```
<AxesSubplot:xlabel='Country'>
```



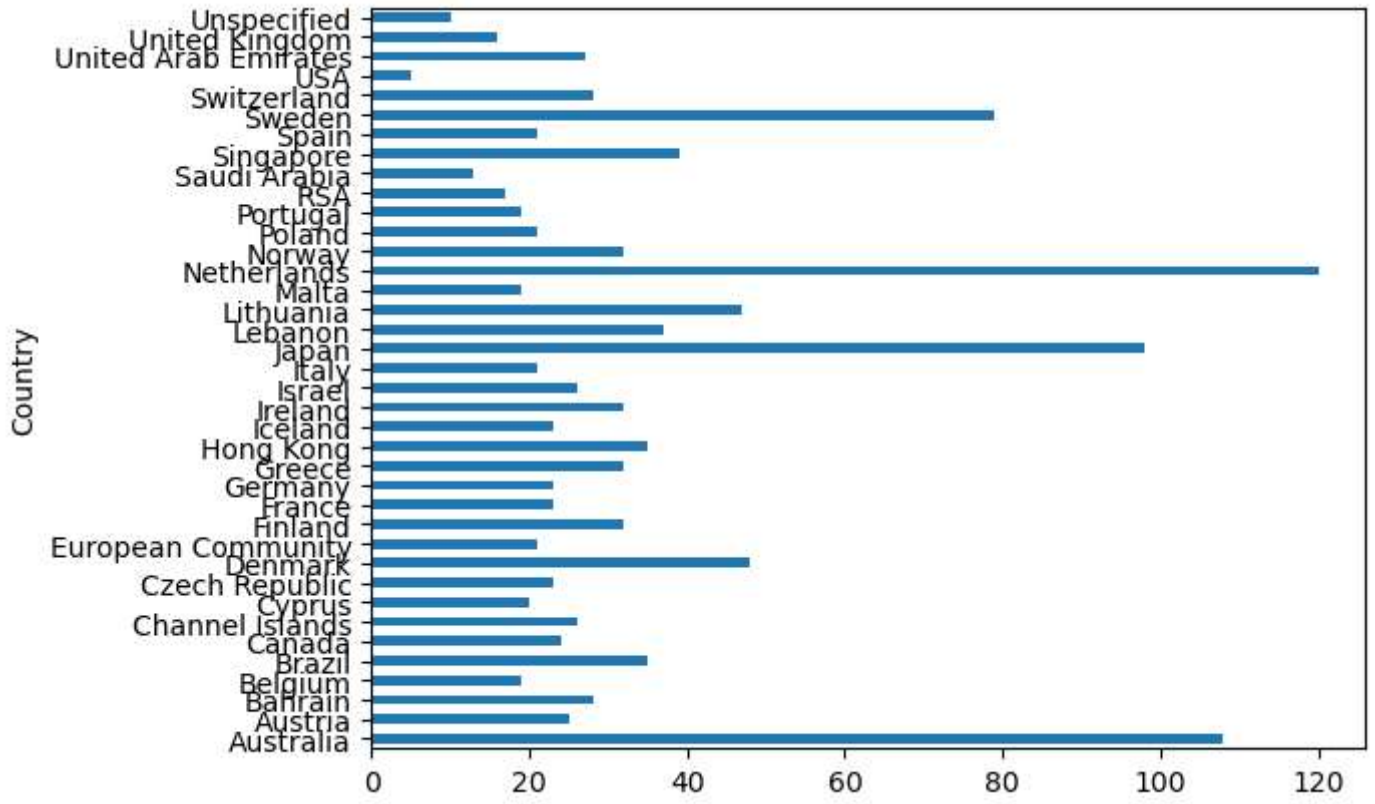
In [69]:
```
avg_sales.plot.barh()
```

Out[69]:
```
<AxesSubplot:ylabel='Country'>
```



In [70]:
```
avg_sales.plot.barh()
```

Out[70]:
```
<AxesSubplot:ylabel='Country'>
```

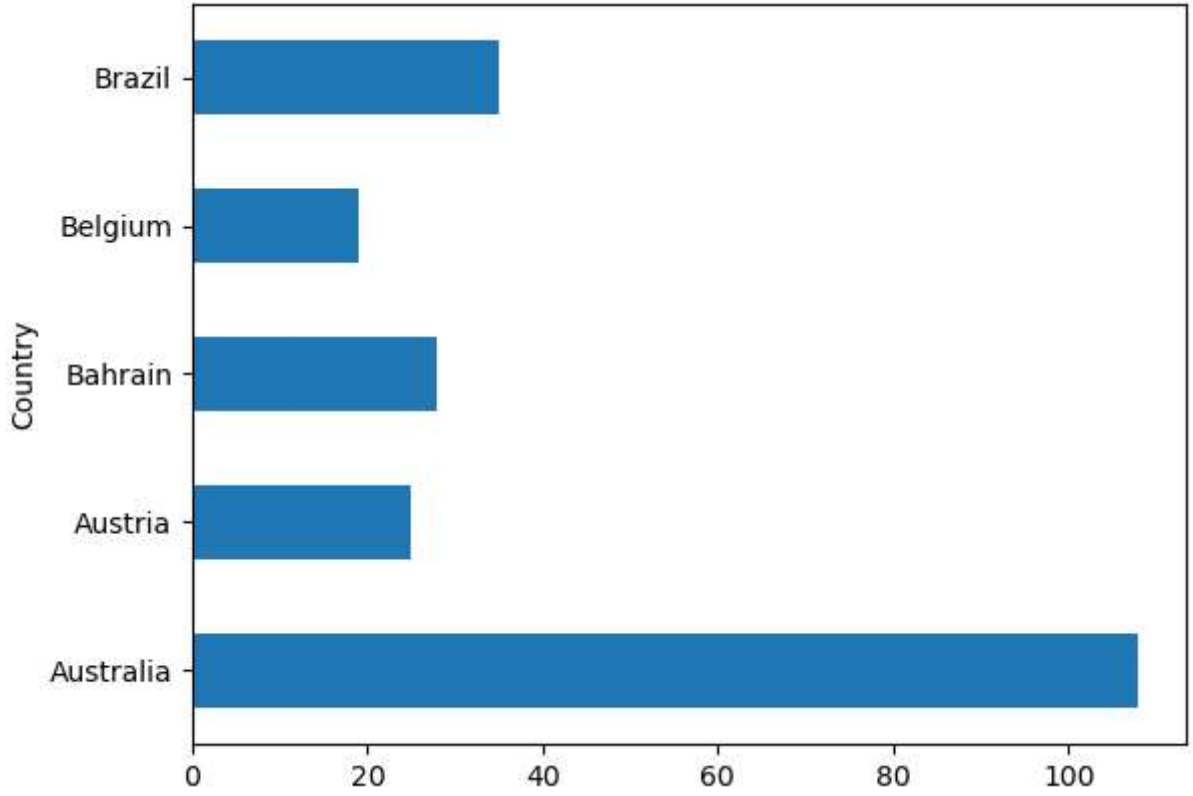


In [71]:
```
avg_sales2 = df.groupby('Country').Sales.mean().astype(int).head()
avg_sales
```

Out[71]:
```
Country
Australia               108
Austria                  25
Bahrain                  28
Belgium                  19
Brazil                   35
Canada                   24
Channel Islands          26
Cyprus                   20
Czech Republic           23
Denmark                  48
European Community       21
Finland                  32
France                   23
Germany                  23
Greece                   32
Hong Kong                35
Iceland                  23
Ireland                  32
Israel                   26
Italy                    21
Japan                    98
Lebanon                  37
Lithuania                47
Malta                    19
Netherlands             120
Norway                   32
Poland                   21
Portugal                 19
RSA                      17
Saudi Arabia             13
Singapore                39
Spain                    21
Sweden                   79
Switzerland              28
USA                       5
United Arab Emirates     27
United Kingdom           16
Unspecified              10
Name: Sales, dtype: int32
```

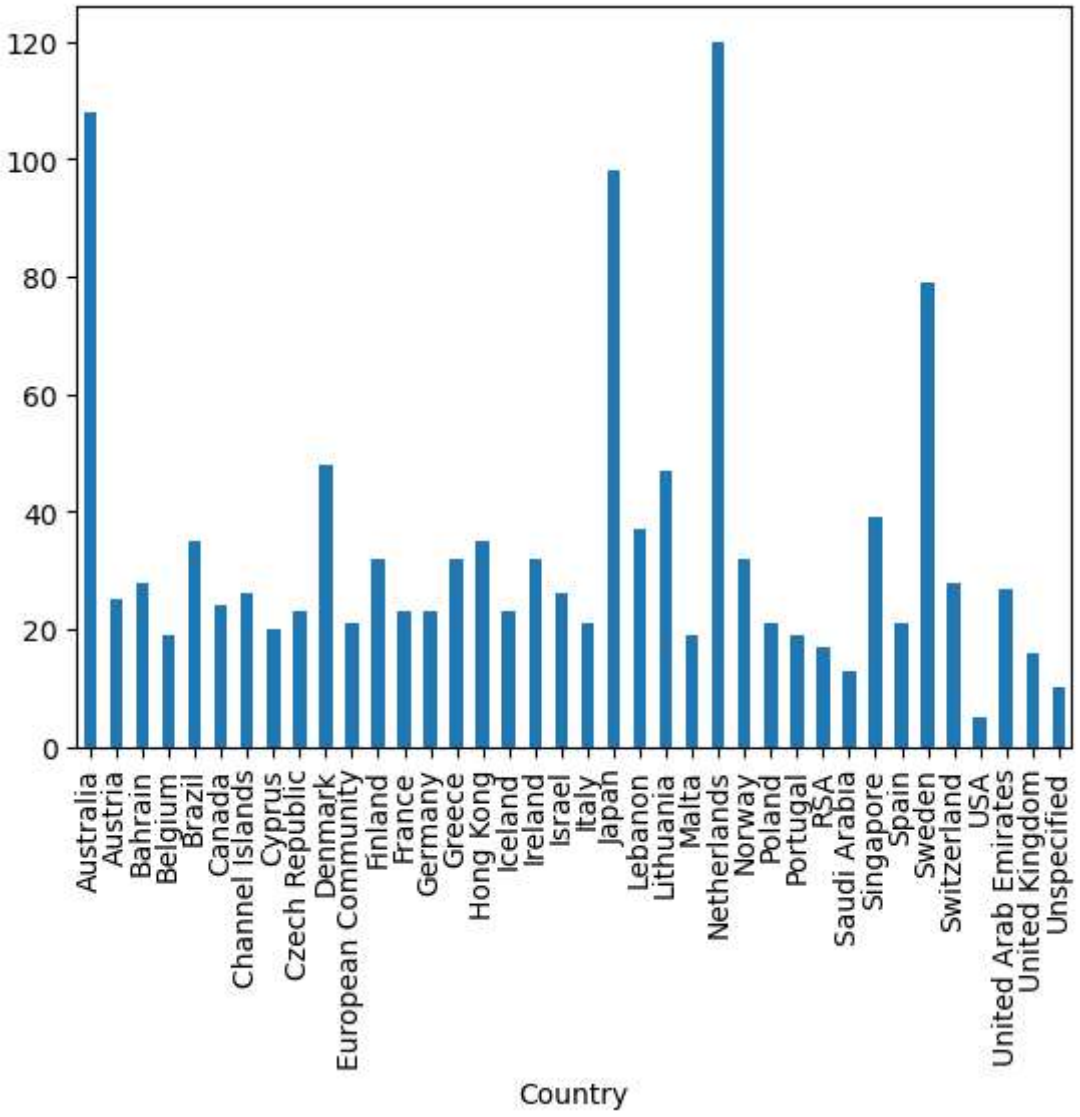
In [72]:
```
avg_sales2.plot.barh()
```

Out[72]:
```
<AxesSubplot:ylabel='Country'>
```



In [73]:
```
avg_sales.plot.bar()
```

Out[73]:
```
<AxesSubplot:xlabel='Country'>
```



In [74]:
```
# We want to talk about Sales by Trend of this Data set, and we often use Line chart for this purpose

df.head()
```

Out[74]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 08:26	2.55	17850	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 08:26	2.75	17850	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 08:26	3.39	17850	United Kingdom	20.34

In [75]:
```
# we will use the invoce date column as it contains the date of sales.
# First we want to convert the coulum (invoice date) to date time data type.
df['invoiceDate'] = pd.to_datetime(df.InvoiceDate)
```

In [76]:
```
# we want to check if the conversion works well
df.dtypes
```

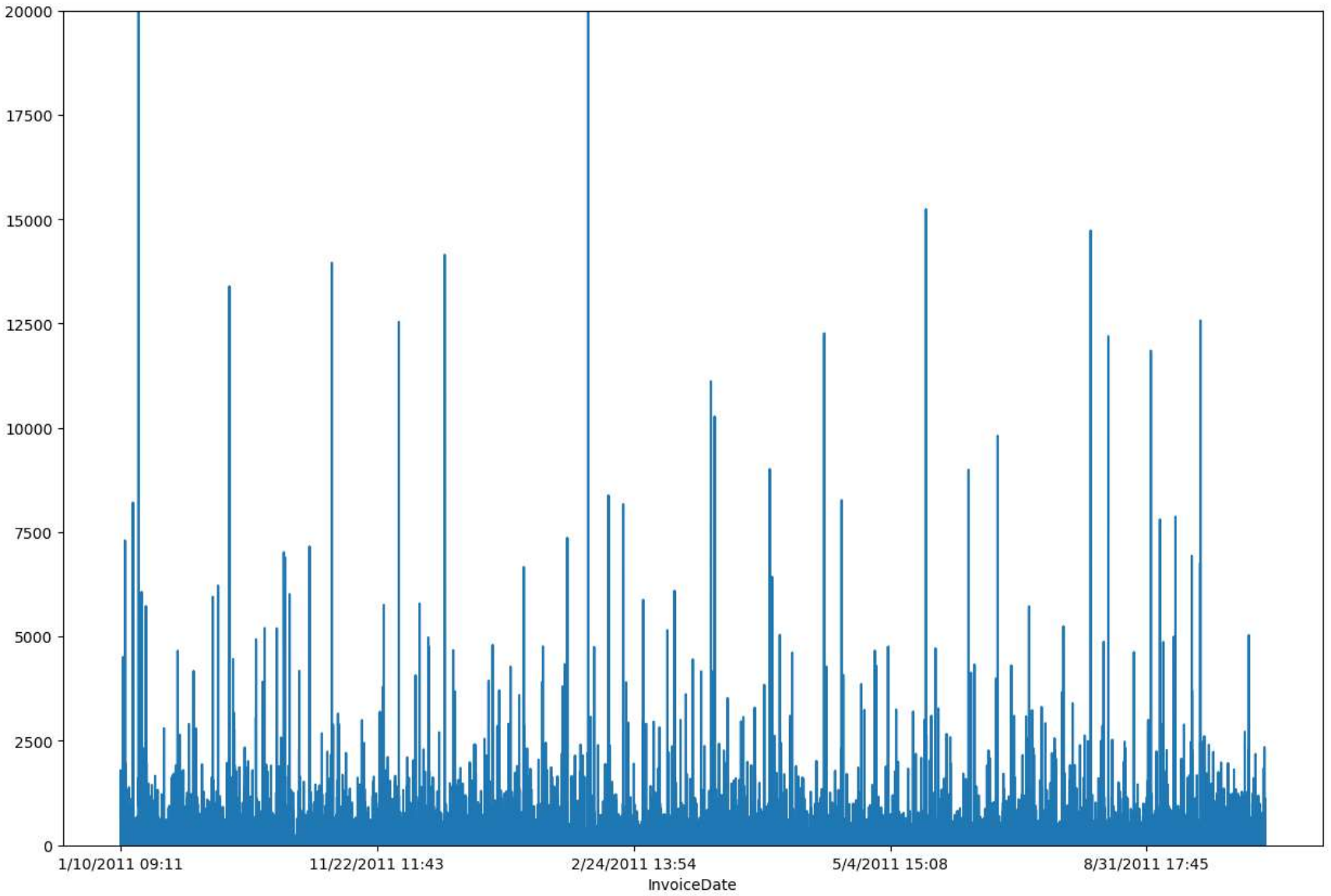
Out[76]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate            object
UnitPrice             float64
CustomerID             object
Country                object
Sales                 float64
invoiceDate    datetime64[ns]
dtype: object
```

In [77]:
```
# We want to now plot the trend using quantity by date

qty_trnd = df.groupby('InvoiceDate').Quantity.sum()
qty_trnd
```

Out[77]:
```
InvoiceDate
1/10/2011 09:11     643
1/10/2011 09:43    1797
1/10/2011 09:44     196
1/10/2011 09:48     243
1/10/2011 09:55      81
                    ...
9/9/2011 15:53     1128
9/9/2011 16:13      492
9/9/2011 16:34       -1
9/9/2011 16:36      -13
9/9/2011 17:52        4
Name: Quantity, Length: 22309, dtype: int64
```

In [78]:
```
# As seen above we have the date and quantity sold as above. Our convertion was okay and we can now visualize using a l

plt.figure(figsize = (15, 10))
qty_trnd.plot()
plt.ylim(0, 20000)
```

Out[78]:
```
(0.0, 20000.0)
```



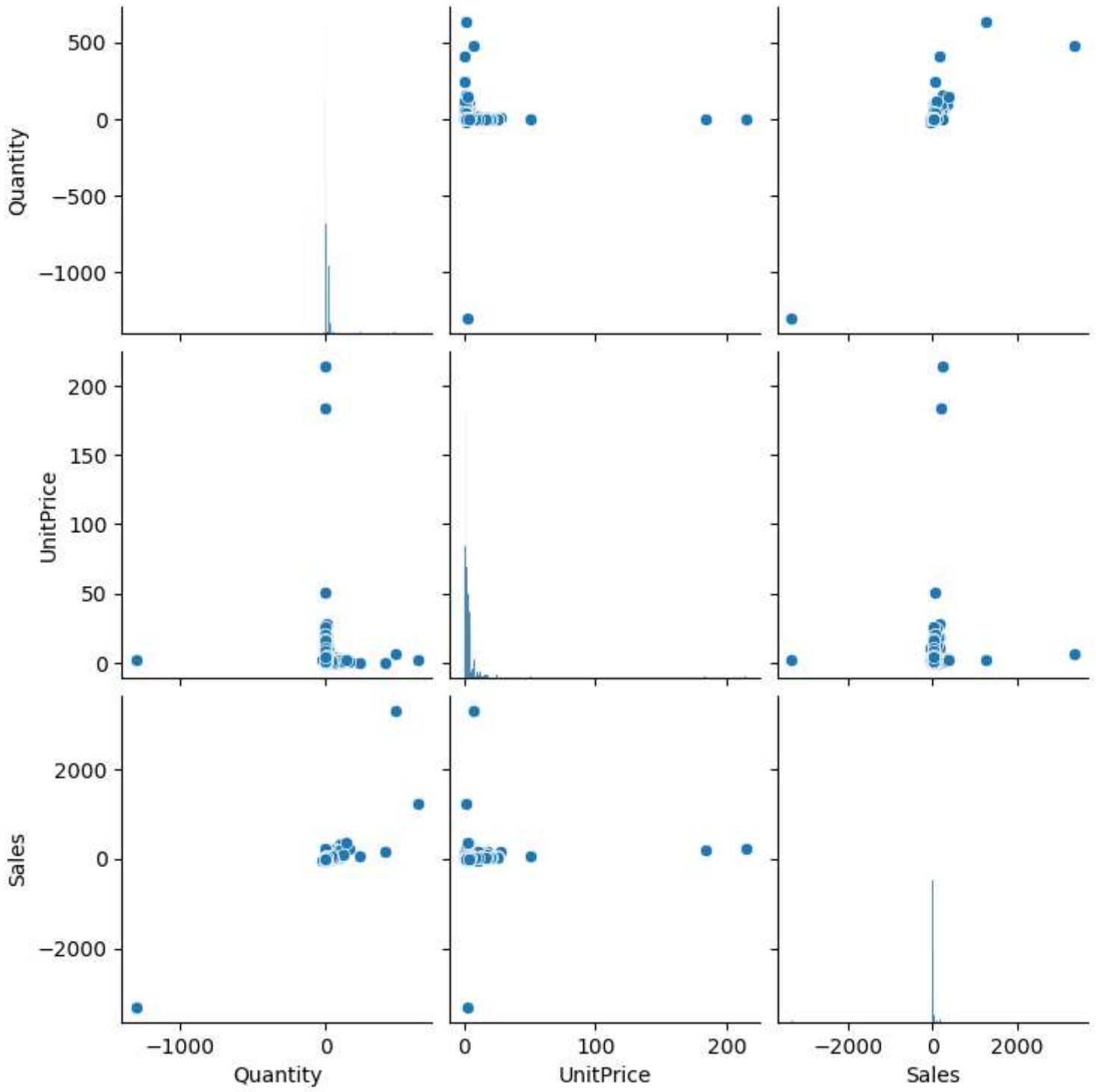
In [79]:
```
# We now want to check ealiest date and latest date
print('Earliest Date: ', df.InvoiceDate.min())
print('Latest Date: ', df.InvoiceDate.max())
```

```
Earliest Date:  1/10/2011 09:11
Latest Date:  9/9/2011 17:52
```

Other Visualization

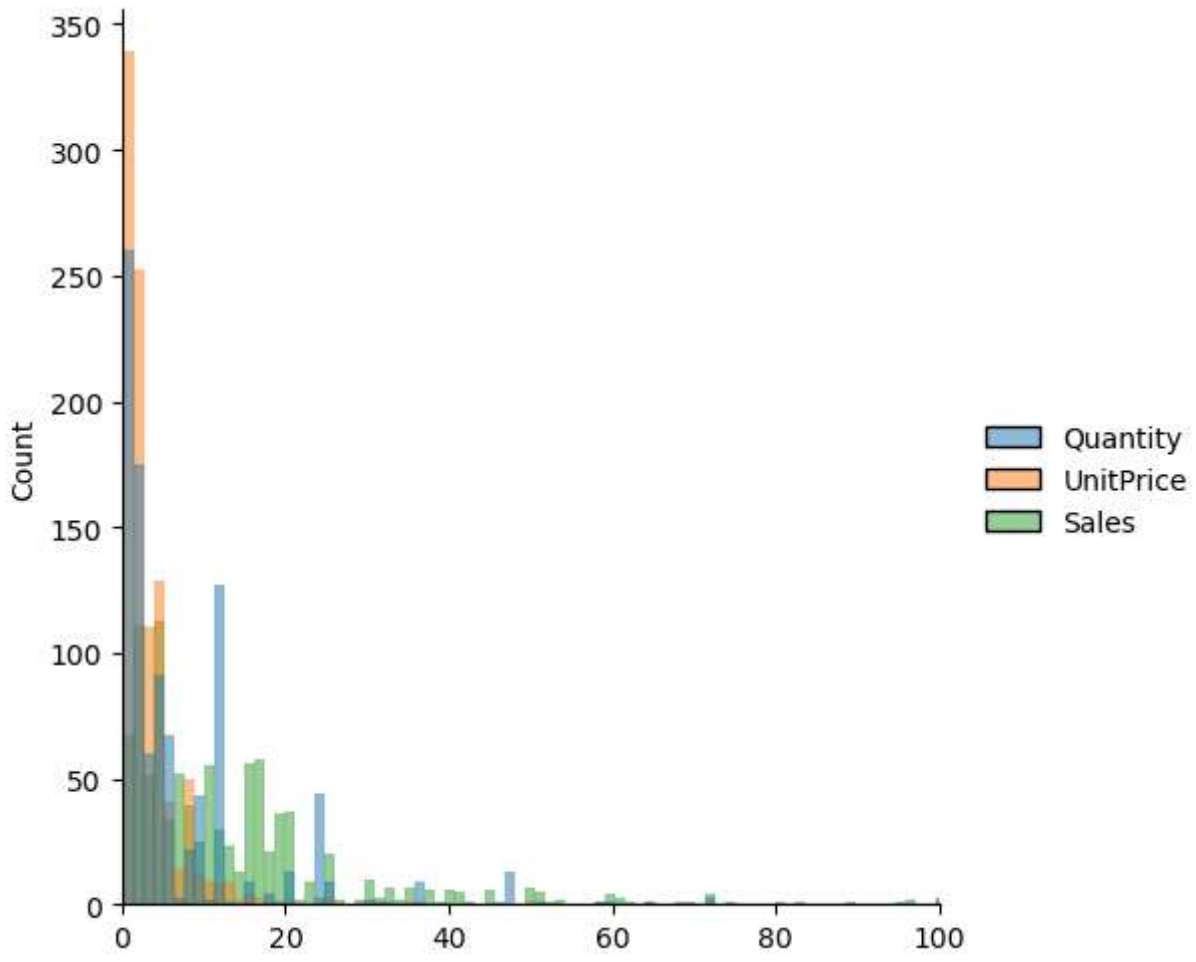
In [80]:
```
# First we want to view relationship between columns
# That mean we want to see how some values relate
sns.pairplot(df.sample(1000))
```

Out[80]:
```
<seaborn.axisgrid.PairGrid at 0x16152ab99a0>
```


Quantity
UnitPrice
Sales
500
0
−500
−1000
200
150
100
50
0
2000
0
−2000
−1000
0
100
200
−2000
0
2000
Quantity
UnitPrice
Sales

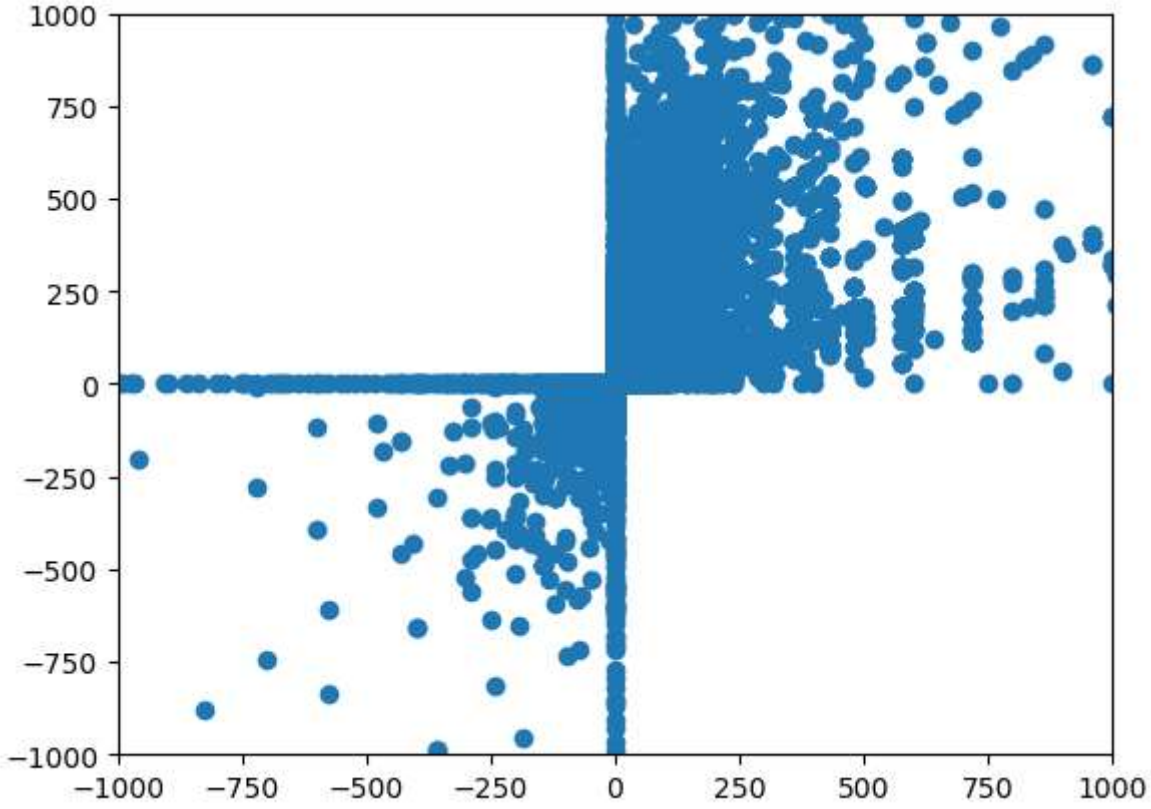
In [81]:
```
# We want to look at displot
sns.displot(data = df.sample(1000))
plt.xlim(0, 100)
```

Out[81]:
```
(0.0, 100.0)
```



In [82]:
```
# Sactter plot
plt.scatter(df.Quantity, df.Sales)
plt.xlim(-1000, 1000)
plt.ylim(-1000, 1000)
```

Out[82]:
```
(-1000.0, 1000.0)
```



Dont forget that there are negative values in the dataset a reason for using - 1000 so we want to see how the values relate both sides.

In [83]:
```
# here are the sales that are negative
df[df.Sales < - 2000]
```

Out[83]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales	invoiceDate
15016	C537630	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:04	13541.33	Unidentify	United Kingdom	-13541.33	2010-12-07 15:04:00
16232	C537644	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:34	13474.79	Unidentify	United Kingdom	-13474.79	2010-12-07 15:34:00
16313	C537647	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:41	5519.25	Unidentify	United Kingdom	-5519.25	2010-12-07 15:41:00
16356	C537651	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:49	13541.33	Unidentify	United Kingdom	-13541.33	2010-12-07 15:49:00
16357	C537652	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:51	6706.71	Unidentify	United Kingdom	-6706.71	2010-12-07 15:51:00
43702	C540117	AMAZONFEE	AMAZON FEE	-1	1/5/2011 09:55	16888.02	Unidentify	United Kingdom	-16888.02	2011-01-05 09:55:00
43703	C540118	AMAZONFEE	AMAZON FEE	-1	1/5/2011 09:57	16453.71	Unidentify	United Kingdom	-16453.71	2011-01-05 09:57:00
61624	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	1/18/2011 10:17	1.04	12346	United Kingdom	-77183.60	2011-01-18 10:17:00
96844	C544587	AMAZONFEE	AMAZON FEE	-1	2/21/2011 15:07	5575.28	Unidentify	United Kingdom	-5575.28	2011-02-21 15:07:00
96845	C544589	AMAZONFEE	AMAZON FEE	-1	2/21/2011 15:11	5258.77	Unidentify	United Kingdom	-5258.77	2011-02-21 15:11:00
119631	C546557	M	Manual	-1	3/15/2011 9:44	2583.76	Unidentify	Hong Kong	-2583.76	2011-03-15 09:44:00
124741	C546987	AMAZONFEE	AMAZON FEE	-1	3/18/2011 12:56	5693.05	Unidentify	United Kingdom	-5693.05	2011-03-18 12:56:00
124787	C546989	AMAZONFEE	AMAZON FEE	-1	3/18/2011 12:59	5225.03	Unidentify	United Kingdom	-5225.03	2011-03-18 12:59:00
144831	C548830	M	Manual	-1	4/4/2011 13:08	2382.92	12744	Singapore	-2382.92	2011-04-04 13:08:00
144834	C548834	M	Manual	-1	4/4/2011 13:09	2053.07	12744	Singapore	-2053.07	2011-04-04 13:09:00

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales	invoiceDate
160141	C550456	48185	DOORMAT FAIRY CAKE	-670	4/18/2011 13:08	6.75	15749	United Kingdom	-4522.50	2011-04-18 13:08:00
160142	C550456	47566B	TEA TIME PARTY BUNTING	-1300	4/18/2011 13:08	2.55	15749	United Kingdom	-3315.00	2011-04-18 13:08:00
160143	C550456	85123A	WHITE HANGING HEART T-LIGHT HOLDER	-1930	4/18/2011 13:08	2.55	15749	United Kingdom	-4921.50	2011-04-18 13:08:00
160144	C550456	21175	GIN + TONIC DIET METAL SIGN	-2000	4/18/2011 13:08	1.85	15749	United Kingdom	-3700.00	2011-04-18 13:08:00
160145	C550456	21108	FAIRY CAKE FLANNEL ASSORTED COLOUR	-3114	4/18/2011 13:08	2.10	15749	United Kingdom	-6539.40	2011-04-18 13:08:00
173277	C551685	POST	POSTAGE	-1	5/3/2011 12:51	8142.75	16029	United Kingdom	-8142.75	2011-05-03 12:51:00
173391	C551699	M	Manual	-1	5/3/2011 14:12	6930.00	16029	United Kingdom	-6930.00	2011-05-03 14:12:00
191385	C553354	AMAZONFEE	AMAZON FEE	-1	5/16/2011 13:54	5876.40	Unidentify	United Kingdom	-5876.40	2011-05-16 13:54:00
191386	C553355	AMAZONFEE	AMAZON FEE	-1	5/16/2011 13:58	7006.83	Unidentify	United Kingdom	-7006.83	2011-05-16 13:58:00
222681	C556445	M	Manual	-1	6/10/2011 15:31	38970.00	15098	United Kingdom	-38970.00	2011-06-10 15:31:00
239250	C558036	AMAZONFEE	AMAZON FEE	-1	6/24/2011 12:31	5791.18	Unidentify	United Kingdom	-5791.18	2011-06-24 12:31:00
239251	C558037	AMAZONFEE	AMAZON FEE	-1	6/24/2011 12:33	4534.24	Unidentify	United Kingdom	-4534.24	2011-06-24 12:33:00
262413	C559915	AMAZONFEE	AMAZON FEE	-1	7/13/2011 15:18	4383.62	Unidentify	United Kingdom	-4383.62	2011-07-13 15:18:00
262414	C559917	AMAZONFEE	AMAZON FEE	-1	7/13/2011 15:21	6497.47	Unidentify	United Kingdom	-6497.47	2011-07-13 15:21:00
268027	C560372	M	Manual	-1	7/18/2011 12:26	4287.63	17448	United Kingdom	-4287.63	2011-07-18 12:26:00

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales	invoiceDate
271151	C560647	M	Manual	-1	7/20/2011 11:31	3060.60	18102	United Kingdom	-3060.60	2011-07-20 11:31:00
287103	C562062	AMAZONFEE	AMAZON FEE	-1	8/2/2011 12:17	4575.64	Unidentify	United Kingdom	-4575.64	2011-08-02 12:17:00
287150	C562086	AMAZONFEE	AMAZON FEE	-1	8/2/2011 12:27	6721.37	Unidentify	United Kingdom	-6721.37	2011-08-02 12:27:00
293842	C562647	M	Manual	-1	8/8/2011 12:56	3155.95	15502	United Kingdom	-3155.95	2011-08-08 12:56:00
299983	A563186	B	Adjust bad debt	1	8/12/2011 14:51	-11062.06	Unidentify	United Kingdom	-11062.06	2011-08-12 14:51:00
299984	A563187	B	Adjust bad debt	1	8/12/2011 14:52	-11062.06	Unidentify	United Kingdom	-11062.06	2011-08-12 14:52:00
312092	C564340	AMAZONFEE	AMAZON FEE	-1	8/24/2011 14:50	4527.65	Unidentify	United Kingdom	-4527.65	2011-08-24 14:50:00
312246	C564341	AMAZONFEE	AMAZON FEE	-1	8/24/2011 14:53	6662.51	Unidentify	United Kingdom	-6662.51	2011-08-24 14:53:00
320580	C565044	22191	IVORY DINER WALL CLOCK	-318	8/31/2011 17:02	7.65	12931	United Kingdom	-2432.70	2011-08-31 17:02:00
342611	C566889	AMAZONFEE	AMAZON FEE	-1	9/15/2011 13:50	5522.14	Unidentify	United Kingdom	-5522.14	2011-09-15 13:50:00
342635	C566899	AMAZONFEE	AMAZON FEE	-1	9/15/2011 13:53	7427.97	Unidentify	United Kingdom	-7427.97	2011-09-15 13:53:00
347947	C567352	M	Manual	-1	9/19/2011 16:13	2653.95	Unidentify	Hong Kong	-2653.95	2011-09-19 16:13:00
349750	C567527	23113	PANTRY CHOPPING BOARD	-756	9/21/2011 9:16	5.06	17450	United Kingdom	-3825.36	2011-09-21 09:16:00
383495	C570025	AMAZONFEE	AMAZON FEE	-1	10/7/2011 10:29	5942.57	Unidentify	United Kingdom	-5942.57	2011-10-07 10:29:00
406404	C571750	M	Manual	-1	10/19/2011 11:16	3949.32	12744	Singapore	-3949.32	2011-10-19 11:16:00

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales	invoiceDate
406405	C571750	M	Manual	-1	10/19/2011 11:16	2118.74	12744	Singapore	-2118.74	2011-10-19 11:16:00
422375	C573079	M	Manual	-2	10/27/2011 14:15	4161.06	12536	France	-8322.12	2011-10-27 14:15:00
429248	C573549	AMAZONFEE	AMAZON FEE	-1	10/31/2011 13:23	5942.57	Unidentify	United Kingdom	-5942.57	2011-10-31 13:23:00
429249	C573550	AMAZONFEE	AMAZON FEE	-1	10/31/2011 13:32	2185.04	Unidentify	United Kingdom	-2185.04	2011-10-31 13:32:00
446434	C574897	AMAZONFEE	AMAZON FEE	-1	11/7/2011 15:03	5877.18	Unidentify	United Kingdom	-5877.18	2011-11-07 15:03:00
446533	C574902	AMAZONFEE	AMAZON FEE	-1	11/7/2011 15:21	8286.22	Unidentify	United Kingdom	-8286.22	2011-11-07 15:21:00
524601	C580604	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:35	11586.50	Unidentify	United Kingdom	-11586.50	2011-12-05 11:35:00
524602	C580605	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:36	17836.46	Unidentify	United Kingdom	-17836.46	2011-12-05 11:36:00
540422	C581484	23843	PAPER CRAFT , LITTLE BIRDIE	-80995	12/9/2011 09:27	2.08	16446	United Kingdom	-168469.60	2011-12-09 09:27:00

Questions

- Lets say you are the store manager:
- 1. What is the invoice number that start with C?
- 1. How much sales has negative quantity?
-

In [55]:
```
df[df.Sales < 0]
```

Out[55]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sales
141	C536379	D	Discount	-1	12/1/2010 09:41	27.50	14527	United Kingdom	-27.50
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	12/1/2010 09:49	4.65	15311	United Kingdom	-4.65
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	12/1/2010 10:24	1.65	17548	United Kingdom	-19.80
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	12/1/2010 10:24	0.29	17548	United Kingdom	-6.96
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	12/1/2010 10:24	0.29	17548	United Kingdom	-6.96
...	...	...	...	...	...	...	...	...	...
540449	C581490	23144	ZINC T-LIGHT HOLDER STARS SMALL	-11	12/9/2011 09:57	0.83	14397	United Kingdom	-9.13
541541	C581499	M	Manual	-1	12/9/2011 10:28	224.69	15498	United Kingdom	-224.69
541715	C581568	21258	VICTORIAN SEWING BOX LARGE	-5	12/9/2011 11:57	10.95	15311	United Kingdom	-54.75
541716	C581569	84978	HANGING HEART JAR T-LIGHT HOLDER	-1	12/9/2011 11:58	1.25	17315	United Kingdom	-1.25
541717	C581569	20979	36 PENCILS TUBE RED RETROSPOT	-5	12/9/2011 11:58	1.25	17315	United Kingdom	-6.25

9290 rows × 9 columns

In [56]:
```
# we want to know how many
# Below is the number of records that has negative

df[df.Sales < 0].shape[0]
```

Out[56]:
```
9290
```

Create Interactive plot with Ploty and Cufflinks

Ploty:

- is Python Library that is interactive. it is also an open source plotting library that support over 40 unique chart types, covering a wide range of statistical, financial, geographic, scientfic and 3 - dimentional use cases.

Cufflinks:

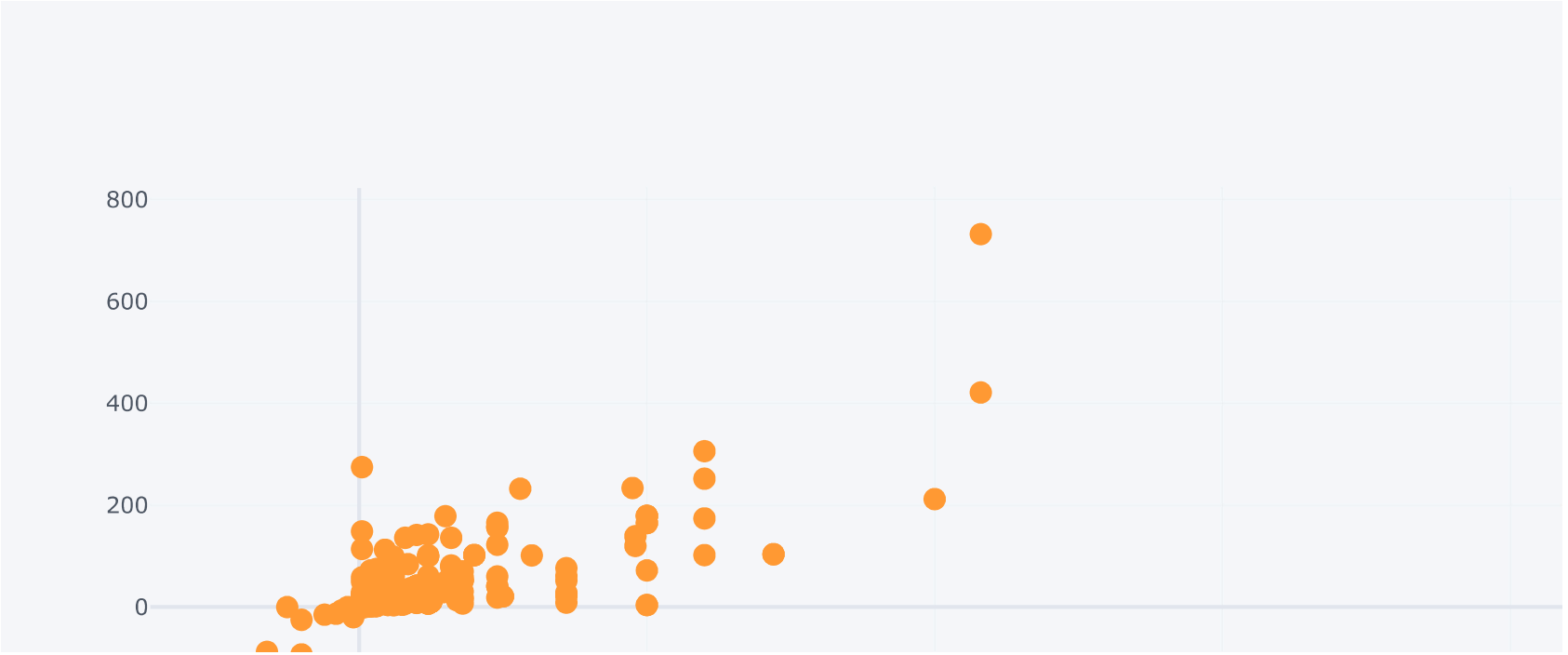
- is also a Python Library that conect plotyy with pandas so that we can create charts directly on a dataframe. it basically acts as a plugin.

In [57]:
```
# Now we want to import plotly and cufflinks after instalation

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import cufflinks as cf
```

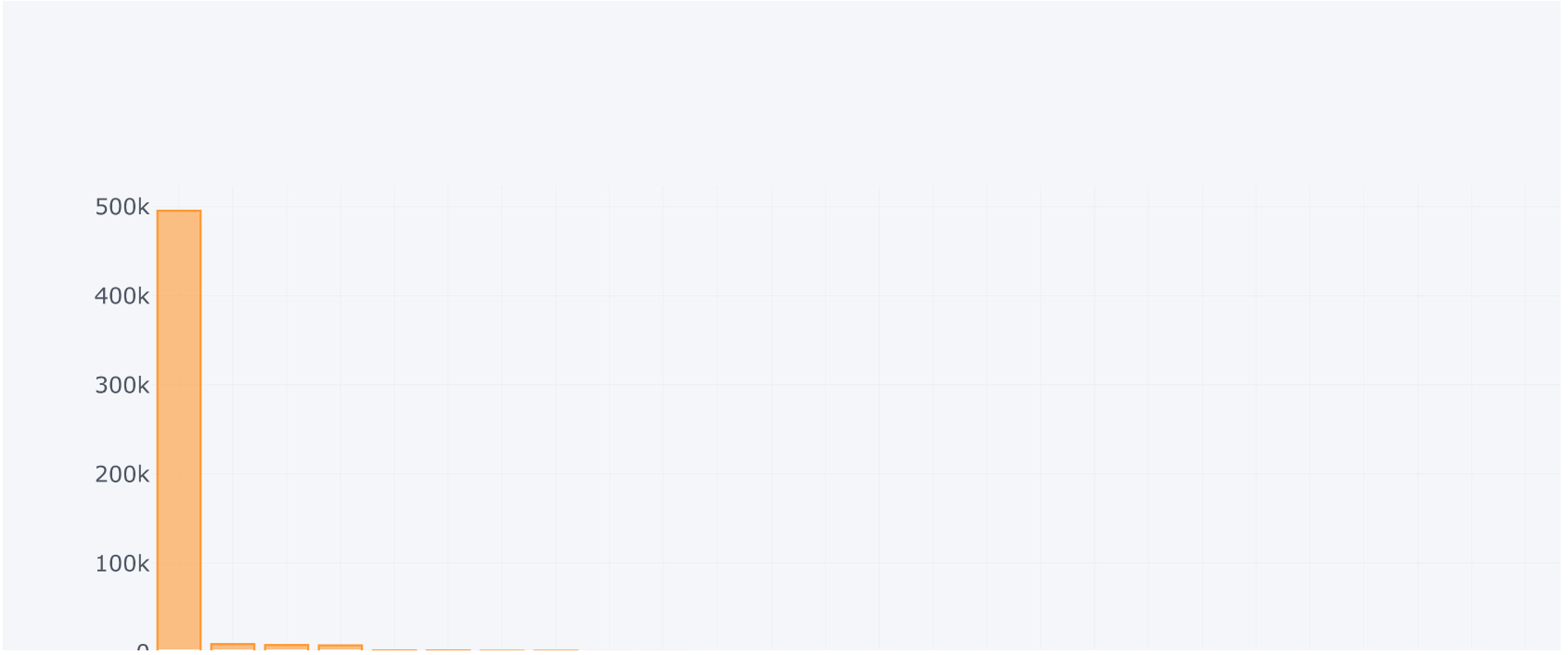
In [58]:
```
# Now we want to initialize plotly
# To initialize means we want to be able to use plotly ofline
init_notebook_mode(connected = True)
cf.go_offline()
```

Now we want to visualize using this Plotly and Cufflinks

In [59]:
```
# We will be plotting Quantity against Sales

df.sample(1000).iplot(kind = 'scatter', x = 'Quantity', y = 'Sales', mode = 'markers')
```



In [60]:
```
# Using bar chart we want to see the count of country

df.Country.value_counts().iplot(kind = 'bar')
```



Pandas Profiling

- This is a package that allows you to create intractive column by column analysis. It can automate the process of basic PDA for your analysis.
- - Now we will Pip instal pandas profilling

Using MarkupSafe

this is done because markupsafe will help to implement text object that escapes character so as to make it safe to use in HTML or XML
This means characters that has extra meaning will be seen as actual character

In [62]:
```
import markupsafe
```

In [63]:
```
markupsafe.__version__
```

Out[63]:
```
'2.0.1'
```

In [64]:
```
from pandas_profiling import ProfileReport
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_10316\2274191625.py:1: DeprecationWarning:

`import pandas_profiling` is going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
```

In [96]:
```
#Create Profile
ProfileReport(df)
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	10
Number of observations	540455
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	4879
Duplicate rows (%)	0.9%
Total size in memory	61.5 MiB
Average record size in memory	119.3 B

Variable types

Categorical	6
Numeric	3
DateTime	1

Alerts

Dataset has 4879 (0.9%) duplicate rows	**Duplicates**
`InvoiceNo` has a high cardinality: 24446 distinct values	**High cardinality**
`StockCode` has a high cardinality: 3958 distinct values	**High cardinality**
`Description` has a high cardinality: 4211 distinct values	**High cardinality**
`InvoiceDate` has a high cardinality: 22309 distinct values	**High cardinality**

Out[96]:

END

In [ ]: