In [1]:
```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
```
```
Matplotlib is building the font cache; this may take a moment.
```

Let's look at the Data:

In [2]:
```
dataSet = pd.read_excel('data.xlsx')
```

In [3]:
```
dataSet.head(10)
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom

In [4]:
```
dataSet.columns
```

Out[4]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

InvoiceNo: Payment Number

StockCode: Product "code"

Description: What is the exact title of the product

Quantity: How many was bought

InvoiceDate: When was this purchase made year-month-day

UnitPrice: Price of product

CustomerID: who bought it

Country: country of the customer

In [5]:
```
#I will change the name of columns for easy access

dataSet.columns = ['invoNum', 'prodCode', 'Desc', 'Quant', 'PurDate', 'Price', 'I
```

In [6]:
```
# checking to see if data matches what I expect in terms of "type"
dataSet.dtypes
```

Out[6]:
```
invoNum             object
prodCode            object
Desc                object
Quant                int64
PurDate     datetime64[ns]
Price              float64
ID                 float64
Country             object
dtype: object
```

Looks like "invoNum" might have some values that are not integers

```
In [7]: non_ints_invoNum = dataSet.invoNum.str.contains('[^0-9.-]')
        dataSet.loc[non_ints_invoNum].head()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-5ec2192805ad> in <module>
      1 non_ints_invoNum = dataSet.invoNum.str.contains('[^0-9.-]')
----> 2 dataSet.loc[non_ints_invoNum].head()

E:\Programming\lib\site-packages\pandas\core\indexing.py in __getitem__(self, k
ey)
    877 
    878             maybe_callable = com.apply_if_callable(key, self.obj)
--> 879             return self._getitem_axis(maybe_callable, axis=axis)
    880 
    881     def _is_scalar_access(self, key: Tuple):

E:\Programming\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self,
key, axis)
   1087             self._validate_key(key, axis)
   1088             return self._get_slice_axis(key, axis=axis)
-> 1089         elif com.is_bool_indexer(key):
   1090             return self._getbool_axis(key, axis=axis)
   1091         elif is_list_like_indexer(key):

E:\Programming\lib\site-packages\pandas\core\common.py in is_bool_indexer(key)
    132                 na_msg = "Cannot mask with non-boolean array containing
NA / NaN values"
    133                 if isna(key).any():
--> 134                     raise ValueError(na_msg)
    135                 return False
    136             return True

ValueError: Cannot mask with non-boolean array containing NA / NaN values
```

The last "ValueError" proves there are some NA/NAN values in this column.

```
In [8]: pd.isnull(dataSet[invoNum])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-668d91cb00f7> in <module>
----> 1 pd.isnull(dataSet[invoNum])

NameError: name 'invoNum' is not defined
```

In [10]:
```
pd.isnull(dataSet['invoNum'])
```

Out[10]:
```
0         False
1         False
2         False
3         False
4         False
          ...  
541904    False
541905    False
541906    False
541907    False
541908    False
Name: invoNum, Length: 541909, dtype: bool
```

In [13]:
```
pd.isnull(dataSet['invoNum']).sum()
```

Out[13]:
```
0
```

Let's see how many different customers there are. There are gonna be duplicates, so this will take care of that.

In [16]:
```
customers_count = len(dataSet['ID'].unique())
customers_count
```

Out[16]:
```
4373
```

In [17]:
```
#How many records do we have?

dataSet.index
```

Out[17]:
```
RangeIndex(start=0, stop=541909, step=1)
```

There are 541,910 purchases from 4373 customers.

In [18]:
```
#Summary Statistics

dataSet.describe()
```

Out[18]:

	Quant	Price	ID
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

In [19]:
```
#How much does the most expensive item cost?
dataSet['Price'].max()
```
Out[19]:
```
38970.0
```

In [20]:
```
#How much does the least expensive item cost?
dataSet['Price'].min()
```
Out[20]:
```
-11062.06
```

That doesn't make any sense. A price cannot be negative. Let's check this out.

In [25]:
```
def count_negatives():
    count = 0

    arr = dataSet['Price'].to_numpy()

    for i in range(arr.size):
        if (arr[i] < 0):
            count += 1

    return count

count_negatives()
```
Out[25]:
```
2
```

This means only 2 values are negative, so I'll just replace those with the median price.

In [26]:
```
#calculate median price
m_price = dataSet['Price'].median()

#find index of negative price values
for i in range(arr.size):
    if (arr[i] < 0):
        print(i)
```
```
299983
299984
```

In [28]:
```
#replace those values

dataSet['Price'].loc[299983]= m_price
dataSet['Price'].loc[299984]= m_price
```

In [29]:
```
count_negatives()
```
Out[29]:
```
0
```