```
from google.colab import files
files.upload()
```

```
Choose Files com_Train.csv
• com_Train.csv(application/vnd.ms-excel) - 15175356 bytes, last modified: 11/6/2020 - 100% done
Saving com_Train.csv to com_Train.csv
{'com_Train.csv': b'InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,Cu
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('com_Train.csv')
```

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	6141	1583	144	3	2011-05-06 16:54:00	3.75	14056.0	
1	6349	1300	3682	6	2011-05-11 07:35:00	1.95	13098.0	
2	16783	2178	1939	4	2011-11-20 13:20:00	5.95	15044.0	
3	16071	2115	2082	1	2011-11-22	0.83	15525.0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284780 entries, 0 to 284779
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    284780 non-null  int64  
 1   StockCode    284780 non-null  int64  
 2   Description  284780 non-null  int64  
 3   Quantity     284780 non-null  int64  
 4   InvoiceDate  284780 non-null  object 
 5   UnitPrice    284780 non-null  float64
 6   CustomerID   284780 non-null  float64
 7   Country      284780 non-null  int64  
dtypes: float64(2), int64(5), object(1)
memory usage: 17.4+ MB
```

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

```
InvoiceNo      0
StockCode      0
Description    0
```

```
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

```
df['Quantity']=df['Quantity'].abs()
```

```
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
```

```
df["month"] = df['InvoiceDate'].dt.month
```

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	6141	1583	144	3	2011-05-06 16:54:00	3.75	14056.0	
1	6349	1300	3682	6	2011-05-11 07:35:00	1.95	13098.0	
2	16783	2178	1939	4	2011-11-20 13:20:00	5.95	15044.0	
3	16971	2115	2983	1	2011-11-22	0.83	15535.0	

```
df.drop(['InvoiceDate'], axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284780 entries, 0 to 284779
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    284780 non-null  int64  
 1   StockCode    284780 non-null  int64  
 2   Description  284780 non-null  int64  
 3   Quantity     284780 non-null  int64  
 4   UnitPrice    284780 non-null  float64
 5   CustomerID   284780 non-null  float64
 6   Country      284780 non-null  int64  
 7   month        284780 non-null  int64  
dtypes: float64(2), int64(6)
memory usage: 17.4 MB
```

```
df.corr()
```

	InvoiceNo	StockCode	Description	Quantity	UnitPrice	CustomerID	Cou
InvoiceNo	1.000000	0.086809	0.024804	0.002176	0.007927	-0.007463	0.0(
StockCode	0.086809	1.000000	-0.013230	0.002186	0.020073	0.002887	0.0(
Description	0.024804	-0.013230	1.000000	0.000794	-0.000424	-0.005227	-0.0
Quantity	0.002176	0.002186	0.000794	1.000000	-0.001011	-0.004811	-0.0(
UnitPrice	0.007927	0.020073	0.000424	0.001011	1.000000	0.004932	0.0(

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True)
plt.show()
```

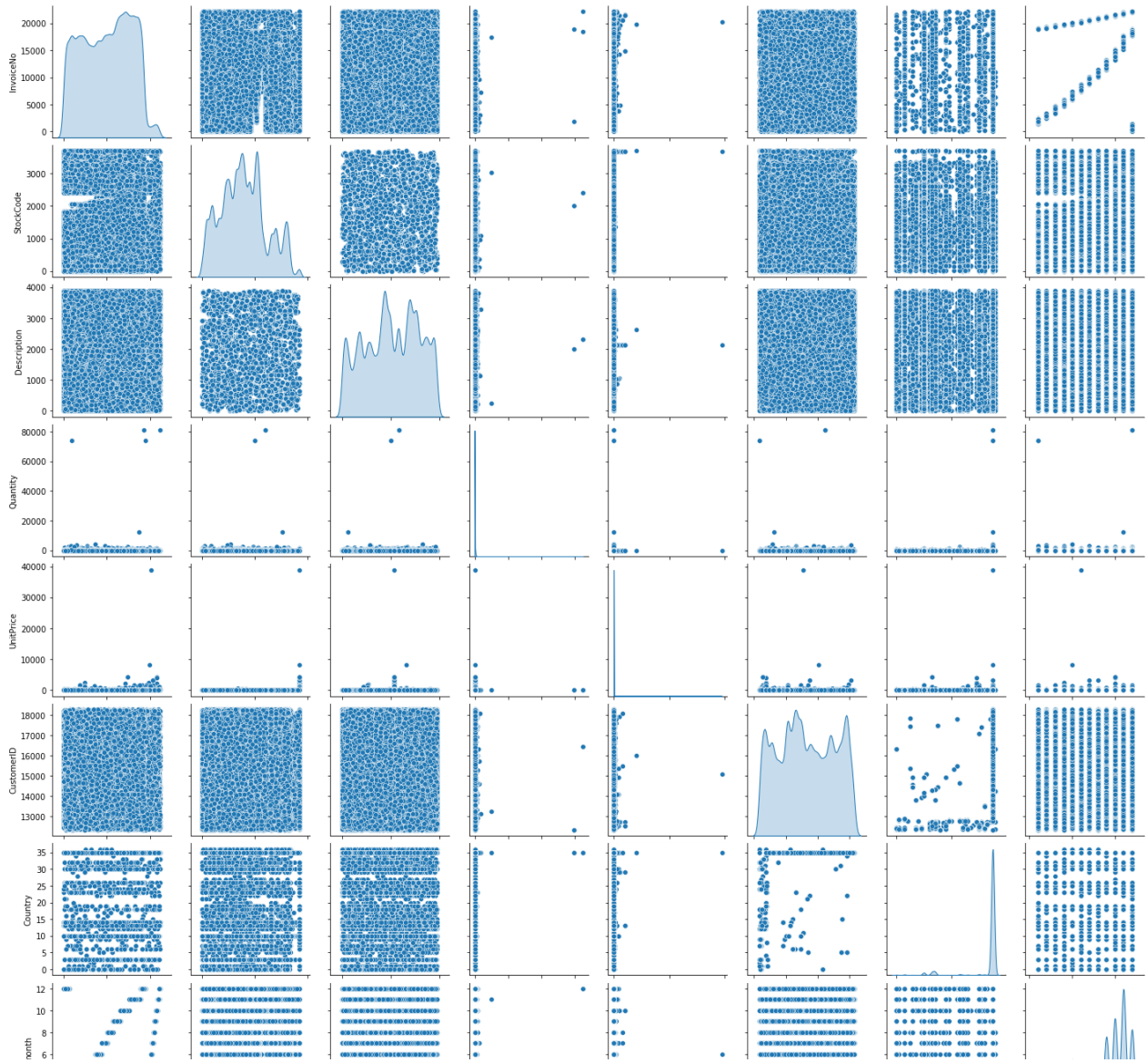

```
new_df=df[['InvoiceNo','StockCode','Description','Quantity','month','CustomerID','Country'

from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(x):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = x.columns
    vif["VIF"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]

    return(vif)
```

```
x = new_df
calc_vif(x)
```

	variables	VIF
0	InvoiceNo	7.240477
1	StockCode	4.375989
2	Description	4.280011
3	Quantity	1.002136
4	month	10.155034
5	CustomerID	36.449234
6	Country	28.367738

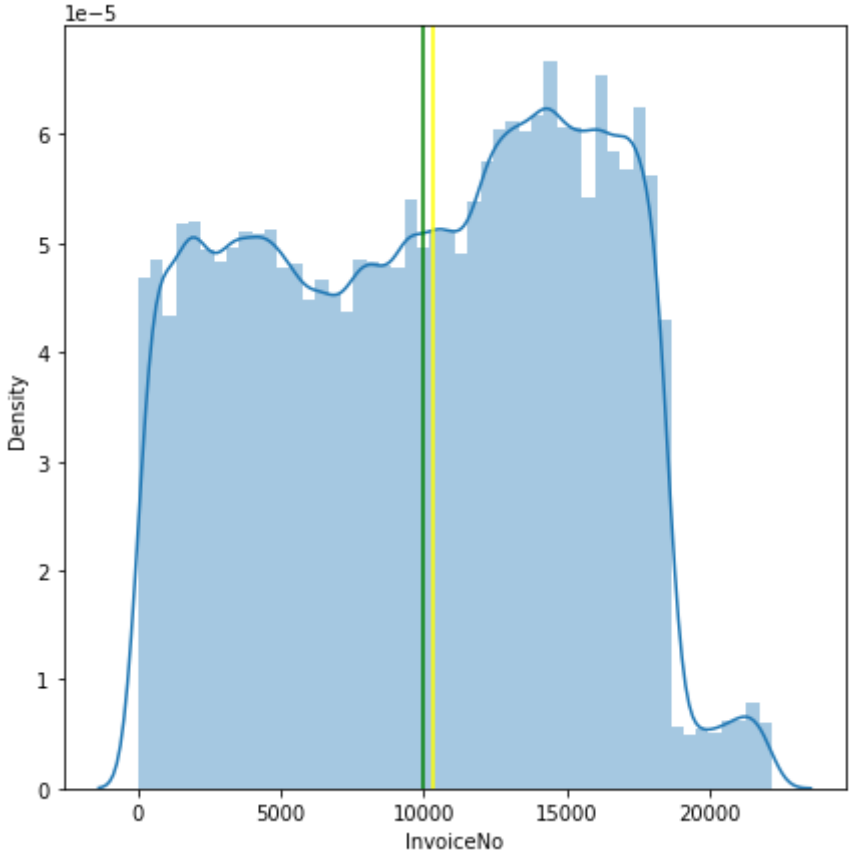
```
plt.figure(figsize=(14, 14))
sns.pairplot(df, diag_kind='kde');
```

```
<Figure size 1008x1008 with 0 Axes>
```

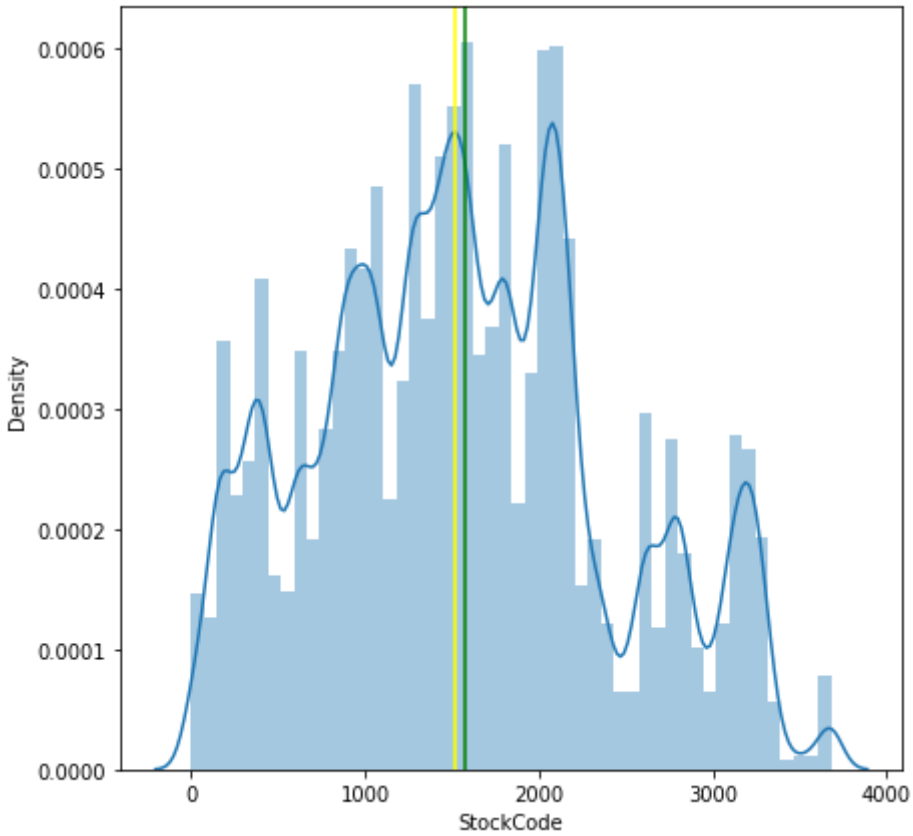



```
for cols in df:
    print(cols)
    width_mean = df[cols].mean()
    width_median = df[cols].median()
    plt.figure(figsize=(7,7))
    sns.distplot(df[cols])
    plt.axvline(width_mean, color="green")
    plt.axvline(width_median, color="yellow")
    plt.show()
    print('-'*100)
```

```
InvoiceNo
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

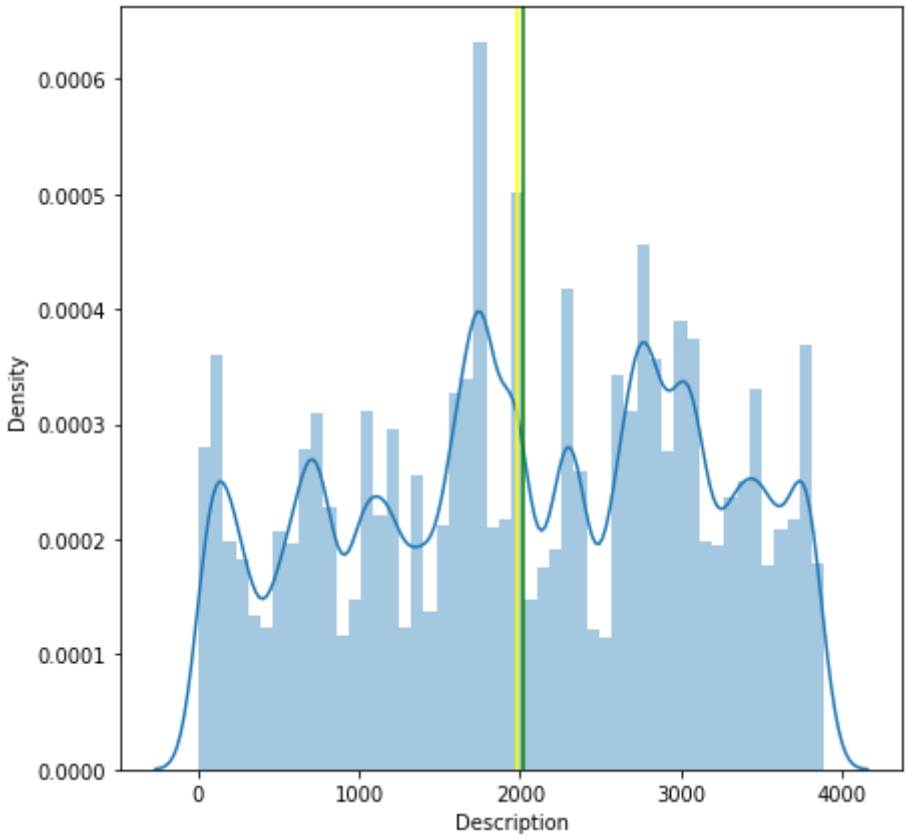


```
StockCode
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

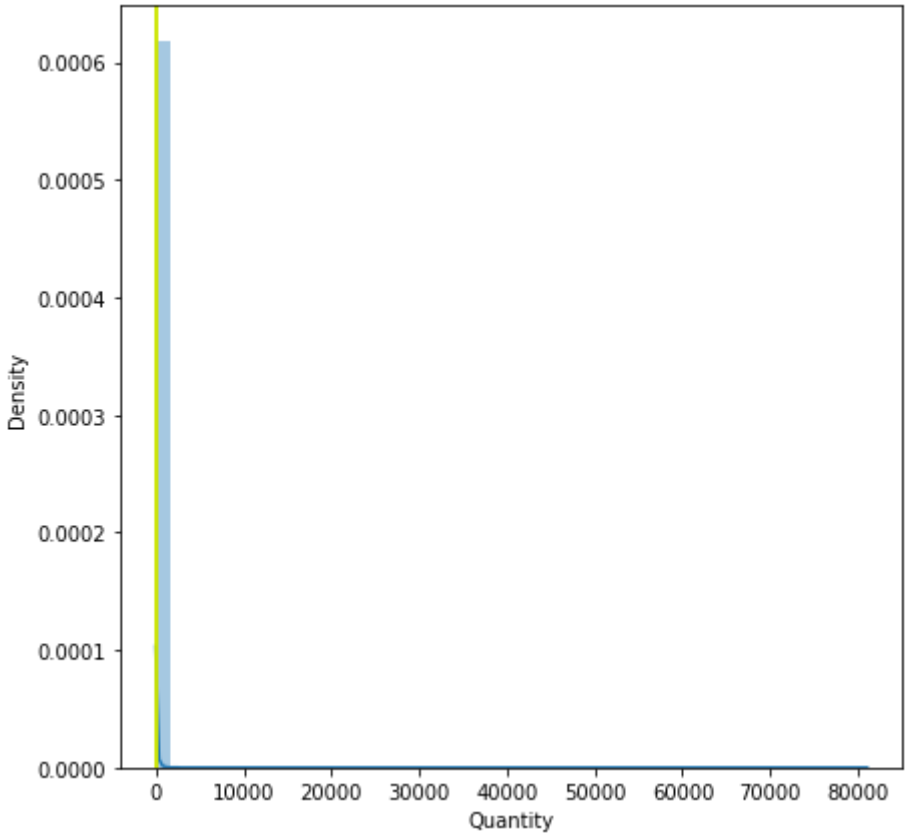


```
Description
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
```

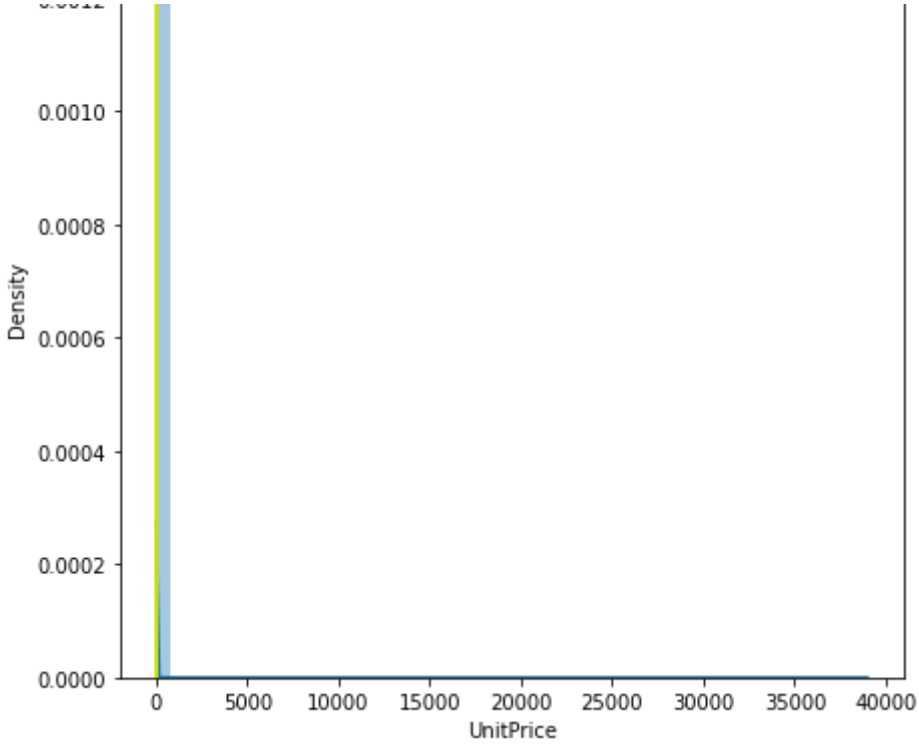
warnings.warn(msg, FutureWarning)



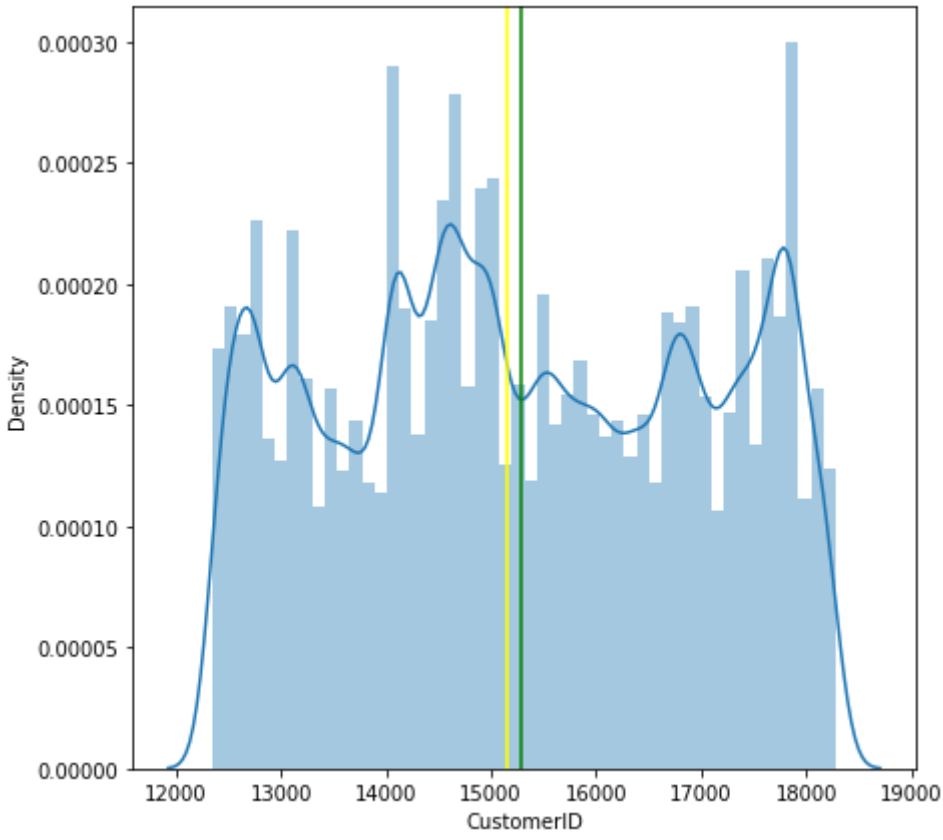
Quantity
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
warnings.warn(msg, FutureWarning)



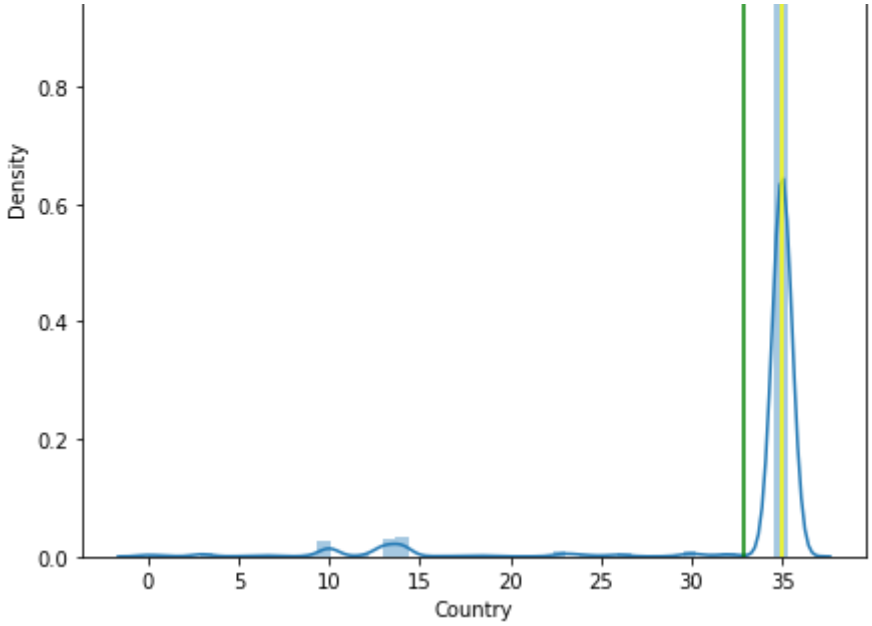
UnitPrice
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
warnings.warn(msg, FutureWarning)



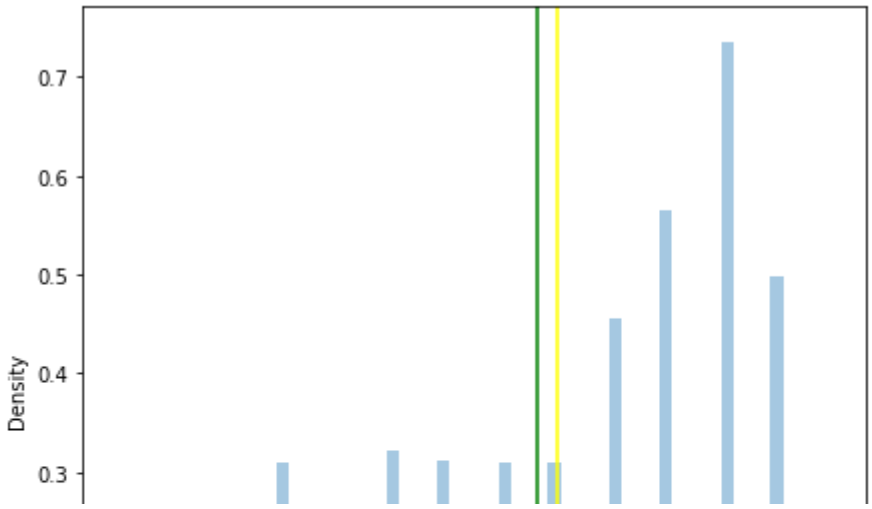
```
CustomerID
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



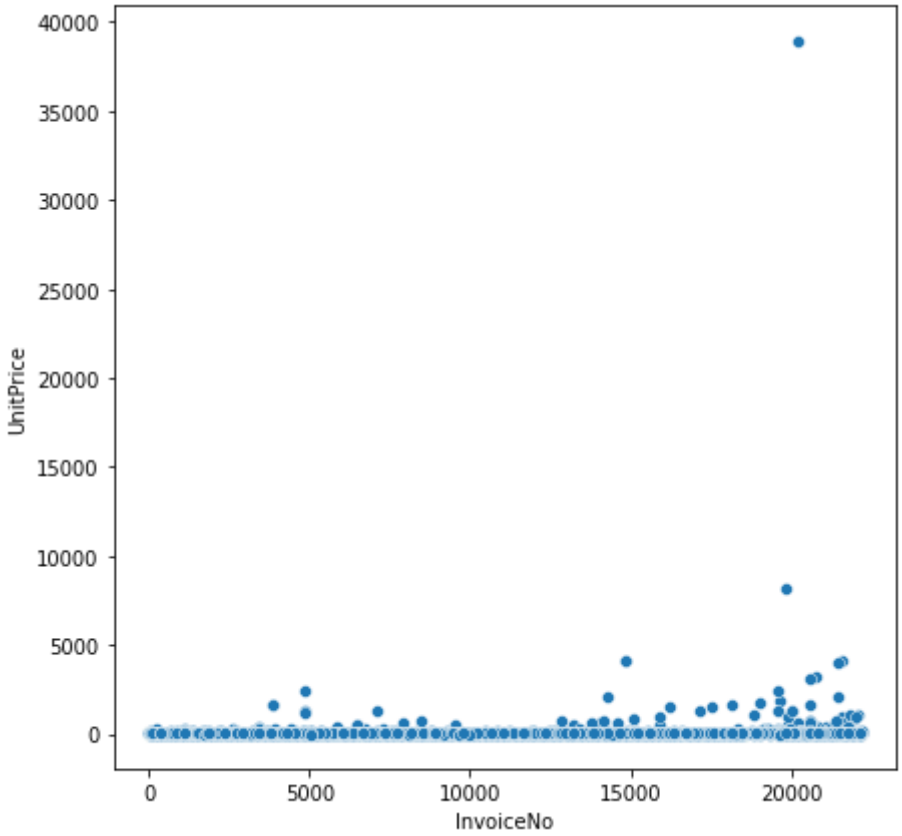
```
Country
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

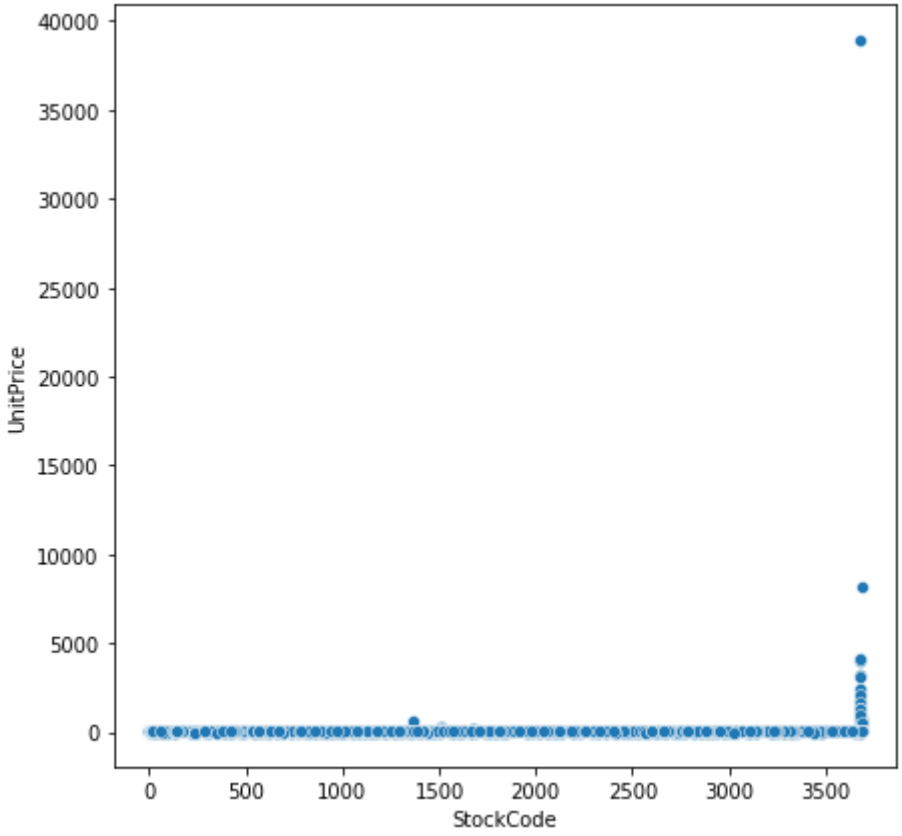



```
--------------------------------------------------------------------------------
month
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

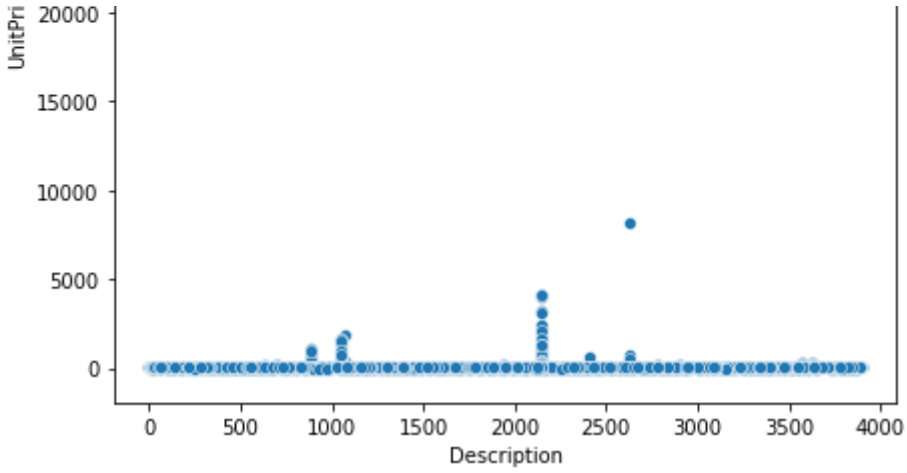


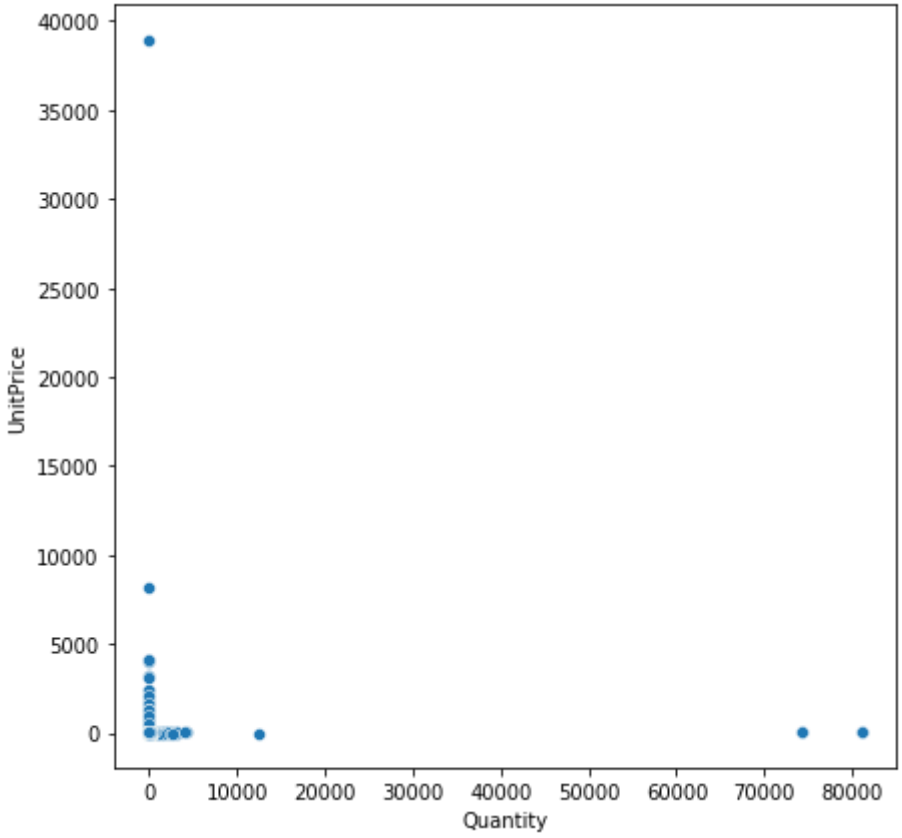

```
for col in df:
    plt.figure(figsize=(7,7))
    sns.scatterplot(data=df,x=col,y="UnitPrice") #always keep the target on y axis
    plt.show()
```

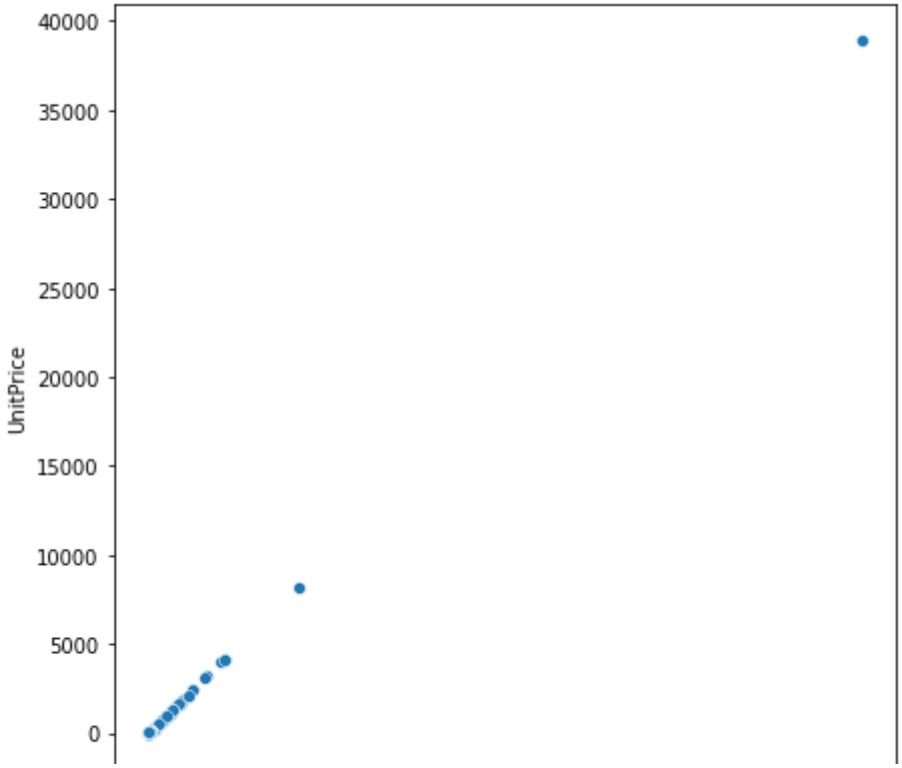

40000
35000
30000
25000
20000
15000
10000
5000
0
UnitPrice
0
5000
10000
15000
20000
InvoiceNo

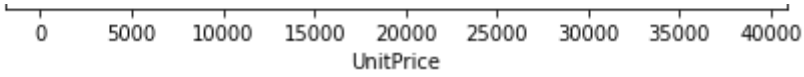

40000
35000
30000
25000
20000
15000
10000
5000
0
UnitPrice
0
500
1000
1500
2000
2500
3000
3500
StockCode

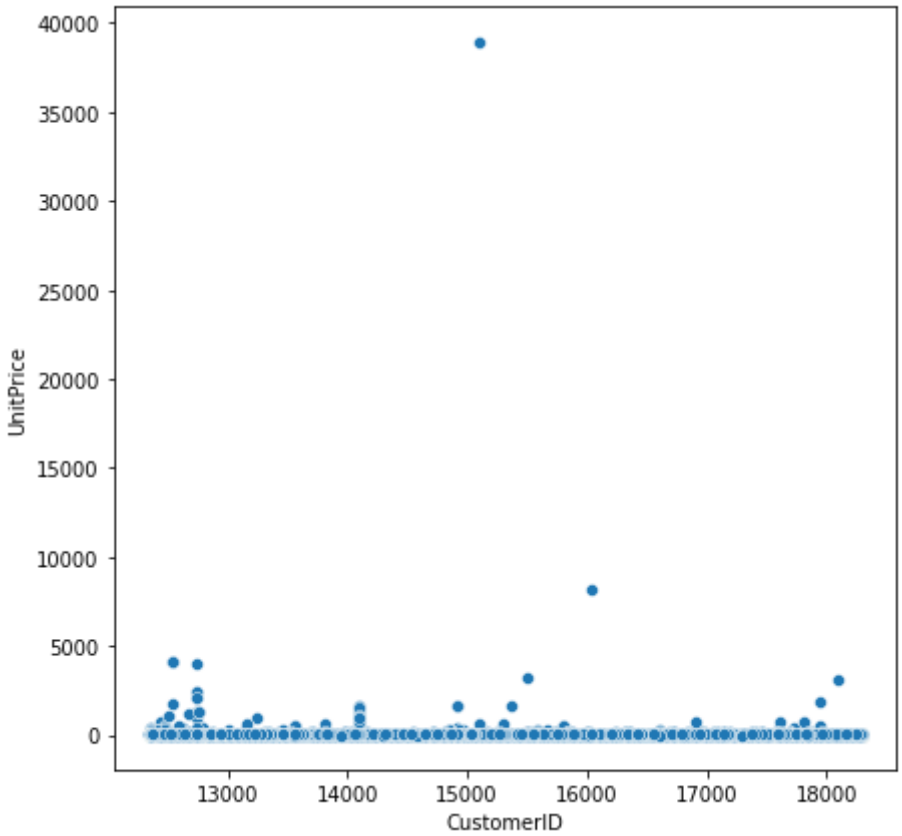

40000
35000
30000
25000

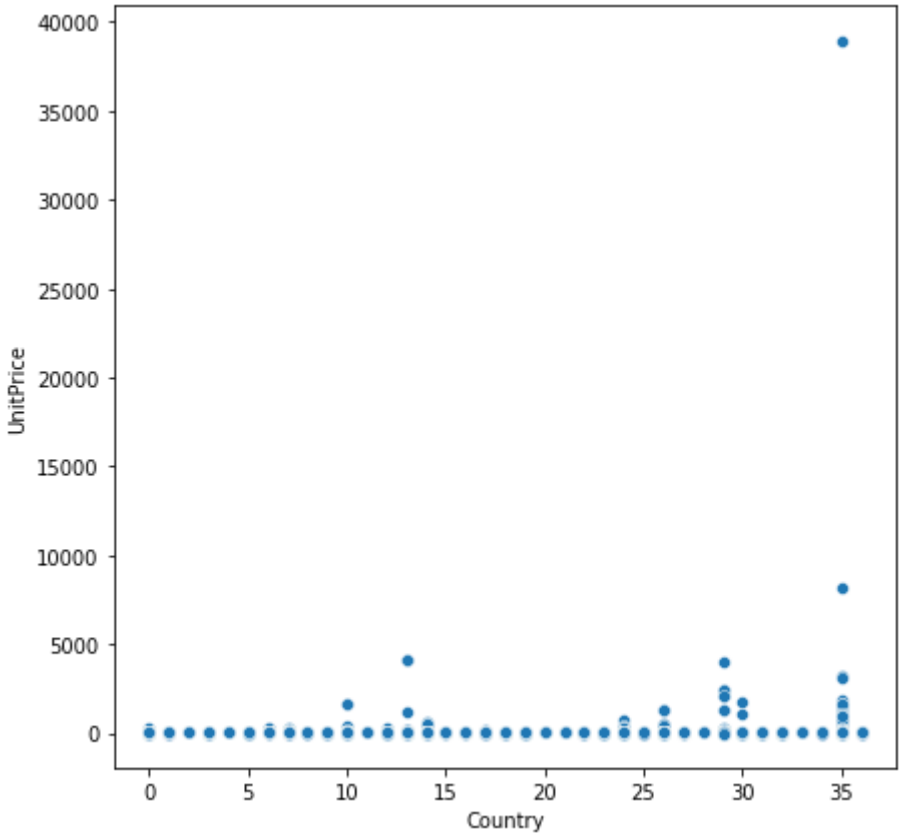

UnitPri
20000
15000
10000
5000
0
0
500
1000
1500
2000
2500
3000
3500
4000
Description


UnitPrice
40000
35000
30000
25000
20000
15000
10000
5000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
Quantity


UnitPrice
40000
35000
30000
25000
20000
15000
10000
5000
0


0
5000
10000
15000
20000
25000
30000
35000
40000
UnitPrice

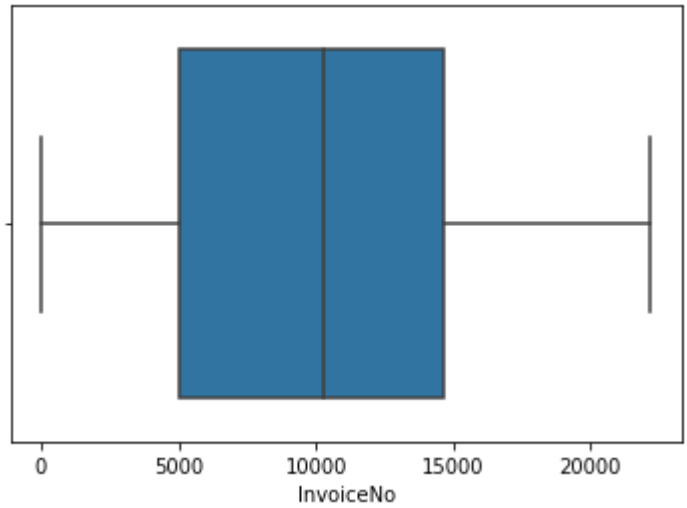

40000
35000
30000
25000
20000
15000
10000
5000
0
UnitPrice
13000
14000
15000
16000
17000
18000
CustomerID

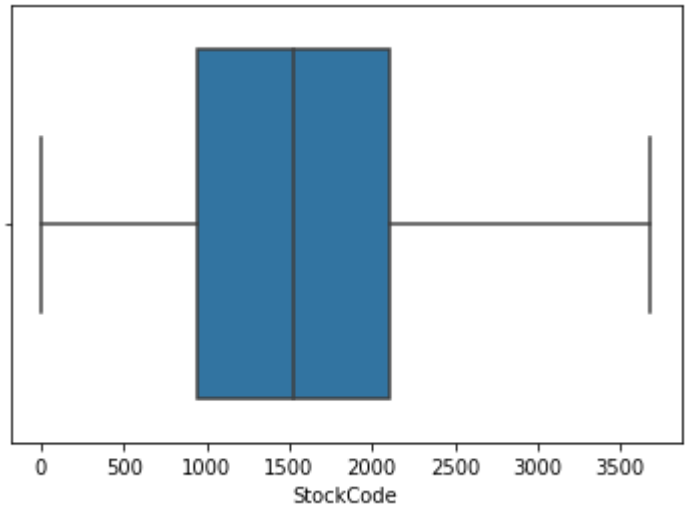

40000
35000
30000
25000
20000
15000
10000
5000
0
UnitPrice
0
5
10
15
20
25
30
35
Country

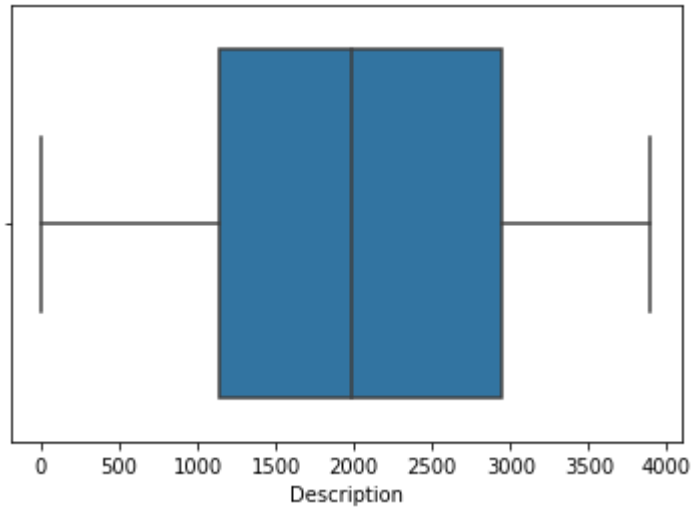

40000
35000
30000

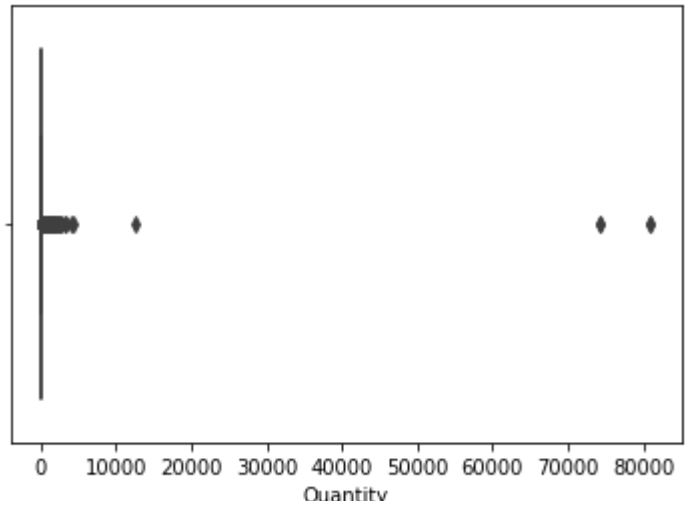
```
for i in df:
```

```
    for i in df:
        sns.boxplot(data=df, x=i)
        plt.show()
```


0
5000
10000
15000
20000
InvoiceNo

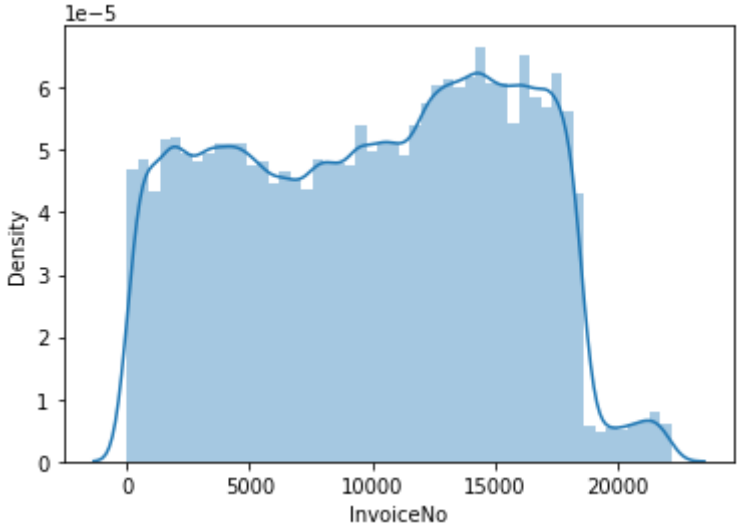

0
500
1000
1500
2000
2500
3000
3500
StockCode


0
500
1000
1500
2000
2500
3000
3500
4000
Description

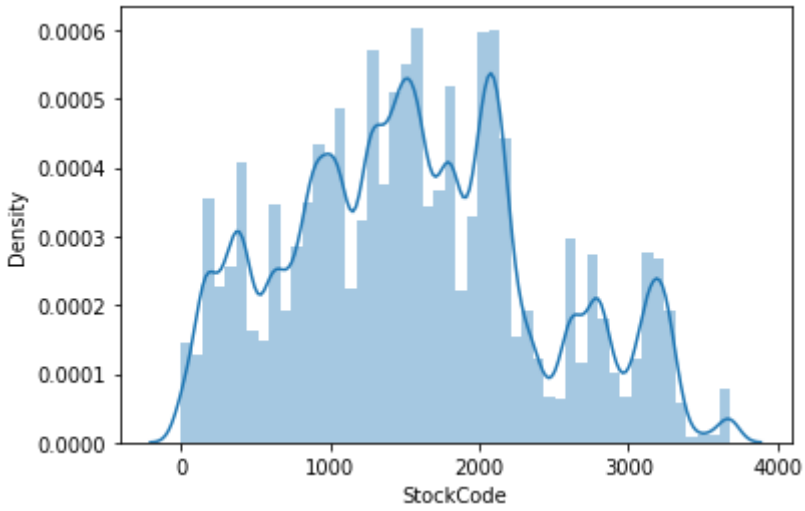

0
10000
20000
30000
40000
50000
60000
70000
80000
Quantity

```
from scipy.stats import skew
for col in df:
    try:
        plt.figure()
        print(col, " : ", skew(df[col]))
        sns.distplot(df[col])
        plt.show()
    except:
        pass
    print("----------------------------------------------------------")
```

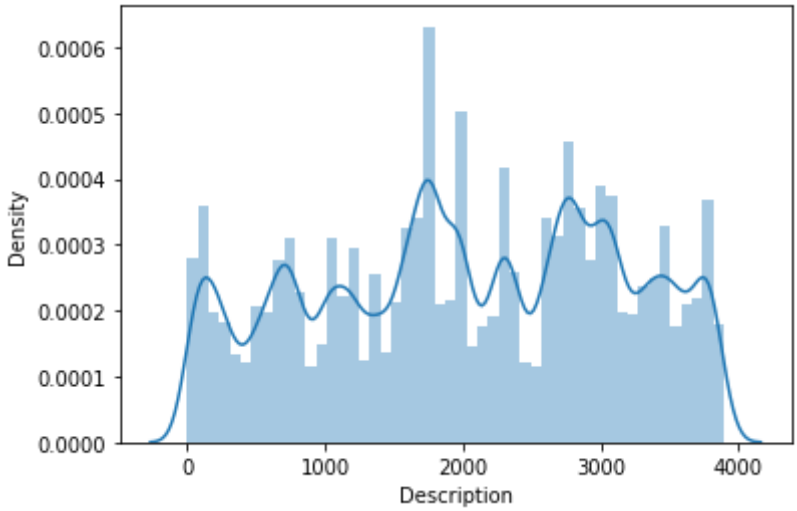
```
InvoiceNo  :  -0.0797039233996088
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```




```
----------------------------------------------------------
StockCode  :  0.27124606831967124
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

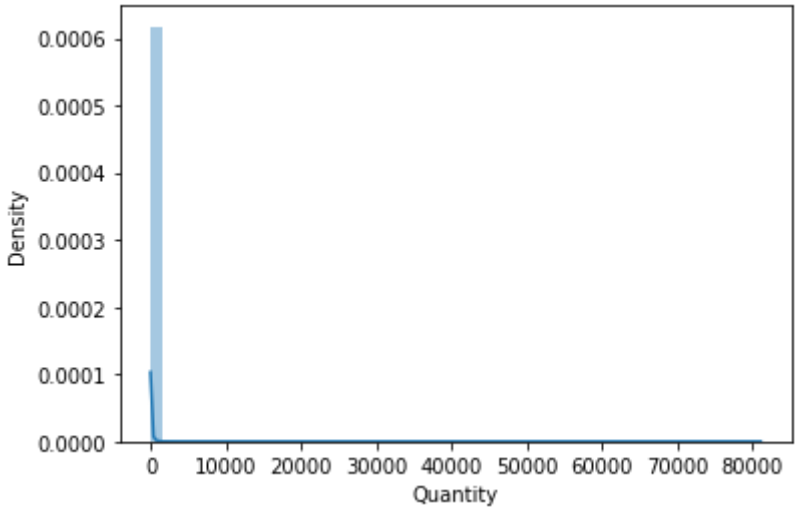



```
----------------------------------------------------------
Description  :  -0.13396626863814126
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

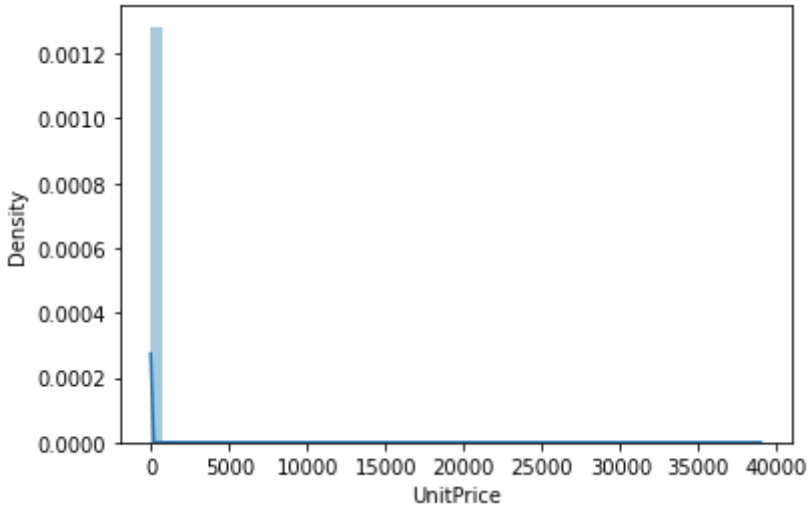



```
----------------------------------------------------------
Quantity  :  257.2867157391663
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
```

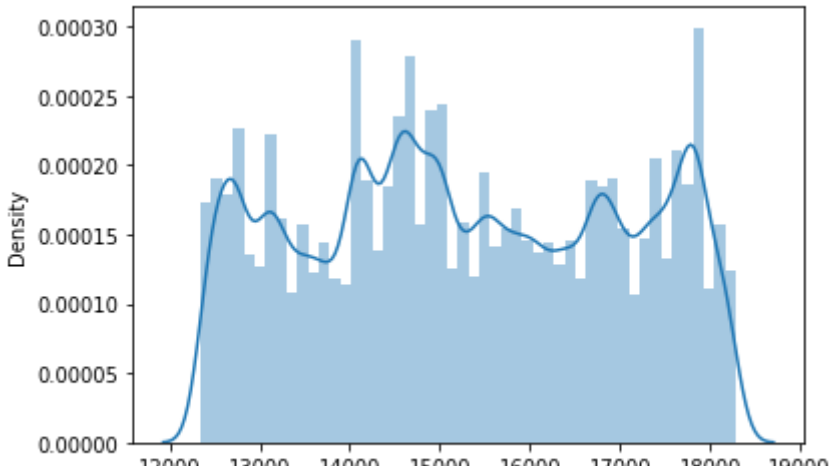
```
  warnings.warn(msg, FutureWarning)
```




```
----------------------------------------------------------
UnitPrice  :  438.3381995247711
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```




```
----------------------------------------------------------
CustomerID  :  0.029524261227648025
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



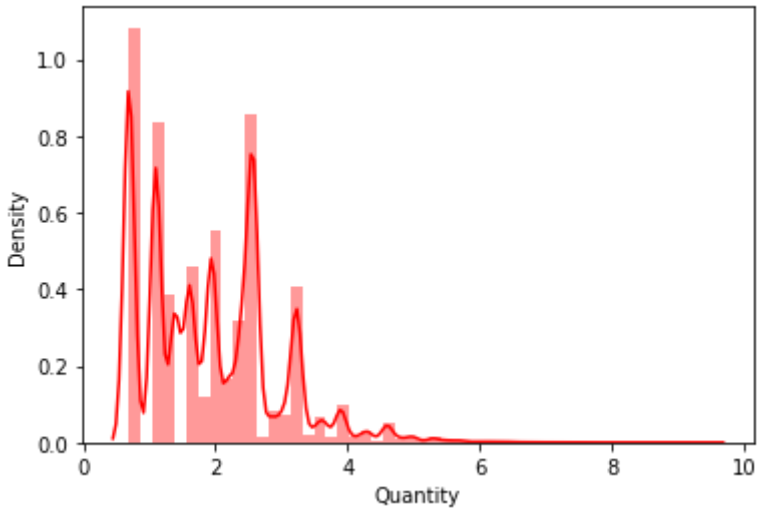

```
df.describe()
```

	InvoiceNo	StockCode	Description	Quantity	UnitPrice	C
count	284780.000000	284780.000000	284780.000000	284780.000000	284780.000000	2847
mean	9955.394083	1573.386807	2023.955573	13.645885	3.451216	152
std	5551.519138	843.604991	1089.812078	295.041223	78.399541	17
min	0.000000	0.000000	0.000000	1.000000	0.000000	123
25%	5069.000000	939.000000	1141.000000	2.000000	1.250000	139
50%	10310.000000	1521.000000	1987.000000	5.000000	1.950000	151

```
df['Quantity'] = np.where(df['Quantity'] > 70000, 10000, df['Quantity'])
```

```
SingleLog_y = np.log1p(df['Quantity'])
sns.distplot(SingleLog_y, color ="r")
print("Skew after 1st Log Transformation: %f" % SingleLog_y.skew())
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
Skew after 1st Log Transformation: 0.835271
```




```
y=df['UnitPrice'].values
x=df.drop(columns=['UnitPrice','CustomerID'])
```

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```
x=ss.fit_transform(x)
x
```

```
array([[-0.68709135,  0.01139539, -1.72503032, -0.15886868,  0.32269957,
        -0.76495649],
       [-0.64962406, -0.32407026,  1.52140665, -0.10971533,  0.32269957,
        -0.76495649],
       [ 1.22986479,  0.71670303, -0.07795447, -0.14248423,  0.32269957,
         0.99159747],
       ...,
       [-0.59702575, -1.07442275,  1.4755271 , -0.14248423,  0.32269957,
```

```
       -0.76495649],
      [ 0.69613607,  0.00309766,  0.97727518,  0.57843148,  0.32269957,
        0.69883848],
      [ 0.19573162,  1.2204942 , -0.79642822, -0.17525312,  0.32269957,
        0.11332049]])
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=None)
```

```
x_train.shape
```

```
(227824, 6)
```

```
import tensorflow as tf
```

```
model=tf.keras.Sequential([tf.keras.layers.Dense(6,input_shape=(6,)),
                           tf.keras.layers.Dense(25,activation='relu'),
                           tf.keras.layers.Dense(20,activation='relu'),
                           tf.keras.layers.Dense(15,activation='relu'),
                           tf.keras.layers.Dense(10,activation='relu'),
                           tf.keras.layers.Dense(5,activation='relu'),

                           tf.keras.layers.Dense(1)
                           ])
```

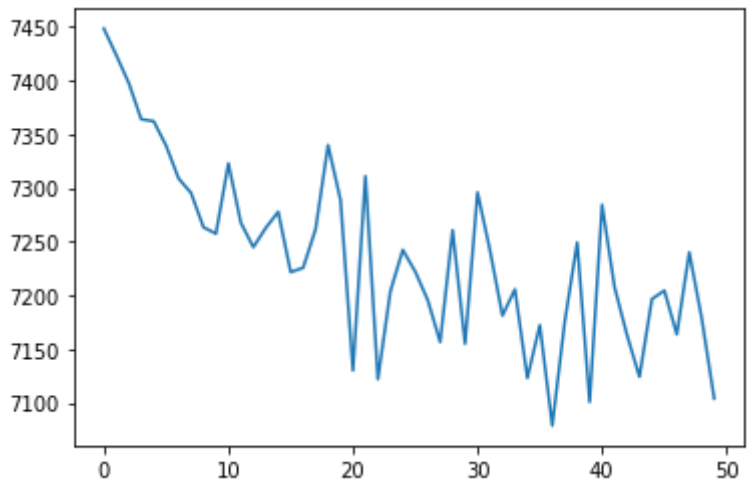
```
model.compile(optimizer='adam',loss='mse')
```

```
trained_model=model.fit(x_train,y_train,epochs=50,batch_size=100)
```

```
Epoch 21/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7130.0732
Epoch 22/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7311.0688
Epoch 23/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7121.9404
Epoch 24/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7203.7271
Epoch 25/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7242.3989
Epoch 26/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7222.1357
Epoch 27/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7195.7476
Epoch 28/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7156.5571
Epoch 29/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7260.7222
Epoch 30/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7155.0020
Epoch 31/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7295.9556
```

```
Epoch 32/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7241.8936
Epoch 33/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7181.2466
Epoch 34/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7205.6118
Epoch 35/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7123.0771
Epoch 36/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7172.2979
Epoch 37/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7078.7549
Epoch 38/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7175.3750
Epoch 39/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7249.2979
Epoch 40/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7100.7598
Epoch 41/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7284.3667
Epoch 42/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7207.7344
Epoch 43/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7163.2085
Epoch 44/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7124.2085
Epoch 45/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7196.5767
Epoch 46/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7204.5015
Epoch 47/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7163.7158
Epoch 48/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7240.1035
Epoch 49/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7178.6968
Epoch 50/50
2279/2279 [==============================] - 3s 1ms/step - loss: 7103.9214
```

```
plt.plot(trained_model.history['loss'])
plt.show()
```




```
y_pred=model.predict(x_test)
```

```
r2_score(y_test,y_pred)
```

```
0.14288778425581206
```

```
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(rmse)
```

```
27.91618954639955
```