Analysis of UK based retail client's transactional data sets

Different Step to perform analytics.

1. Importing the data set in Jupyter Lab Environment.

```
# Importing the data set
df = pd.read_excel('data_set/Online Retail.xlsx')
```

2. Perform EDA
    - Checking the shape of the data set.
    - Checking how many rows and column are there. Here are 541909 no of columns and 8 rows.

```
# Checking the dimension of the dataset
df.shape
```

```
(541909, 8)
```

3. Splitting the column 'Invoice Date' into two column, One is only date and another is only time column.

```
# Spliting the data set
df['new_date'] = [d.date() for d in df['InvoiceDate']]
df['new_time'] = [d.time() for d in df['InvoiceDate']]
```

```
df['year'] = pd. DatetimeIndex(df['InvoiceDate']). year
```

```
df['Month'] = pd. DatetimeIndex(df['InvoiceDate']). month
import calendar
df['Month'] = df['Month'].apply(lambda x: calendar.month_abbr[x])
```

4. Now creating a new column which is sales column using the column unit price and quantity.

```
# Creating a new column sales.
df["Sales"]=df["Quantity"]*df["UnitPrice"]
```

5. Now using the describe function we will get some idea about integer type column.

```
df.describe()
```

	Quantity	UnitPrice	CustomerID	year	Sales
count	541909.000000	541909.000000	406829.000000	541909.000000	541909.000000
mean	9.552250	4.611114	15287.690570	2010.921609	17.987795
std	218.081158	96.759853	1713.600303	0.268787	378.810824
min	-80995.000000	-11062.060000	12346.000000	2010.000000	-168469.600000
25%	1.000000	1.250000	13953.000000	2011.000000	3.400000
50%	3.000000	2.080000	15152.000000	2011.000000	9.750000
75%	10.000000	4.130000	16791.000000	2011.000000	17.400000
max	80995.000000	38970.000000	18287.000000	2011.000000	168469.600000

6. Using info function we will get the about each column i.e they are numerical or categorical. How many null values are present in the data set. For this data set there is no null value.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 13 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  datetime64[ns]
 5   UnitPrice    541909 non-null  float64       
 6   CustomerID   406829 non-null  float64       
 7   Country      541909 non-null  object        
 8   new_date     541909 non-null  object        
 9   new_time     541909 non-null  object        
 10  year         541909 non-null  int64         
 11  Month        541909 non-null  object        
 12  Sales        541909 non-null  float64       
dtypes: datetime64[ns](1), float64(3), int64(2), object(7)
memory usage: 53.7+ MB
```

7. Now using “nunique “ function we will get how many unique value is there in each column.

```
#Checking the no of unique value in the dataset
df.nunique()
```

```
InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate    23260
UnitPrice       1630
CustomerID      4372
Country           38
new_date         305
new_time         774
year               2
Month             12
Sales           6204
dtype: int64
```

8. Now in this step we can see the sales by country wise. How much sales are generated in each country? Here we first group by data and the use the aggregation function on sales.

```
# Country wise sales
df.groupby(["Country"]).agg({"Sales":"sum"}).sort_values(by='Sales', ascending=False,)
```

Country	Sales
United Kingdom	8.187806e+06
Netherlands	2.846615e+05
EIRE	2.632768e+05
Germany	2.216982e+05
France	1.974039e+05
Australia	1.370773e+05
Switzerland	5.638535e+04
Spain	5.477458e+04
Belgium	4.091096e+04
Sweden	3.659591e+04
Japan	3.534062e+04
Norway	3.516346e+04
Portugal	2.936702e+04
Finland	2.232674e+04
Channel Islands	2.008629e+04

9. Then we check country wise how much quantity are sold.

```
# Country wise Quantity
df.groupby(["Country"]).agg({"Quantity":"sum"}).sort_values(by='Quantity', ascending=False,)
```

Country	Quantity
United Kingdom	4263829
Netherlands	200128
EIRE	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

10. Then I find the year wise and month wise sales .

```
# Year wise sales
df.groupby(["year"]).agg({"Sales":"sum"})
```

year	Sales
2010	7.489570e+05
2011	8.998791e+06

```
# Month wise sales
df.groupby(["Month"]).agg({"Sales":"sum"}).sort_values(by='Sales', ascending=False,)
```

Month	Sales
Nov	1461756.250
Dec	1182625.030
Oct	1070704.670
Sep	1019687.622
May	723333.510
Jun	691123.120
Mar	683267.080
Aug	682680.510
Jul	681300.111
Jan	560000.260
Feb	498062.650
Apr	493207.121

11. Then I did some analysis of sales with respect to stock code and description.

```
# Stock code wise quantity
df.groupby(["StockCode"]).agg({"Quantity":"sum"}).sort_values(by='Quantity', ascending=False,)
```

StockCode	Quantity
22197	56450
84077	53847
85099B	47363
85123A	38830
84879	36221
...	...
79323LP	-2618
79323W	-4838
72140F	-5368
23003	-8516
23005	-14418

4070 rows × 1 columns

```
# Description wise sales
df.groupby(["Description"]).agg({"Sales":"sum"}).sort_values(by='Sales', ascending=False,)
```

Description	Sales
DOTCOM POSTAGE	206245.480
REGENCY CAKESTAND 3 TIER	164762.190
WHITE HANGING HEART T-LIGHT HOLDER	99668.470
PARTY BUNTING	98302.980
JUMBO BAG RED RETROSPOT	92356.030
...	...
Bank Charges	-7175.639
CRUK Commission	-7933.430
Adjust bad debt	-11062.060
Manual	-68671.640
AMAZON FEE	-221520.500

4223 rows × 1 columns