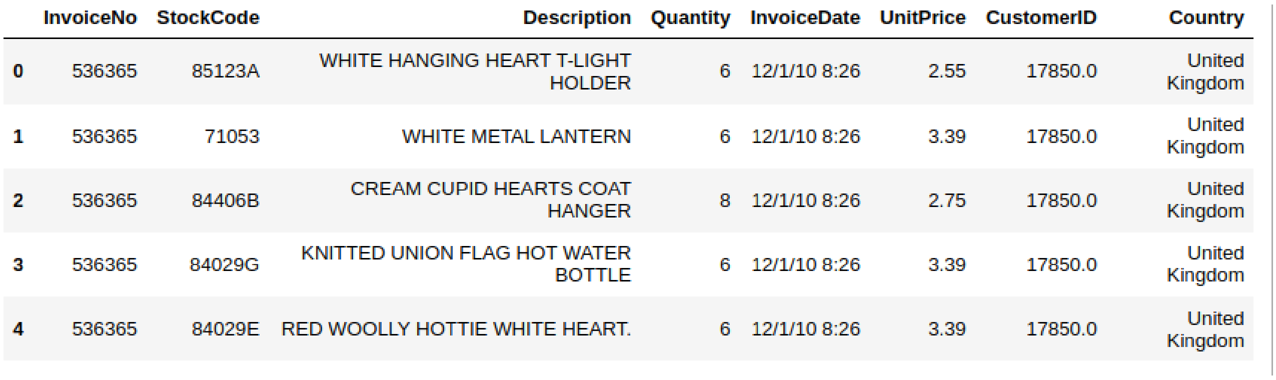
## PART I

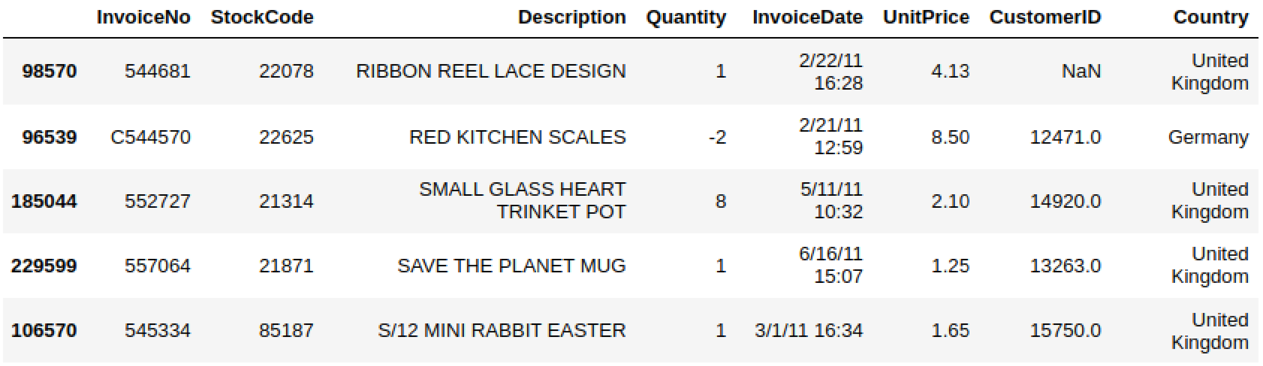
### 1. Import necessary libraries and read the provided dataset(online_sales.csv).

For this project I imported numpy, pandas for data manipulation operations;
matplotlib, seaborn — for visualization

### 2. Check top 5 and random 5 samples of the dataframe.



| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| **0** | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 12/1/10 8:26 | 2.55 | 17850.0 | United Kingdom |
| **1** | 536365 | 71053 | WHITE METAL LANTERN | 6 | 12/1/10 8:26 | 3.39 | 17850.0 | United Kingdom |
| **2** | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 12/1/10 8:26 | 2.75 | 17850.0 | United Kingdom |
| **3** | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 12/1/10 8:26 | 3.39 | 17850.0 | United Kingdom |
| **4** | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 12/1/10 8:26 | 3.39 | 17850.0 | United Kingdom |



| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| **98570** | 544681 | 22078 | RIBBON REEL LACE DESIGN | 1 | 2/22/11 16:28 | 4.13 | NaN | United Kingdom |
| **96539** | C544570 | 22625 | RED KITCHEN SCALES | -2 | 2/21/11 12:59 | 8.50 | 12471.0 | Germany |
| **185044** | 552727 | 21314 | SMALL GLASS HEART TRINKET POT | 8 | 5/11/11 10:32 | 2.10 | 14920.0 | United Kingdom |
| **229599** | 557064 | 21871 | SAVE THE PLANET MUG | 1 | 6/16/11 15:07 | 1.25 | 13263.0 | United Kingdom |
| **106570** | 545334 | 85187 | S/12 MINI RABBIT EASTER | 1 | 3/1/11 16:34 | 1.65 | 15750.0 | United Kingdom |

In the sample dataframe, rows are selected randomly from all dataset

### 3. Check info of the dataframe and write your observations. Comment on datatypes and shape of the dataset.

Dataset has 8 columns and 240,007 rows

The dataset consist float, int, and object data types. However, InvoiceDate

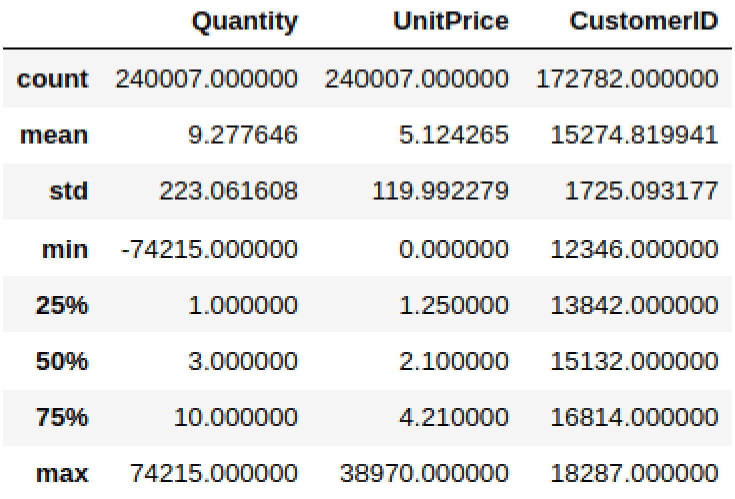
is consider to be datatype.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240007 entries, 0 to 240006
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    240007 non-null  object 
 1   StockCode    240007 non-null  object 
 2   Description  239106 non-null  object 
 3   Quantity     240007 non-null  int64  
 4   InvoiceDate  240007 non-null  object 
 5   UnitPrice    240007 non-null  float64
 6   CustomerID   172782 non-null  float64
 7   Country      240007 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 14.6+ MB
```

The dataset provided is for about 2-year invoice cycle period.

Minimum values of Quantity fields have negative values. This seems to be an wrong invoice.

In dataset were 1970 dublicates, and I have successfuly delete them.



| | Quantity | UnitPrice | CustomerID |
|---|---|---|---|
| **count** | 240007.000000 | 240007.000000 | 172782.000000 |
| **mean** | 9.277646 | 5.124265 | 15274.819941 |
| **std** | 223.061608 | 119.992279 | 1725.093177 |
| **min** | -74215.000000 | 0.000000 | 12346.000000 |
| **25%** | 1.000000 | 1.250000 | 13842.000000 |
| **50%** | 3.000000 | 2.100000 | 15132.000000 |
| **75%** | 10.000000 | 4.210000 | 16814.000000 |
| **max** | 74215.000000 | 38970.000000 | 18287.000000 |

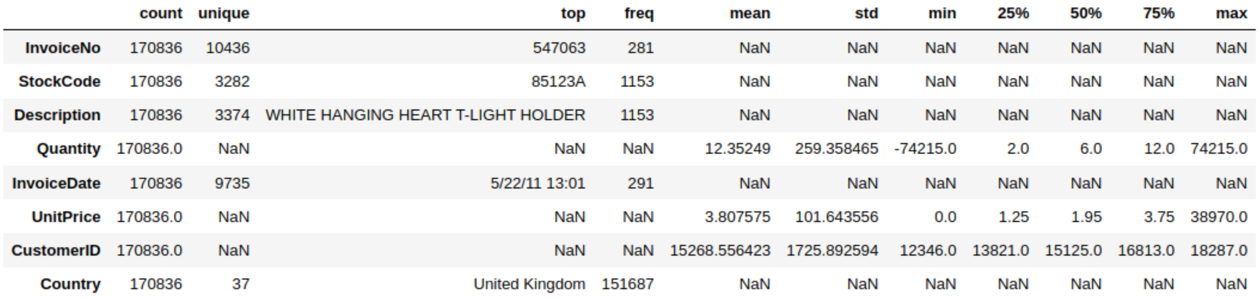
**4. Check for null values and report the percentage ofnull values of each column.And drop the rows having null values in it.**

28, 23 % - CustomerID
0.37 % - Description

There are 28, 23 % of missing values in column CustomerID, and 0, 37 % in column Description. I consider 28% of data is valuable for analysis. Maybe the NaN CustomerID is not registred Customer, so we are analysing only registrud users.

However, the Description data could be filled with values which have the same **StockCode.**

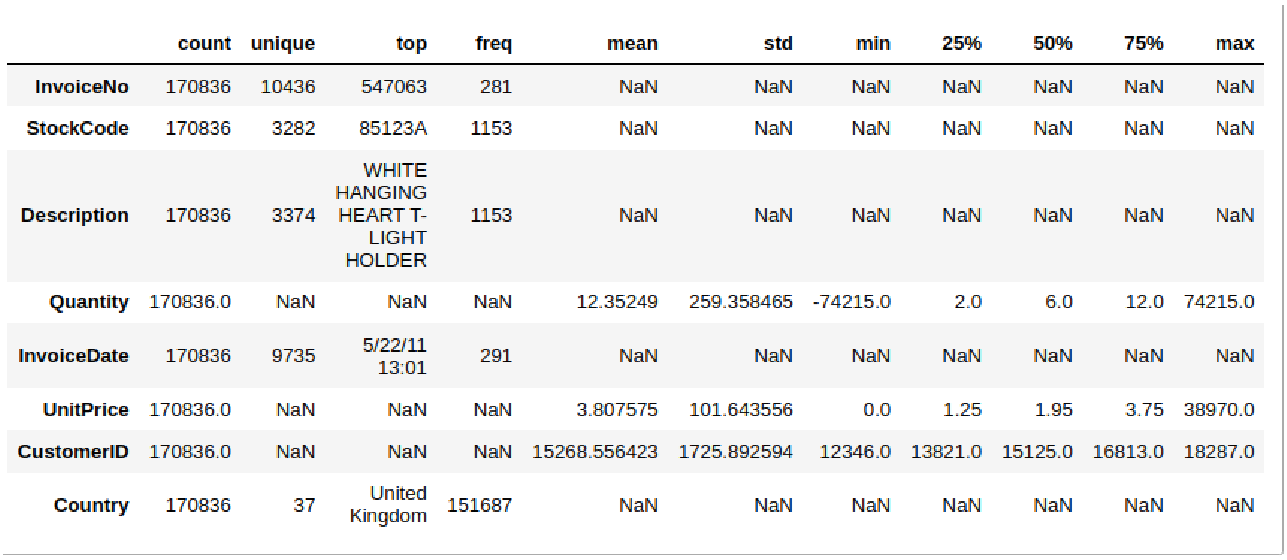
**5. Check statistical summary of the dataset.**



| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **InvoiceNo** | 170836 | 10436 | 547063 | 281 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **StockCode** | 170836 | 3282 | 85123A | 1153 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Description** | 170836 | 3374 | WHITE HANGING HEART T-LIGHT HOLDER | 1153 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Quantity** | 170836.0 | NaN | NaN | NaN | 12.35249 | 259.358465 | -74215.0 | 2.0 | 6.0 | 12.0 | 74215.0 |
| **InvoiceDate** | 170836 | 9735 | 5/22/11 13:01 | 291 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **UnitPrice** | 170836.0 | NaN | NaN | NaN | 3.807575 | 101.643556 | 0.0 | 1.25 | 1.95 | 3.75 | 38970.0 |
| **CustomerID** | 170836.0 | NaN | NaN | NaN | 15268.556423 | 1725.892594 | 12346.0 | 13821.0 | 15125.0 | 16813.0 | 18287.0 |
| **Country** | 170836 | 37 | United Kingdom | 151687 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

On 22th of May 2011 there was the highest number of sales 291.

**6. Drop the instances having quantity less than zero.**



| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **InvoiceNo** | 170836 | 10436 | 547063 | 281 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **StockCode** | 170836 | 3282 | 85123A | 1153 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Description** | 170836 | 3374 | WHITE HANGING HEART T-LIGHT HOLDER | 1153 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Quantity** | 170836.0 | NaN | NaN | NaN | 12.35249 | 259.358465 | -74215.0 | 2.0 | 6.0 | 12.0 | 74215.0 |
| **InvoiceDate** | 170836 | 9735 | 5/22/11 13:01 | 291 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **UnitPrice** | 170836.0 | NaN | NaN | NaN | 3.807575 | 101.643556 | 0.0 | 1.25 | 1.95 | 3.75 | 38970.0 |
| **CustomerID** | 170836.0 | NaN | NaN | NaN | 15268.556423 | 1725.892594 | 12346.0 | 13821.0 | 15125.0 | 16813.0 | 18287.0 |
| **Country** | 170836 | 37 | United Kingdom | 151687 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

There were 4090 instances with Quantity less than zero and they were successfully deleted

**7. Check unique values of the country and report then ame of the country that hast he highest number of instances.**

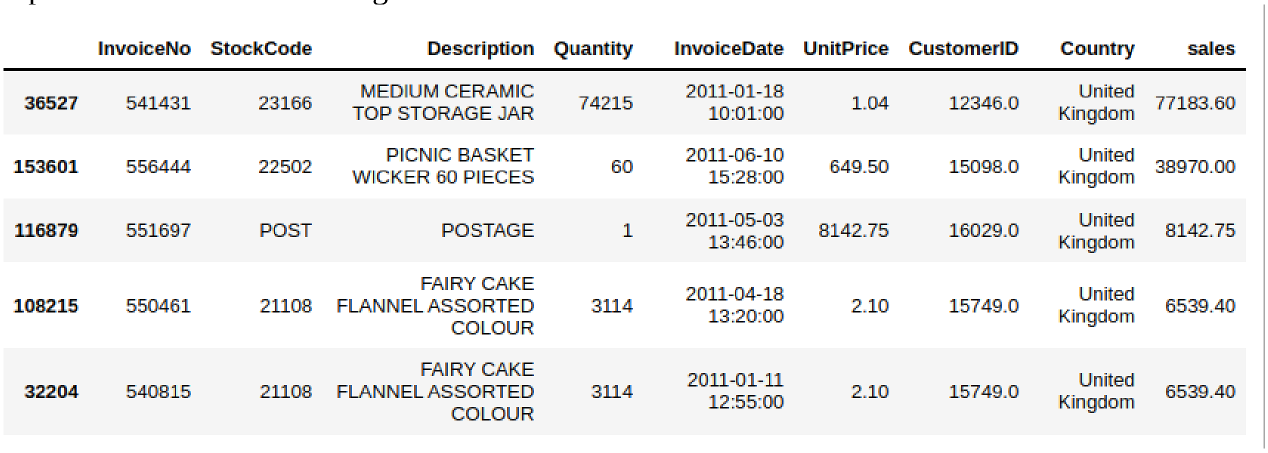
Unique values of the ‘Country’ :

United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Finland',
'Austria', 'Greece', 'Singapore', 'Lebanon',
'United Arab Emirates', 'Israel', 'Saudi Arabia', 'Czech Republic',
'Canada', 'Unspecified', 'Brazil', 'USA', 'European Community', 'Bahrain', 'Malta', 'Unit'

United Kingdom - 148 130

**8. Create a new column with the name as ‘sales’ havingtotal sales. The total salesis defined as Quantity*UnitPrice.**

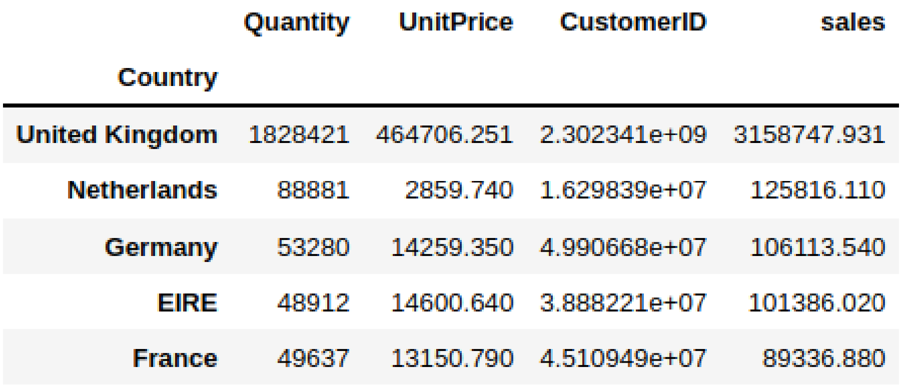
Top 5 values of df with the highest sales



| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country | sales |
|---|---|---|---|---|---|---|---|---|---|
| **36527** | 541431 | 23166 | MEDIUM CERAMIC TOP STORAGE JAR | 74215 | 2011-01-18 10:01:00 | 1.04 | 12346.0 | United Kingdom | 77183.60 |
| **153601** | 556444 | 22502 | PICNIC BASKET WICKER 60 PIECES | 60 | 2011-06-10 15:28:00 | 649.50 | 15098.0 | United Kingdom | 38970.00 |
| **116879** | 551697 | POST | POSTAGE | 1 | 2011-05-03 13:46:00 | 8142.75 | 16029.0 | United Kingdom | 8142.75 |
| **108215** | 550461 | 21108 | FAIRY CAKE FLANNEL ASSORTED COLOUR | 3114 | 2011-04-18 13:20:00 | 2.10 | 15749.0 | United Kingdom | 6539.40 |
| **32204** | 540815 | 21108 | FAIRY CAKE FLANNEL ASSORTED COLOUR | 3114 | 2011-01-11 12:55:00 | 2.10 | 15749.0 | United Kingdom | 6539.40 |

**9. Report the top 5 countries in terms of sales.**

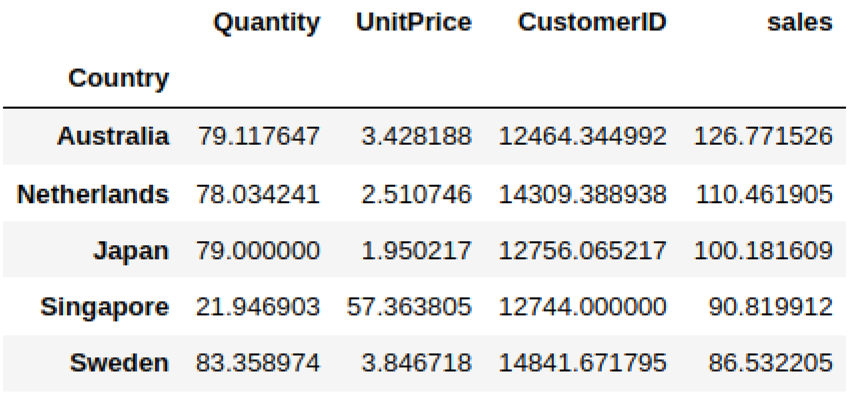
**9.1 Consider the size of sales.**
Top 5 countries in terms of size of sales



| | Quantity | UnitPrice | CustomerID | sales |
|---|---|---|---|---|
| **Country** | | | | |
| **United Kingdom** | 1828421 | 464706.251 | 2.302341e+09 | 3158747.931 |
| **Netherlands** | 88881 | 2859.740 | 1.629839e+07 | 125816.110 |
| **Germany** | 53280 | 14259.350 | 4.990668e+07 | 106113.540 |
| **EIRE** | 48912 | 14600.640 | 3.888221e+07 | 101386.020 |
| **France** | 49637 | 13150.790 | 4.510949e+07 | 89336.880 |

**9.2 Consider the mean value of sales.**

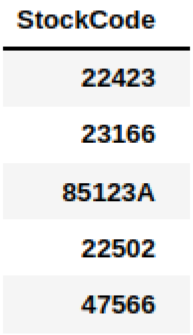
Top 5 countries in terms of mean of sales



| | Quantity | UnitPrice | CustomerID | sales |
|---|---|---|---|---|
| **Country** | | | | |
| **Australia** | 79.117647 | 3.428188 | 12464.344992 | 126.771526 |
| **Netherlands** | 78.034241 | 2.510746 | 14309.388938 | 110.461905 |
| **Japan** | 79.000000 | 1.950217 | 12756.065217 | 100.181609 |
| **Singapore** | 21.946903 | 57.363805 | 12744.000000 | 90.819912 |
| **Sweden** | 83.358974 | 3.846718 | 14841.671795 | 86.532205 |

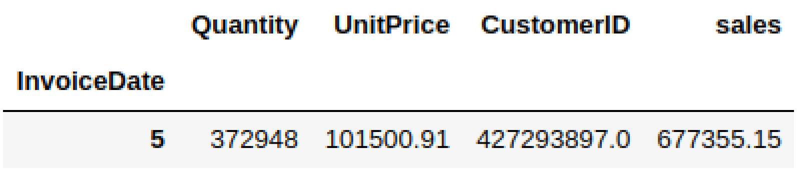
UK has the highest sum of sales, but it is not in top five with mean of sales.
According to number of sales, UK has about 1.8 milion and Australia 79.

**10. Report the top 5 products which bring the highestsales. Use StockCode forproduct information.**



| StockCode |
|---|
| **22423** |
| **23166** |
| **85123A** |
| **22502** |
| **47566** |

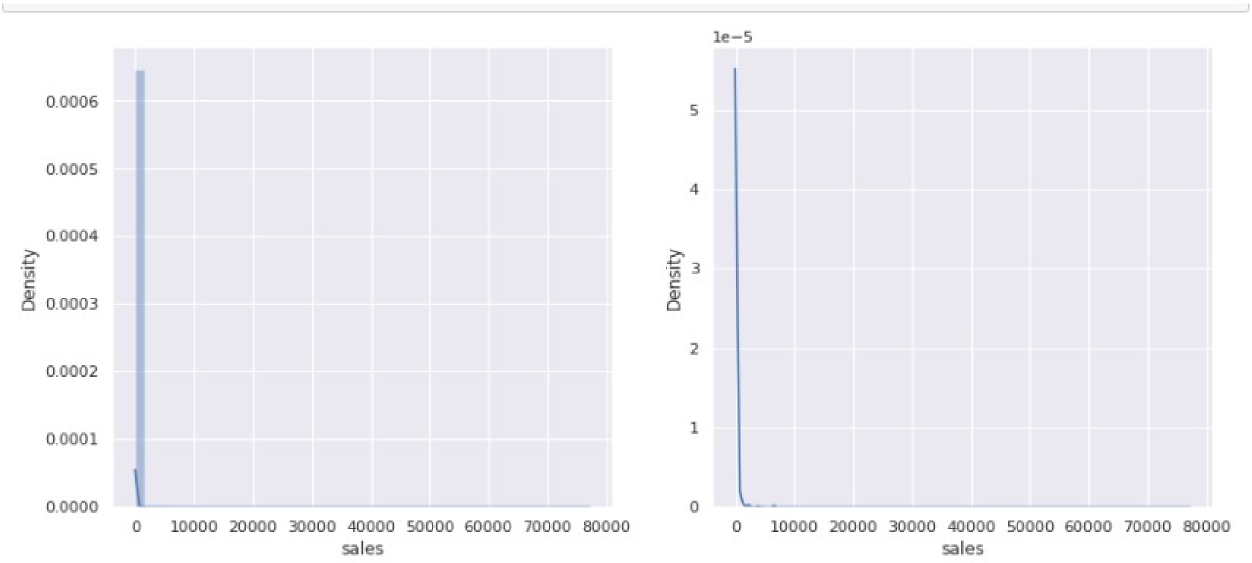
**11. Convert the ‘InvoiceDate’ into a date format and reportthe month on whichthe maximum sales occur?**

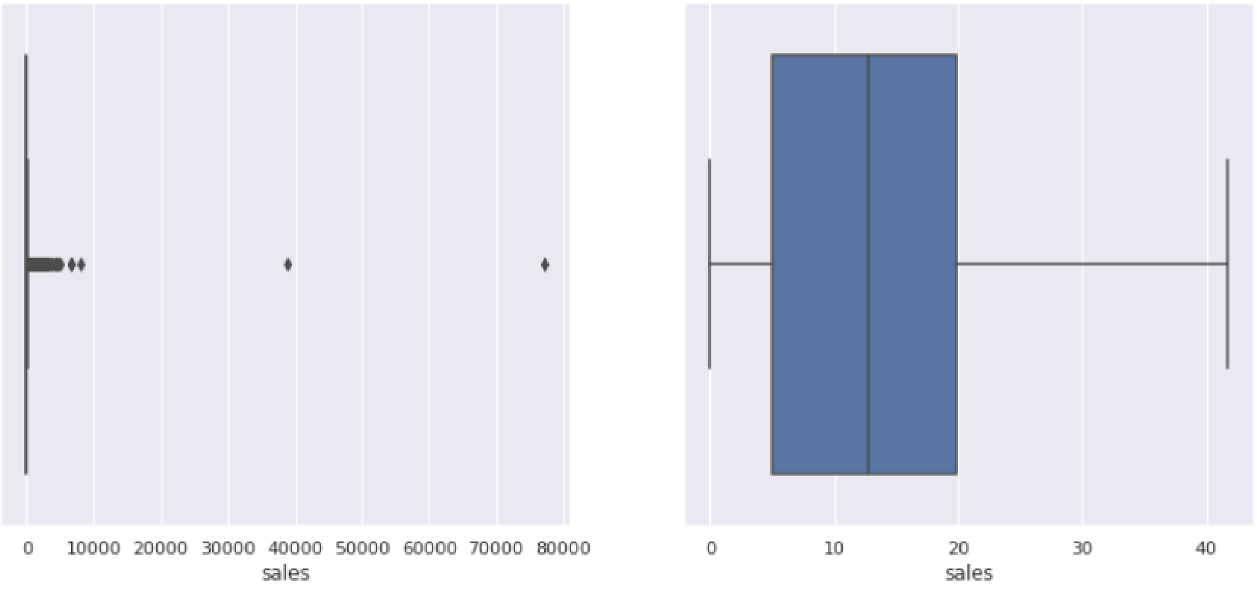


| | Quantity | UnitPrice | CustomerID | sales |
|---|---|---|---|---|
| InvoiceDate | | | | |
| 5 | 372948 | 101500.91 | 427293897.0 | 677355.15 |

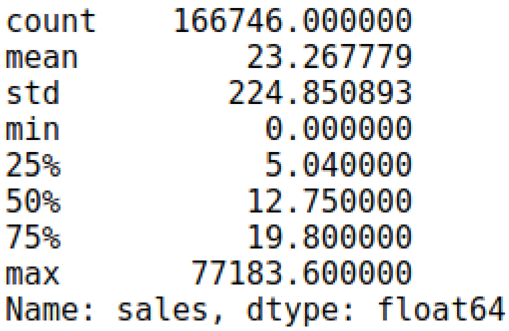
In May there was 677 355 total sales and also the highest quantity of sales.

**12. Check statistical summary of the sales and use anappropriate plot to displaythe distribution of sales and write your inferences.**





‘sales’ has ouliers in upper values.



```
count    166746.000000
mean         23.267779
std         224.850893
min           0.000000
25%           5.040000
50%          12.750000
75%          19.800000
max       77183.600000
Name: sales, dtype: float64
```

sales ranges from a minimum of 0 to maximum of 77 183

75% of the sales have less than or equal to 19, 8  of sales

Mean sale of customers is 23.26 which is higher than the median value indicating that the ditribution is right tailed

**13. Submit a business report including your findings andinterpretations of theabove project. Please refer to the do’s and don’tdocument for moreinformation.**

I found features that had missing values, bad data which had to be cleaned. I also found negative quantity and dublicates in the data based on the business context which had to be deleted.

The dataset provided is for about 2-year invoice cycle period.

Minimum values of Quantity fields have negative values.

In dataset were 1970 dublicates, and I have successfuly delete them.

There are 28, 23 % of missing values in column CustomerID, and 0, 37 %  in column Description. I consider 28% of data is valuable for analysis. Maybe the NaN  CustomerID is not registred Customer, so we are analysing only registrud users.

However, the Description data could be filled with values which have the same  StockCode.

There were found  4090 instances with Quantity less than zero and they were successfully deleted

UK has the highest sum of sales, but it is not in top five with mean of sales. According to number of sales, UK has about 1.8 milion and Australia 79.

In May there was 677 355 total sales and also the highest quantity of sales.

sales ranges from a minimum of 0 to maximum of 77 183

75% of the sales have less than or equal to 19, 8  of sales

Mean sale of customers is 23.26 which is higher than the median value indicating that the ditribution is right tailed

Several statistical measurements and distributions corresponding to categorical and numeric features. This can be useful is choosing an apporpriate technique to build the classification model which is the next step.

Frequency and distribution of the features. This will help us validate the assumptions that are made before implementing a technique.

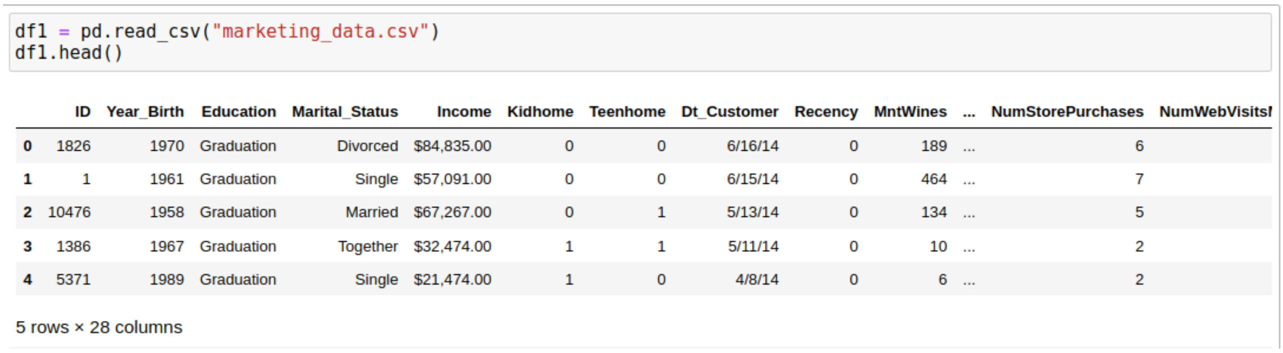
## PART II

### 1. Import necessary libraries.

For this project I imported numpy, pandas for data manipulation operations;
matplotlib, seaborn — for visualization,  KNNImputer for fillin NaN data,
StandardScaler — for scaling 'Income' column

### 2. Load the file and display the first 5 and last5 instances.

Fist 5 rows

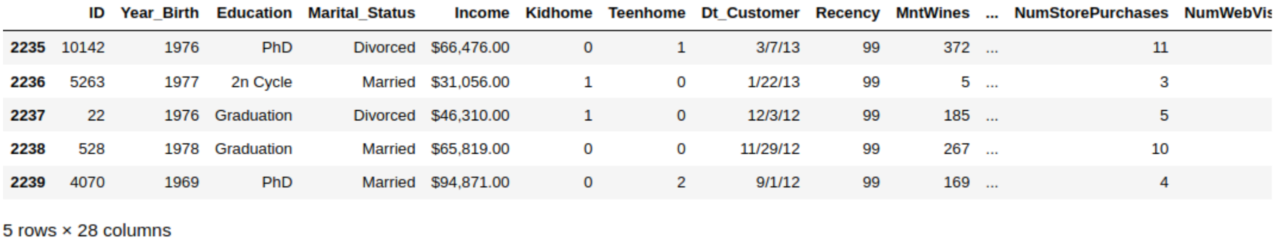


```
df1 = pd.read_csv("marketing_data.csv")
df1.head()
```

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Recency | MntWines | ... | NumStorePurchases | NumWebVisits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | $84,835.00 | 0 | 0 | 6/16/14 | 0 | 189 | ... | 6 | |
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/14 | 0 | 464 | ... | 7 | |
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/14 | 0 | 134 | ... | 5 | |
| **3** | 1386 | 1967 | Graduation | Together | $32,474.00 | 1 | 1 | 5/11/14 | 0 | 10 | ... | 2 | |
| **4** | 5371 | 1989 | Graduation | Single | $21,474.00 | 1 | 0 | 4/8/14 | 0 | 6 | ... | 2 | |

5 rows × 28 columns

Last 5 rows



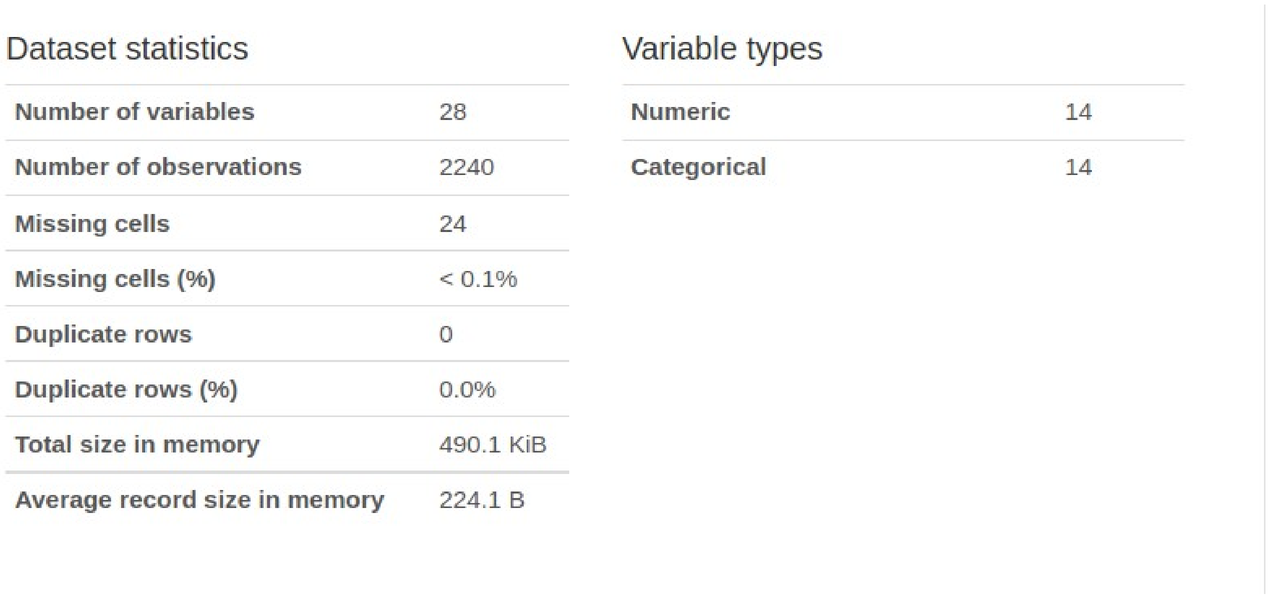
| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Recency | MntWines | ... | NumStorePurchases | NumWebVis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2235** | 10142 | 1976 | PhD | Divorced | $66,476.00 | 0 | 1 | 3/7/13 | 99 | 372 | ... | 11 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | $31,056.00 | 1 | 0 | 1/22/13 | 99 | 5 | ... | 3 | |
| **2237** | 22 | 1976 | Graduation | Divorced | $46,310.00 | 1 | 0 | 12/3/12 | 99 | 185 | ... | 5 | |
| **2238** | 528 | 1978 | Graduation | Married | $65,819.00 | 0 | 0 | 11/29/12 | 99 | 267 | ... | 10 | |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/12 | 99 | 169 | ... | 4 | |

5 rows × 28 columns

### 3. Check the shape of the data (number of rows and column).

The dataset has 2240 rows and 28 columns

### 4. Generate pandas profiling report of the original data.

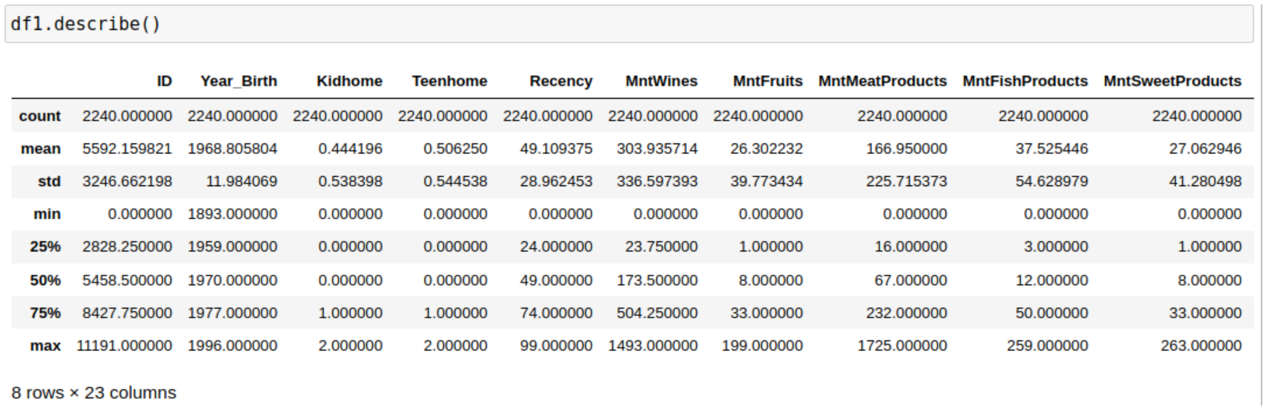


Dataset statistics

| | |
|---|---|
| Number of variables | 28 |
| Number of observations | 2240 |
| Missing cells | 24 |
| Missing cells (%) | < 0.1% |
| Duplicate rows | 0 |
| Duplicate rows (%) | 0.0% |
| Total size in memory | 490.1 KiB |
| Average record size in memory | 224.1 B |

Variable types

| | |
|---|---|
| Numeric | 14 |
| Categorical | 14 |



```
df1.describe()
```

| | ID | Year_Birth | Kidhome | Teenhome | Recency | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 |
| mean | 5592.159821 | 1968.805804 | 0.444196 | 0.506250 | 49.109375 | 303.935714 | 26.302232 | 166.950000 | 37.525446 | 27.062946 |
| std | 3246.662198 | 11.984069 | 0.538398 | 0.544538 | 28.962453 | 336.597393 | 39.773434 | 225.715373 | 54.628979 | 41.280498 |
| min | 0.000000 | 1893.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 2828.250000 | 1959.000000 | 0.000000 | 0.000000 | 24.000000 | 23.750000 | 1.000000 | 16.000000 | 3.000000 | 1.000000 |
| 50% | 5458.500000 | 1970.000000 | 0.000000 | 0.000000 | 49.000000 | 173.500000 | 8.000000 | 67.000000 | 12.000000 | 8.000000 |
| 75% | 8427.750000 | 1977.000000 | 1.000000 | 1.000000 | 74.000000 | 504.250000 | 33.000000 | 232.000000 | 50.000000 | 33.000000 |
| max | 11191.000000 | 1996.000000 | 2.000000 | 2.000000 | 99.000000 | 1493.000000 | 199.000000 | 1725.000000 | 259.000000 | 263.000000 |

8 rows × 23 columns

Minimum Year_Birth is 1893, that needs to be checks. We don`t know exactly from which period of time data was collected or it might be input error.

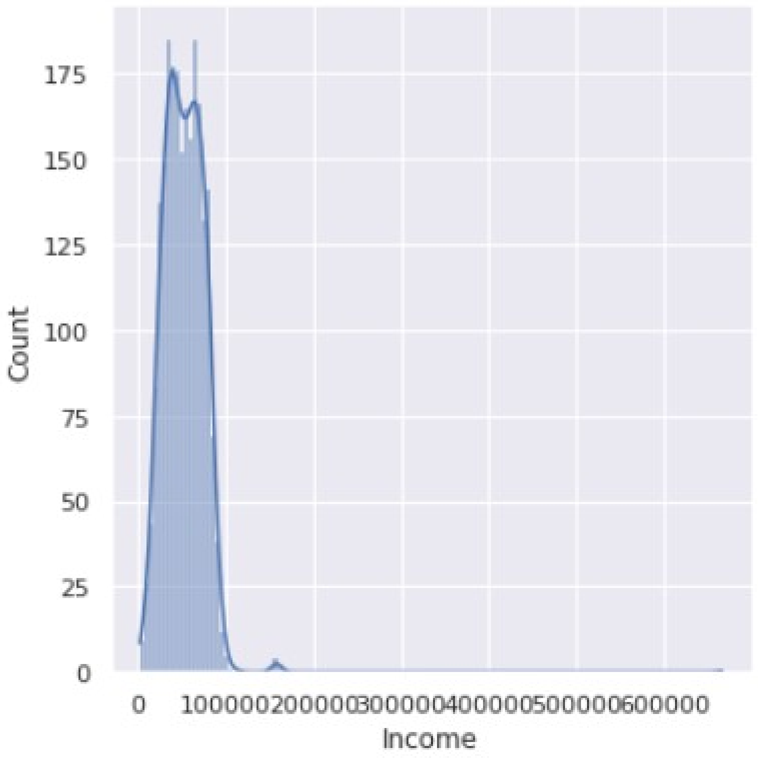
**5. Check the dtype of values in column 'Income'.**

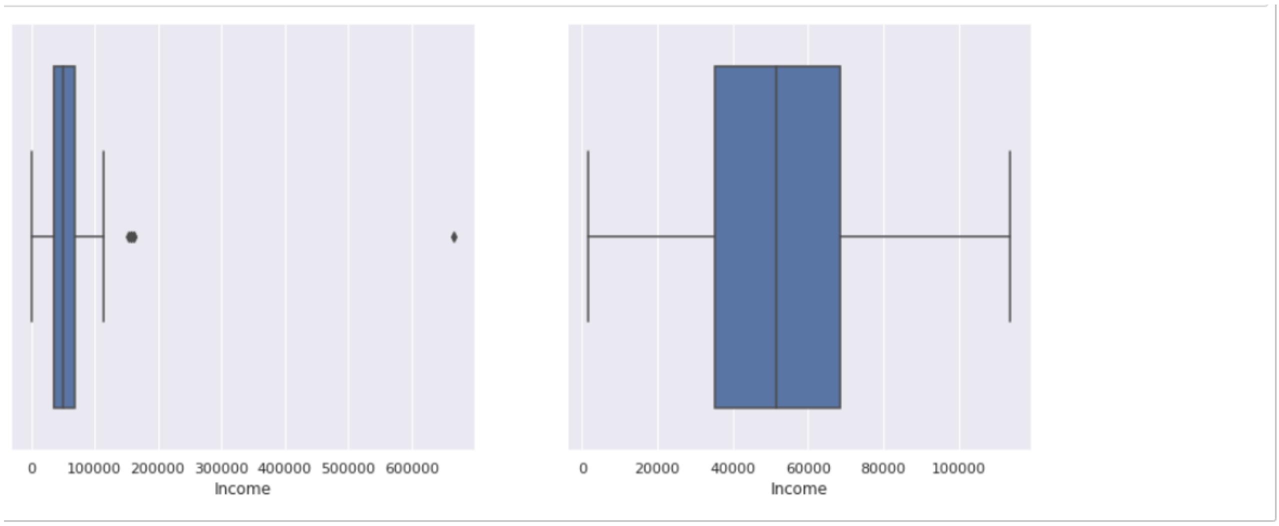
The dtype of values in column 'Income' is ‘object’. For further analysis it should be numeric

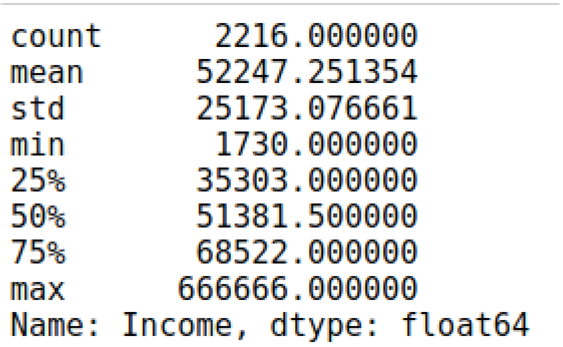
**6. Convert the values in the 'Income' column to numeric format.**

I have changed column name ' Income ' to 'Income'. Than I’ve deleted all $ and comas in string
# and convert it to "float"

**7. Check the distribution of the income column.**







```
count      2216.000000
mean      52247.251354
std       25173.076661
min        1730.000000
25%       35303.000000
50%       51381.500000
75%       68522.000000
max      666666.000000
Name: Income, dtype: float64
```

Income ranges from a minimum of 1730 to maximum of 666 666

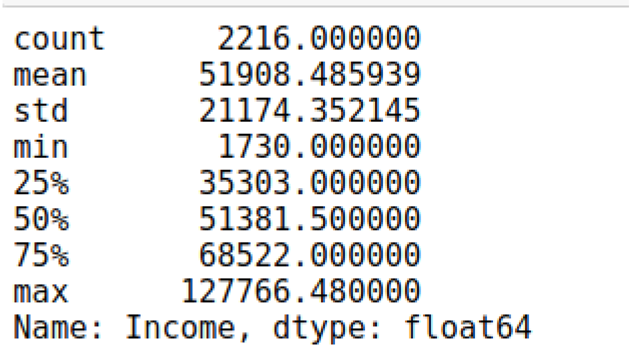
75% of the sales have less than or equal to 68522

Mean sale of customers is 52247 which is higher than the median value (51381) indicating that the ditribution is right tailed

**8. Check the presence of outliers in the feature 'Income'.**

For checking the outliers it was used Z-Score.
It was foud 8 outliers, having z-score greater than 3'

The value for Income if ZScore has to be 3 is equal to 127766.48



```
count      2216.000000
mean      51908.485939
std       21174.352145
min        1730.000000
25%       35303.000000
50%       51381.500000
75%       68522.000000
max      127766.480000
Name: Income, dtype: float64
```

After changing the value of outliers to 127766.48 the difference between 75% of data and the max value is about 60 000.
I consider we should continue treating outliers.

**9. Encode categorical features to numerical.**

**9.1 Convert the column'Education'from categorical tonumerical format.Map them as Basic=1, Graduation=2, Master=3, PhD=4,2n Cycle=5**

To convert ‘Education’ column I have used replace func

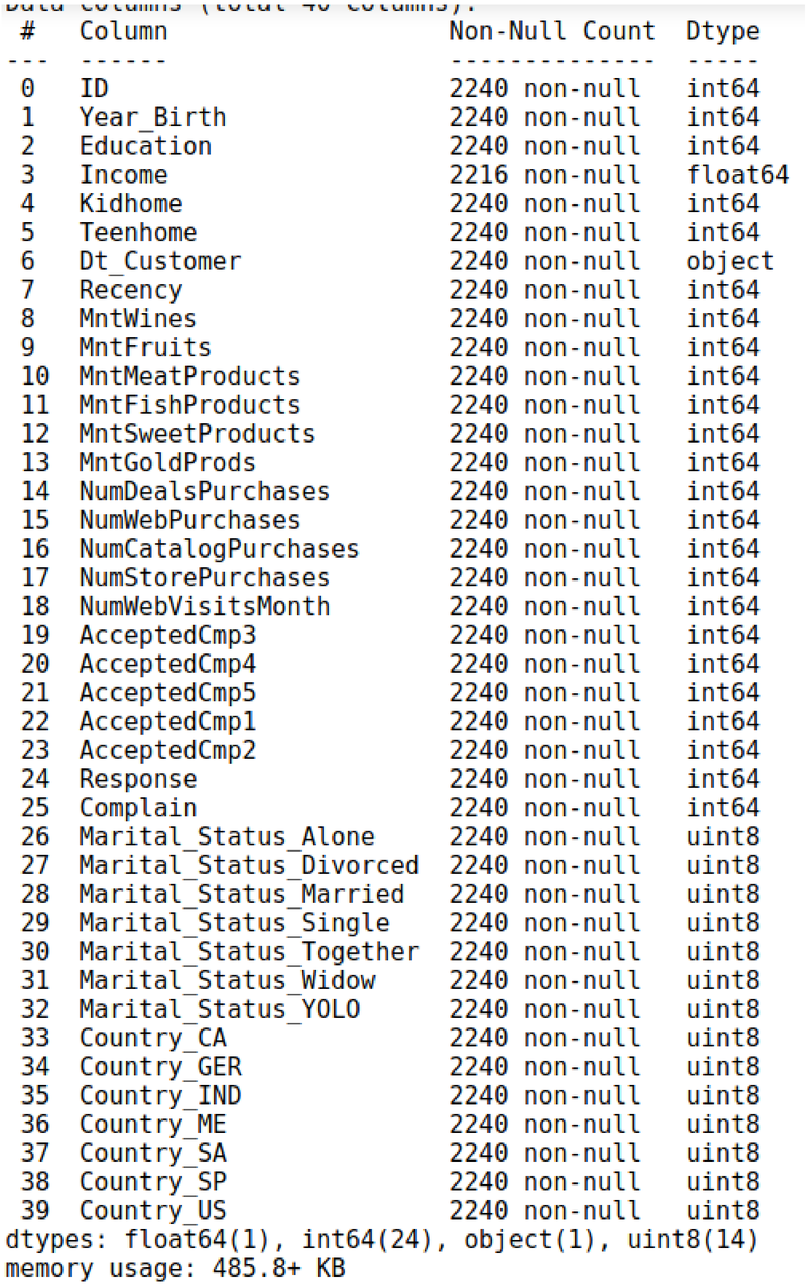
**9.2 Check the number of unique values in the column "Country"**

There are 8 unique countries.

SP appears in dataset 1095 times,
SA — 337,
CA — 268,
AUS — 160 ,
IND — 148,
GER — 120,
US — 109,
ME — 3.

**9.3 So we will one-hot encode these variables.**

Since the column Country and Marital Status is Nominal



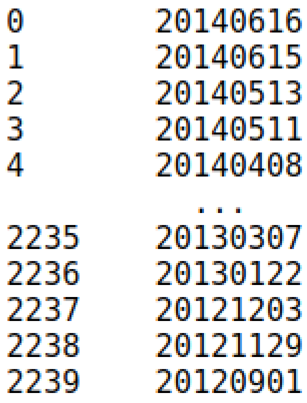
```
Data columns (total 40 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   ID                       2240 non-null   int64  
 1   Year_Birth               2240 non-null   int64  
 2   Education                2240 non-null   int64  
 3   Income                   2216 non-null   float64
 4   Kidhome                  2240 non-null   int64  
 5   Teenhome                 2240 non-null   int64  
 6   Dt_Customer              2240 non-null   object 
 7   Recency                  2240 non-null   int64  
 8   MntWines                 2240 non-null   int64  
 9   MntFruits                2240 non-null   int64  
 10  MntMeatProducts          2240 non-null   int64  
 11  MntFishProducts          2240 non-null   int64  
 12  MntSweetProducts         2240 non-null   int64  
 13  MntGoldProds             2240 non-null   int64  
 14  NumDealsPurchases        2240 non-null   int64  
 15  NumWebPurchases          2240 non-null   int64  
 16  NumCatalogPurchases      2240 non-null   int64  
 17  NumStorePurchases        2240 non-null   int64  
 18  NumWebVisitsMonth        2240 non-null   int64  
 19  AcceptedCmp3             2240 non-null   int64  
 20  AcceptedCmp4             2240 non-null   int64  
 21  AcceptedCmp5             2240 non-null   int64  
 22  AcceptedCmp1             2240 non-null   int64  
 23  AcceptedCmp2             2240 non-null   int64  
 24  Response                 2240 non-null   int64  
 25  Complain                 2240 non-null   int64  
 26  Marital_Status_Alone     2240 non-null   uint8  
 27  Marital_Status_Divorced  2240 non-null   uint8  
 28  Marital_Status_Married   2240 non-null   uint8  
 29  Marital_Status_Single    2240 non-null   uint8  
 30  Marital_Status_Together  2240 non-null   uint8  
 31  Marital_Status_Widow     2240 non-null   uint8  
 32  Marital_Status_YOLO      2240 non-null   uint8  
 33  Country_CA               2240 non-null   uint8  
 34  Country_GER              2240 non-null   uint8  
 35  Country_IND              2240 non-null   uint8  
 36  Country_ME               2240 non-null   uint8  
 37  Country_SA               2240 non-null   uint8  
 38  Country_SP               2240 non-null   uint8  
 39  Country_US               2240 non-null   uint8  
dtypes: float64(1), int64(24), object(1), uint8(14)
memory usage: 485.8+ KB
```

For  One-Hot Encoding it was used pd.get_dummies()

After One-Hot Encoding there are 39 column is dataset

**10. Convert the values in column 'Dt_Customer' to datetime.**

**10.1 After converting the values to datetime, convert it to numerical values**



```
0       20140616
1       20140615
2       20140513
3       20140511
4       20140408
          ...
2235    20130307
2236    20130122
2237    20121203
2238    20121129
2239    20120901
```

After applying all necessary function to convert to datetime the sample of column ‘Dt_Customer' is int 20140616 (year+month+day)

**11. Check the number of null values present in eachcolumn**

Only ‘Income’ column has null values
There is 24 values to be filled.

**12. Handle null values using the below given approaches.**

**12.1 1st Approach: Since the number of instances havingnull values is too less, we candrop the null instances. And drop the null instancesand save it in a new DataFramedf2**

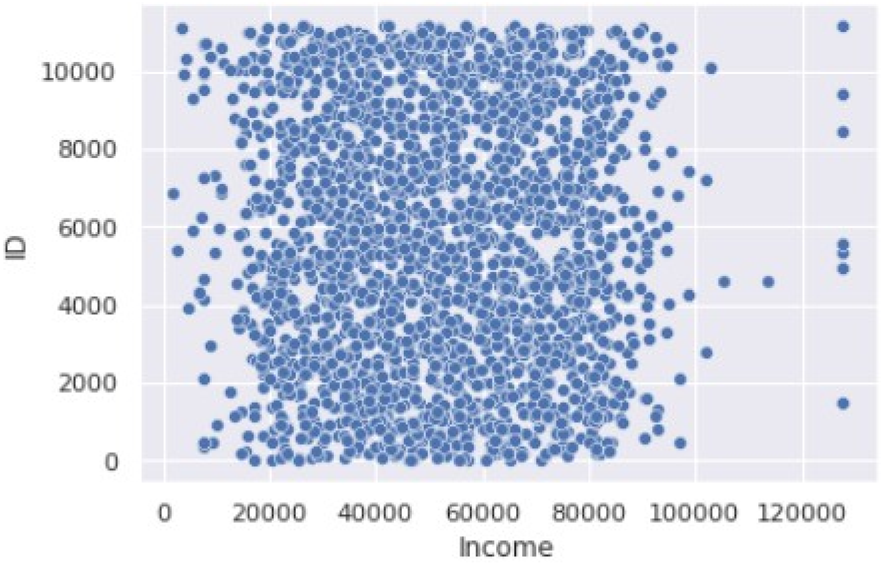
**12.2 2nd Approach: Fill the null instances with median value and save it in new data frame df3 We are not using mean as the column contains someextreme values**

**12.3 3rd Approach: Use sklearn's KNNImputer to impute thedata, and save it indataframe df4**

In my opinion  the best way to treat missing values of column ‘Income’  is using KNNImputer, because it finds the rows in df with simmular features and implement their values to missing ones.

Fill NaN with median values is also a good option.

**13. Visualize the outliers using a scatter plot.**



There are few outliers after Income 120 000

**14. Handle the outlier values in the columnIncome.**

**14.1 1st Approach:Drop the instances where income is greater than 1,50,000, save it indf2**

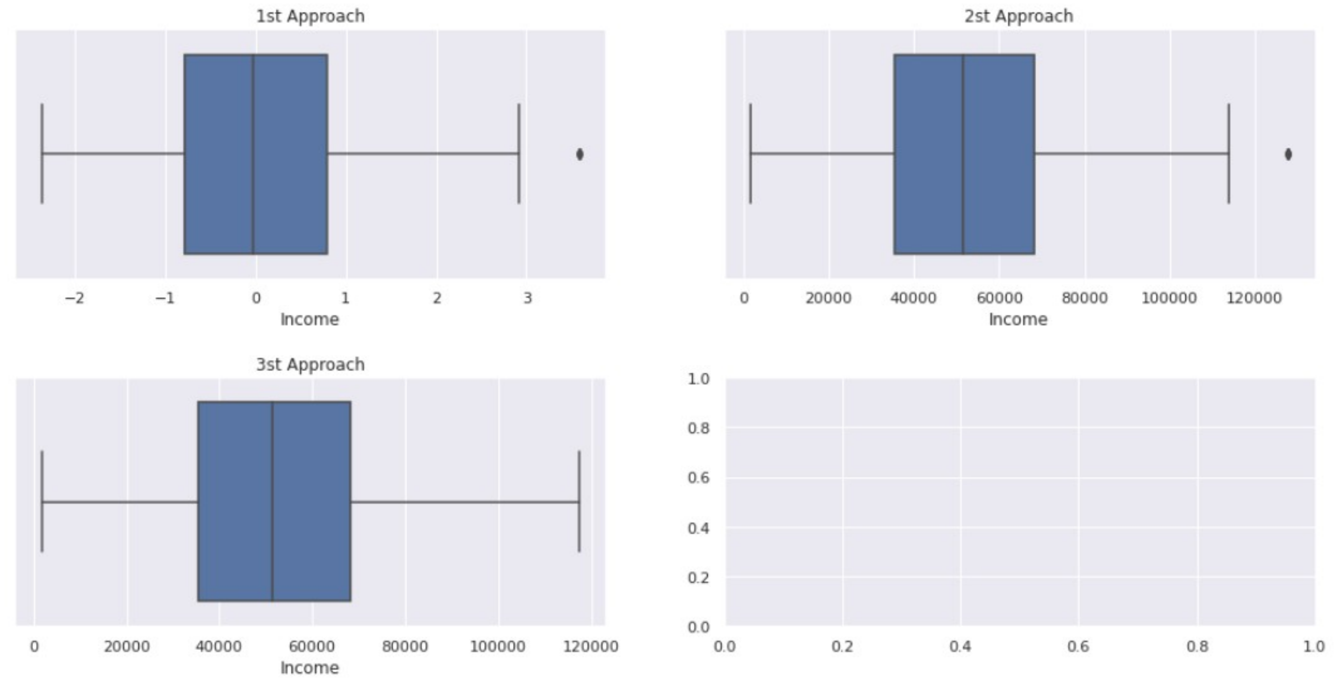
Max income 127766.48
Min income 1730.0

**14.2 2nd Approach:Drop the instances which have outliervalues using the IQR, save itin df3**

Max income 127766.48
Min income 1730.0

**14.3 3rd Approach:Cap the instances to max or min valueusing the IQR, save it in df4**

Max income 117416.25
Min income 1730.0



After 3d Approach there is no detected outliers in the boxplot, because of the maximum income of 117416 which is smaller than in approach 1 and 2

**15. Scale the data in column 'Income' to have mean=0and standard deviation = 1**

For Scaling I have used sklearn.preprocessing StandardScaler.
Mean = -2.4048152157945628e-18
Standart deviation = 1.0

I have some trouble to scale data with output mean equals to 0