PANDAS – PROJECT

DATAFRAME & VISUALIZATION

COURSE – MASTERS IN DATA SCIENCE AND ANALYSTS WITH AI

SUBMITTED BY – MR. YATEEN S. PAWAR

BATCH – 10:00 AM TO 12:00 PM

SUBMISSION DATE – 08th MAY,2023

PROJECT GUIDE

MRS. REKHA ADAK

SUBMITTED TO

MRS. REKHA ADAK

I.T.VEDANT

Decode your dreams

I.T.VEDANT EDUCATION PVT. LTD

- VASHI,NAVI MUMBAI

PROJECT SUMMARY

Under the guidance of Mrs. Rekha Adak, I undertook a project to analyze a sales dataset using Pandas dataframes in Python. The dataset consisted of sales data from a store that sells 3 product categories: accessories, bikes, and clothing, with a total of 17 sub-categories. This data consist for 15 columns and 34866 rows

My primary objective was to extract the required data by running queries on the dataset. To achieve this, I conducted a comprehensive analysis of the sales data using Pandas dataframes. I then used query functions to extract the necessary information from the dataset.

Overall, the project provided an excellent opportunity to apply my data analysis skills using Pandas and gain insights into the sales performance of the store's product categories and sub-categories. The project also helped me gain a deeper understanding of using Pandas dataframes for data analysis purposes.

Source of Dataset: https://www.kaggle.com/datasets/abhishekrp1517/sales-data-for-economic-data-analysis

Column Descriptors:

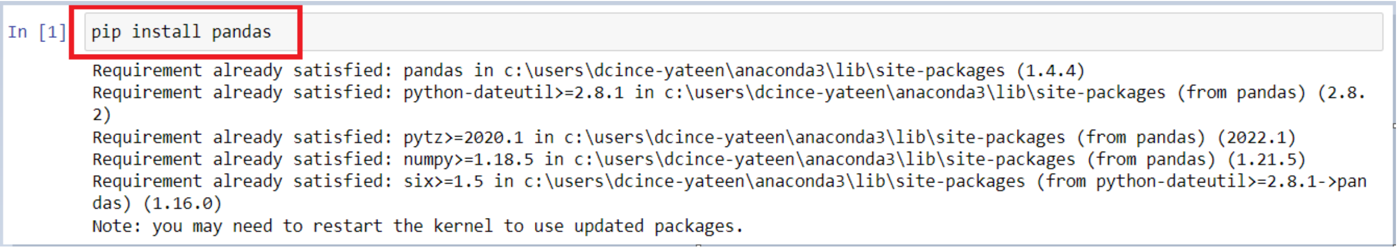
- **Date** – This column represents the date of the transaction i.e. day/month/Year
- **Year** – Represents the Year of the transaction (YYYY)
- **Month** – Represents the Month of the transaction (January,..)
- **Customer Age** – Represents the age of the customer who purchased the product.
- **Customer Gender** – Represents the Gender of the customer (M/F)
- **Country –** Represent the country of the customer (US/UK/FR/GR)
- **State –** Represents the State of the country from which customer belongs to.
- **Product Category –** Represents the primary categories of the products.
- **Sub Category –** This represents the actual product name purchased by customer.
- **Quantity –** Represents the quantity of the purchase
- **Unit Cost -** Represents the cost of producing one unit of the product
- **Unit Price –** Represents the price at which 1 unit was sold.
- **Cost -** Represents the total cost of product sold
- **Revenue -** Represents the total revenue generated by the sales
- **Customer type –** Represents the type of customer (*cx. Bifurcation on the basis of age category*)

Sales Dataset

Assuming you have your python environment ready to work on the Pandas library in python. First we need to install the Pandas module as it is not an inbuild in python.

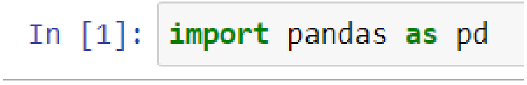
Installation of Pandas module: -

Open the command prompt on your system and write the below command & enter.



```
In [1]: pip install pandas

        Requirement already satisfied: pandas in c:\users\dcince-yateen\anaconda3\lib\site-packages (1.4.4)
        Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dcince-yateen\anaconda3\lib\site-packages (from pandas) (2.8.
        2)
        Requirement already satisfied: pytz>=2020.1 in c:\users\dcince-yateen\anaconda3\lib\site-packages (from pandas) (2022.1)
        Requirement already satisfied: numpy>=1.18.5 in c:\users\dcince-yateen\anaconda3\lib\site-packages (from pandas) (1.21.5)
        Requirement already satisfied: six>=1.5 in c:\users\dcince-yateen\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pan
        das) (1.16.0)
        Note: you may need to restart the kernel to use updated packages.
```

As you can see the above image the pandas module is successfully install and now you just need to restart your kernel. After the installation you can use it in your python script with below command.

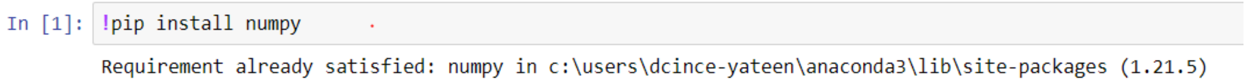


```
In [1]: import pandas as pd
```

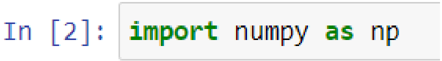
Import pandas as pd commands provides you to use the pandas function on the python scripts. Pd is a variable you can give any variable you like. Most comely used variable is pd

As we will be working on Pandas DataFrame, we also need some addition modules while working, hence we will install the required modules as well.

Installation of Numpy module: -



```
In [1]: !pip install numpy

        Requirement already satisfied: numpy in c:\users\dcince-yateen\anaconda3\lib\site-packages (1.21.5)
```



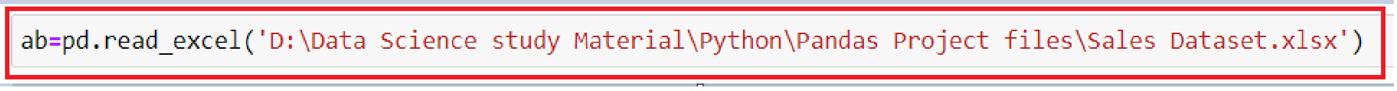
```
In [2]: import numpy as np
```

Import numpy as pd commands provides you to use the numpy function on python scripts. Np is a variable you can given any variable you like. Most commonly used variable is np.

We also need some other libraries as well but we will install it once we will be using it, as of now we will continue with the further steps to import our data-set and build a pandas DataFrame on which we will be working ahead.

Importing file and Build a DataFrame :-

To import the file, type the below command:-



```
ab=pd.read_excel('D:\Data Science study Material\Python\Pandas Project files\Sales Dataset.xlsx')
```

Breakdown as below:-

ab – in the above command ab is given a name to a DataFrame which will be the output of the above command. If we need to see the created DataFrame, we can see it anytime by using ab as dataframe name.

pd. – pd is a pandas variable that we need to use as this will inform system to use the pandas library to build the dataframe.(*make sure you give dot after pd*)

read_excel – this command is important as we are importing an Excel file through pandas library to create Dataframe. Like Excel file we can also import files

example: - csv file,json file,etc. we just need to change the extension in the command to import file. Like here we have excel file hence we use the command -> read_excel

() – After adding the file extension now we need to add the file path to import it from where it is saved. This path is pasted within the **()** parenthesis and the entire path should be in single or double quote.

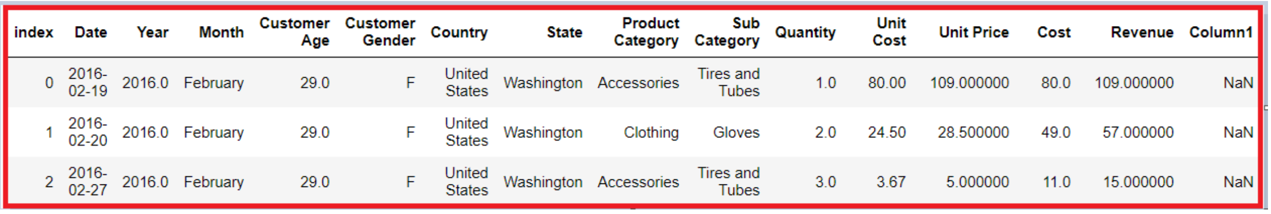
Once the entire command is ready you can enter check the out of with the name given to the dataframe.

Apart from the above procedure you can also import the file through the file name by avoiding the path procedure. You just have to mention the name of the file instead of file path. Make sure the original file is saved in the same folder where your python files is saved in the system.

Now lets check if we have created the DataFrame or not.

Command as below:-

- Enter the name of the file and run the program.
  (*image below*)

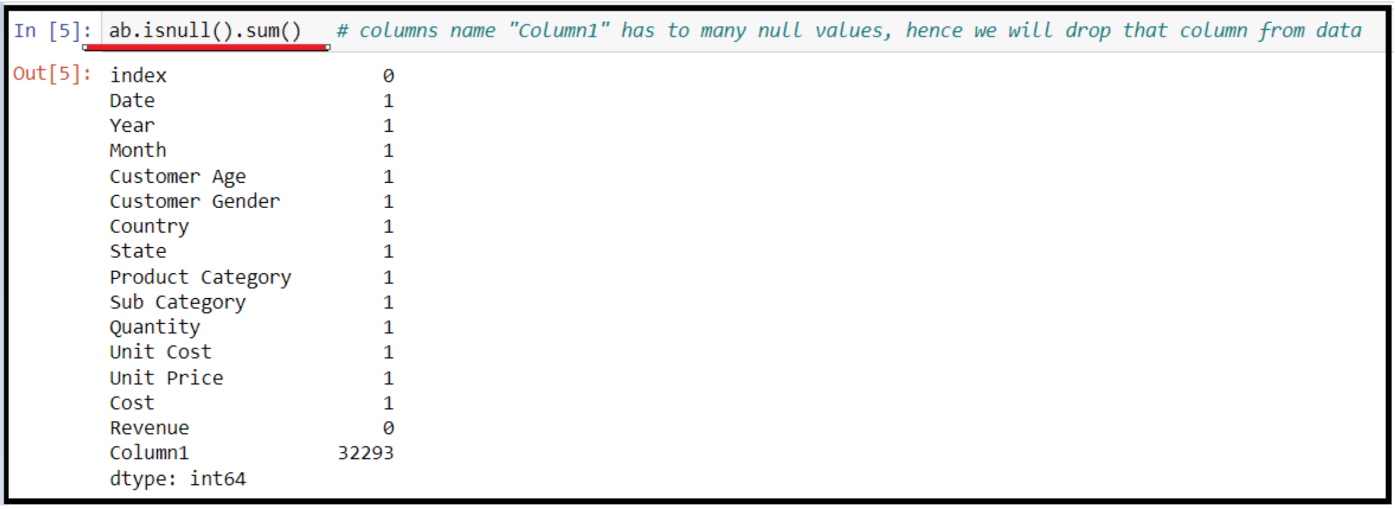


index	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Column1
0	2016-02-19	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.000000	80.0	109.000000	NaN
1	2016-02-20	2016.0	February	29.0	F	United States	Washington	Clothing	Gloves	2.0	24.50	28.500000	49.0	57.000000	NaN
2	2016-02-27	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	3.0	3.67	5.000000	11.0	15.000000	NaN

We have successfully imported the file and created the DataFrame. Now you can access this dataframe any time while working on this same. Just you need to enter the name and run the program.

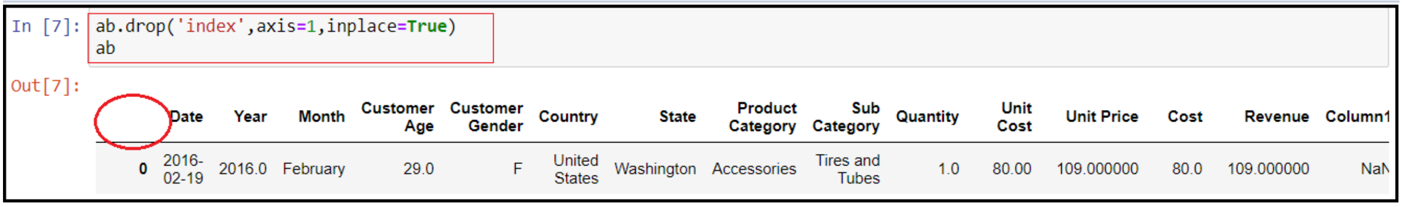
Further we will perform pandas functions on the above created dataframe and also analyze the data.

Check the Null values in the dataframe:-



```
In [5]: ab.isnull().sum()   # columns name "Column1" has to many null values, hence we will drop that column from data

Out[5]: index                  0
        Date                   1
        Year                   1
        Month                  1
        Customer Age           1
        Customer Gender        1
        Country                1
        State                  1
        Product Category       1
        Sub Category           1
        Quantity               1
        Unit Cost              1
        Unit Price             1
        Cost                   1
        Revenue                0
        Column1            32293
        dtype: int64
```

As we can see the dataframe the last column i.e. “Column1” has too many null values hence we will drop that column. Before that we will also drop the index column as it is not required.



```
In [7]: ab.drop('index',axis=1,inplace=True)
        ab
```

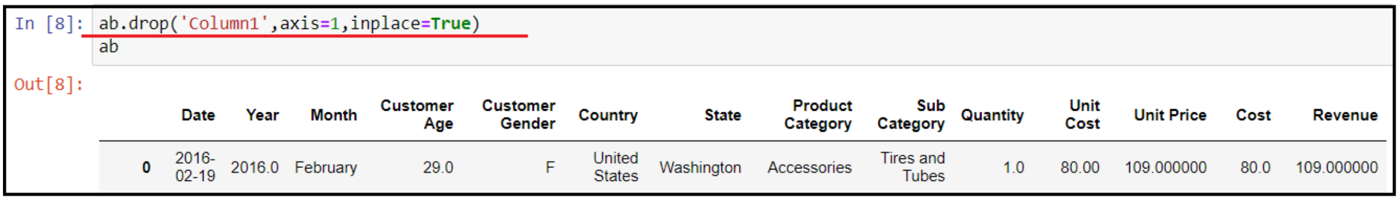
Out[7]:

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Column1
0	2016-02-19	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.000000	80.0	109.000000	NaN

As the above result shows, the index column is removed from the dataframe permanently.

We have used the drop function - **drop()**

we will also drop the last columns i.e. “Column1”



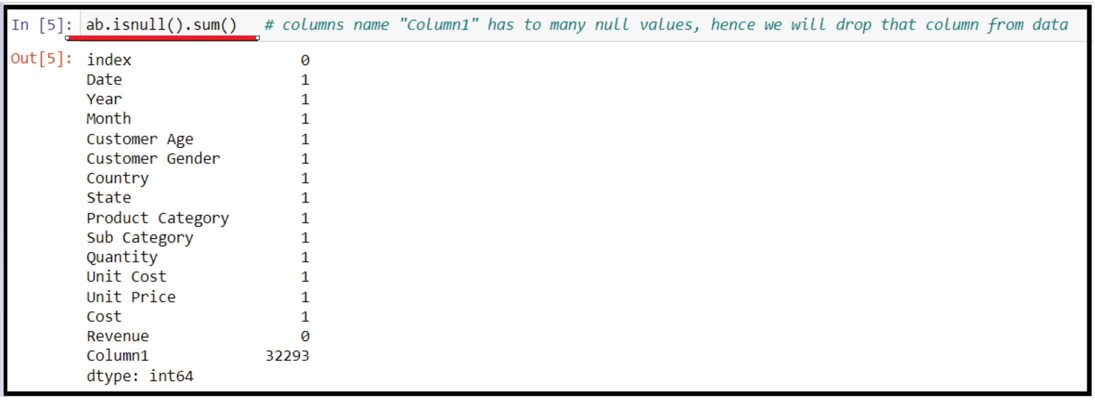
```
In [8]: ab.drop('Column1',axis=1,inplace=True)
        ab
```

Out[8]:

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
0	2016-02-19	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.000000	80.0	109.000000

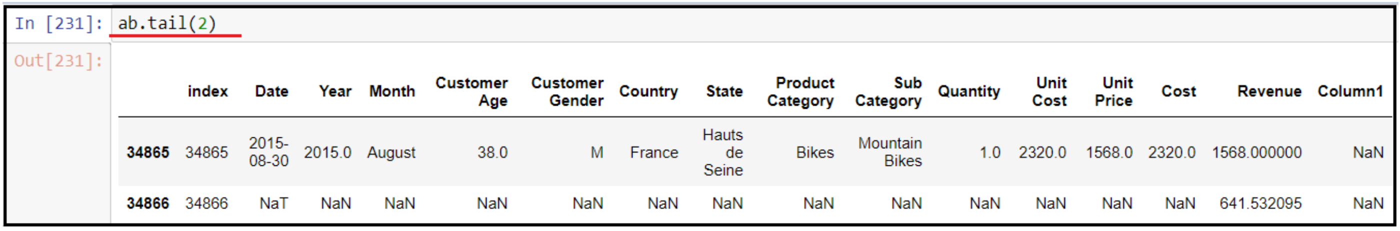
As we can see the last column now is a “Revenue” column . the earlier “columns1” has been dropped permanently.

As we check the above null values we found too many null values in the “column1” and we have already dropped the same now, but still there are null values in every column as it is showing 1 count for each column. (*find the image below*)



```
In [5]: ab.isnull().sum()   # columns name "Column1" has to many null values, hence we will drop that column from data

Out[5]: index                  0
        Date                   1
        Year                   1
        Month                  1
        Customer Age           1
        Customer Gender        1
        Country                1
        State                  1
        Product Category       1
        Sub Category           1
        Quantity               1
        Unit Cost              1
        Unit Price             1
        Cost                   1
        Revenue                0
        Column1            32293
        dtype: int64
```

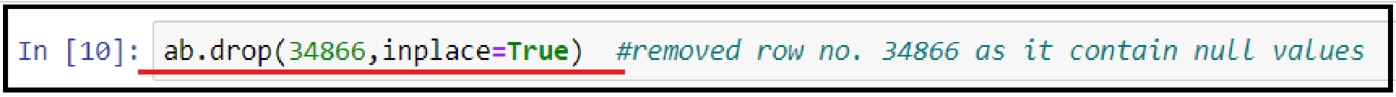
Check the null value row and remove it from the dataframe :-



```
In [231]: ab.tail(2)
Out[231]:
```

	index	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Column1
34865	34865	2015-08-30	2015.0	August	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	1.0	2320.0	1568.0	2320.0	1568.000000	NaN
34866	34866	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	641.532095	NaN

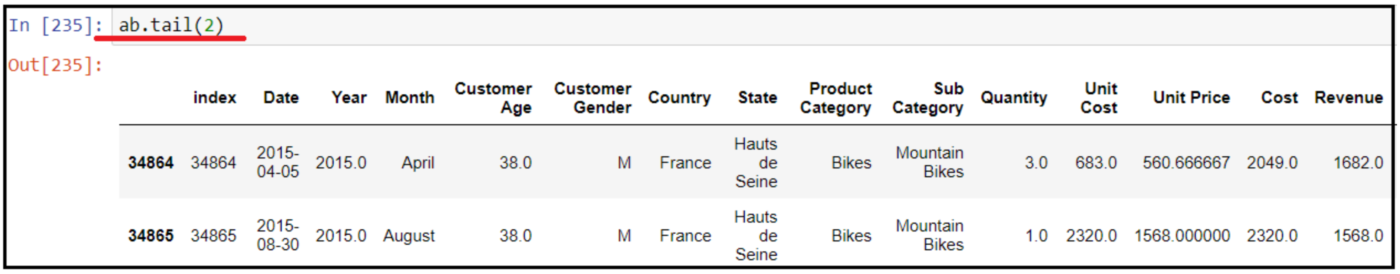
Using the tail function, we find the row which is having null values. We can also find the row number of the null values, using the row number we will just erase the column.



```
In [10]: ab.drop(34866,inplace=True)   #removed row no. 34866 as it contain null values
```

We have dropped the row through drop function and using row number as an argument.

Lets check if the row has been erased or not. YES! It is successfully removed.

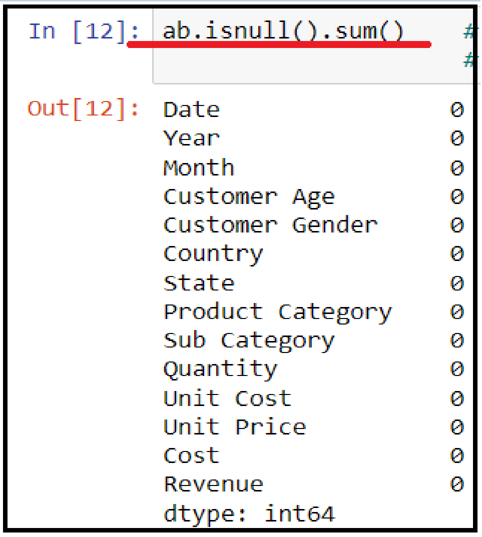


```
In [235]: ab.tail(2)
Out[235]:
```

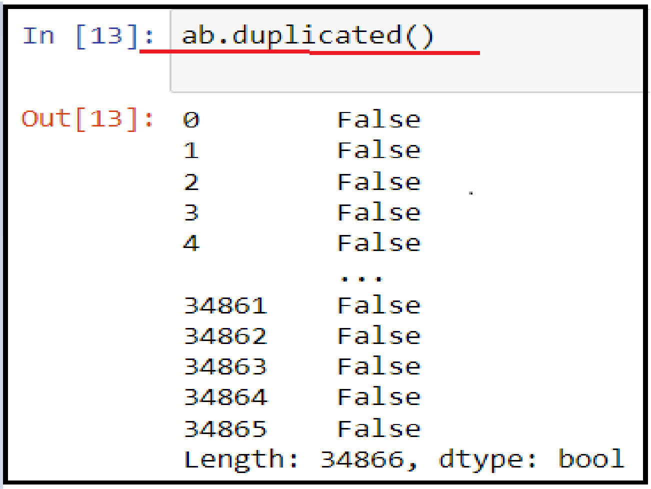
	index	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
34864	34864	2015-04-05	2015.0	April	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	3.0	683.0	560.666667	2049.0	1682.0
34865	34865	2015-08-30	2015.0	August	38.0	M	France	Hauts de Seine	Bikes	Mountain Bikes	1.0	2320.0	1568.000000	2320.0	1568.0

Recheck the null values if any :-

➔ The result for the Null values are 0, which means there is no Null values in our dataframe. The dataframe is clean.



```
In [12]: ab.isnull().sum()                #
                                          #

Out[12]: Date                 0
         Year                 0
         Month                0
         Customer Age         0
         Customer Gender      0
         Country              0
         State                0
         Product Category     0
         Sub Category         0
         Quantity             0
         Unit Cost            0
         Unit Price           0
         Cost                 0
         Revenue              0
         dtype: int64
```



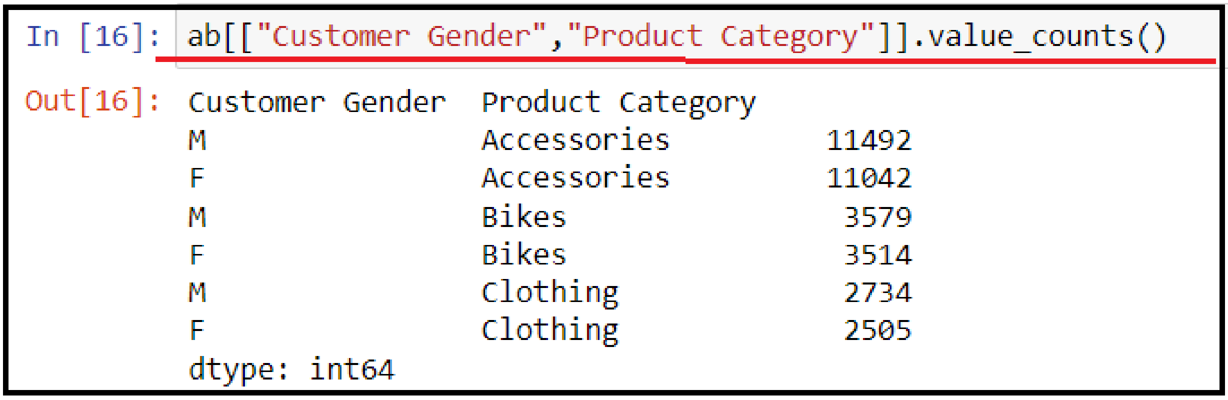
```
In [13]: ab.duplicated()

Out[13]: 0        False
         1        False
         2        False
         3        False
         4        False
                  ...
         34861    False
         34862    False
         34863    False
         34864    False
         34865    False
         Length: 34866, dtype: bool
```

Check if there is any duplicate transaction :-

Using the duplicated function we checked there is no duplicate values in the dataframe.
The bool values turns True if there is any duplicate entry.

Value Counts Function:-

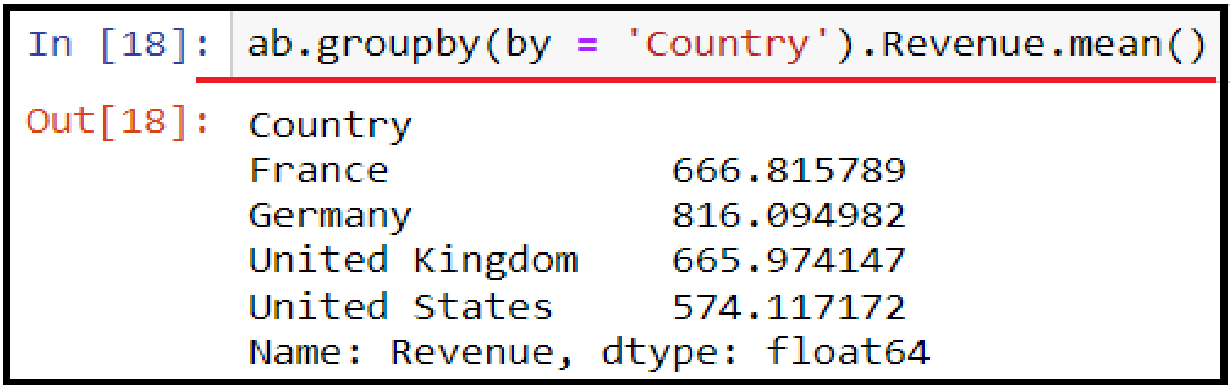
- Count product category wise sales purchased by Male and Female



```
In [16]: ab[["Customer Gender","Product Category"]].value_counts()

Out[16]: Customer Gender  Product Category
         M                Accessories         11492
         F                Accessories         11042
         M                Bikes                3579
         F                Bikes                3514
         M                Clothing             2734
         F                Clothing             2505
         dtype: int64
```

Group by function:-

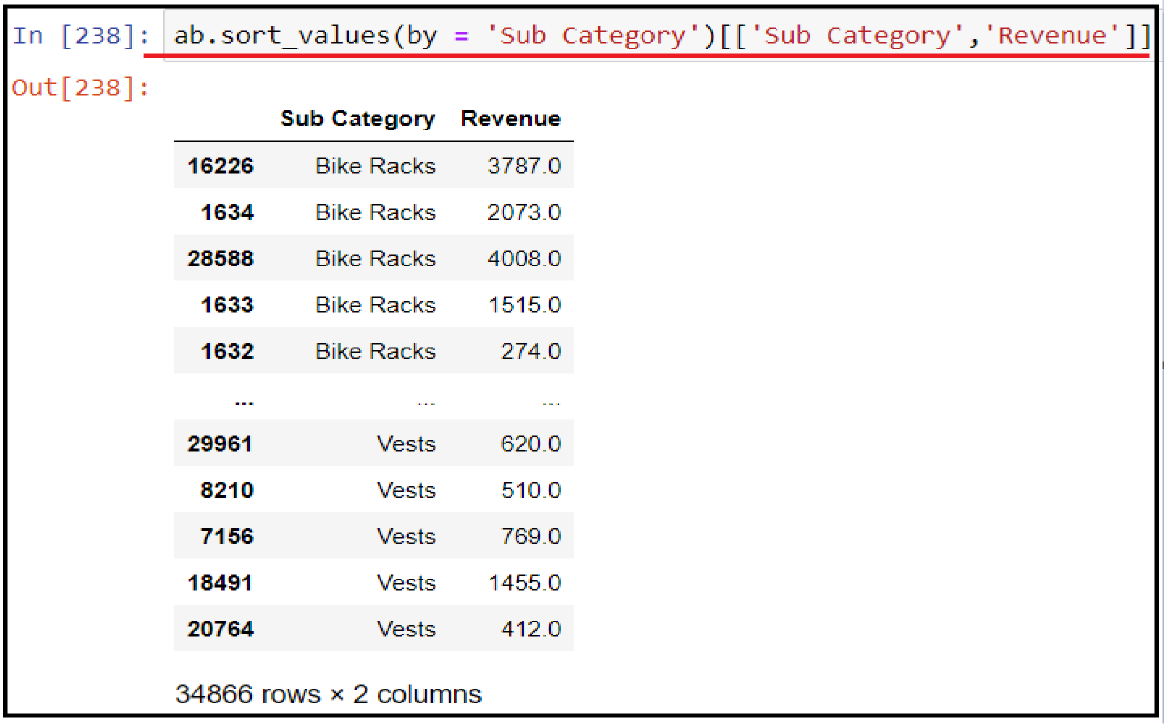
- Show the country wise average revenue.



```
In [18]: ab.groupby(by = 'Country').Revenue.mean()

Out[18]: Country
         France            666.815789
         Germany           816.094982
         United Kingdom    665.974147
         United States     574.117172
         Name: Revenue, dtype: float64
```

Sort Values function:-

- Show the sub-category wise revenue.



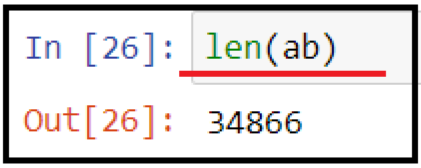
```
In [238]: ab.sort_values(by = 'Sub Category')[['Sub Category','Revenue']]

Out[238]:
```

	Sub Category	Revenue
16226	Bike Racks	3787.0
1634	Bike Racks	2073.0
28588	Bike Racks	4008.0
1633	Bike Racks	1515.0
1632	Bike Racks	274.0
...	...	...
29961	Vests	620.0
8210	Vests	510.0
7156	Vests	769.0
18491	Vests	1455.0
20764	Vests	412.0

34866 rows × 2 columns

Length function:-

- Find the length of the dataframe.

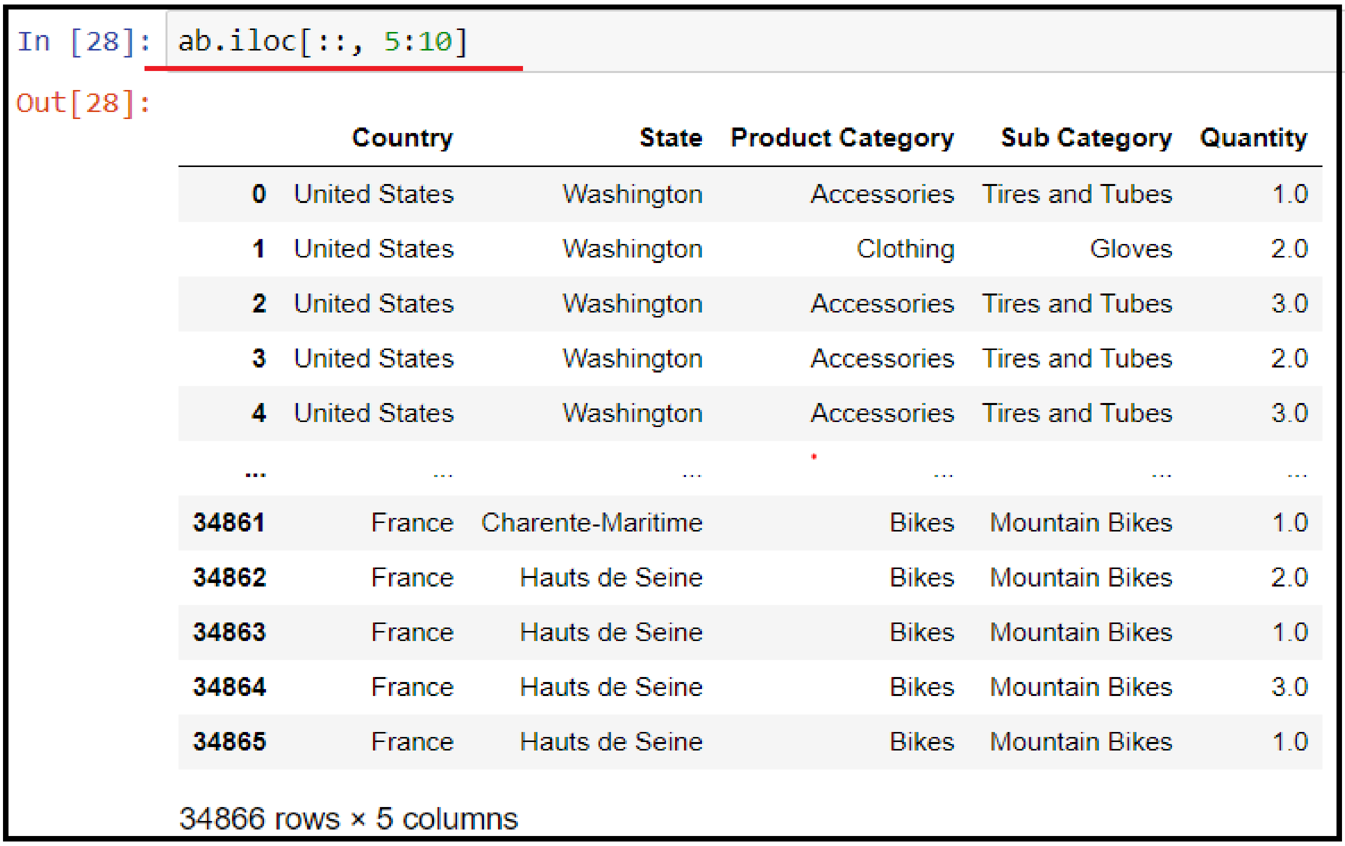


```
In [26]: len(ab)

Out[26]: 34866
```

This show the output of number of rows in the dataframe.

iloc function :-

- Retrieve the columns from Country to Quantity.



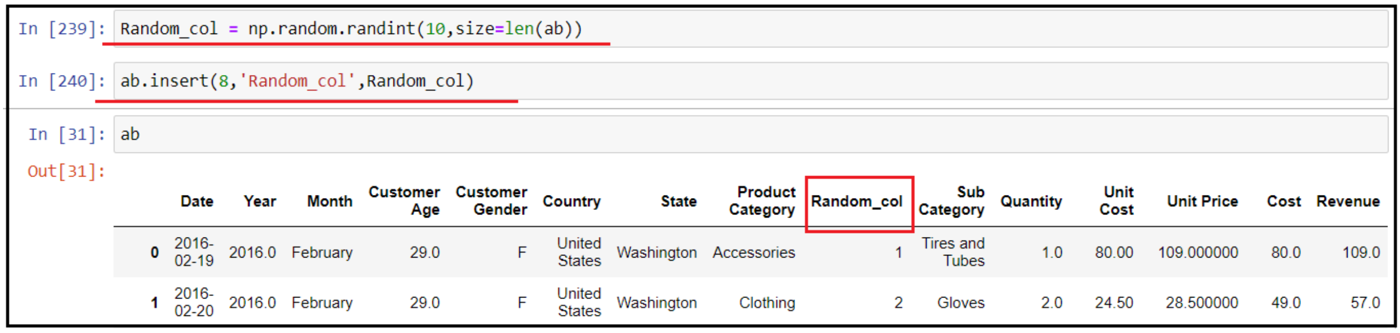
```
In [28]: ab.iloc[::, 5:10]

Out[28]:
```

	Country	State	Product Category	Sub Category	Quantity
0	United States	Washington	Accessories	Tires and Tubes	1.0
1	United States	Washington	Clothing	Gloves	2.0
2	United States	Washington	Accessories	Tires and Tubes	3.0
3	United States	Washington	Accessories	Tires and Tubes	2.0
4	United States	Washington	Accessories	Tires and Tubes	3.0
...	...	...	...	...	...
34861	France	Charente-Maritime	Bikes	Mountain Bikes	1.0
34862	France	Hauts de Seine	Bikes	Mountain Bikes	2.0
34863	France	Hauts de Seine	Bikes	Mountain Bikes	1.0
34864	France	Hauts de Seine	Bikes	Mountain Bikes	3.0
34865	France	Hauts de Seine	Bikes	Mountain Bikes	1.0

34866 rows × 5 columns

Random function (Numpy) :-

- Add a column to a dataframe and insert the random values between 0 to 10.

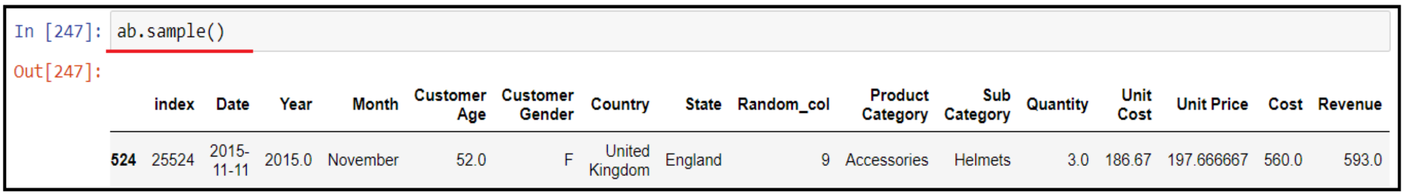


```
In [239]: Random_col = np.random.randint(10,size=len(ab))

In [240]: ab.insert(8,'Random_col',Random_col)

In [31]: ab
Out[31]:
```

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Random_col	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
0	2016-02-19	2016.0	February	29.0	F	United States	Washington	Accessories	1	Tires and Tubes	1.0	80.00	109.000000	80.0	109.0
1	2016-02-20	2016.0	February	29.0	F	United States	Washington	Clothing	2	Gloves	2.0	24.50	28.500000	49.0	57.0

Using Numpy random.randint function we have add random values to the dataframe between 0 -10 also we have created a new column inthe existing dataframe.

Sample Function :-



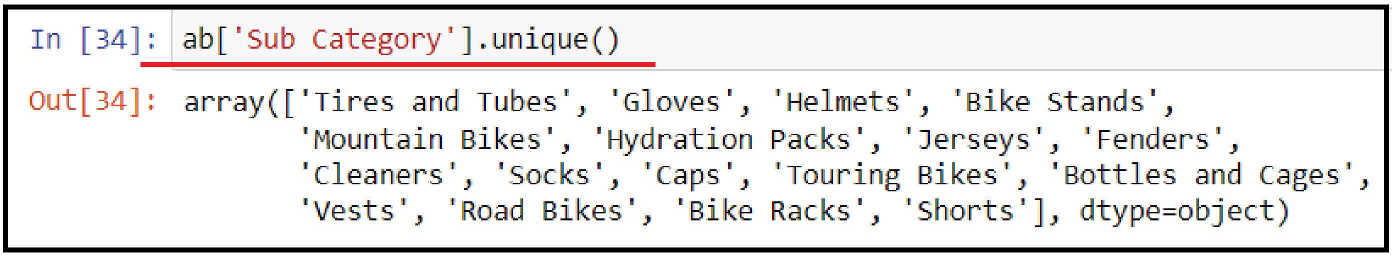
```
In [247]: ab.sample()
Out[247]:
```

	index	Date	Year	Month	Customer Age	Customer Gender	Country	State	Random_col	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
524	25524	2015-11-11	2015.0	November	52.0	F	United Kingdom	England	9	Accessories	Helmets	3.0	186.67	197.666667	560.0	593.0

Sample function will result any single row from the dataframe as a sample of the data. If you want to see more rows or more sample then you just need to pass an argument in the **()** parenthesis.

Unique Function:-

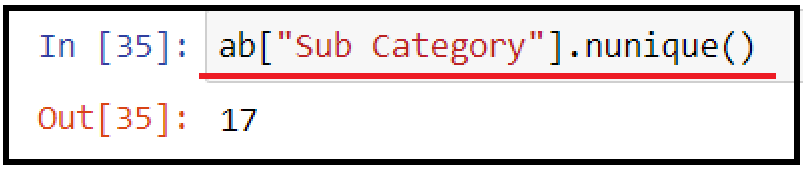
- Find the unique values in the sub-category column.



```
In [34]: ab['Sub Category'].unique()

Out[34]: array(['Tires and Tubes', 'Gloves', 'Helmets', 'Bike Stands',
                'Mountain Bikes', 'Hydration Packs', 'Jerseys', 'Fenders',
                'Cleaners', 'Socks', 'Caps', 'Touring Bikes', 'Bottles and Cages',
                'Vests', 'Road Bikes', 'Bike Racks', 'Shorts'], dtype=object)
```

Nunique function:-

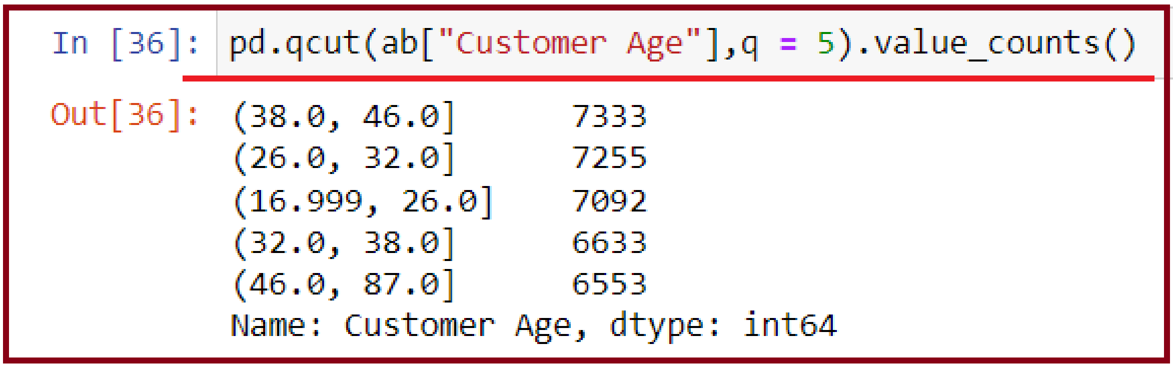
- Find how many unique values are mention in the column sub-category



```
In [35]: ab["Sub Category"].nunique()

Out[35]: 17
```

qcut function:-

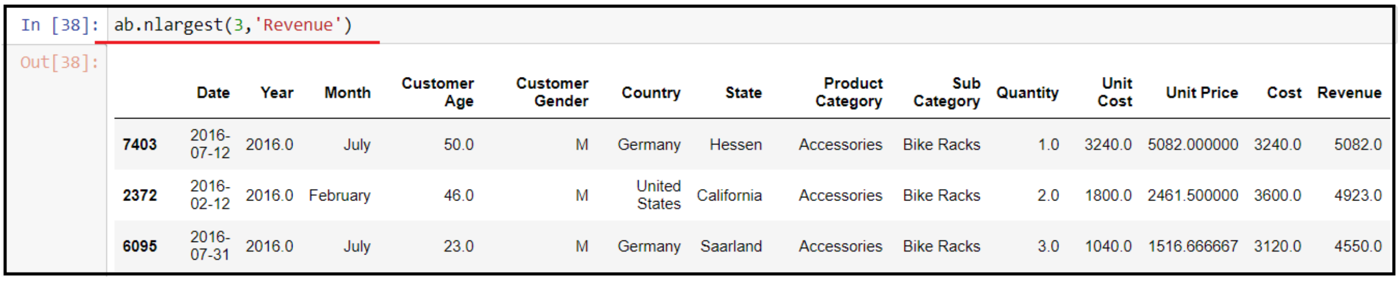
- Distribue the customer in 5 category considering their age.



```
In [36]: pd.qcut(ab["Customer Age"],q = 5).value_counts()

Out[36]: (38.0, 46.0]      7333
         (26.0, 32.0]      7255
         (16.999, 26.0]    7092
         (32.0, 38.0]      6633
         (46.0, 87.0]      6553
         Name: Customer Age, dtype: int64
```

nlargest & nsmallest function:-

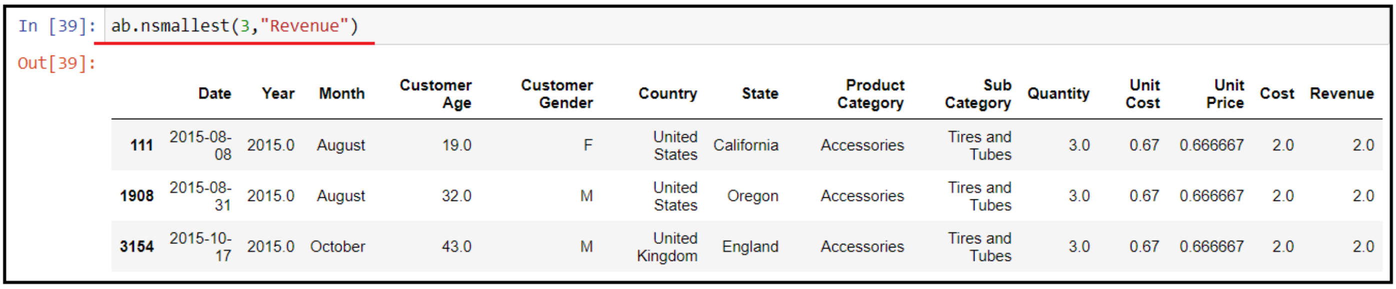
- **Show the 3 largest revenue**



```
In [38]: ab.nlargest(3,'Revenue')
Out[38]:
```

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
7403	2016-07-12	2016.0	July	50.0	M	Germany	Hessen	Accessories	Bike Racks	1.0	3240.0	5082.000000	3240.0	5082.0
2372	2016-02-12	2016.0	February	46.0	M	United States	California	Accessories	Bike Racks	2.0	1800.0	2461.500000	3600.0	4923.0
6095	2016-07-31	2016.0	July	23.0	M	Germany	Saarland	Accessories	Bike Racks	3.0	1040.0	1516.666667	3120.0	4550.0

- **Show the 3 smallest revenue**

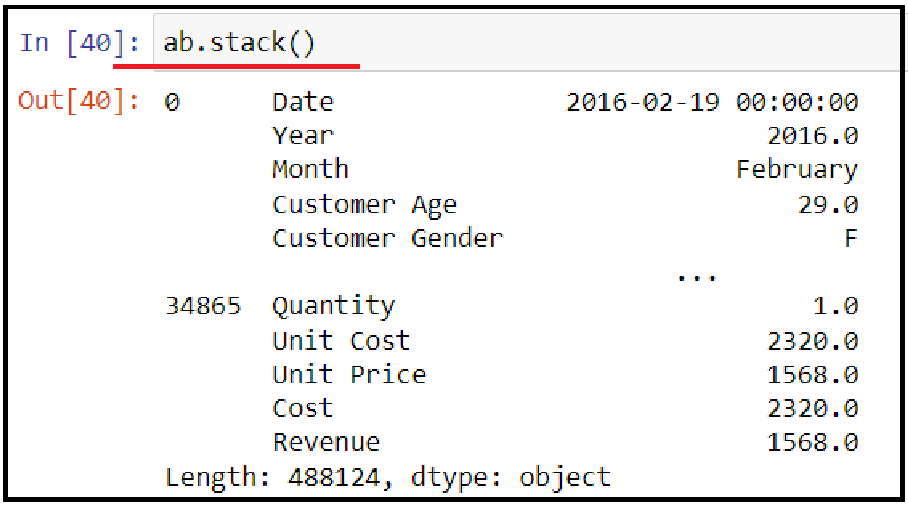


```
In [39]: ab.nsmallest(3,"Revenue")
Out[39]:
```

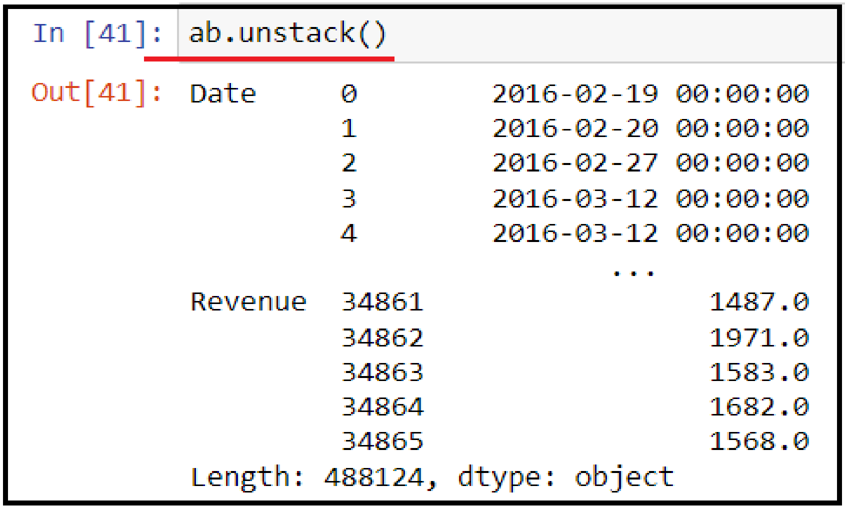
	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue
111	2015-08-08	2015.0	August	19.0	F	United States	California	Accessories	Tires and Tubes	3.0	0.67	0.666667	2.0	2.0
1908	2015-08-31	2015.0	August	32.0	M	United States	Oregon	Accessories	Tires and Tubes	3.0	0.67	0.666667	2.0	2.0
3154	2015-10-17	2015.0	October	43.0	M	United Kingdom	England	Accessories	Tires and Tubes	3.0	0.67	0.666667	2.0	2.0

stack & unstack function:-

- Stack function turns columns to rows
- Unstack function turns rows to columns



```
In [40]: ab.stack()

Out[40]: 0      Date                   2016-02-19 00:00:00
                Year                                2016.0
                Month                             February
                Customer Age                          29.0
                Customer Gender                          F
                                                ...
         34865  Quantity                               1.0
                Unit Cost                           2320.0
                Unit Price                          1568.0
                Cost                                2320.0
                Revenue                             1568.0
         Length: 488124, dtype: object
```

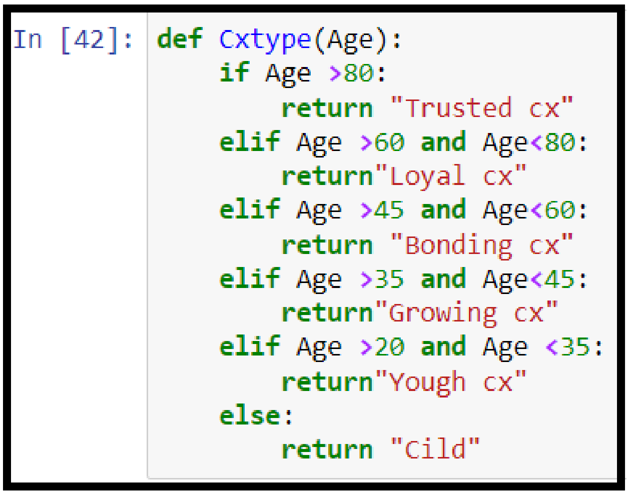


```
In [41]: ab.unstack()

Out[41]: Date     0        2016-02-19 00:00:00
                  1        2016-02-20 00:00:00
                  2        2016-02-27 00:00:00
                  3        2016-03-12 00:00:00
                  4        2016-03-12 00:00:00
                                   ...
         Revenue  34861                 1487.0
                  34862                 1971.0
                  34863                 1583.0
                  34864                 1682.0
                  34865                 1568.0
         Length: 488124, dtype: object
```

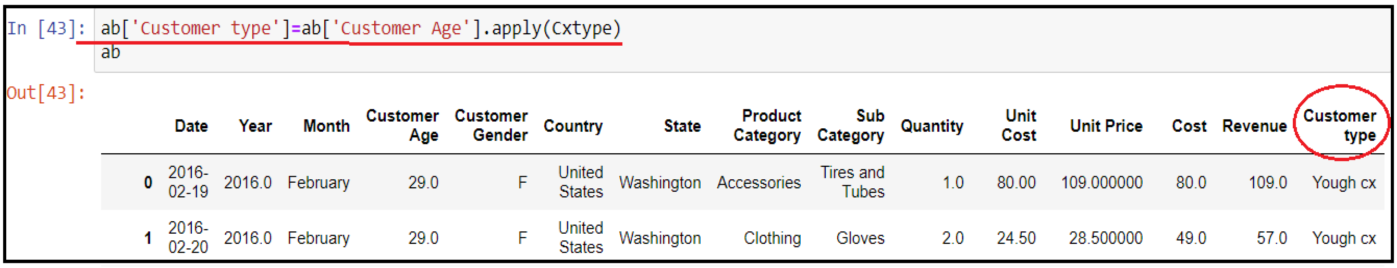
Apply function:-

- Sort the customer by their age group and add new column “Customer type”
- Age >80 – 99 = “Trusted cx”
- Age >60 < 80 = “Loyal cx”
- Age >45 <60 = “Bonding cx”
- Age >35 <45 = “Growing cx”
- Age >20 <35 = “Young cx”
- Age <20 = “Child cx”

User defined function:



```
In [42]: def Cxtype(Age):
             if Age >80:
                 return "Trusted cx"
             elif Age >60 and Age<80:
                 return"Loyal cx"
             elif Age >45 and Age<60:
                 return "Bonding cx"
             elif Age >35 and Age<45:
                 return"Growing cx"
             elif Age >20 and Age <35:
                 return"Yough cx"
             else:
                 return "Cild"
```

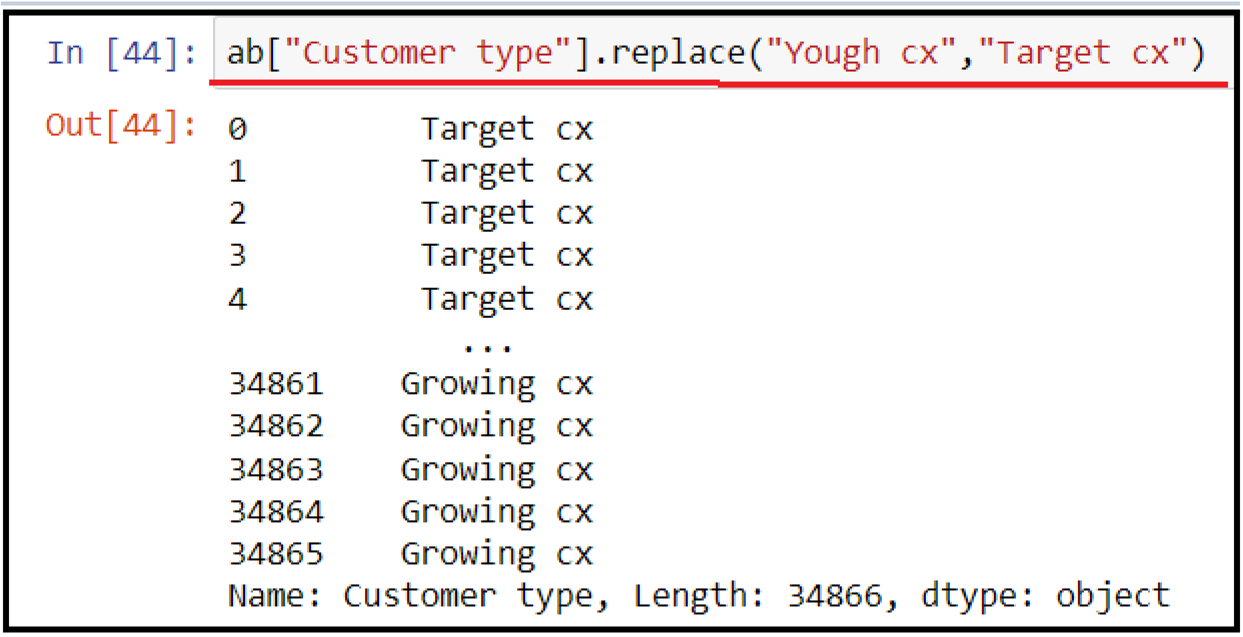


```
In [43]: ab['Customer type']=ab['Customer Age'].apply(Cxtype)
         ab
Out[43]:
```

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Customer type
0	2016-02-19	2016.0	February	29.0	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.000000	80.0	109.0	Yough cx
1	2016-02-20	2016.0	February	29.0	F	United States	Washington	Clothing	Gloves	2.0	24.50	28.500000	49.0	57.0	Yough cx

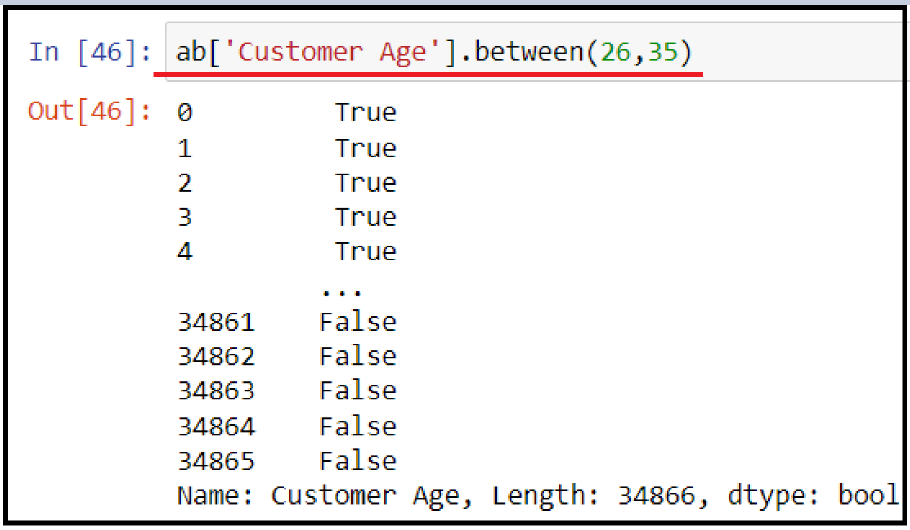
Replace function:-

- Replace customer type from “Young cx” to “Target cx”.



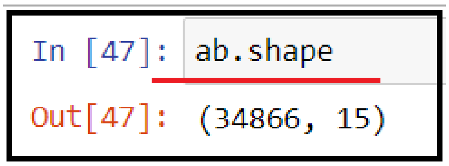
```
In [44]: ab["Customer type"].replace("Yough cx","Target cx")

Out[44]: 0            Target cx
         1            Target cx
         2            Target cx
         3            Target cx
         4            Target cx
                        ...
         34861      Growing cx
         34862      Growing cx
         34863      Growing cx
         34864      Growing cx
         34865      Growing cx
         Name: Customer type, Length: 34866, dtype: object
```

Between Function:-



```
In [46]: ab['Customer Age'].between(26,35)

Out[46]: 0          True
         1          True
         2          True
         3          True
         4          True
                   ...
         34861     False
         34862     False
         34863     False
         34864     False
         34865     False
         Name: Customer Age, Length: 34866, dtype: bool
```

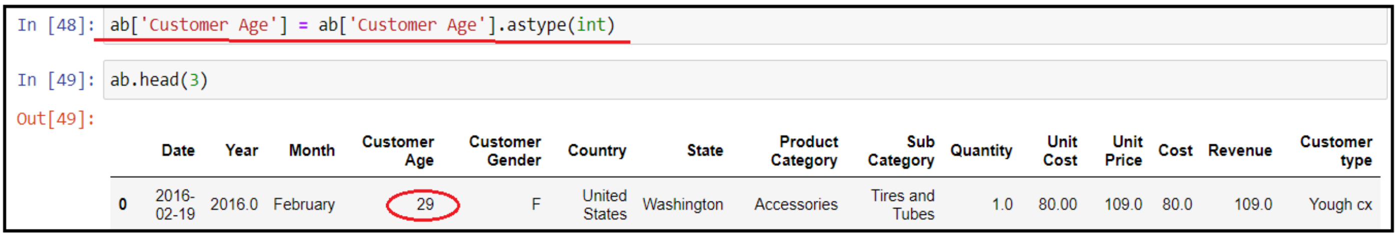
Shape function:-



```
In [47]: ab.shape

Out[47]: (34866, 15)
```

astype function:-

- Convert "Customer Age" column into integer type.

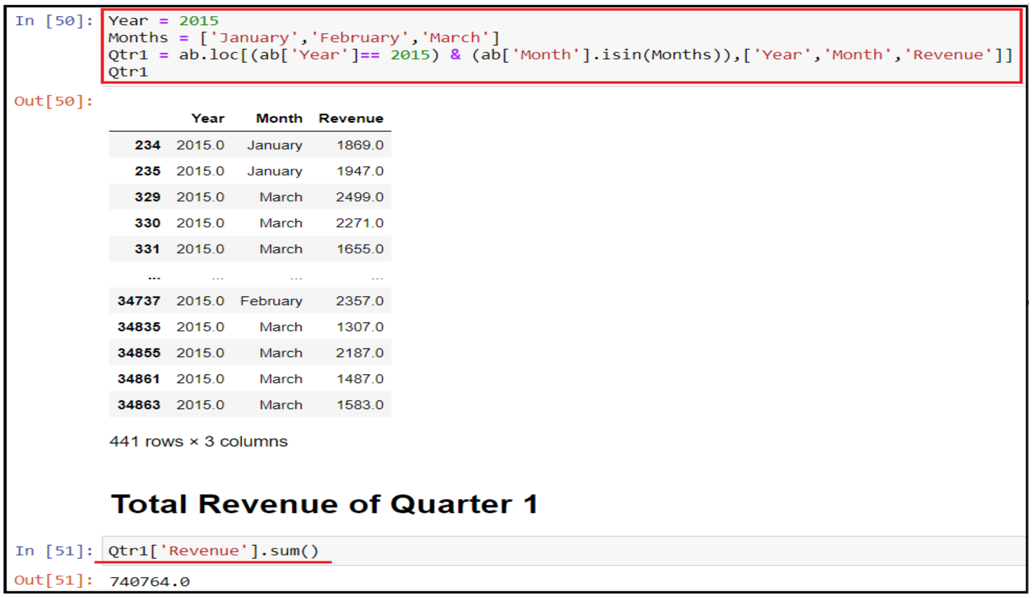


```
In [48]: ab['Customer Age'] = ab['Customer Age'].astype(int)

In [49]: ab.head(3)

Out[49]:
```

	Date	Year	Month	Customer Age	Customer Gender	Country	State	Product Category	Sub Category	Quantity	Unit Cost	Unit Price	Cost	Revenue	Customer type
0	2016-02-19	2016.0	February	29	F	United States	Washington	Accessories	Tires and Tubes	1.0	80.00	109.0	80.0	109.0	Yough cx

Loc function:-

- Calculate the Sales Revenue of year 2015 1st Quarter.



```
In [50]: Year = 2015
         Months = ['January','February','March']
         Qtr1 = ab.loc[(ab['Year']== 2015) & (ab['Month'].isin(Months)),['Year','Month','Revenue']]
         Qtr1

Out[50]:
```

	Year	Month	Revenue
234	2015.0	January	1869.0
235	2015.0	January	1947.0
329	2015.0	March	2499.0
330	2015.0	March	2271.0
331	2015.0	March	1655.0
...	...	...	...
34737	2015.0	February	2357.0
34835	2015.0	March	1307.0
34855	2015.0	March	2187.0
34861	2015.0	March	1487.0
34863	2015.0	March	1583.0

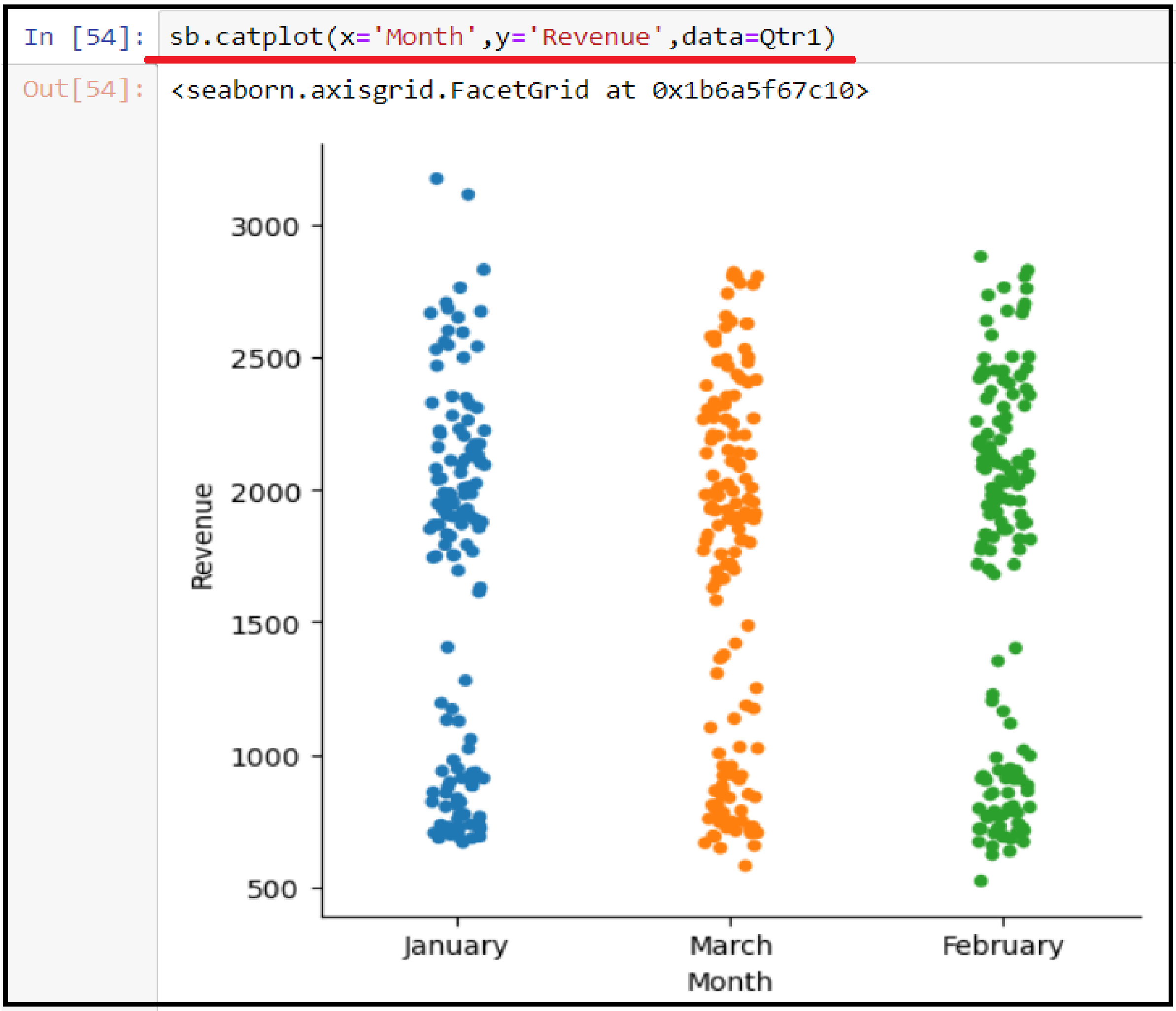
441 rows × 3 columns

Total Revenue of Quarter 1

```
In [51]: Qtr1['Revenue'].sum()

Out[51]: 740764.0
```

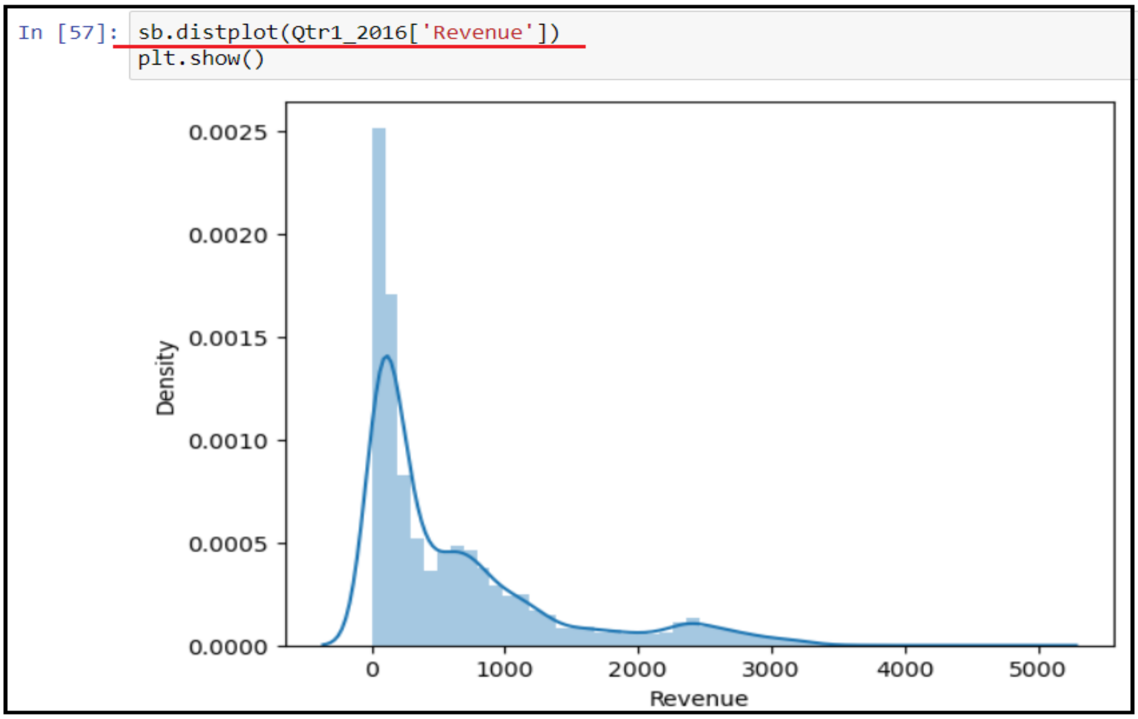
Analysis of the dataset and represent the information through graphs.

Considering the above picture, showcasing the sales of first Quarter 2015.Lets represent the data in graphs.



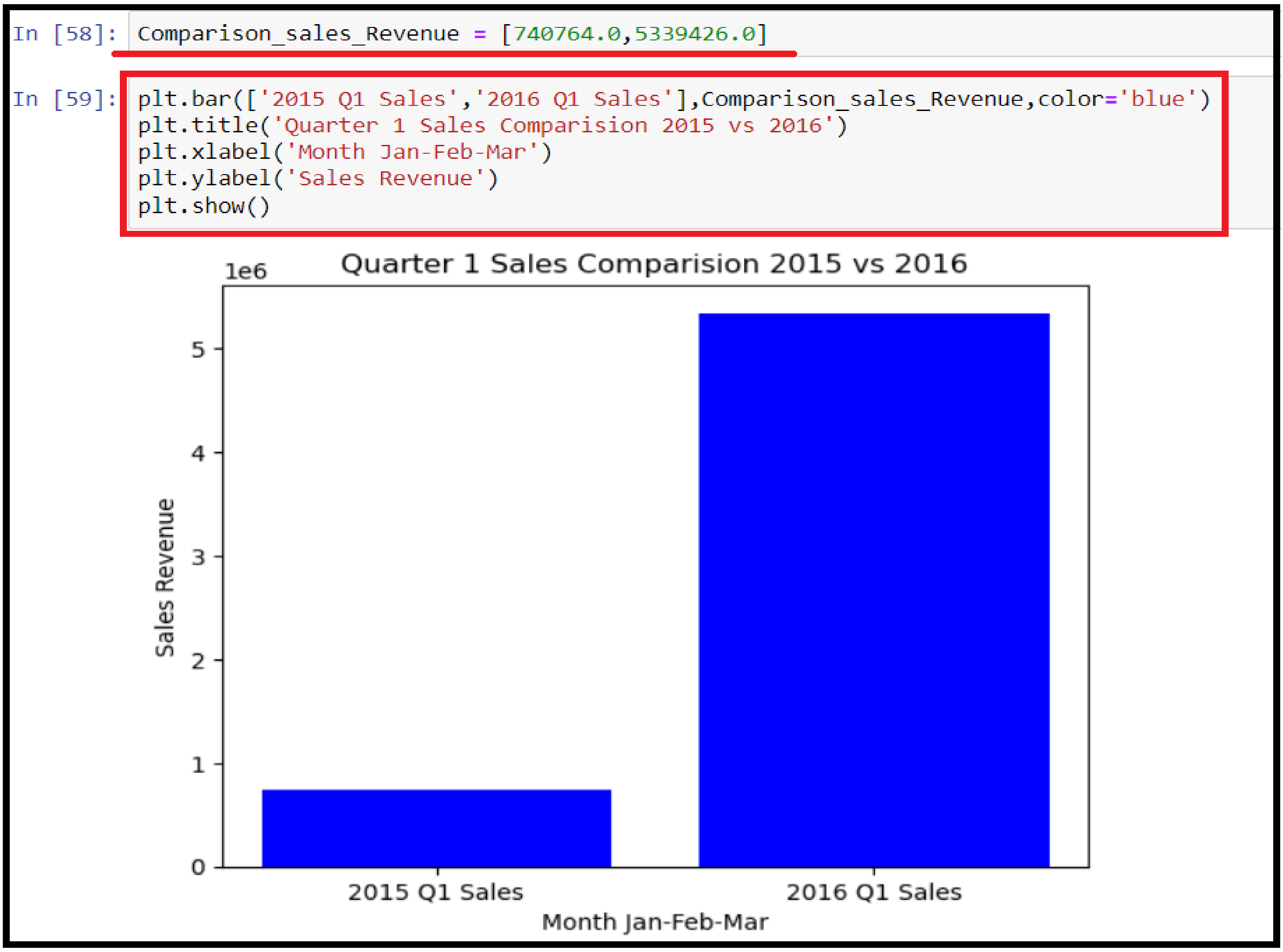
```
In [54]: sb.catplot(x='Month',y='Revenue',data=Qtr1)
Out[54]: <seaborn.axisgrid.FacetGrid at 0x1b6a5f67c10>
```

Using the same formula we will calculate the 2016 year sales for Quarter 1 and show the sales on graph.



```
In [57]: sb.distplot(Qtr1_2016['Revenue'])
         plt.show()
```

Show the Revenue comparison between 2015 vs 2016 for Quarter 1

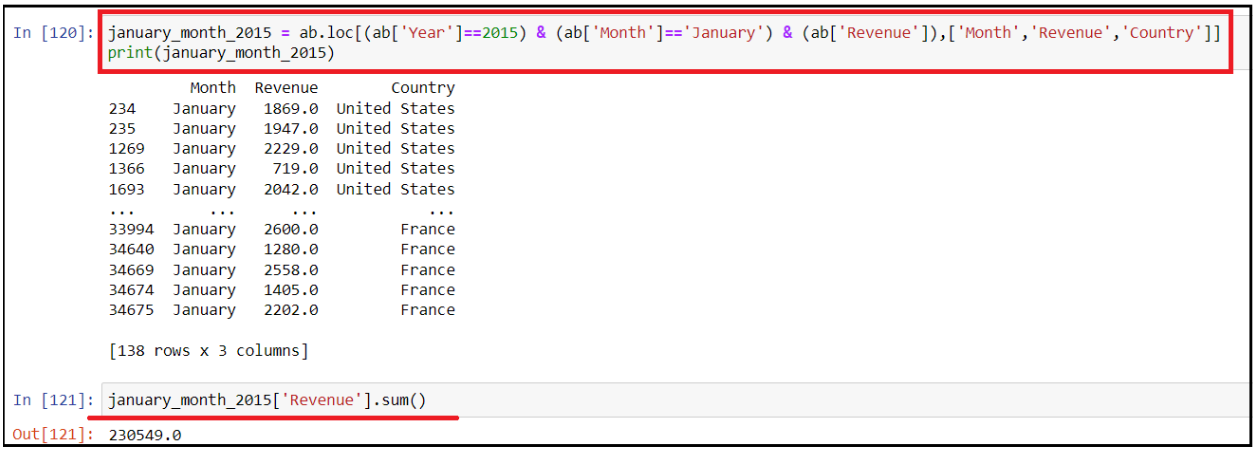


```
In [58]: Comparison_sales_Revenue = [740764.0,5339426.0]

In [59]: plt.bar(['2015 Q1 Sales','2016 Q1 Sales'],Comparison_sales_Revenue,color='blue')
         plt.title('Quarter 1 Sales Comparision 2015 vs 2016')
         plt.xlabel('Month Jan-Feb-Mar')
         plt.ylabel('Sales Revenue')
         plt.show()
```

In the same process we can calculate the Total sales revenue for Quarter 2 / Quarter 3 and present them through the graph. As per our dataset, We have data for year 2016 till the month of Jul. That means the store can make the strategy and plan of action to enhance the sales revenue and achieve the set target.

Show 2015 monthly sales revenue

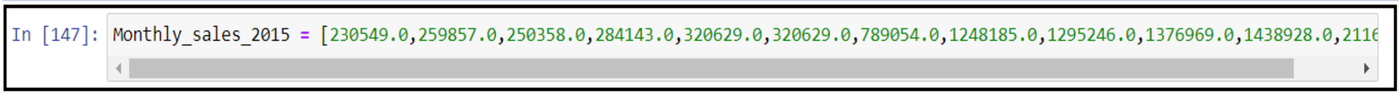
- Calculation of 2015 January Sales Revenue:-



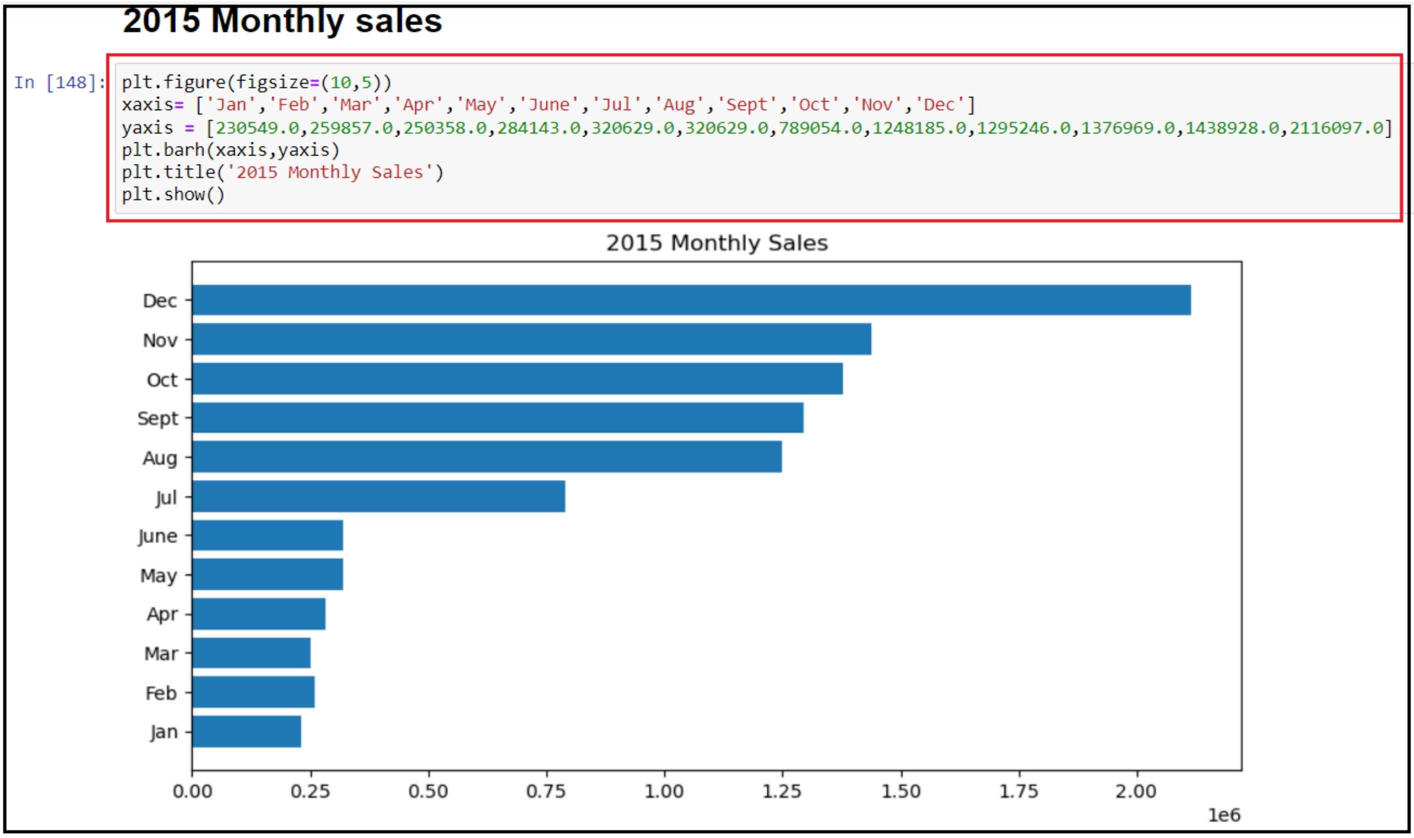
```
In [120]: january_month_2015 = ab.loc[(ab['Year']==2015) & (ab['Month']=='January') & (ab['Revenue']),['Month','Revenue','Country']]
          print(january_month_2015)

             Month  Revenue        Country
          234    January   1869.0  United States
          235    January   1947.0  United States
          1269   January   2229.0  United States
          1366   January    719.0  United States
          1693   January   2042.0  United States
          ...        ...      ...            ...
          33994  January   2600.0         France
          34640  January   1280.0         France
          34669  January   2558.0         France
          34674  January   1405.0         France
          34675  January   2202.0         France

          [138 rows x 3 columns]

In [121]: january_month_2015['Revenue'].sum()

Out[121]: 230549.0
```

By running the above query we get the total Revenue of January month. In the same way we can calculate for the rest of the months and at the last club all the month wise revenue and store it in a list form so that we can use it for the graph representation. (*find the image below*)

Storing all the monthly sales revenue in list.



```
In [147]: Monthly_sales_2015 = [230549.0,259857.0,250358.0,284143.0,320629.0,320629.0,789054.0,1248185.0,1295246.0,1376969.0,1438928.0,2116
```



2015 Monthly sales

```
In [148]: plt.figure(figsize=(10,5))
          xaxis= ['Jan','Feb','Mar','Apr','May','June','Jul','Aug','Sept','Oct','Nov','Dec']
          yaxis = [230549.0,259857.0,250358.0,284143.0,320629.0,320629.0,789054.0,1248185.0,1295246.0,1376969.0,1438928.0,2116097.0]
          plt.barh(xaxis,yaxis)
          plt.title('2015 Monthly Sales')
          plt.show()
```

As we can see in the above image, the monthly revenue for the year 2015.

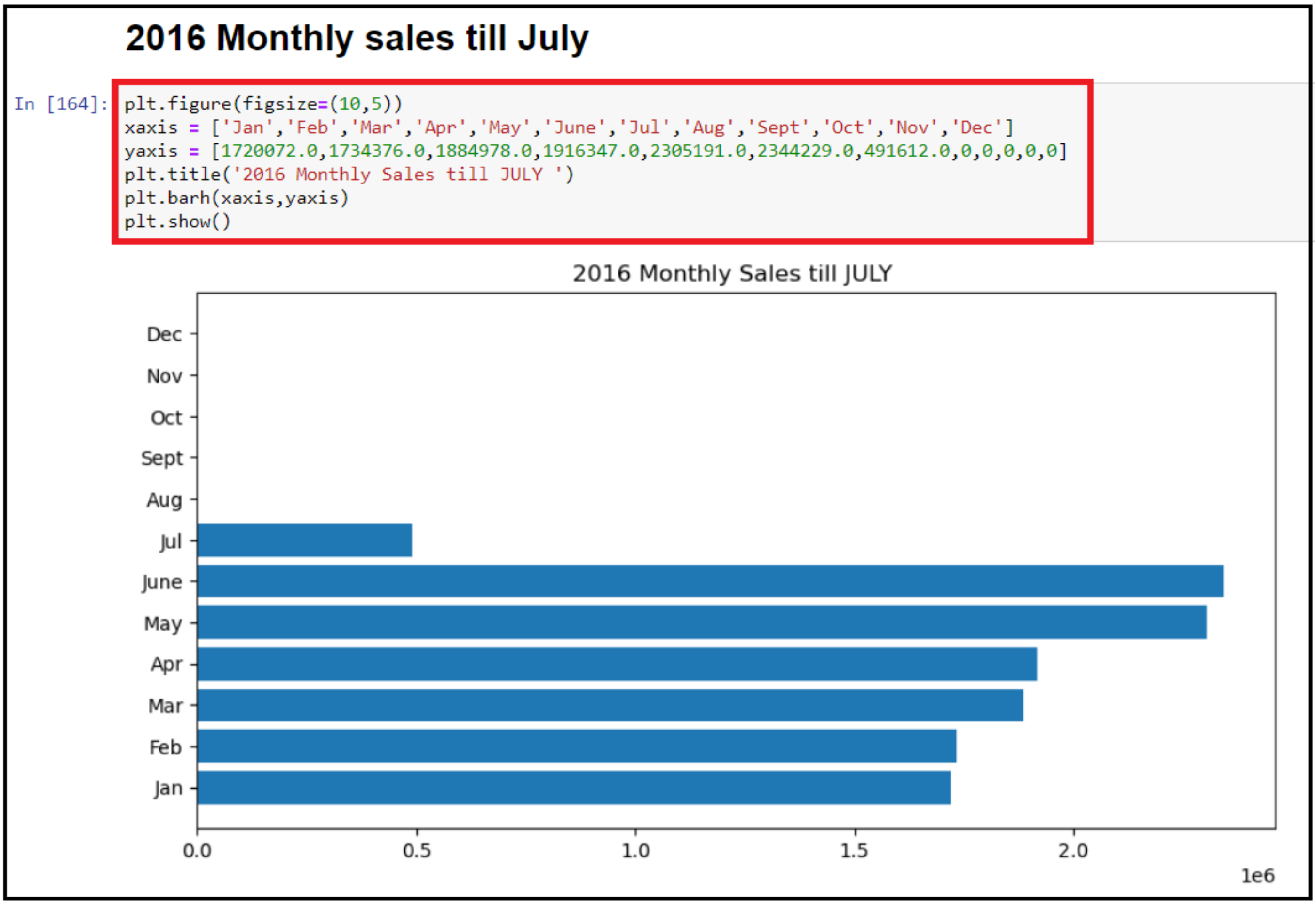
The sales of the store was at the lower side till the monthly of June, there is a good spike in sales of the store from the month of July and it continues till the year end. This shows that the store was cleared its low sales months at the end the for 4th Quarter.

The month of December,2015 had a good sales considering the Christmas season and New year Celebration. The store should plan good offer to attract more consumers to the stores during the festival times as it turn out to be a good revenue generating events.

As we can see in the below image, the monthly revenue for the year 2015.

Considering the previous month December sales the January sales has been on the lower side.

From the month of March the sales is back on track, hoping the coming months will see more growth in the sales. As per the available data the highest revenue collection was made in the month of May&June 2016.

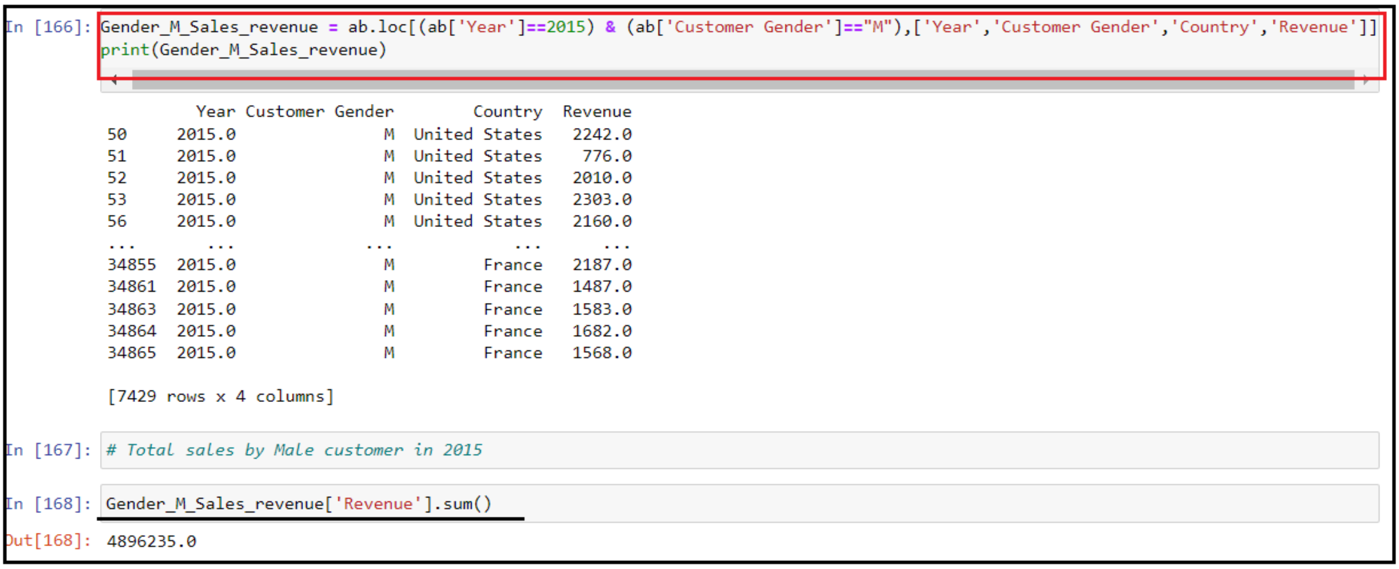


2016 Monthly sales till July

```
In [164]: plt.figure(figsize=(10,5))
          xaxis = ['Jan','Feb','Mar','Apr','May','June','Jul','Aug','Sept','Oct','Nov','Dec']
          yaxis = [1720072.0,1734376.0,1884978.0,1916347.0,2305191.0,2344229.0,491612.0,0,0,0,0,0]
          plt.title('2016 Monthly Sales till JULY ')
          plt.barh(xaxis,yaxis)
          plt.show()
```

Show the Sales report by gender 2015 & 2016

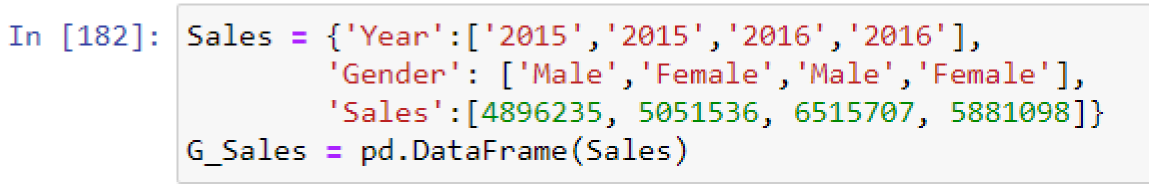
To extract the data from the dataframe, first we need to calculate the 2015 Sales by gender Male & Female. (*find the image below*)



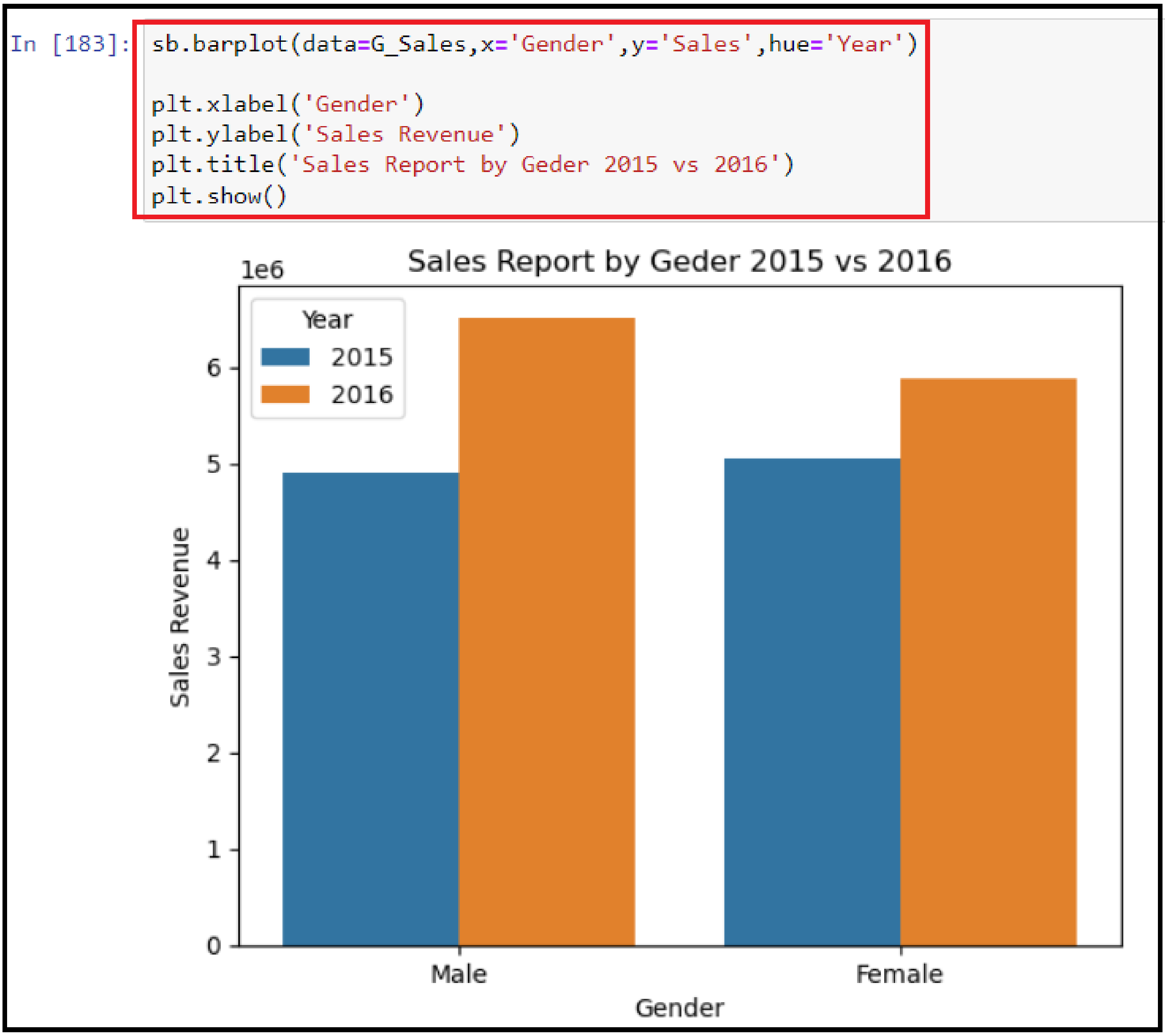
```
In [166]: Gender_M_Sales_revenue = ab.loc[(ab['Year']==2015) & (ab['Customer Gender']=="M"),['Year','Customer Gender','Country','Revenue']]
          print(Gender_M_Sales_revenue)

             Year Customer Gender        Country  Revenue
50     2015.0               M  United States   2242.0
51     2015.0               M  United States    776.0
52     2015.0               M  United States   2010.0
53     2015.0               M  United States   2303.0
56     2015.0               M  United States   2160.0
...       ...             ...            ...      ...
34855  2015.0               M         France   2187.0
34861  2015.0               M         France   1487.0
34863  2015.0               M         France   1583.0
34864  2015.0               M         France   1682.0
34865  2015.0               M         France   1568.0

[7429 rows x 4 columns]

In [167]: # Total sales by Male customer in 2015

In [168]: Gender_M_Sales_revenue['Revenue'].sum()

Out[168]: 4896235.0
```

Using the loc function we calculated the total sales for year 2015 by gender Male. The total sales revenue generated by Gender Male is $4896235.0.

In the same way we can calculate for Gender female as well as for the Year 2016.



```
In [182]: Sales = {'Year':['2015','2015','2016','2016'],
                   'Gender': ['Male','Female','Male','Female'],
                   'Sales':[4896235, 5051536, 6515707, 5881098]}
          G_Sales = pd.DataFrame(Sales)
```



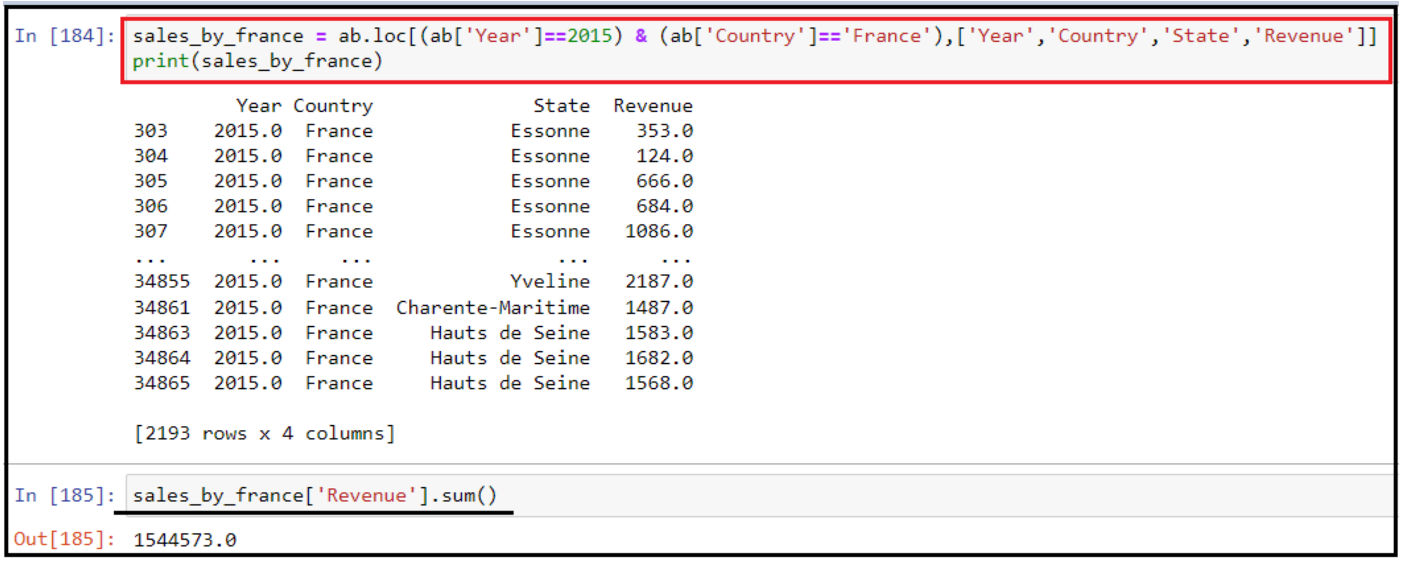
```
In [183]: sb.barplot(data=G_Sales,x='Gender',y='Sales',hue='Year')

          plt.xlabel('Gender')
          plt.ylabel('Sales Revenue')
          plt.title('Sales Report by Geder 2015 vs 2016')
          plt.show()
```

Here we can see, The sales revenue for the year 2015 is on a higher side by gender “Female”.
For the year 2016, the sales revenue is on a higher side by the gender “Male”.

To plot the above graph we have calculated the sales revenue Year and Gender wise and club them together forming a small dataframe and present the data into bar graph.

Show the country wise sales Revenue 2015 vs 2016

- *Calculation of Revenue for 2015, country = France (FR)*

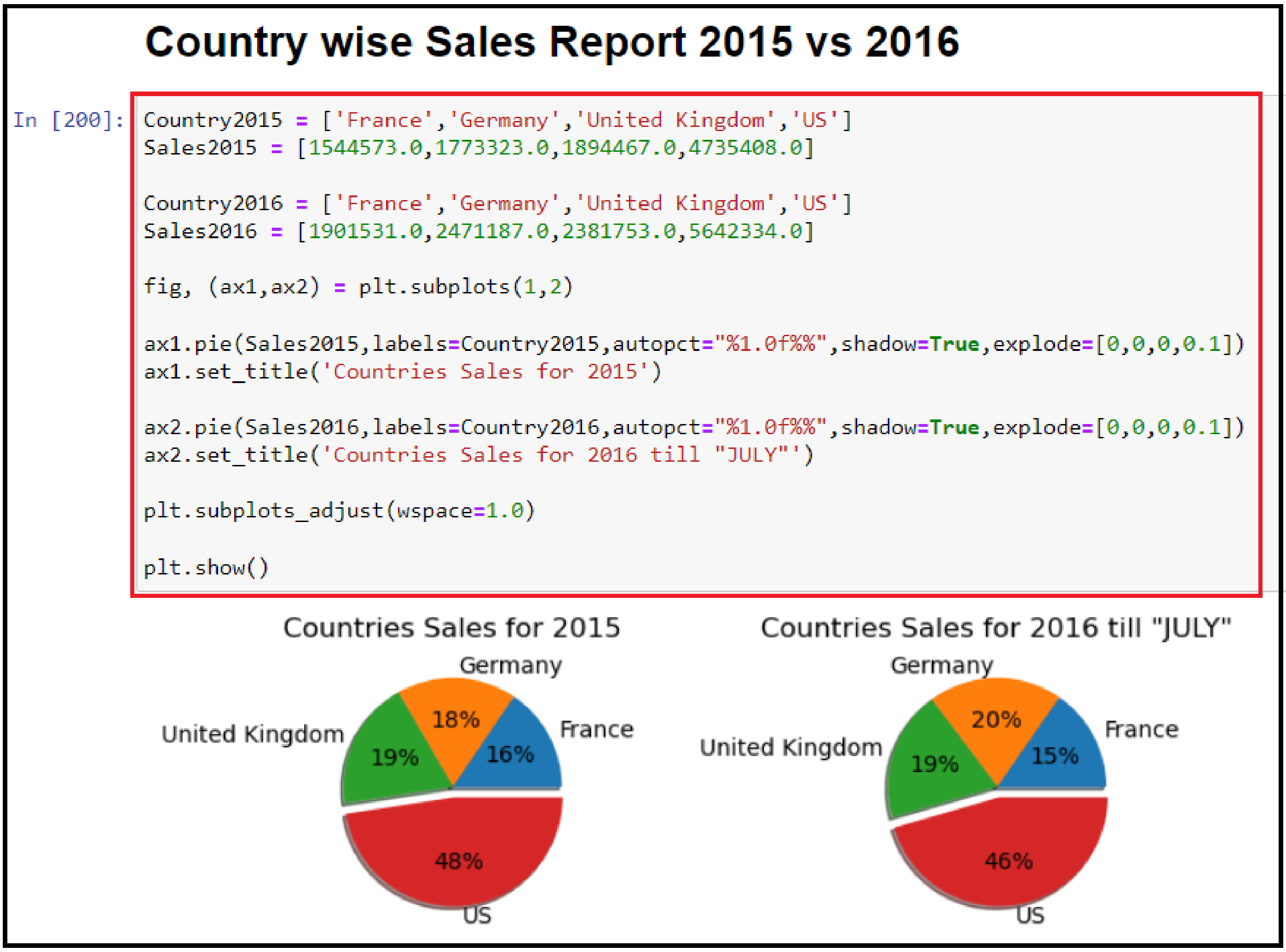


```
In [184]: sales_by_france = ab.loc[(ab['Year']==2015) & (ab['Country']=='France'),['Year','Country','State','Revenue']]
          print(sales_by_france)

                 Year Country              State  Revenue
          303    2015.0  France            Essonne    353.0
          304    2015.0  France            Essonne    124.0
          305    2015.0  France            Essonne    666.0
          306    2015.0  France            Essonne    684.0
          307    2015.0  France            Essonne   1086.0
          ...       ...     ...                ...      ...
          34855  2015.0  France             Yveline   2187.0
          34861  2015.0  France  Charente-Maritime   1487.0
          34863  2015.0  France      Hauts de Seine   1583.0
          34864  2015.0  France      Hauts de Seine   1682.0
          34865  2015.0  France      Hauts de Seine   1568.0

          [2193 rows x 4 columns]

In [185]: sales_by_france['Revenue'].sum()

Out[185]: 1544573.0
```

Using the loc function and the condition Year == 2015 & Country == “France”.

We will get the revenue for the year 2015 from the France and using the sum function we will get the total revenue. By using the same syntax and altering the name of the country and year you will get the required data from the dataframe.

Lets represent the Sales revenue group by countries.



Country wise Sales Report 2015 vs 2016

```
In [200]: Country2015 = ['France','Germany','United Kingdom','US']
          Sales2015 = [1544573.0,1773323.0,1894467.0,4735408.0]

          Country2016 = ['France','Germany','United Kingdom','US']
          Sales2016 = [1901531.0,2471187.0,2381753.0,5642334.0]

          fig, (ax1,ax2) = plt.subplots(1,2)

          ax1.pie(Sales2015,labels=Country2015,autopct="%1.0f%%",shadow=True,explode=[0,0,0,0.1])
          ax1.set_title('Countries Sales for 2015')

          ax2.pie(Sales2016,labels=Country2016,autopct="%1.0f%%",shadow=True,explode=[0,0,0,0.1])
          ax2.set_title('Countries Sales for 2016 till "JULY"')

          plt.subplots_adjust(wspace=1.0)

          plt.show()
```

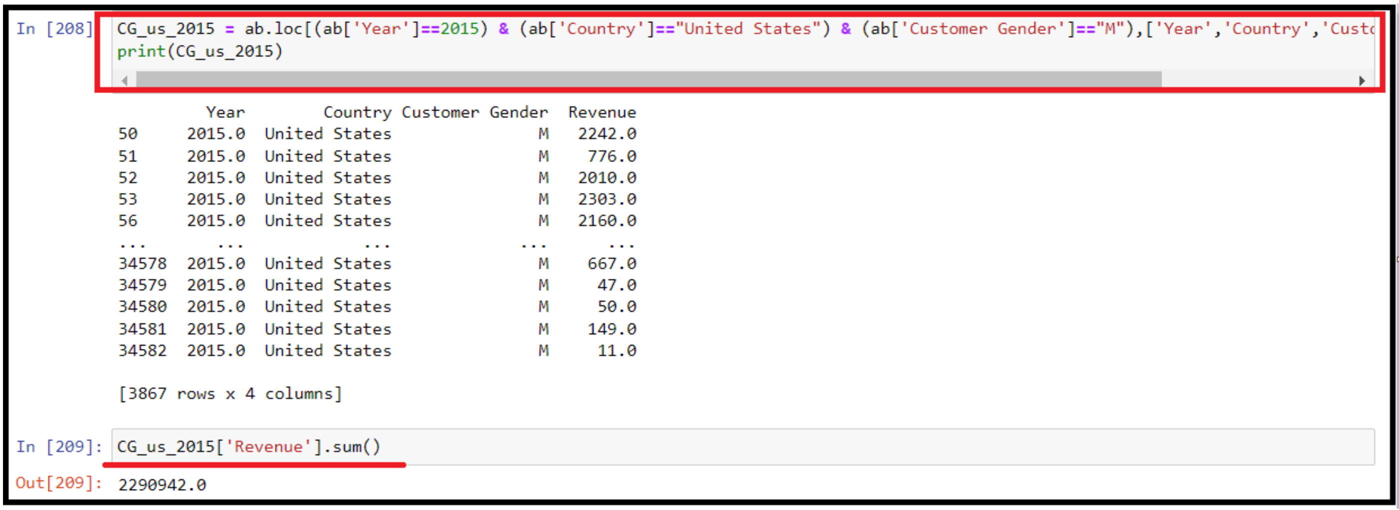
Here we create 2 pie chart in a single frame using subplots.

Left side = The first pie chart show the sales revenue for the year 2015. The country of USA has collected 48% revenue compare to the France,Germany and UK.

Right side = The second pie chart shows the sales revenue for the year 2016. The country USA has collected 46% revenue compare to the other countries. Still the store have a change to enhance the sales revenue as there are few more months still to perform.

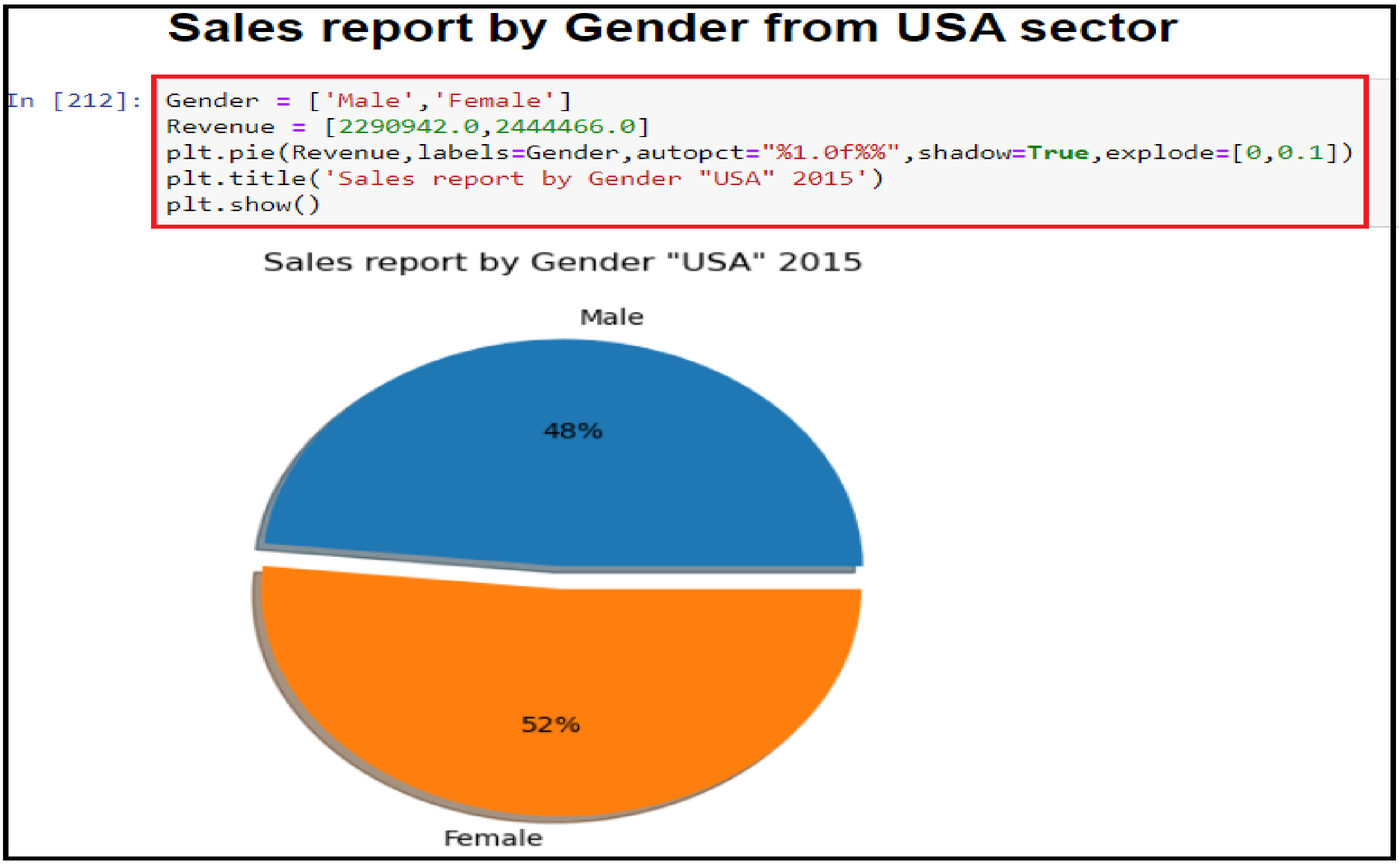
Show the USA Sales for the year 2015 group by gender.

- *Calculate the sales revenue by Male :-*



```
In [208]: CG_us_2015 = ab.loc[(ab['Year']==2015) & (ab['Country']=="United States") & (ab['Customer Gender']=="M"),['Year','Country','Cust
          print(CG_us_2015)

                 Year        Country Customer Gender  Revenue
          50     2015.0  United States               M   2242.0
          51     2015.0  United States               M    776.0
          52     2015.0  United States               M   2010.0
          53     2015.0  United States               M   2303.0
          56     2015.0  United States               M   2160.0
          ...       ...            ...             ...      ...
          34578  2015.0  United States               M    667.0
          34579  2015.0  United States               M     47.0
          34580  2015.0  United States               M     50.0
          34581  2015.0  United States               M    149.0
          34582  2015.0  United States               M     11.0

          [3867 rows x 4 columns]

In [209]: CG_us_2015['Revenue'].sum()

Out[209]: 2290942.0
```

Using loc function we calculated the sales revenue collected by USA giving the condition for Gender “Male”. We store the output to a variable (CG_us_2015) and use the sum function to calculate the total revenue. In the same way we can calculate the revenue for Female and store it in a list.

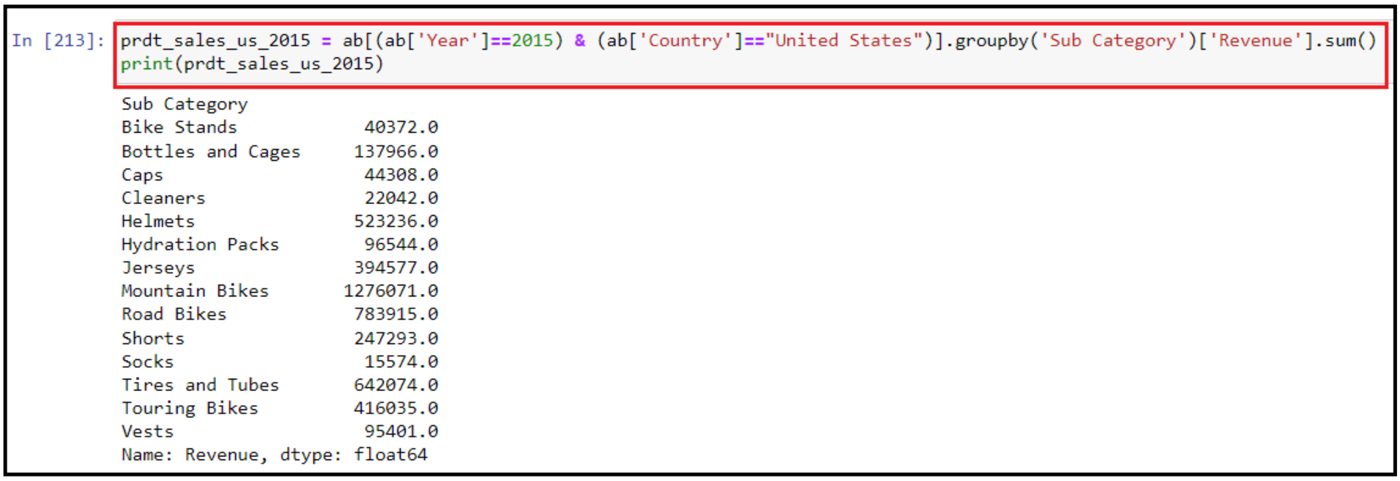


Sales report by Gender from USA sector

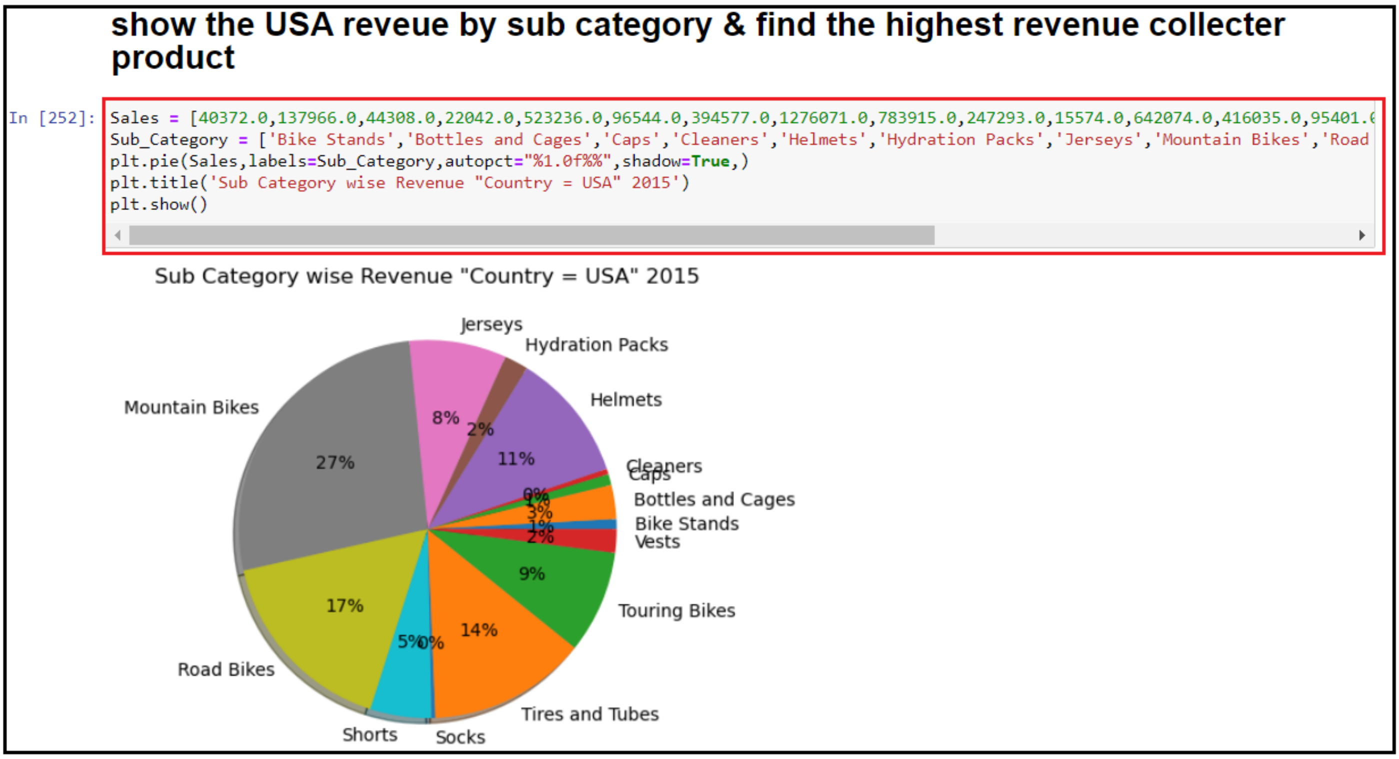
```
In [212]: Gender = ['Male','Female']
          Revenue = [2290942.0,2444466.0]
          plt.pie(Revenue,labels=Gender,autopct="%1.0f%%",shadow=True,explode=[0,0.1])
          plt.title('Sales report by Gender "USA" 2015')
          plt.show()
```

Show the USA sales Revenue group by sub category & Highest revenue generating product.

- *Calculate the revenue by Sub-Category.*



```
In [213]: prdt_sales_us_2015 = ab[(ab['Year']==2015) & (ab['Country']=="United States")].groupby('Sub Category')['Revenue'].sum()
          print(prdt_sales_us_2015)

          Sub Category
          Bike Stands           40372.0
          Bottles and Cages    137966.0
          Caps                  44308.0
          Cleaners              22042.0
          Helmets              523236.0
          Hydration Packs       96544.0
          Jerseys              394577.0
          Mountain Bikes      1276071.0
          Road Bikes           783915.0
          Shorts               247293.0
          Socks                 15574.0
          Tires and Tubes      642074.0
          Touring Bikes        416035.0
          Vests                 95401.0
          Name: Revenue, dtype: float64
```



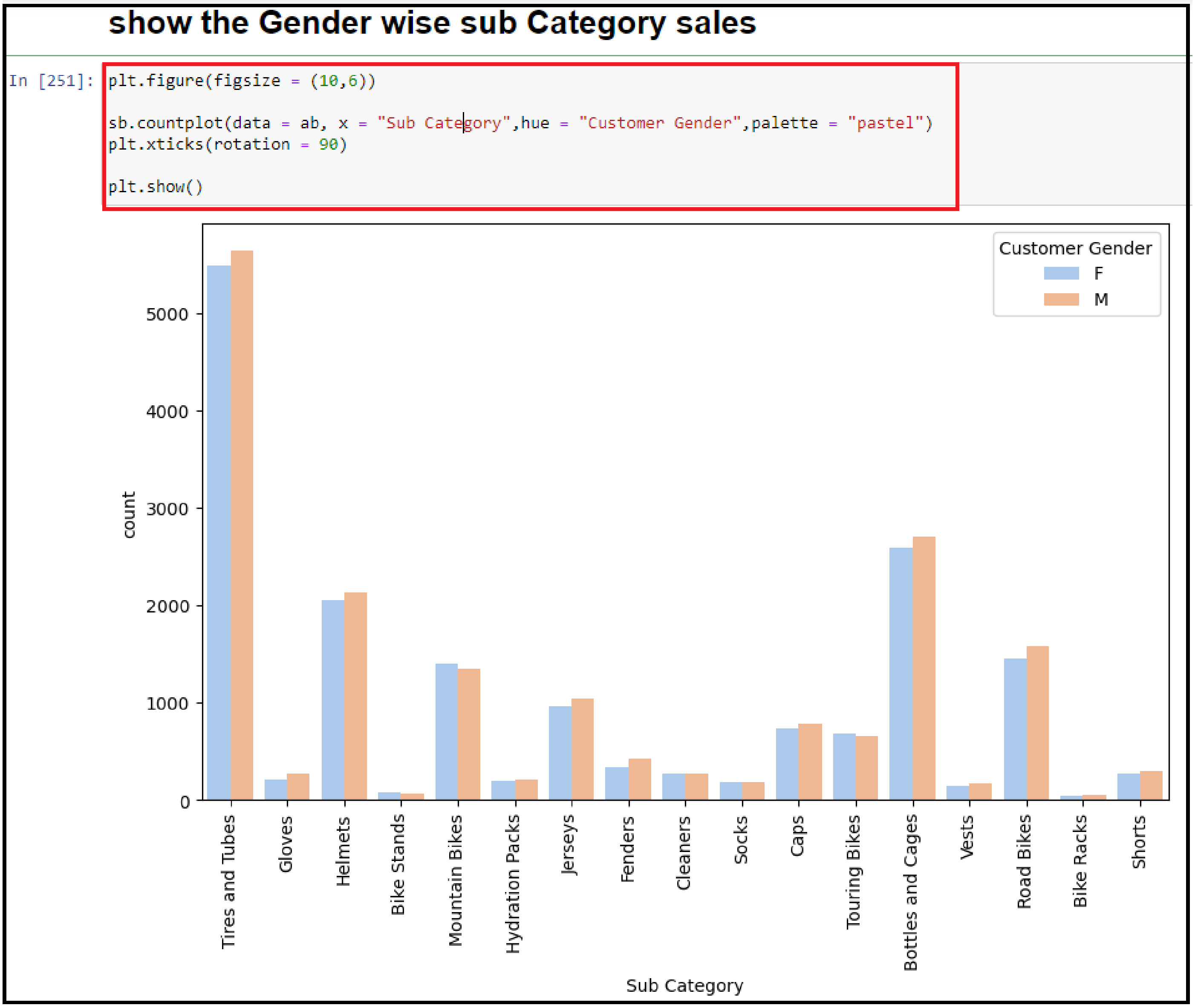
show the USA reveue by sub category & find the highest revenue collecter product

```
In [252]: Sales = [40372.0,137966.0,44308.0,22042.0,523236.0,96544.0,394577.0,1276071.0,783915.0,247293.0,15574.0,642074.0,416035.0,95401.
          Sub_Category = ['Bike Stands','Bottles and Cages','Caps','Cleaners','Helmets','Hydration Packs','Jerseys','Mountain Bikes','Road
          plt.pie(Sales,labels=Sub_Category,autopct="%1.0f%%",shadow=True,)
          plt.title('Sub Category wise Revenue "Country = USA" 2015')
          plt.show()
```

Sub Category wise Revenue "Country = USA" 2015

The above calculated revenue by sub-category and the name of the products was stored in a list to showcase in a pie chart. As we can see the above chart the highest revenue generating product is “Mountain Bike ” with 27% of the total sales revenue followed by “Road Bikes” with 17% of the total revenue.

Show the Gender wise sales for sub-category



show the Gender wise sub Category sales

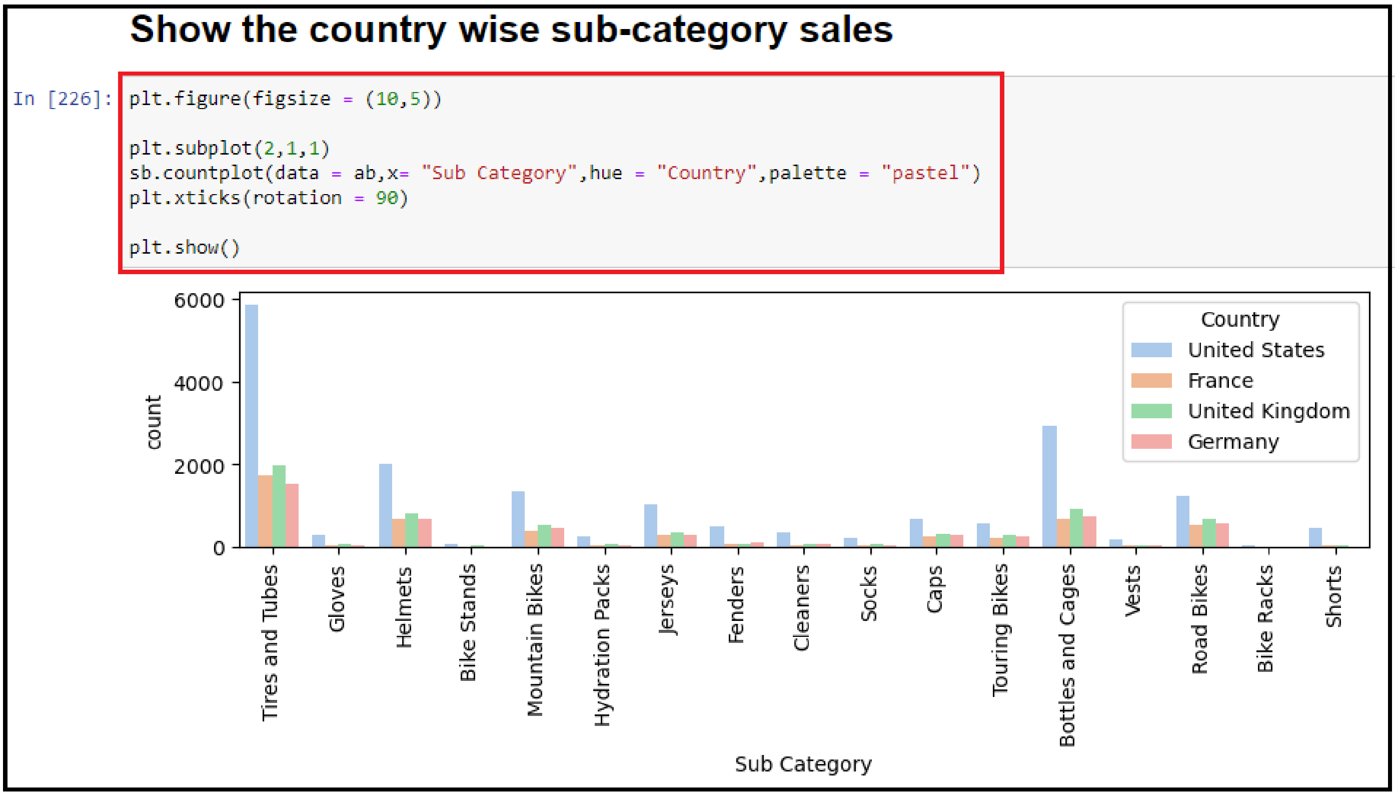
In [251]:

```
plt.figure(figsize = (10,6))

sb.countplot(data = ab, x = "Sub Category",hue = "Customer Gender",palette = "pastel")
plt.xticks(rotation = 90)

plt.show()
```

Using seaborn library and the countplot(sb.countplot) we create gender wise sales count for sub-category.

In the above graph, the product Tires and Tubes is the most selling product (count wise). The customer gender Male and Female both have purchased the item “Tires and Tubes” with maximum quantity.

Show the country wise sales, group by sub-category.

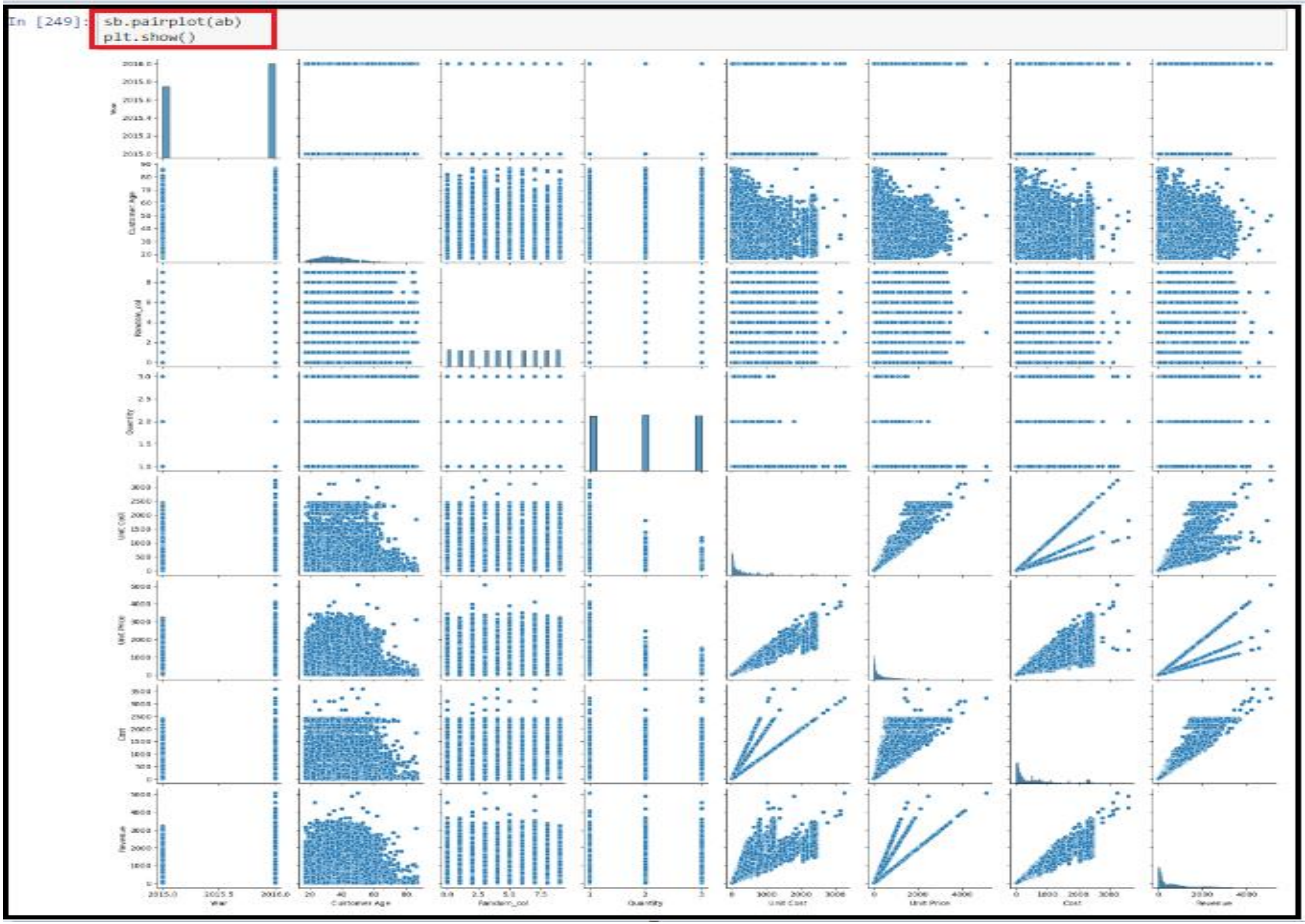


Show the country wise sub-category sales

```
In [226]: plt.figure(figsize = (10,5))

          plt.subplot(2,1,1)
          sb.countplot(data = ab,x= "Sub Category",hue = "Country",palette = "pastel")
          plt.xticks(rotation = 90)

          plt.show()
```

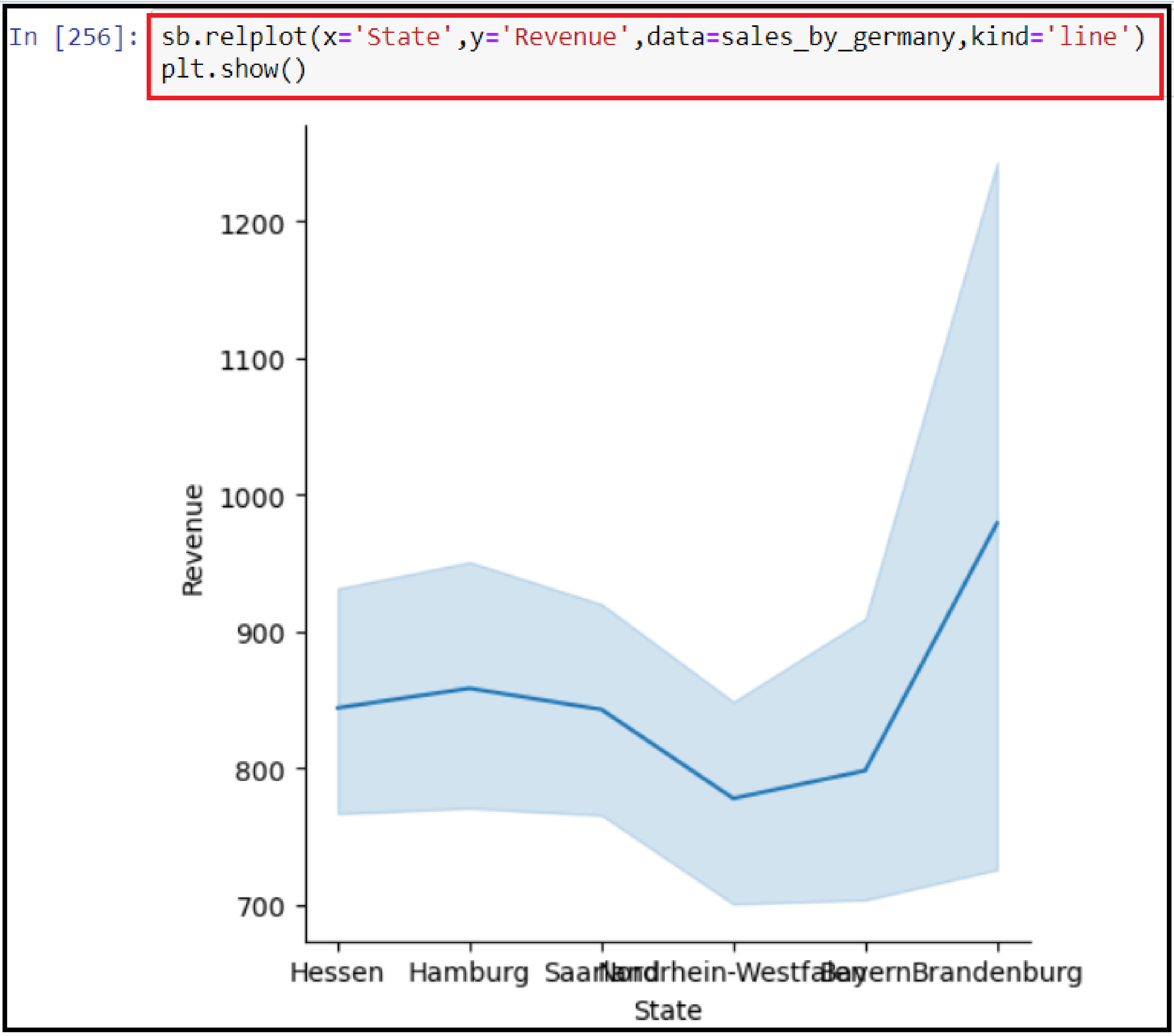
In the above chart, we can see the sales graph country wise. In this chart the country USA is Dominating the sales compare to other countries. Being store is located in USA itself, the store has a good sales within USA.

- ***Pair-plot:-***



```
In [249]: sb.pairplot(ab)
          plt.show()
```

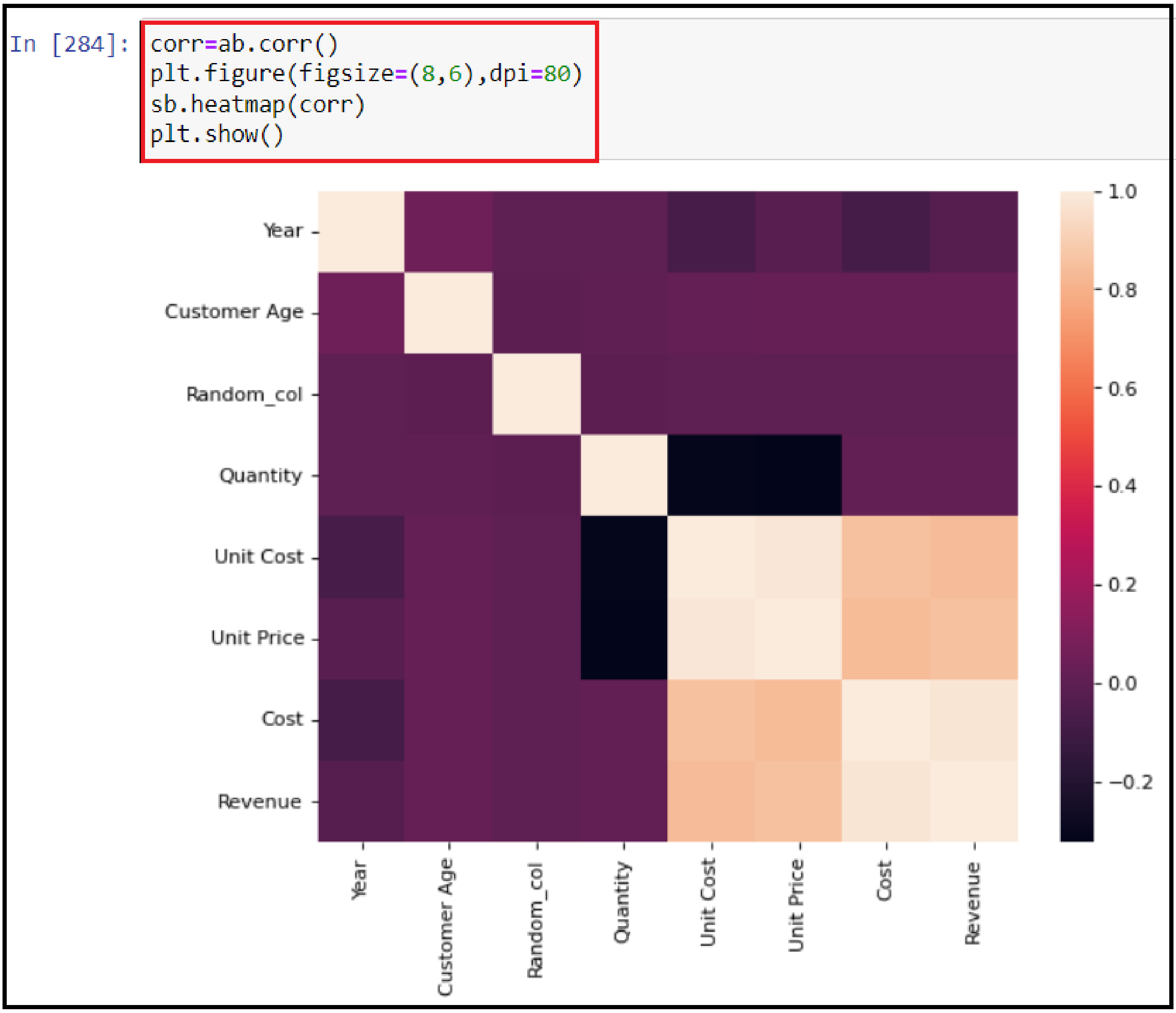
Show the Sales revenue of Germany



In [256]:
```
sb.relplot(x='State',y='Revenue',data=sales_by_germany,kind='line')
plt.show()
```

The above chart showcase the Sales revenue of the country of Germany. The above revenue was collected from the 6 provinces from Germany i.e. Hessen,Hamburg,Saarland,Nordrh-Westfalen,Bayern & Brandenburg.

Heatmap:-



```
In [284]: corr=ab.corr()
          plt.figure(figsize=(8,6),dpi=80)
          sb.heatmap(corr)
          plt.show()
```