

# Data visualization in Python

BY BHAGYASHREE CHAVAN

# Goal and Task



'Amazing Mart' data set with 15 European countries contain 4 year that is 2011 to 2014 of information.

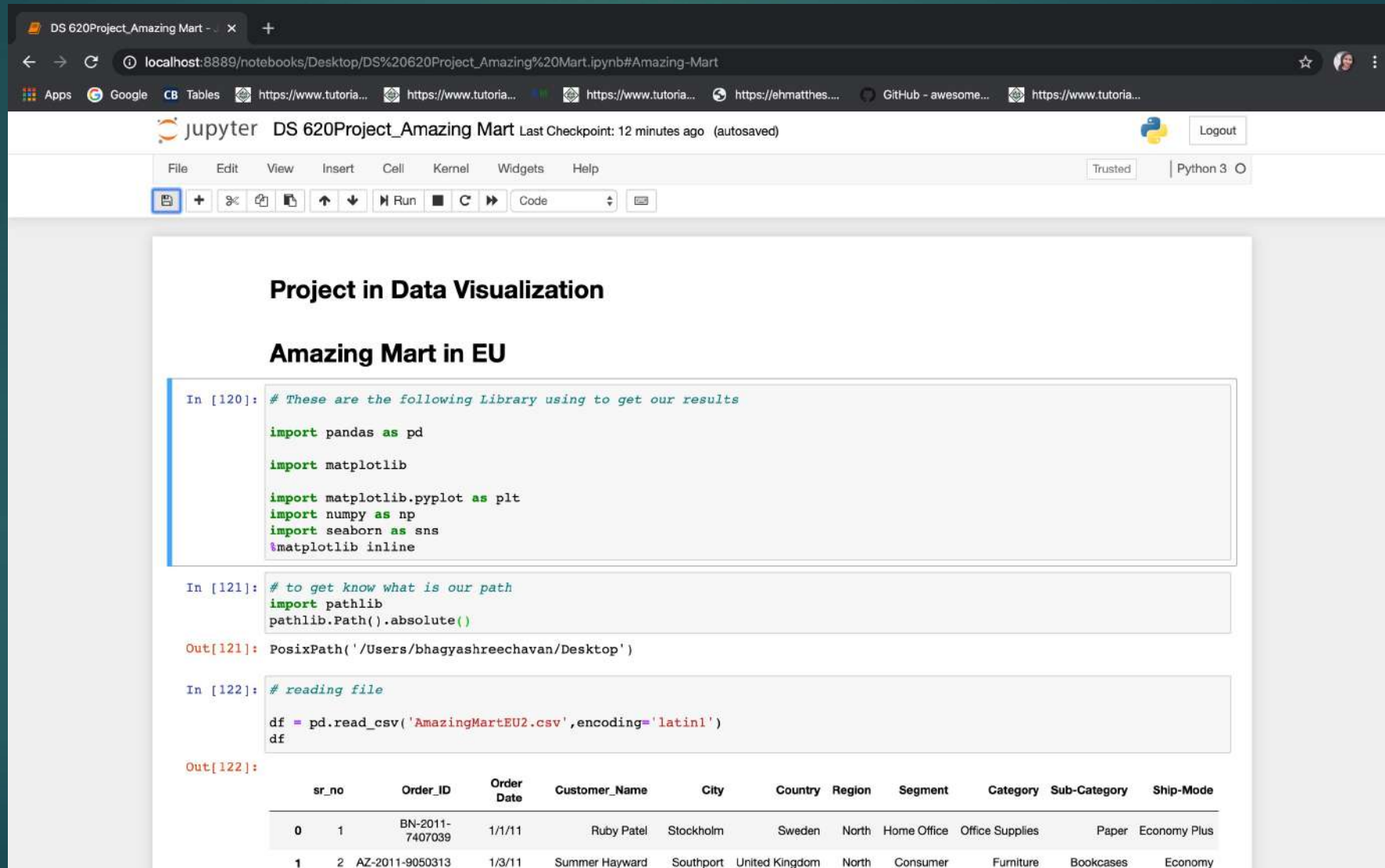


Analysis of this data set with total count of order place in year.



Month wise how many order has been placed in specific year?

# Library Use



The screenshot shows a Jupyter Notebook titled "DS 620Project\_Amazing Mart" with a last checkpoint of 12 minutes ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The notebook content is as follows:

## Project in Data Visualization

### Amazing Mart in EU

```
In [120]: # These are the following Library using to get our results

import pandas as pd

import matplotlib

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
```

```
In [121]: # to get know what is our path
import pathlib
pathlib.Path().absolute()
```

```
Out[121]: PosixPath('/Users/bhagyashreechavan/Desktop')
```

```
In [122]: # reading file

df = pd.read_csv('AmazingMartEU2.csv',encoding='latin1')
df
```

```
Out[122]:
```

	sr_no	Order_ID	Order Date	Customer_Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode
0	1	BN-2011-7407039	1/1/11	Ruby Patel	Stockholm	Sweden	North	Home Office	Office Supplies	Paper	Economy Plus
1	2	AZ-2011-9050313	1/3/11	Summer Hayward	Southport	United Kingdom	North	Consumer	Furniture	Bookcases	Economy

# Reading File

DS 620Project\_Amazing Mart - x

localhost:8889/notebooks/Desktop/DS%20620Project\_Amazing%20Mart.ipynb#Amazing-Mart

Apps Google CB Tables https://www.tutoria... https://www.tutoria... https://www.tutoria... https://ehmatthes.... GitHub - awesome... https://www.tutoria...

jupyter DS 620Project\_Amazing Mart Last Checkpoint: 13 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [122]: `# reading file`

```
df = pd.read_csv('AmazingMartEU2.csv', encoding='latin1')
df
```

Out[122]:

	sr_no	Order_ID	Order Date	Customer Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode
0	1	BN-2011-7407039	1/1/11	Ruby Patel	Stockholm	Sweden	North	Home Office	Office Supplies	Paper	Economy Plus
1	2	AZ-2011-9050313	1/3/11	Summer Hayward	Southport	United Kingdom	North	Consumer	Furniture	Bookcases	Economy
2	3	AZ-2011-6674300	1/4/11	Devin Huddleston	Valence	France	Central	Consumer	Office Supplies	Art	Economy
3	4	BN-2011-2819714	1/4/11	Mary Parker	Birmingham	United Kingdom	North	Corporate	Office Supplies	Art	Economy
4	5	AZ-2011-617423	1/5/11	Daniel Burke	Echirolles	France	Central	Home Office	Office Supplies	Storage	Priority
...	...	...	...	...	...	...	...	...	...	...	...
4112	4113	AZ-2014-8174835	12/31/14	Eloise Sykes	Bielefeld	Germany	Central	Consumer	Office Supplies	Paper	Economy
4113	4114	AZ-2014-766953	12/31/14	Jose Gambino	Maidenhead	United Kingdom	North	Corporate	Office Supplies	Art	Economy
4114	4115	AZ-2014-1412225	12/31/14	Leon Barnes	Worcester	United Kingdom	North	Consumer	Office Supplies	Art	Priority
4115	4116	AZ-2014-7604524	12/31/14	Rebecca Chamberlain	Hamburg	Germany	Central	Home Office	Office Supplies	Supplies	Economy
4116	4117	BN-2014-4140795	12/31/14	Daniel Hamilton	Eindhoven	Netherlands	Central	Home Office	Technology	Phones	Economy Plus

4117 rows x 11 columns

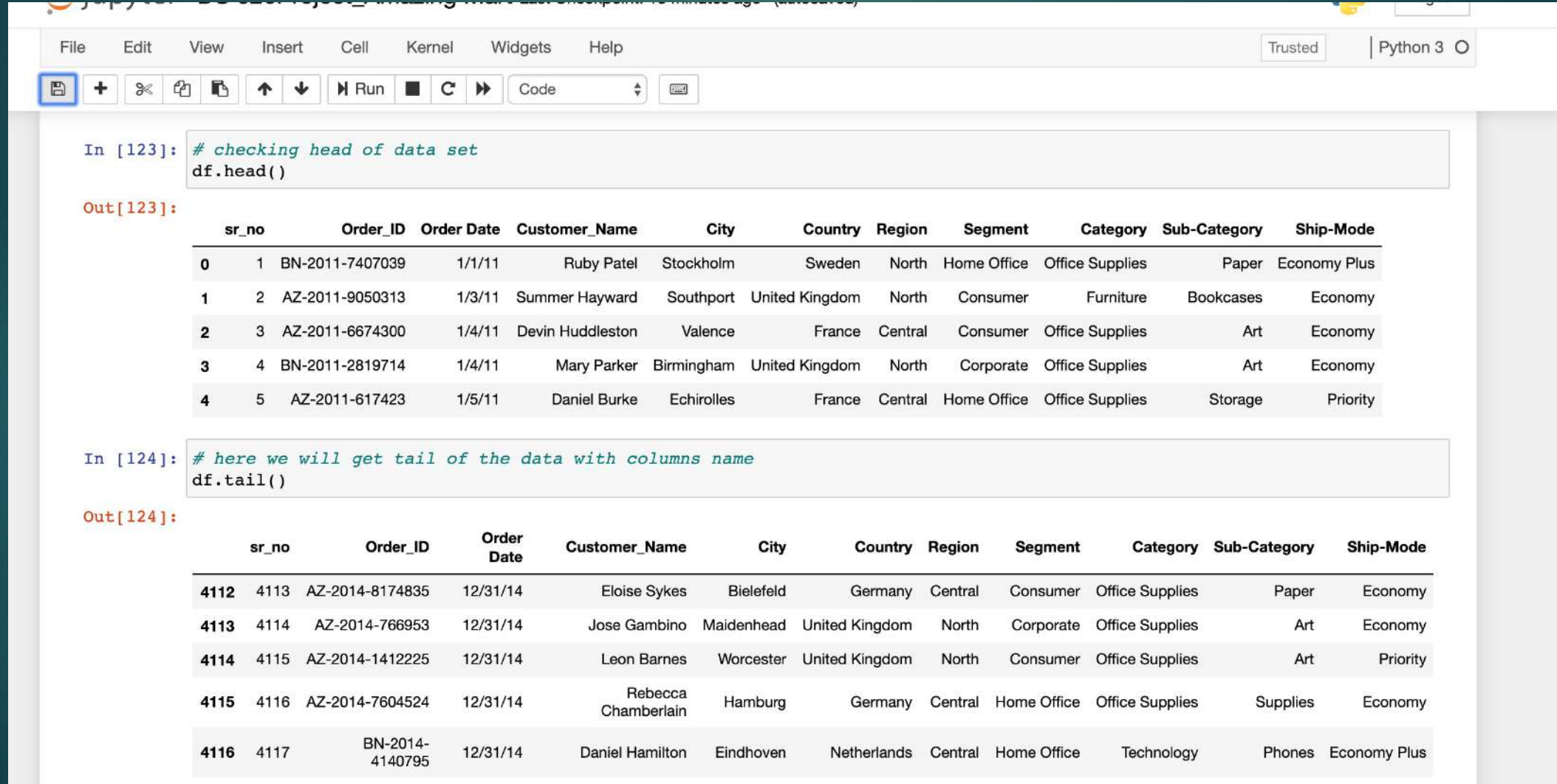
In [123]: `# checking head of data set`

```
df.head()
```

Out[123]:

	sr_no	Order_ID	Order Date	Customer Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode
0	1	BN-2011-7407039	1/1/11	Ruby Patel	Stockholm	Sweden	North	Home Office	Office Supplies	Paper	Economy Plus

# Understanding of Data



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding, deleting, and running code. The notebook is running Python 3. The code in the first cell checks the head of a DataFrame, and the second cell checks the tail. Both outputs are displayed as tables.

```
In [123]: # checking head of data set
df.head()
```

Out[123]:

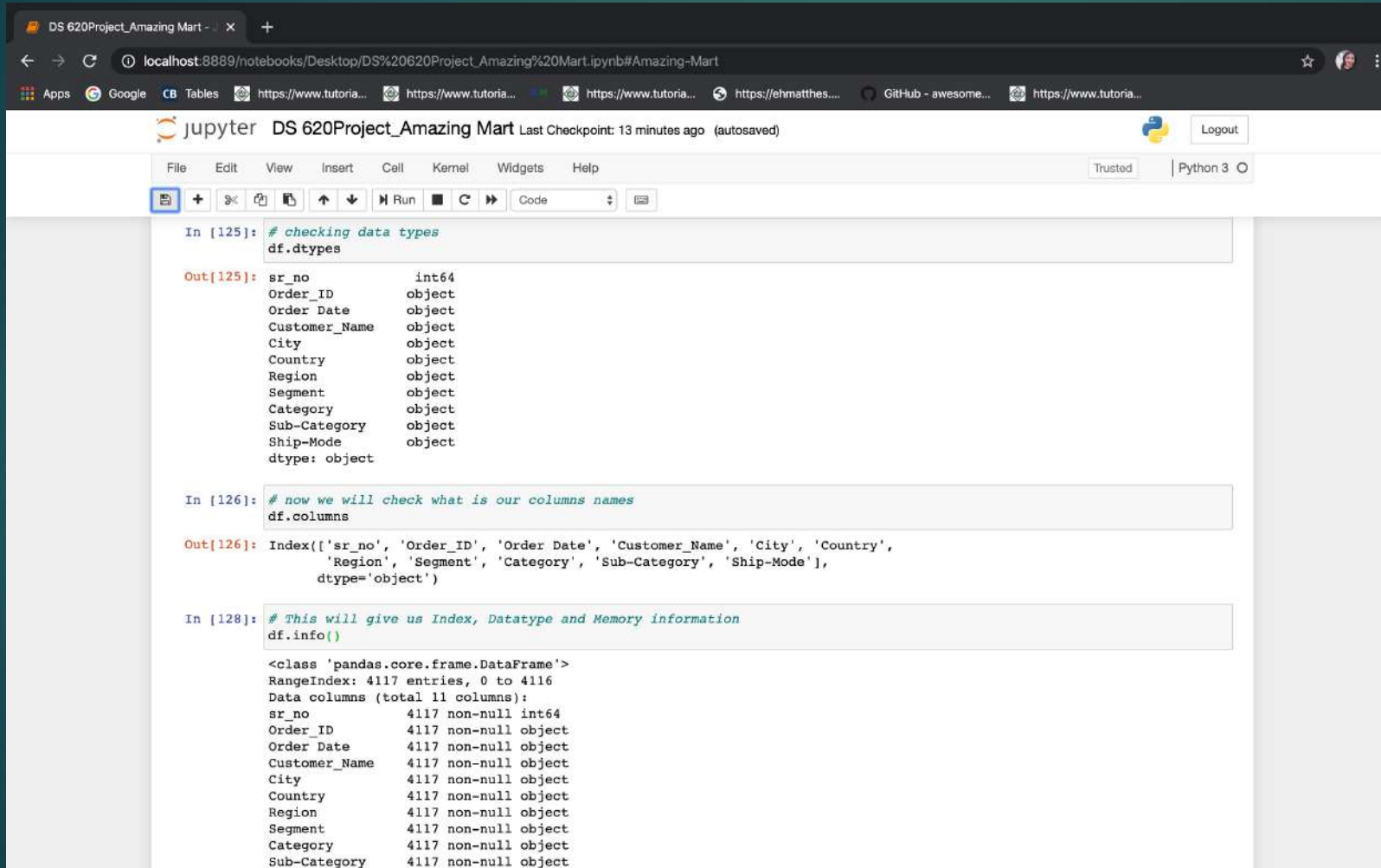
	sr_no	Order_ID	Order Date	Customer_Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode
0	1	BN-2011-7407039	1/1/11	Ruby Patel	Stockholm	Sweden	North	Home Office	Office Supplies	Paper	Economy Plus
1	2	AZ-2011-9050313	1/3/11	Summer Hayward	Southport	United Kingdom	North	Consumer	Furniture	Bookcases	Economy
2	3	AZ-2011-6674300	1/4/11	Devin Huddleston	Valence	France	Central	Consumer	Office Supplies	Art	Economy
3	4	BN-2011-2819714	1/4/11	Mary Parker	Birmingham	United Kingdom	North	Corporate	Office Supplies	Art	Economy
4	5	AZ-2011-617423	1/5/11	Daniel Burke	Echirolles	France	Central	Home Office	Office Supplies	Storage	Priority

```
In [124]: # here we will get tail of the data with columns name
df.tail()
```

Out[124]:

	sr_no	Order_ID	Order Date	Customer_Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode
4112	4113	AZ-2014-8174835	12/31/14	Eloise Sykes	Bielefeld	Germany	Central	Consumer	Office Supplies	Paper	Economy
4113	4114	AZ-2014-766953	12/31/14	Jose Gambino	Maidenhead	United Kingdom	North	Corporate	Office Supplies	Art	Economy
4114	4115	AZ-2014-1412225	12/31/14	Leon Barnes	Worcester	United Kingdom	North	Consumer	Office Supplies	Art	Priority
4115	4116	AZ-2014-7604524	12/31/14	Rebecca Chamberlain	Hamburg	Germany	Central	Home Office	Office Supplies	Supplies	Economy
4116	4117	BN-2014-4140795	12/31/14	Daniel Hamilton	Eindhoven	Netherlands	Central	Home Office	Technology	Phones	Economy Plus

# Understanding of Data



The screenshot shows a Jupyter Notebook titled "DS 620Project\_Amazing Mart" running on a local host. The notebook contains three code cells. The first cell checks the data types of the columns in a DataFrame. The second cell checks the column names. The third cell provides a summary of the DataFrame, including the number of entries and the data types of the columns.

```
In [125]: # checking data types
df.dtypes

Out[125]: sr_no          int64
Order_ID         object
Order Date       object
Customer_Name    object
City             object
Country          object
Region           object
Segment          object
Category         object
Sub-Category     object
Ship-Mode        object
dtype: object

In [126]: # now we will check what is our columns names
df.columns

Out[126]: Index(['sr_no', 'Order_ID', 'Order Date', 'Customer_Name', 'City', 'Country',
               'Region', 'Segment', 'Category', 'Sub-Category', 'Ship-Mode'],
              dtype='object')

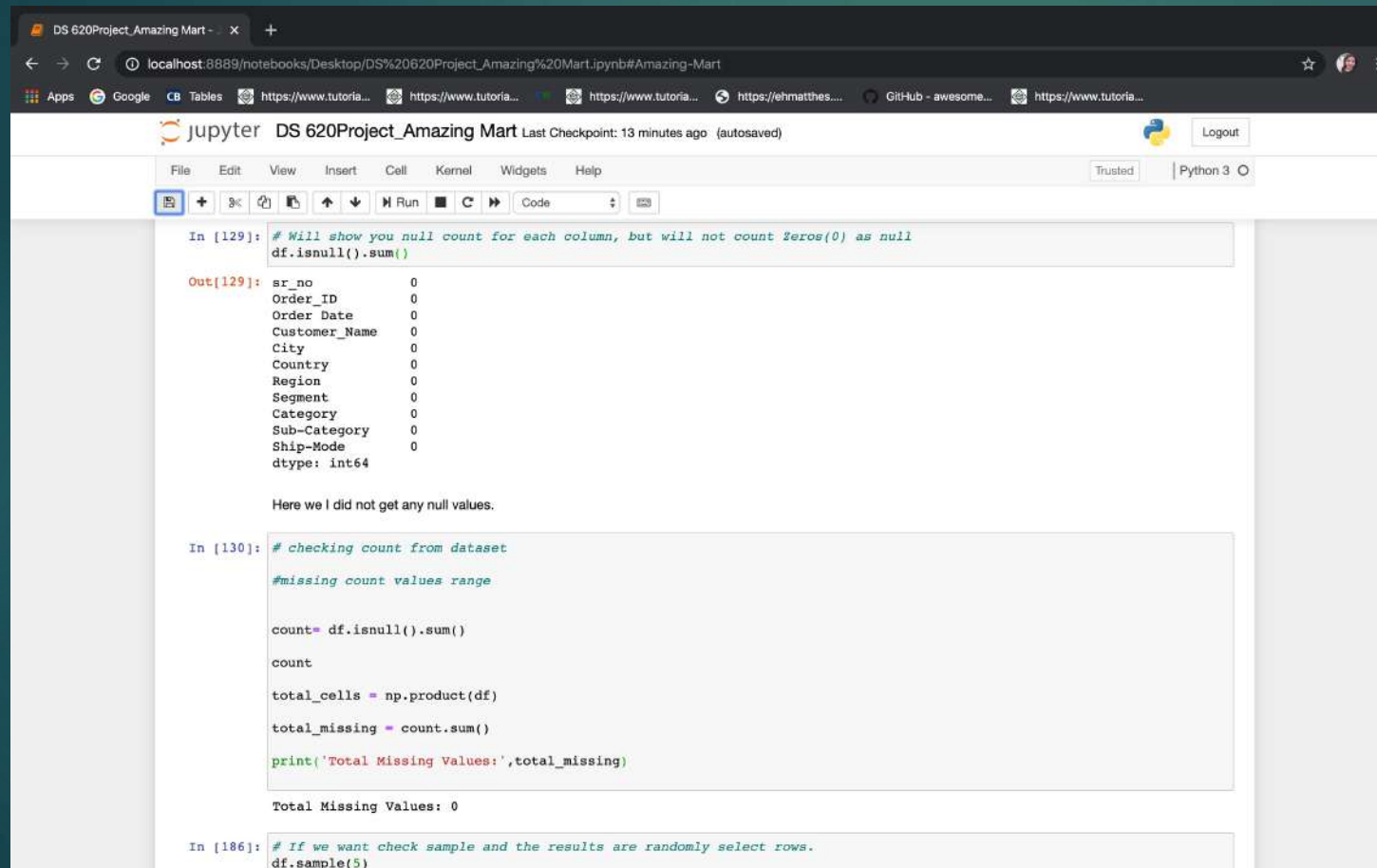
In [128]: # This will give us Index, Datatype and Memory information
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4117 entries, 0 to 4116
Data columns (total 11 columns):
sr_no          4117 non-null int64
Order_ID       4117 non-null object
Order Date     4117 non-null object
Customer_Name  4117 non-null object
City           4117 non-null object
Country        4117 non-null object
Region         4117 non-null object
Segment        4117 non-null object
Category       4117 non-null object
Sub-Category   4117 non-null object
```

- Understating data of type, columns name or features in dataset.
- Info() code represents information about dataset.



# Missing Values



The screenshot shows a Jupyter Notebook titled "DS 620Project\_Amazing Mart" with a last checkpoint of 13 minutes ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running code, and viewing output. The notebook contains three code cells:

```
In [129]: # Will show you null count for each column, but will not count Zeros(0) as null
df.isnull().sum()

Out[129]: sr_no      0
Order_ID    0
Order_Date  0
Customer_Name 0
City        0
Country     0
Region      0
Segment     0
Category    0
Sub-Category 0
Ship-Mode   0
dtype: int64

Here we did not get any null values.

In [130]: # checking count from dataset
#missing count values range

count = df.isnull().sum()

count

total_cells = np.product(df)
total_missing = count.sum()

print('Total Missing Values:', total_missing)

Total Missing Values: 0

In [186]: # If we want check sample and the results are randomly select rows.
df.sample(5)
```

In data set after doing analysis, there is no Missing value in data.

# Adding More Columns

```
DS 620Project_Amazing Mart - x +
localhost:8889/notebooks/Desktop/DS%20620Project_Amazing%20Mart.ipynb#Amazing-Mart
jupyter DS 620Project_Amazing Mart Last Checkpoint: 13 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [132]: #Slipping order date column
df['Year']=pd.to_datetime(df['Order Date']).apply(lambda x:x.year)

In [133]: #Slipping order date column
df['Month']=pd.to_datetime(df['Order Date']).apply(lambda x:x.month)

In [134]: #Slipping order date column
df['Day']=pd.to_datetime(df['Order Date']).apply(lambda x:x.day)

In [135]: # For checking result
df.head(20)

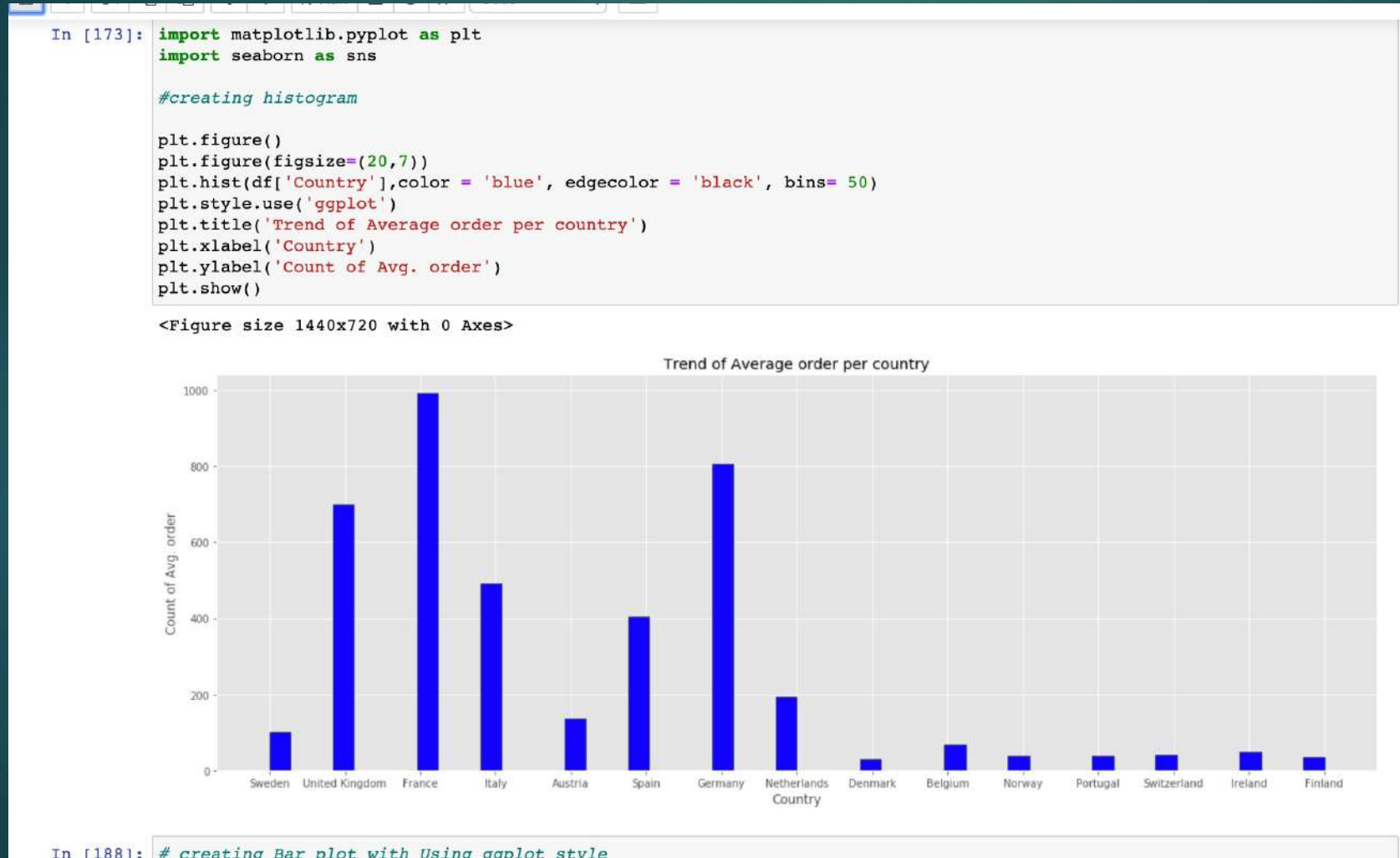
Out[135]:
```

	sr_no	Order_ID	Order Date	Customer_Name	City	Country	Region	Segment	Category	Sub-Category	Ship-Mode	Year	Month	Day
0	1	BN-2011-7407039	1/1/11	Ruby Patel	Stockholm	Sweden	North	Home Office	Office Supplies	Paper	Economy Plus	2011	1	1
1	2	AZ-2011-9050313	1/3/11	Summer Hayward	Southport	United Kingdom	North	Consumer	Furniture	Bookcases	Economy	2011	1	3
2	3	AZ-2011-6674300	1/4/11	Devin Huddleston	Valence	France	Central	Consumer	Office Supplies	Art	Economy	2011	1	4
3	4	BN-2011-2819714	1/4/11	Mary Parker	Birmingham	United Kingdom	North	Corporate	Office Supplies	Art	Economy	2011	1	4
4	5	AZ-2011-617423	1/5/11	Daniel Burke	Echirolles	France	Central	Home Office	Office Supplies	Storage	Priority	2011	1	5
5	6	AZ-2011-2918397	1/7/11	Fredrick Beveridge	La Seyne-sur-Mer	France	Central	Corporate	Office Supplies	Art	Priority	2011	1	7
6	7	BN-2011-3248724	1/8/11	Archer Hort	Toulouse	France	Central	Consumer	Office Supplies	Art	Economy	2011	1	8
7	8	AZ-2011-6712797	1/11/11	Evie Flockhart	Genoa	Italy	South	Consumer	Furniture	Bookcases	Economy	2011	1	11
8	9	AZ-2011-4827146	1/11/11	Faith Greenwood	Vienna	Austria	Central	Consumer	Office Supplies	Fasteners	Economy	2011	1	11

- 3 more columns added during analysis for further depth investigation.

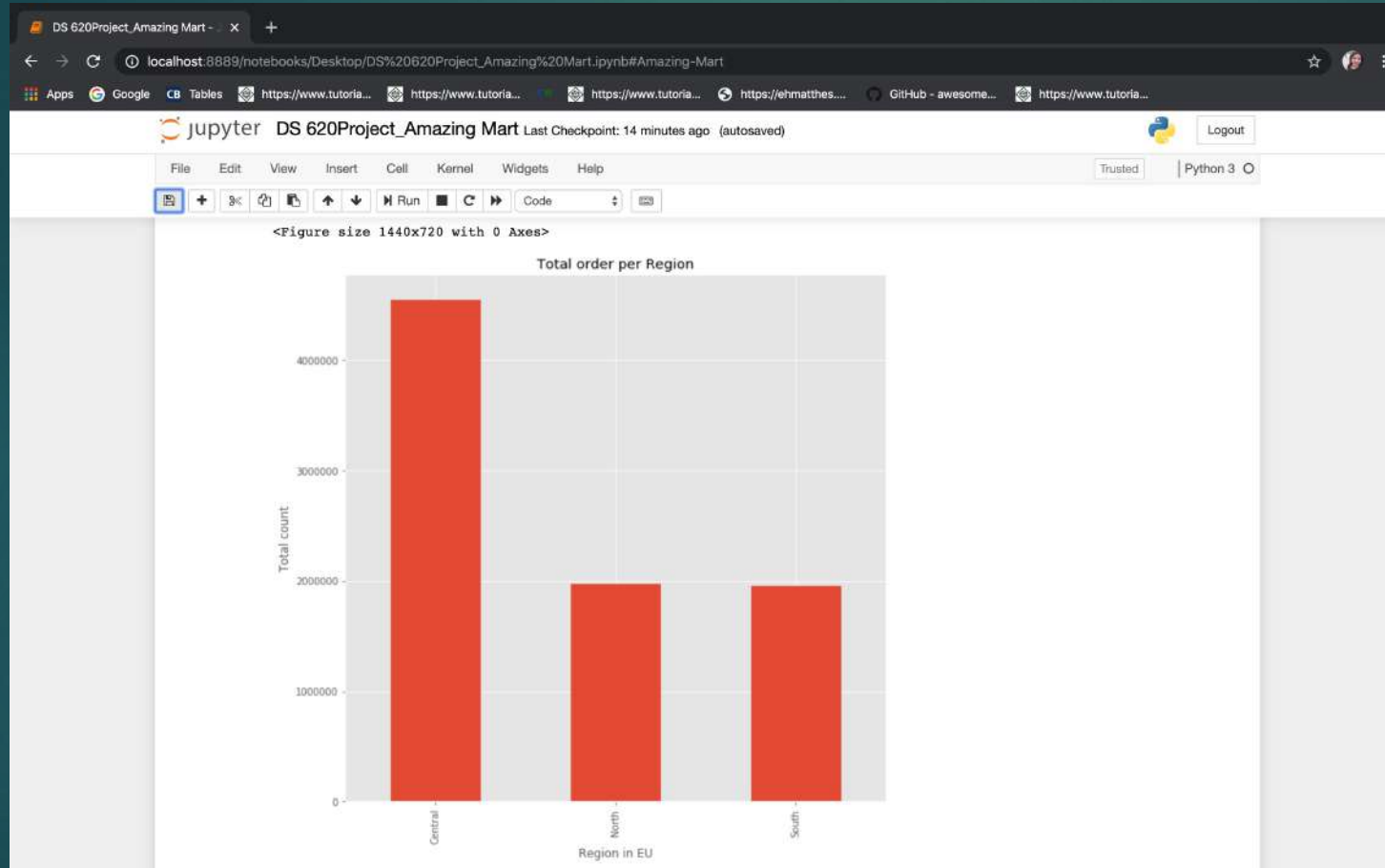


# Country Wise Order



- Histogram of country wise order.
- ggplot plot style for the graph
- In result France, U.K , Germany order the most, while comparing others.

# Region Wise Analysis

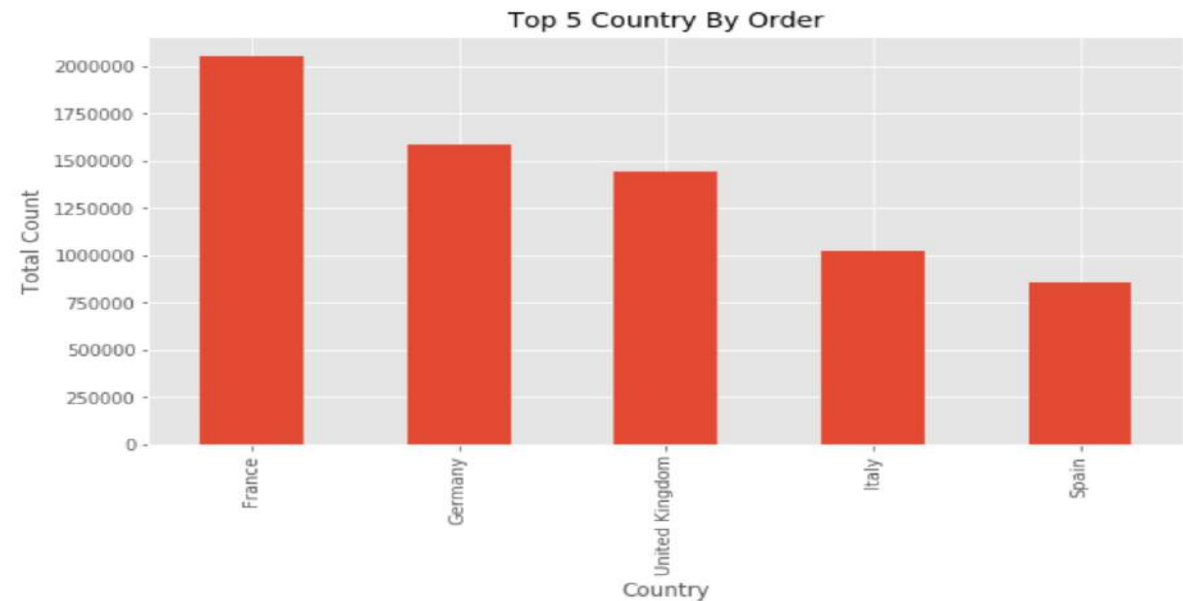


In result it shows that Central part of Europe order most.

# Top 5 Country wise

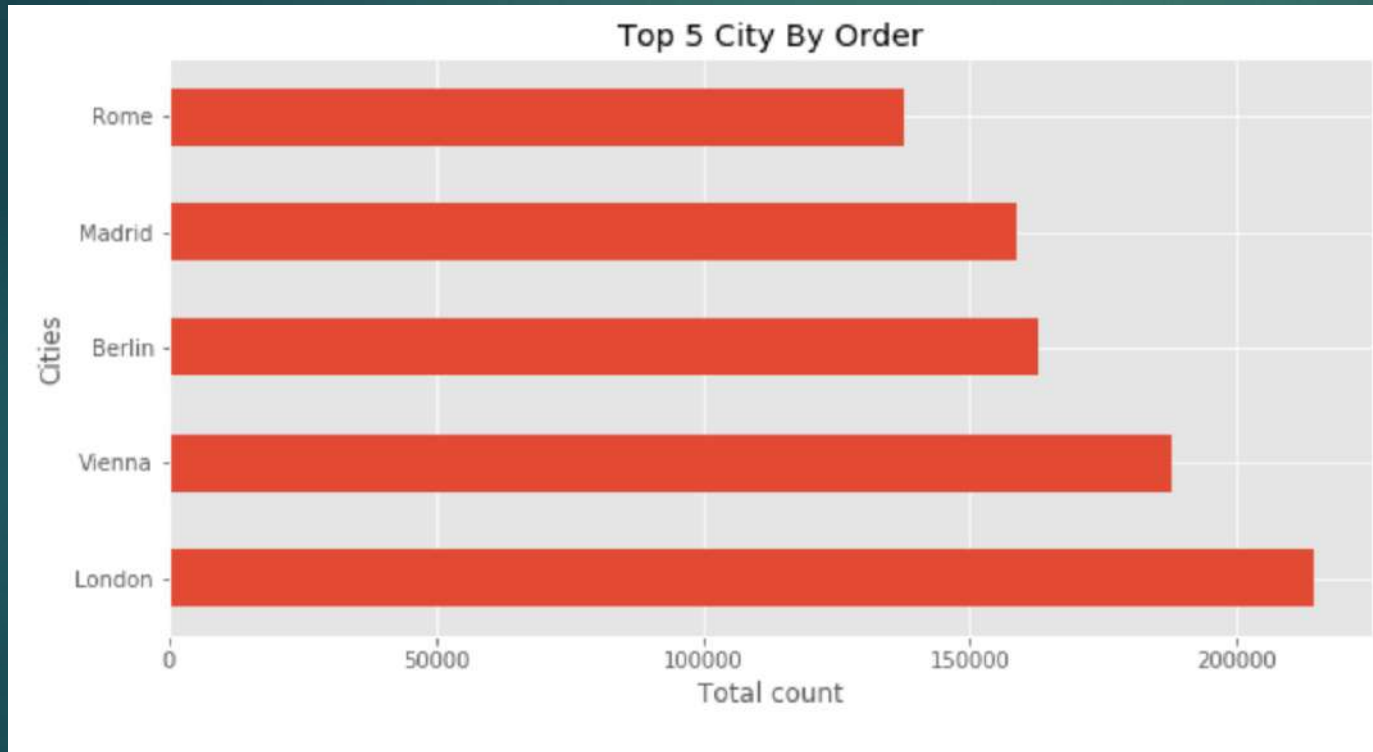
```
In [190]: # creating bar plot with top 5 values with ascending order
plt.figure()
plt.figure(figsize=(10,5))
df.groupby("Country").sr_no.sum().sort_values(ascending=False)[:5].plot.bar()
plt.title('Top 5 Country By Order')
plt.xlabel('Country')
plt.ylabel('Total Count')
plt.show()
```

<Figure size 1440x720 with 0 Axes>



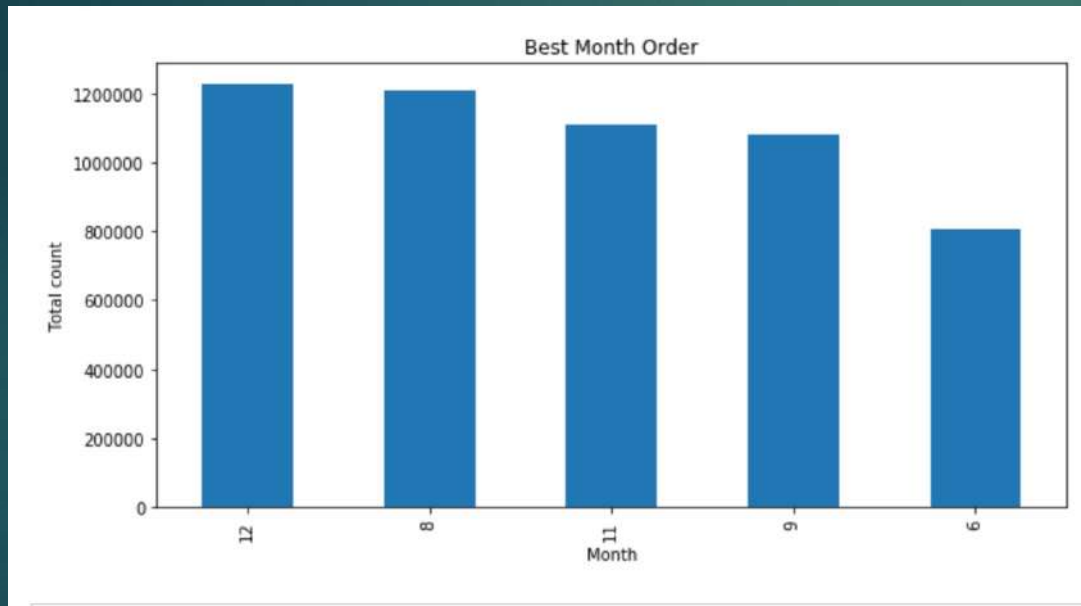
In result France, Germany, UK, Italy and Spain order most as comparing to other.

# Top 5 City order



City wise London,  
Vienna, Berlin, Madrid  
and Rome ordered  
most

# Best Month and Days



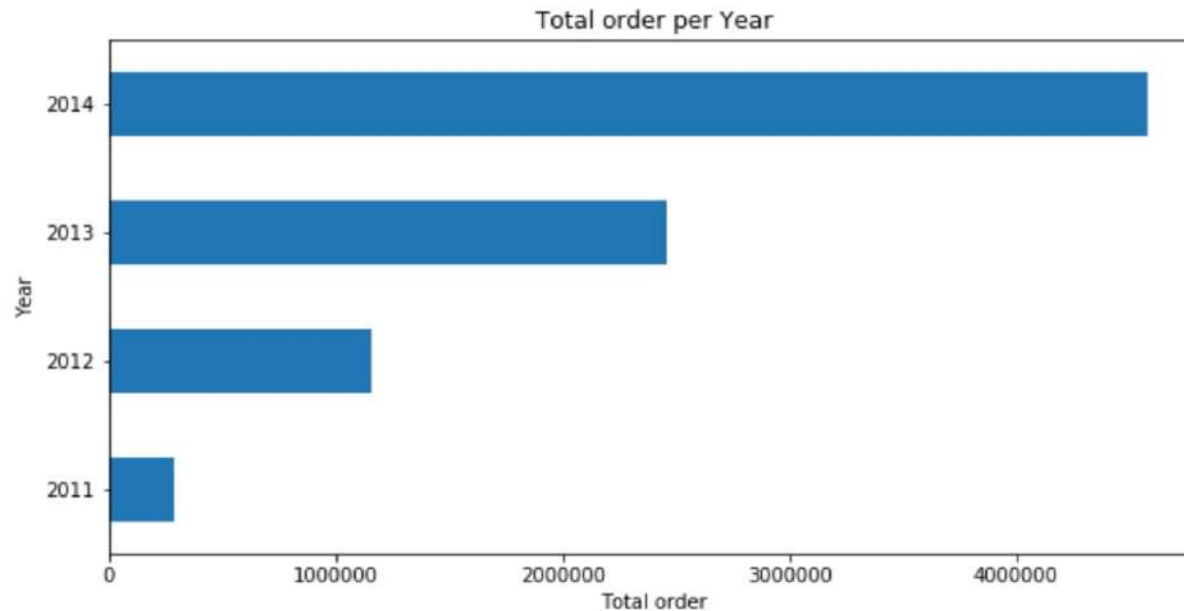
**Best month is December and Best day is 26 in whole year.**



# Yearly ordered

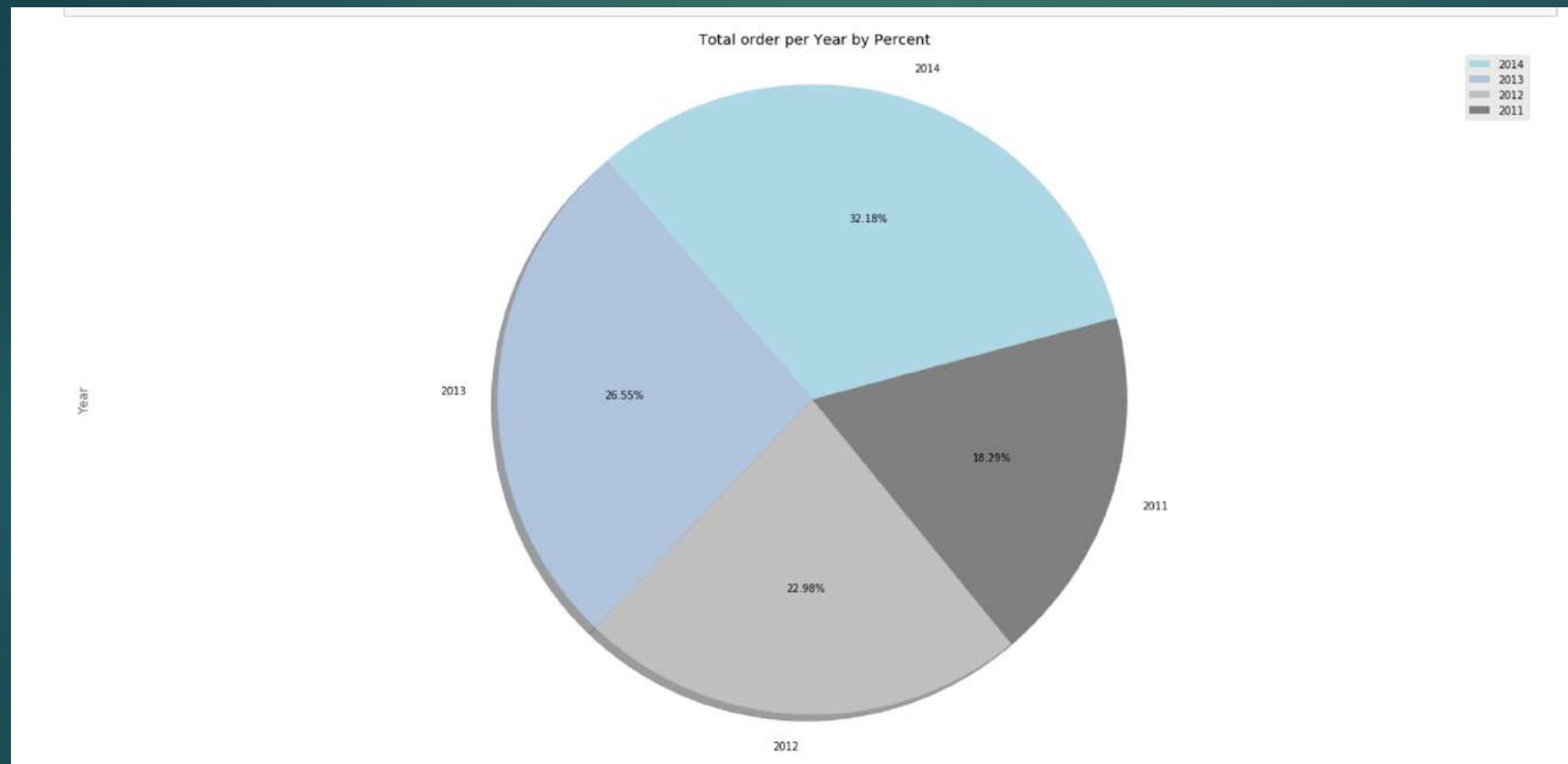
```
In [143]: plt.figure()
plt.figure(figsize=(10,5))
df.groupby('Year').sr_no.sum().plot.barh()
plt.title('Total order per Year')
plt.xlabel('Total order')
plt.ylabel('Year')
plt.show()
```

<Figure size 432x288 with 0 Axes>



Comparing  
2011 to 2014  
order has been  
increased.

# Yearly Order in Percent %



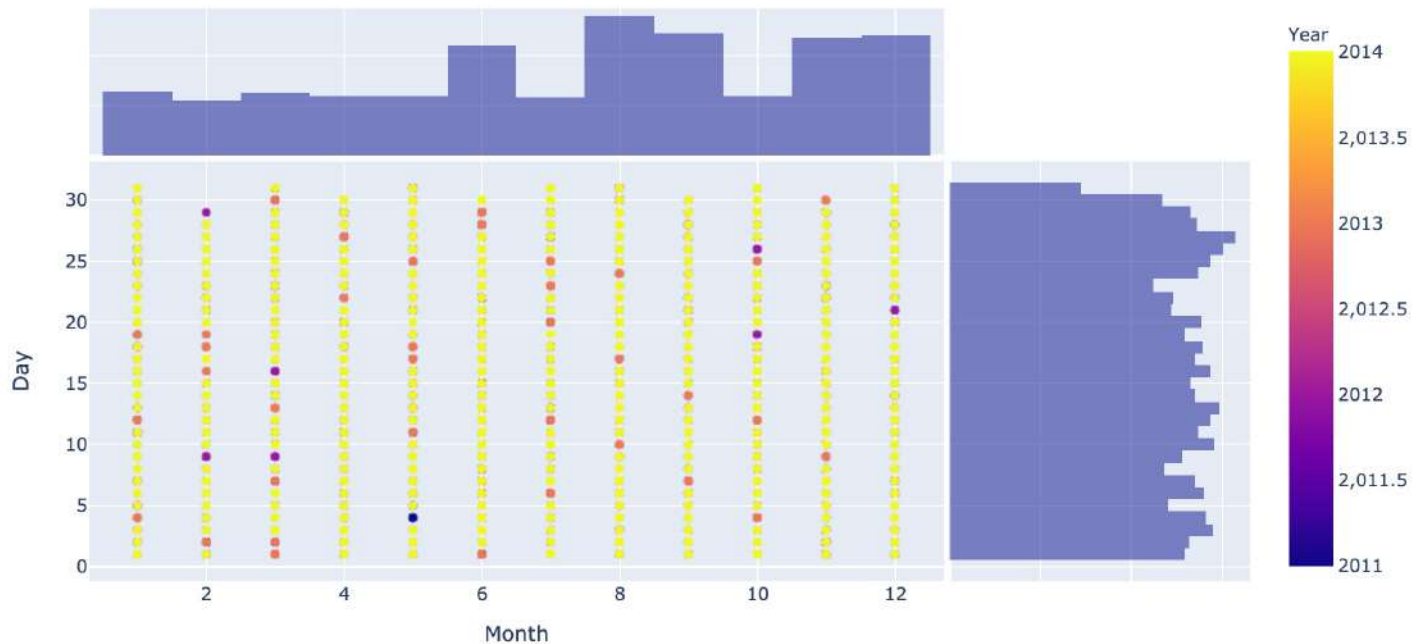
**In 2014 32.18%  
order are the  
most  
comparing  
rest of years.**

# Segment Analysis

```
In [145]: # Scatter plot with histogram
```

```
import plotly_express as px
```

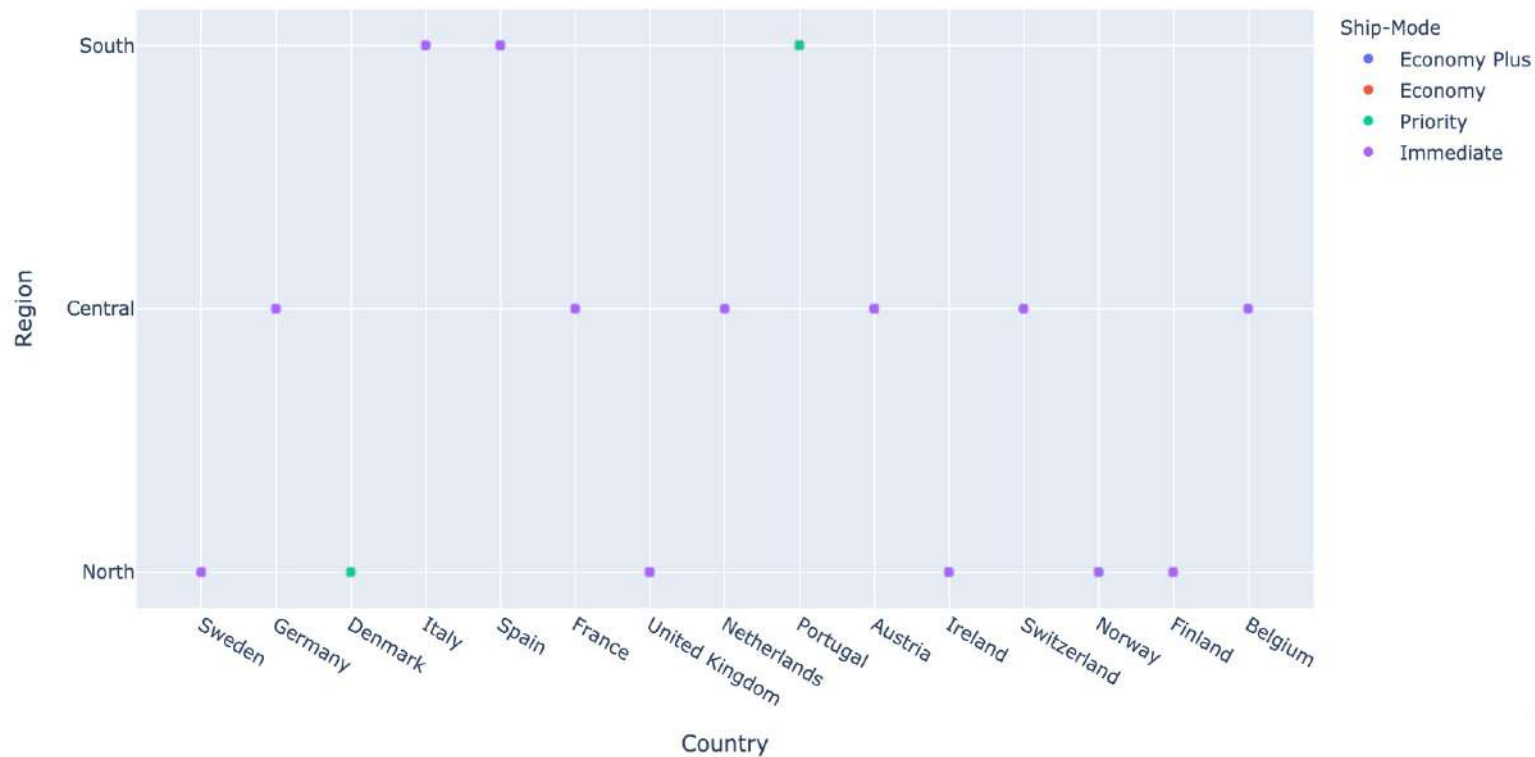
```
px.scatter(df, x = 'Month', y = 'Day', color = 'Year',marginal_y="histogram", marginal_x="histogram")
```



In this analysis  
graphs has  
shown count of  
order month  
and day wise.

# Segment Analysis

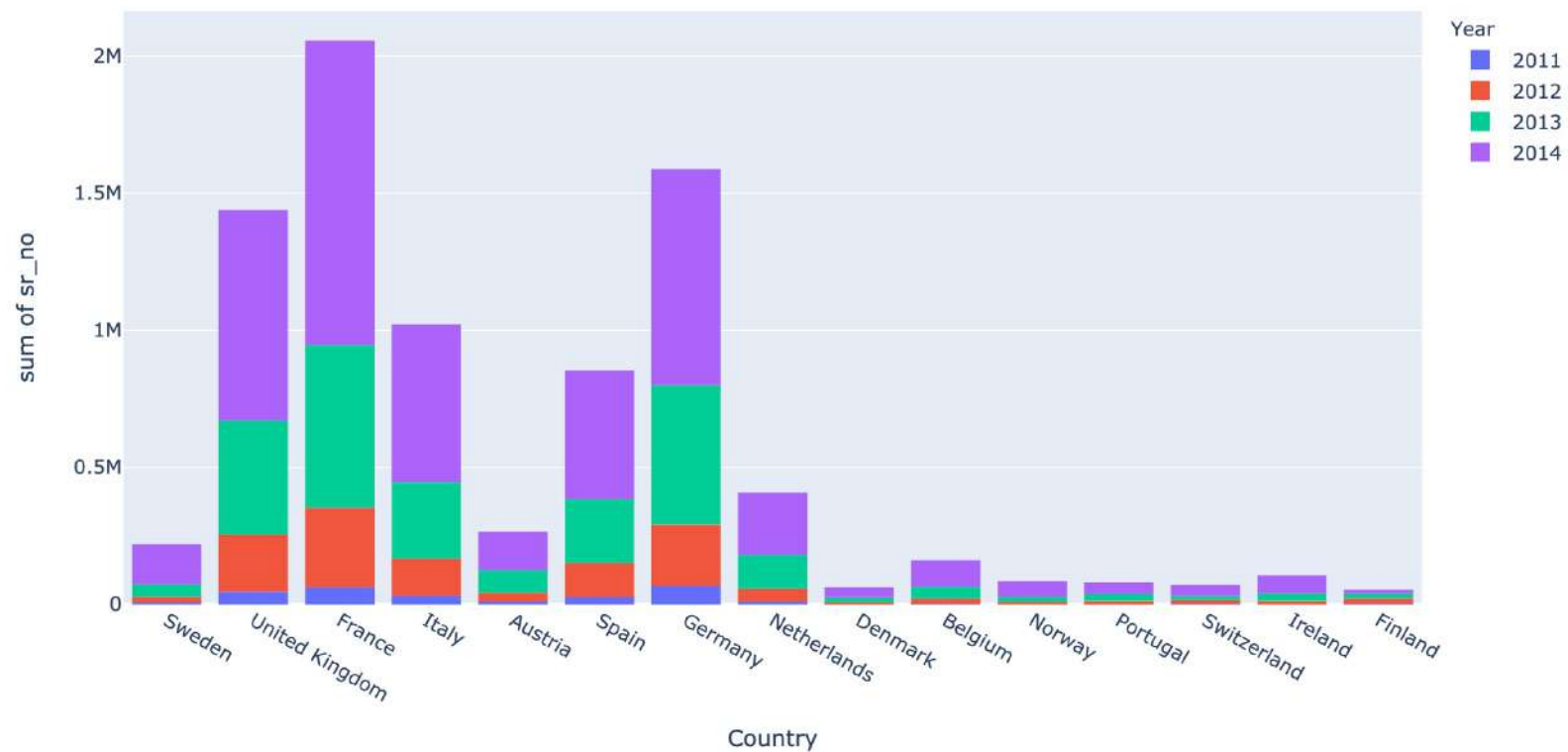
```
In [144]: # creating scatter plot with plotly library  
  
import plotly_express as px  
  
px.scatter(df, x = 'Country', y = 'Region', color = 'Ship-Mode')
```



People are mostly preferred Economy shipping mode. In only south region of Europe Portugal has ordered at priority base.

# Segment Analysis

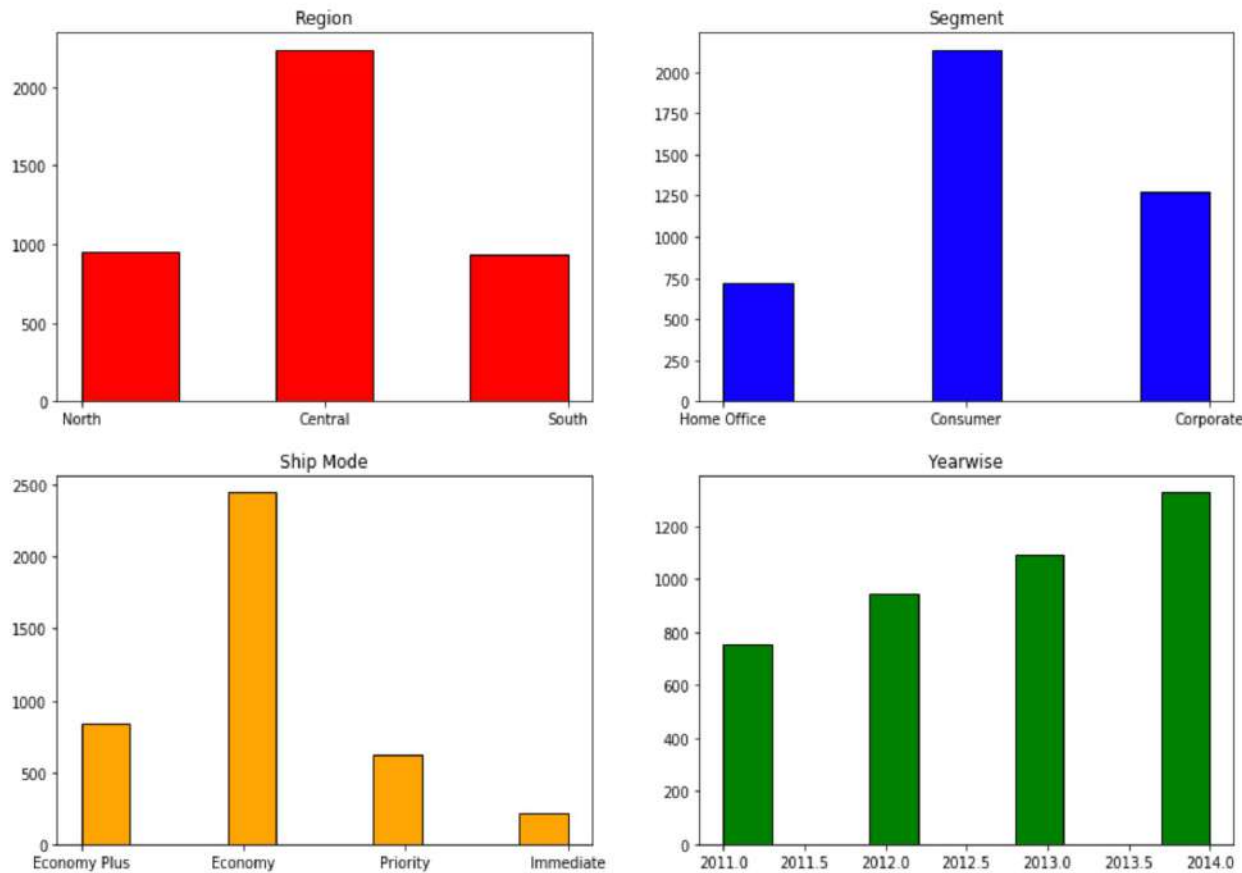
```
In [146]: # created graph with plotly with 3 category  
px.histogram(df, x = 'Country', y = 'sr_no', histfunc = 'sum', color = 'Year')
```





# Segment Analysis

plt.show()



```
import matplotlib.pyplot as plt
import seaborn as sns
```

*#creating histogram subplots*

```
fig,ax = plt.subplots(2,2, figsize=(15,10))
ax[0,0].hist(df['Region'], color = 'red', edgecolor = 'black', bins= 5)
ax[0,0].set_title('Region')
ax[0,1].hist(df['Segment'], color = 'blue',edgecolor = 'black', bins= 7)
ax[0,1].set_title('Segment')
ax[1,0].hist(df['Ship-Mode'], color = 'orange',edgecolor = 'black', bins= 10)
ax[1,0].set_title('Ship Mode')
ax[1,1].hist(df['Year'], color = 'green', edgecolor = 'black', bins= 10)
ax[1,1].set_title('Yearwise')
plt.show()
```

# Conclusion

- ▶ In Europe best region is central and if company want to grow their profit they need to focus on north and south region equally.
- ▶ In segment section Consumer are doing well but company has low orders from home office. they need to improve this section as well for better order in future.
- ▶ In ship mode people Prefer more economy mode than immediate.
- ▶ In France and UK are doing well but rest of country need attention.
- ▶ In Year wise order are increasing but for more profit and more orders company need to focus on other part like Segment and shipment as well.

Thank You