Introduction

The beauty of an art lies in the message it conveys. At times, reality is not what we see or perceive. The endless efforts from the likes of Vinci and Picasso have tried to bring people closer to the reality using their exceptional artworks on a certain topic/matter.

Data scientists are no less than artists. They make paintings in form of digital visualization (of data) with a motive of manifesting the hidden patterns / insights in it. It is even more interesting to know that, the tendency of human perception, cognition and communication increases when he / she gets exposed to visualized form of any content/data.

There are multiple tools for performing visualization in data science. In this article, I have demonstrated various visualization charts using Python.

What does it take to make visualization in Python?

Not much ! Python has already made it easy for you – with two exclusive libraries for visualization, commonly known as *matplotlib and seaborn.* Heard of them?

Matplotlib: Python based plotting library offers *matplotlib* with a complete 2D support along with limited 3D graphic support. It is useful in producing publication quality figures in interactive environment across platforms. It can also be used for animations as well. To know more about this library, check this link.

Seaborn: Seaborn is a library for creating informative and attractive statistical graphics in python. This library is based on matplotlib. Seaborn offers various features such as built in themes, color palettes, functions and tools to visualize univariate, bivariate, linear regression, matrices of data, statistical time series etc which lets us to build complex visualizations. To know more about this library, check this link.

What are the different visualizations I can make?

Last week, A comprehensive guide on Data Visualization was published to introduce you to the most commonly used visualizations techniques. We recommend you to refer that before proceeding further, in case you haven’t.

Below are the python codes with their output. I have used following data set to create these visualization:

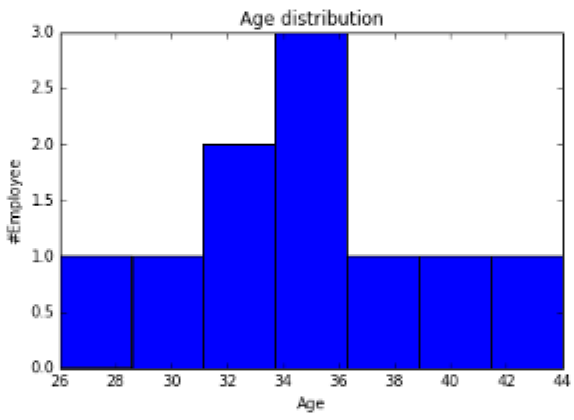
EMPID	Gender	Age	Sales	BMI	Income
E001	M	34	123	Normal	350
E002	F	40	114	Overweight	450
E003	F	37	135	Obesity	169
E004	M	30	139	Underweight	189
E005	F	44	117	Underweight	183
E006	M	36	121	Normal	80
E007	M	32	133	Obesity	166
E008	F	26	140	Normal	120
E009	M	32	133	Normal	75
E010	M	36	133	Underweight	40

Import Data Set:

```
import matplotlib.pyplot as plt

import pandas as pd

df=pd.read_excel("E:/First.xlsx", "Sheet1")
```

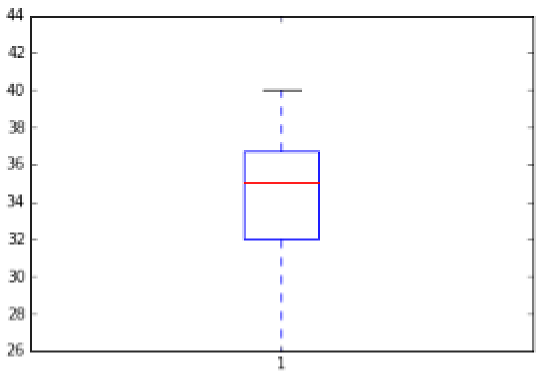
Histogram :

```
fig=plt.figure() #Plots in matplotlib reside within a figure object, use plt.figure to create new fi
gure

#Create one or more subplots using add_subplot, because you can't create blank figure

ax = fig.add_subplot(1,1,1)

#Variable

ax.hist(df['Age'],bins = 7) # Here you can play with number of bins

Labels and Tit

plt.title('Age distribution')

plt.xlabel('Age')

plt.ylabel('#Employee')

plt.show()
```



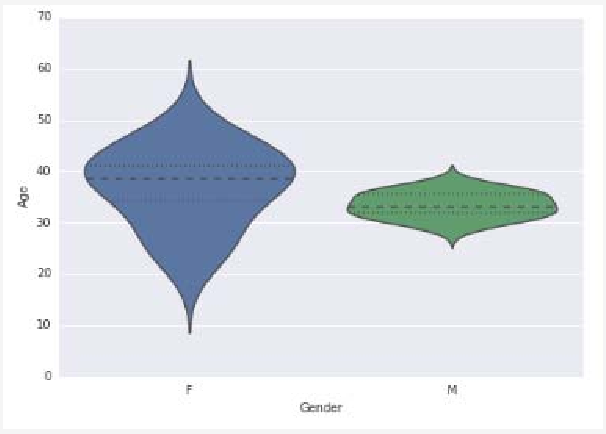
Box Plot

```
import matplotlib.pyplot as plt

import pandas as pd

fig=plt.figure()

ax = fig.add_subplot(1,1,1)

#Variable

ax.boxplot(df['Age'])

plt.show()
```



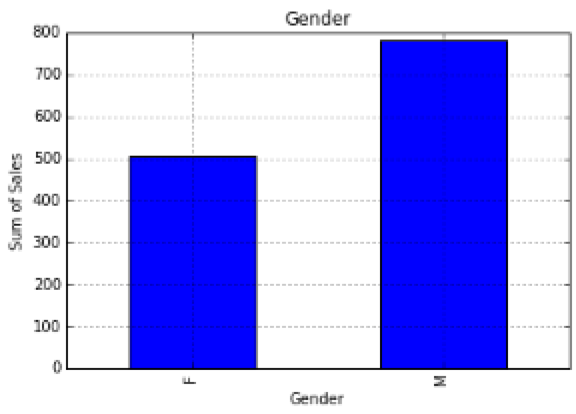
Violin Plot

```
import seaborn as sns

sns.violinplot(df['Age'], df['Gender']) #Variable Plot

sns.despine()
```



Bar Chart

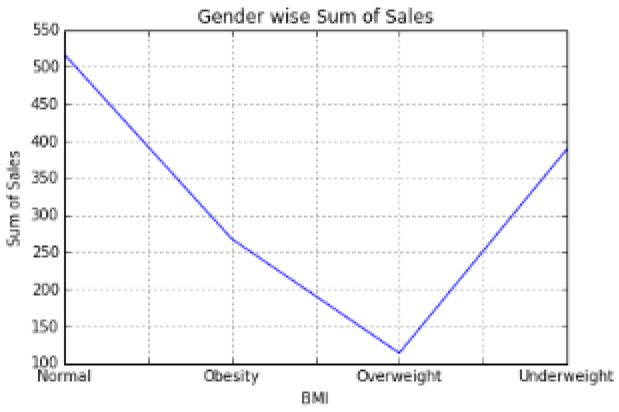
```
var = df.groupby('Gender').Sales.sum() #grouped sum of sales at Gender level

fig = plt.figure()

ax1 = fig.add_subplot(1,1,1)

ax1.set_xlabel('Gender')

ax1.set_ylabel('Sum of Sales')

ax1.set_title("Gender wise Sum of Sales")

var.plot(kind='bar')
```



You can read more about pandas **groupby**here and for dataframe. For plot refer this link.

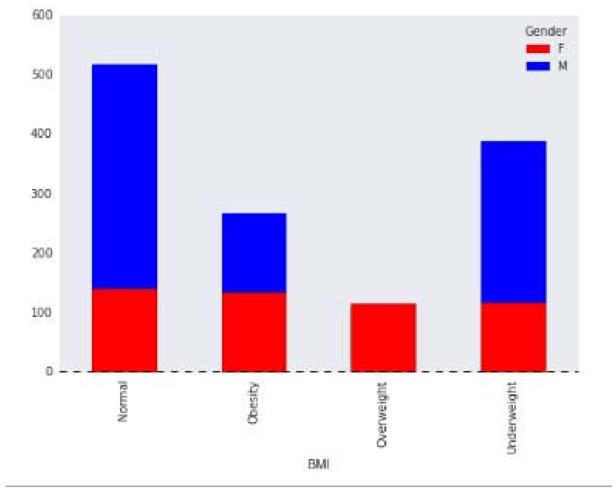
Line Chart

```
var = df.groupby('BMI').Sales.sum()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('BMI')
ax1.set_ylabel('Sum of Sales')
ax1.set_title("BMI wise Sum of Sales")
var.plot(kind='line')
```



Stacked Column Chart

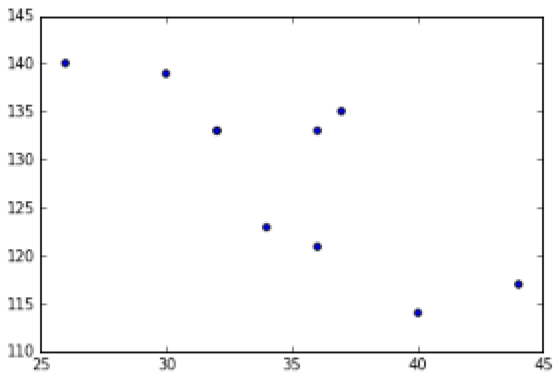
```
var = df.groupby(['BMI','Gender']).Sales.sum()
var.unstack().plot(kind='bar',stacked=True,  color=['red','blue'], grid=False)
```



Dataframe.unstack() returns a Data Frame having a new level of column labels whose inner-most level consists of the pivoted index labels. Read more about dataframe.unstack here.

Scatter Plot

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(df['Age'],df['Sales']) #You can also add more variables here to represent color and size.
plt.show()
```



Bubble Plot

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
```