```
import numpy as np
import pandas as pd
from google.colab import files
fupload = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

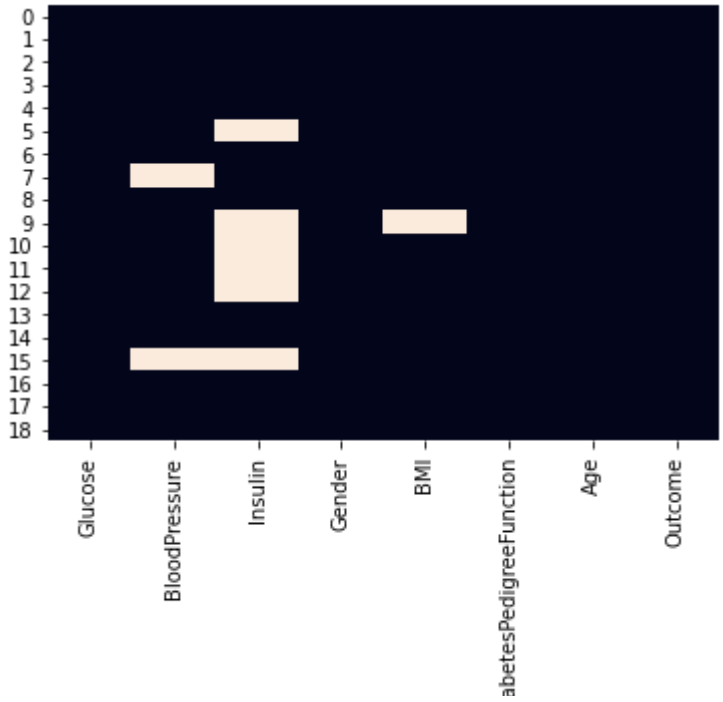
Saving diabetes2.csv to diabetes2.csv

```
data=pd.read_csv("diabetes2.csv")
data
```

	Glucose	BloodPressure	Insulin	Gender	BMI	DiabetesPedigreeFunction	Age	Out
0	148	72.0	88.0	M	33.6	0.627	50	
1	85	66.0	90.0	F	26.6	0.351	31	
2	183	64.0	75.0	M	23.3	0.672	32	
3	89	66.0	94.0	M	28.1	0.167	21	
4	137	40.0	168.0	M	43.1	2.288	33	
5	116	74.0	NaN	F	25.6	0.201	30	
6	78	50.0	88.0	M	31.0	0.248	26	
7	115	NaN	78.0	F	35.3	0.134	29	
8	197	70.0	543.0	F	30.5	0.158	53	
9	125	96.0	NaN	M	NaN	0.232	54	
10	110	92.0	NaN	F	37.6	0.191	30	
11	168	74.0	NaN	M	38.0	0.537	34	
12	139	80.0	NaN	F	27.1	1.441	57	
13	189	60.0	846.0	M	30.1	0.398	59	
14	166	72.0	175.0	M	25.8	0.587	51	
15	100	NaN	NaN	M	30.0	0.484	32	
16	118	84.0	230.0	M	45.8	0.551	31	
17	107	74.0	98.0	M	29.6	0.254	31	
18	103	30.0	83.0	F	43.3	0.183	33	

```
#checking NULL values is present or not in CSV file using heatmap
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(data.isnull(), cbar=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa4cd0c85c0>
```




```
print(data.describe())
```

```
            Glucose  BloodPressure  ...        Age    Outcome
count   19.000000      17.000000  ...  19.000000  19.000000
mean   130.157895      68.470588  ...  37.736842   0.631579
std     36.155933      16.845317  ...  11.836113   0.495595
min     78.000000      30.000000  ...  21.000000   0.000000
25%    105.000000      64.000000  ...  30.500000   0.000000
50%    118.000000      72.000000  ...  32.000000   1.000000
75%    157.000000      74.000000  ...  50.500000   1.000000
max    197.000000      96.000000  ...  59.000000   1.000000

[8 rows x 7 columns]
```

```
print(data.shape)
```

```
(19, 8)
```

```
print(data.head(5))
```

```
   Glucose  BloodPressure  Insulin  ... DiabetesPedigreeFunction  Age  Outcome
0      148           72.0     88.0  ...                    0.627   50        1
1       85           66.0     90.0  ...                    0.351   31        0
2      183           64.0     75.0  ...                    0.672   32        1
3       89           66.0     94.0  ...                    0.167   21        0
4      137           40.0    168.0  ...                    2.288   33        1

[5 rows x 8 columns]
```

```
x=data[['Glucose','BloodPressure','Insulin','Gender','DiabetesPedigreeFunction','Age']].va
print(x[0:8])
#print(x)
```

```
[[148 72.0 88.0 'M' 0.627 50]
 [85 66.0 90.0 'F' 0.35100000000000003 31]
```

```
 [183 64.0 75.0 'M' 0.672 32]
 [89 66.0 94.0 'M' 0.16699999999999998 21]
 [137 40.0 168.0 'M' 2.2880000000000003 33]
 [116 74.0 nan 'F' 0.201 30]
 [78 50.0 88.0 'M' 0.248 26]
 [115 nan 78.0 'F' 0.134 29]]
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(['M','F'])
x[:,3]=le.transform(x[:,3])
print(x)
```

```
[[148 72.0 88.0 1 0.627 50]
 [85 66.0 90.0 0 0.35100000000000003 31]
 [183 64.0 75.0 1 0.672 32]
 [89 66.0 94.0 1 0.16699999999999998 21]
 [137 40.0 168.0 1 2.2880000000000003 33]
 [116 74.0 nan 0 0.201 30]
 [78 50.0 88.0 1 0.248 26]
 [115 nan 78.0 0 0.134 29]
 [197 70.0 543.0 0 0.158 53]
 [125 96.0 nan 1 0.23199999999999998 54]
 [110 92.0 nan 0 0.191 30]
 [168 74.0 nan 1 0.537 34]
 [139 80.0 nan 0 1.4409999999999998 57]
 [189 60.0 846.0 1 0.39799999999999996 59]
 [166 72.0 175.0 1 0.5870000000000001 51]
 [100 nan nan 1 0.484 32]
 [118 84.0 230.0 1 0.551 31]
 [107 74.0 98.0 1 0.254 31]
 [103 30.0 83.0 0 0.183 33]]
```

```
y=data['Outcome'].values
print(y[0:5])
print(y)
```

```
[1 0 1 0 1]
[1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 1 0]
```

```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy="mean")
x=imputer.fit_transform(x)
print(x)
```

```
[[1.48000000e+02 7.20000000e+01 8.80000000e+01 1.00000000e+00
  6.27000000e-01 5.00000000e+01]
 [8.50000000e+01 6.60000000e+01 9.00000000e+01 0.00000000e+00
  3.51000000e-01 3.10000000e+01]
 [1.83000000e+02 6.40000000e+01 7.50000000e+01 1.00000000e+00
  6.72000000e-01 3.20000000e+01]
 [8.90000000e+01 6.60000000e+01 9.40000000e+01 1.00000000e+00
  1.67000000e-01 2.10000000e+01]
 [1.37000000e+02 4.00000000e+01 1.68000000e+02 1.00000000e+00
  2.28800000e+00 3.30000000e+01]
 [1.16000000e+02 7.40000000e+01 2.04307692e+02 0.00000000e+00
```

```
 2.01000000e-01 3.00000000e+01]
[7.80000000e+01 5.00000000e+01 8.80000000e+01 1.00000000e+00
 2.48000000e-01 2.60000000e+01]
[1.15000000e+02 6.84705882e+01 7.80000000e+01 0.00000000e+00
 1.34000000e-01 2.90000000e+01]
[1.97000000e+02 7.00000000e+01 5.43000000e+02 0.00000000e+00
 1.58000000e-01 5.30000000e+01]
[1.25000000e+02 9.60000000e+01 2.04307692e+02 1.00000000e+00
 2.32000000e-01 5.40000000e+01]
[1.10000000e+02 9.20000000e+01 2.04307692e+02 0.00000000e+00
 1.91000000e-01 3.00000000e+01]
[1.68000000e+02 7.40000000e+01 2.04307692e+02 1.00000000e+00
 5.37000000e-01 3.40000000e+01]
[1.39000000e+02 8.00000000e+01 2.04307692e+02 0.00000000e+00
 1.44100000e+00 5.70000000e+01]
[1.89000000e+02 6.00000000e+01 8.46000000e+02 1.00000000e+00
 3.98000000e-01 5.90000000e+01]
[1.66000000e+02 7.20000000e+01 1.75000000e+02 1.00000000e+00
 5.87000000e-01 5.10000000e+01]
[1.00000000e+02 6.84705882e+01 2.04307692e+02 1.00000000e+00
 4.84000000e-01 3.20000000e+01]
[1.18000000e+02 8.40000000e+01 2.30000000e+02 1.00000000e+00
 5.51000000e-01 3.10000000e+01]
[1.07000000e+02 7.40000000e+01 9.80000000e+01 1.00000000e+00
 2.54000000e-01 3.10000000e+01]
[1.03000000e+02 3.00000000e+01 8.30000000e+01 0.00000000e+00
 1.83000000e-01 3.30000000e+01]]
```

```
#Handling High Variance
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=scaler.fit_transform(x)
print(x)
```

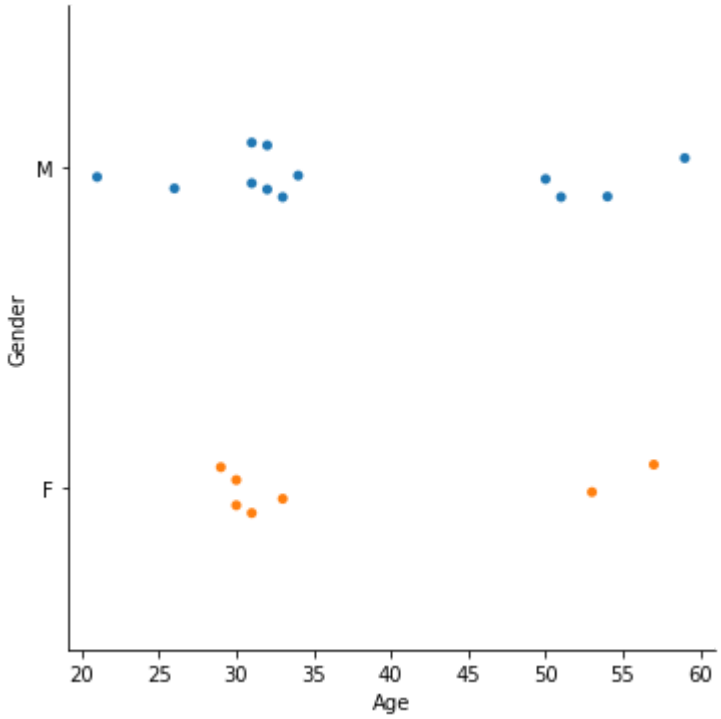
```
[[ 0.50699897  0.22831789 -0.63356898  0.76376262  0.22633617  1.06447085]
 [-1.28320093 -0.15982253 -0.62267428 -1.30930734 -0.31096889 -0.58477368]
 [ 1.50155447 -0.28920267 -0.70438456  0.76376262  0.31394026 -0.49797134]
 [-1.16953745 -0.15982253 -0.60088487  0.76376262 -0.66917226 -1.45279712]
 [ 0.19442438 -1.84176434 -0.19778079  0.76376262  3.45990032 -0.411169  ]
 [-0.40230892  0.35769803  0.         -1.30930734 -0.60298251 -0.67157603]
 [-1.48211203 -1.19486364 -0.63356898  0.76376262 -0.51148491 -1.0187854 ]
 [-0.43072479  0.         -0.6880425  -1.30930734 -0.73341526 -0.75837837]
 [ 1.89937667  0.09893775  1.84497634 -1.30930734 -0.68669308  1.32487788]
 [-0.14656607  1.78087957  0.          0.76376262 -0.54263303  1.41168022]
 [-0.57280415  1.52211929  0.         -1.30930734 -0.62245008 -0.67157603]
 [ 1.0753164   0.35769803  0.          0.76376262  0.051128   -0.32436665]
 [ 0.25125613  0.74583845  0.         -1.30930734  1.81099675  1.67208725]
 [ 1.6720497  -0.54796294  3.4955241   0.76376262 -0.21947129  1.84569194]
 [ 1.01848466  0.22831789 -0.15964933  0.76376262  0.14846587  1.15127319]
 [-0.85696286  0.          0.          0.76376262 -0.05205015 -0.49797134]
 [-0.34547717  1.00459873  0.13995505  0.76376262  0.0783826  -0.58477368]
 [-0.65805176  0.35769803 -0.57909546  0.76376262 -0.49980436 -0.58477368]
 [-0.77171525 -2.48866504 -0.66080574 -1.30930734 -0.63802414 -0.411169  ]]
```

```
#min-max normalization
from sklearn.preprocessing import MinMaxScaler
s=MinMaxScaler(feature_range=(0,1))
rescaledx=s.fit_transform(x)
np.set_printoptions(precision=3)
```

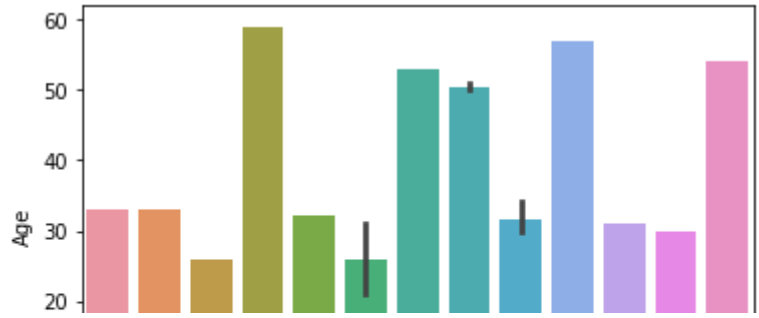
```
np.set_printoptions(precision=3)
print(x)
```

```
[[ 0.507  0.228 -0.634  0.764  0.226  1.064]
 [-1.283 -0.16  -0.623 -1.309 -0.311 -0.585]
 [ 1.502 -0.289 -0.704  0.764  0.314 -0.498]
 [-1.17  -0.16  -0.601  0.764 -0.669 -1.453]
 [ 0.194 -1.842 -0.198  0.764  3.46  -0.411]
 [-0.402  0.358  0.    -1.309 -0.603 -0.672]
 [-1.482 -1.195 -0.634  0.764 -0.511 -1.019]
 [-0.431  0.    -0.688 -1.309 -0.733 -0.758]
 [ 1.899  0.099  1.845 -1.309 -0.687  1.325]
 [-0.147  1.781  0.     0.764 -0.543  1.412]
 [-0.573  1.522  0.    -1.309 -0.622 -0.672]
 [ 1.075  0.358  0.     0.764  0.051 -0.324]
 [ 0.251  0.746  0.    -1.309  1.811  1.672]
 [ 1.672 -0.548  3.496  0.764 -0.219  1.846]
 [ 1.018  0.228 -0.16   0.764  0.148  1.151]
 [-0.857  0.     0.     0.764 -0.052 -0.498]
 [-0.345  1.005  0.14   0.764  0.078 -0.585]
 [-0.658  0.358 -0.579  0.764 -0.5   -0.585]
 [-0.772 -2.489 -0.661 -1.309 -0.638 -0.411]]
```

```
import seaborn as sns
res = sns.catplot(x="Age", y="Gender", data=data)
plt.show()
```

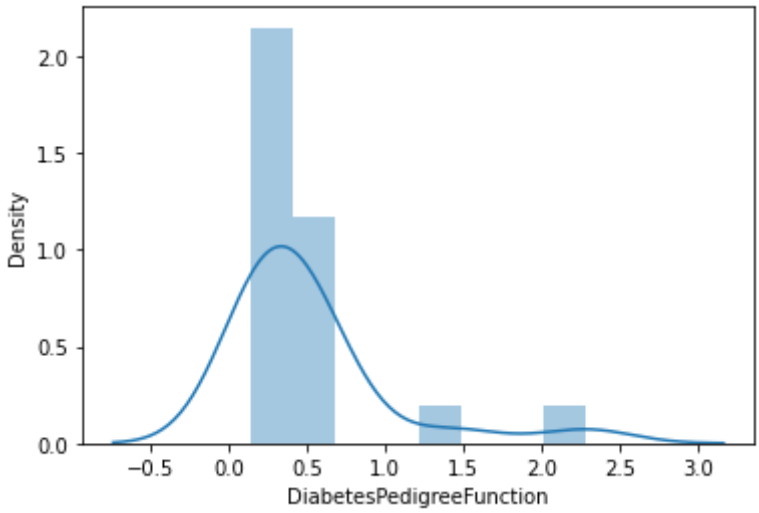



```
res = sns.barplot(x=data['BloodPressure'], y=data['Age'])
plt.show()
```




```
sns.distplot(data["DiabetesPedigreeFunction"], hist=True)
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
```




```
sns.histplot(data["Age"], binwidth=3)
plt.show()
```