Task 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value: • Glucose • BloodPressure • SkinThickness • Insulin • BMI
2. Visually explore these variables using histograms. Treat the missing values accordingly.
3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

In [2]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

In [6]:
```
df=pd.read_csv("health care diabetes.csv")
df.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

DESCRIPTIVE ANALYSIS

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [12]:
```
df.describe()
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [13]:
```
print("Standard Deviation of each variables are ==> ")
df.apply(np.std)
```

```
Standard Deviation of each variables are ==>
```

Out[13]:
```
Pregnancies                   3.367384
Glucose                      31.951796
BloodPressure                19.343202
SkinThickness                15.941829
Insulin                     115.168949
BMI                           7.879026
DiabetesPedigreeFunction      0.331113
Age                          11.752573
Outcome                       0.476641
dtype: float64
```

Replace 0 to NAN

In [7]:
```
df.isnull().sum().sum()
```

Out[7]: 0

In [16]:
```
df1 = df.copy(deep = True)
df1[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']]= df[['Glucose',
'BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

In [9]:
```
print(df1)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6    148.0           72.0           35.0      NaN  33.6
1              1     85.0           66.0           29.0      NaN  26.6
2              8    183.0           64.0            NaN      NaN  23.3
3              1     89.0           66.0           23.0     94.0  28.1
4              0    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
763           10    101.0           76.0           48.0    180.0  32.9
764            2    122.0           70.0           27.0      NaN  36.8
765            5    121.0           72.0           23.0    112.0  26.2
766            1    126.0           60.0            NaN      NaN  30.1
767            1     93.0           70.0           31.0      NaN  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

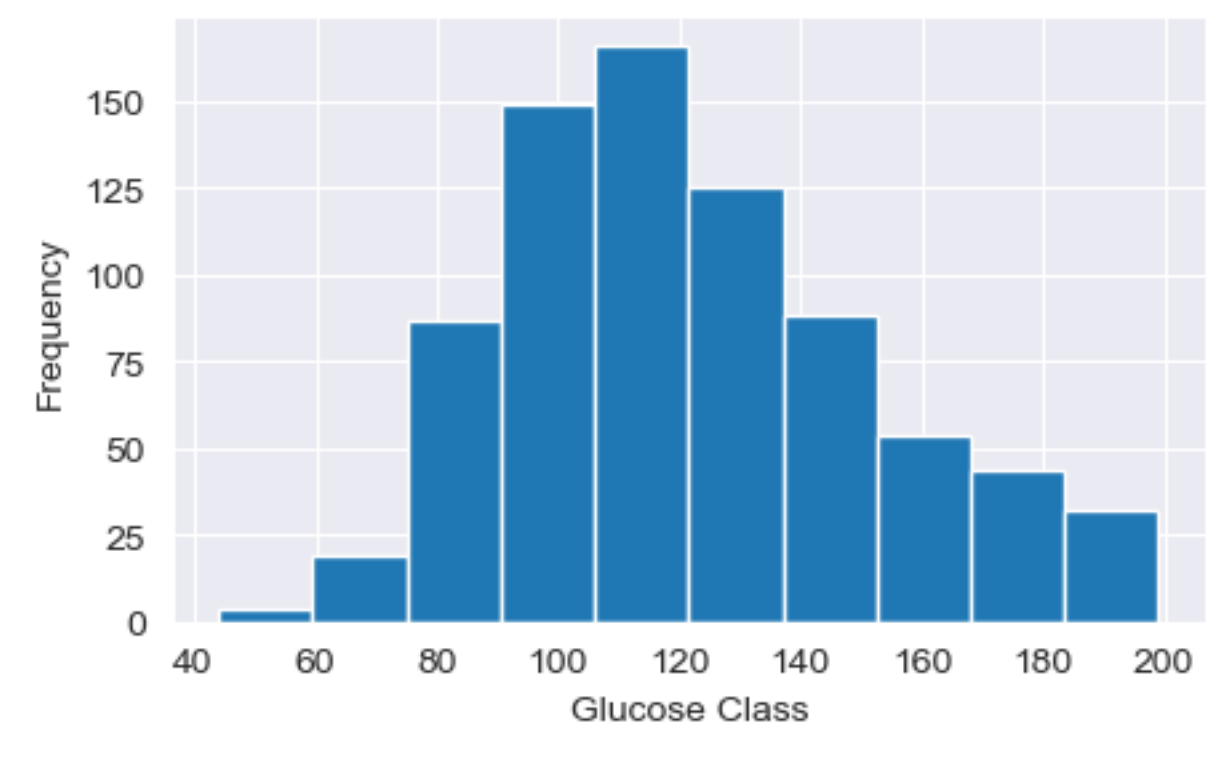
In [10]:
```
df1.isnull().sum().sum()
```

Out[10]: 652

Treating Missing Values

In [28]:
```
plt.figure(figsize=(5,3),dpi=100)
plt.xlabel('Glucose Class')
df['Glucose'].plot.hist()
sns.set_style(style='darkgrid')
print("Mean of Glucose level is :-", df['Glucose'].mean())
print("Datatype of Glucose Variable is:",df['Glucose'].dtypes)
```

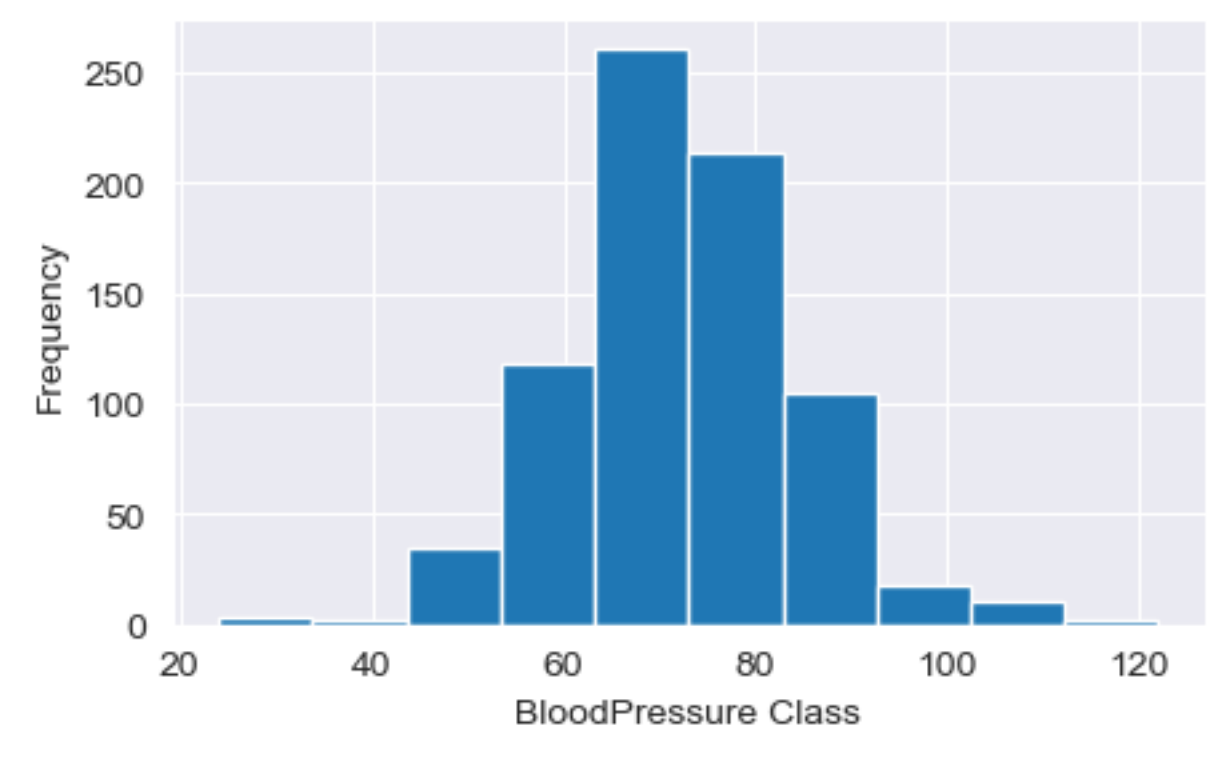
```
Mean of Glucose level is :- 121.68160502115886
Datatype of Glucose Variable is: float64
```



In [20]:
```
df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())
```

In [29]:
```
plt.figure(figsize=(5,3),dpi=100)
plt.xlabel("BloodPressure Class")
df['BloodPressure'].plot.hist()
sns.set_style(style='darkgrid')
print("Mean of BloodPressure level is :-", df['BloodPressure'].mean())
print("Datatype of BloodPressure Variable is:",df['BloodPressure'].dtypes)
```

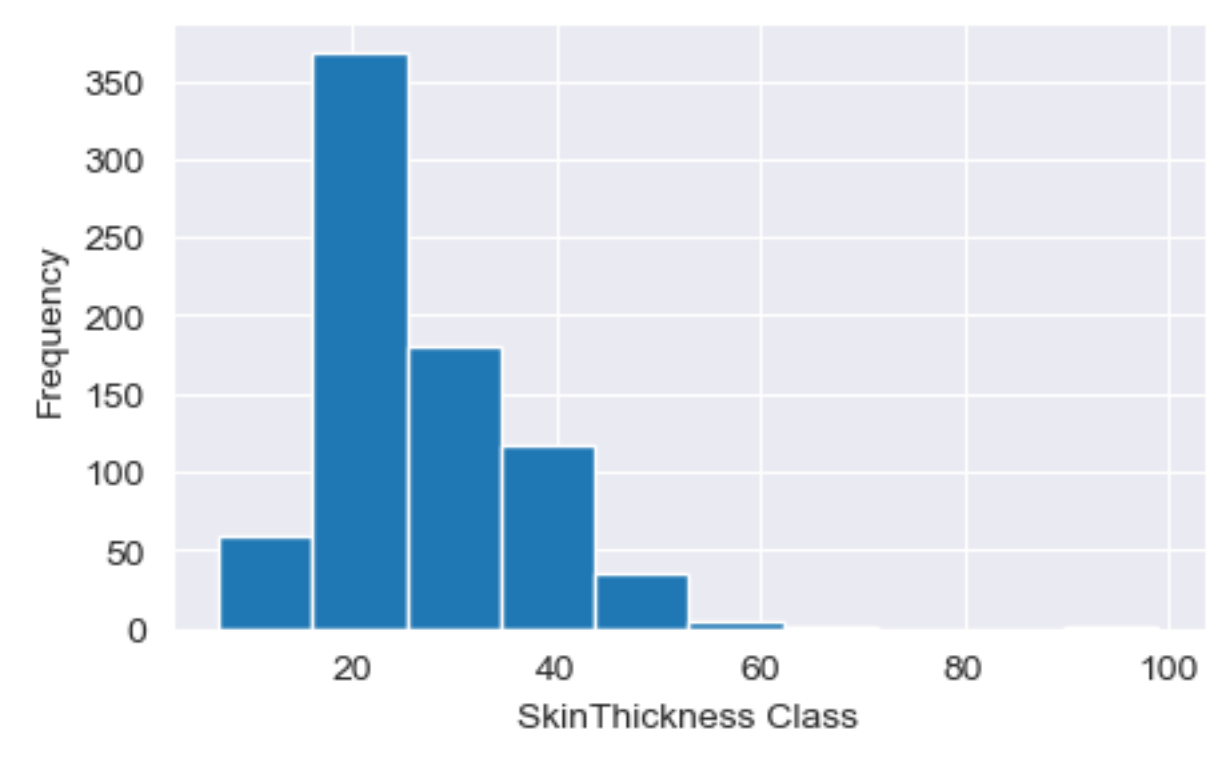
```
Mean of BloodPressure level is :- 72.25480651855469
Datatype of BloodPressure Variable is: float64
```



In [23]:
```
df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())
```

In [30]:
```
plt.figure(figsize=(5,3,),dpi=100)
plt.xlabel('SkinThickness Class')
df['SkinThickness'].plot.hist()
sns.set_style(style='darkgrid')
print("Mean of SkinThickness is :-", df['SkinThickness'].mean())
print("Datatype of SkinThickness Variable is:",df['SkinThickness'].dtypes)
```

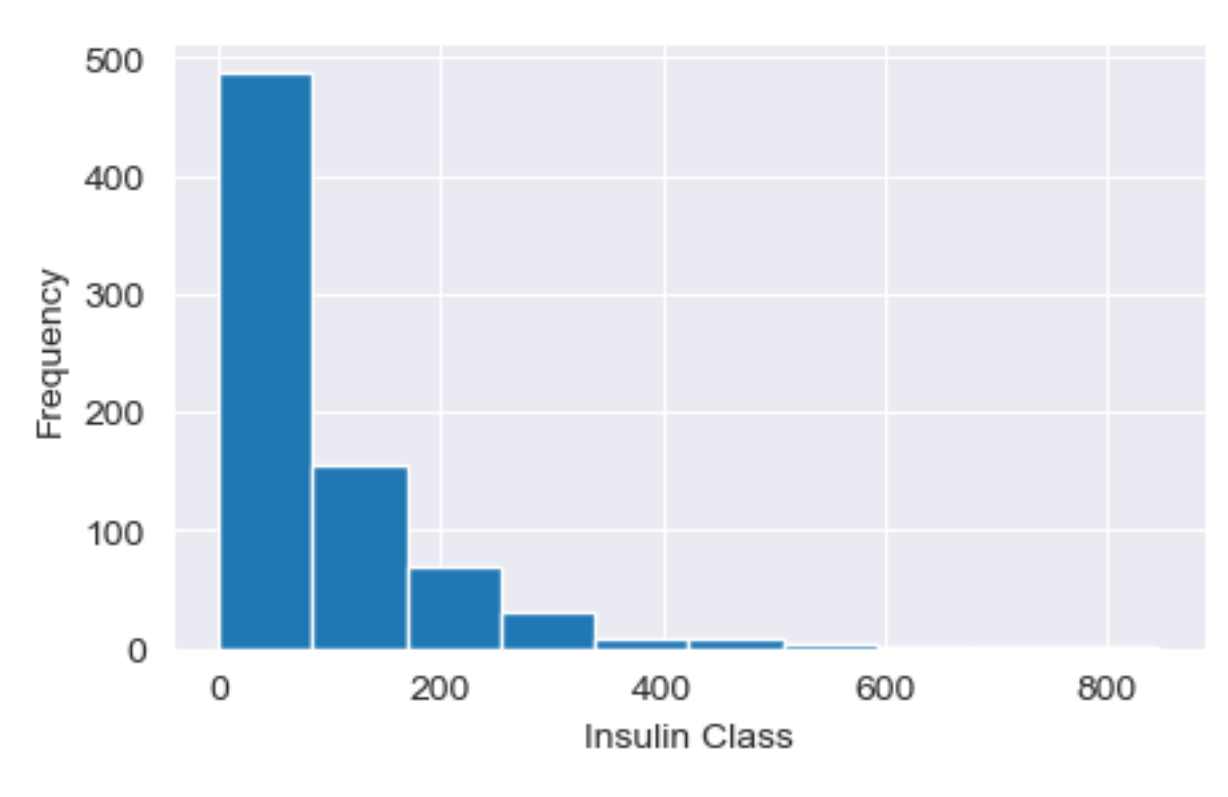
```
Mean of SkinThickness is :- 26.606479220920118
Datatype of SkinThickness Variable is: float64
```



In [27]:
```
df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].mean())
```

In [31]:
```
plt.figure(figsize=(5,3),dpi=100)
plt.xlabel('Insulin Class')
df['Insulin'].plot.hist()
sns.set_style(style='darkgrid')
print("Mean of Insulin is :-", df['Insulin'].mean())
print("Datatype of Insulin Variable is:",df['Insulin'].dtypes)
```

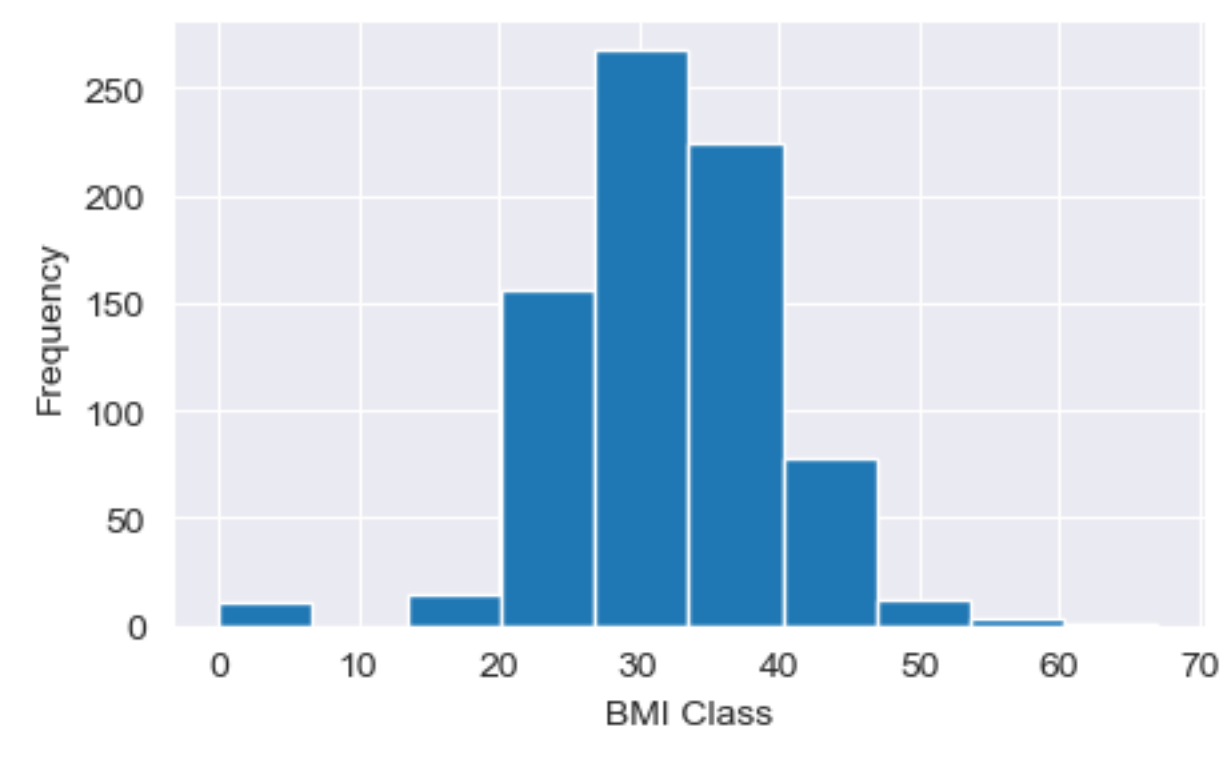
```
Mean of Insulin is :- 79.79947916666667
Datatype of Insulin Variable is: int64
```



In [32]:
```
df['Insulin']=df['Insulin'].replace(0,df['Insulin'].mean())
```

In [33]:
```
plt.figure(figsize=(5,3),dpi=100)
plt.xlabel('BMI Class')
df['BMI'].plot.hist()
sns.set_style(style='darkgrid')
print("Mean of BMI is :-", df['BMI'].mean())
print("Datatype of BMI Variable is:",df['BMI'].dtypes)
```

```
Mean of BMI is :- 31.992578124999977
Datatype of BMI Variable is: float64
```



In [34]:
```
df['BMI']=df['BMI'].replace(0,df['BMI'].mean())
```

In [35]:
```
df.head()
```

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

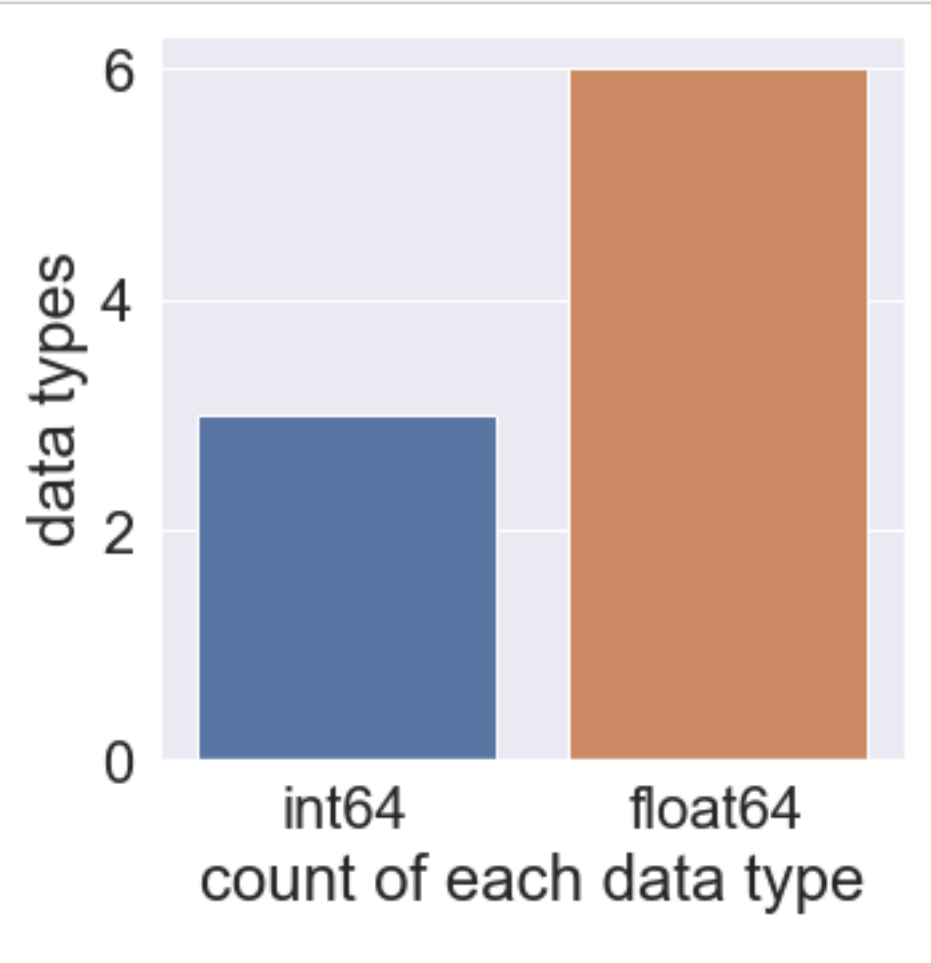
In [36]:
```
df.tail()
```

Out[36]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340	27	0
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349	47	1
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315	23	0

frequency plot of data types and count of variables

In [48]:
```
plt.figure(figsize=(5,5))
sns.set(font_scale=2)
sns.countplot(df.dtypes.map(str))
plt.xlabel("count of each data type")
plt.ylabel("data types")
plt.show()
```



Task 2

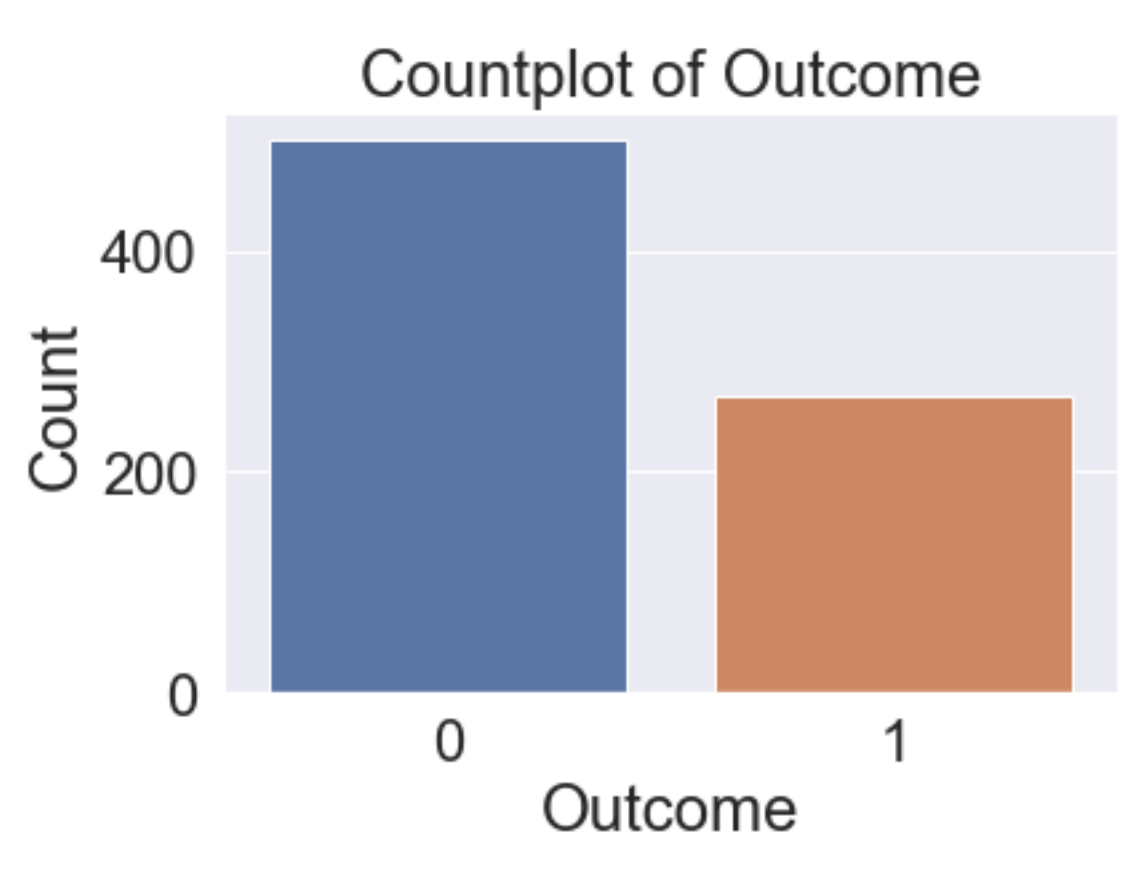
Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.
3. Perform correlation analysis. Visually explore it using a heat map.

Countplot

In [49]:
```
sns.set_style('darkgrid')
sns.countplot(df['Outcome'])
plt.title("Countplot of Outcome")
plt.xlabel('Outcome')
plt.ylabel("Count")
print("Count of class is:\n",df['Outcome'].value_counts())
```

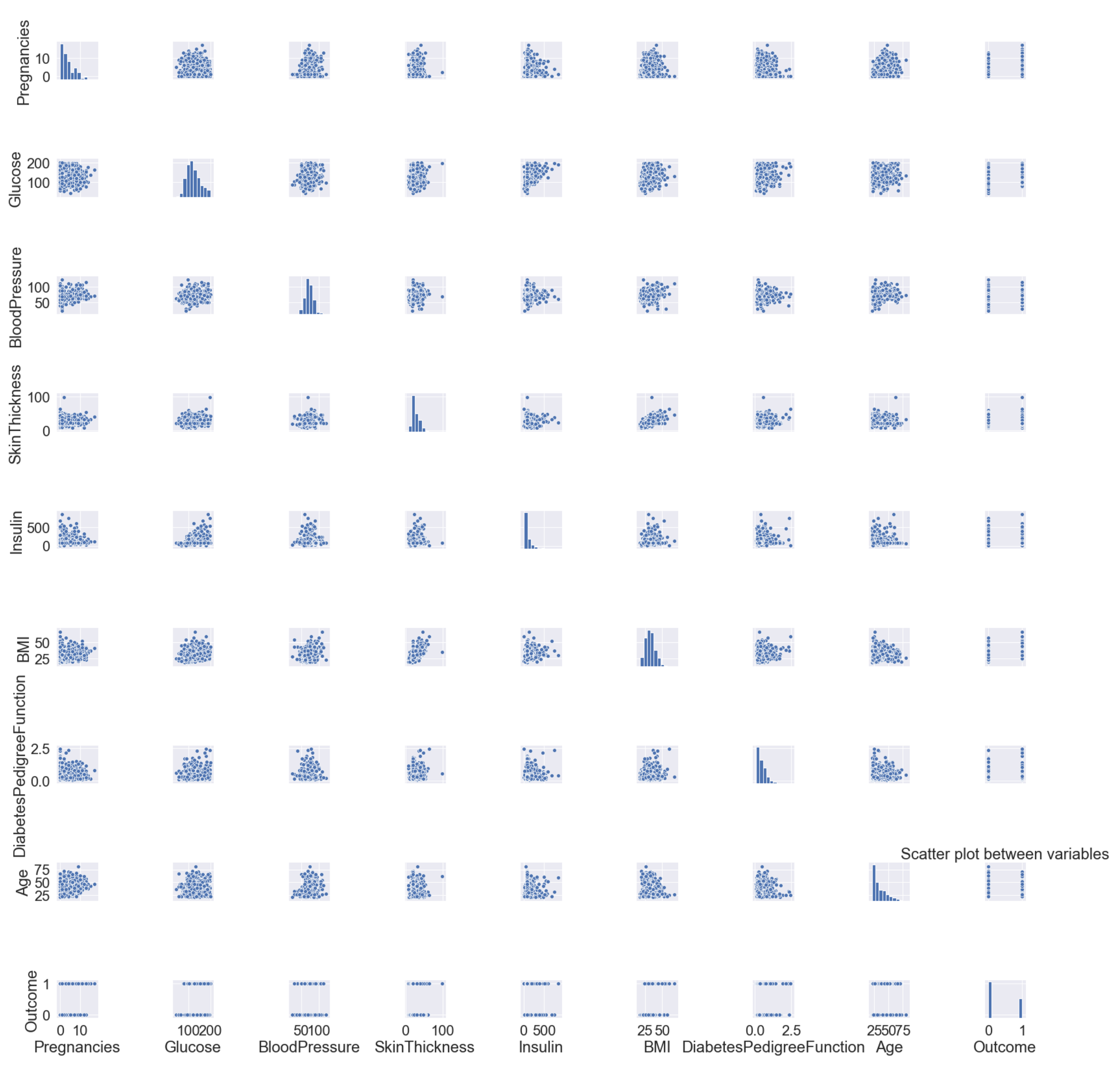
```
Count of class is:
 0    500
1    268
Name: Outcome, dtype: int64
```



Scatter Plot

In [50]:
```
sns.pairplot(df)
plt.title('Scatter plot between variables')
```

Out[50]: Text(0.5, 1, 'Scatter plot between variables')



Correlation Analysis

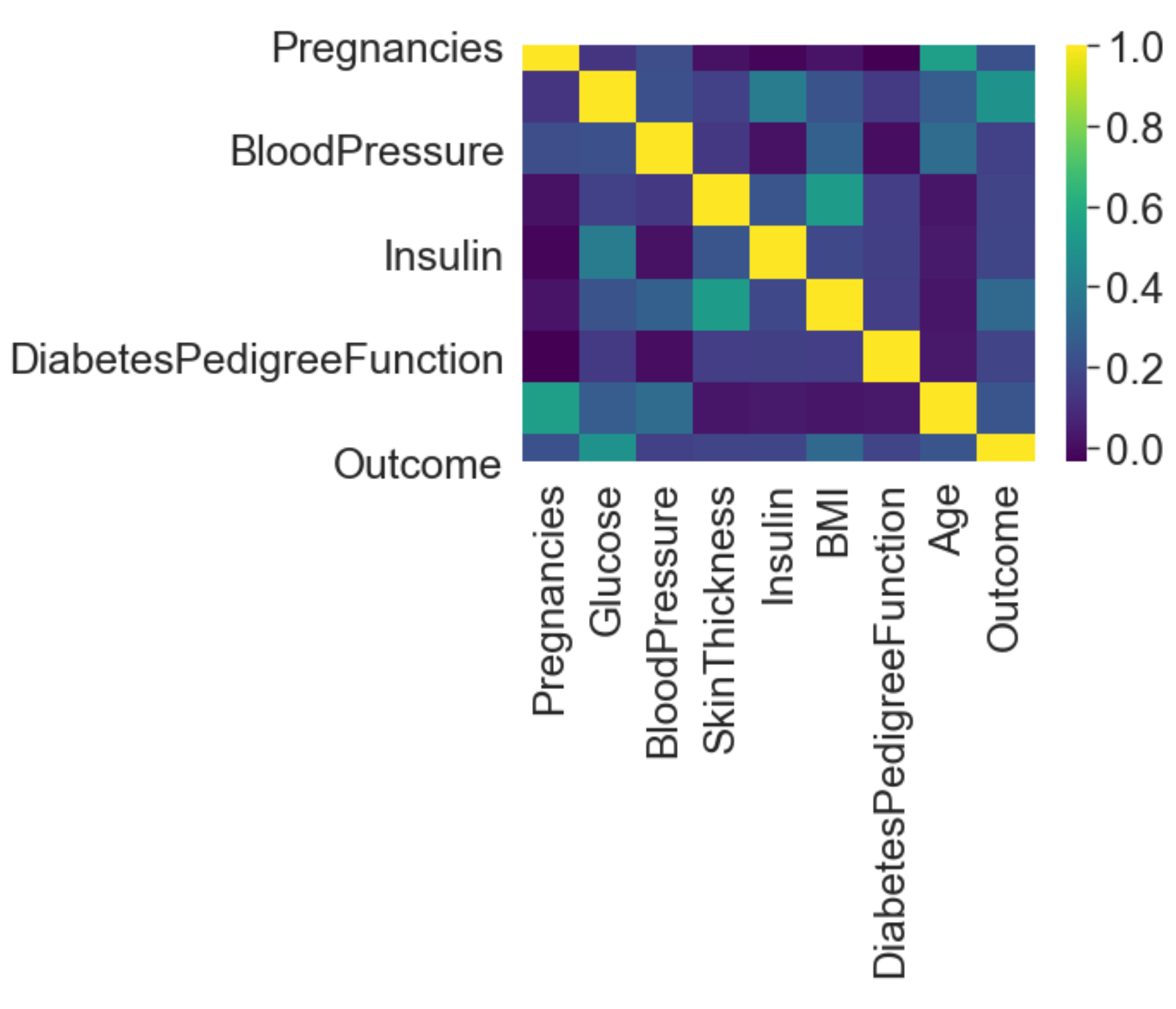
In [51]:
```
df.corr()
```

Out[51]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546	-0.033523	0.544341	0.221898
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106	0.266600	0.492908
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371	0.326740	0.162986
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961	0.026423	0.175026
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806	0.038652	0.179185
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508	0.025748	0.312254
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000	0.033561	0.173844
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561	1.000000	0.238356
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844	0.238356	1.000000

In [52]:
```
plt.figure(dpi=80)
sns.heatmap(df.corr(),cmap='viridis')
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1a15ce9910>



In [ ]: