21bai1217-pat2

June 7, 2023

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

1. Write a program to load the Diabetes Data Set from a given csv file into a dataframe and print the shape of the data, type of the data and first and last 5 rows.

[2]:
```
df=pd.read_csv("/content/diabetes (1).csv")
df
```

[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
..	…	…	…	…	…	…	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	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]
```

[4]:
```
df.shape
```

[4]:
```
(768, 9)
```

[5]:
```
df.dtypes
```

[5]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

[6]:
```
df.head(5)
```

[6]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   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
```

[7]:
```
df.tail(5)
```

[7]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
```

```
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
```

2. Write a program using to print the keys, number of rows-columns, feature names and the description of the Diabetes Data Set

[8]:
```
df.keys
```

[8]:
```
<bound method NDFrame.keys of      Pregnancies  Glucose  BloodPressure
SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …      …               …              …       …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  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]>
```

[9]:
```
print(len(df.index))
```

```
768
```

[10]:
```
print(len(df.columns))
```

```
9
```

[13]:
```
df.describe()
```

[13]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[14]:
```
df.columns
```

[14]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

3. Write a program to convert the data from nd-array to data frame and adding feature names to the data

[15]:
```
a=np.array([[1,3],[4,5],[6,7]])
df_a=pd.DataFrame(a,columns=['first','second'])
df_a
```

[15]:
```
   first  second
0      1       3
1      4       5
2      6       7
```

4. Write a program to get the number of observations, missing values and nan values

[17]:
```
df.info
```

[17]:
```
<bound method DataFrame.info of      Pregnancies  Glucose  BloodPressure
SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
```

```
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  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]>
```

[18]:
```
df.isnull().any()
```

[18]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

[19]:
```
df.isnull().sum()
```

[19]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
```

```
Age                         0
Outcome                     0
dtype: int64
```

5. Write a program to create a 2-D array with ones on the diagonal and zeros elsewhere. Now convert the array to a sparse matrix in CSR format.

[20]:
```
from scipy import sparse
```

[21]:
```
n=int(input())
arr = np.eye(n)
arr
```

```
10
```

[21]:
```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]])
```

[22]:
```
sparse_matrix = sparse.csr_matrix(arr)
sparse_matrix
```

[22]:
```
<10x10 sparse matrix of type '<class 'numpy.float64'>'
        with 10 stored elements in Compressed Sparse Row format>
```

6. Write a program to view basic statistical details like percentile, mean, std etc. of Diabetes Database Data Set.

[24]:
```
df.describe()
```

[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	
std	3.369578	31.972618	19.355807	15.952218	115.244002	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	

	BMI	DiabetesPedigreeFunction	Age	Outcome

```
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

7. Write a program to get observations of each category of diabetic from Diabetes Data

[26]:
```
df['Dibetic']=df['Outcome'].apply(lambda x: "Dibetic" if x==1 else "Non␣
 ↪Dibetic")
df
```

[26]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …      …              …              …       …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

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

[768 rows x 10 columns]
```

8. Write a program to Count zero values per column

[27]:
```
for col in df.columns:
    column = df[col]
    count = (column == 0).sum()
```

```
    print('Count of zeros in column ', col, ' is : ', count)
```

```
Count of zeros in column  Pregnancies  is :  111
Count of zeros in column  Glucose  is :  5
Count of zeros in column  BloodPressure  is :  35
Count of zeros in column  SkinThickness  is :  227
Count of zeros in column  Insulin  is :  374
Count of zeros in column  BMI  is :  11
Count of zeros in column  DiabetesPedigreeFunction  is :  0
Count of zeros in column  Age  is :  0
Count of zeros in column  Outcome  is :  500
Count of zeros in column  Dibetic  is :  0
```

9. Write a program to Determine correlation between variables

[28]:
```
df.corr()
```

[28]:

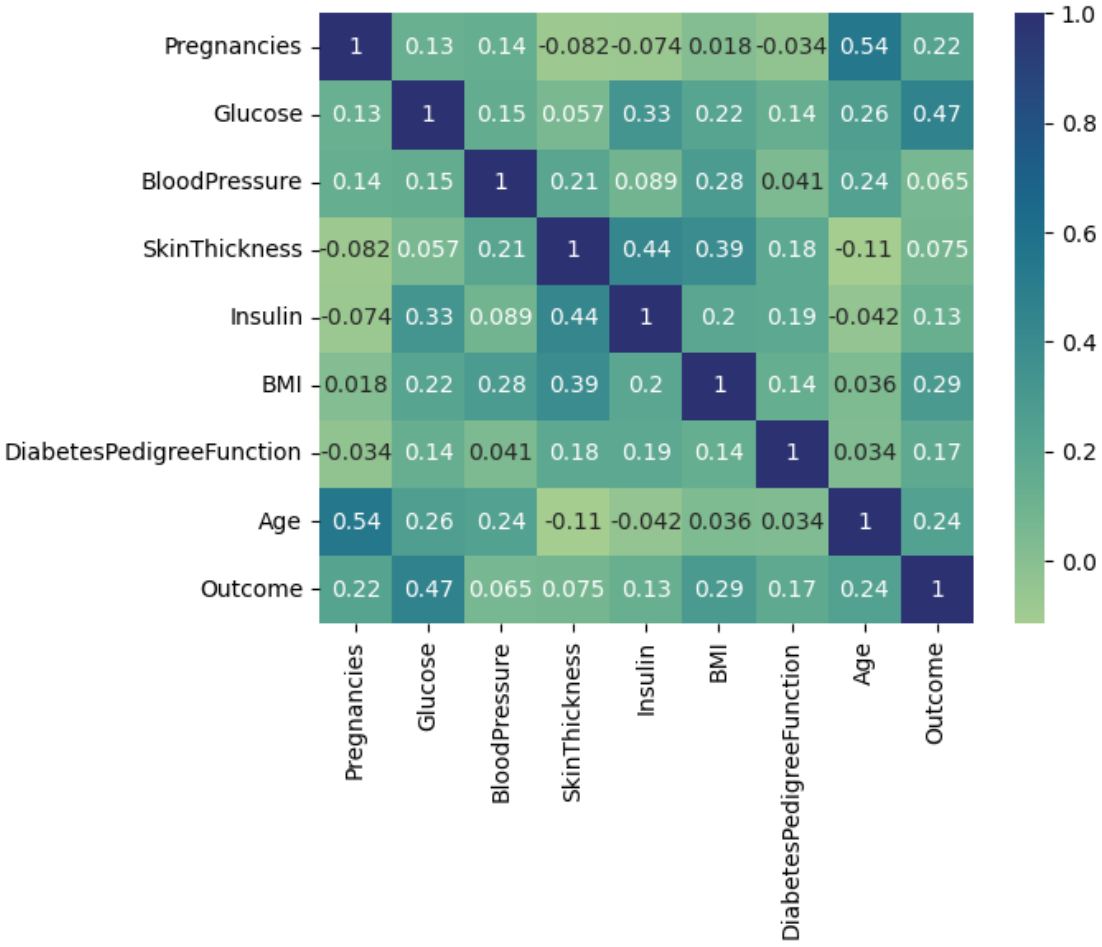
	Pregnancies	Glucose	BloodPressure	SkinThickness
Pregnancies	1.000000	0.129459	0.141282	-0.081672
Glucose	0.129459	1.000000	0.152590	0.057328
BloodPressure	0.141282	0.152590	1.000000	0.207371
SkinThickness	-0.081672	0.057328	0.207371	1.000000
Insulin	-0.073535	0.331357	0.088933	0.436783
BMI	0.017683	0.221071	0.281805	0.392573
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928
Age	0.544341	0.263514	0.239528	-0.113970
Outcome	0.221898	0.466581	0.065068	0.074752

	Insulin	BMI	DiabetesPedigreeFunction
Pregnancies	-0.073535	0.017683	-0.033523
Glucose	0.331357	0.221071	0.137337
BloodPressure	0.088933	0.281805	0.041265
SkinThickness	0.436783	0.392573	0.183928
Insulin	1.000000	0.197859	0.185071
BMI	0.197859	1.000000	0.140647
DiabetesPedigreeFunction	0.185071	0.140647	1.000000
Age	-0.042163	0.036242	0.033561
Outcome	0.130548	0.292695	0.173844

	Age	Outcome
Pregnancies	0.544341	0.221898
Glucose	0.263514	0.466581
BloodPressure	0.239528	0.065068
SkinThickness	-0.113970	0.074752
Insulin	-0.042163	0.130548
BMI	0.036242	0.292695
DiabetesPedigreeFunction	0.033561	0.173844
Age	1.000000	0.238356

```
Outcome                   0.238356   1.000000
```

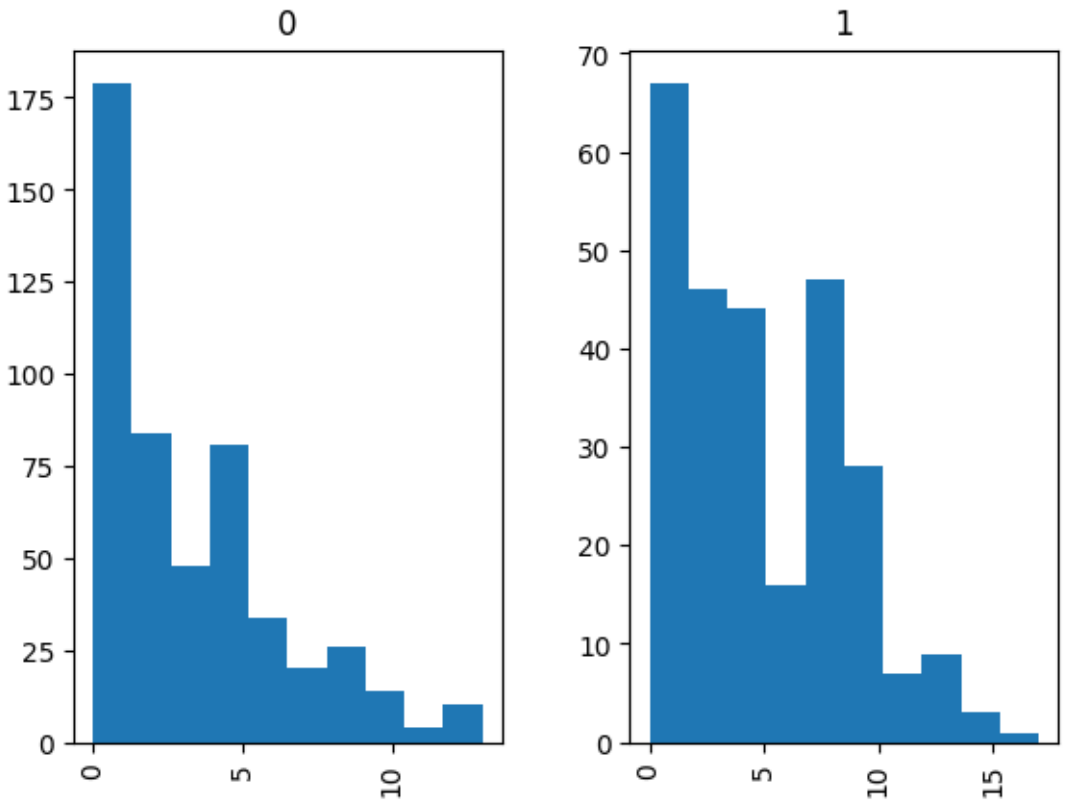
[34]:
```
import seaborn as sns
sns.heatmap(df.corr(), annot=True, cmap = 'crest')
plt.show()
```



10. Write a program to Plot histogram for Outcome vs Pregnancies

[30]:
```
df['Pregnancies'].hist(by=df['Outcome'])
```

[30]:
```
array([<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>],
      dtype=object)
```



11. Create new dataframe wherein the unwanted rows are not included

[33]:
```
df.duplicated()
```

[33]:
```
0      False
1      False
2      False
3      False
4      False
       …
763    False
764    False
765    False
766    False
767    False
Length: 768, dtype: bool
```

12. Split the dataset into training and test sets

[35]:
```
from sklearn.model_selection import train_test_split
X=df.drop(['Dibetic','Outcome'],axis=1)
y=df['Outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state=42)
```

[36]:
```
X_train
```

[36]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
60             2       84              0              0        0   0.0
618            9      112             82             24        0  28.2
346            1      139             46             19       83  28.7
294            0      161             50              0        0  21.9
231            6      134             80             37      370  46.2
..           ...      ...            ...            ...      ...  ...
71             5      139             64             35      140  28.6
106            1       96            122              0        0  22.4
270           10      101             86             37        0  45.6
435            0      141              0              0        0  42.4
102            0      125             96              0        0  22.5

     DiabetesPedigreeFunction  Age
60                      0.304   21
618                     1.282   50
346                     0.654   22
294                     0.254   65
231                     0.238   46
..                        ...  ...
71                      0.411   26
106                     0.207   27
270                     1.136   38
435                     0.205   29
102                     0.262   21

[614 rows x 8 columns]
```

[37]:
```
X_test
```

[37]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
668            6       98             58             33      190  34.0
324            2      112             75             32        0  35.7
624            2      108             64              0        0  30.8
690            8      107             80              0        0  24.6
473            7      136             90              0        0  29.9
..           ...      ...            ...            ...      ...  ...
355            9      165             88              0        0  30.4
534            1       77             56             30       56  33.3
344            8       95             72              0        0  36.8
296            2      146             70             38      360  28.0
462            8       74             70             40       49  35.3
```

```
     DiabetesPedigreeFunction  Age
668                      0.430   43
324                      0.148   21
624                      0.158   21
690                      0.856   34
473                      0.210   50
..                         …  …
355                      0.302   49
534                      1.251   24
344                      0.485   57
296                      0.337   29
462                      0.705   39

[154 rows x 8 columns]
```

[38]:
```
y_train
```

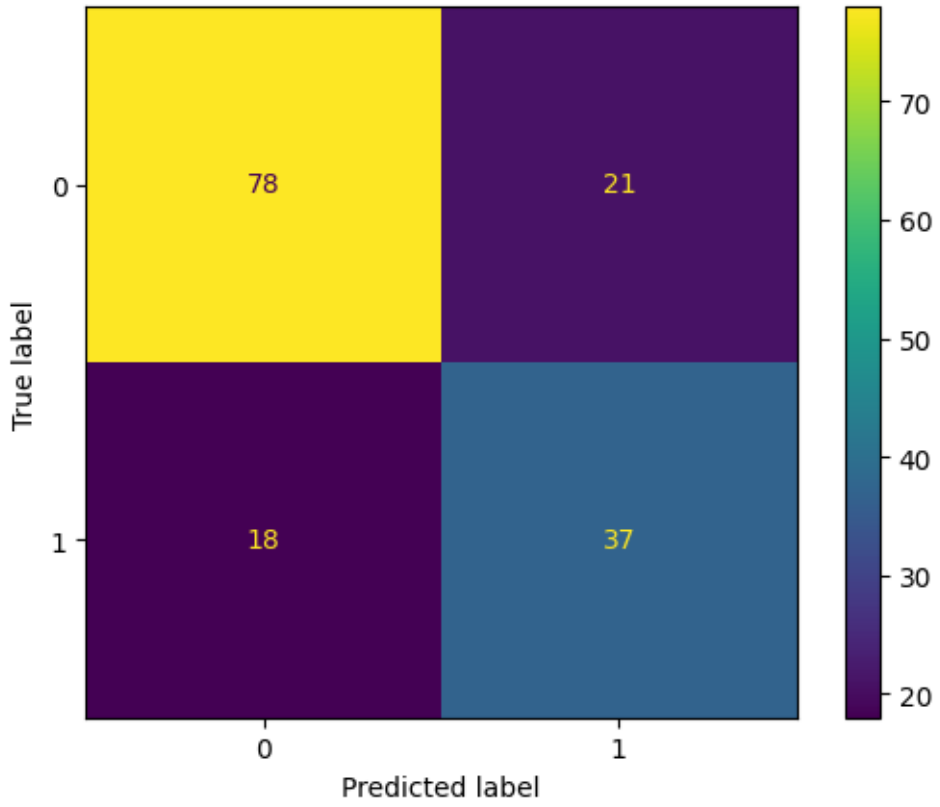
[38]:
```
60     0
618    1
346    0
294    0
231    1
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 614, dtype: int64
```

[39]:
```
y_test
```

[39]:
```
668    0
324    0
624    0
690    0
473    0
      ..
355    1
534    0
344    0
296    1
462    0
Name: Outcome, Length: 154, dtype: int64
```

13. Plot the confusion matrix

[40]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
clf = LogisticRegression(random_state=42)
scores = cross_val_score(clf, X, y, cv=5)
scores
clf.fit(X_train,y_train)
y_pred_class = clf.predict(X_test)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
cm = confusion_matrix(y_test, y_pred_class, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=clf.classes_)
disp.plot()
plt.show()
```



14. Get performance metrics

[41]:
```
print(accuracy_score(y_test,y_pred_class))
```

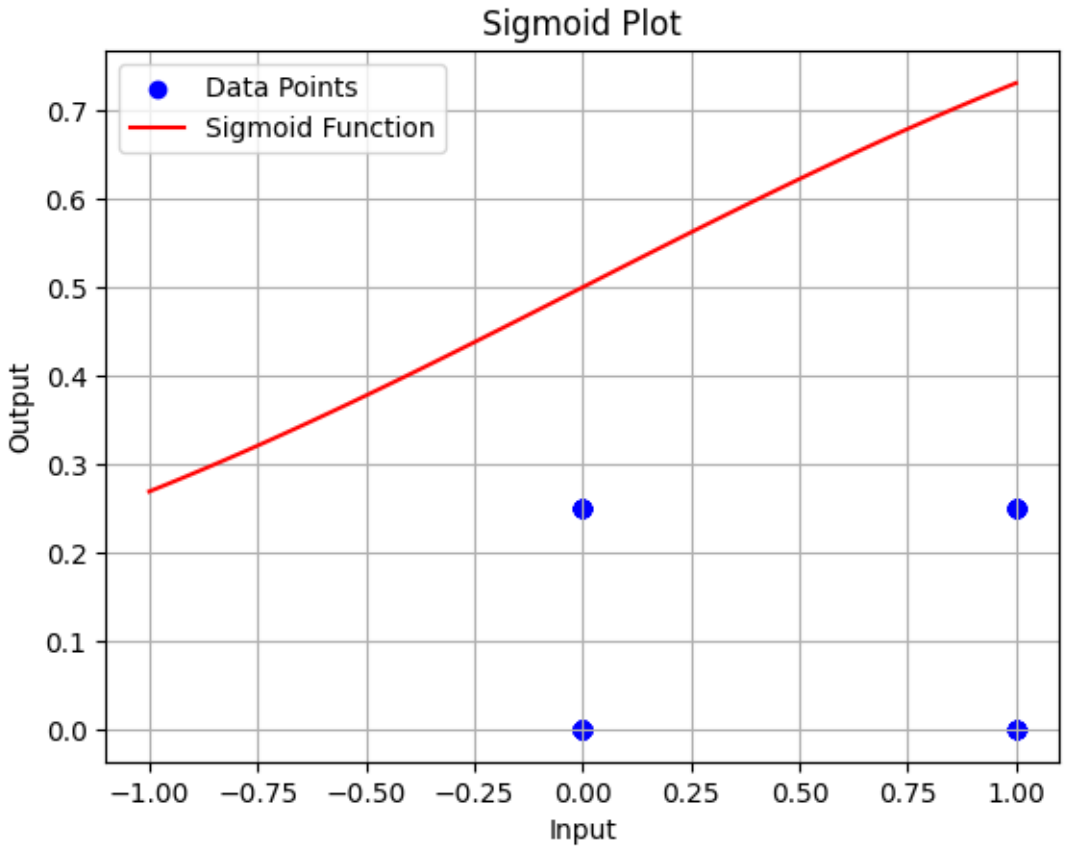
```
0.7467532467532467
```

15. What kind of relationship do you see? e.g. positive, negative? linear? non-linear? Is there

anything else strange or interesting about the data? What about outliers?

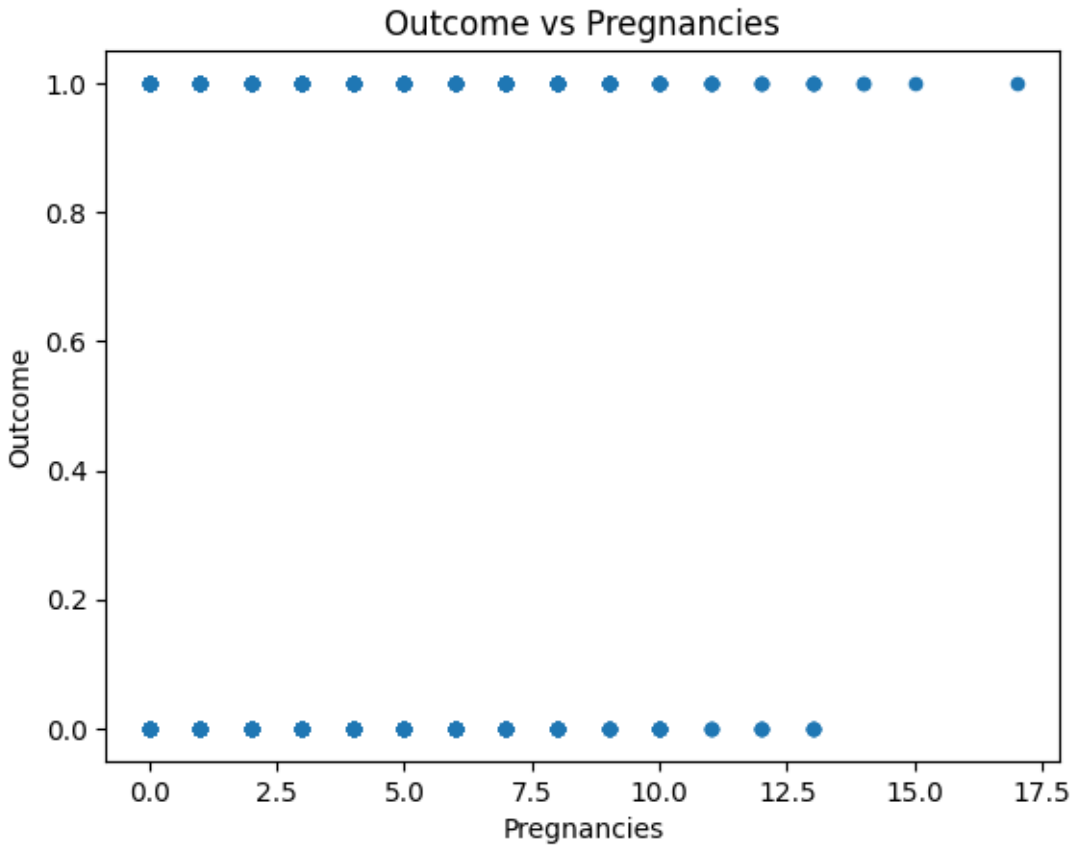
[42]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Plotting
plt.scatter(y_test, y_pred_class/4, color='blue', label='Data Points')#we␣
 ↪normalize the points as x/4
x = np.linspace(-(min(y_test)+max(y_test)), min(y_test)+max(y_test), 100)
y = sigmoid(x)
plt.plot(x, y, color='red', label='Sigmoid Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.title('Sigmoid Plot')
plt.legend()
plt.grid(True)
plt.show()
```

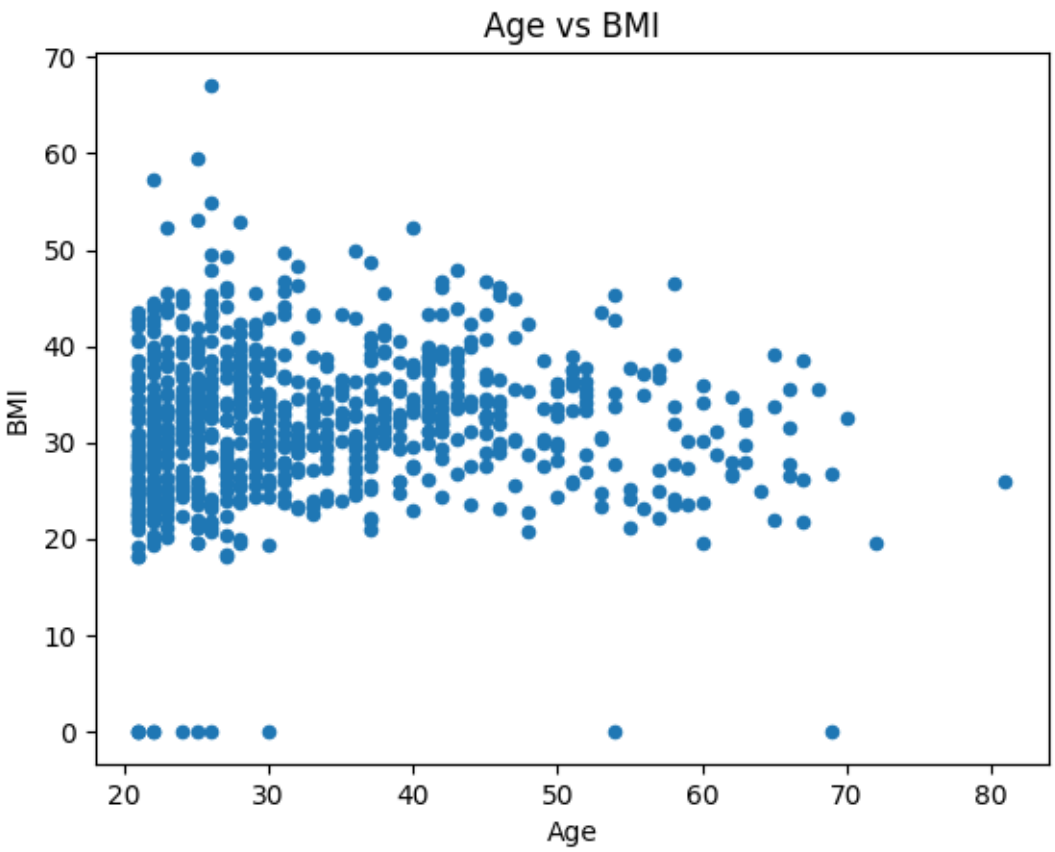


16. Create scatter plots between * Outcome * and * Pregnancies , *and* BMI * and * Age *. Label your axes appropriately using human readable labels. Tell a story about what you see

[46]:
```
df.plot.scatter(x='Pregnancies',y='Outcome',title='Outcome vs Pregnancies')
plt.show()
```

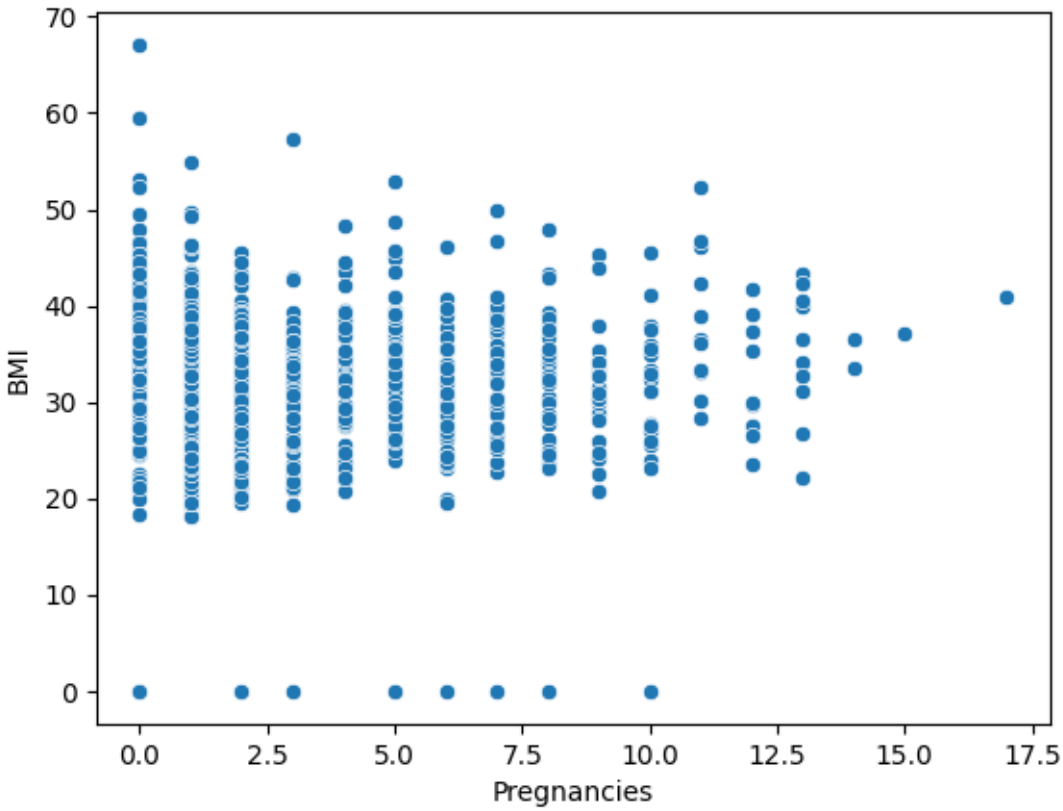


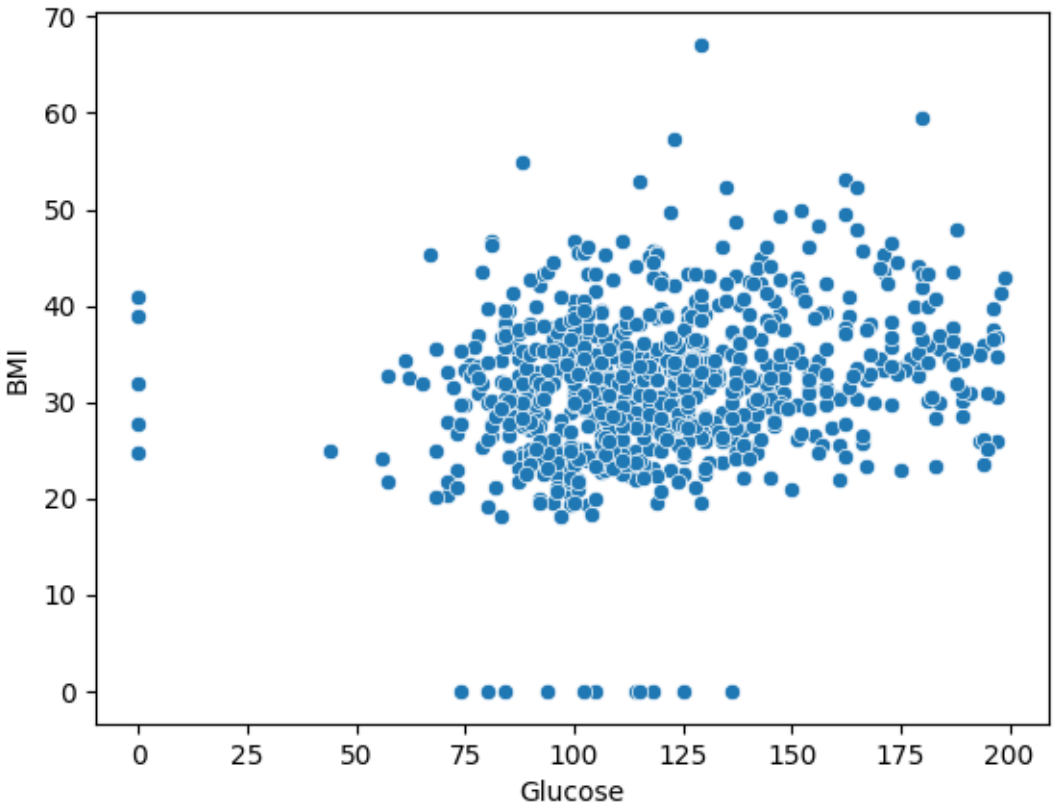
[47]:
```
df.plot.scatter(x='Age',y='BMI',title='Age vs BMI')
plt.show()
```

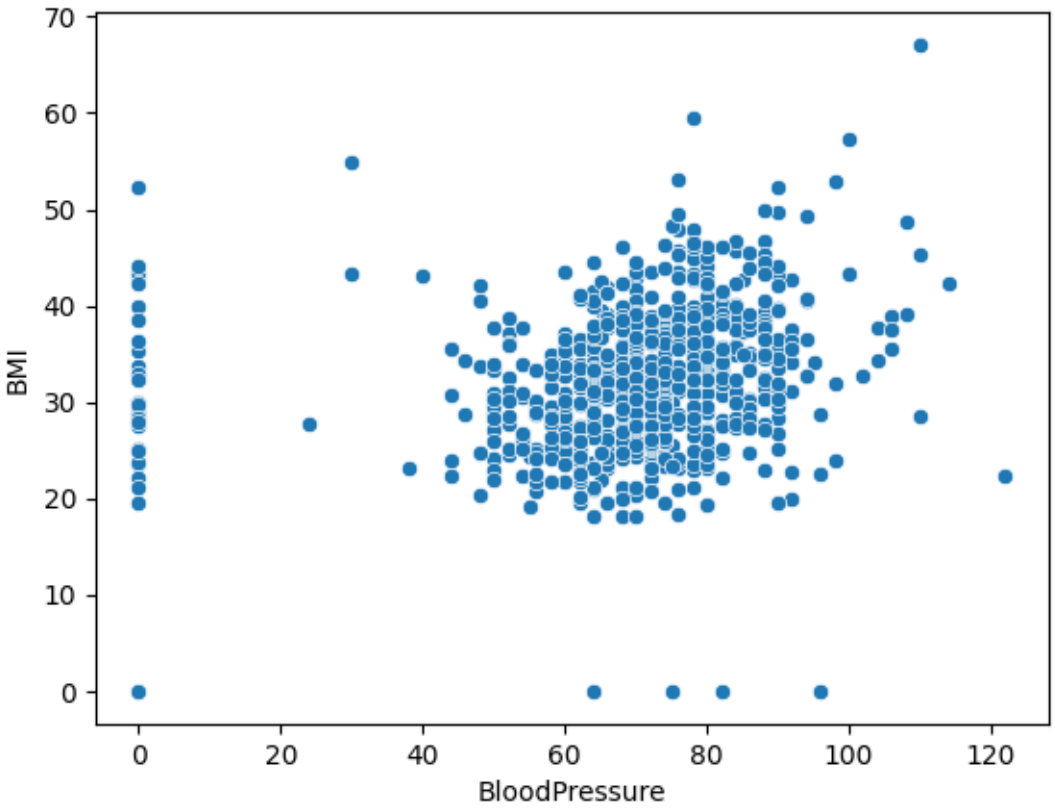


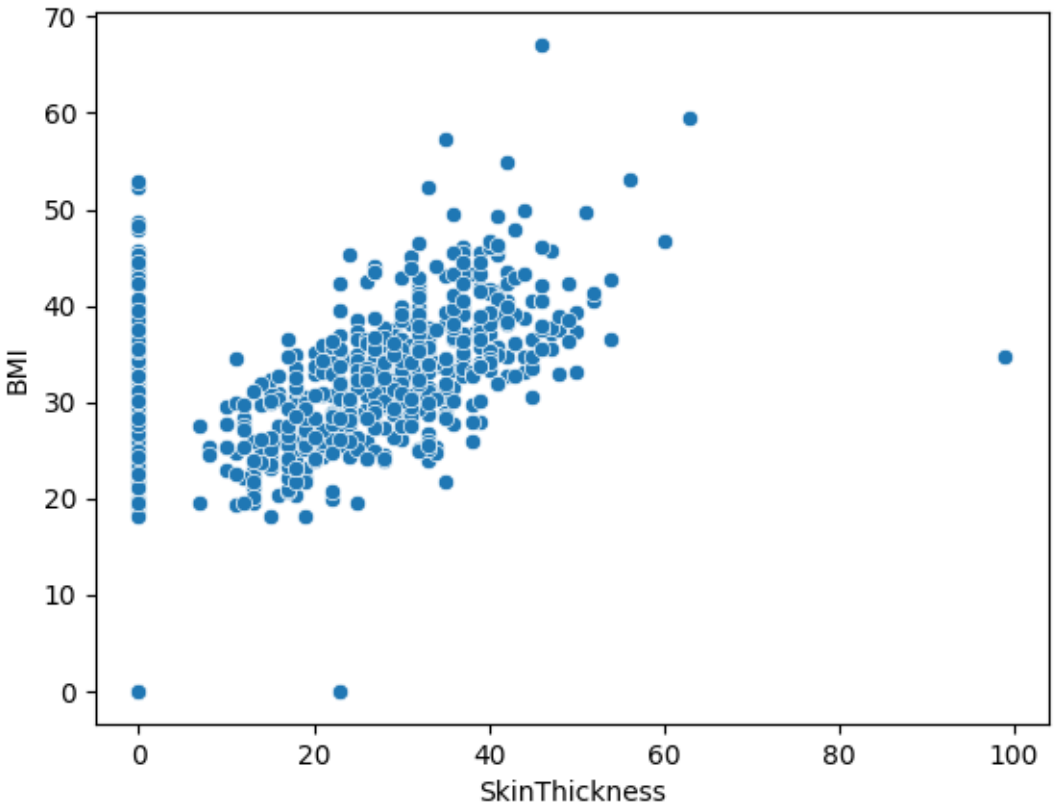
17. What are some other numeric variables of interest? Why do you think they are interesting? Plot scatterplots with these variables and * BMI * and tell a story about what you see

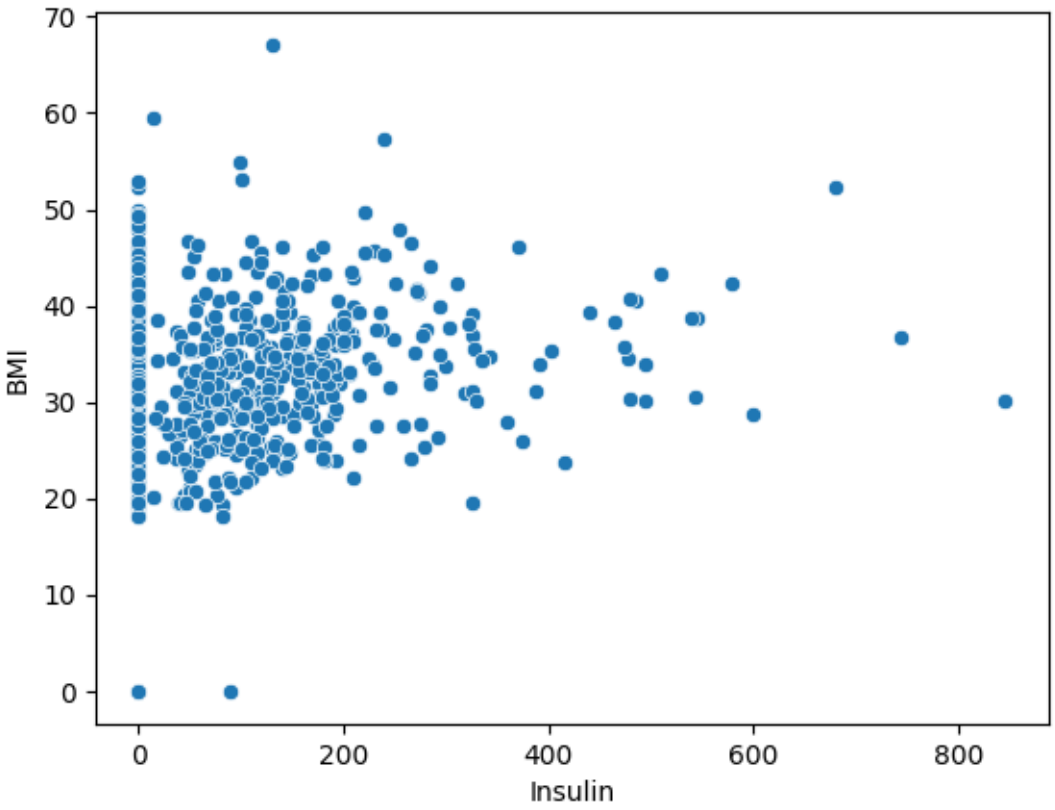
[56]:
```
for col in df.columns:
  if df[col].dtypes =='int64':
    sns.scatterplot(data=df,x=col,y='BMI')
    plt.show()
```

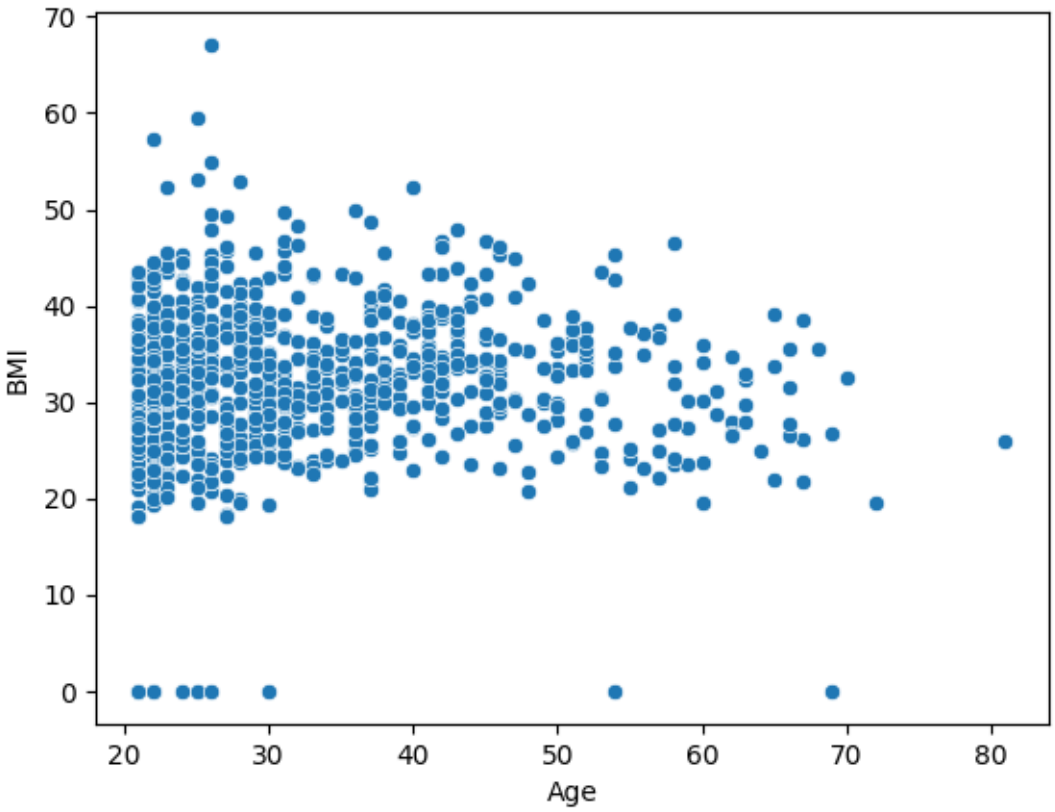

70
60
50
40
30
20
10
0
BMI
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pregnancies

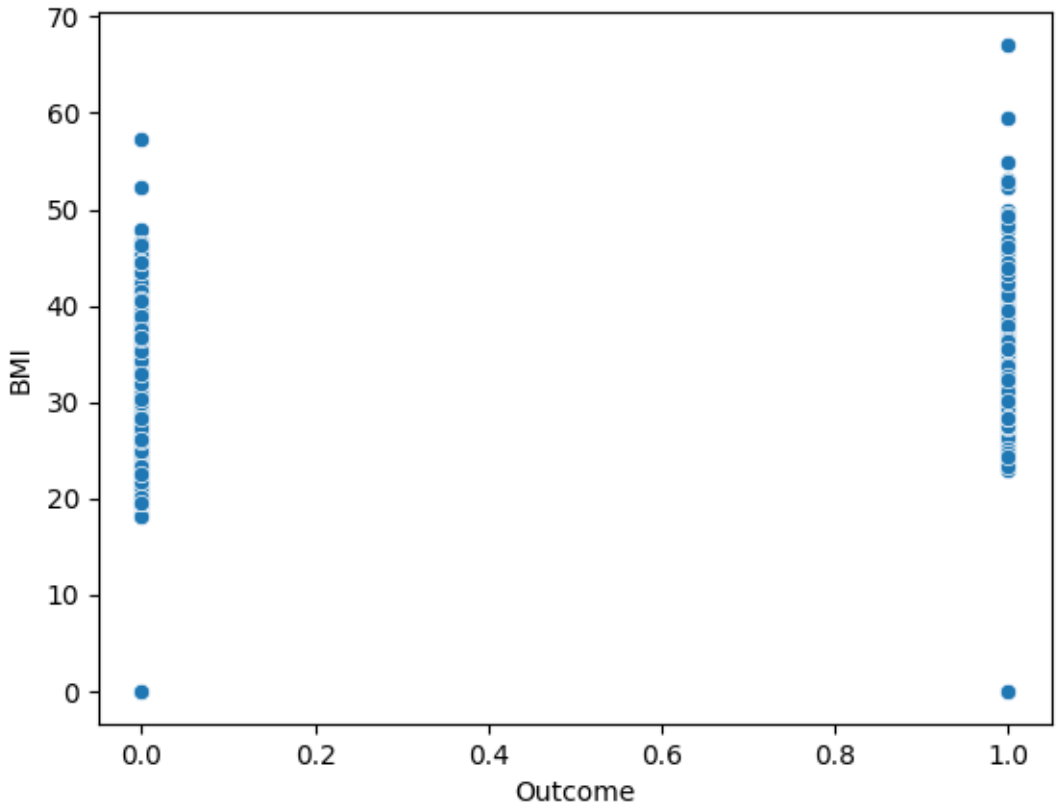

70
60
50
40
30
20
10
0
BMI
0
25
50
75
100
125
150
175
200
Glucose


70
60
50
40
30
20
10
0
BMI
0
20
40
60
80
100
120
BloodPressure


70
60
50
40
30
20
10
0
BMI
0
20
40
60
80
100
SkinThickness


70
60
50
40
30
20
10
0
BMI
0
200
400
600
800
Insulin


70
60
50
40
30
20
10
0
BMI
20
30
40
50
60
70
80
Age



18. Different kinds of classes in every categorical column

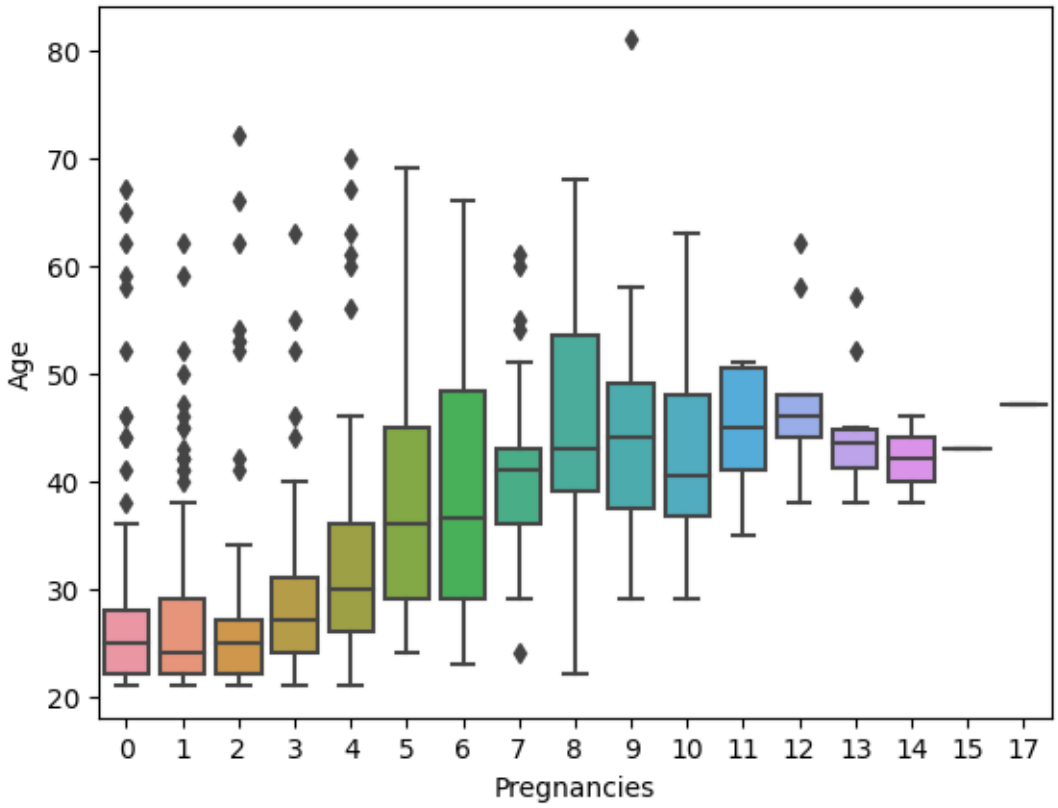
```
[55]: df['Outcome'].values
```

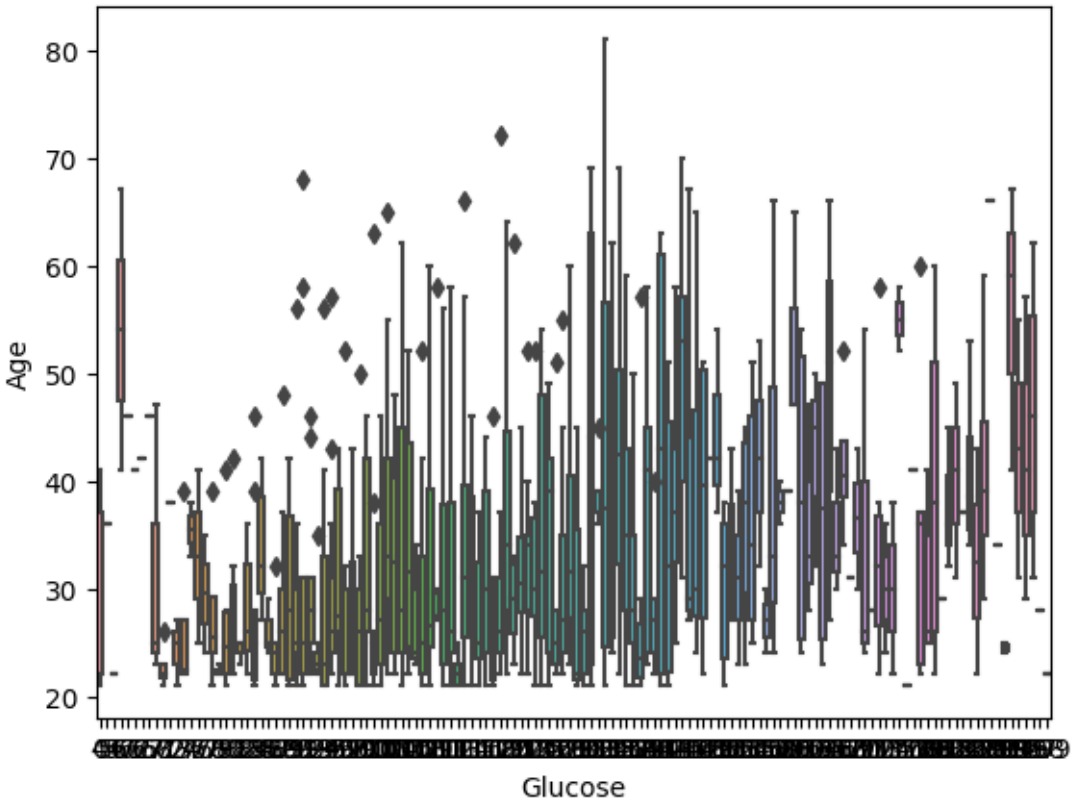
```
[55]: array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
             1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
             0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
             1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
             1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
             1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
             1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
             1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
             0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
             1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
             1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
             1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
             1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
             0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
             1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
             0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
             0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
```

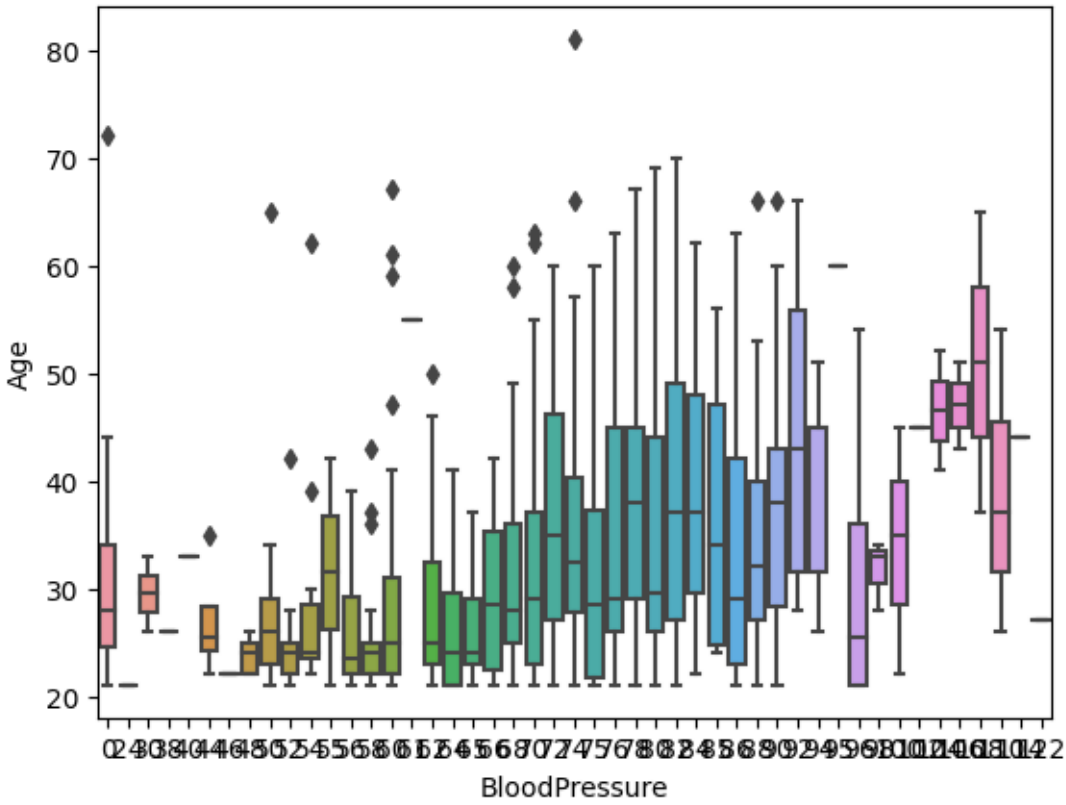
```
0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

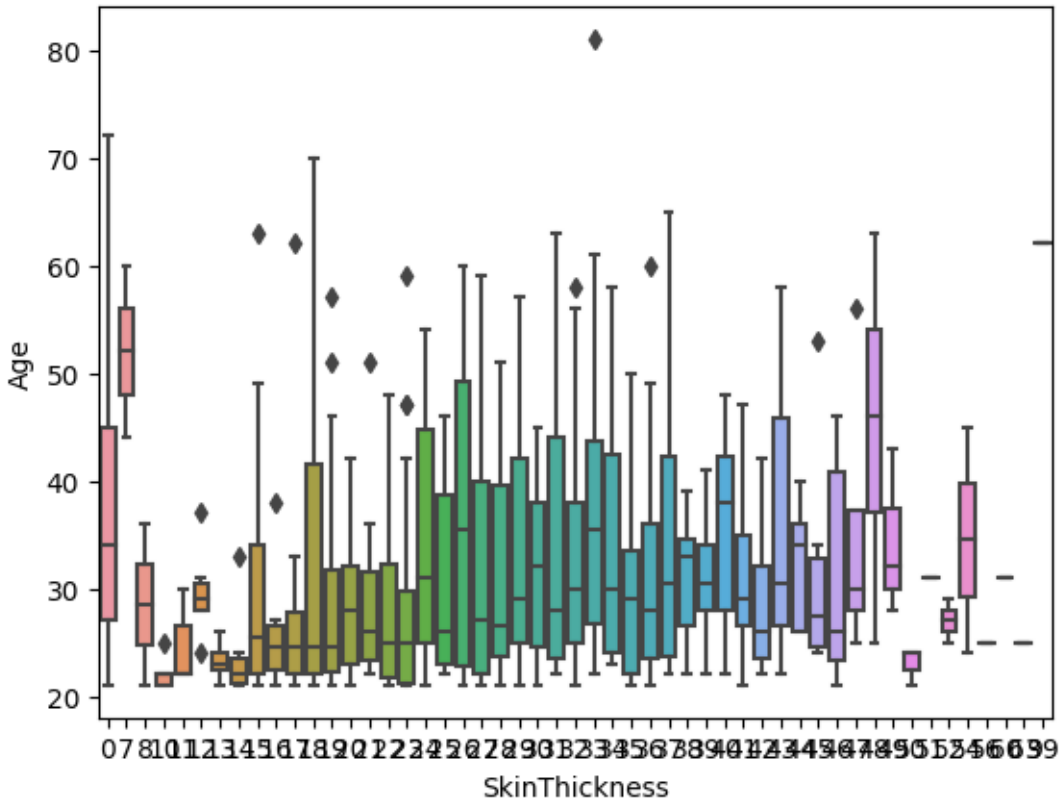
19. Write a code to draw the Boxplots Grouped by Age

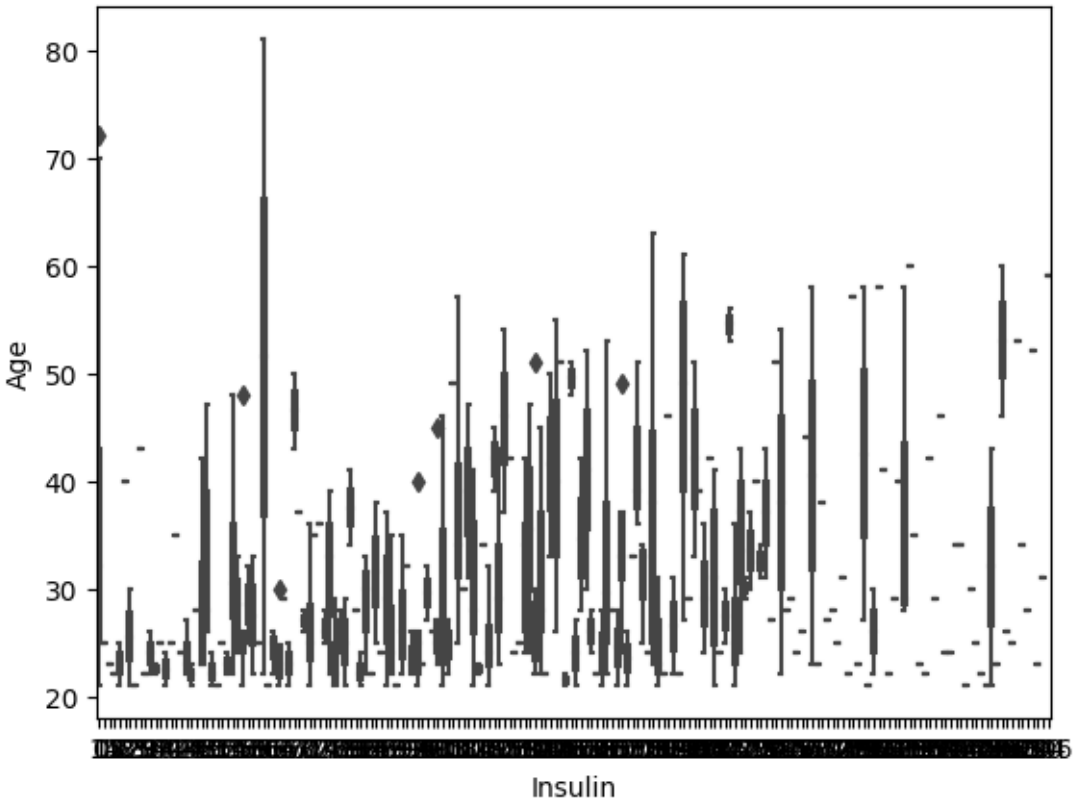
[60]:
```
figsize=(20,10)
for col in df.columns:
    sns.boxplot(data=df,x=col,y='Age')
    plt.show()
```

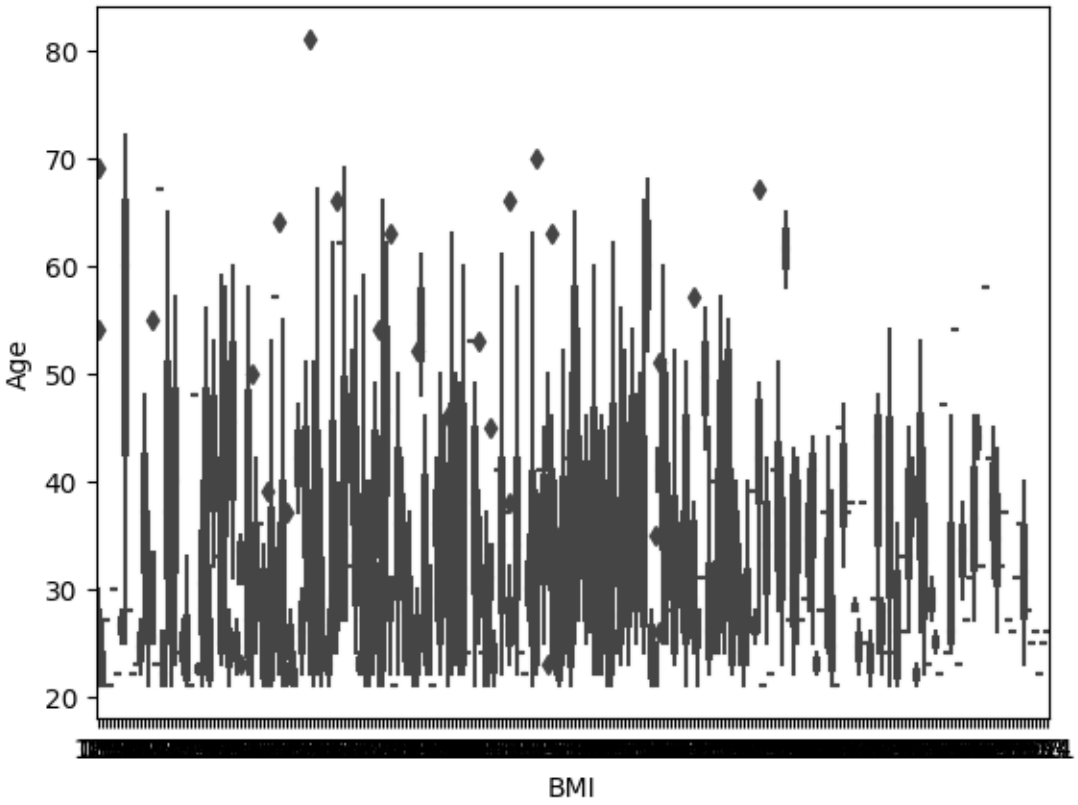

Age
80
70
60
50
40
30
20
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
17
Pregnancies

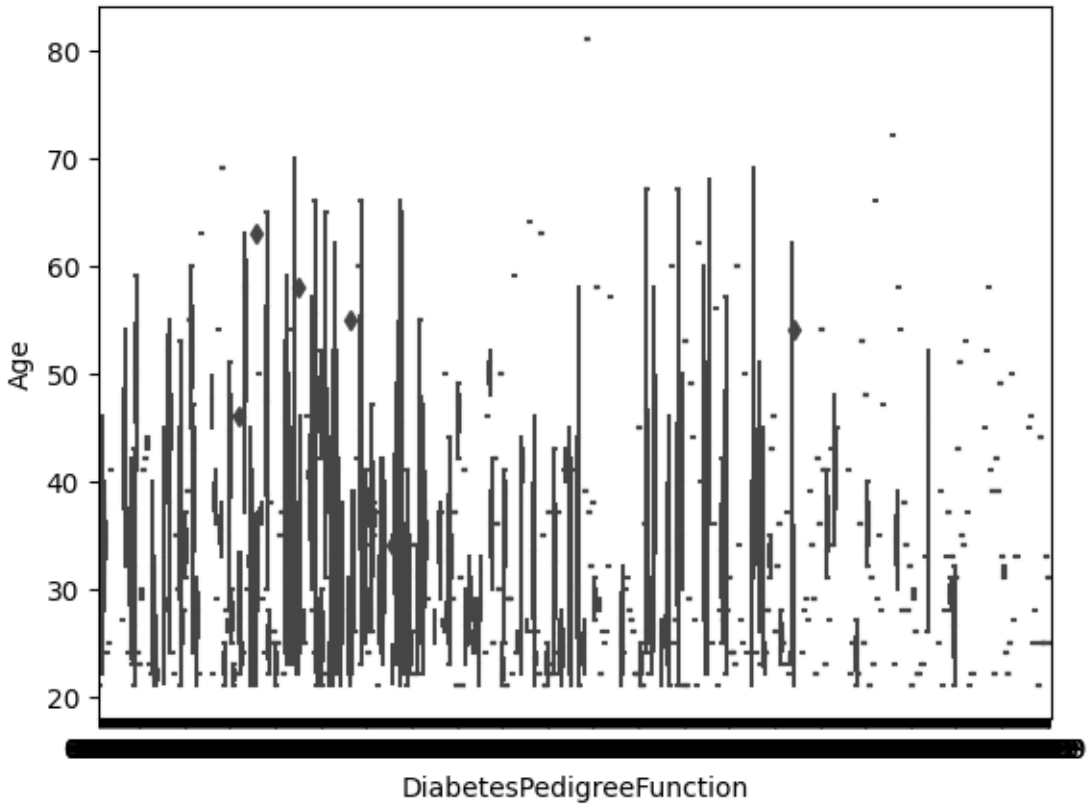

80
70
60
50
40
30
20
Age
Glucose

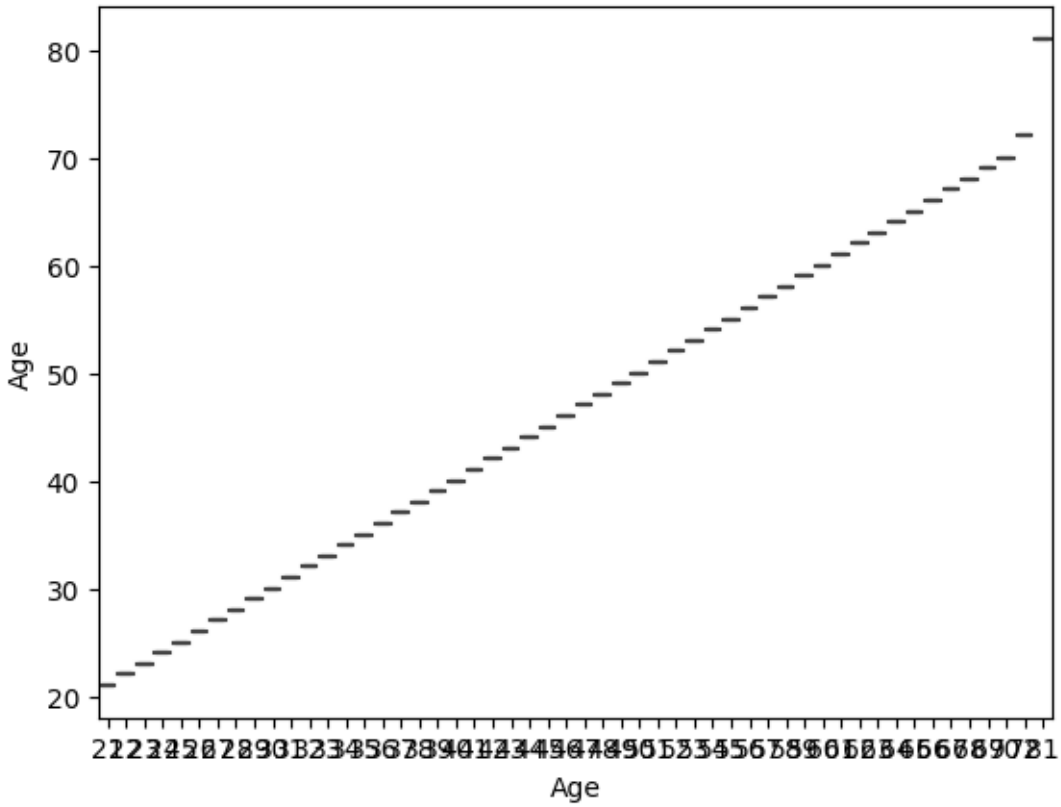

80
70
60
50
40
30
20
Age
BloodPressure

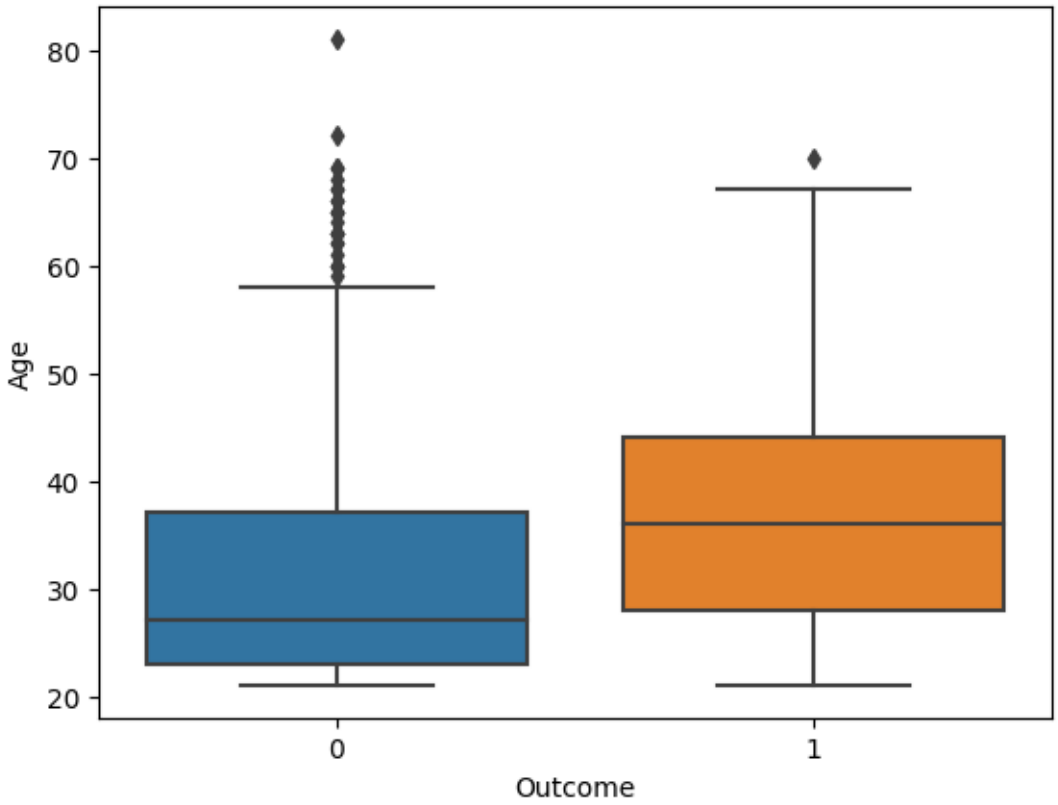

Age
80
70
60
50
40
30
20
SkinThickness

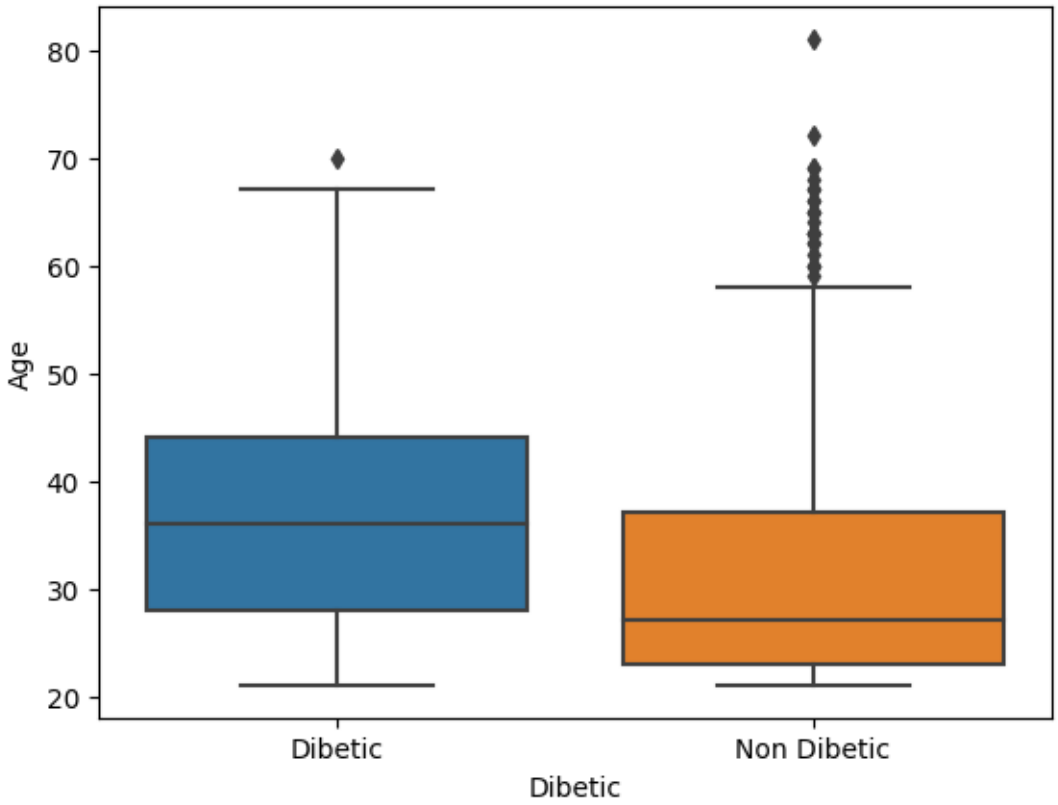

80
70
60
50
40
30
20
Age
Insulin


80
70
60
50
40
30
20
Age
BMI


80
70
60
50
40
30
20
Age
DiabetesPedigreeFunction


80
70
60
50
40
30
20
Age
Age

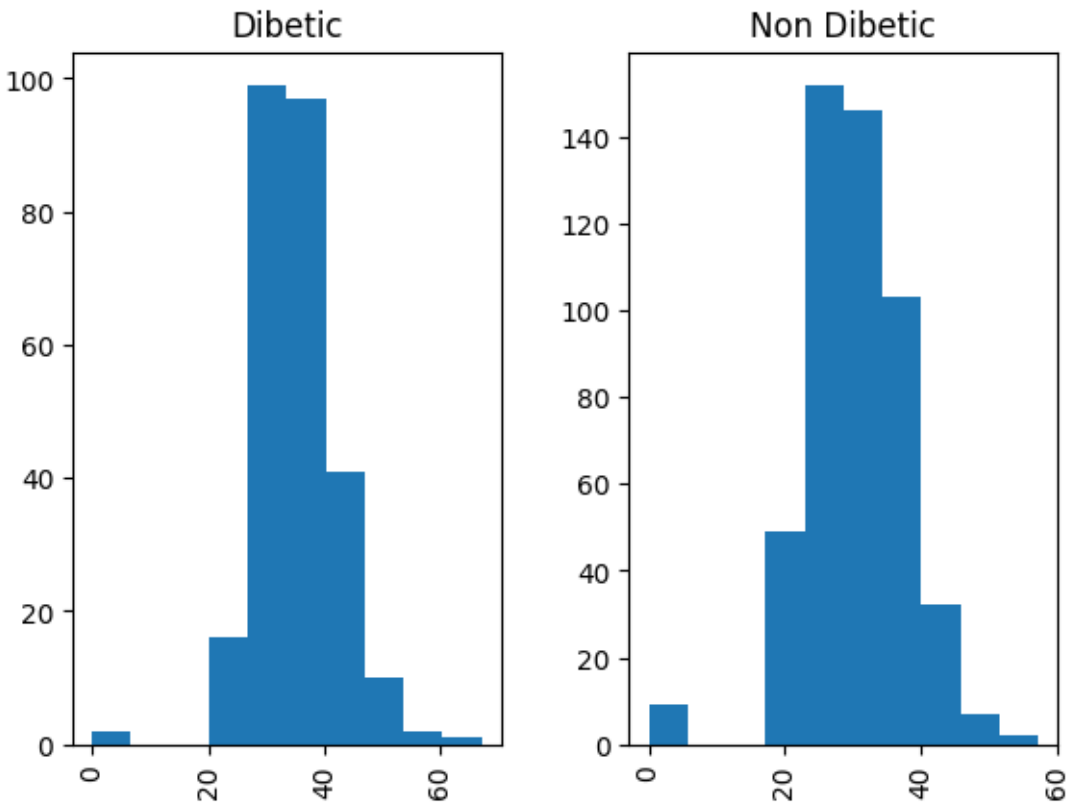

80
70
60
50
40
30
20
Age
0
1
Outcome



20. Write a code to draw the Histograms Grouped by BMI

[77]:
```
df['BMI'].hist(by=df['Dibetic'])
```

[77]:
```
array([<Axes: title={'center': 'Dibetic'}>,
       <Axes: title={'center': 'Non Dibetic'}>], dtype=object)
```



21. Check any NULL value in the dataset

[67]:
```
df.isnull().sum()
```

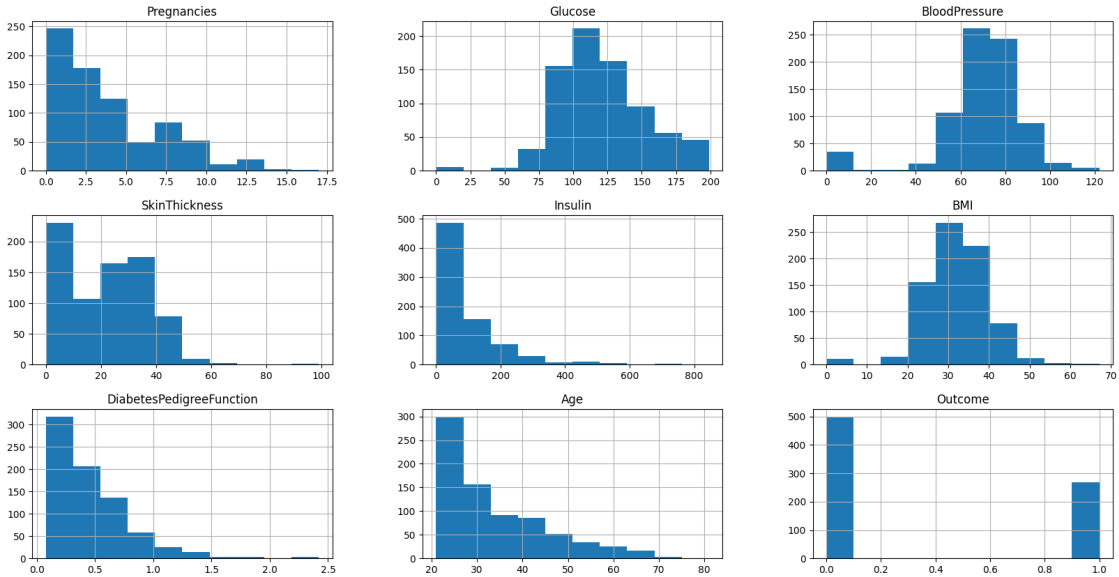
[67]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
Dibetic                     0
dtype: int64
```

22. histogram of each columns data

[71]:
```
df.hist(figsize=(20,10))
```

[71]:
```
array([[<Axes: title={'center': 'Pregnancies'}>,
        <Axes: title={'center': 'Glucose'}>,
```

```
 <Axes: title={'center': 'BloodPressure'}>],
[<Axes: title={'center': 'SkinThickness'}>,
 <Axes: title={'center': 'Insulin'}>,
 <Axes: title={'center': 'BMI'}>],
[<Axes: title={'center': 'DiabetesPedigreeFunction'}>,
 <Axes: title={'center': 'Age'}>,
 <Axes: title={'center': 'Outcome'}>]], dtype=object)
```



23. Finding Outliers

[72]:
```
outliers=[]
def detect_outlier(data_1):

    threshold=3
    mean_1 = np.mean(data_1)
    std_1 =np.std(data_1)


    for y in data_1:
        z_score= (y - mean_1)/std_1
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
```

[75]:
```
for col in X_train.columns:
    column_name=col
    outlier_datapoints = detect_outlier(df[column_name])
    print(outlier_datapoints)
```

[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 99]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600,
440, 540, 480, 510]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600,
440, 540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0,
57.3, 0.0, 0.0]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600,
440, 540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0,
57.3, 0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42,
1.699, 1.698]
[69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600, 440,
540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0, 57.3,
0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42, 1.699,
1.698, 69, 72, 81, 70, 69, 15, 17, 14, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 99, 543, 846, 495, 485, 495, 478, 744, 680, 545, 465, 579, 474, 480, 600,
440, 540, 480, 510, 0.0, 0.0, 0.0, 0.0, 0.0, 67.1, 0.0, 0.0, 59.4, 0.0, 0.0,
57.3, 0.0, 0.0, 2.288, 1.893, 1.781, 2.329, 1.476, 2.137, 1.731, 1.6, 2.42,
1.699, 1.698, 69, 72, 81, 70, 69]