In [2]:

```
import numpy as np
import pandas as pd
from collections import Counter
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
df=pd.read_csv("diabetes.csv")
```

In [4]:

```
df
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [6]:

```
df.columns
```

Out[6]:

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

In [7]:

```
df.head(10)
```

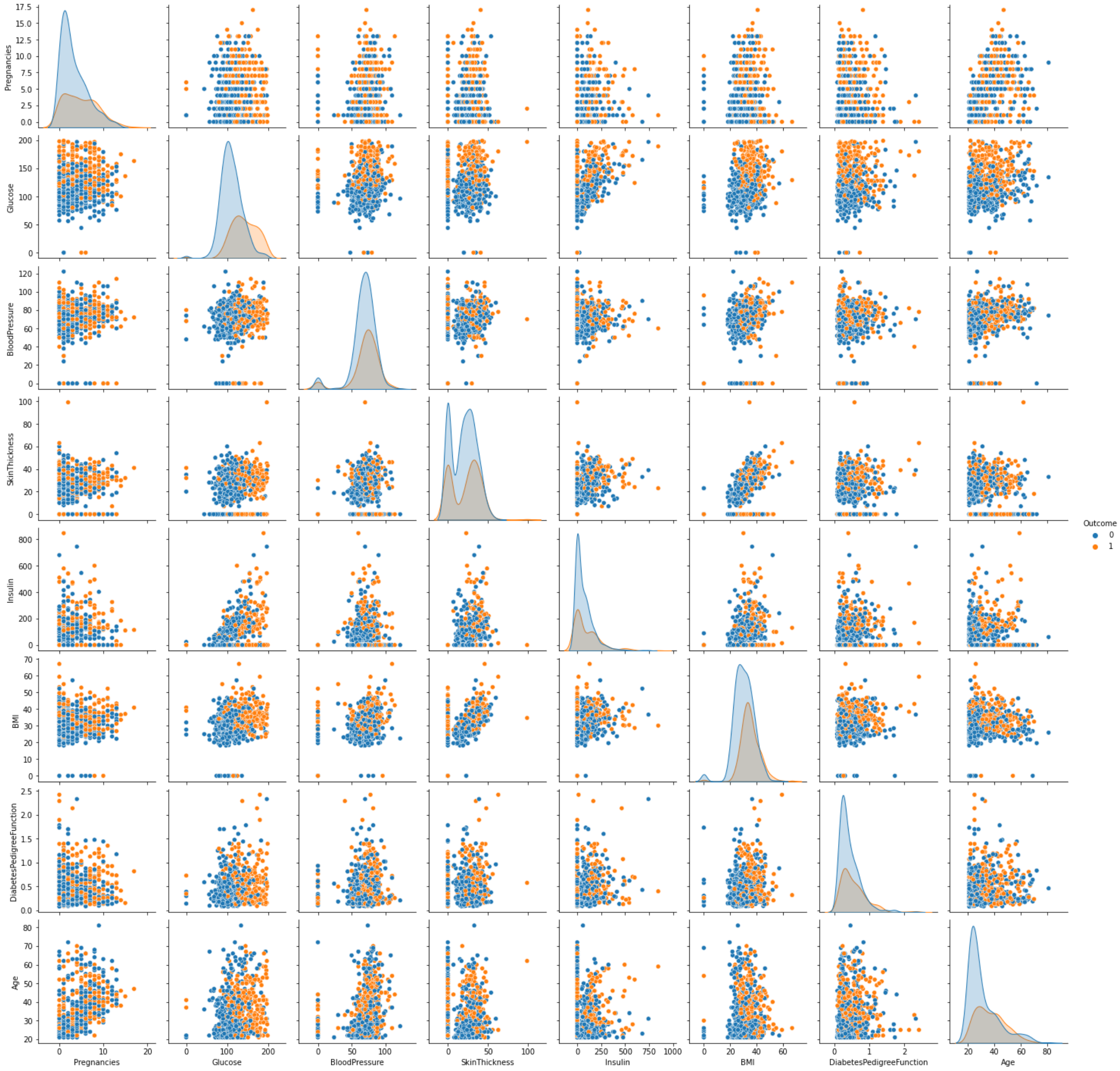
Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	
5	5	116	74	0	0	25.6	0.201	
6	3	78	50	32	88	31.0	0.248	
7	10	115	0	0	0	35.3	0.134	
8	2	197	70	45	543	30.5	0.158	
9	8	125	96	0	0	0.0	0.232	

Data Visualization

In [8]:
```
sns.pairplot(df,hue="Outcome")
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1b29630f4f0>
```



```
In [9]: df.describe().T
```

```
Out[9]:
                           count        mean         std     min       25%       50%        75%
              Pregnancies  768.0    3.845052    3.369578   0.000   1.00000    3.0000    6.00000   1
                  Glucose  768.0  120.894531   31.972618   0.000  99.00000  117.0000  140.25000  19
            BloodPressure  768.0   69.105469   19.355807   0.000  62.00000   72.0000   80.00000  12
            SkinThickness  768.0   20.536458   15.952218   0.000   0.00000   23.0000   32.00000   9
                  Insulin  768.0   79.799479  115.244002   0.000   0.00000   30.5000  127.25000  84
                      BMI  768.0   31.992578    7.884160   0.000  27.30000   32.0000   36.60000   6
 DiabetesPedigreeFunction  768.0    0.471876    0.331329   0.078   0.24375    0.3725    0.62625
                      Age  768.0   33.240885   11.760232  21.000  24.00000   29.0000   41.00000   8
                  Outcome  768.0    0.348958    0.476951   0.000   0.00000    0.0000    1.00000
```

```
In [10]: df.describe()
```

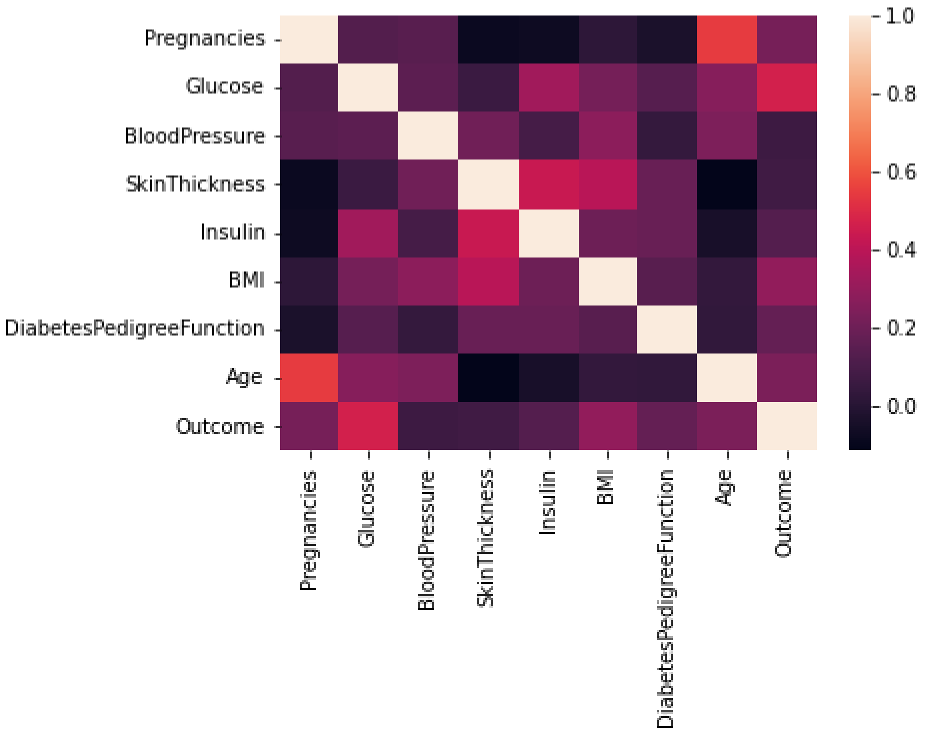
```
Out[10]:
        Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin         BMI  DiabetesPe
 count   768.000000  768.000000     768.000000     768.000000  768.000000  768.000000
  mean     3.845052  120.894531      69.105469      20.536458   79.799479   31.992578
   std     3.369578   31.972618      19.355807      15.952218  115.244002    7.884160
   min     0.000000    0.000000       0.000000       0.000000    0.000000    0.000000
   25%     1.000000   99.000000      62.000000       0.000000    0.000000   27.300000
   50%     3.000000  117.000000      72.000000      23.000000   30.500000   32.000000
   75%     6.000000  140.250000      80.000000      32.000000  127.250000   36.600000
   max    17.000000  199.000000     122.000000      99.000000  846.000000   67.100000
```

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

```
Out[11]:
                           Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BI
               Pregnancies    1.000000  0.129459       0.141282      -0.081672 -0.073535  0.01768
                   Glucose    0.129459  1.000000       0.152590       0.057328  0.331357  0.22107
             BloodPressure    0.141282  0.152590       1.000000       0.207371  0.088933  0.28180
             SkinThickness   -0.081672  0.057328       0.207371       1.000000  0.436783  0.39257
                   Insulin   -0.073535  0.331357       0.088933       0.436783  1.000000  0.19785
                       BMI    0.017683  0.221071       0.281805       0.392573  0.197859  1.00000
  DiabetesPedigreeFunction   -0.033523  0.137337       0.041265       0.183928  0.185071  0.14064
                       Age    0.544341  0.263514       0.239528      -0.113970 -0.042163  0.03624
                   Outcome    0.221898  0.466581       0.065068       0.074752  0.130548  0.29269
```

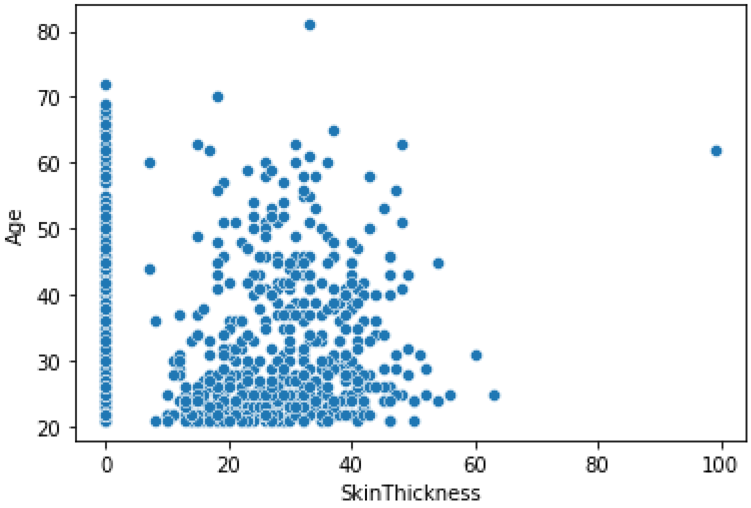
```
In [12]: sns.heatmap(df.corr())
```

```
Out[12]: <AxesSubplot:>
```



In [13]:

```
sns.scatterplot(y='Age',x='SkinThickness',data=df)
plt.show()
```

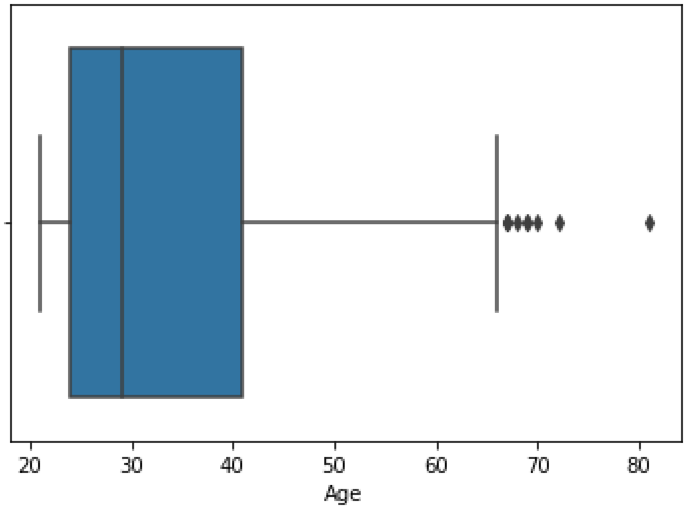


In [14]:

```
sns.boxplot(x='Age',data=df)
```

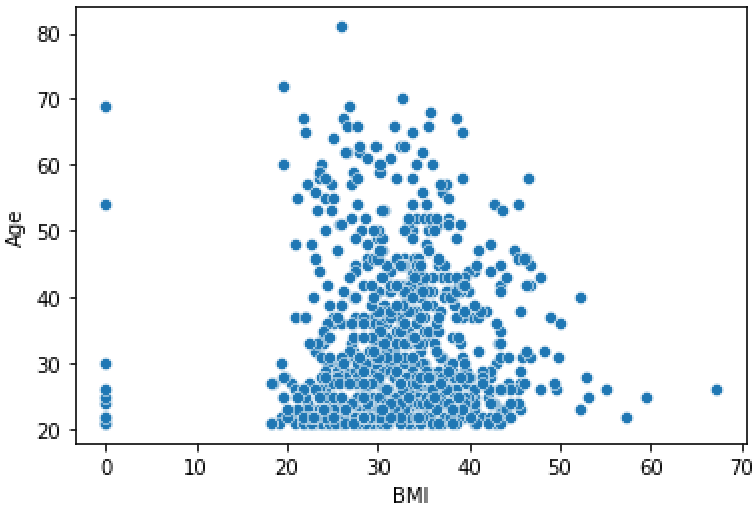
Out[14]:

```
<AxesSubplot:xlabel='Age'>
```



In [15]:

```
sns.scatterplot(x='BMI',y='Age',data=df)
plt.show()
```

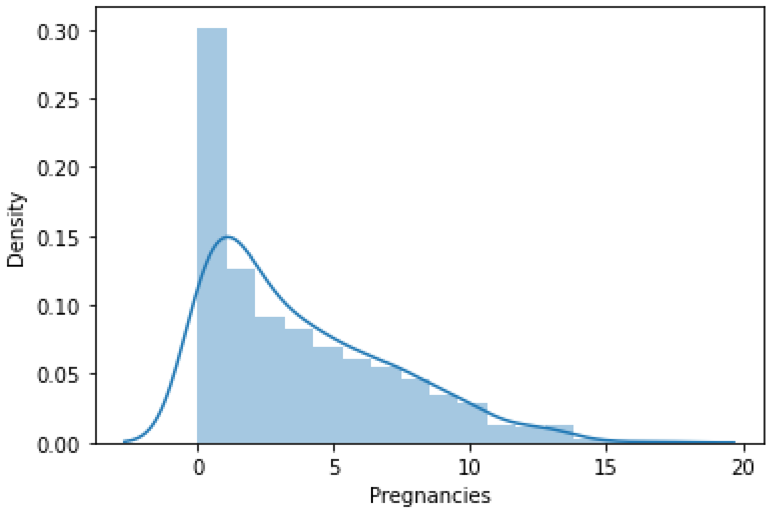


In [16]:

```
sns.distplot(df['Pregnancies'])
```

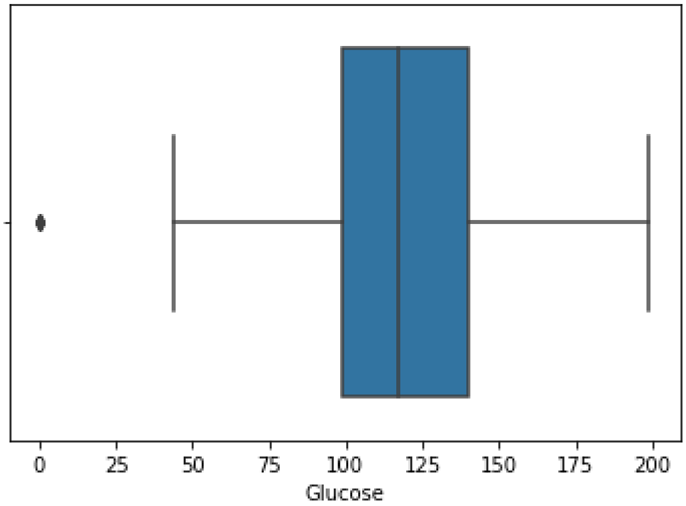
Out[16]:

```
<AxesSubplot:xlabel='Pregnancies', ylabel='Density'>
```



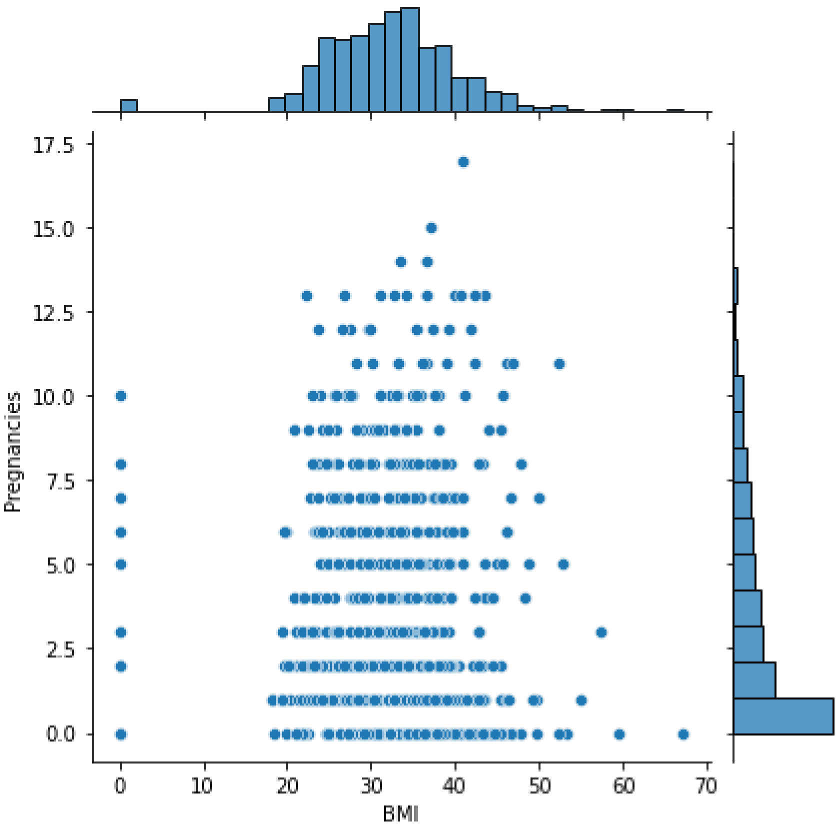
```
In [17]: sns.boxplot(x='Glucose',data=df)
```

```
Out[17]: <AxesSubplot:xlabel='Glucose'>
```



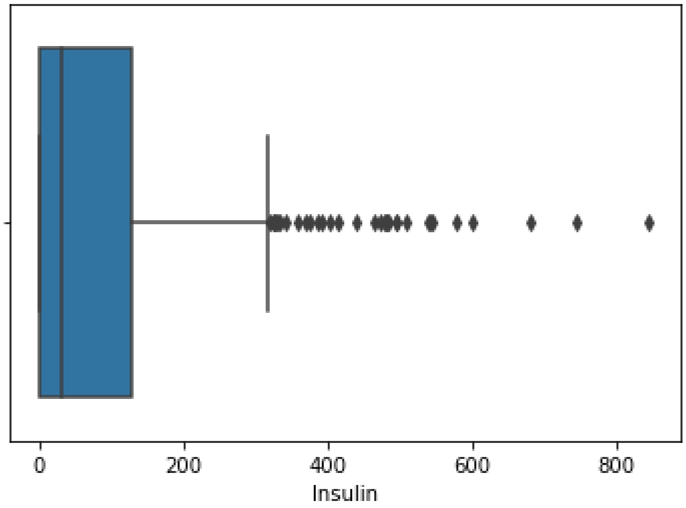
```
In [18]: sns.jointplot(x='BMI',y='Pregnancies',data=df)

Out[18]: <seaborn.axisgrid.JointGrid at 0x1b299bf68b0>
```



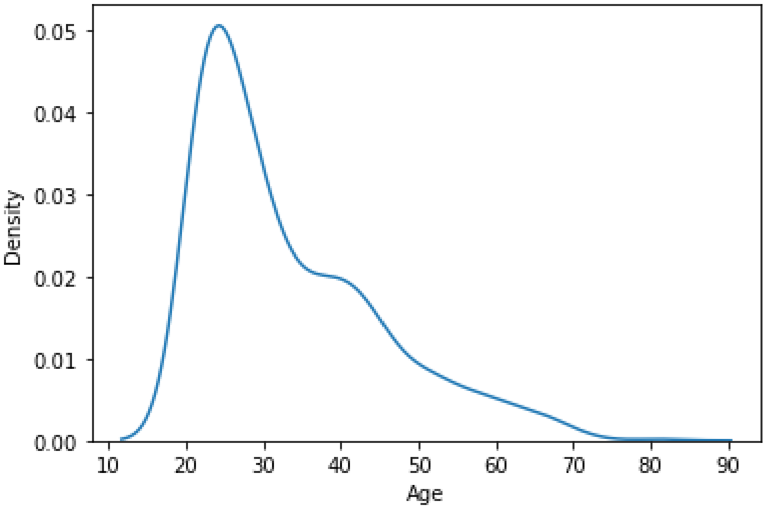
```
In [19]: sns.boxplot(x='Insulin',data=df)

Out[19]: <AxesSubplot:xlabel='Insulin'>
```



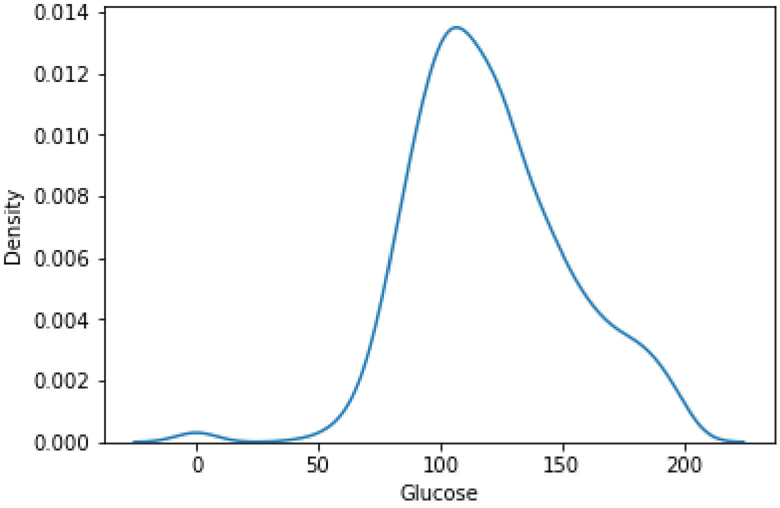
```
In [20]: sns.kdeplot(x='Age',data=df)
```

```
Out[20]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```



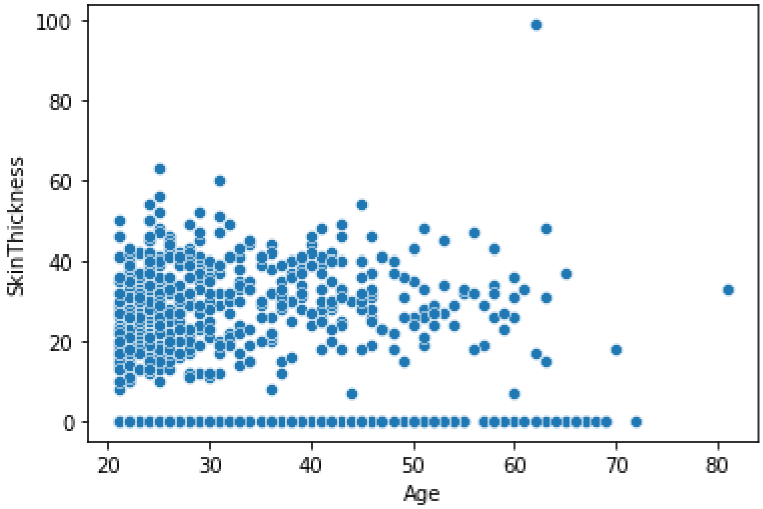
```
In [21]: sns.kdeplot(x='Glucose',data=df)
```

```
Out[21]: <AxesSubplot:xlabel='Glucose', ylabel='Density'>
```



In [22]:

```
sns.scatterplot(y='SkinThickness',x='Age',data=df)
plt.show()
```



Data Cleaning

In [23]:

```
df.describe()
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In the above table there is 0 value in the specific colunms that does't make any sens so have to replace all the zeroes to an nan value for esay readig and understanding and we can replace this Nan value into the some value for better data processing.

There is 0 value in the dataset in columns 1)Glucose 2)BloodPressure 3)SkinThickness 4)Insulin

5) BMI

In [24]:
```
df_copy=df.copy(deep=True)
```

In [25]:
```
df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']]=df_copy[['Gl
```

In [26]:
```
df_copy
```

Out[26]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.0	NaN	33.6	0.627
1	1	85.0	66.0	29.0	NaN	26.6	0.351
2	8	183.0	64.0	NaN	NaN	23.3	0.672
3	1	89.0	66.0	23.0	94.0	28.1	0.167
4	0	137.0	40.0	35.0	168.0	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171
764	2	122.0	70.0	27.0	NaN	36.8	0.340
765	5	121.0	72.0	23.0	112.0	26.2	0.245
766	1	126.0	60.0	NaN	NaN	30.1	0.349
767	1	93.0	70.0	31.0	NaN	30.4	0.315

768 rows × 9 columns

In [27]:
```
df_copy.isnull().sum()
```

Out[27]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [28]:

```
df_copy.describe()
```

Out[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	763.000000	733.000000	541.000000	394.000000	757.000000	
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	
std	3.369578	30.535641	12.382158	10.476982	118.775855	6.924988	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.000000	64.000000	22.000000	76.250000	27.500000	
50%	3.000000	117.000000	72.000000	29.000000	125.000000	32.300000	
75%	6.000000	141.000000	80.000000	36.000000	190.000000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [29]:

```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   763 non-null    float64
 2   BloodPressure             733 non-null    float64
 3   SkinThickness             541 non-null    float64
 4   Insulin                   394 non-null    float64
 5   BMI                       757 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

In [32]:

```
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(),inplace=True)
```

In [33]:

```
df_copy.isnull().sum()
```

Out[33]:

```
Pregnancies                   0
Glucose                       0
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [34]:
```
df_copy['Pregnancies'].fillna(df_copy['Pregnancies'].mean(),inplace=True)
```

In [42]:
```
df_copy.isnull().sum()
```

Out[42]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [43]:
```
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(),inplace=True)
```

In [44]:
```
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(),inplace=True)
```

In [45]:
```
df_copy['Insulin'].fillna(df_copy['Insulin'].median(),inplace=True)
```

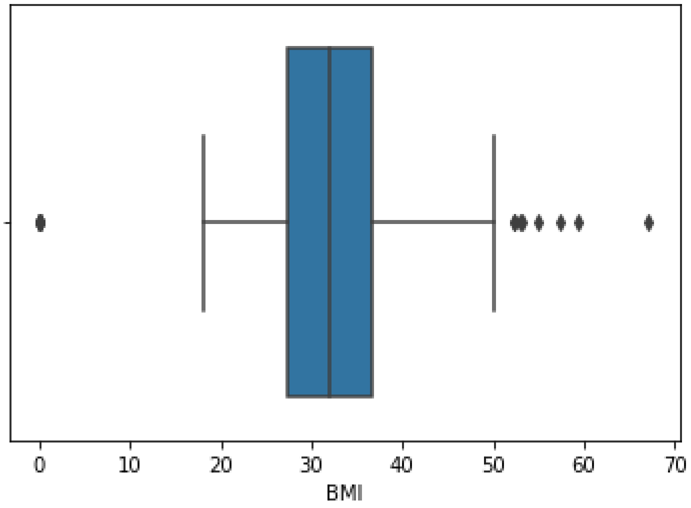
In [48]:
```
df_copy['BMI'].fillna(df_copy['BMI'].median(),inplace=True)
```

In [49]:
```
df_copy.isnull().sum()
```

Out[49]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [51]:
```
sns.boxplot(x='BMI',data=df)
```

Out[51]:
```
<AxesSubplot:xlabel='BMI'>
```



In [53]:
```
df_copy.describe()
```

Out[53]:

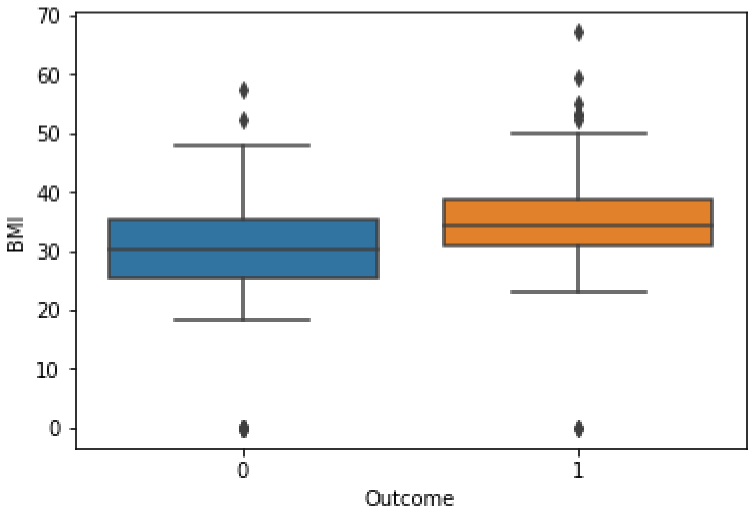
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.686763	72.405184	29.153420	140.671875	32.457464	
std	3.369578	30.435949	12.096346	8.790942	86.383060	6.875151	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	
50%	3.000000	117.000000	72.202592	29.153420	125.000000	32.400000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

Scalling The Data

#WE use the Scalling for dealing with the outlier outlier basically those data which present in outdside the graph the outlier will be affected to the our data so that we want standrazise our data

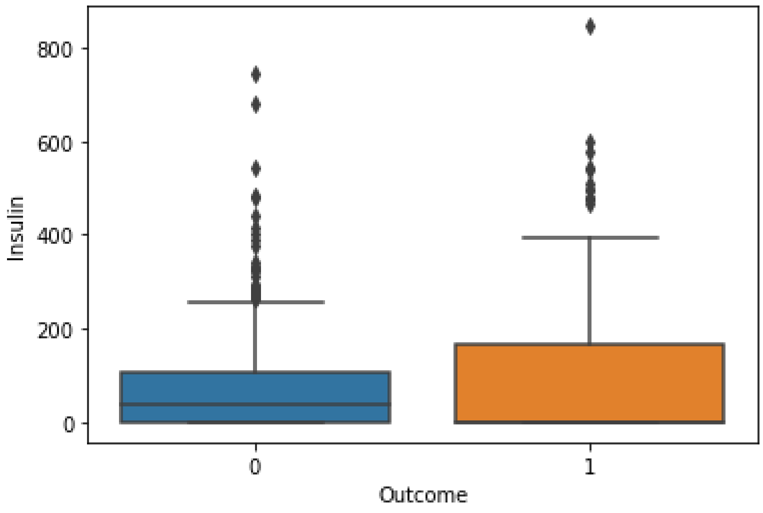
In [57]:
```
sns.boxplot(x='Outcome',y='BMI',data=df)
```

Out[57]:
```
<AxesSubplot:xlabel='Outcome', ylabel='BMI'>
```



In [58]:
```
sns.boxplot(y='Insulin',x='Outcome',data=df)
```

Out[58]:
```
<AxesSubplot:xlabel='Outcome', ylabel='Insulin'>
```



In [62]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [63]:

```
scaler.fit(df_copy.drop('Outcome',axis=1))
```

Out[63]:

```
StandardScaler()
```

In [69]:

```
scaled_features=scaler.transform(df.drop('Outcome',axis=1))
```

In [70]:

```
scaled_features
```

Out[70]:

```
array([[ 0.63994726,  0.86510807, -0.03351824, ...,  0.16629174,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.20616153, -0.52985903, ..., -0.85253118,
        -0.36506078, -0.19067191],
       [ 1.23388019,  2.0158134 , -0.69530596, ..., -1.33283341,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 , -0.0225789 , -0.03351824, ..., -0.91074963,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.14180757, -1.02619983, ..., -0.34311972,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.94314317, -0.19896517, ..., -0.29945588,
        -0.47378505, -0.87137393]])
```

In [73]:

```
df_stand=pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_stand.head(10)
```

Out[73]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	0.639947	0.865108	-0.033518	0.665502	-1.629527	0.166292	0.4
1	-0.844885	-1.206162	-0.529859	-0.017463	-1.629527	-0.852531	-0.3
2	1.233880	2.015813	-0.695306	-3.318463	-1.629527	-1.332833	0.6
3	-0.844885	-1.074652	-0.529859	-0.700429	-0.540642	-0.634212	-0.9
4	-1.141852	0.503458	-2.680669	0.665502	0.316566	1.548980	5.4
5	0.342981	-0.186965	0.131929	-3.318463	-1.629527	-0.998077	-0.8
6	-0.250952	-1.436303	-1.853434	0.324019	-0.610145	-0.212128	-0.6
7	1.827813	-0.219843	-5.989608	-3.318463	-1.629527	0.413720	-1.0
8	-0.547919	2.476096	-0.198965	1.803778	4.660524	-0.284901	-0.9
9	1.233880	0.108930	1.951845	-3.318463	-1.629527	-4.724058	-0.7

In [74]:

```
df_stand['Outcome']=df['Outcome']
```

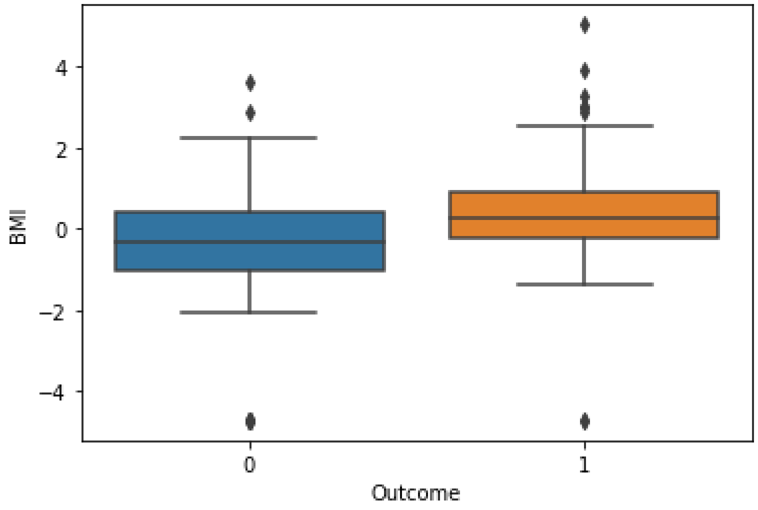
```
In [75]: df_stand.head(10)
```

Out[75]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	0.639947	0.865108	-0.033518	0.665502	-1.629527	0.166292	0.4
1	-0.844885	-1.206162	-0.529859	-0.017463	-1.629527	-0.852531	-0.3
2	1.233880	2.015813	-0.695306	-3.318463	-1.629527	-1.332833	0.6
3	-0.844885	-1.074652	-0.529859	-0.700429	-0.540642	-0.634212	-0.9
4	-1.141852	0.503458	-2.680669	0.665502	0.316566	1.548980	5.4
5	0.342981	-0.186965	0.131929	-3.318463	-1.629527	-0.998077	-0.8
6	-0.250952	-1.436303	-1.853434	0.324019	-0.610145	-0.212128	-0.6
7	1.827813	-0.219843	-5.989608	-3.318463	-1.629527	0.413720	-1.0
8	-0.547919	2.476096	-0.198965	1.803778	4.660524	-0.284901	-0.9
9	1.233880	0.108930	1.951845	-3.318463	-1.629527	-4.724058	-0.7

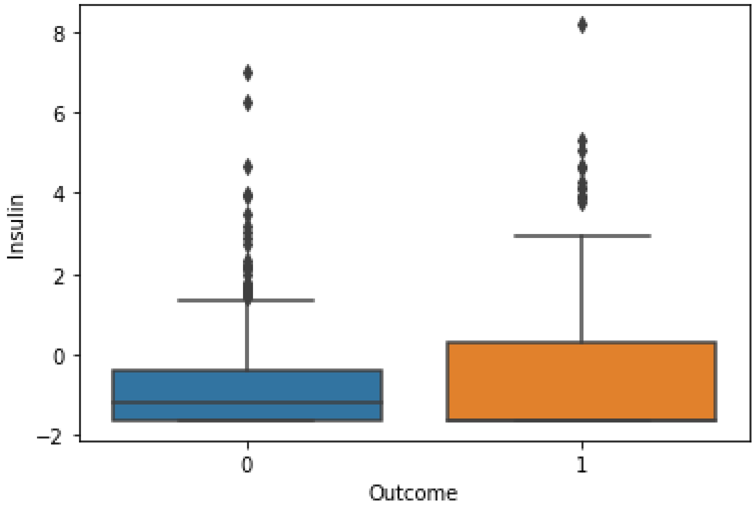
```
In [76]: sns.boxplot(x='Outcome',y='BMI',data=df_stand)
```

```
Out[76]: <AxesSubplot:xlabel='Outcome', ylabel='BMI'>
```



```
In [77]: sns.boxplot(x='Outcome',y='Insulin',data=df_stand)
```

```
Out[77]: <AxesSubplot:xlabel='Outcome', ylabel='Insulin'>
```



Dependent And Independent Set

```
In [81]: x=df_stand.drop('Outcome',axis=1)
```

In [82]:
```
x
```

Out[82]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
0	0.639947	0.865108	-0.033518	0.665502	-1.629527	0.166292	
1	-0.844885	-1.206162	-0.529859	-0.017463	-1.629527	-0.852531	-
2	1.233880	2.015813	-0.695306	-3.318463	-1.629527	-1.332833	
3	-0.844885	-1.074652	-0.529859	-0.700429	-0.540642	-0.634212	-
4	-1.141852	0.503458	-2.680669	0.665502	0.316566	1.548980	
...	...	...	...	...	...	...	
763	1.827813	-0.680125	0.297376	2.145261	0.455573	0.064409	-
764	-0.547919	0.010298	-0.198965	-0.245119	-1.629527	0.632039	-
765	0.342981	-0.022579	-0.033518	-0.700429	-0.332132	-0.910750	-
766	-0.844885	0.141808	-1.026200	-3.318463	-1.629527	-0.343120	·
767	-0.844885	-0.943143	-0.198965	0.210192	-1.629527	-0.299456	-

768 rows × 8 columns

In [83]:
```
y=df_stand['Outcome']
```

In [84]:
```
y
```

Out[84]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Train Test Split

In [86]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [87]:

```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 537 entries, 537 to 648
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               537 non-null    float64
 1   Glucose                   537 non-null    float64
 2   BloodPressure             537 non-null    float64
 3   SkinThickness             537 non-null    float64
 4   Insulin                   537 non-null    float64
 5   BMI                       537 non-null    float64
 6   DiabetesPedigreeFunction  537 non-null    float64
 7   Age                       537 non-null    float64
dtypes: float64(8)
memory usage: 37.8 KB
```

In [90]:

```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 231 entries, 152 to 349
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               231 non-null    float64
 1   Glucose                   231 non-null    float64
 2   BloodPressure             231 non-null    float64
 3   SkinThickness             231 non-null    float64
 4   Insulin                   231 non-null    float64
 5   BMI                       231 non-null    float64
 6   DiabetesPedigreeFunction  231 non-null    float64
 7   Age                       231 non-null    float64
dtypes: float64(8)
memory usage: 16.2 KB
```

Logistic Regression Algorithms

In [92]:

```
from sklearn.linear_model import LogisticRegression
```

In [93]:

```
logmodel=LogisticRegression()
```

In [95]:

```
logmodel.fit(X_train,y_train)
```

Out[95]:

```
LogisticRegression()
```

In [96]:

```
predictions=logmodel.predict(X_test)
```

In [97]:

```
from sklearn.metrics import classification_report
```

In [99]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.79      0.91      0.85       152
           1       0.75      0.54      0.63        79

    accuracy                           0.78       231
   macro avg       0.77      0.73      0.74       231
weighted avg       0.78      0.78      0.77       231
```

In [100]:
```
from sklearn.metrics import confusion_matrix
```

In [101]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[138  14]
 [ 36  43]]
```

In [ ]:
```

```