In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv('C:/Users/User/Desktop/Data Science Notes/Data Science Capstone/Project_Healthcare PGP/Healthcare - Diabetes/hea
```

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.dtypes
```

Out[4]:

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

In [5]:

```
df.dtypes.value_counts()
```

Out[5]:

```
int64      7
float64    2
dtype: int64
```

In [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
df.columns
```

Out[7]:

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

Missing Values Treatment

In [8]:

```
df.isna().any()
```

Out[8]:

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

In [9]:

```
df.isna().sum()
```

Out[9]:

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

In [10]:

```
df.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                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [11]:

```
cols_null_as_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[cols_null_as_zero] = df[cols_null_as_zero].replace(0, np.NaN)
```

In [12]:

```
df.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 [13]:

```
df.isna().sum()
```

Out[13]:

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

In [14]:

```
df.describe()
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	763.000000	733.000000	541.000000	394.000000	757.000000	768.000000	768.000000	768.000000
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	0.471876	33.240885	0.348958
std	3.369578	30.535641	12.382158	10.476982	118.775855	6.924988	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	22.000000	76.250000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	29.000000	125.000000	32.300000	0.372500	29.000000	0.000000
75%	6.000000	141.000000	80.000000	36.000000	190.000000	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

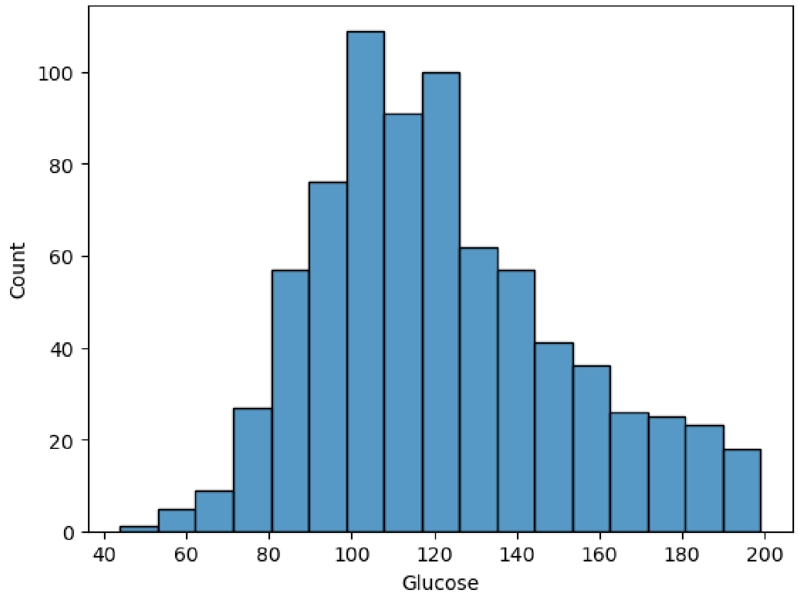
Histogram

In [15]:

```
sns.histplot(data=df['Glucose'])
```

Out[15]:

```
<AxesSubplot:xlabel='Glucose', ylabel='Count'>
```

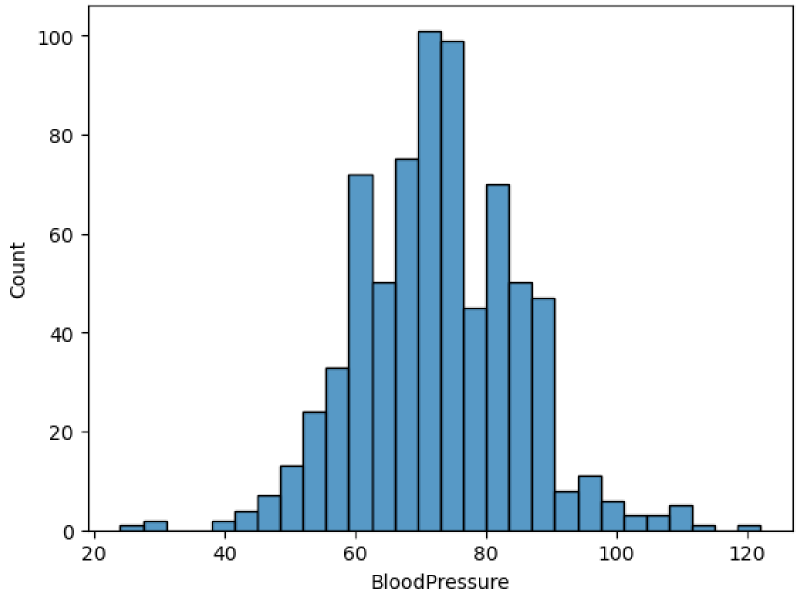


In [16]:

```
sns.histplot(data=df['BloodPressure'])
```

Out[16]:

```
<AxesSubplot:xlabel='BloodPressure', ylabel='Count'>
```

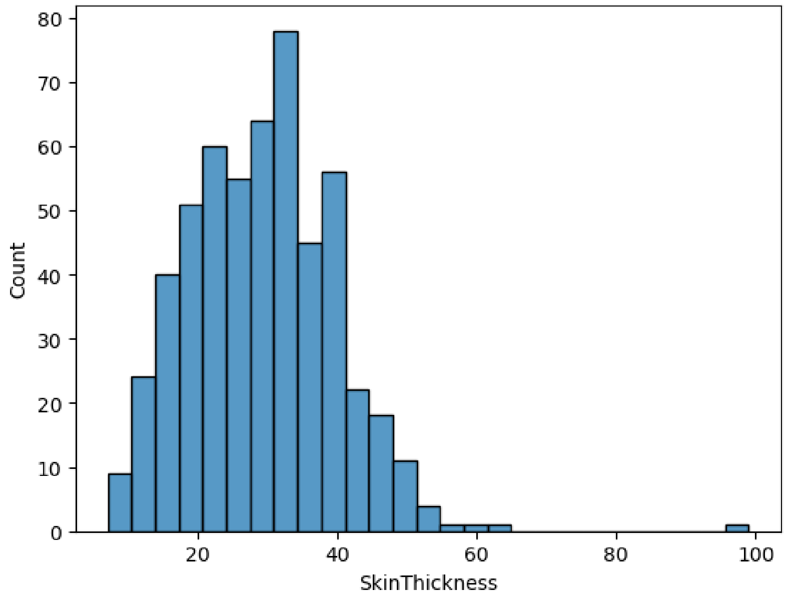


In [17]:

```
sns.histplot(data=df['SkinThickness'])
```

Out[17]:

```
<AxesSubplot:xlabel='SkinThickness', ylabel='Count'>
```

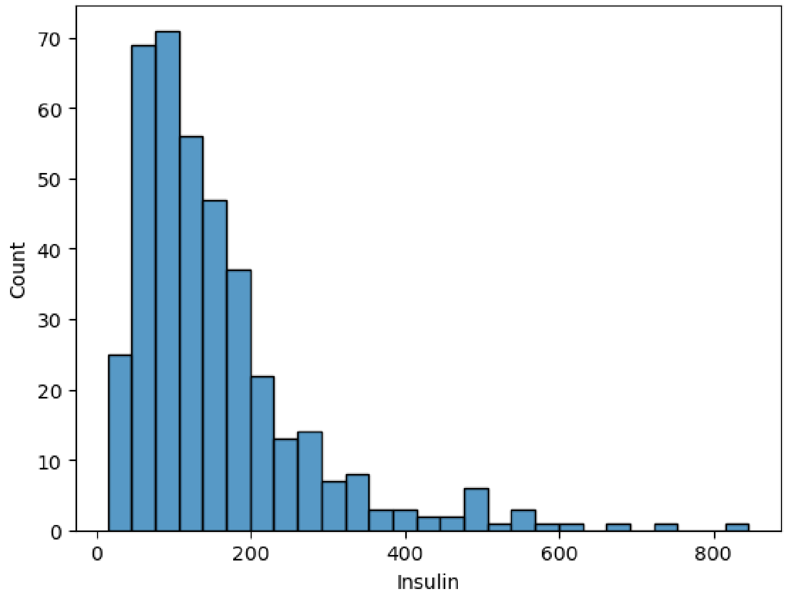


In [18]:

```
sns.histplot(data=df['Insulin'])
```

Out[18]:

```
<AxesSubplot:xlabel='Insulin', ylabel='Count'>
```

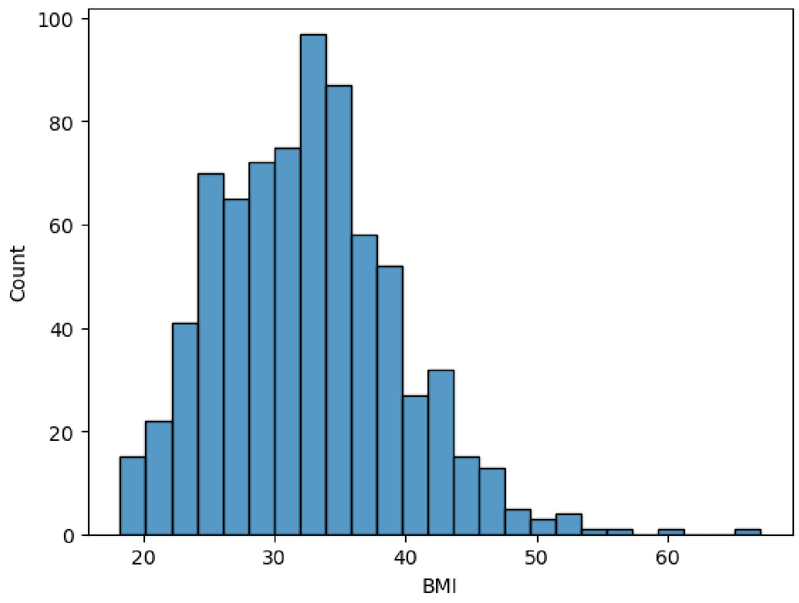


In [19]:

```
sns.histplot(data=df['BMI'])
```

Out[19]:

```
<AxesSubplot:xlabel='BMI', ylabel='Count'>
```



In [20]:

```
# Median Imputation for 'Insulin'
df['Insulin'] = df['Insulin'].fillna(df['Insulin'].median())
```

In [21]:

```
# Mean Imputation
cols_mean_for_null = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI']
df[cols_mean_for_null] = df[cols_mean_for_null].fillna(df[cols_mean_for_null].mean())
```

In [22]:

```
df.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                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 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
```

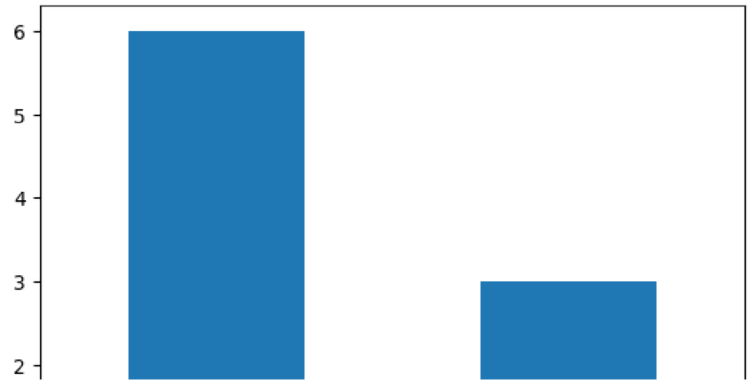
Frequency Plot

In [23]:

```
df.dtypes.value_counts().plot(kind='bar')
```

Out[23]:

```
<AxesSubplot:>
```

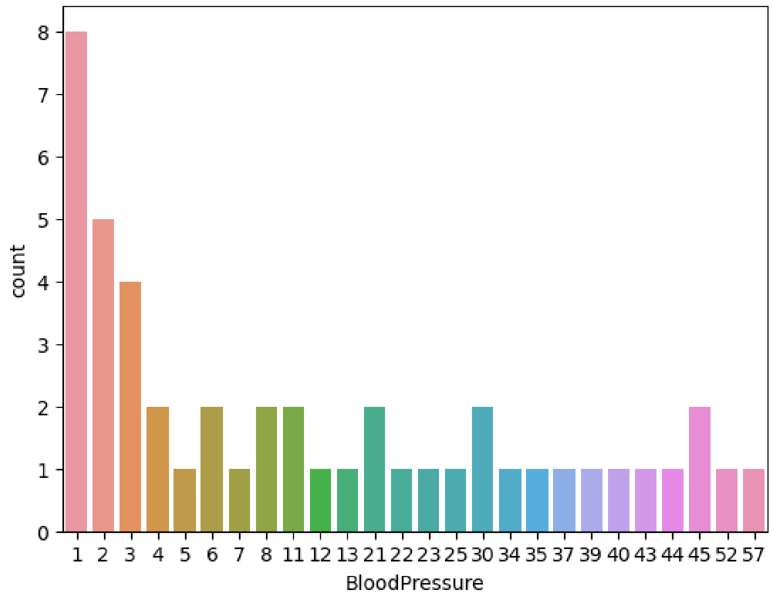


In [24]:

```
sns.countplot(x=df['BloodPressure'].value_counts())
```

Out[24]:

```
<AxesSubplot:xlabel='BloodPressure', ylabel='count'>
```



Data Exploration

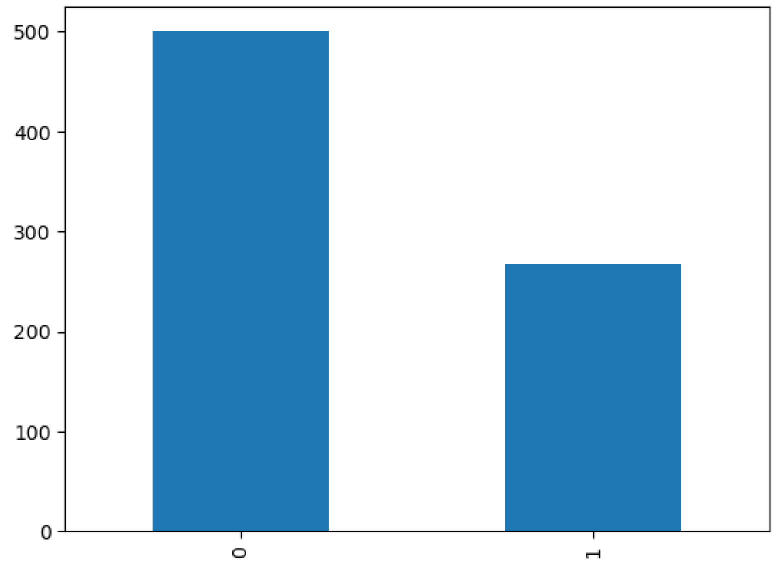
Checking the Balance of Data

In [25]:

```
df['Outcome'].value_counts().plot(kind='bar')
df['Outcome'].value_counts()
```

Out[25]:

```
0    500
1    268
Name: Outcome, dtype: int64
```



SMOTE to Balance the Imbalanced Data

In [26]:

```
df_x = df.drop('Outcome', axis=1)
df_y = df['Outcome']
print(df_x.shape, df_y.shape)
```

```
(768, 8) (768,)
```

In [27]:

```
from imblearn.over_sampling import SMOTE
```

In [28]:

```
df_x_res, df_y_res = SMOTE(random_state=108).fit_resample(df_x, df_y)
print(df_x_res.shape, df_y_res.shape)
```

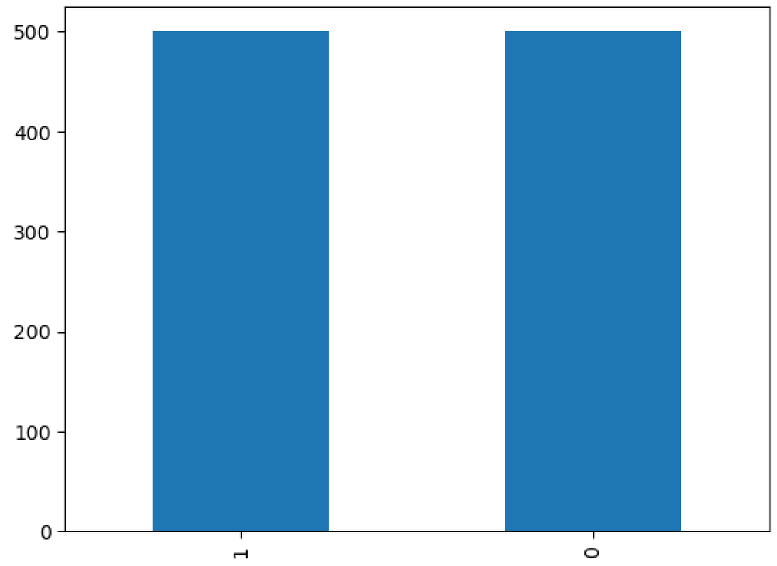
```
(1000, 8) (1000,)
```

In [29]:

```
df_y_res.value_counts().plot(kind='bar')
df_y_res.value_counts()
```

Out[29]:

```
1    500
0    500
Name: Outcome, dtype: int64
```



Scatter Plots between the pair of variables to understand the relationships

In [30]:

```
df_res = pd.concat([df_x_res, df_y_res], axis=1)
df_res
```

Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.000000	72.000000	35.000000	125.000000	33.600000	0.627000	50	1
1	1	85.000000	66.000000	29.000000	125.000000	26.600000	0.351000	31	0
2	8	183.000000	64.000000	29.153420	125.000000	23.300000	0.672000	32	1
3	1	89.000000	66.000000	23.000000	94.000000	28.100000	0.167000	21	0
4	0	137.000000	40.000000	35.000000	168.000000	43.100000	2.288000	33	1
...	...	...	...	...	...	...	...	...	...
995	3	164.686765	74.249021	29.153420	125.000000	42.767110	0.726091	29	1
996	0	138.913540	69.022720	27.713033	127.283849	39.177649	0.703702	24	1
997	10	131.497740	66.331574	33.149837	125.000000	45.820819	0.498032	38	1
998	0	105.571347	83.238205	29.153420	125.000000	27.728596	0.649204	60	1
999	0	127.727025	108.908879	44.468195	129.545366	65.808840	0.308998	26	1

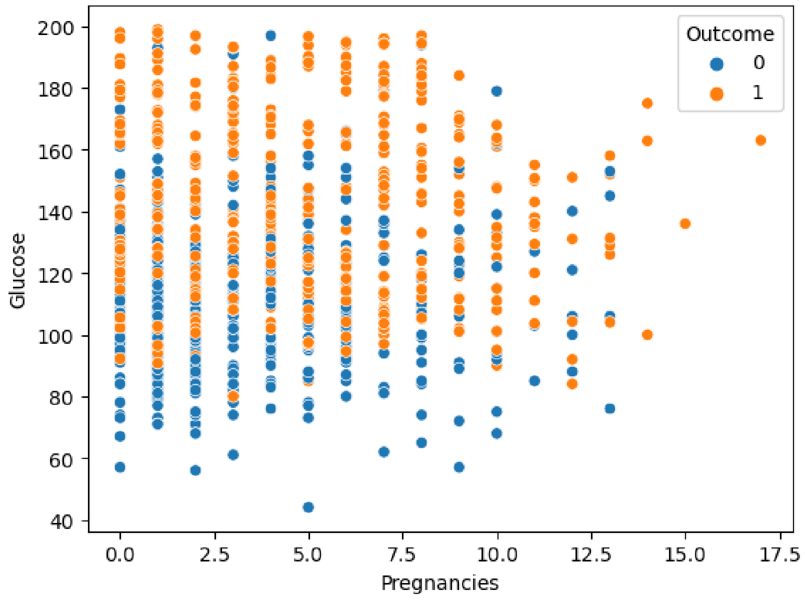
1000 rows × 9 columns

In [31]:

```
sns.scatterplot(x='Pregnancies', y='Glucose', data=df_res, hue='Outcome')
```

Out[31]:

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

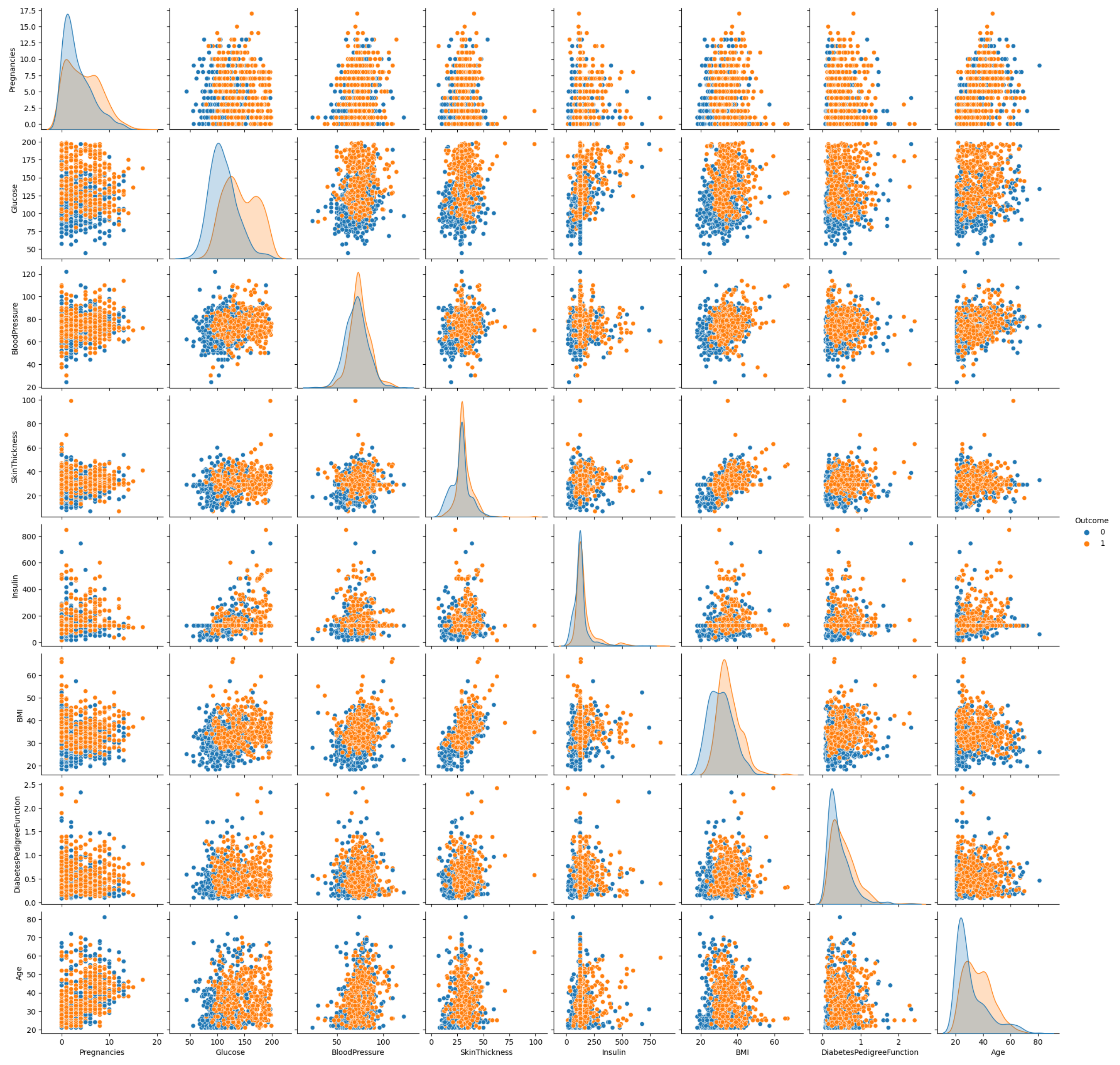


In [32]:

```
sns.pairplot(data=df_res, hue='Outcome')
```

Out[32]:

```
<seaborn.axisgrid.PairGrid at 0x761521adc0>
```



Perform correlation analysis. Visually explore it using a heat map

In [33]:

```
df_res.corr()
```

Out[33]:

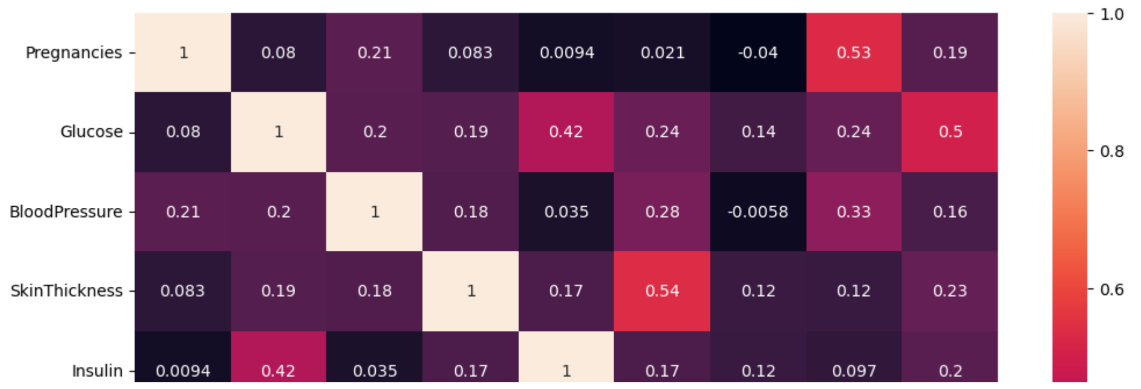
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outco
Pregnancies	1.000000	0.079953	0.205232	0.082752	0.009365	0.021006	-0.040210	0.532660	0.193
Glucose	0.079953	1.000000	0.200717	0.189776	0.418830	0.242501	0.138945	0.235522	0.500
BloodPressure	0.205232	0.200717	1.000000	0.176496	0.034861	0.277565	-0.005850	0.332015	0.162
SkinThickness	0.082752	0.189776	0.176496	1.000000	0.170719	0.538207	0.120799	0.117644	0.230
Insulin	0.009365	0.418830	0.034861	0.170719	1.000000	0.168702	0.115187	0.096940	0.200
BMI	0.021006	0.242501	0.277565	0.538207	0.168702	1.000000	0.177915	0.017529	0.334
DiabetesPedigreeFunction	-0.040210	0.138945	-0.005850	0.120799	0.115187	0.177915	1.000000	0.010532	0.187
Age	0.532660	0.235522	0.332015	0.117644	0.096940	0.017529	0.010532	1.000000	0.234
Outcome	0.193421	0.500321	0.162788	0.230017	0.200911	0.334196	0.187519	0.234059	1.000

In [34]:

```
plt.figure(figsize=(12,8))
sns.heatmap(df_res.corr(), annot=True)
```

Out[34]:

```
<AxesSubplot:>
```



Data Modeling

Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

In [35]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

In [36]:

```
df_x_res = df.drop('Outcome', axis=1)
df_y_res = df['Outcome']
print(df_x_res.shape, df_y_res.shape)
```

```
(768, 8) (768,)
```

In [37]:

```
x_train, x_test, y_train, y_test = train_test_split(df_x_res,df_y_res,test_size=0.30,random_state=12345)
print(x_train.shape, x_test.shape)
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
```

```
(537, 8) (231, 8)

C:\Users\User\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed t
o converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessi
ng.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/
modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[37]:

```
LogisticRegression()
```

In [38]:

```
y_pred = logreg.predict(x_test)
```

In [39]:

```
y_pred
```

Out[39]:

```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1], dtype=int64)
```

In [40]:

```
logreg.score(x_train, y_train)
```

Out[40]:

```
0.770949720670391
```

In [41]:

```
logreg.score(x_test, y_test)
```

Out[41]:

```
0.8225108225108225
```

Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc.

In [47]:

```
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve
```

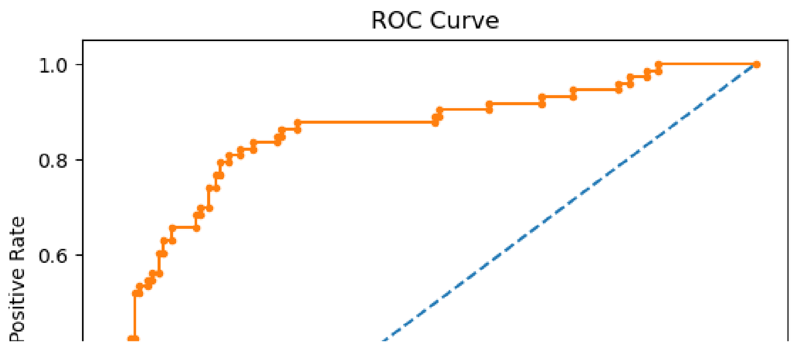
In [48]:

```
prob = logreg.predict_proba(x_test)
prob = prob[:,1]
auc_lr = roc_auc_score(y_test, prob)
print('AUC : %.3f' %auc_lr)
fpr, tpr, thresholds = roc_curve(y_test, prob)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
```

```
AUC : 0.859
```

Out[48]:

```
Text(0.5, 1.0, 'ROC Curve')
```



In [58]:

```
# Classification Report with Logistic Regression
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.95      0.88       158
           1       0.83      0.55      0.66        73

    accuracy                           0.82       231
   macro avg       0.83      0.75      0.77       231
weighted avg       0.82      0.82      0.81       231
```

Compare various models with the results from KNN algorithm.

In [59]:

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(x_train, y_train)
```

Out[59]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [60]:

```
y_k_pred = neigh.predict(x_test)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other red
uction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over
which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to T
rue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [61]:

```
neigh.score(x_train, y_train)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other red
uction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over
which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to T
rue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[61]:

```
0.8324022346368715
```

In [62]:

```
neigh.score(x_test, y_test)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other red
uction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over
which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to T
rue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[62]:

```
0.7316017316017316
```

In [63]:

```
# Classification Report with KNN Algorithm
print(classification_report(y_test, y_k_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.81      0.81       158
           1       0.58      0.56      0.57        73

    accuracy                           0.73       231
   macro avg       0.69      0.69      0.69       231
weighted avg       0.73      0.73      0.73       231
```

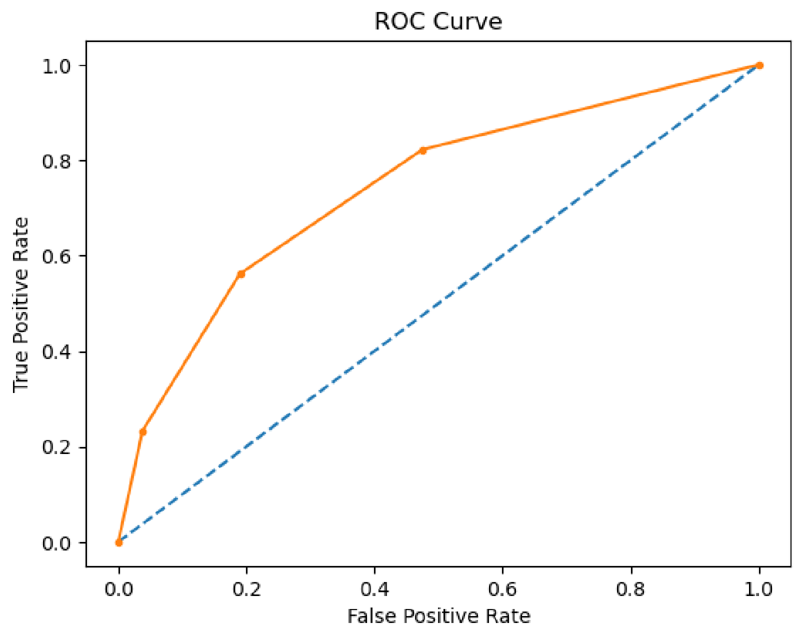
In [65]:

```
k_prob = neigh.predict_proba(x_test)
k_prob = k_prob[:,1]
auc_n = roc_auc_score(y_test, k_prob)
print('AUC : %.3f' %auc_n)
fpr, tpr, thresholds = roc_curve(y_test, k_prob)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
```

```
AUC : 0.740
```

Out[65]:

```
Text(0.5, 1.0, 'ROC Curve')
```



In [ ]: