In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import numpy as np
import seaborn as sns
import os
```

In [2]:
```
df=pd.read_csv(r'C:\Users\ashuk\Downloads\diabetes.csv')
```

In [3]:
```
df  #ASHU PANDIT
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
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	(

768 rows × 9 columns

In [4]:
```
df.head
```

Out[4]:
```
<bound method NDFrame.head of      Pregnancies  Glucose  BloodPressure  Ski
nThickness  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]>
```

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

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

In [6]:
```
df.tail()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
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	(

In [7]:
```
df.describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabe
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 [8]:
```
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 [ ]:

In [9]:
```
df.isnull()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
763	False	False	False	False	False	False	
764	False	False	False	False	False	False	
765	False	False	False	False	False	False	
766	False	False	False	False	False	False	
767	False	False	False	False	False	False	

768 rows × 9 columns

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

Out[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [11]:
```
df.isnull().sum().sum()
```

Out[11]: 0

In [12]:
```
df.shape
```

Out[12]: (768, 9)

In [13]:
```
df.value_counts()
```

Out[13]:
```
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI   Diabetes
PedigreeFunction  Age  Outcome
0            57       60             0              0        21.7  0.735
67   0          1
             67       76             0              0        45.3  0.194
46   0          1
5            103      108            37             0        39.2  0.305
65   0          1
             104      74             0              0        28.8  0.153
48   0          1
             105      72             29             325      36.9  0.159
28   0          1

..
2            84       50             23             76       30.4  0.968
21   0          1
             85       65             0              0        39.6  0.930
27   0          1
             87       0              23             0        28.9  0.773
25   0          1
                      58             16             52       32.7  0.166
25   0          1
17           163      72             41             114      40.9  0.817
47   1          1
Length: 768, dtype: int64
```

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

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

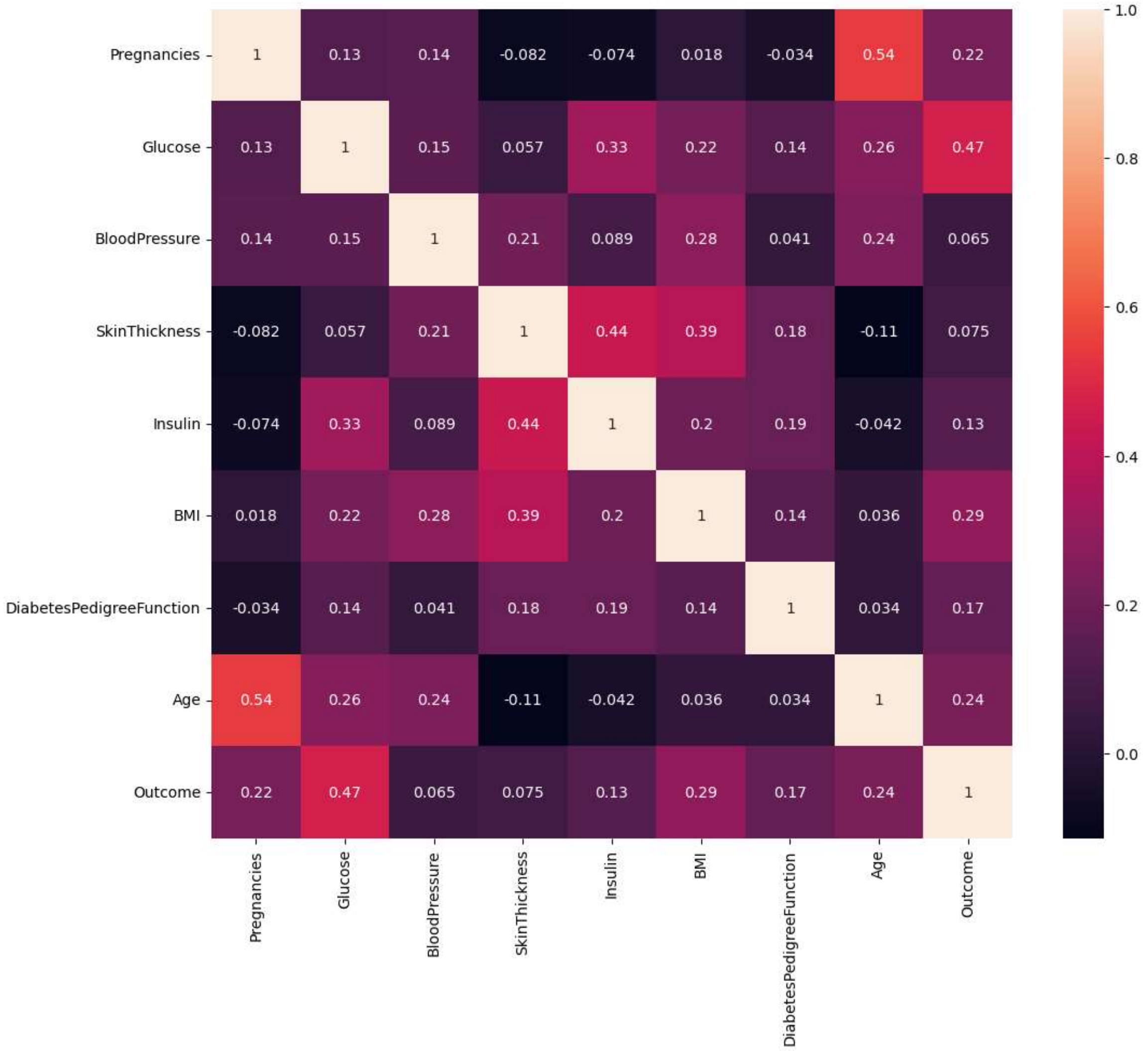
In [15]:
```
df.corr()
```

Out[15]:

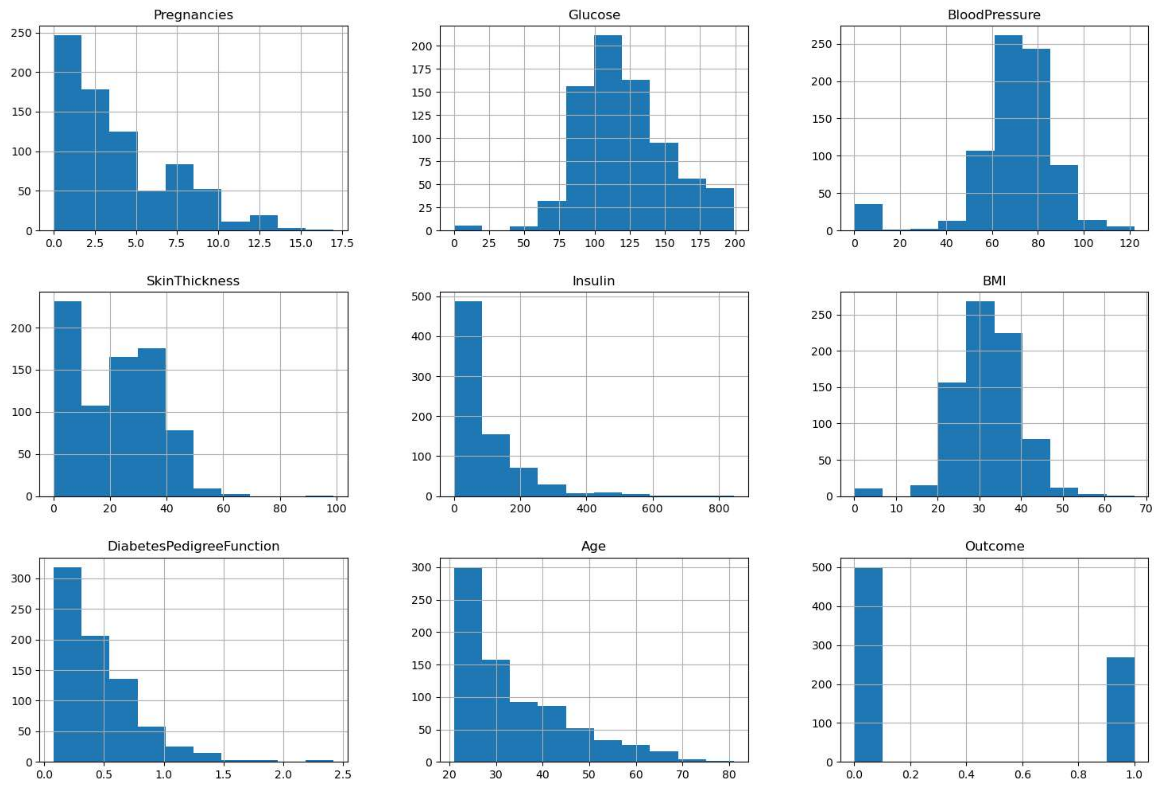
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.

In [16]:
```python
plt.figure(figsize = (12,10))
sns.heatmap(df.corr(),annot = True)
```

Out[16]: <Axes: >



In [17]:
```
df.hist(figsize=(18,12))
plt.show()
```

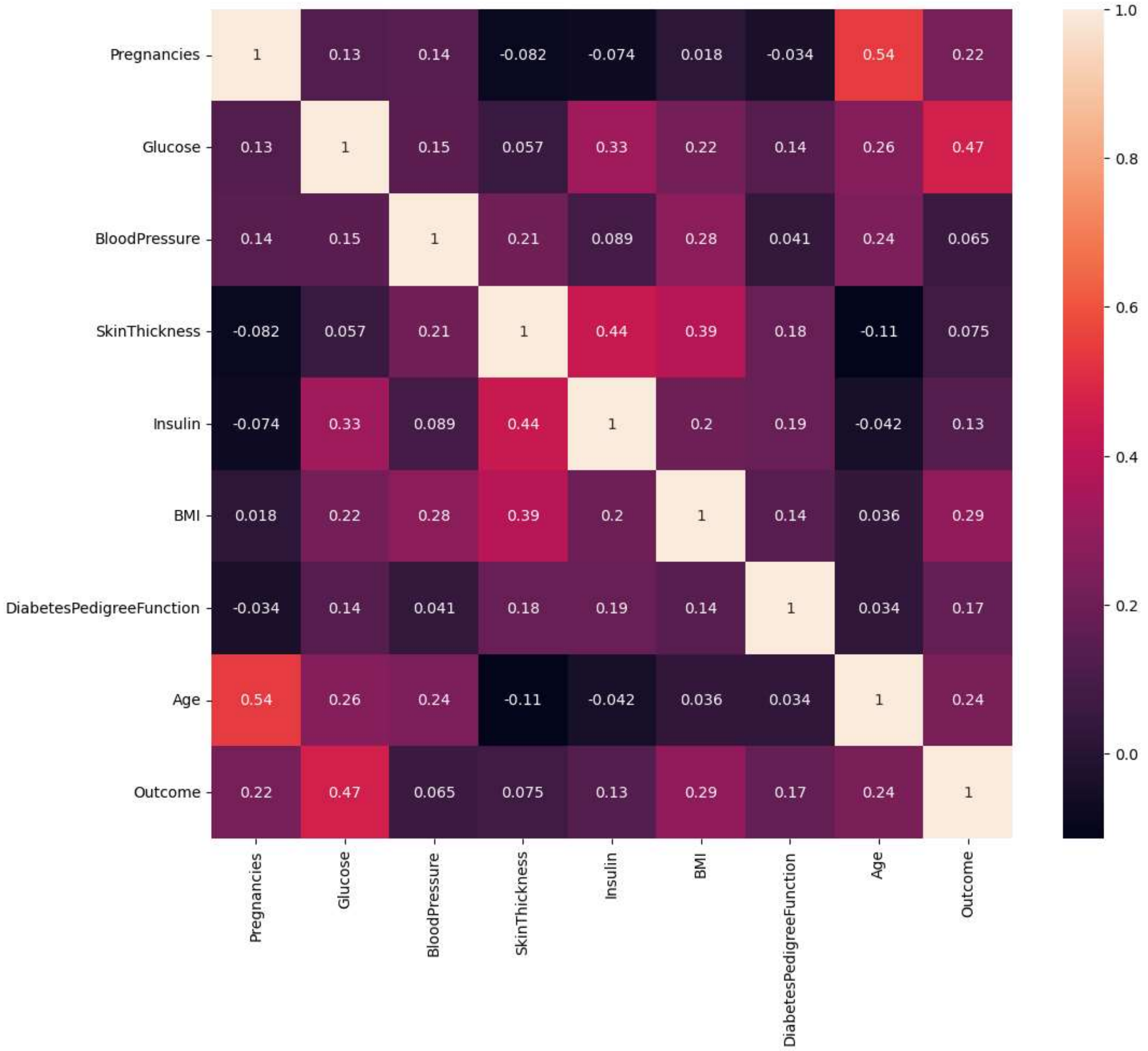


In [18]:
```
features = ['Glucose', 'BloodPressure', 'Insulin', 'BMI', 'Age','SkinThickne
plt.figure(figsize=(14, 10))
for i, feature in enumerate(features, start=1):
    plt.subplot(2, 3, i)
    sns.boxplot(x=feature, data=df)
plt.tight_layout()
plt.show()
```

...

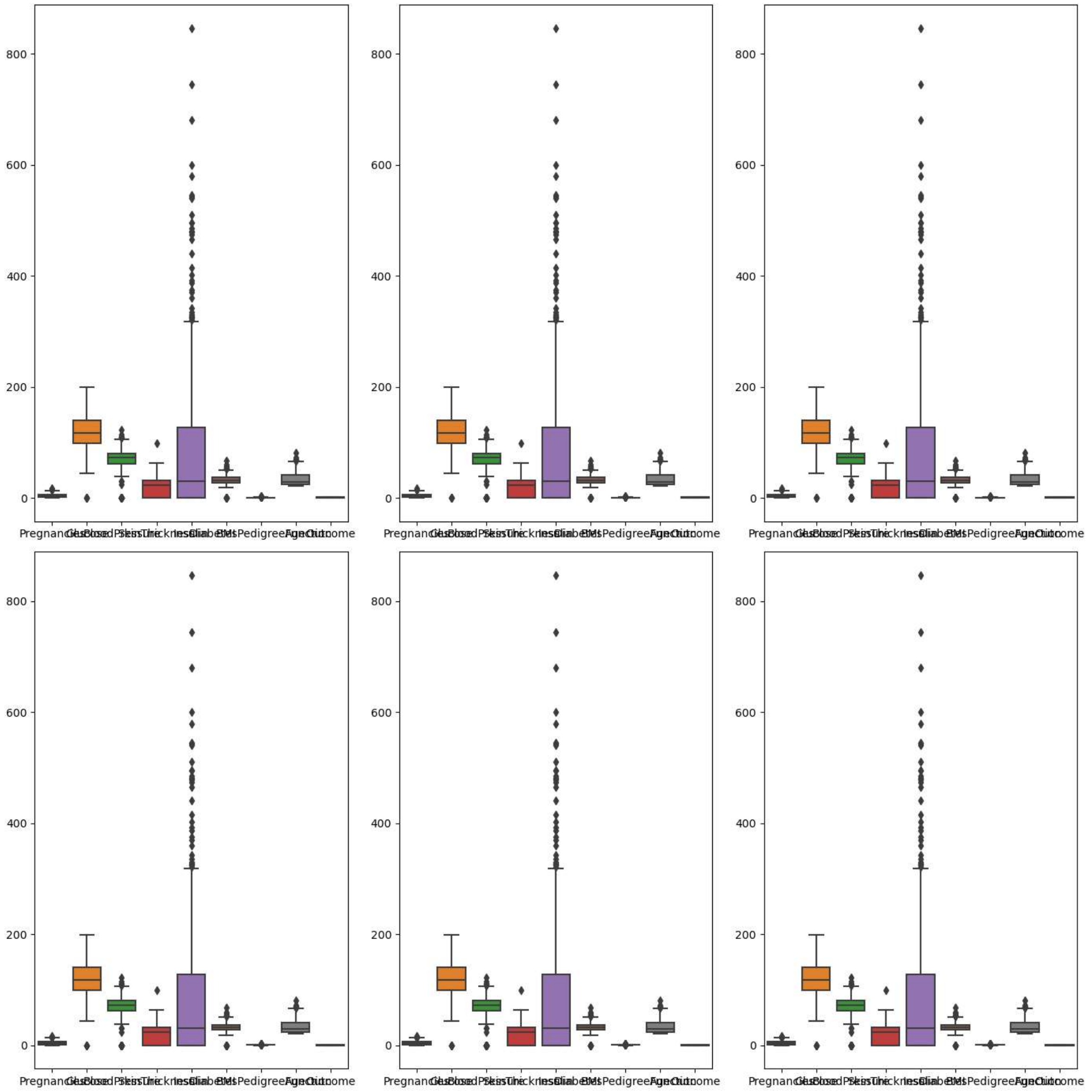
In [19]:
```
plt.figure(figsize = (12,10))
sns.heatmap(df.corr(), annot =True)
```

Out[19]: <Axes: >



In [22]:

```python
features = ['Glucose', 'BloodPressure', 'Insulin', 'BMI', 'Age','SkinThickne
plt.figure(figsize=(14, 14))
for i, feature in enumerate(features, start=1):
    plt.subplot(2, 3, i)
    sns.boxplot(df)
plt.tight_layout()
plt.show()
```



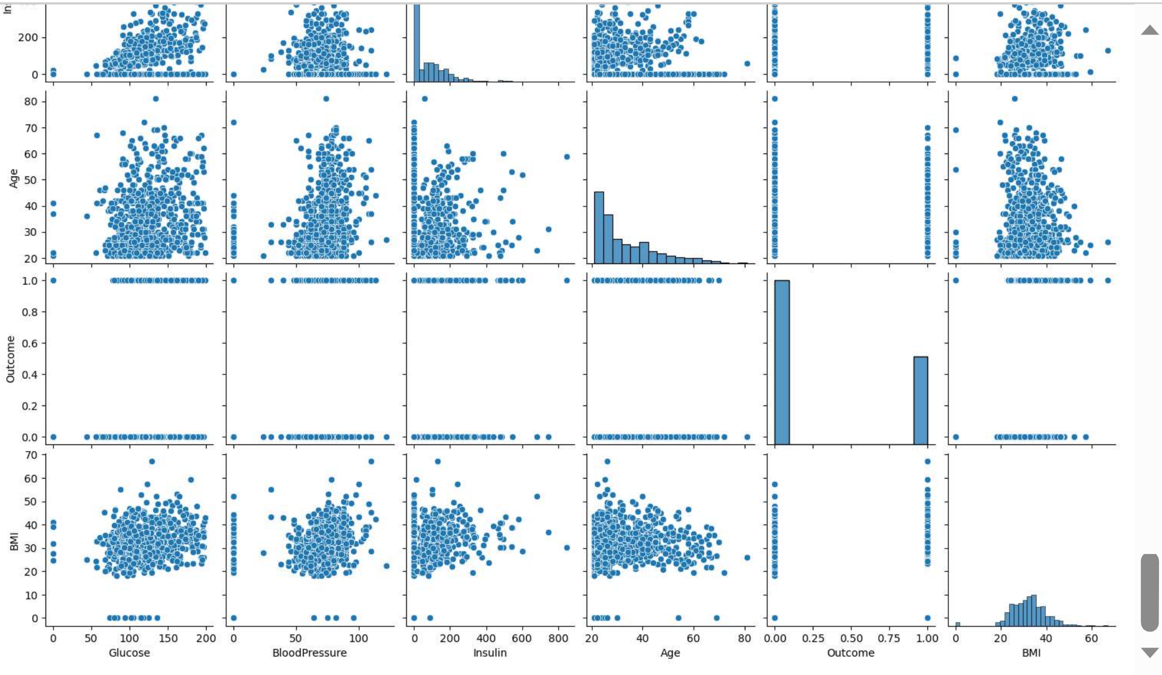
In [21]:
```
df
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
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	2
...	...	...	...	...	...	...	
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	(

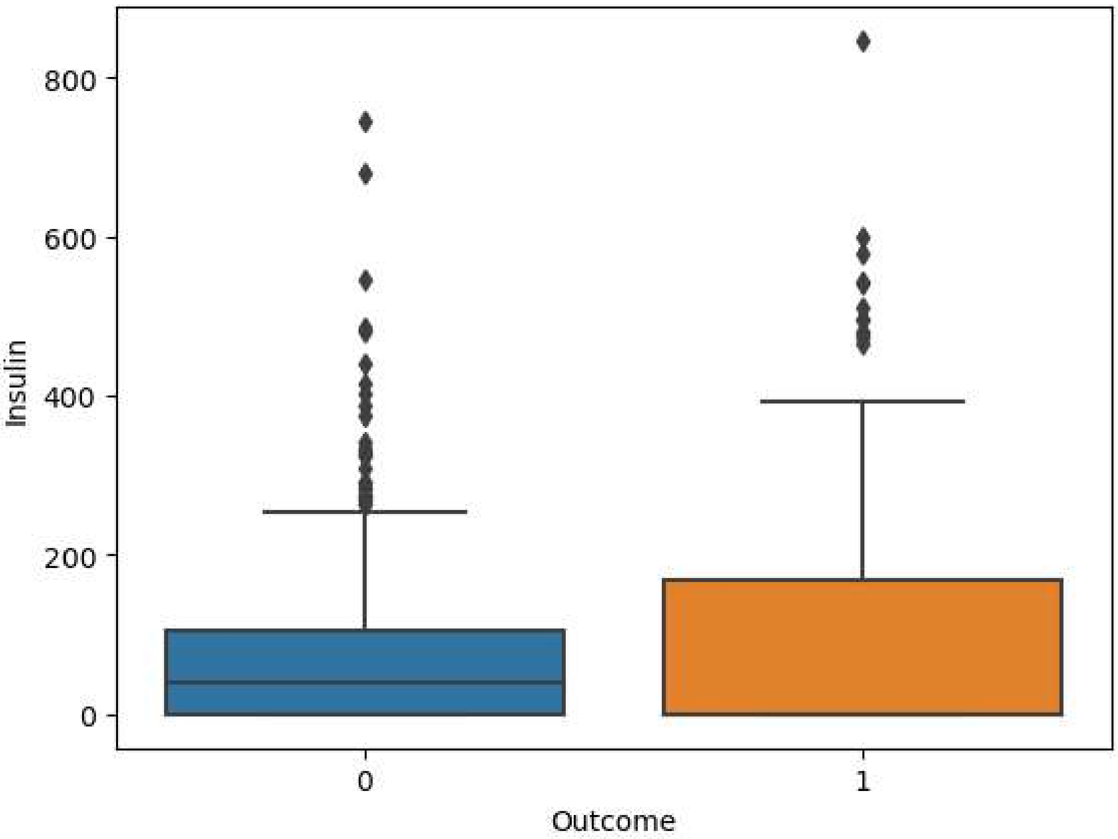
768 rows × 9 columns

In [23]:
```
mean_col = ['Glucose','BloodPressure','Insulin','Age','Outcome','BMI']
sns.pairplot(df[mean_col],palette='dark')
```



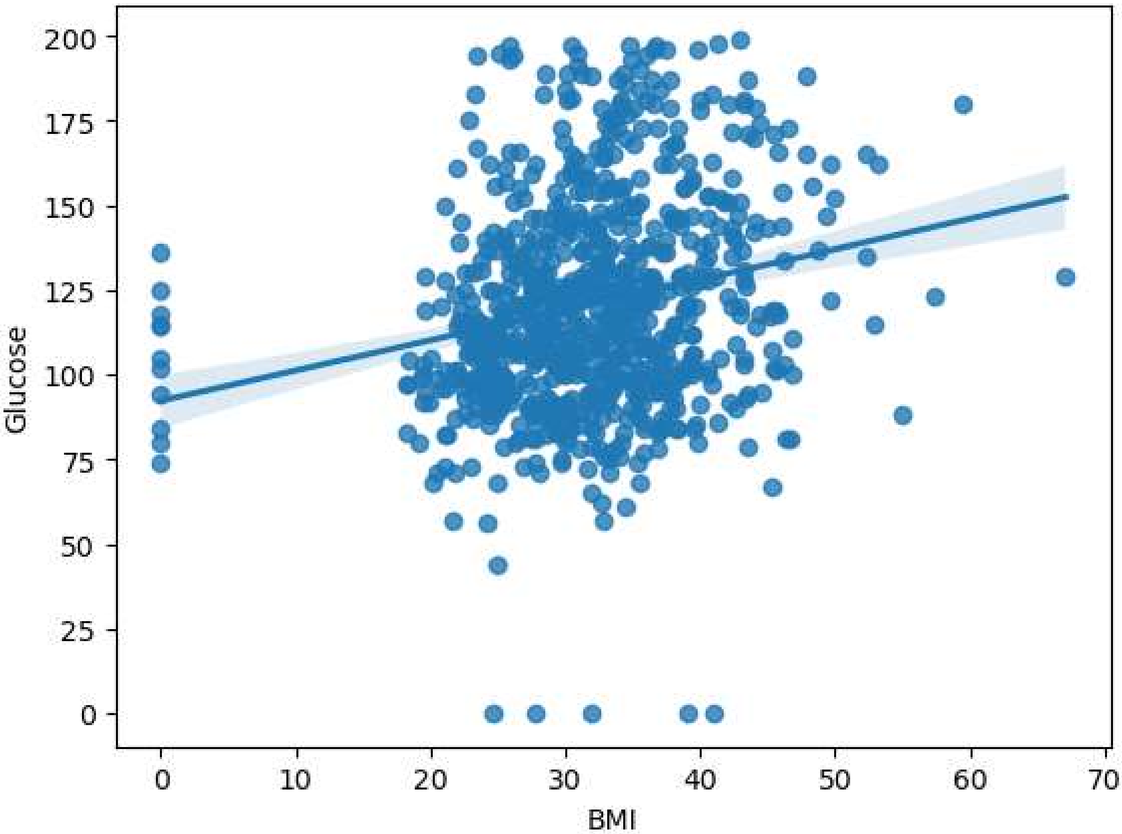
In [24]:
```
sns.boxplot(x='Outcome',y='Insulin',data=df)
```

Out[24]: <Axes: xlabel='Outcome', ylabel='Insulin'>



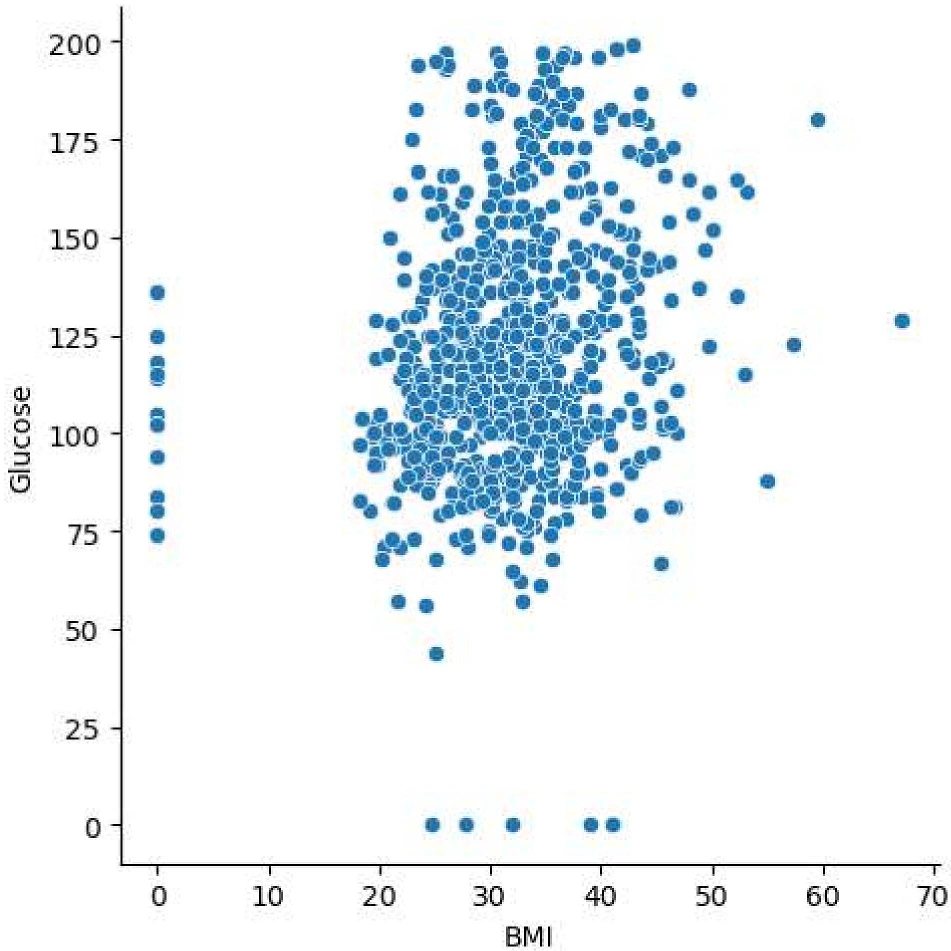
In [25]:
```
sns.regplot(x='BMI', y= 'Glucose', data=df)
```

Out[25]: <Axes: xlabel='BMI', ylabel='Glucose'>



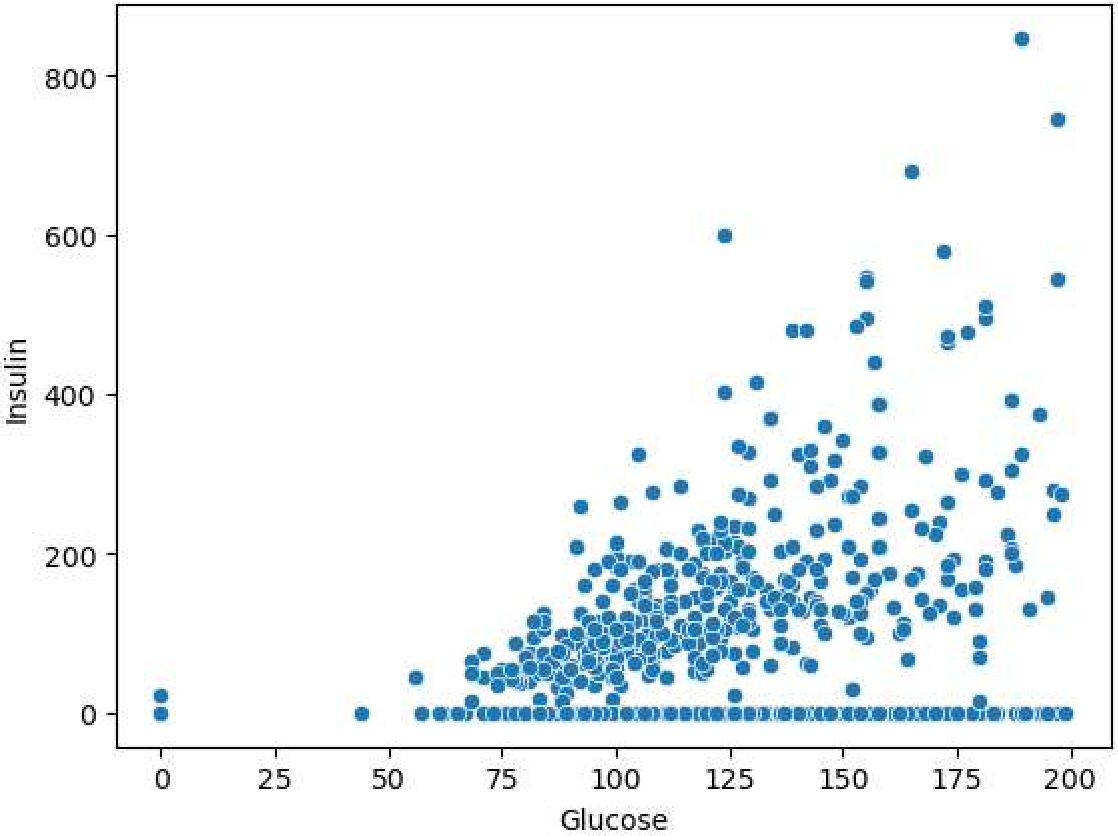
In [27]:
```
sns.relplot(x='BMI', y= 'Glucose', data=df)
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x28b652c1ea0>



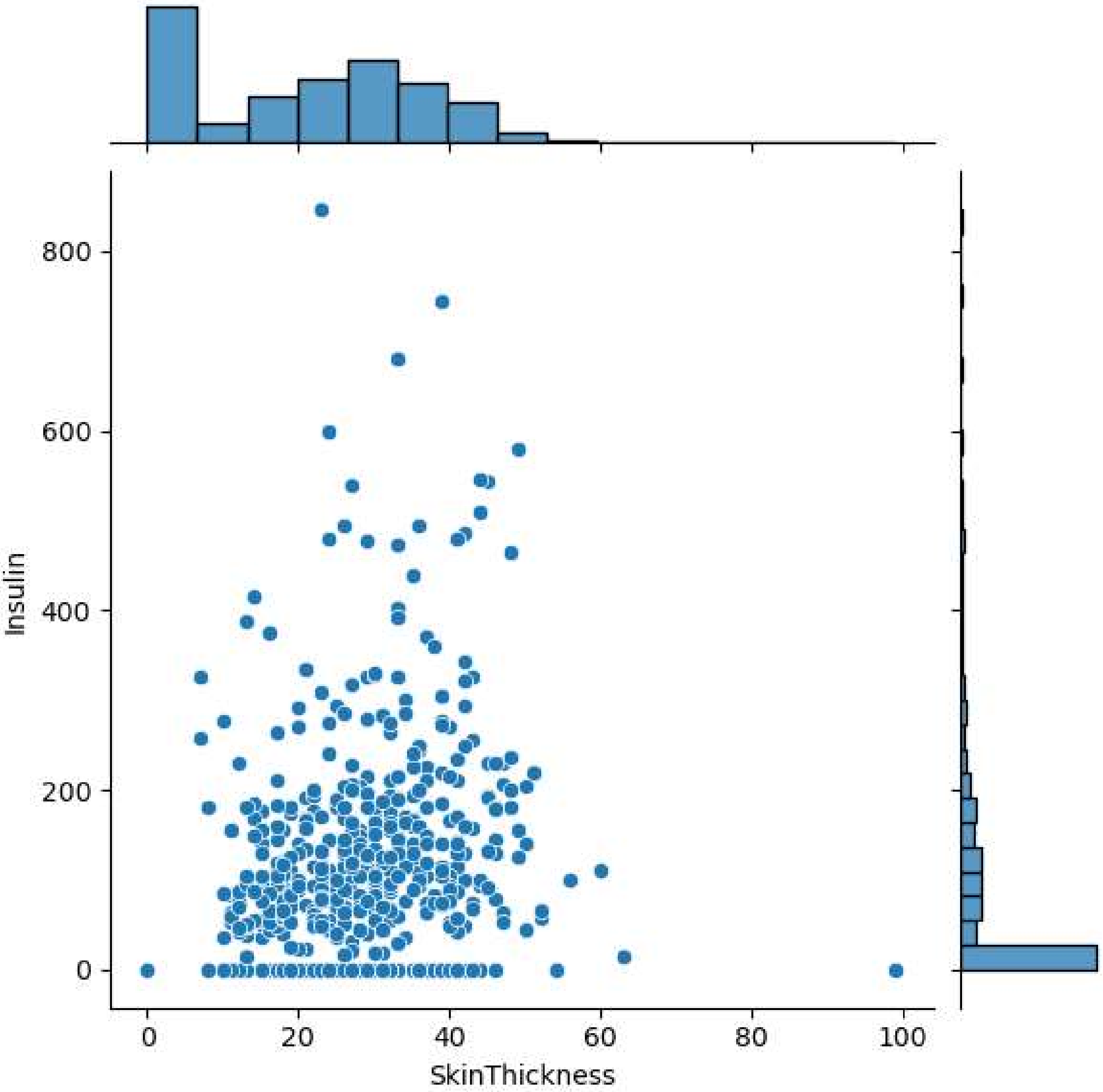
In [28]:
```
sns.scatterplot(x='Glucose', y= 'Insulin', data=df)
```

Out[28]: <Axes: xlabel='Glucose', ylabel='Insulin'>



In [29]:
```
sns.jointplot(x='SkinThickness', y= 'Insulin', data=df)
```

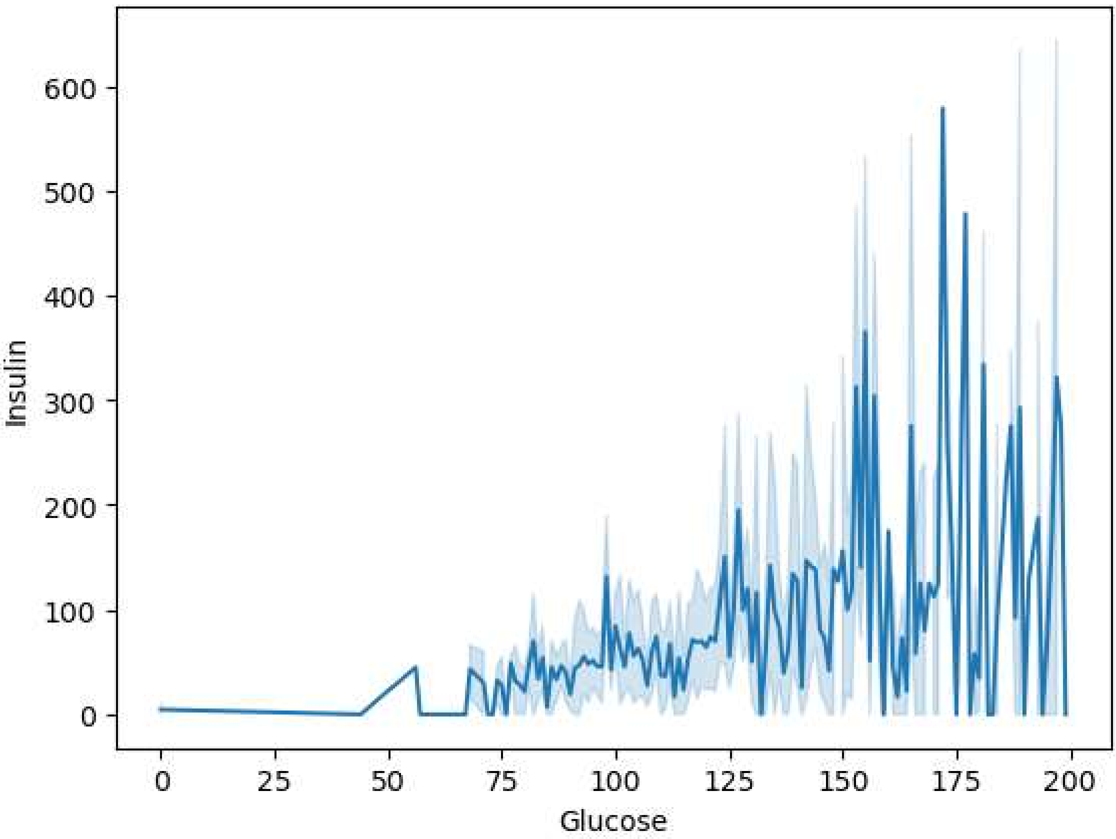
Out[29]: <seaborn.axisgrid.JointGrid at 0x28b6547a9e0>



In [30]:
```
sns.pairplot(df,hue='Outcome')
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x28b654f6ce0>

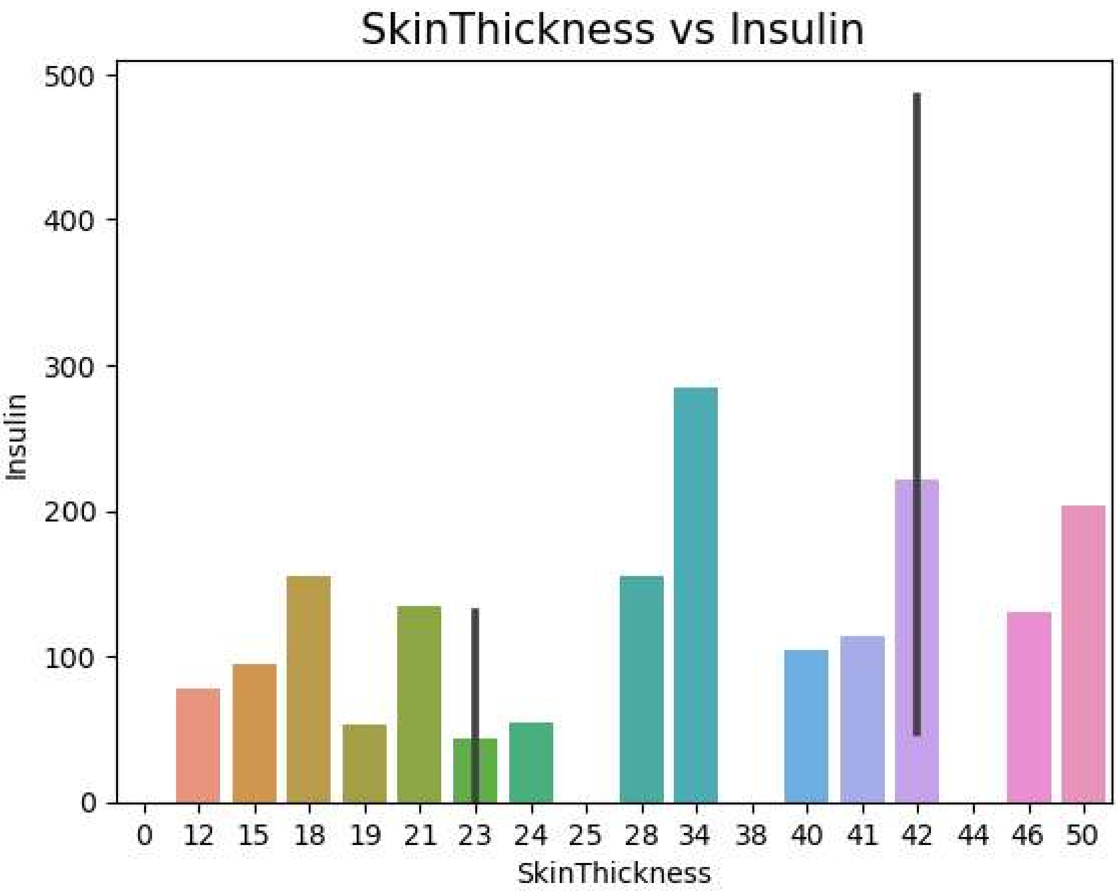
In [31]:
```python
sns.lineplot(x='Glucose', y= 'Insulin', data=df)
```
Out[31]: <Axes: xlabel='Glucose', ylabel='Insulin'>



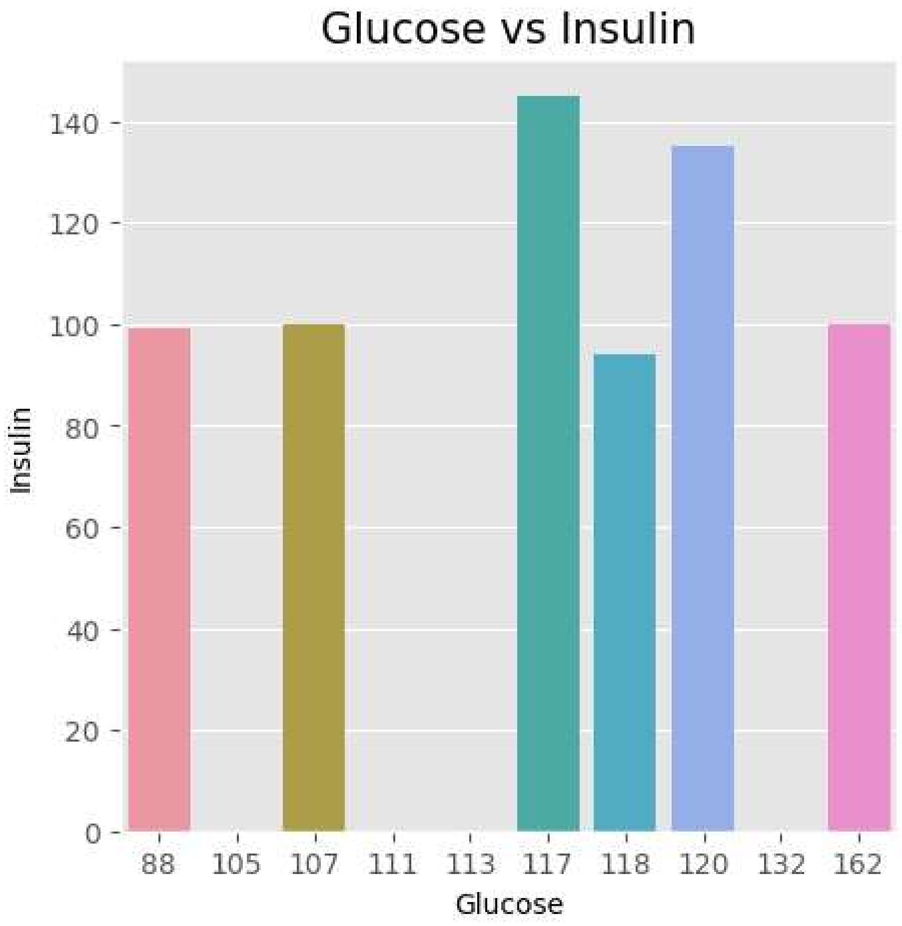
In [32]:
```python
sns.swarmplot(x='Glucose', y= 'Insulin', data=df)
```
```
C:\Users\ashuk\anaconda3\lib\site-packages\seaborn\categorical.py:3544: U
serWarning: 40.0% of the points cannot be placed; you may want to decreas
e the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\ashuk\anaconda3\lib\site-packages\seaborn\categorical.py:3544: U
serWarning: 80.0% of the points cannot be placed; you may want to decreas
e the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\ashuk\anaconda3\lib\site-packages\seaborn\categorical.py:3544: U
serWarning: 16.7% of the points cannot be placed; you may want to decreas
e the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\ashuk\anaconda3\lib\site-packages\seaborn\categorical.py:3544: U
serWarning: 62.5% of the points cannot be placed; you may want to decreas
e the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\ashuk\anaconda3\lib\site-packages\seaborn\categorical.py:3544: U
serWarning: 20.0% of the points cannot be placed; you may want to decreas
e the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

In [33]:

```
sns.barplot(x="SkinThickness", y="Insulin", data=df[150:180])
plt.title("SkinThickness vs Insulin",fontsize=15)
plt.xlabel("SkinThickness")
plt.ylabel("Insulin")
plt.show()
plt.style.use("ggplot")
```



In [34]:
```
plt.figure(figsize=(5,5))
sns.barplot(x="Glucose", y="Insulin", data=df[120:130])
plt.title("Glucose vs Insulin",fontsize=15)
plt.xlabel("Glucose")
plt.ylabel("Insulin")
plt.show()
```



In [37]:
```
x = df.drop(columns = 'Outcome')
y = df['Outcome']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_st
```

In [42]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
C:\Users\ashuk\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:460: ConvergenceWarning: lbfgs failed to 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://sci
kit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

In [43]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [44]:
```
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
LRAcc = accuracy_score(y_pred,y_test)
print('Logistic Regression accuracy is: {:.2f}%'.format(LRAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.84      0.92      0.88       107
           1       0.76      0.62      0.68        47

    accuracy                           0.82       154
   macro avg       0.80      0.77      0.78       154
weighted avg       0.82      0.82      0.82       154

[[98  9]
 [18 29]]
Logistic Regression accuracy is: 82.47%
```

In [45]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=7)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
KNAcc = accuracy_score(y_pred,y_test)
print('KNeighborsClassifier accuracy is: {:.2f}%'.format(KNAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       107
           1       0.61      0.57      0.59        47

    accuracy                           0.76       154
   macro avg       0.72      0.71      0.71       154
weighted avg       0.76      0.76      0.76       154

[[90 17]
 [20 27]]
KNeighborsClassifier accuracy is: 75.97%
```

In [46]:
```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
SVCAcc = accuracy_score(y_pred,y_test)
print('SVC accuracy is: {:.2f}%'.format(SVCAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.81      0.92      0.86       107
           1       0.73      0.51      0.60        47

    accuracy                           0.79       154
   macro avg       0.77      0.71      0.73       154
weighted avg       0.78      0.79      0.78       154

[[98  9]
 [23 24]]
SVC accuracy is: 79.22%
```

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       107
           1       0.67      0.62      0.64        47

    accuracy                           0.79       154
   macro avg       0.76      0.74      0.75       154
weighted avg       0.79      0.79      0.79       154

[[93 14]
 [18 29]]
```

In [48]:
```
from sklearn.metrics import accuracy_score
RFAcc = accuracy_score(y_pred,y_test)
print('RFC accuracy is: {:.2f}%'.format(RFAcc*100))
```

```
RFC accuracy is: 79.22%
```

In [49]:
```
from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
GBCAcc = accuracy_score(y_pred,y_test)
print('GBC accuracy is: {:.2f}%'.format(GBCAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       107
           1       0.68      0.68      0.68        47

    accuracy                           0.81       154
   macro avg       0.77      0.77      0.77       154
weighted avg       0.81      0.81      0.81       154

[[92 15]
 [15 32]]
GBC accuracy is: 80.52%
```

In [50]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
GNBAcc = accuracy_score(y_pred,y_test)
print('GNB accuracy is: {:.2f}%'.format(GNBAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       107
           1       0.67      0.62      0.64        47

    accuracy                           0.79       154
   macro avg       0.76      0.74      0.75       154
weighted avg       0.79      0.79      0.79       154

[[93 14]
 [18 29]]
GNB accuracy is: 79.22%
```

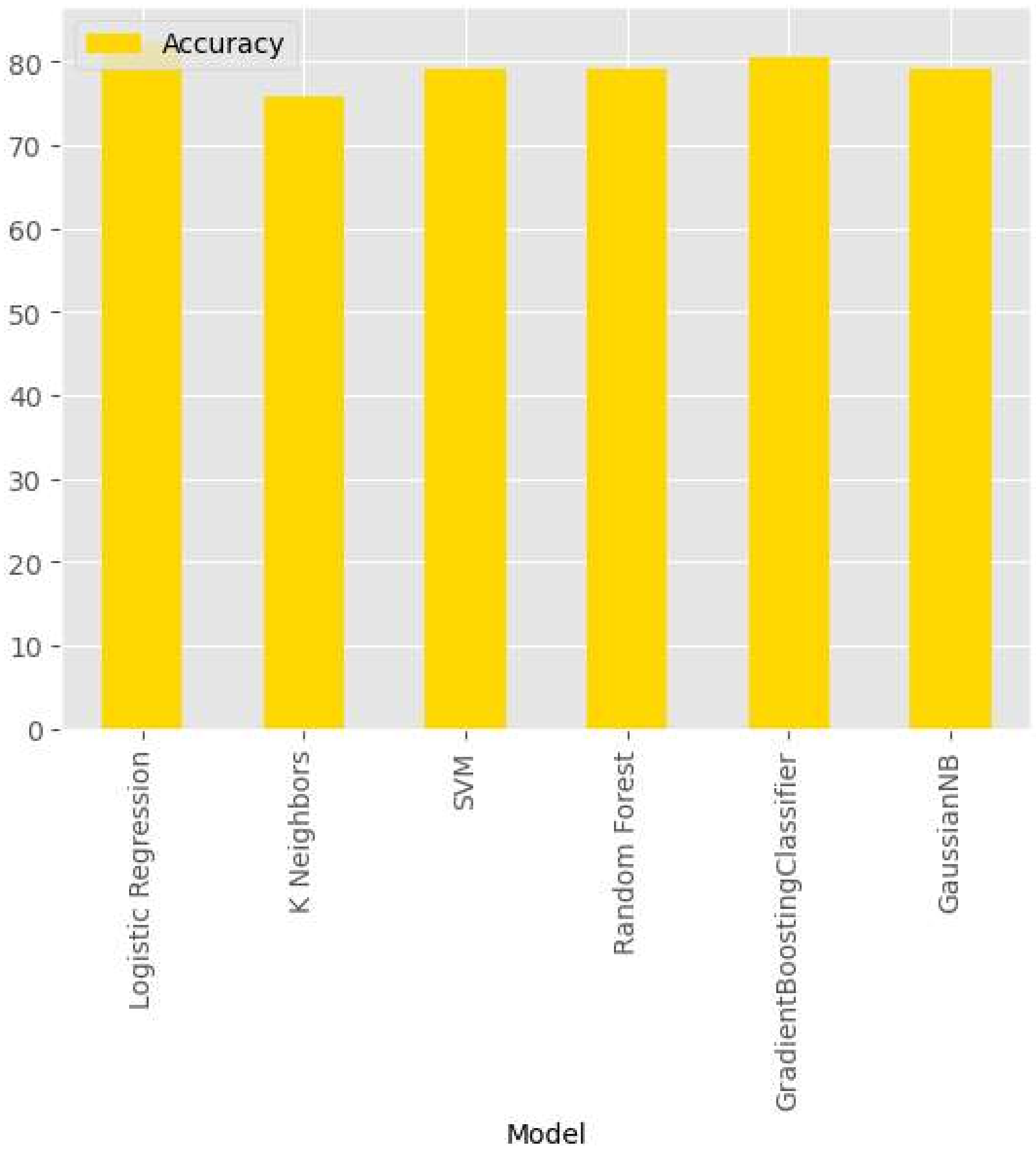
In [52]:
```
compare = pd.DataFrame({'Model': ['Logistic Regression', 'K Neighbors', 'SVM
'Accuracy': [LRAcc*100, KNAcc*100, SVCAcc*100,RFAcc*100, GBCAcc*100, GNBAcc*
compare.sort_values(by='Accuracy', ascending=False)
```

Out[52]:

	Model	Accuracy
0	Logistic Regression	82.467532
4	GradientBoostingClassifier	80.519481
2	SVM	79.220779
3	Random Forest	79.220779
5	GaussianNB	79.220779
1	K Neighbors	75.974026

In [54]:
```
compare.plot(x='Model', y='Accuracy', kind='bar', color='gold')
```

Out[54]: <Axes: xlabel='Model'>



In [ ]: