Health_Care_Project

April 12, 2023

0.1 Importing Required Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#Load the data
df=pd.read_csv('health_care_diabetes.csv')
df.head()
```

[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

   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
```

[3]:
```
df.isnull().sum()
```

[3]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
```

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

There is no Missing value. The data is clean.

[4]:
```python
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
```

[5]:
```python
df.describe()
```

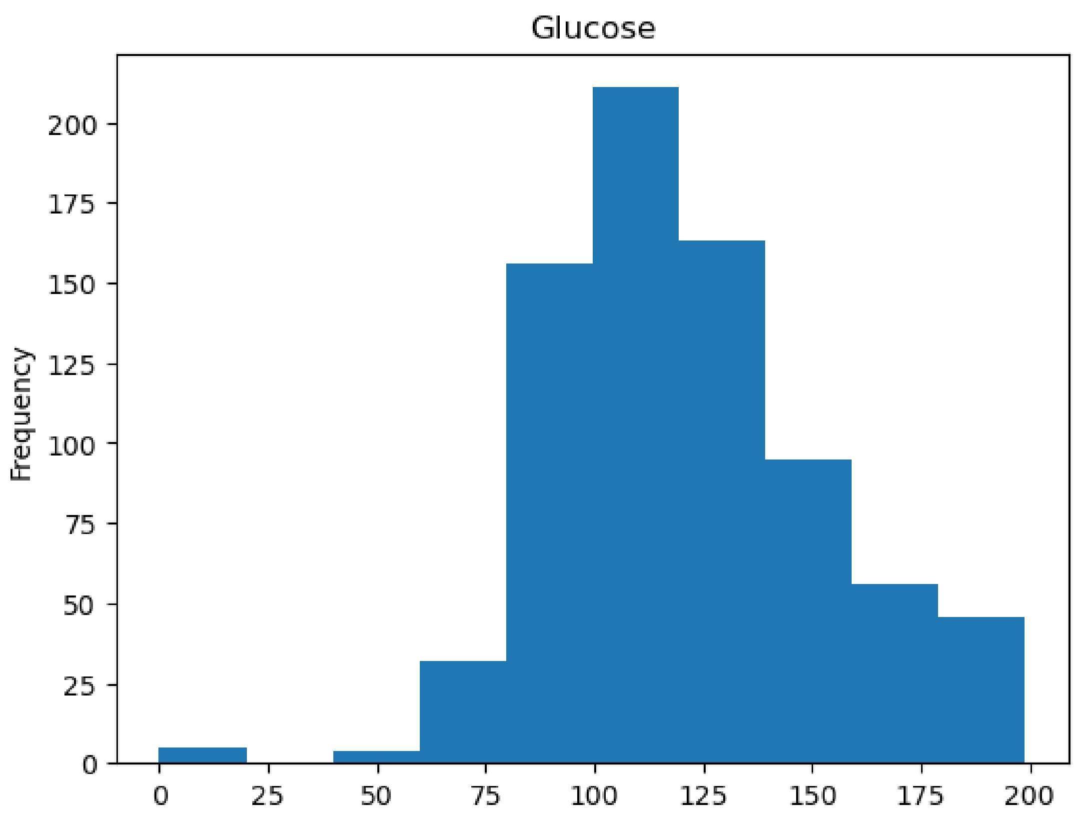
[5]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
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
```

[6]:
```
df['Glucose'].value_counts()
```

[6]:
```
99     17
100    17
111    14
129    14
125    14
       ..
191     1
177     1
44      1
62      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

[7]:
```
df['Glucose'].plot(kind='hist')
plt.title('Glucose')
plt.show()
```

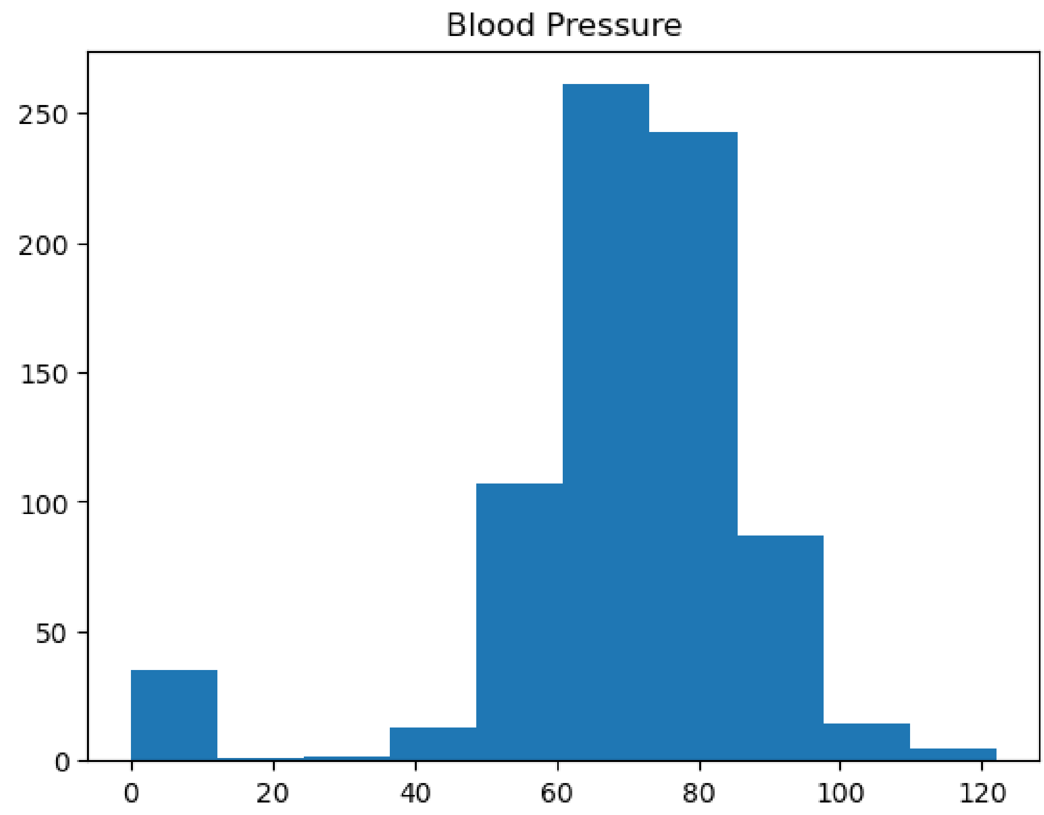


[8]:
```
df['BloodPressure'].value_counts()
```

[8]:
```
70     57
74     52
78     45
68     45
72     44
64     43
80     40
76     39
60     37
0      35
62     34
66     30
82     30
88     25
84     23
90     22
86     21
58     21
50     13
56     12
52     11
54     11
75      8
92      8
65      7
85      6
94      6
48      5
96      4
44      4
100     3
106     3
98      3
110     3
55      2
108     2
104     2
46      2
30      2
122     1
95      1
102     1
61      1
24      1
38      1
40      1
114     1
```

```
Name: BloodPressure, dtype: int64
```

[9]:
```python
plt.hist(x=df['BloodPressure'])
plt.title('Blood Pressure')
plt.show()
```

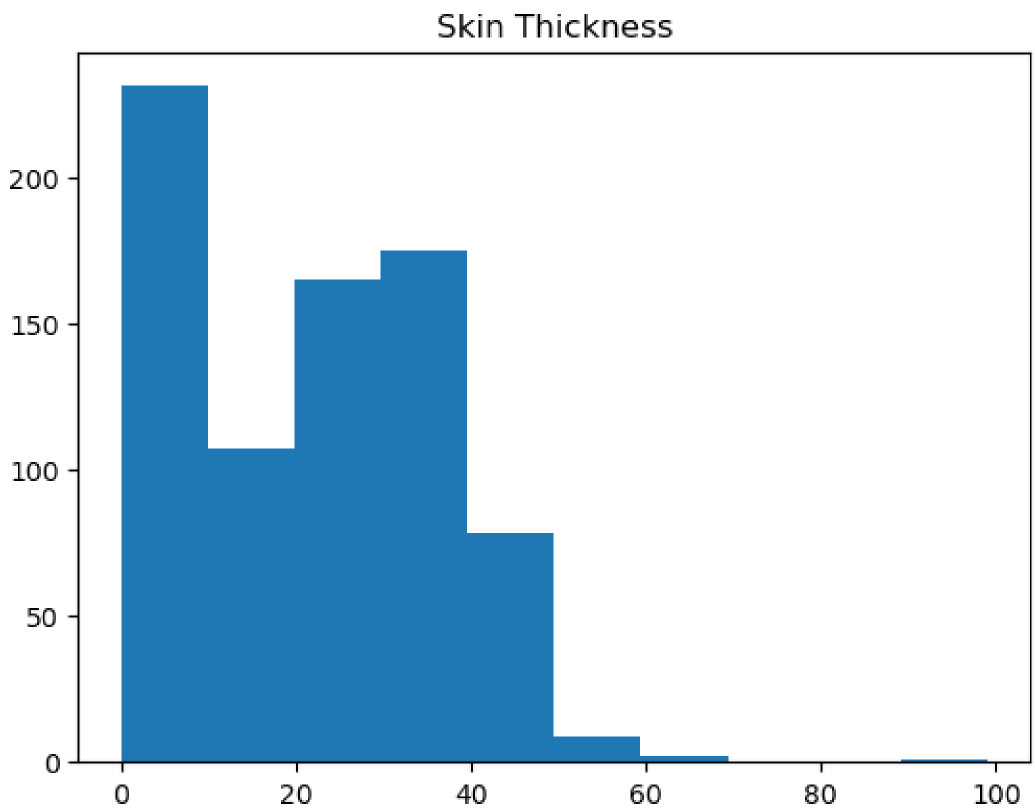


[10]:
```python
df['SkinThickness'].value_counts()
```

[10]:
```
0      227
32      31
30      27
27      23
23      22
33      20
28      20
18      20
31      19
19      18
39      18
29      17
40      16
```

```
25      16
26      16
22      16
37      16
41      15
35      15
36      14
15      14
17      14
20      13
24      12
42      11
13      11
21      10
46       8
34       8
12       7
38       7
11       6
43       6
16       6
45       6
14       6
44       5
10       5
48       4
47       4
49       3
50       3
8        2
7        2
52       2
54       2
63       1
60       1
56       1
51       1
99       1
Name: SkinThickness, dtype: int64
```

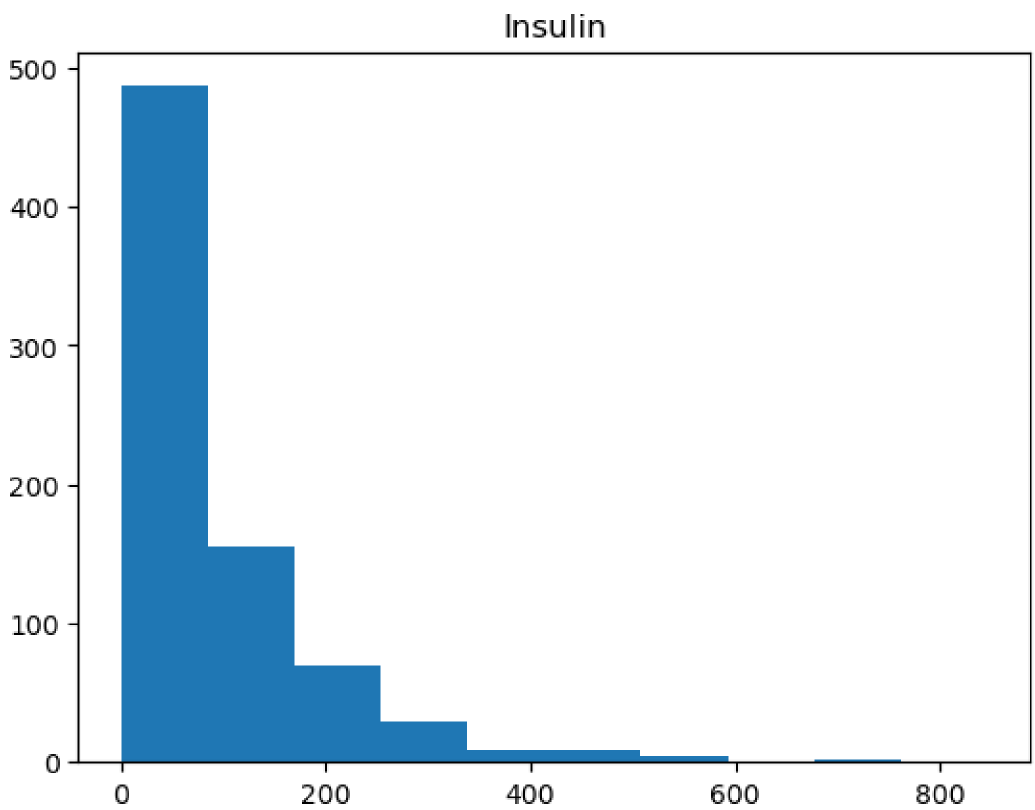
[11]:
```
plt.hist(x=df['SkinThickness'])
plt.title('Skin Thickness')
plt.show()
```



[12]:
```
df['Insulin'].value_counts()
```

[12]:
```
0      374
105     11
130      9
140      9
120      8
      ...
73       1
171      1
255      1
52       1
112      1
Name: Insulin, Length: 186, dtype: int64
```

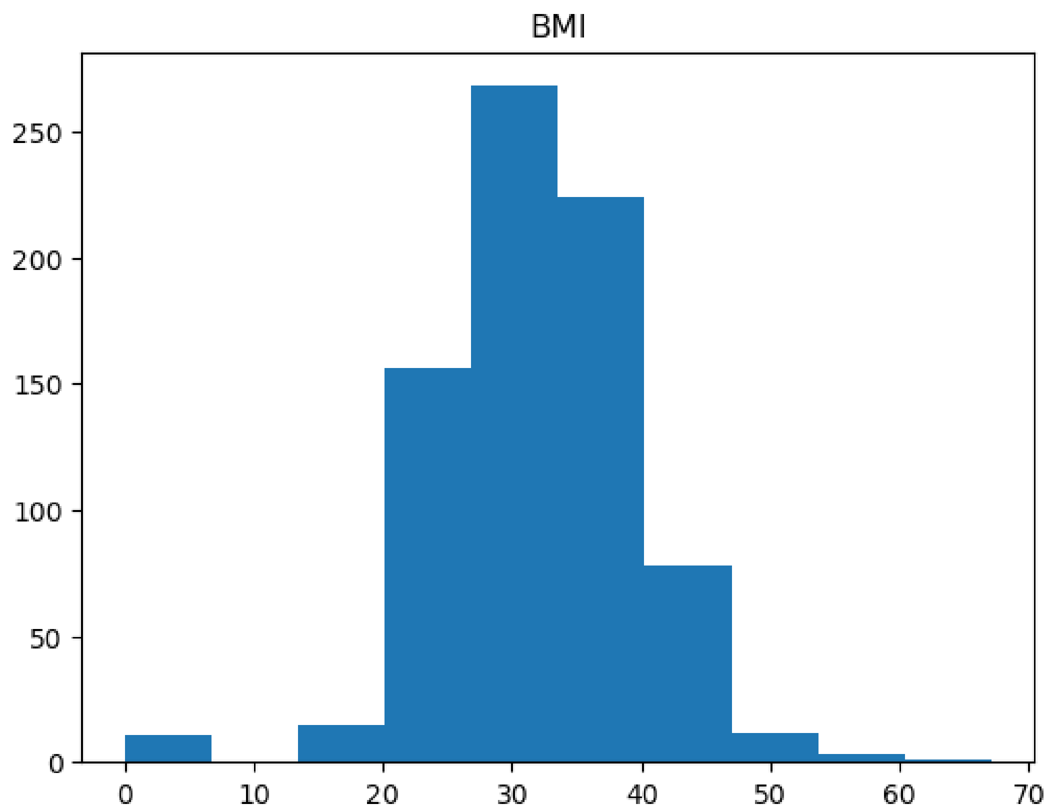
[13]:
```
plt.hist(x=df['Insulin'])
plt.title('Insulin')
plt.show()
```



[14]:
```
df['BMI'].value_counts()
```

[14]:
```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
        ..
36.7     1
41.8     1
42.6     1
42.8     1
46.3     1
Name: BMI, Length: 248, dtype: int64
```

[15]:
```
plt.hist(x=df['BMI'])
plt.title('BMI')
plt.show()
```



[16]:
```
df.describe().transpose()
```

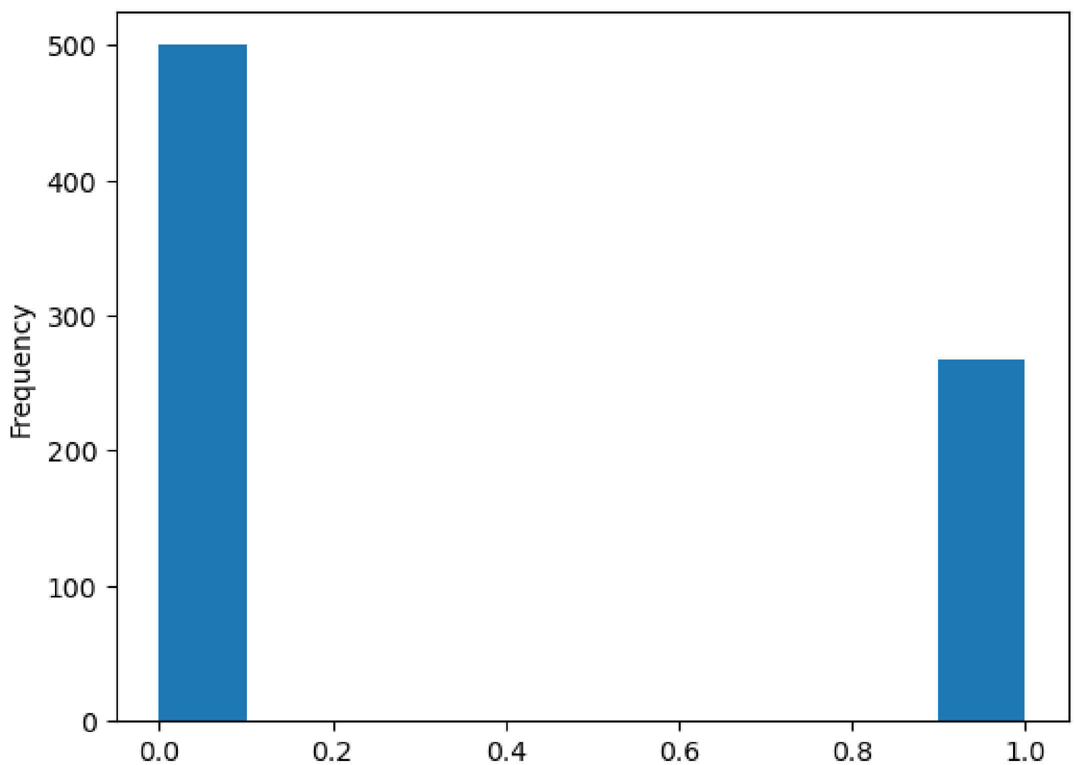
[16]:
```
                          count        mean         std     min       25%  \
Pregnancies               768.0    3.845052    3.369578   0.000   1.00000
Glucose                   768.0  120.894531   31.972618   0.000  99.00000
BloodPressure             768.0   69.105469   19.355807   0.000  62.00000
SkinThickness             768.0   20.536458   15.952218   0.000   0.00000
Insulin                   768.0   79.799479  115.244002   0.000   0.00000
BMI                       768.0   31.992578    7.884160   0.000  27.30000
DiabetesPedigreeFunction  768.0    0.471876    0.331329   0.078   0.24375
Age                       768.0   33.240885   11.760232  21.000  24.00000
Outcome                   768.0    0.348958    0.476951   0.000   0.00000

                               50%        75%     max
Pregnancies                 3.0000    6.00000   17.00
Glucose                   117.0000  140.25000  199.00
BloodPressure              72.0000   80.00000  122.00
SkinThickness              23.0000   32.00000   99.00
Insulin                    30.5000  127.25000  846.00
BMI                        32.0000   36.60000   67.10
DiabetesPedigreeFunction    0.3725    0.62625    2.42
```

```
Age                          29.0000    41.00000    81.00
Outcome                       0.0000     1.00000     1.00
```

[17]:
```
#Checking the Balance of the data by outcome
df['Outcome'].plot(kind='hist')
```

[17]:
```
<Axes: ylabel='Frequency'>
```



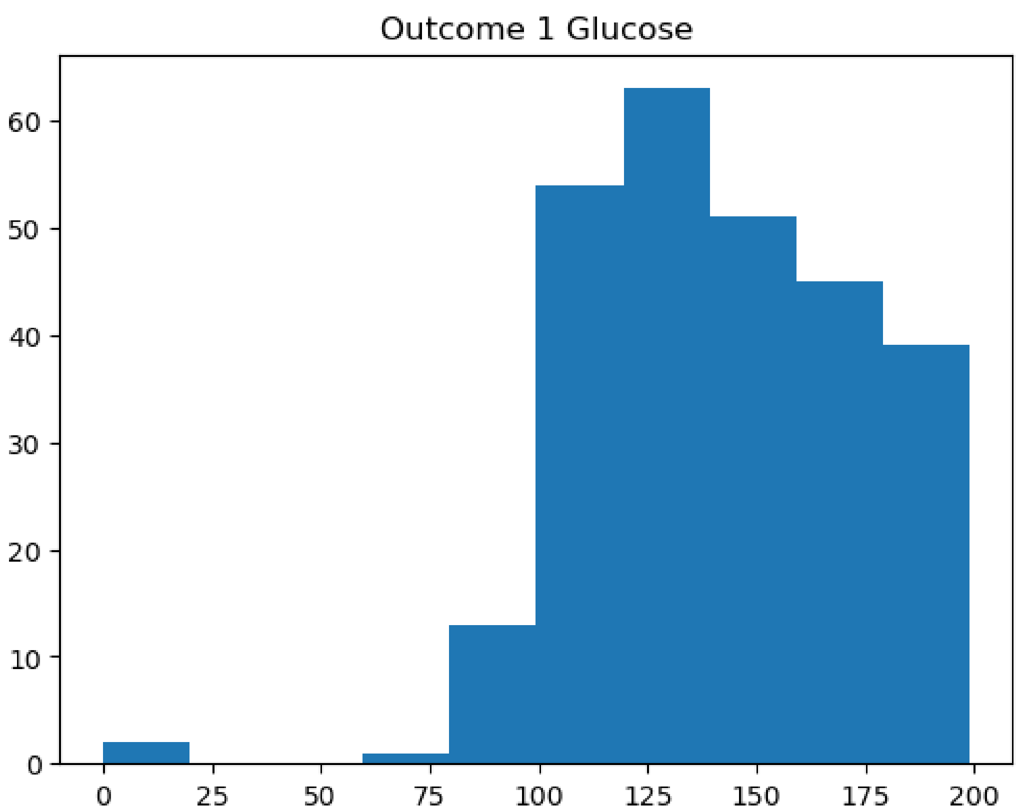
The Data is not much Imbalanced. Lets Check the values of other column where the Outcome column has 1. And lets plot them and check their difference with the plots that we created earlier. And let us look in details of this plot.

[18]:
```
Outcome_1=df[df['Outcome']==1]
Outcome_1.head()
```

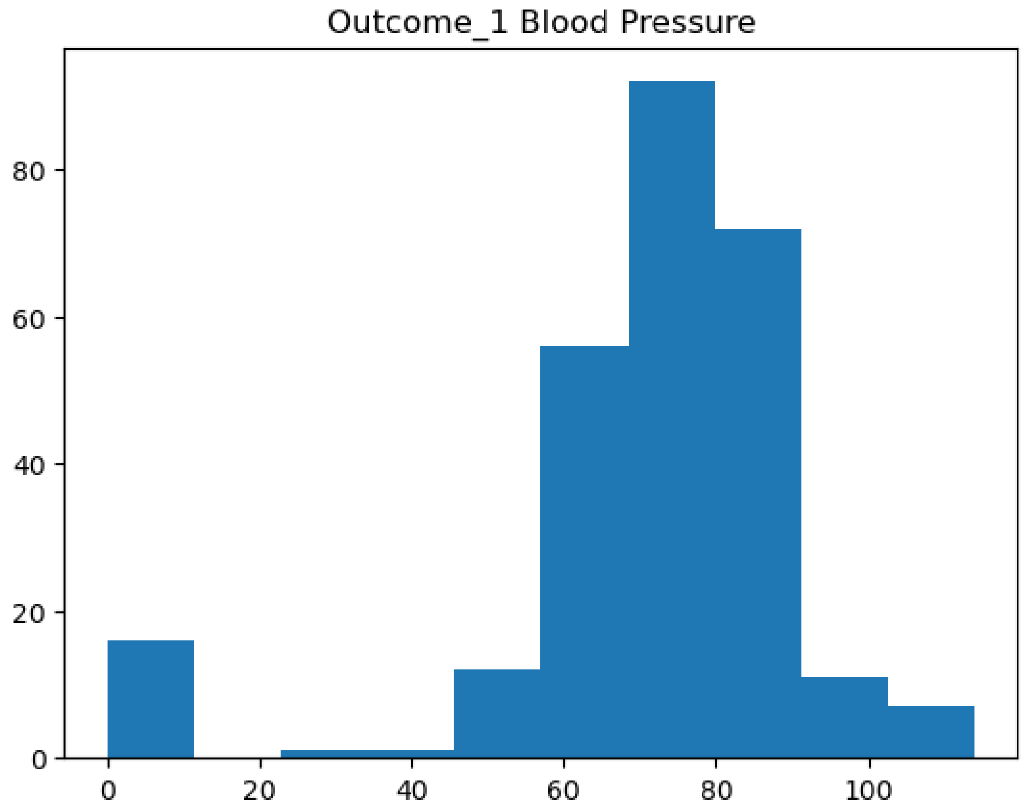
[18]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
2            8      183             64              0        0  23.3
4            0      137             40             35      168  43.1
6            3       78             50             32       88  31.0
8            2      197             70             45      543  30.5
```

```
   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
2                     0.672   32        1
4                     2.288   33        1
6                     0.248   26        1
8                     0.158   53        1
```

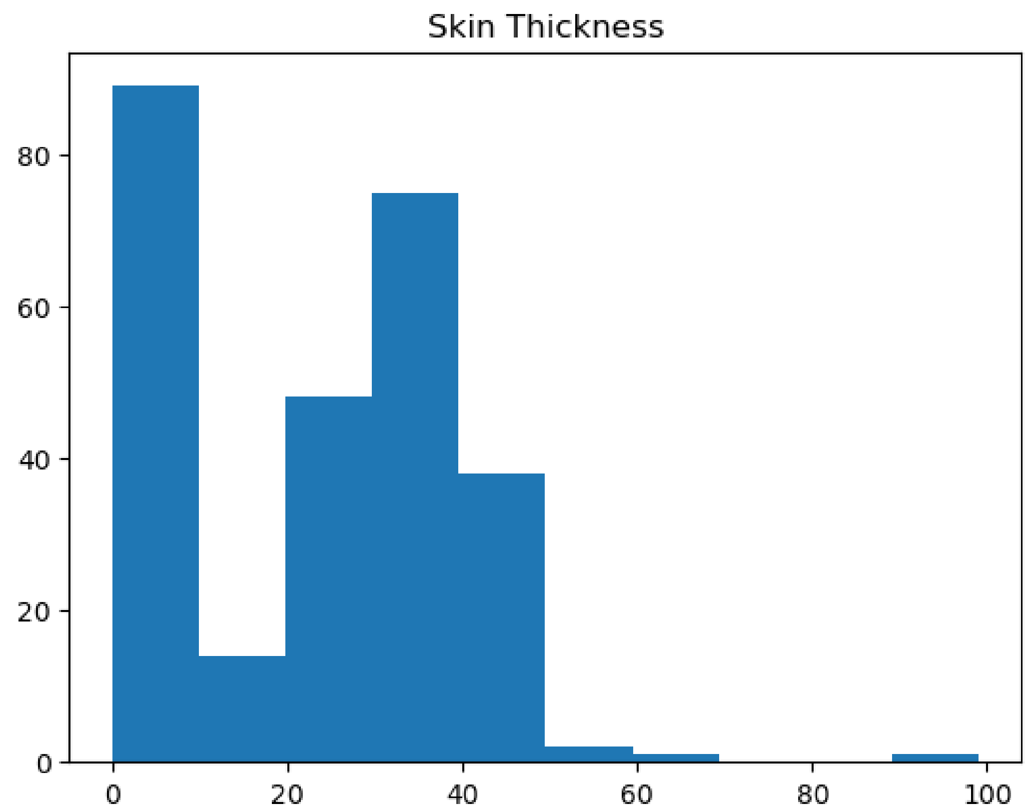
[19]:
```
plt.hist(x=Outcome_1['Glucose'])
plt.title('Outcome 1 Glucose')
plt.show()
```



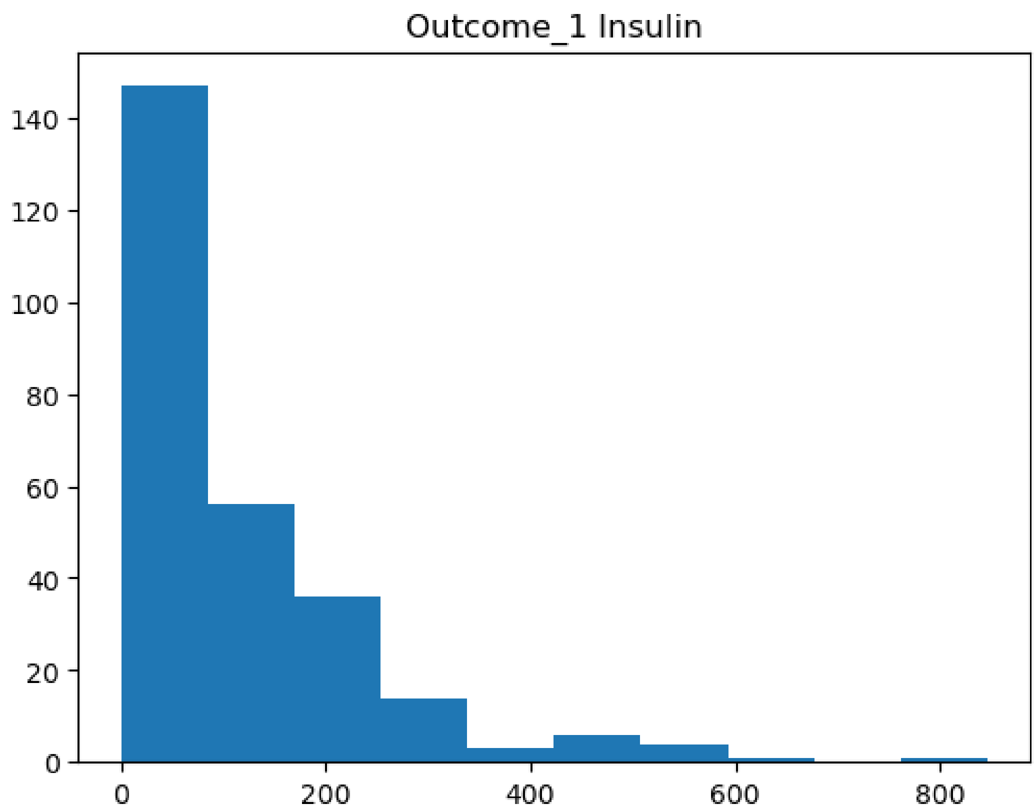
[20]:
```
plt.hist(x=Outcome_1['BloodPressure'])
plt.title('Outcome_1 Blood Pressure')
plt.show()
```



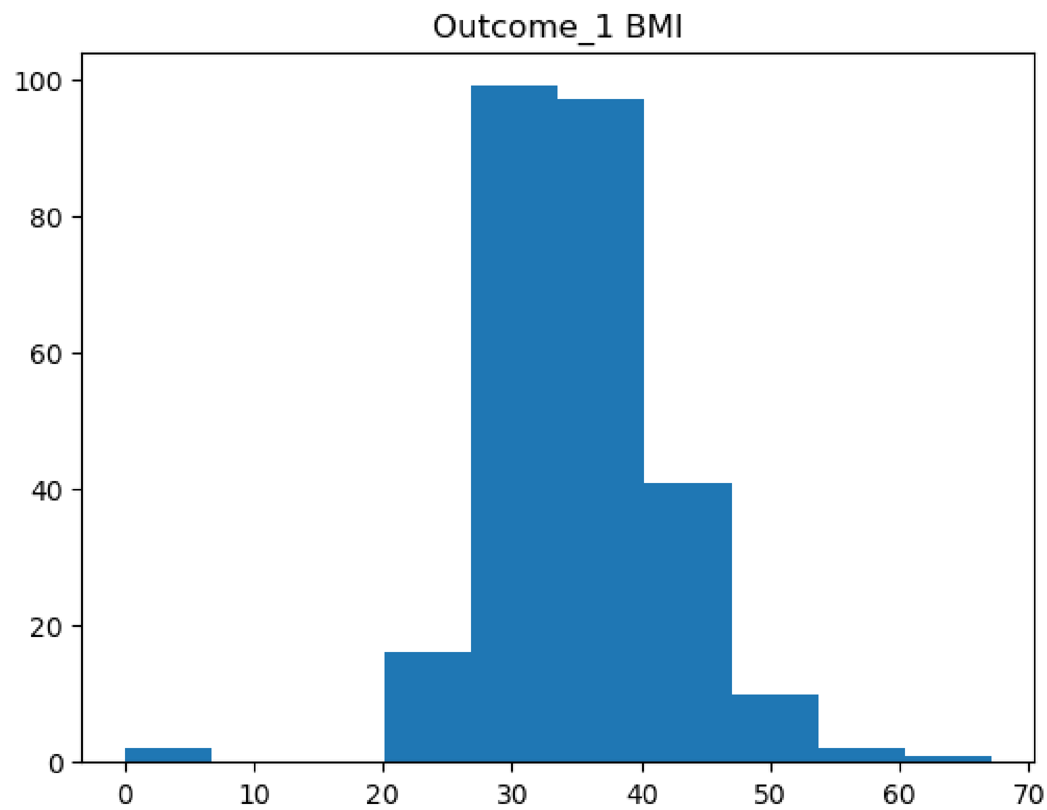
[21]:
```
plt.hist(x=Outcome_1['SkinThickness'])
plt.title('Skin Thickness')
plt.show()
```




```
[22]: plt.hist(x=Outcome_1['Insulin'])
      plt.title('Outcome_1 Insulin')
      plt.show()
```



[23]:
```
plt.hist(x=Outcome_1['BMI'])
plt.title('Outcome_1 BMI')
plt.show()
```



There are Zeros in every column. To drop this values will make the data too small and we will miss some valuable data. So We can replace the zeros with the mean of the repsective columns so that no data is missed.

[24]:
```
df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())
df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())
df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].mean())
df['Insulin']=df['Insulin'].replace(0,df['Insulin'].mean())
df['BMI']=df['BMI'].replace(0,df['BMI'].mean())
```

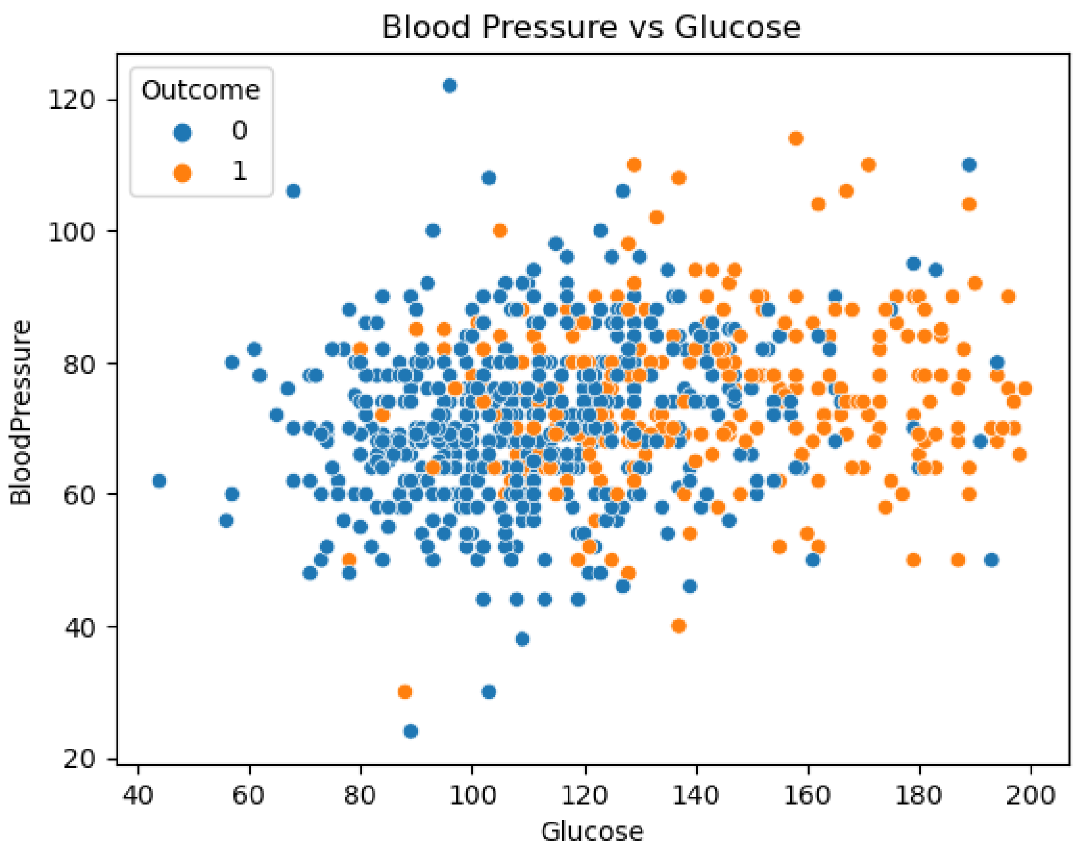
[25]:
```
print((df['Glucose'].values==0).sum())
print((df['BloodPressure'].values==0).sum())
print((df['SkinThickness'].values==0).sum())
print((df['Insulin'].values==0).sum())
print((df['BMI'].values==0).sum())
```

```
0
0
0
0
0
```

Now we can see that all the zeros has been replaced by the mean of the respective columns.

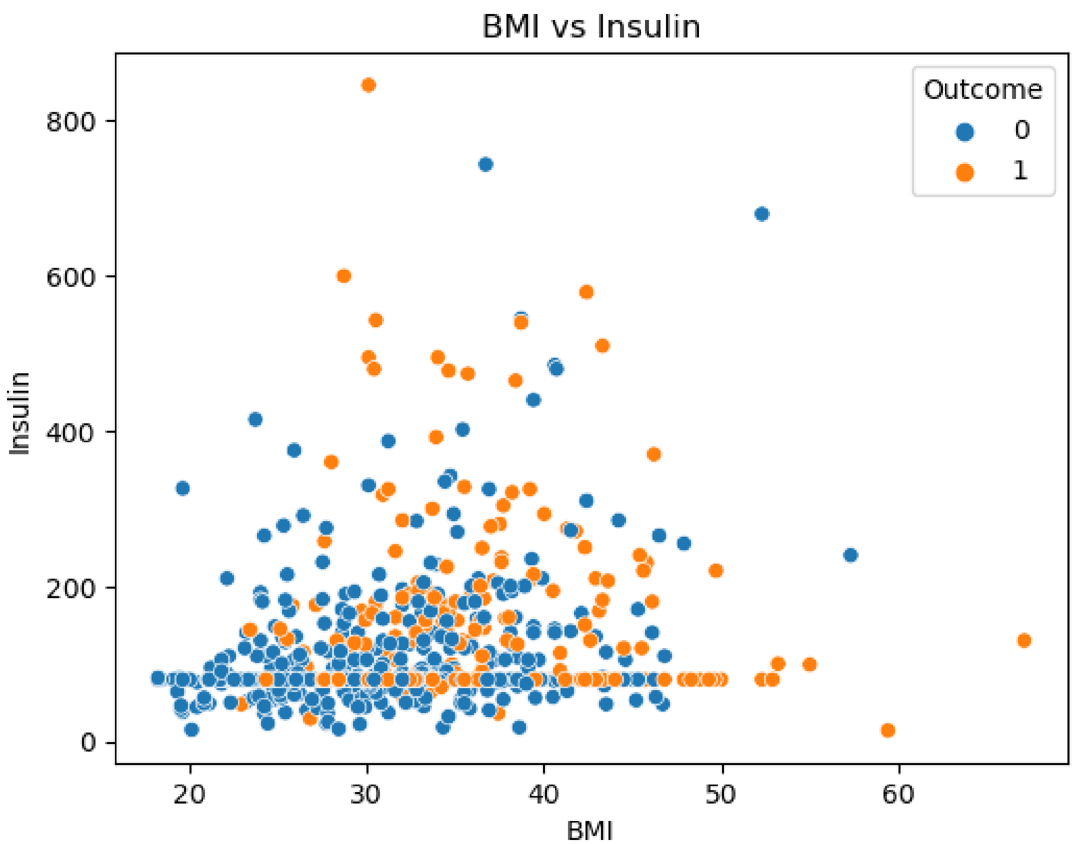
0.1.1 Lets Create Scatter Plot to understand the relation between different variables.

[26]:
```
sns.scatterplot(x='Glucose',y='BloodPressure',hue='Outcome',data=df)
plt.title('Blood Pressure vs Glucose')
plt.show()
```



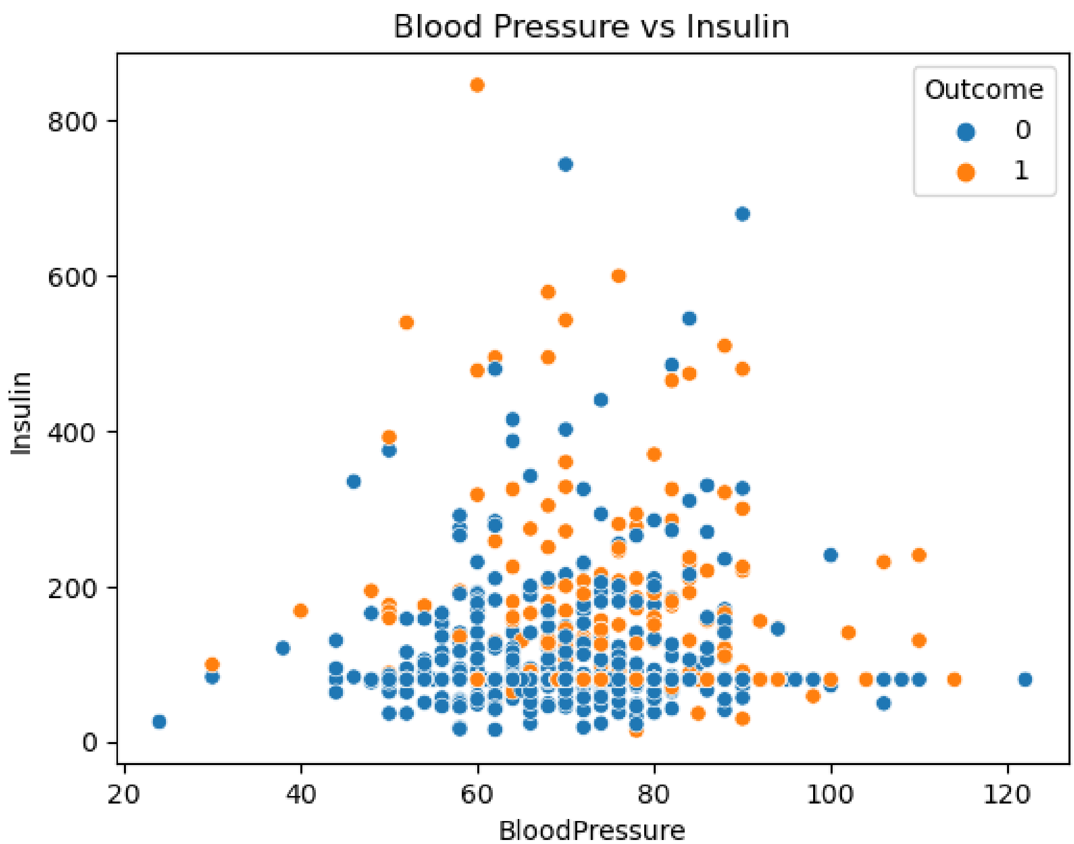
If the Glucose is high with an average of Blood Pressure has the Diabetes. We can see that even if Blood Pressure is about normal, there is chance of getting Diabetes if the level of Glucose is high.

[27]:
```
sns.scatterplot(x='BMI',y='Insulin',hue='Outcome',data=df)
plt.title('BMI vs Insulin')
plt.show()
```

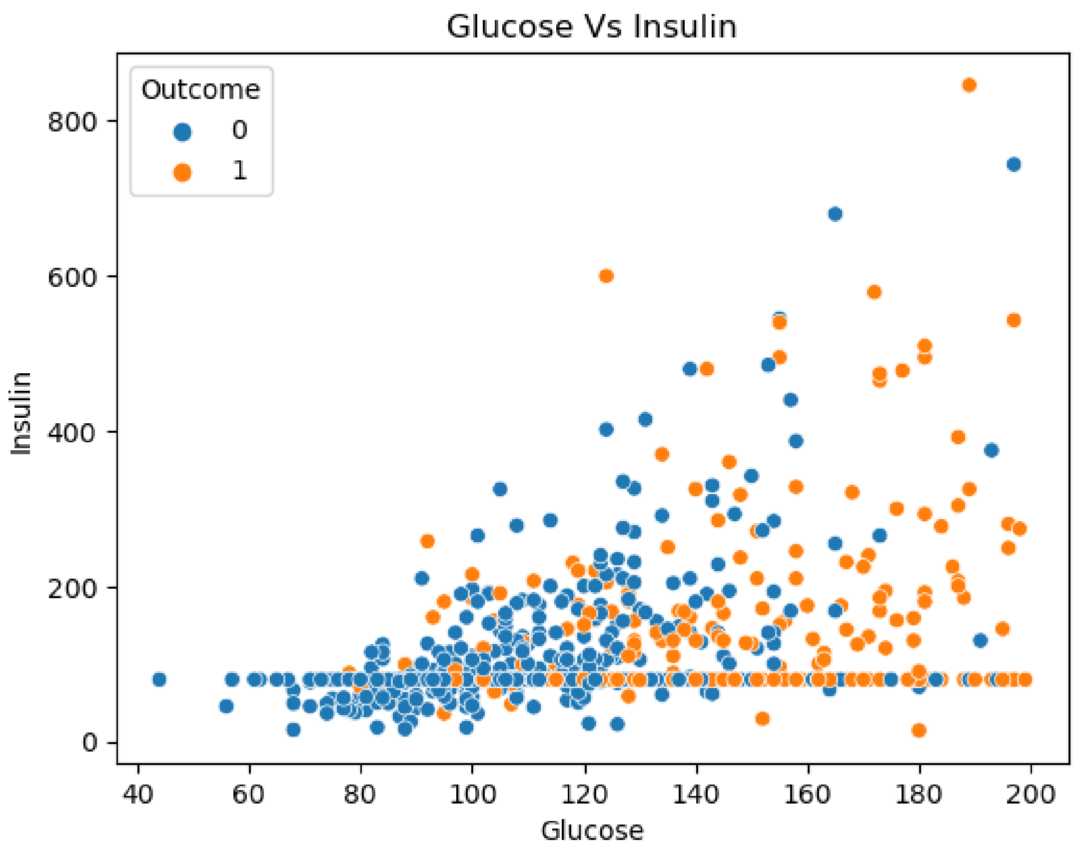


If the Insulin is low, the person will be Diabetic. The body mass index will also effect when it is above average. The person with low Insulin and High BMI has the chance of getting Diabetes. Most of our data is spread across the Low Area of the insulin.

[28]:
```
sns.scatterplot(x='BloodPressure',y='Insulin',hue='Outcome',data=df)
plt.title('Blood Pressure vs Insulin')
plt.show()
```



[29]:
```
sns.scatterplot(x='Glucose',y='Insulin',hue='Outcome',data=df)
plt.title('Glucose Vs Insulin')
plt.show()
```



If Glucose is high there is a great chance of getting Diabetics. And here we can see that There is line in the plot that the Outcome is one where the glucose level is high.

[30]:
```
df.corr()
```

[30]:

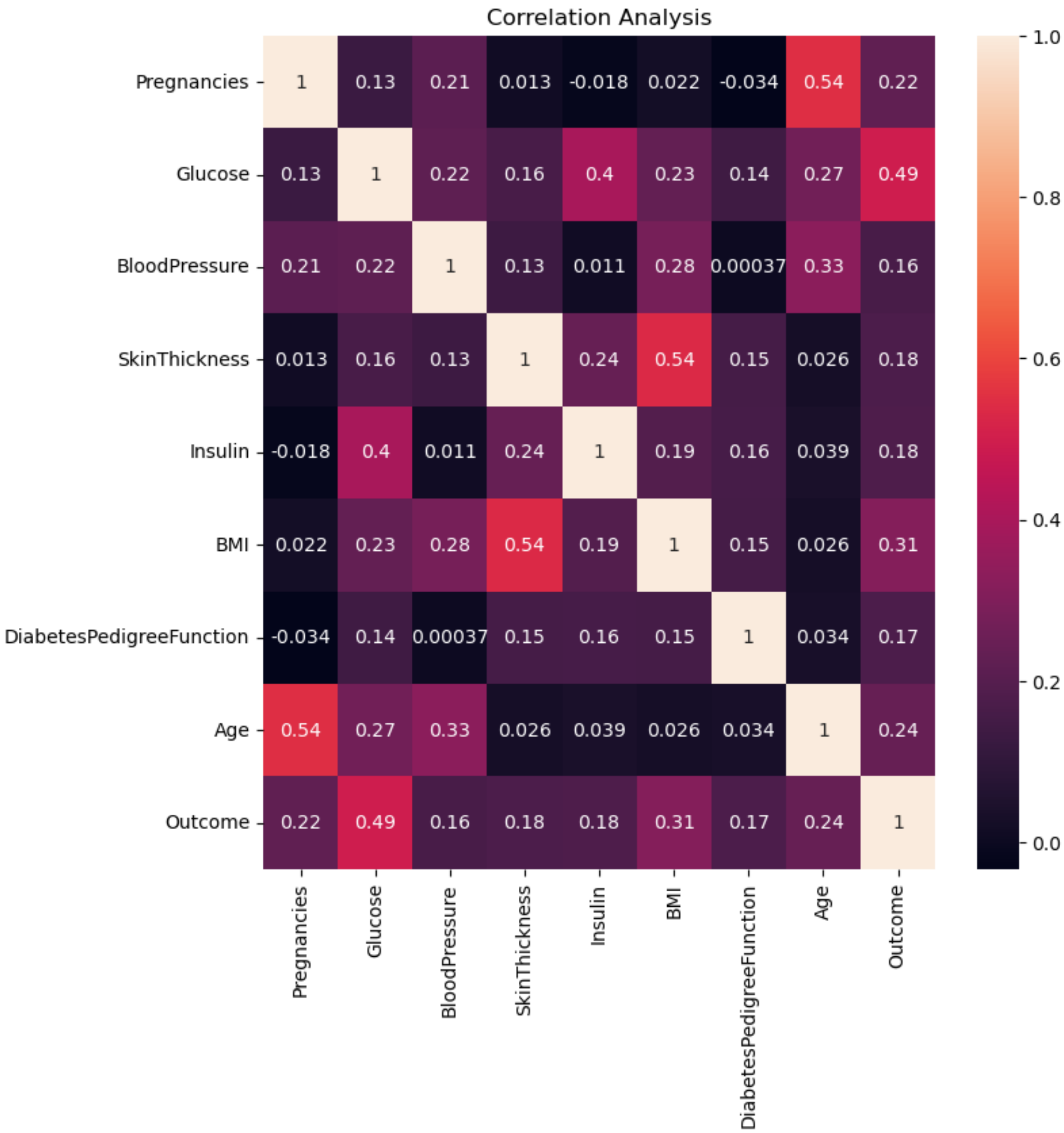
	Pregnancies	Glucose	BloodPressure	SkinThickness	\
Pregnancies	1.000000	0.127964	0.208984	0.013376	
Glucose	0.127964	1.000000	0.219666	0.160766	
BloodPressure	0.208984	0.219666	1.000000	0.134155	
SkinThickness	0.013376	0.160766	0.134155	1.000000	
Insulin	-0.018082	0.396597	0.010926	0.240361	
BMI	0.021546	0.231478	0.281231	0.535703	
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	
Age	0.544341	0.266600	0.326740	0.026423	
Outcome	0.221898	0.492908	0.162986	0.175026	

	Insulin	BMI	DiabetesPedigreeFunction	\
Pregnancies	-0.018082	0.021546	-0.033523	
Glucose	0.396597	0.231478	0.137106	
BloodPressure	0.010926	0.281231	0.000371	

```
SkinThickness             0.240361  0.535703                  0.154961
Insulin                   1.000000  0.189856                  0.157806
BMI                       0.189856  1.000000                  0.153508
DiabetesPedigreeFunction  0.157806  0.153508                  1.000000
Age                       0.038652  0.025748                  0.033561
Outcome                   0.179185  0.312254                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.266600  0.492908
BloodPressure             0.326740  0.162986
SkinThickness             0.026423  0.175026
Insulin                   0.038652  0.179185
BMI                       0.025748  0.312254
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

[31]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
plt.title('Correlation Analysis')
plt.show()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.21	0.013	-0.018	0.022	-0.034	0.54	0.22
Glucose	0.13	1	0.22	0.16	0.4	0.23	0.14	0.27	0.49
BloodPressure	0.21	0.22	1	0.13	0.011	0.28	0.00037	0.33	0.16
SkinThickness	0.013	0.16	0.13	1	0.24	0.54	0.15	0.026	0.18
Insulin	-0.018	0.4	0.011	0.24	1	0.19	0.16	0.039	0.18
BMI	0.022	0.23	0.28	0.54	0.19	1	0.15	0.026	0.31
DiabetesPedigreeFunction	-0.034	0.14	0.00037	0.15	0.16	0.15	1	0.034	0.17
Age	0.54	0.27	0.33	0.026	0.039	0.026	0.034	1	0.24
Outcome	0.22	0.49	0.16	0.18	0.18	0.31	0.17	0.24	1

In this Correlation Matrix, Glucose has the highest correlation with the Outcome Column. Glucose has 0.49 correlation with Outcome. BMI has the second highest correlation of 0.31 with Outcome Column. The insulin has correlation of 0.4 with Glucose. The BMI, Blood Pressure and age has a slightly correlation with Glucose which is correlated with Outcome.

0.1.2 Lets do the spliting of the Dataset into Training and testing

[32]:
```
X=df.iloc[:,[0,1,2,3,4,5,6,7]]
y=df.iloc[:,8]
```

[33]:
```
X.head()
```

[33]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0            6    148.0           72.0      35.000000   79.799479  33.6
1            1     85.0           66.0      29.000000   79.799479  26.6
2            8    183.0           64.0      20.536458   79.799479  23.3
3            1     89.0           66.0      23.000000   94.000000  28.1
4            0    137.0           40.0      35.000000  168.000000  43.1

   DiabetesPedigreeFunction  Age
0                     0.627   50
1                     0.351   31
2                     0.672   32
3                     0.167   21
4                     2.288   33
```

[34]:
```
y.head()
```

[34]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

[35]:
```
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)
```

[36]:
```
print(X_train.shape,X_test.shape)
```

```
(614, 8) (154, 8)
```

[37]:
```
print(y_train.shape,y_test.shape)
```

```
(614,) (154,)
```

Logistic Regression Model

[38]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
```

[38]:
```
LogisticRegression()
```

[39]:
```
y_pred=lr.predict(X_test)
```

```
[40]: print(lr.score(X_train,y_train))
      print(lr.score(X_test,y_test))
```

```
0.7736156351791531
0.7792207792207793
```

```
[41]: from sklearn.metrics import confusion_matrix
      cm=confusion_matrix(y_test,y_pred)
      cm
```

```
[41]: array([[88,  8],
             [26, 32]], dtype=int64)
```

The model has predicted 94 true positive and 30 true negative.

```
[42]: from sklearn.metrics import classification_report
      print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.92      0.84        96
           1       0.80      0.55      0.65        58

    accuracy                           0.78       154
   macro avg       0.79      0.73      0.75       154
weighted avg       0.78      0.78      0.77       154
```

```
[43]: from sklearn.metrics import accuracy_score
      print(accuracy_score(y_test,y_pred))
```

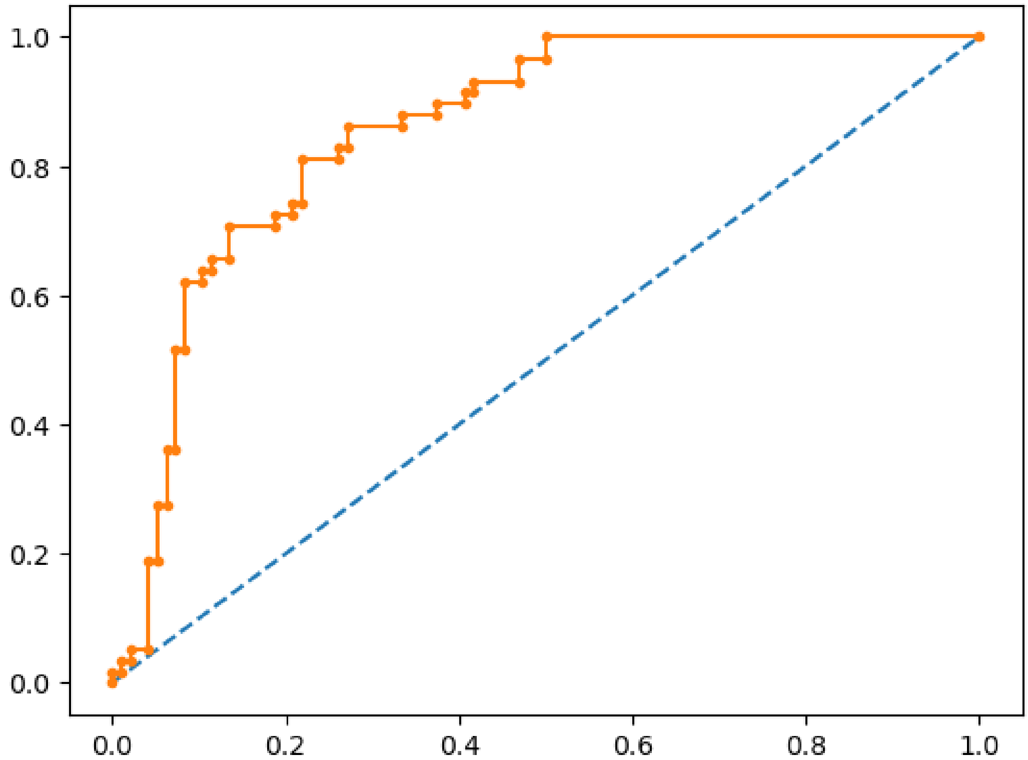
```
0.7792207792207793
```

```
[44]: #Preparing ROC Curve
      from sklearn.metrics import roc_curve
      from sklearn.metrics import roc_auc_score

      probs = lr.predict_proba(X_test)
      probs = probs[:, 1]
      auc = roc_auc_score(y_test, probs)
      print('AUC: %.3f' % auc)
      # calculate roc curve
      fpr, tpr, thresholds = roc_curve(y_test, probs)
      plt.plot([0, 1], [0, 1], linestyle='--')
      # plot the roc curve for the model
      plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.858
```

[44]: [<matplotlib.lines.Line2D at 0x2b2e63d04c0>]



Lets go for another classifier: Decision tree Classifier

[45]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(max_depth=5)
dt.fit(X_train,y_train)
```

[45]: DecisionTreeClassifier(max_depth=5)

[46]:
```
print(dt.score(X_train,y_train))
print(dt.score(X_test,y_test))
```

```
0.8208469055374593
0.7597402597402597
```

[47]:
```
pred_y=dt.predict(X_test)
```

[48]:
```
print(accuracy_score(y_test,pred_y))
```

```
0.7597402597402597
```

Random Forest Classifier

[49]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=12)
rf.fit(X_train,y_train)
```

[49]:
```
RandomForestClassifier(n_estimators=12)
```

[50]:
```
print(rf.score(X_train,y_train))
print(rf.score(X_test,y_test))
```

```
0.995114006514658
0.7337662337662337
```

[51]:
```
rf_pred=rf.predict(X_test)
```

[52]:
```
print(accuracy_score(y_test,rf_pred))
```

```
0.7337662337662337
```

Support Vector Machine

[53]:
```
from sklearn.svm import SVC
sv=SVC(kernel='rbf',gamma='auto')
sv.fit(X_train,y_train)
```

[53]:
```
SVC(gamma='auto')
```

[54]:
```
print(sv.score(X_test,y_test))
```

```
0.6233766233766234
```

KNN Classifier

[55]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7,metric='minkowski',p = 2)
knn.fit(X_train,y_train)
```

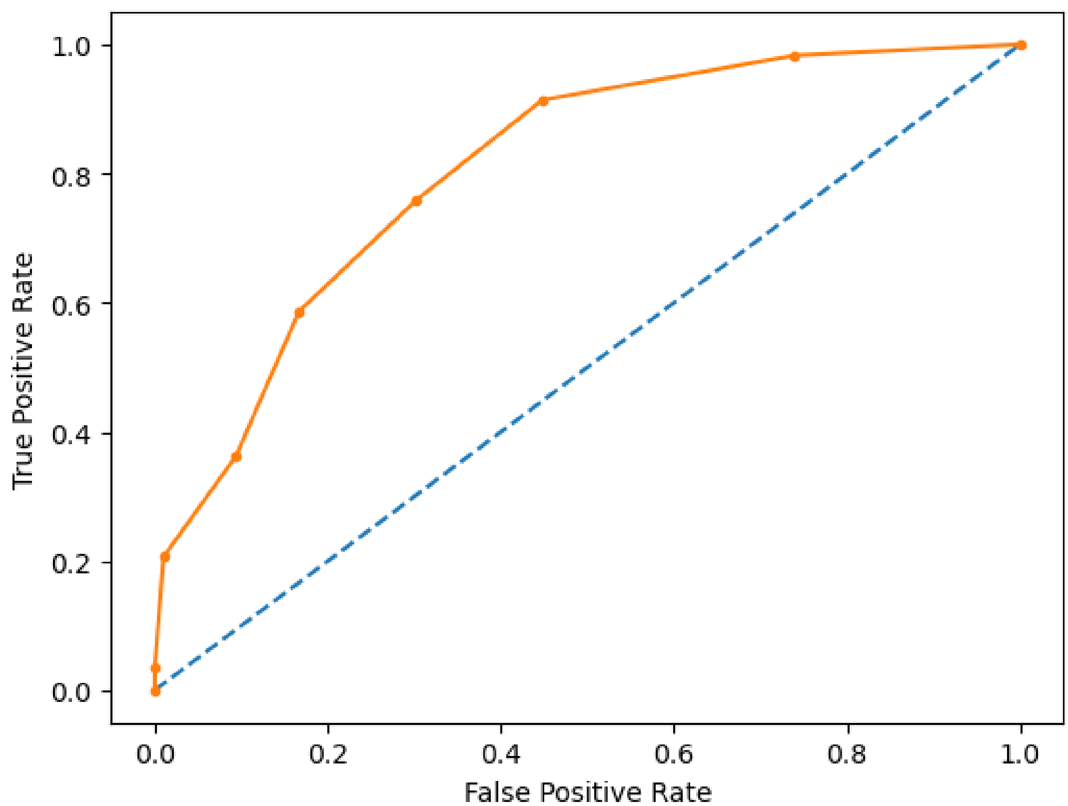
[55]:
```
KNeighborsClassifier(n_neighbors=7)
```

[56]:
```
probs = knn.predict_proba(X_test)
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".
 ↪format(tpr,fpr,thresholds))
```

```
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.807
True Positive Rate - [0.         0.03448276 0.20689655 0.36206897 0.5862069
0.75862069
 0.9137931  0.98275862 1.        ], False Positive Rate - [0.         0.
0.01041667 0.09375    0.16666667 0.30208333
 0.44791667 0.73958333 1.        ] Thresholds - [2.         1.
0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

[56]: Text(0, 0.5, 'True Positive Rate')



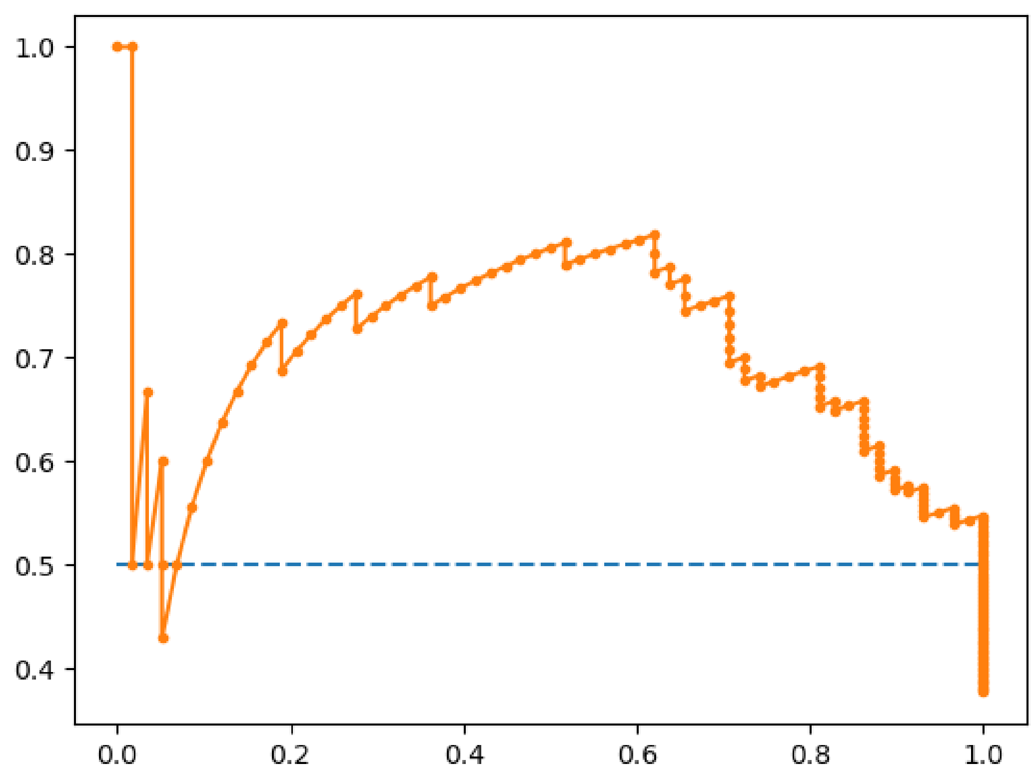
[57]:
```
#Precision Recall Curve for Logistic Regression
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
```

```
from sklearn.metrics import average_precision_score
# predict probabilities
probs = lr.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = lr.predict(X_test)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, probs)
# calculate F1 score
f1 = f1_score(y_test, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))

plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.653 auc=0.703 ap=0.711
```

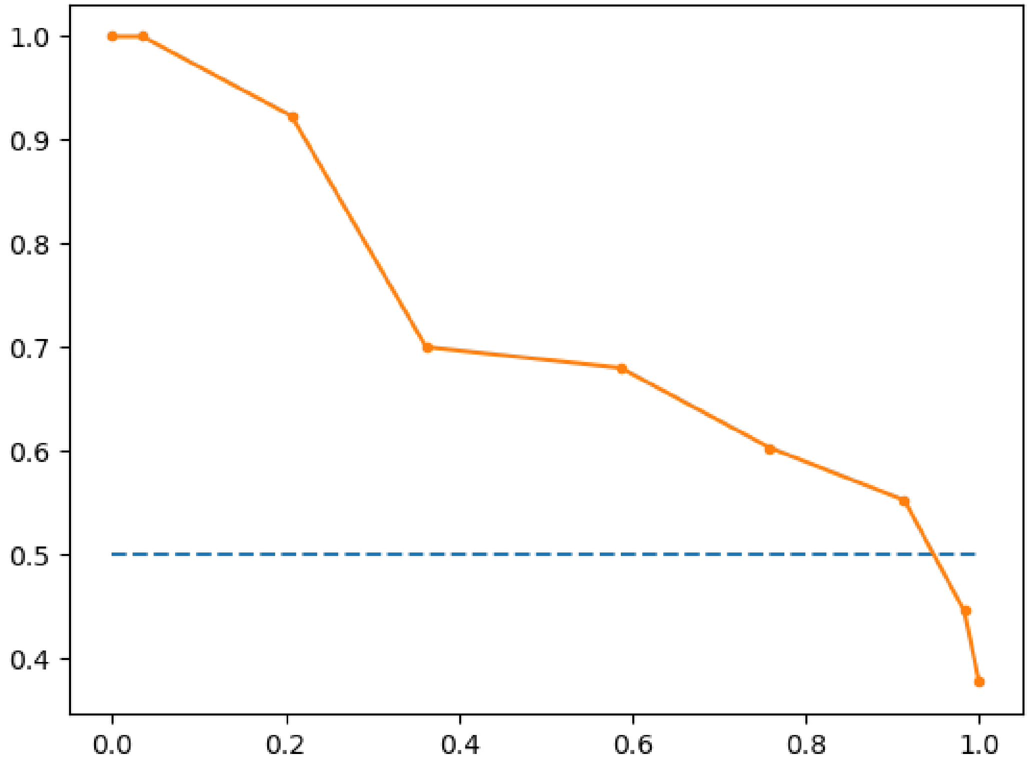
[57]: [<matplotlib.lines.Line2D at 0x2b2e6494250>]



[58]:
```
#Precision Recall Curve for KNN
from sklearn.metrics import auc
# predict probabilities
probs = knn.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = knn.predict(X_test)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, probs)
# calculate F1 score
f1 = f1_score(y_test, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.630 auc=0.723 ap=0.681
```

[58]:
```
[<matplotlib.lines.Line2D at 0x2b2e6506f70>]
```



[59]:

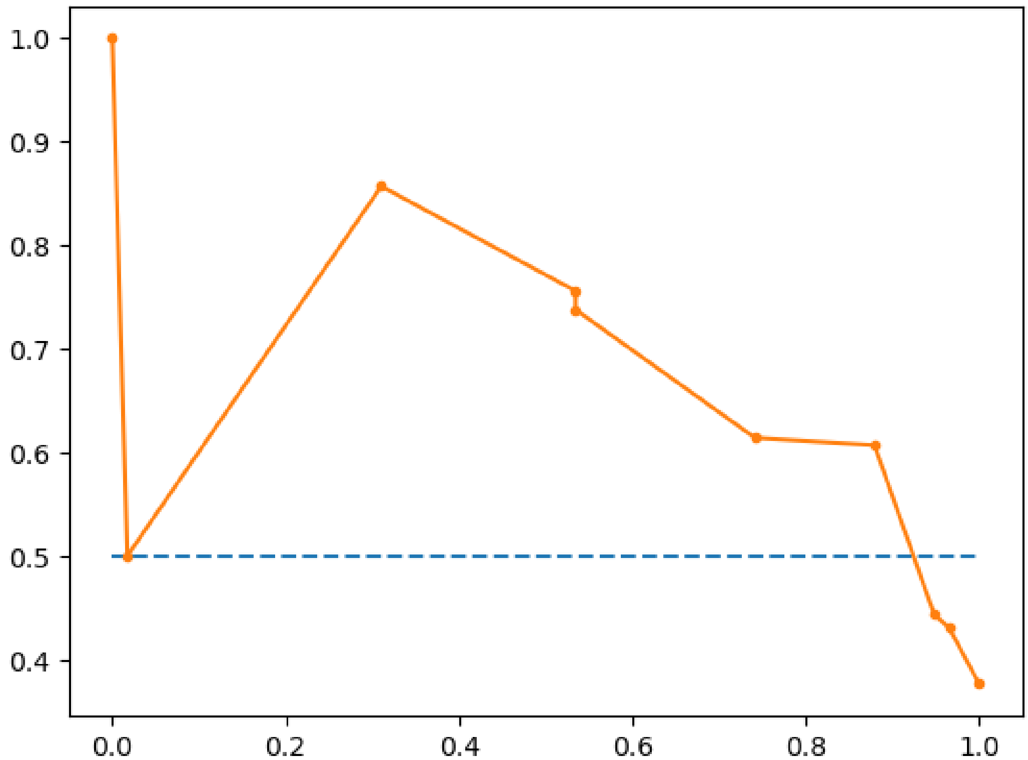
```
#Precision Recall Curve for Decission Tree Classifier

from sklearn.metrics import auc
# predict probabilities
probs = dt.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = dt.predict(X_test)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, probs)
# calculate F1 score
f1 = f1_score(y_test, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y_test, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))

plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

f1=0.626 auc=0.674 ap=0.691

[59]: [<matplotlib.lines.Line2D at 0x2b2e6ab5400>]



[60]:
```
#Precision Recall Curve for Random Forest

from sklearn.metrics import auc

# predict probabilities
probs = rf.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = rf.predict(X_test)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, probs)
# calculate F1 score
f1 = f1_score(y_test, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y_test, probs)
```

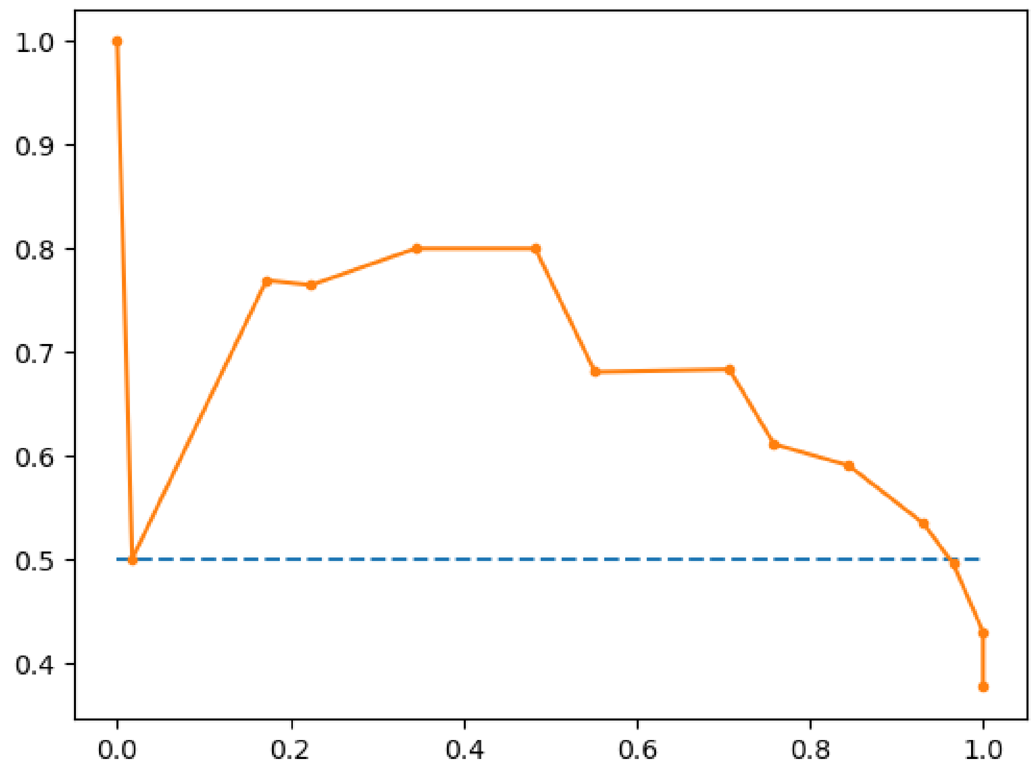
```python
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))

plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.610 auc=0.680 ap=0.688
```

[60]:

```
[<matplotlib.lines.Line2D at 0x2b2e6435640>]
```



[ ]:

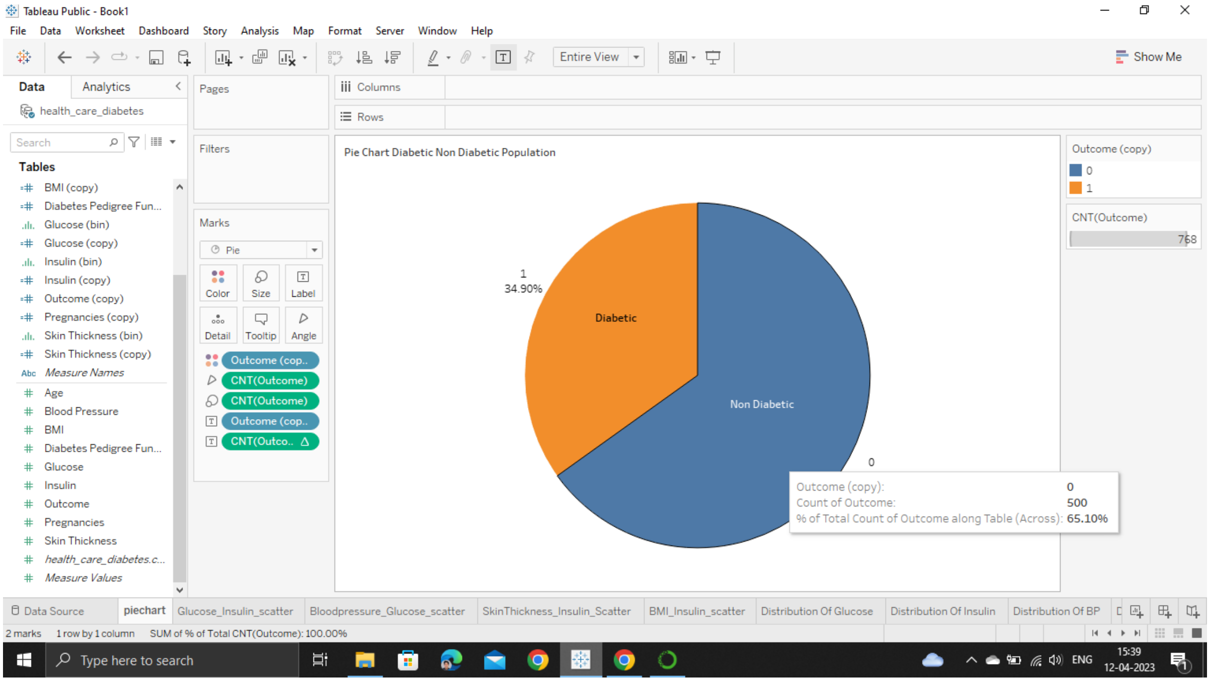


Tableau Public - Book1
Pie Chart Diabetic Non Diabetic Population
Diabetic
Non Diabetic
1
34.90%
0
Outcome (copy): 0
Count of Outcome: 500
% of Total Count of Outcome along Table (Across): 65.10%
Outcome (copy)
CNT(Outcome)
768

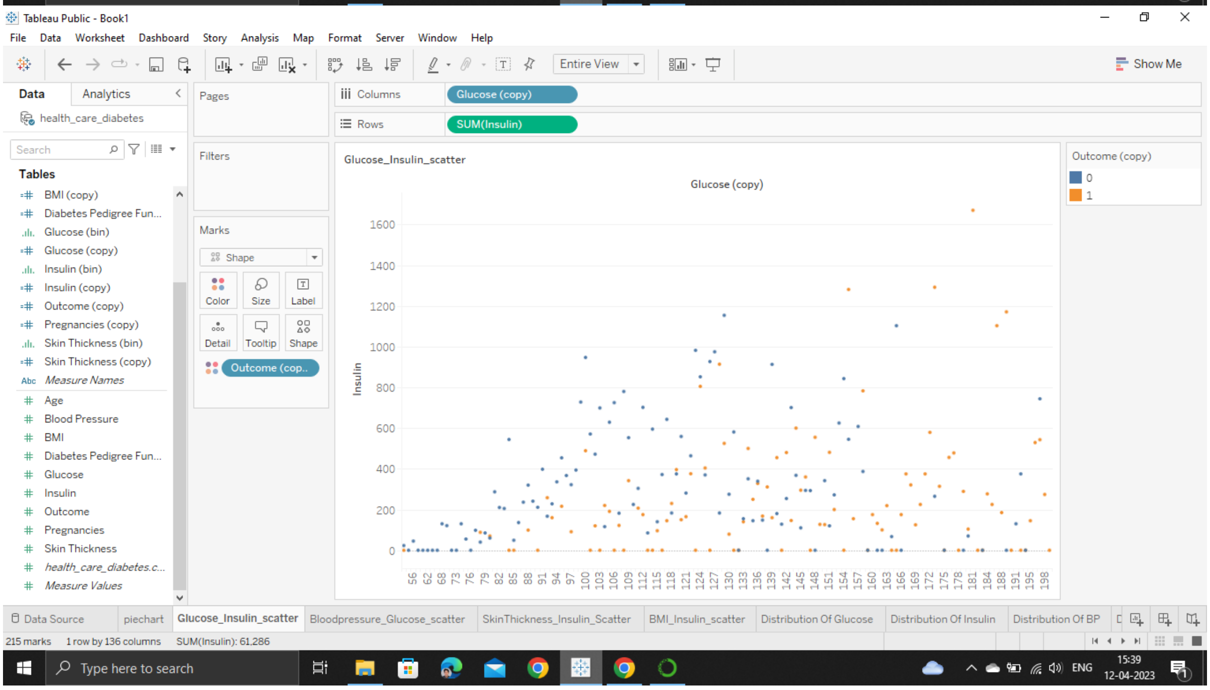
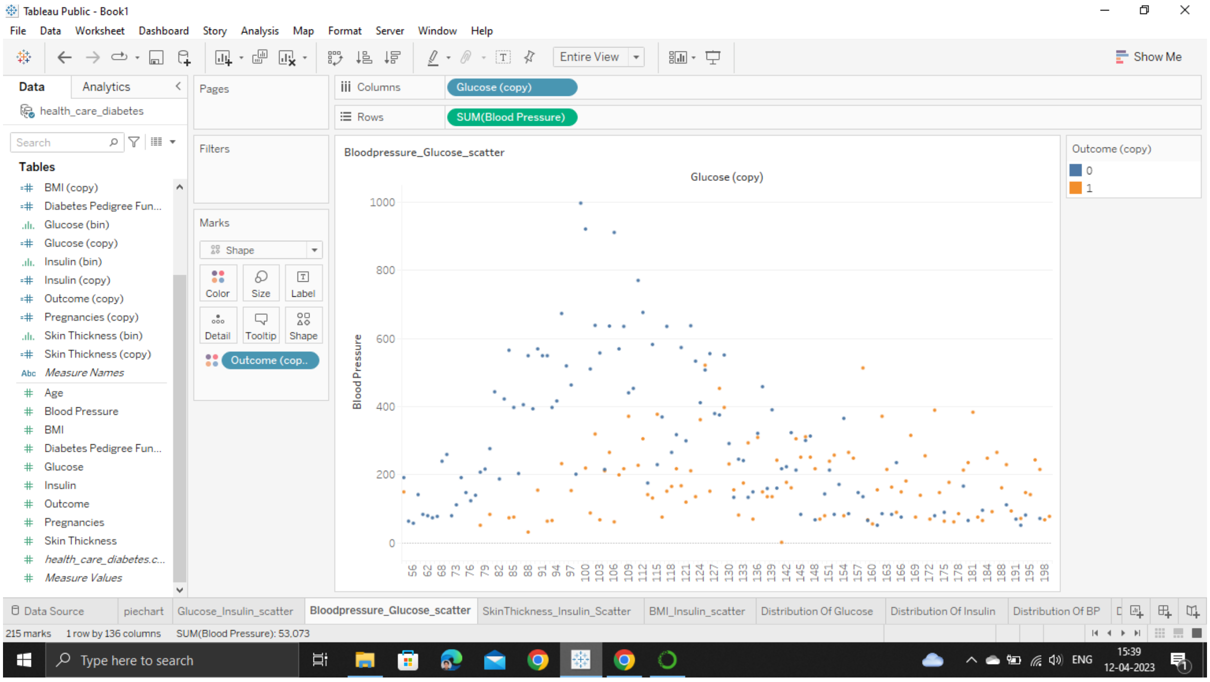
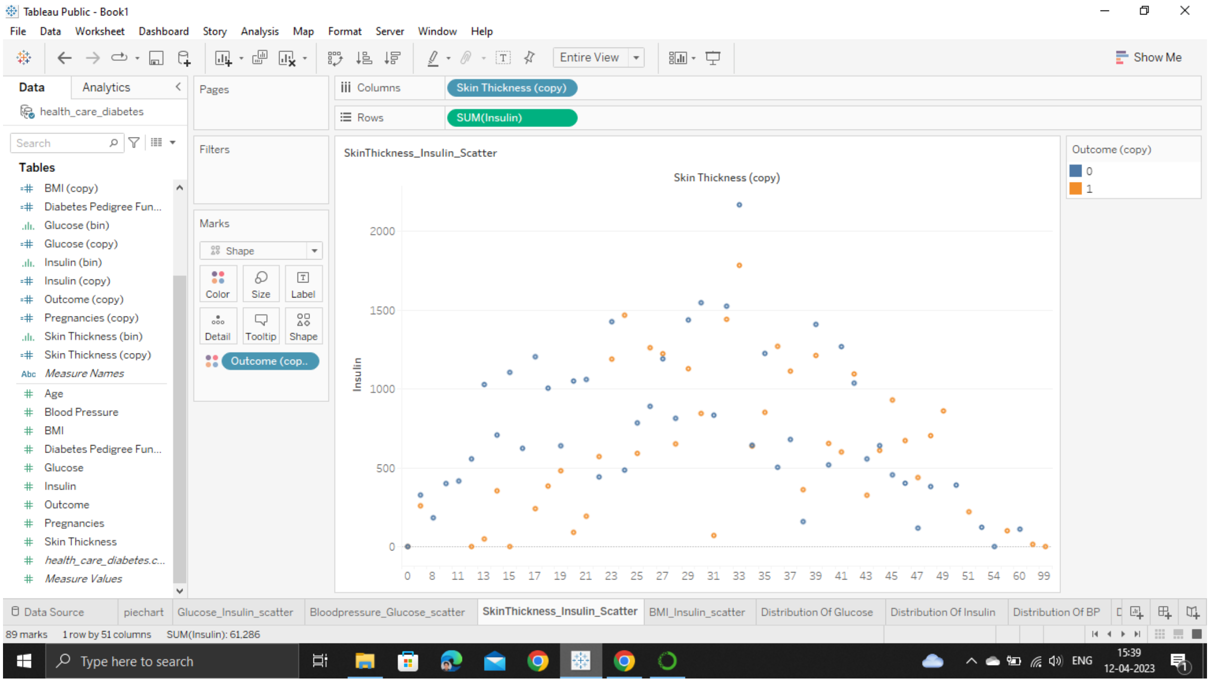


Tableau Public - Book1
Columns Glucose (copy)
Rows SUM(Insulin)
Glucose_Insulin_scatter
Glucose (copy)
Insulin
Outcome (copy)
215 marks
1 row by 136 columns
SUM(Insulin): 61,286


Bloodpressure_Glucose_scatter
Glucose (copy)
Blood Pressure
Outcome (copy)
0
1


SkinThickness_Insulin_Scatter
Skin Thickness (copy)
Insulin
Outcome (copy)
0
1

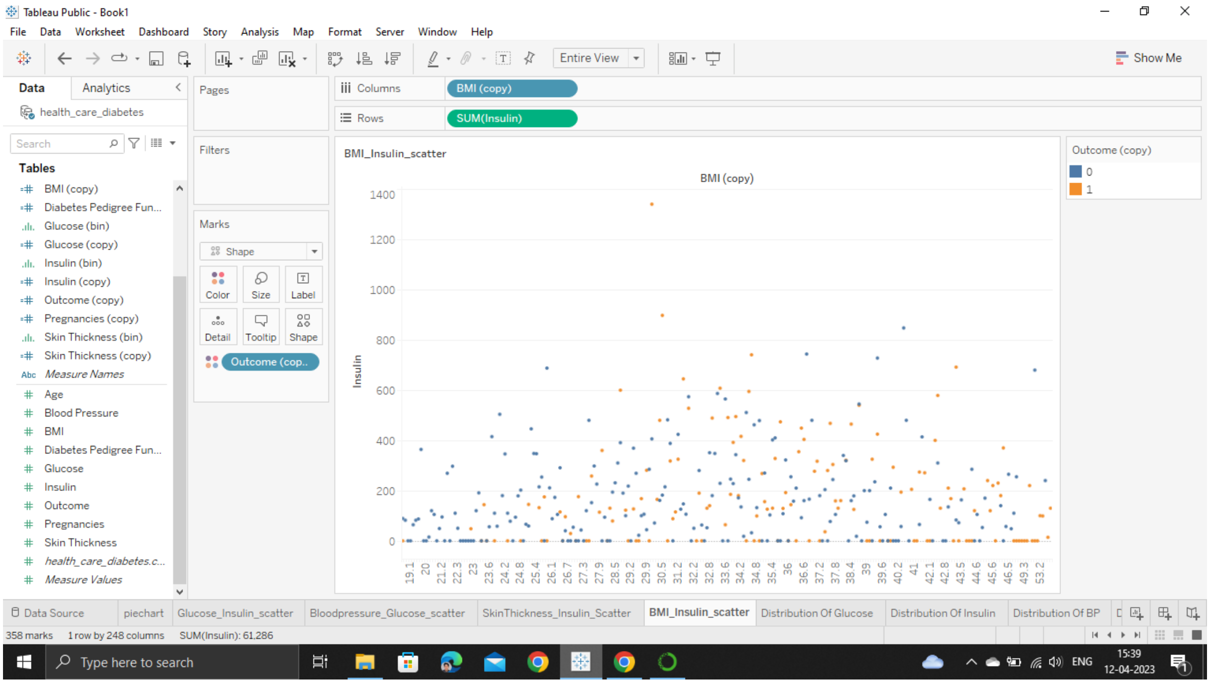


Tableau Public - Book1
Columns
BMI (copy)
Rows
SUM(Insulin)
BMI_Insulin_scatter
BMI (copy)
Insulin
Outcome (copy)
0
1
BMI_Insulin_scatter
358 marks
1 row by 248 columns
SUM(Insulin): 61,286

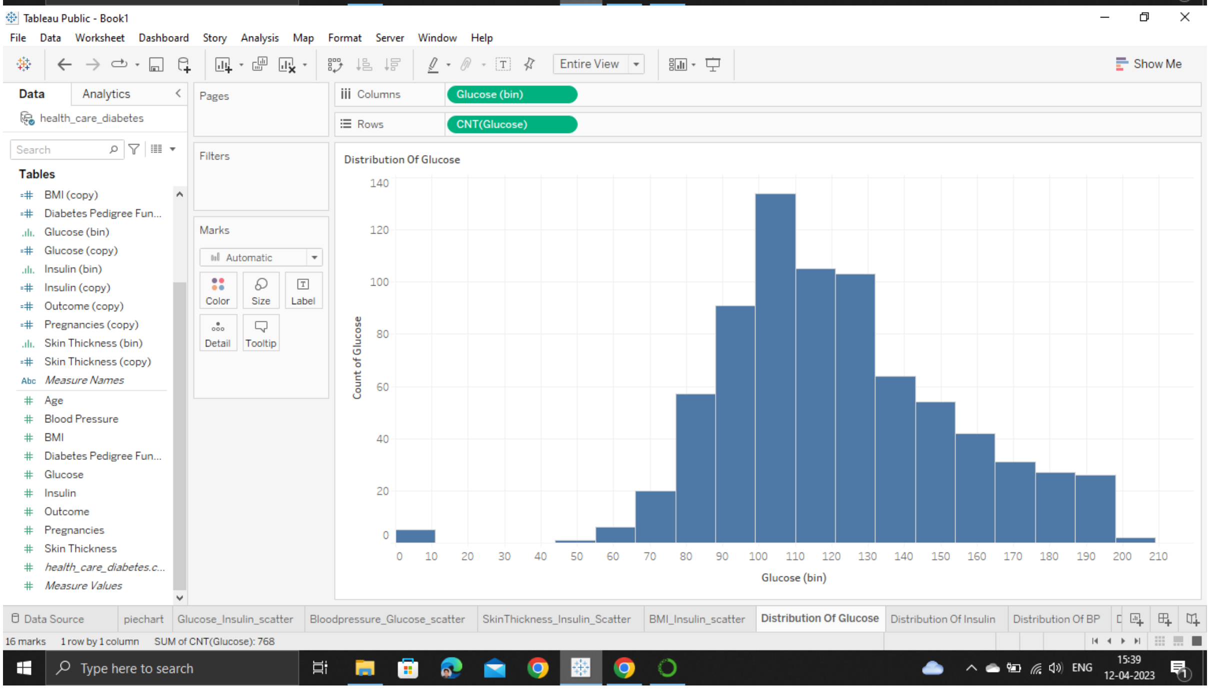



Tableau Public - Book1
Columns
Glucose (bin)
Rows
CNT(Glucose)
Distribution Of Glucose
Count of Glucose
Glucose (bin)
16 marks
1 row by 1 column
SUM of CNT(Glucose): 768

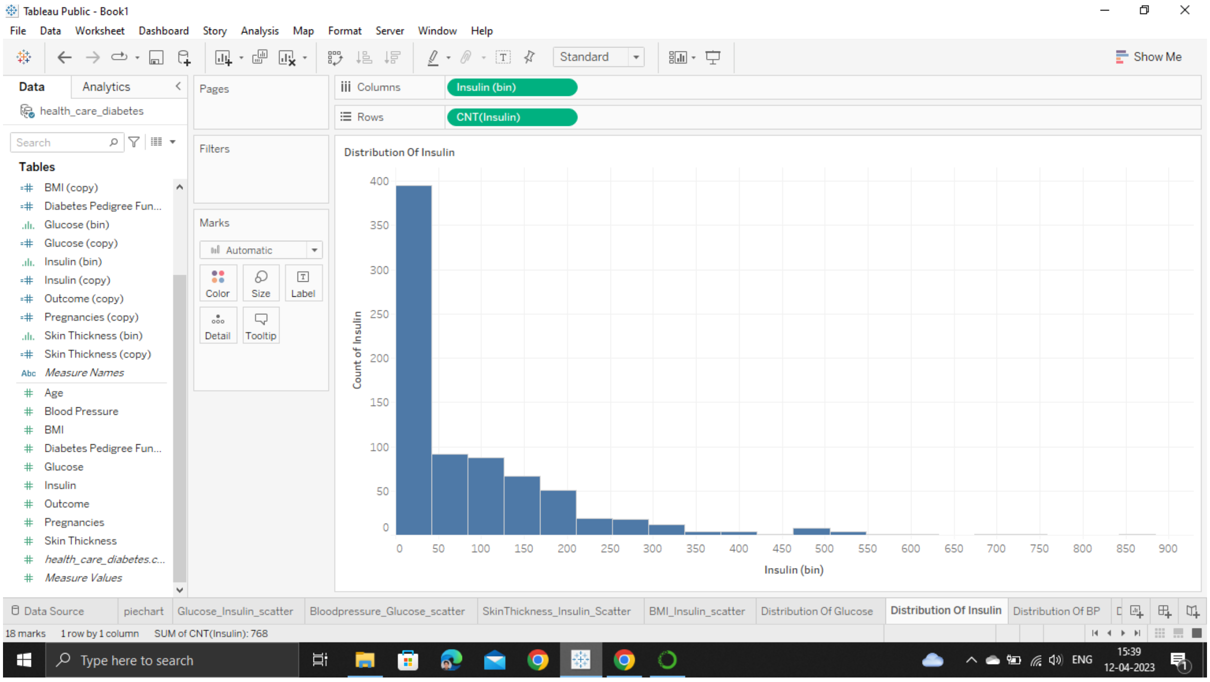


Tableau Public - Book1
Columns
Insulin (bin)
Rows
CNT(Insulin)
Distribution Of Insulin
Count of Insulin
Insulin (bin)
18 marks
1 row by 1 column
SUM of CNT(Insulin): 768

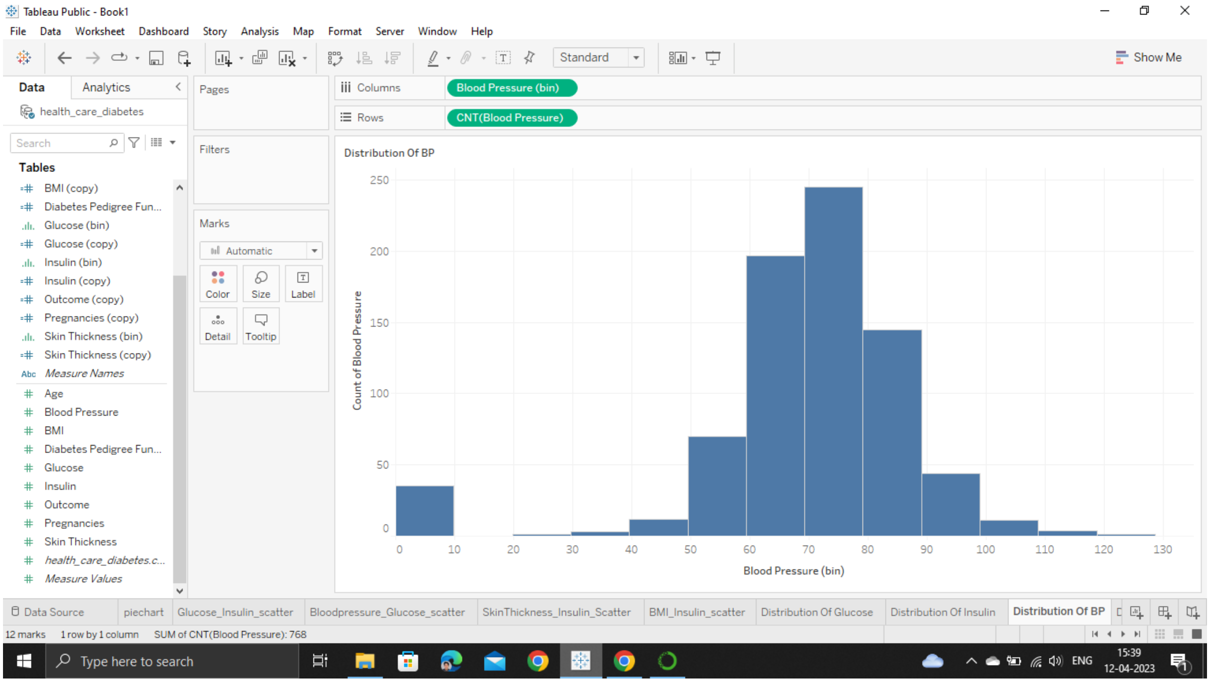


Tableau Public - Book1
Columns
Blood Pressure (bin)
Rows
CNT(Blood Pressure)
Distribution Of BP
Count of Blood Pressure
Blood Pressure (bin)
12 marks
1 row by 1 column
SUM of CNT(Blood Pressure): 768

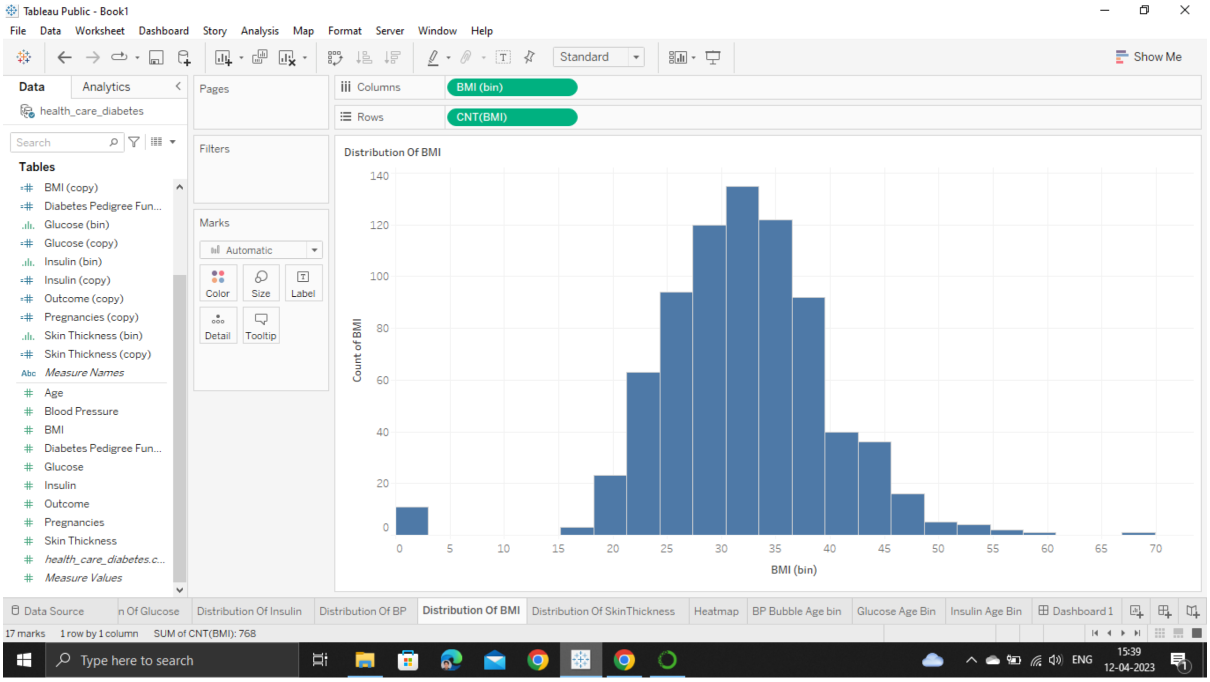


Tableau Public - Book1
Columns BMI (bin)
Rows CNT(BMI)
Distribution Of BMI
Count of BMI
BMI (bin)
Data Source
n Of Glucose
Distribution Of Insulin
Distribution Of BP
Distribution Of BMI
Distribution Of SkinThickness
Heatmap
BP Bubble Age bin
Glucose Age Bin
Insulin Age Bin
Dashboard 1
17 marks 1 row by 1 column SUM of CNT(BMI): 768

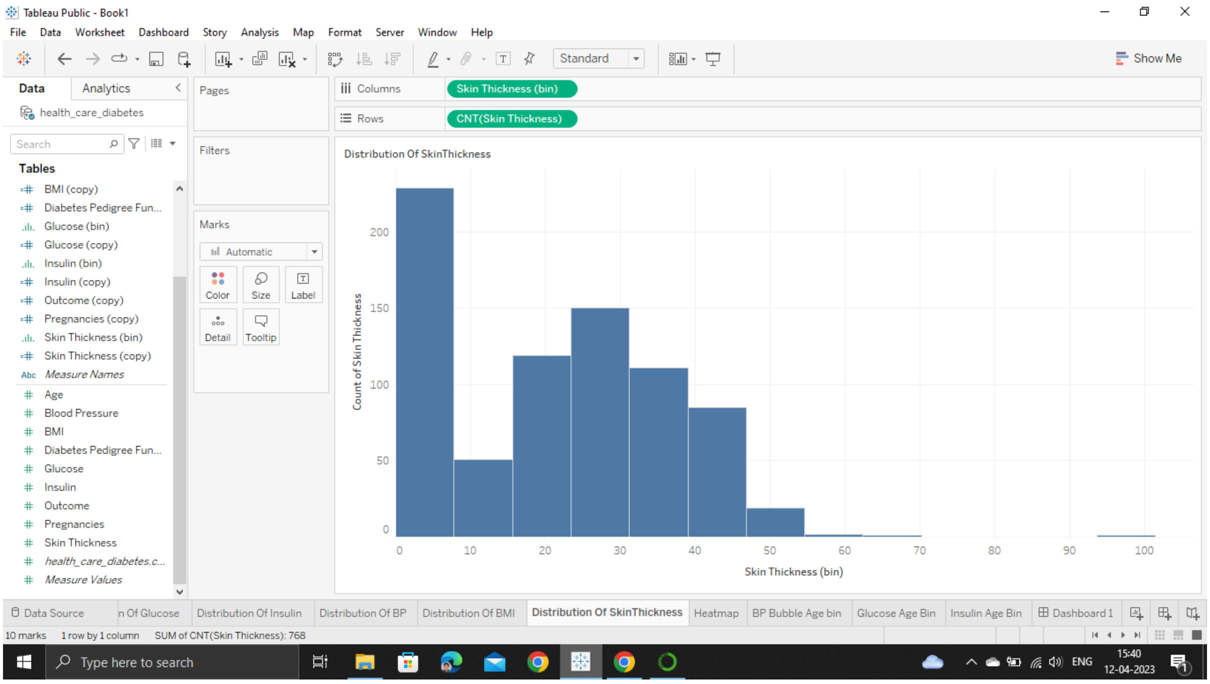


Tableau Public - Book1
Columns Skin Thickness (bin)
Rows CNT(Skin Thickness)
Distribution Of SkinThickness
Count of Skin Thickness
Skin Thickness (bin)
Data Source
n Of Glucose
Distribution Of Insulin
Distribution Of BP
Distribution Of BMI
Distribution Of SkinThickness
Heatmap
BP Bubble Age bin
Glucose Age Bin
Insulin Age Bin
Dashboard 1
10 marks 1 row by 1 column SUM of CNT(Skin Thickness): 768

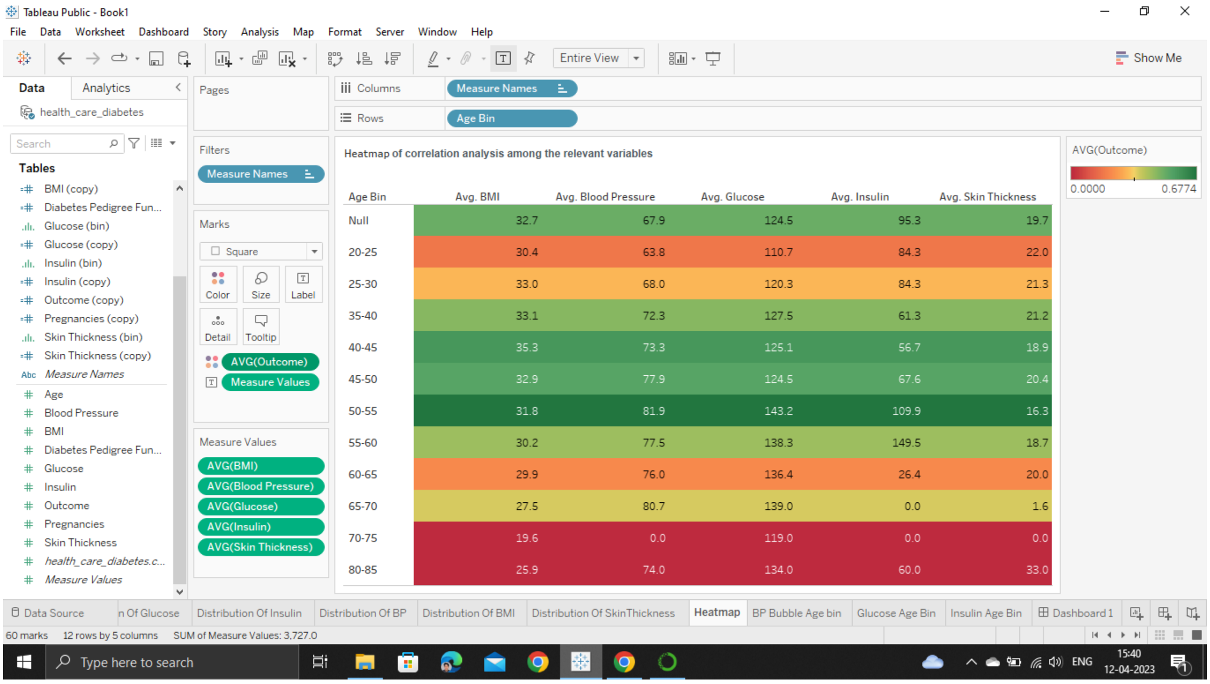


Tableau Public - Book1
Heatmap of correlation analysis among the relevant variables
Age Bin | Avg. BMI | Avg. Blood Pressure | Avg. Glucose | Avg. Insulin | Avg. Skin Thickness
Null | 32.7 | 67.9 | 124.5 | 95.3 | 19.7
20-25 | 30.4 | 63.8 | 110.7 | 84.3 | 22.0
25-30 | 33.0 | 68.0 | 120.3 | 84.3 | 21.3
35-40 | 33.1 | 72.3 | 127.5 | 61.3 | 21.2
40-45 | 35.3 | 73.3 | 125.1 | 56.7 | 18.9
45-50 | 32.9 | 77.9 | 124.5 | 67.6 | 20.4
50-55 | 31.8 | 81.9 | 143.2 | 109.9 | 16.3
55-60 | 30.2 | 77.5 | 138.3 | 149.5 | 18.7
60-65 | 29.9 | 76.0 | 136.4 | 26.4 | 20.0
65-70 | 27.5 | 80.7 | 139.0 | 0.0 | 1.6
70-75 | 19.6 | 0.0 | 119.0 | 0.0 | 0.0
80-85 | 25.9 | 74.0 | 134.0 | 60.0 | 33.0
AVG(Outcome)
0.0000
0.6774
60 marks
12 rows by 5 columns
SUM of Measure Values: 3,727.0

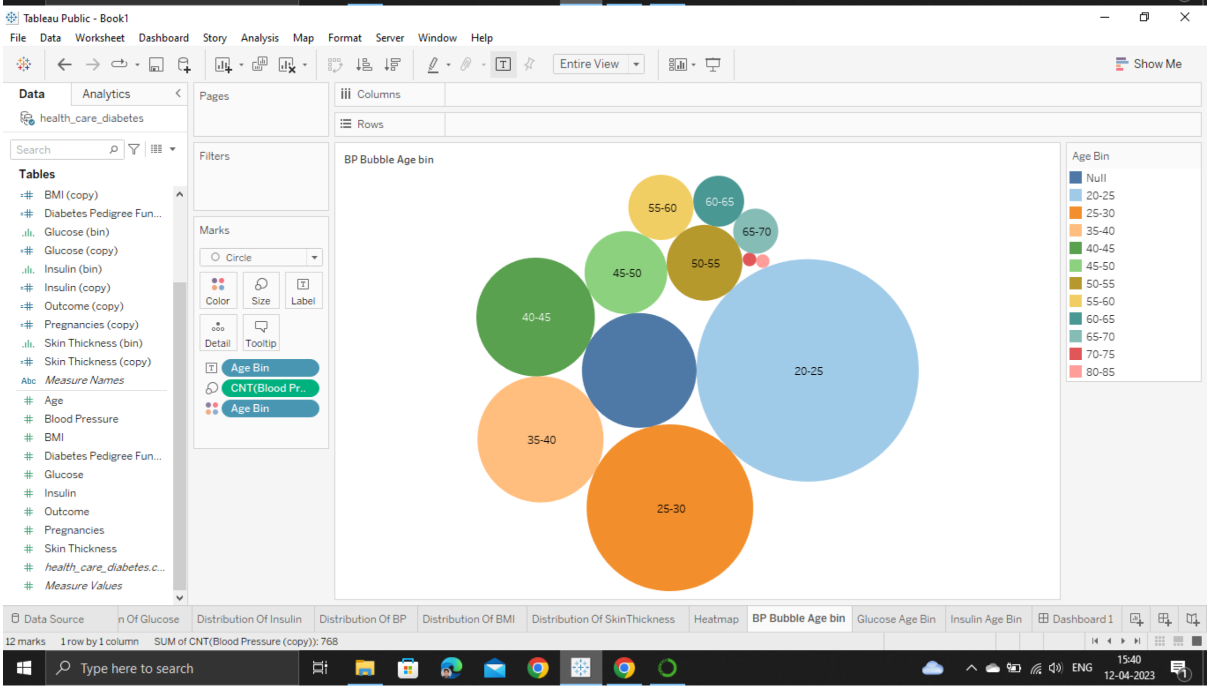


Tableau Public - Book1
BP Bubble Age bin
55-60
60-65
65-70
45-50
50-55
40-45
20-25
35-40
25-30
Age Bin
Null
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
65-70
70-75
80-85
12 marks
1 row by 1 column
SUM of CNT(Blood Pressure (copy)): 768

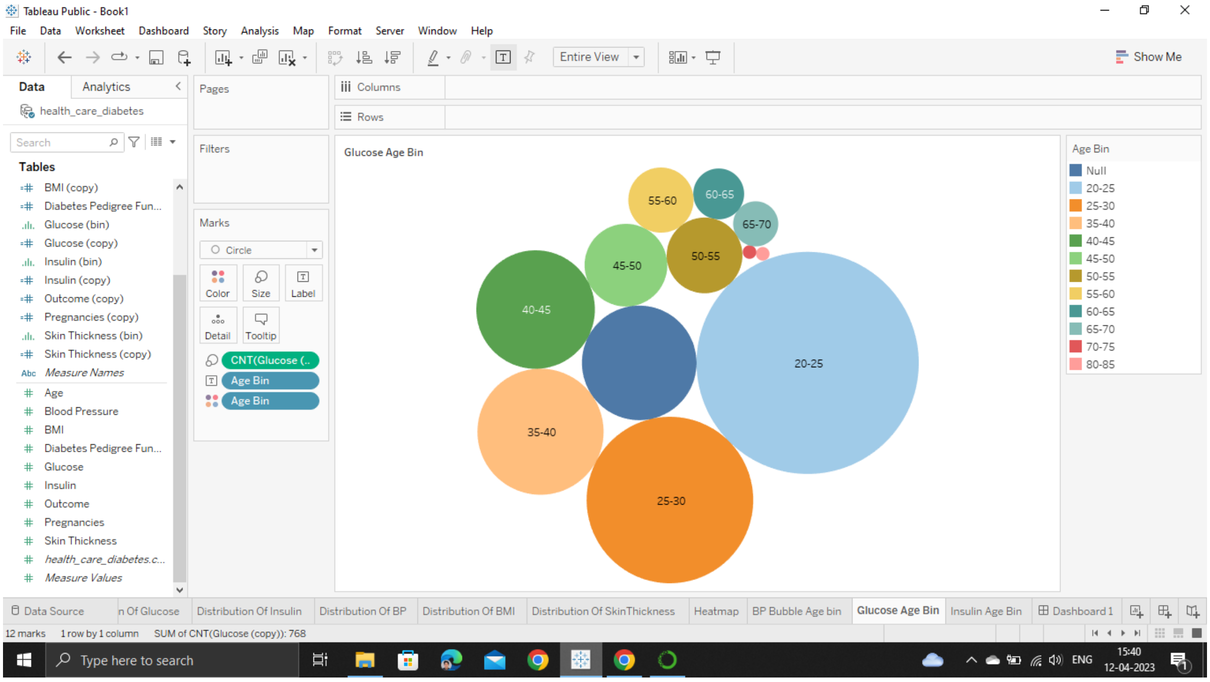


Tableau Public - Book1
Glucose Age Bin
55-60
60-65
65-70
45-50
50-55
40-45
20-25
35-40
25-30
Age Bin
Null
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
65-70
70-75
80-85
CNT(Glucose (..
Age Bin
Age Bin
12 marks
1 row by 1 column
SUM of CNT(Glucose (copy)): 768

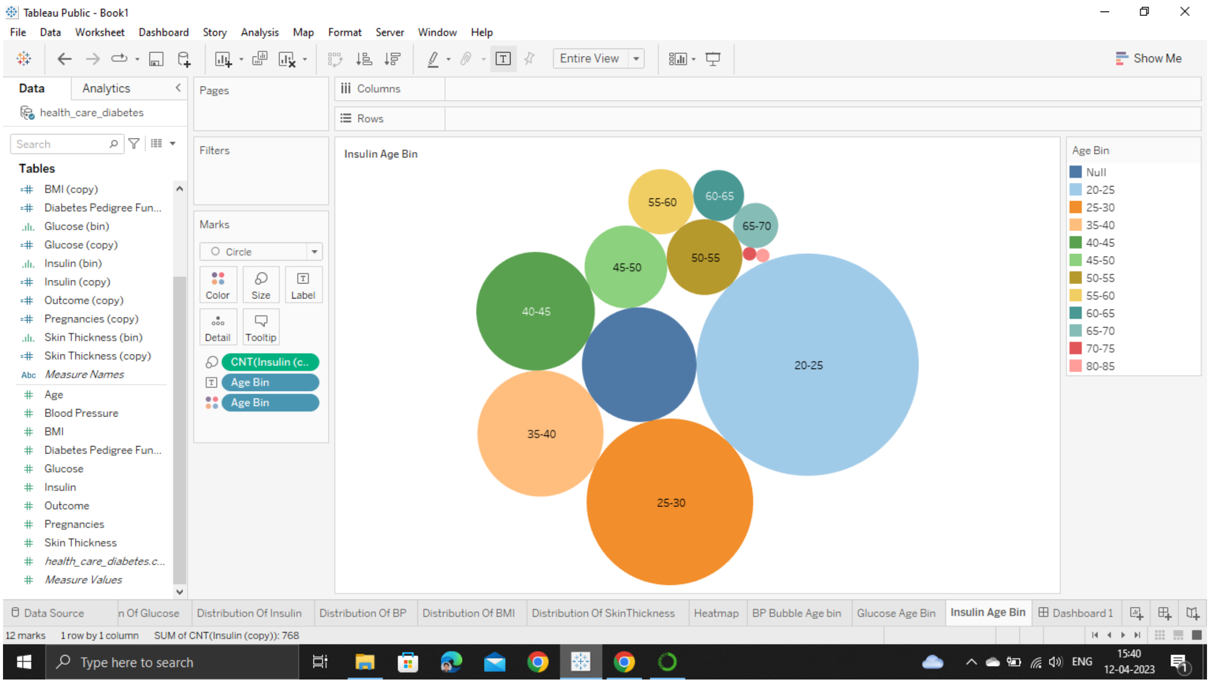


Tableau Public - Book1
Insulin Age Bin
55-60
60-65
65-70
45-50
50-55
40-45
20-25
35-40
25-30
Age Bin
Null
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
65-70
70-75
80-85
CNT(Insulin (c..
Age Bin
Age Bin
12 marks
1 row by 1 column
SUM of CNT(Insulin (copy)): 768

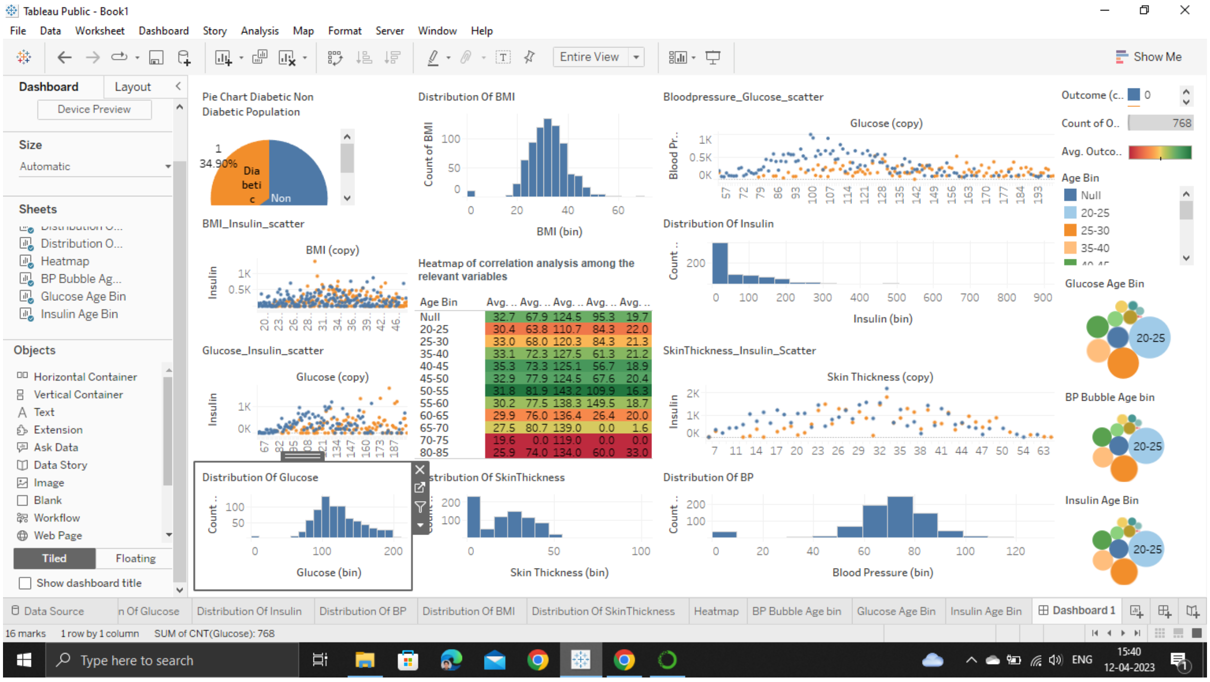


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Dashboard
Layout
Device Preview
Size
Automatic
Sheets
Distribution O...
Heatmap
BP Bubble Ag...
Glucose Age Bin
Insulin Age Bin
Objects
Horizontal Container
Vertical Container
Text
Extension
Ask Data
Data Story
Image
Blank
Workflow
Web Page
Tiled
Floating
Show dashboard title
Pie Chart Diabetic Non Diabetic Population
Distribution Of BMI
Bloodpressure_Glucose_scatter
BMI_Insulin_scatter
Heatmap of correlation analysis among the relevant variables
Distribution Of Insulin
Glucose_Insulin_scatter
SkinThickness_Insulin_Scatter
Distribution Of Glucose
Distribution Of SkinThickness
Distribution Of BP
Glucose Age Bin
BP Bubble Age bin
Insulin Age Bin
Age Bin
Null
20-25
25-30
35-40
Data Source
Distribution Of Insulin
Distribution Of BP
Distribution Of BMI
Distribution Of SkinThickness
Heatmap
BP Bubble Age bin
Glucose Age Bin
Insulin Age Bin
Dashboard 1
16 marks
1 row by 1 column
SUM of CNT(Glucose): 768
Type here to search
15:40
12-04-2023