Data Science Capstone

Problem Statement

The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

Build a model to accurately predict whether the patients in the dataset have diabetes or not.

In [3]:

```
import numpy as np
import pandas as pd


import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

%matplotlib inline
```

In [4]:

```
data = pd.read_csv('health care diabetes.csv')
```

In [5]:

```
data.head()
```

Out[5]:

	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

In [6]:

```
data.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
```

Out[6]:
```
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
data.corr()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outco
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000

In [8]:
```
data.describe().transpose()
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00

	count	mean	std	min	25%	50%	75%	max
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [9]:
```
data.Insulin.value_counts(normalize = True).to_frame().loc[0,:].values[0]*100
```

Out[9]: 48.69791666666667

In [10]:
```
data.Glucose.value_counts(normalize = True).to_frame().loc[0,:].values[0]*100
```

Out[10]: 0.6510416666666667

In [11]:
```
data.BloodPressure.value_counts(normalize = True).to_frame().loc[0,:].values[0]*100
```

Out[11]: 4.557291666666666

In [12]:
```
data.SkinThickness.value_counts(normalize = True).to_frame().loc[0,:].values[0]*100
```

Out[12]: 29.557291666666668

In [13]:
```
data.BMI.value_counts(normalize = True).to_frame().loc[0,:].values[0]*100
```

Out[13]: 1.4322916666666665

In [14]:
```
select_col = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']

for i in select_col:
    median=data[data[i]!=0][i].median()
    data[i]=data[i].apply(lambda x: median if x==0 else x)
```

In [15]:
```
data.head()
```

Out[15]:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

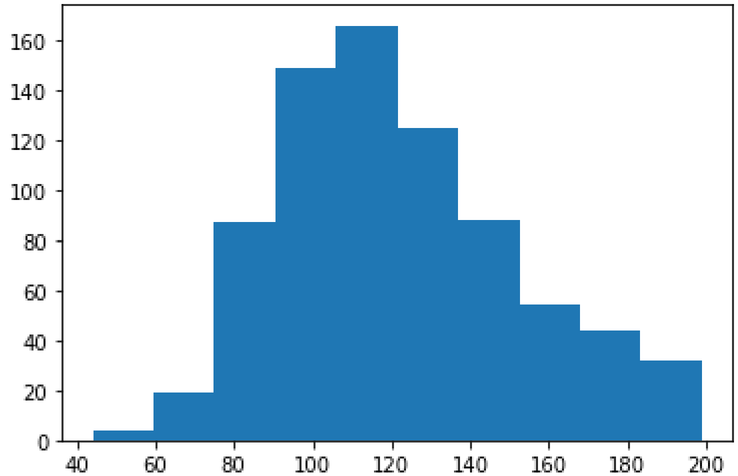
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

In [16]:

```
plt.hist(data['Glucose'])
```

Out[16]:

```
(array([  4.,  19.,  87., 149., 166., 125.,  88.,  54.,  44.,  32.]),
 array([ 44. ,  59.5,  75. ,  90.5, 106. , 121.5, 137. , 152.5, 168. ,
        183.5, 199. ]),
 <BarContainer object of 10 artists>)
```

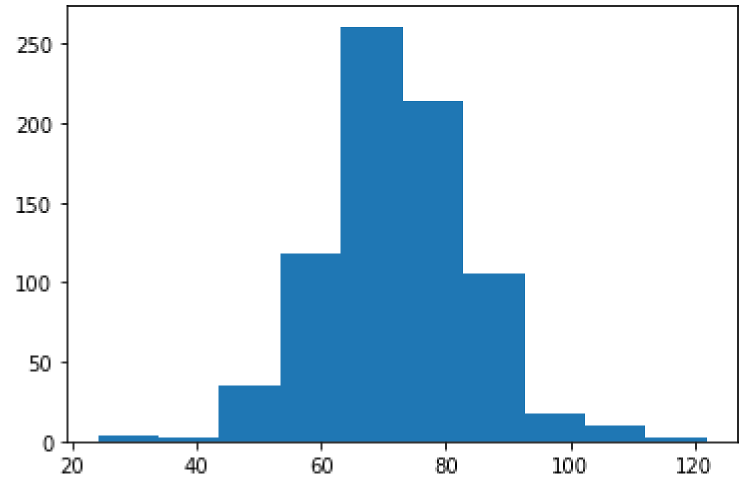


In [17]:

```
plt.hist(data['BloodPressure'])
```

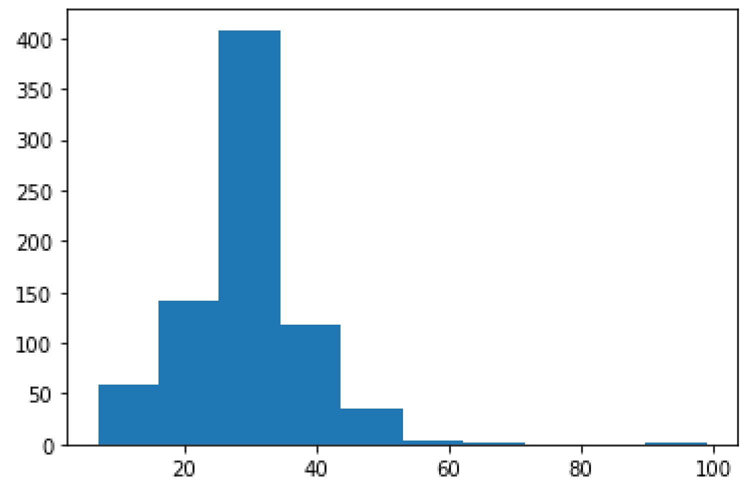
Out[17]:

```
(array([  3.,   2.,  35., 118., 261., 214., 105.,  18.,  10.,   2.]),
 array([ 24. ,  33.8,  43.6,  53.4,  63.2,  73. ,  82.8,  92.6, 102.4,
        112.2, 122. ]),
 <BarContainer object of 10 artists>)
```



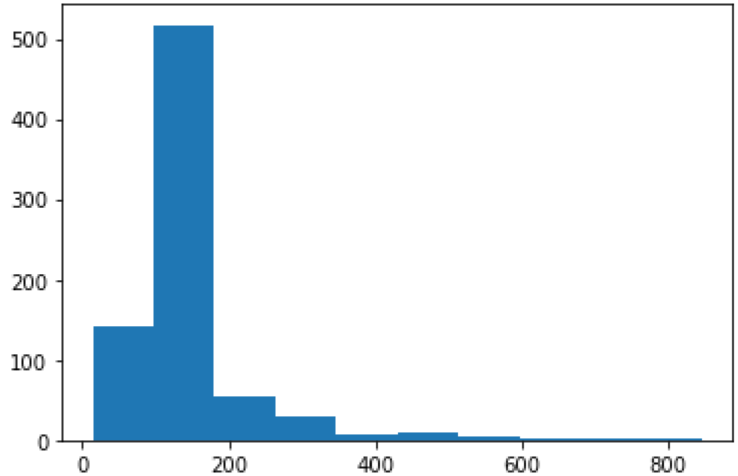
In [18]:
```
plt.hist(data['SkinThickness'])
```

Out[18]:
```
(array([ 59., 141., 408., 118.,  36.,   4.,   1.,   0.,   0.,   1.]),
 array([ 7. , 16.2, 25.4, 34.6, 43.8, 53. , 62.2, 71.4, 80.6, 89.8, 99. ]),
 <BarContainer object of 10 artists>)
```



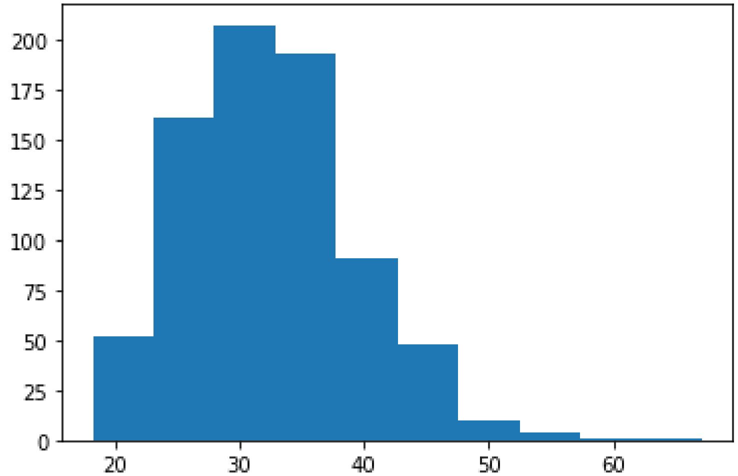
In [19]:
```
plt.hist(data['Insulin'])
```

Out[19]:
```
(array([142., 517.,  55.,  29.,   7.,  10.,   4.,   1.,   2.,   1.]),
 array([ 14. ,  97.2, 180.4, 263.6, 346.8, 430. , 513.2, 596.4, 679.6,
        762.8, 846. ]),
 <BarContainer object of 10 artists>)
```



In [20]:
```
plt.hist(data['BMI'])
```

Out[20]:
```
(array([ 52., 161., 207., 193.,  91.,  48.,  10.,   4.,   1.,   1.]),
 array([18.2 , 23.09, 27.98, 32.87, 37.76, 42.65, 47.54, 52.43, 57.32,
        62.21, 67.1 ]),
 <BarContainer object of 10 artists>)
```



In [21]:
```
data.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
```

In [22]:

```
data.dtypes.value_counts()
```

Out[22]:

```
float64    6
int64      3
dtype: int64
```

In [23]:

```
data.Outcome.value_counts(normalize=True)
```

Out[23]:

```
0    0.651042
1    0.348958
Name: Outcome, dtype: float64
```

In [24]:

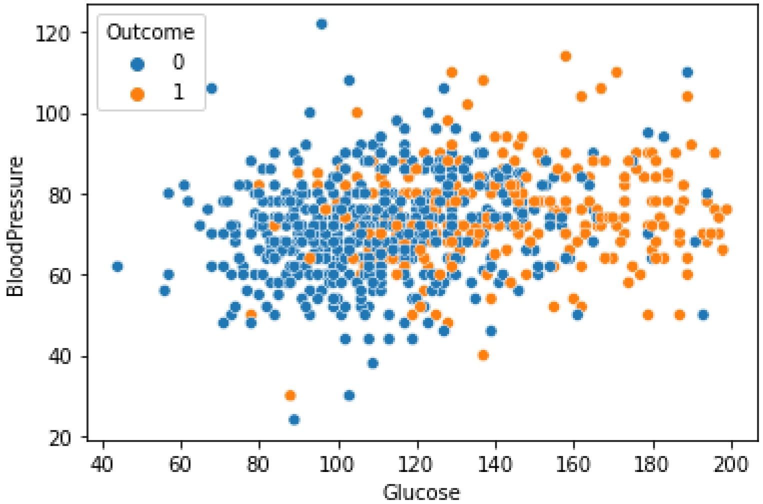
```
Positive = data[data['Outcome']==1]
```

In [25]:

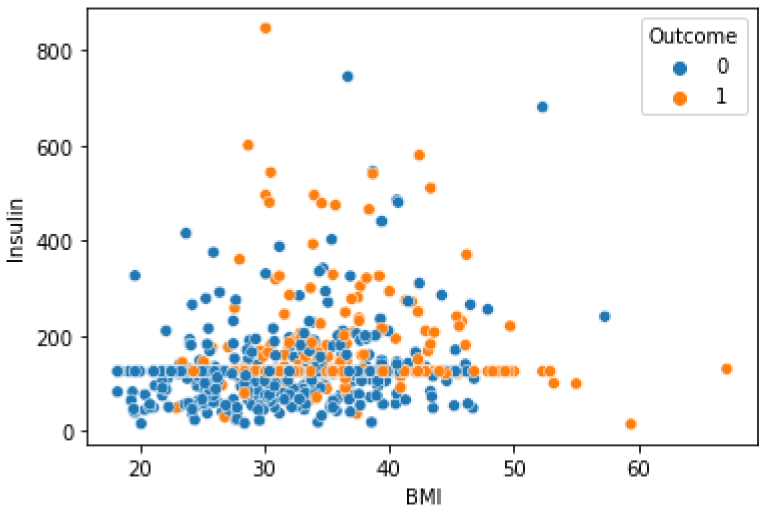
```
BloodPressure = Positive['BloodPressure']
Glucose = Positive['Glucose']
SkinThickness = Positive['SkinThickness']
Insulin = Positive['Insulin']
BMI = Positive['BMI']
```

In [26]:

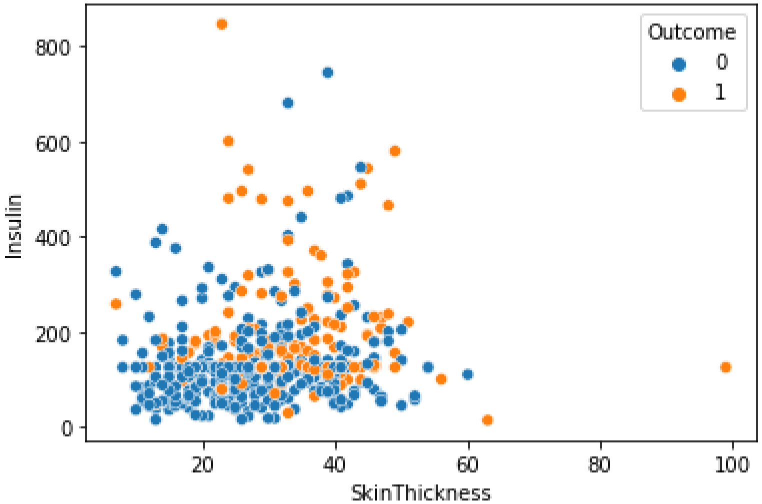
```
g =sns.scatterplot(x= "Glucose" ,y= "BloodPressure",
              hue="Outcome",
              data=data);
```



In [27]:
```
B =sns.scatterplot(x= "BMI" ,y= "Insulin",
              hue="Outcome",
              data=data);
```

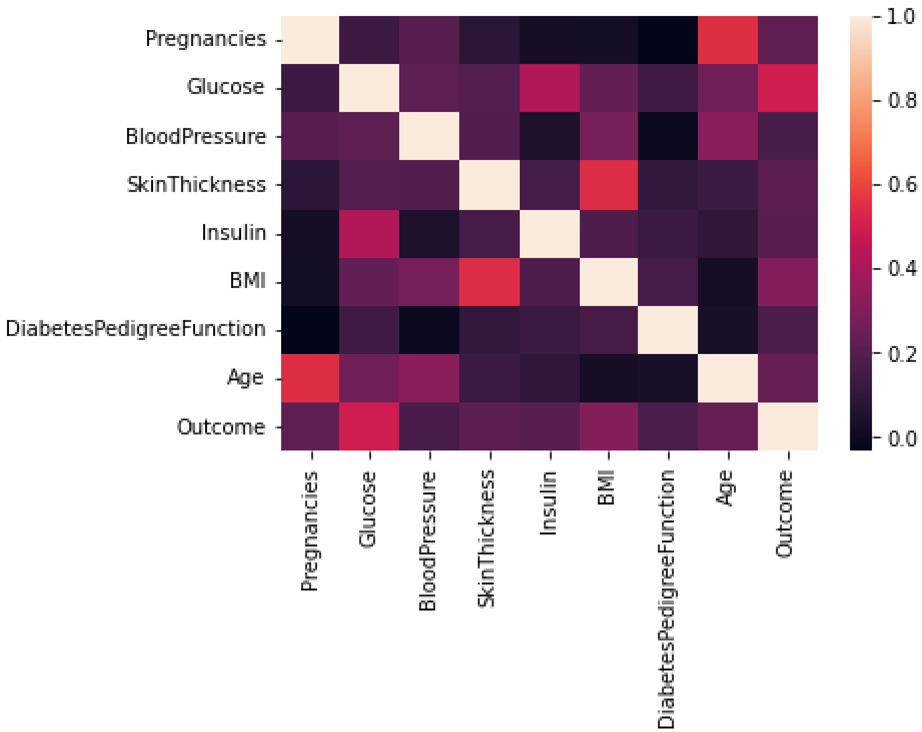


In [28]:
```
S =sns.scatterplot(x= "SkinThickness" ,y= "Insulin",
              hue="Outcome",
              data=data);
```



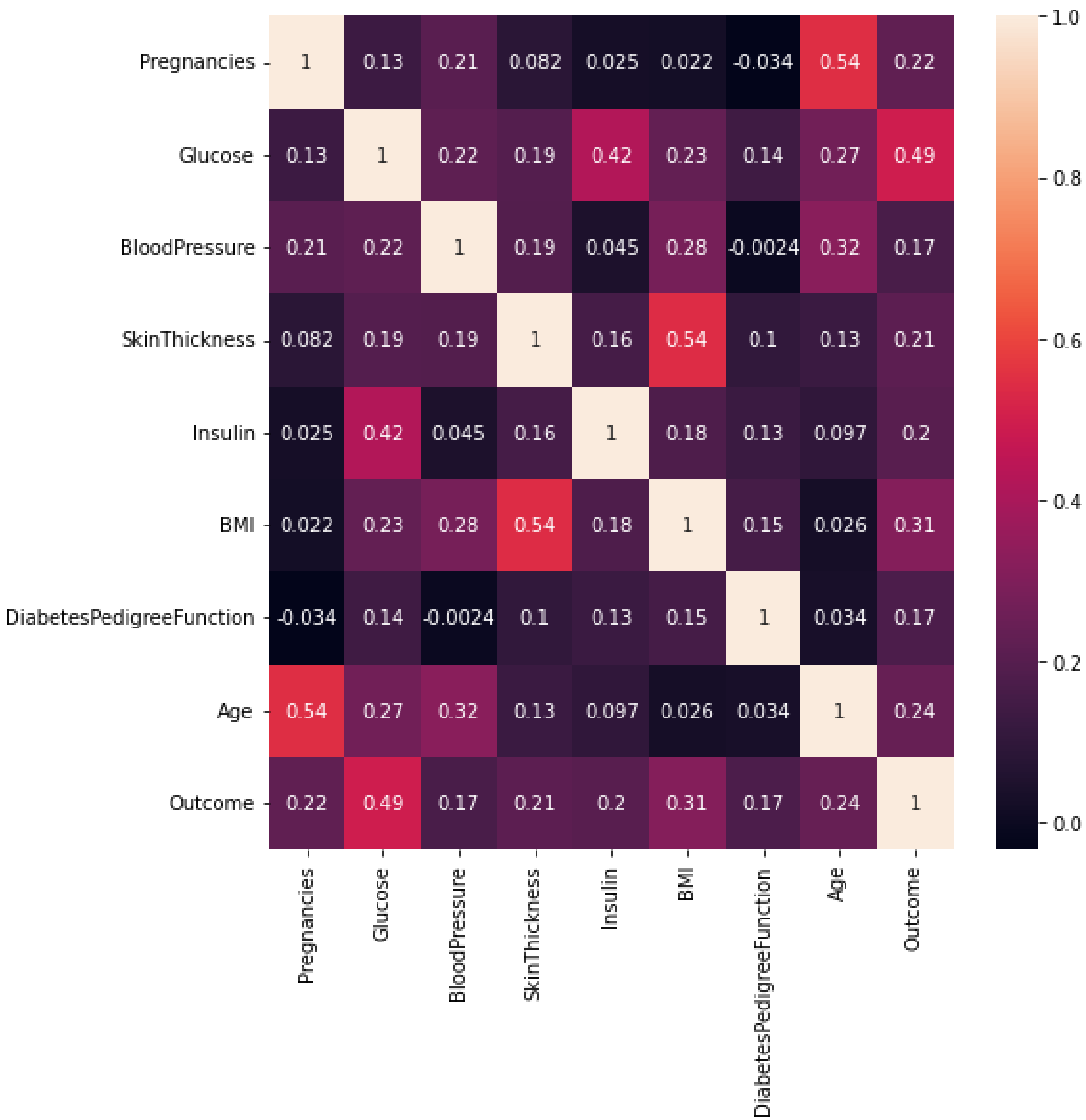
In [29]:
```
sns.heatmap(data.corr())
```

Out[29]: <AxesSubplot:>



In [30]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(data.corr(),annot=True)
```

Out[30]: <AxesSubplot:>



In [31]:
```
features = data.drop('Outcome',axis=1)
```

```
label = data['Outcome']
```

In [32]:
```
from sklearn.model_selection import train_test_split
```

In [34]:
```
x_train,x_test,y_train,y_test = train_test_split(features,label,test_size=.25,random_state=42, stratify=label)
```

In [35]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

In [36]:
```
model.fit(x_train,y_train)
```

Out[36]:
```
RandomForestClassifier()
```

In [38]:
```
model.score(x_train,y_train)
```

Out[38]:
```
1.0
```

In [40]:
```
model.score(x_test,y_test)
```

Out[40]:
```
0.7291666666666666
```

In [41]:
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       500
           1       0.92      0.89      0.90       268

    accuracy                           0.93       768
   macro avg       0.93      0.92      0.92       768
weighted avg       0.93      0.93      0.93       768
```

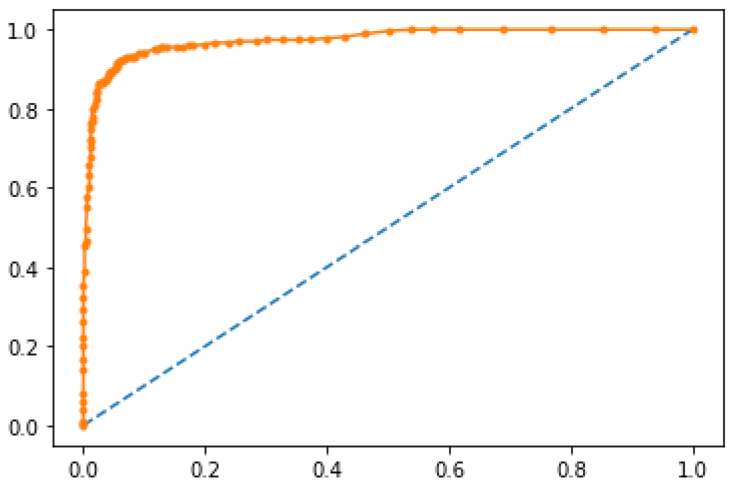
TO GO WITH F1 SCORE : It is imbalance class, Harmonic mean of precision and Recall.

In [55]:
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.974
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x197f5f4a100>]
```



In [56]:
```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(x_train,y_train)
```

Out[56]:
```
DecisionTreeClassifier(max_depth=5)
```

In [45]:
```
model3.score(x_train,y_train)
```
Out[45]:
```
0.8472222222222222
```
In [46]:
```
model3.score(x_test,y_test)
```
Out[46]:
```
0.75
```
In [47]:
```
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(x_train,y_train)
```
Out[47]:
```
RandomForestClassifier(n_estimators=11)
```
In [48]:
```
model4.score(x_train,y_train)
```
Out[48]:
```
0.9947916666666666
```
In [49]:
```
model4.score(x_test,y_test)
```
Out[49]:
```
0.7395833333333334
```
In [50]:
```
#Support Vector Classifier

from sklearn.svm import SVC
model5 = SVC(kernel='rbf',
             gamma='auto')
model5.fit(x_train,y_train)
```
Out[50]:
```
SVC(gamma='auto')
```
In [51]:
```
model5.score(x_test,y_test)
```
Out[51]:
```
0.6510416666666666
```

In [52]:

```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
model2.fit(x_train,y_train)
```

Out[52]:

```
KNeighborsClassifier(n_neighbors=7)
```

In [53]:

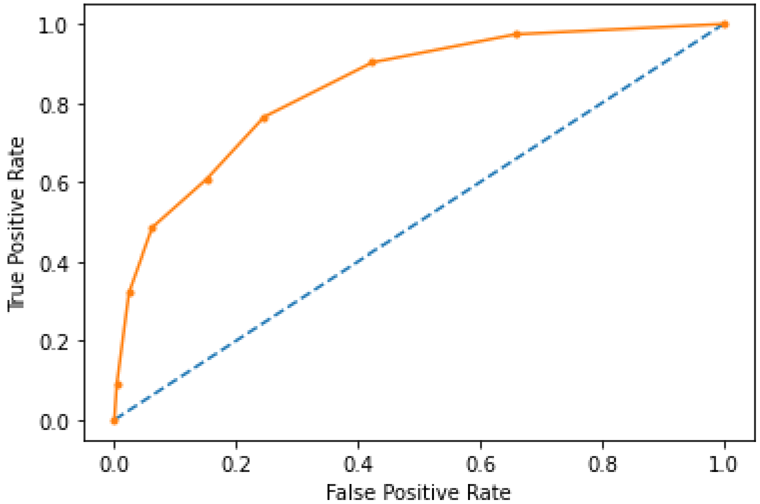
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".format(tpr,fpr,thresholds))
# plot no skill
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: 0.839
True Positive Rate - [0.         0.08955224 0.32089552 0.48507463 0.60820896 0.76492537
 0.90298507 0.9738806  1.        ], False Positive Rate - [0.    0.004 0.024 0.062 0.152 0.246 0.424 0.658 1.   ] Thresho
lds - [2.         1.         0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

Out[53]:

```
Text(0, 0.5, 'True Positive Rate')
```



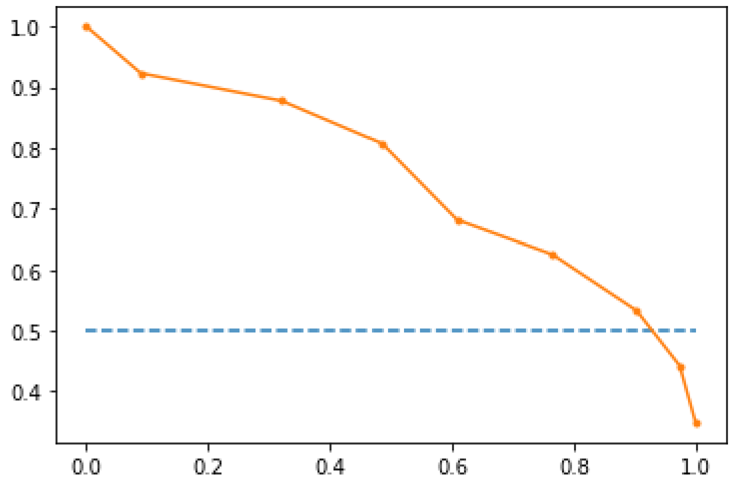
In [54]:

```
#Precision Recall Curve for KNN

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 = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

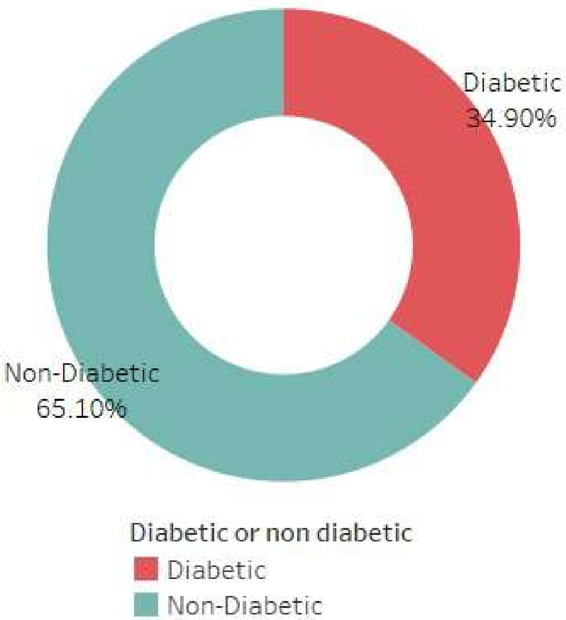
```
f1=0.643 auc=0.752 ap=0.714
```

Out[54]: [<matplotlib.lines.Line2D at 0x197f5ef48b0>]

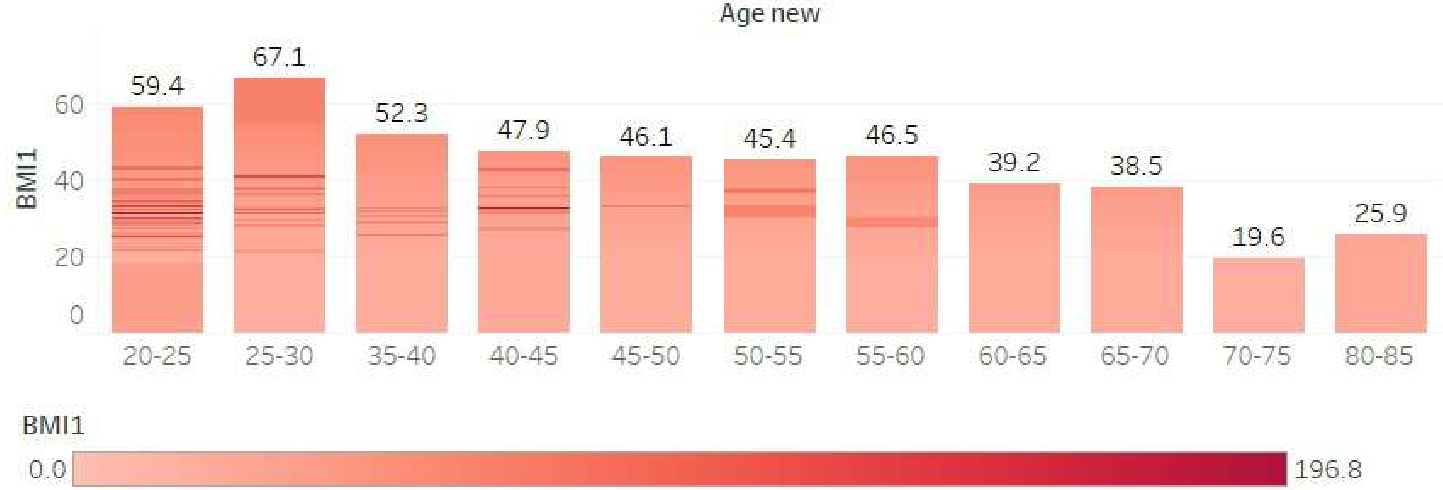


In [ ]:

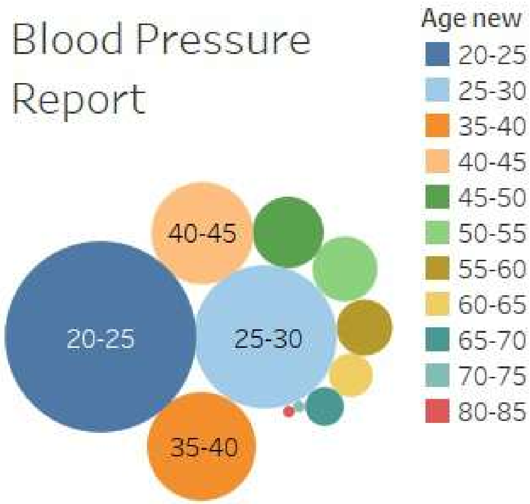
Diabetes / Non Diabetes



Body Mass Index Related to Age



Blood Pressure Report



Heat Map with Diff Variables

	Age new										
	20-25	25-30	35-40	40-45	45-50	50-55	55-60	60-65	65-70	70-75	80-85
Avg. BMI1	30.4	33.0	33.1	35.3	32.9	31.8	30.2	29.9	27.5	19.6	25.9
Avg. Blood Pressure	63.8	68.0	72.3	73.3	77.9	81.9	77.5	76.0	80.7	0.0	74.0
Avg. Glucose1	110.7	120.3	127.5	125.1	124.5	143.2	138.3	136.4	139.0	119.0	134.0
Avg. Insulin1	84.3	84.3	61.3	56.7	67.6	109.9	149.5	26.4	0.0	0.0	60.0
Avg. Skin Thickness	22.0	21.3	21.2	18.9	20.4	16.3	18.7	20.0	1.6	0.0	33.0

Measure Values

0.0 149.5

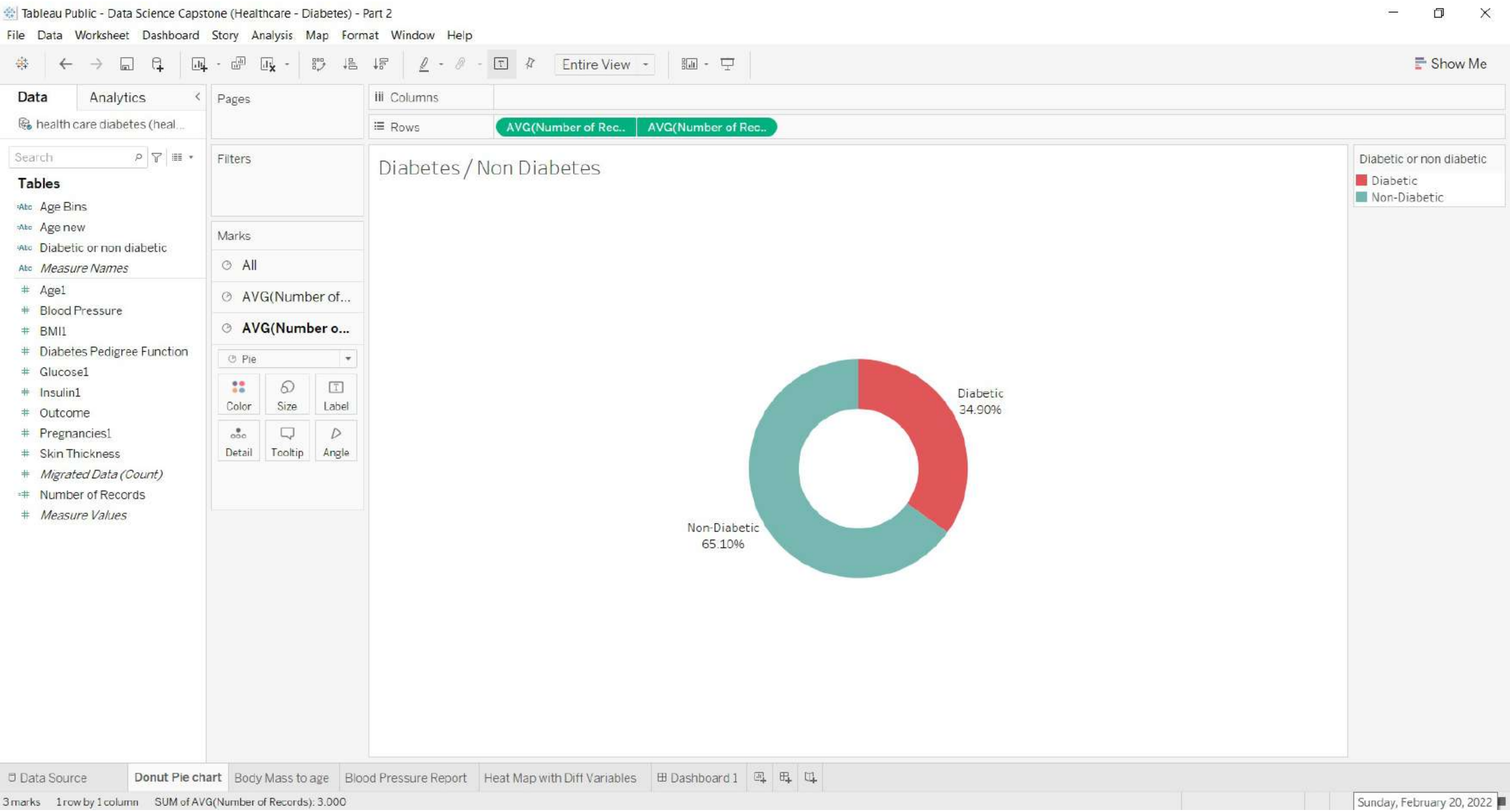


Tableau Public - Data Science Capstone (Healthcare - Diabetes) - Part 2
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Entire View
Show Me
Data
Analytics
health care diabetes (heal...
Pages
Columns
Rows
AVG(Number of Rec..
AVG(Number of Rec..
Filters
Marks
All
AVG(Number of...
AVG(Number o...
Pie
Color
Size
Label
Detail
Tooltip
Angle
Tables
Age Bins
Age new
Diabetic or non diabetic
Measure Names
Age1
Blood Pressure
BMI1
Diabetes Pedigree Function
Glucose1
Insulin1
Outcome
Pregnancies1
Skin Thickness
Migrated Data (Count)
Number of Records
Measure Values
Diabetes / Non Diabetes
Diabetic or non diabetic
Diabetic
Non-Diabetic
Diabetic
34.90%
Non-Diabetic
65.10%
Data Source
Donut Pie chart
Body Mass to age
Blood Pressure Report
Heat Map with Diff Variables
Dashboard 1
3 marks
1 row by 1 column
SUM of AVG(Number of Records): 3.000
Sunday, February 20, 2022

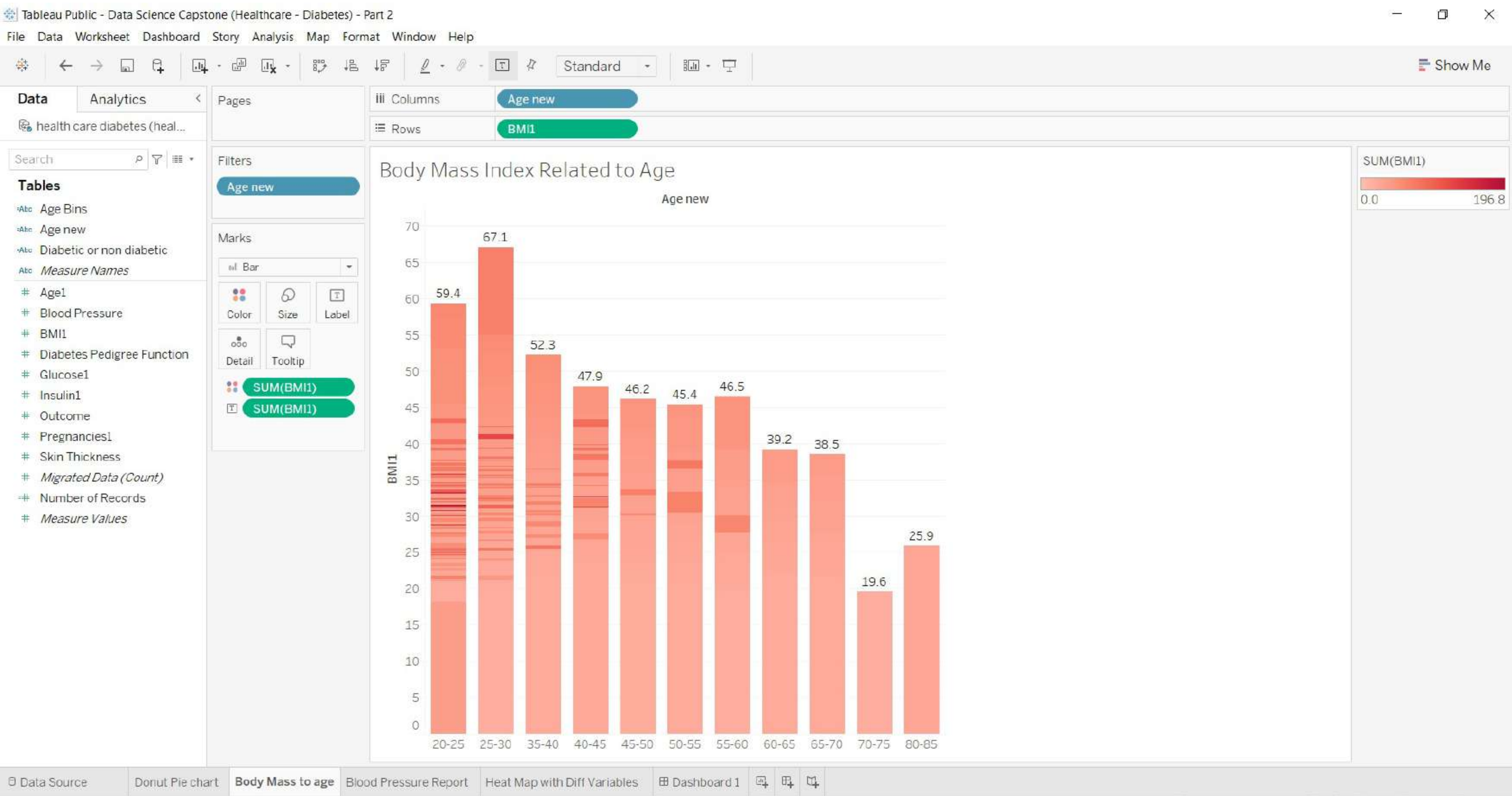


Tableau Public - Data Science Capstone (Healthcare - Diabetes) - Part 2
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Standard
Show Me
Data
Analytics
health care diabetes (heal...
Pages
Columns
Age new
Rows
BMI1
Search
Tables
Age Bins
Age new
Diabetic or non diabetic
Measure Names
Age1
Blood Pressure
BMI1
Diabetes Pedigree Function
Glucose1
Insulin1
Outcome
Pregnancies1
Skin Thickness
Migrated Data (Count)
Number of Records
Measure Values
Filters
Age new
Marks
Bar
Color
Size
Label
Detail
Tooltip
SUM(BMI1)
SUM(BMI1)
Body Mass Index Related to Age
Age new
SUM(BMI1)
0.0
196.8
BMI1
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
59.4
67.1
52.3
47.9
46.2
45.4
46.5
39.2
38.5
19.6
25.9
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
65-70
70-75
80-85
Data Source
Donut Pie chart
Body Mass to age
Blood Pressure Report
Heat Map with Diff Variables
Dashboard 1

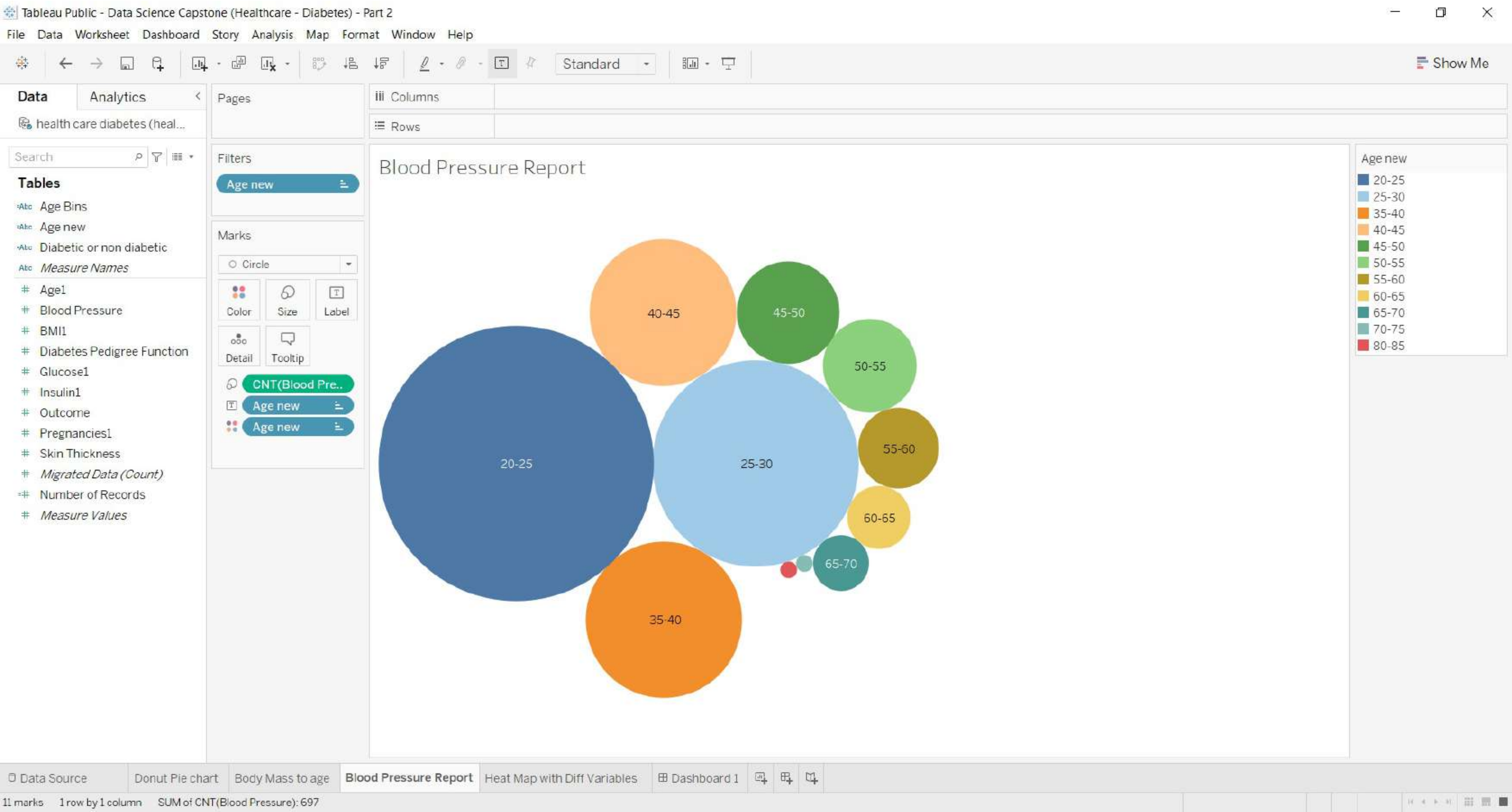


Tableau Public - Data Science Capstone (Healthcare - Diabetes) - Part 2
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Standard
Show Me
Data
Analytics
health care diabetes (heal...
Search
Tables
Age Bins
Age new
Diabetic or non diabetic
Measure Names
Age1
Blood Pressure
BMI1
Diabetes Pedigree Function
Glucose1
Insulin1
Outcome
Pregnancies1
Skin Thickness
Migrated Data (Count)
Number of Records
Measure Values
Pages
Filters
Age new
Marks
Circle
Color
Size
Label
Detail
Tooltip
CNT(Blood Pre..
Age new
Age new
Columns
Rows
Blood Pressure Report
40-45
45-50
50-55
20-25
25-30
55-60
60-65
65-70
35-40
Age new
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
65-70
70-75
80-85
Data Source
Donut Pie chart
Body Mass to age
Blood Pressure Report
Heat Map with Diff Variables
Dashboard 1
11 marks
1 row by 1 column
SUM of CNT(Blood Pressure): 697

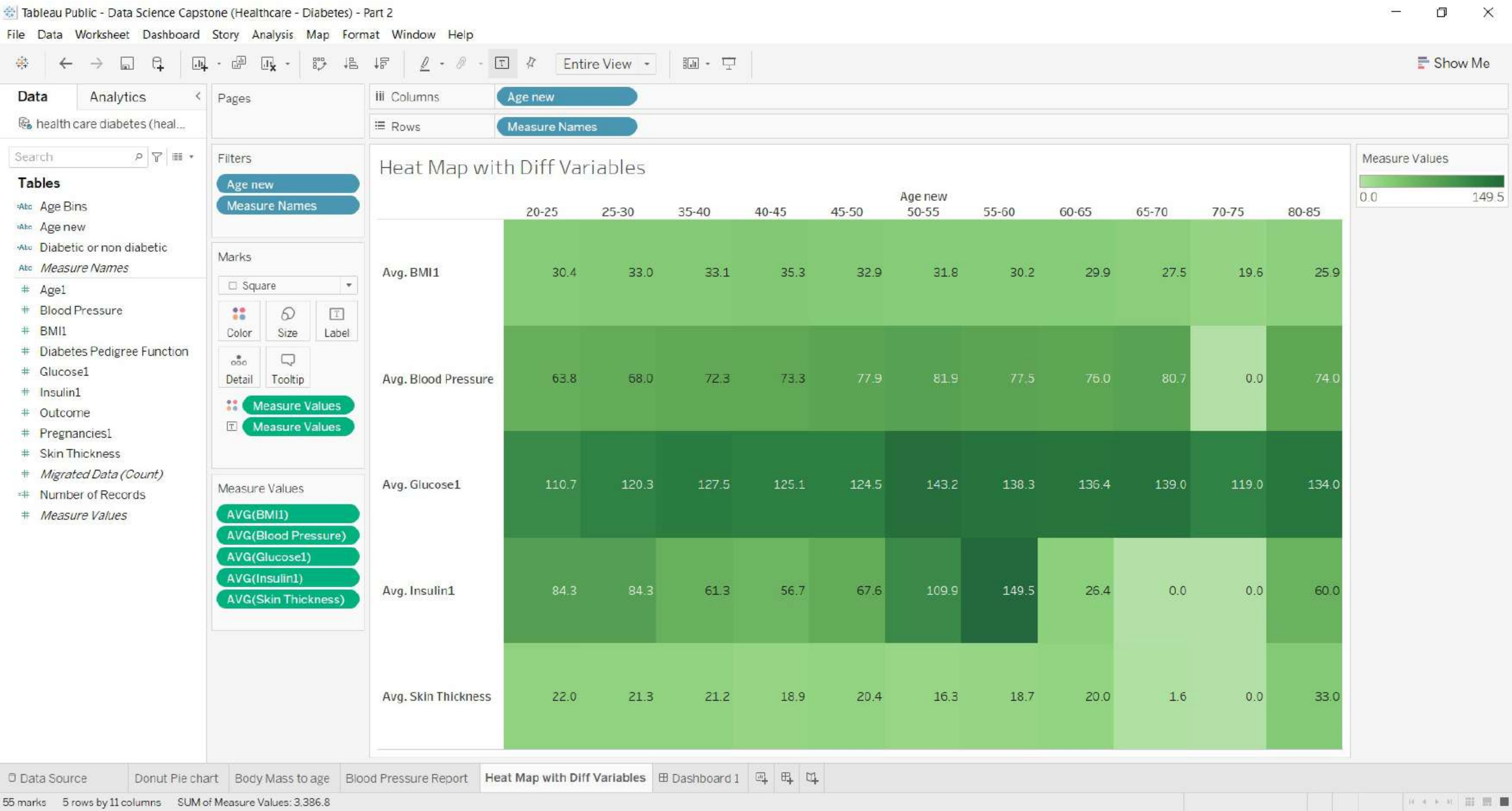


Tableau Public - Data Science Capstone (Healthcare - Diabetes) - Part 2
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Entire View
Show Me
Data
Analytics
health care diabetes (heal...
Pages
Columns
Age new
Rows
Measure Names
Filters
Age new
Measure Names
Tables
Age Bins
Age new
Diabetic or non diabetic
Measure Names
Age1
Blood Pressure
BMI1
Diabetes Pedigree Function
Glucose1
Insulin1
Outcome
Pregnancies1
Skin Thickness
Migrated Data (Count)
Number of Records
Measure Values
Marks
Square
Color
Size
Label
Detail
Tooltip
Measure Values
Measure Values
Measure Values
AVG(BMI1)
AVG(Blood Pressure)
AVG(Glucose1)
AVG(Insulin1)
AVG(Skin Thickness)
Heat Map with Diff Variables
Age new
20-25 25-30 35-40 40-45 45-50 50-55 55-60 60-65 65-70 70-75 80-85
Avg. BMI1 30.4 33.0 33.1 35.3 32.9 31.8 30.2 29.9 27.5 19.6 25.9
Avg. Blood Pressure 63.8 68.0 72.3 73.3 77.9 81.9 77.5 76.0 80.7 0.0 74.0
Avg. Glucose1 110.7 120.3 127.5 125.1 124.5 143.2 138.3 136.4 139.0 119.0 134.0
Avg. Insulin1 84.3 84.3 61.3 56.7 67.6 109.9 149.5 26.4 0.0 0.0 60.0
Avg. Skin Thickness 22.0 21.3 21.2 18.9 20.4 16.3 18.7 20.0 1.6 0.0 33.0
Measure Values
0.0 149.5
Data Source
Donut Pie chart
Body Mass to age
Blood Pressure Report
Heat Map with Diff Variables
Dashboard 1
55 marks 5 rows by 11 columns SUM of Measure Values: 3,386.8