In [224]:
```
# NAME- ANKIT KUMAR PATHAK
#ROLL NO -201046016


# EXPLORATORY DATA ANALYSIS
# This dataset was originally from the National Institute of Diabetes
# and Digestive and Kidney Diseases.
# The purpose of the dataset is to diagnostically predict whether a patient
# has diabetes based on the specific diagnostic measures included in the data set.
# Various restrictions have been imposed on the selection of these
# samples from a larger database
```

In [199]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import plotly.graph_objs as go
import plotly.offline as py
```

In [200]:
```
# Reading Data
diabetes=pd.read_csv('diabetes.csv')
```

df = diabetes.copy() df.head()

In [201]:
```
df
```

Out[201]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0
2	8	183.0	64.0	32.0	169.5	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
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	102.5	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	32.0	169.5	30.1	0.349	47	1
767	1	93.0	70.0	31.0	102.5	30.4	0.315	23	0

768 rows × 9 columns

In [202]:
```
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
```

In [ ]:

In [203]:
```
df.describe().T
```

Out[203]:

	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	121.677083	30.464161	44.000	99.75000	117.0000	140.25000	199.00
BloodPressure	768.0	72.389323	12.106039	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	768.0	29.089844	8.890820	7.000	25.00000	28.0000	32.00000	99.00
Insulin	768.0	141.753906	89.100847	14.000	102.50000	102.5000	169.50000	846.00
BMI	768.0	32.434635	6.880498	18.200	27.50000	32.0500	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
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

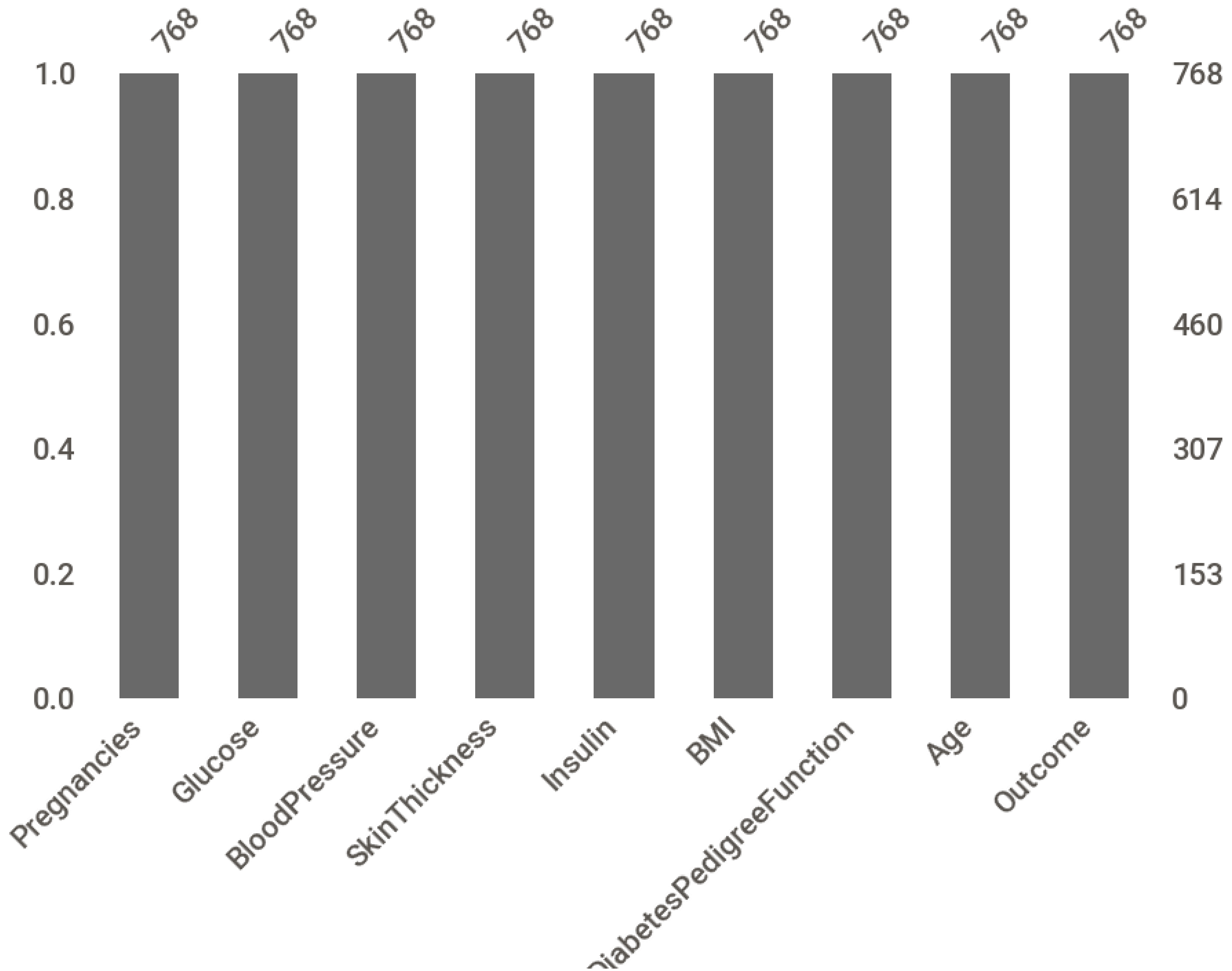
In [204]:
```
# Handling with Missing Values
# In this dataset missing data are filled with 0. First, we are gonna change zeros with NaN
```

In [205]:
```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness',
                                                                      'Insulin','BMI']].replace(0, np.NaN)
```

In [206]:
```
# After filling the 0s with the value of NaN, the missing values
# will be visualized. We use the missingno library for this.
```

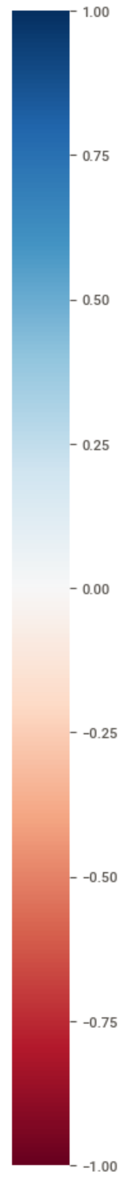
In [207]:
```
msno.bar(df,figsize=(10,6))
```

```
Out[207]: <matplotlib.axes._subplots.AxesSubplot at 0x2058e788c10>
```


768
768
768
768
768
768
768
768
768
1.0
0.8
0.6
0.4
0.2
0.0
768
614
460
307
153
0
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

In [208]:
```
msno.heatmap(df);
```

```
C:\anaconda\lib\site-packages\seaborn\matrix.py:305: UserWarning:

Attempting to set identical bottom == top == 0 results in singular transformations; automatically expanding.

C:\anaconda\lib\site-packages\seaborn\matrix.py:305: UserWarning:

Attempting to set identical left == right == 0 results in singular transformations; automatically expanding.
```



In [209]:
```
# We will fill in each missing value with its median value.
```

In [210]:
```
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return temp
```

In [211]:
```
columns = df.columns
columns = columns.drop("Outcome")
for i in columns:
    median_target(i)
    df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] = median_target(i)[i][0]
    df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] = median_target(i)[i][1]
```

In [212]:
```
# After filling if we examine null values in dataset, we will see there are not any missing values.
```

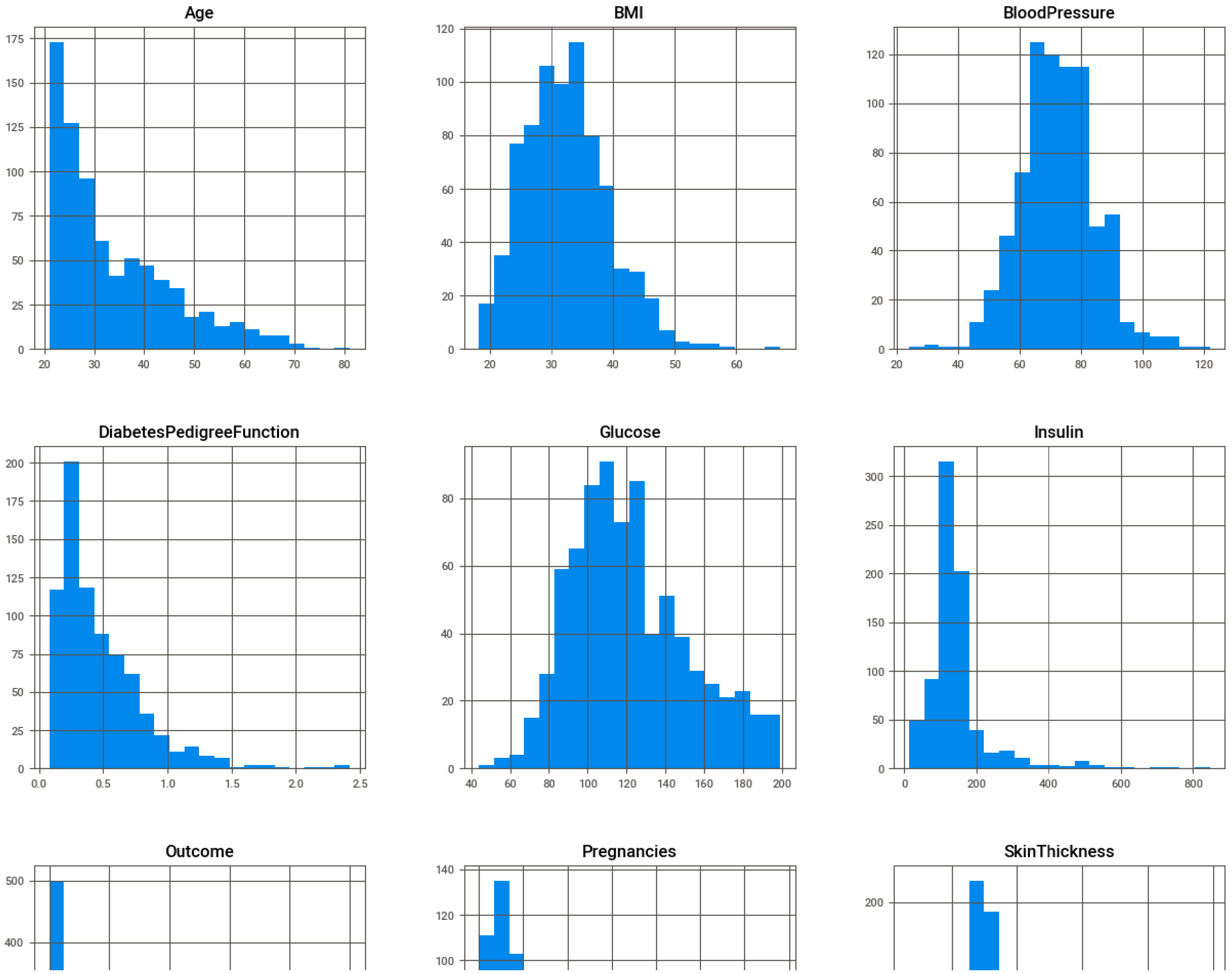
In [213]:
```
df.isnull().sum()
```

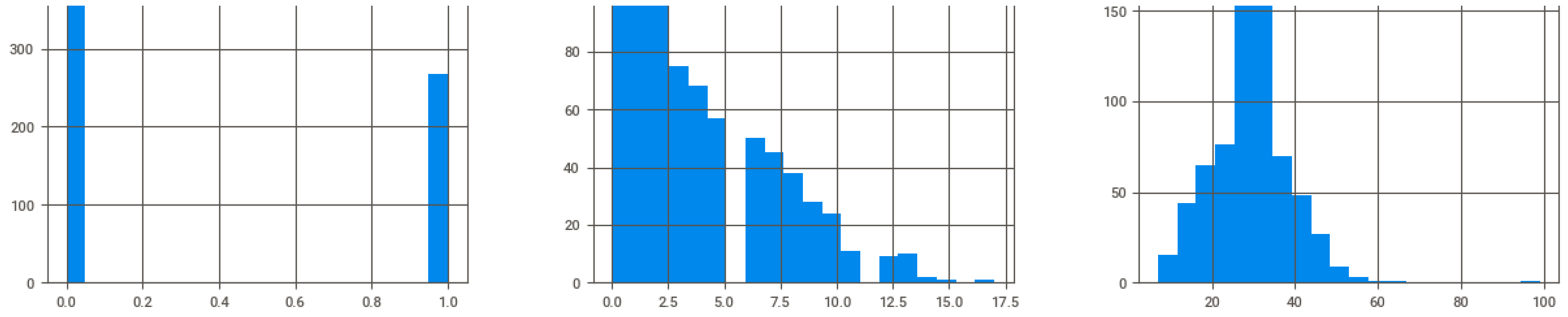
Out[213]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [214]:
```
# Data Visualization
```

In [215]:
```
# Histogram-A histogram is a bar graph representation of a grouped data distribution
```

In [216]:
```
df.hist(bins=20,figsize = (15,15));
```

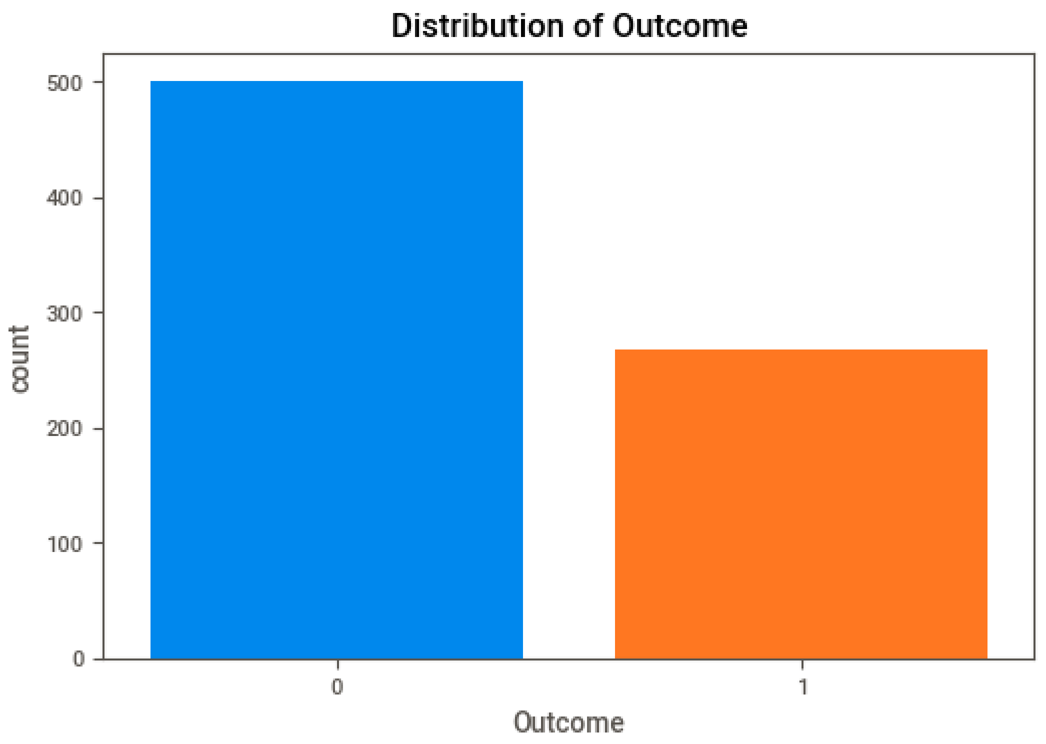

Age
BMI
BloodPressure
DiabetesPedigreeFunction
Glucose
Insulin
Outcome
Pregnancies
SkinThickness



In [217]:
```
# count plot
```

In [218]:
```
plt.title("Distribution of Outcome")
sns.countplot(df["Outcome"], saturation=1)
```

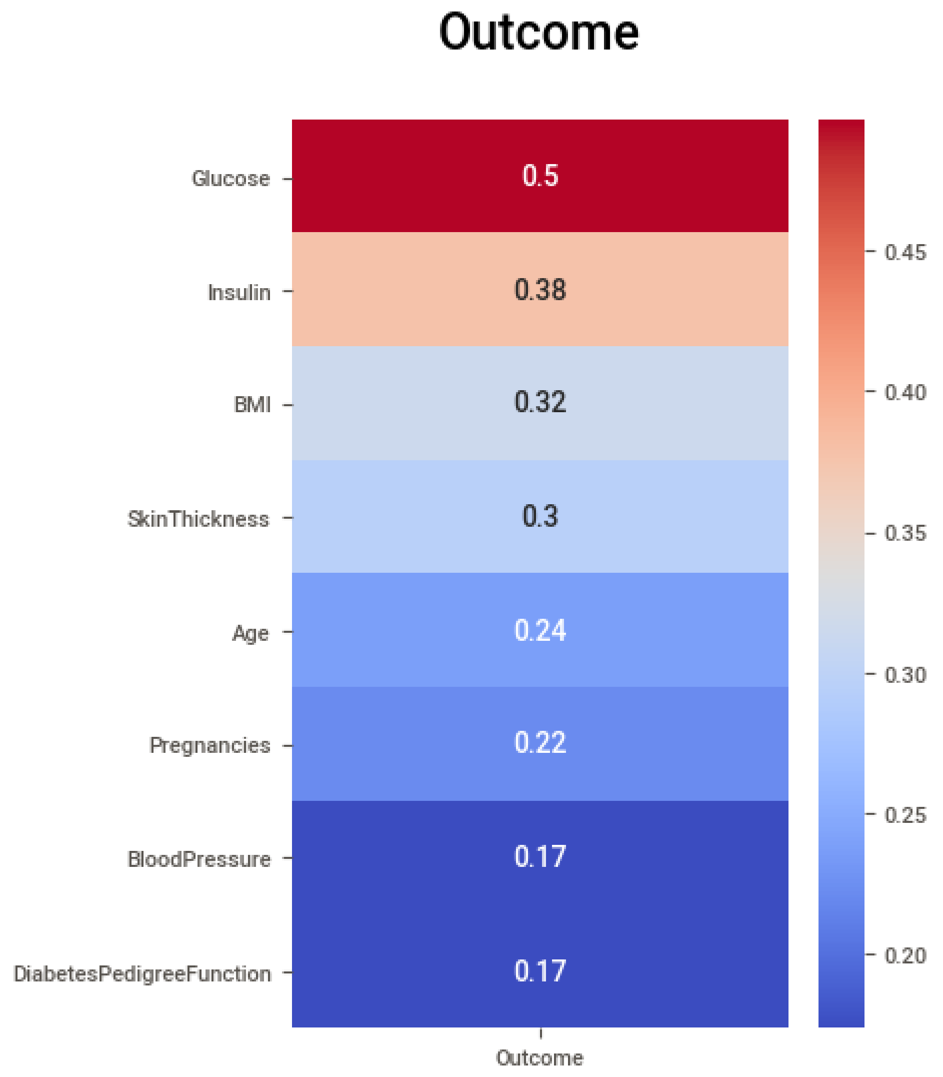
Out[218]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x205975c0820>
```



In [219]:
```
# pie plot
```

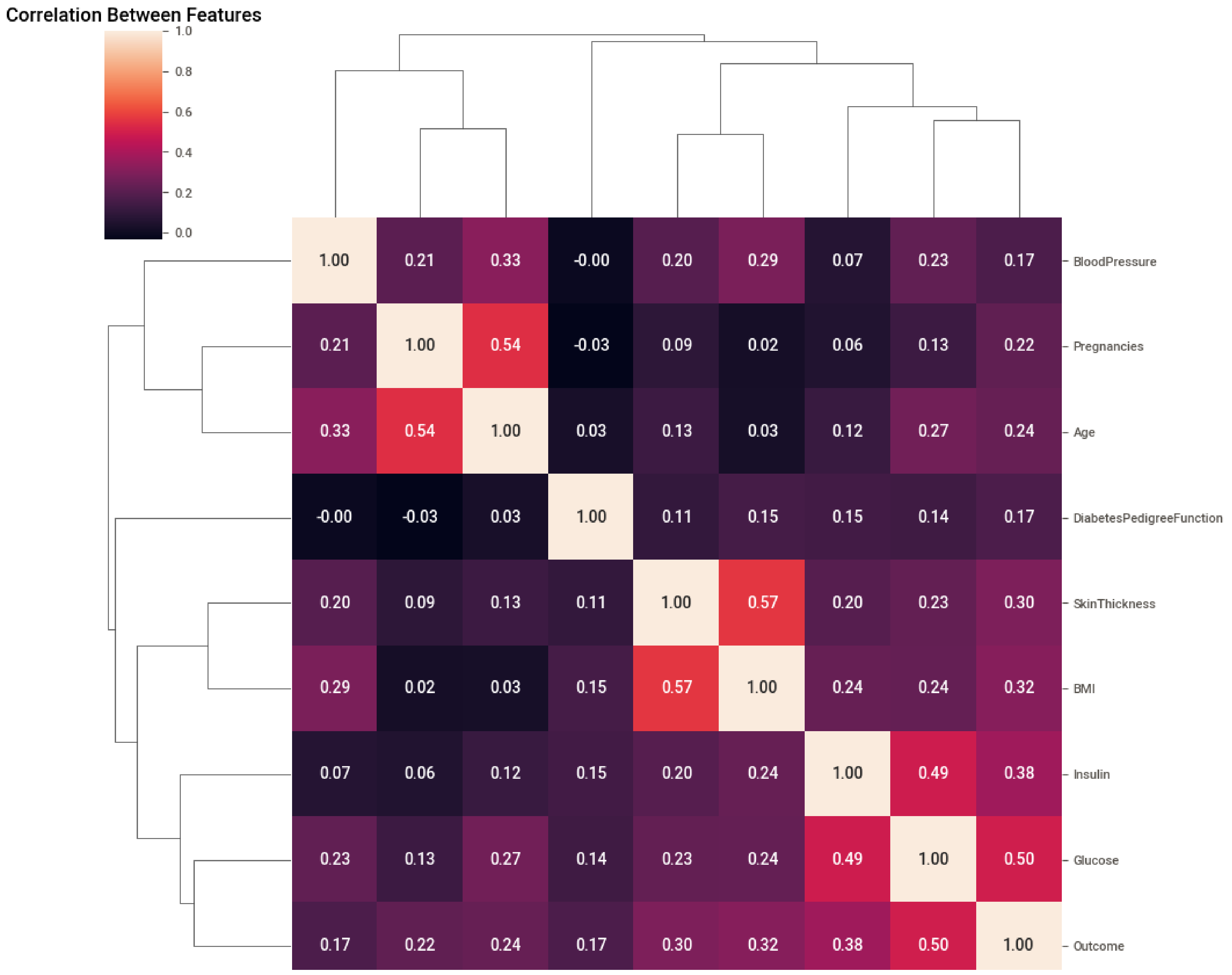
In [220]:
```
def PlotPie(df, nameOfFeature):
    labels = [str(df[nameOfFeature].unique()[i]) for i in range(df[nameOfFeature].nunique())]
    values = [df[nameOfFeature].value_counts()[i] for i in range(df[nameOfFeature].nunique())]

    trace=go.Pie(labels=labels,values=values)

    py.iplot([trace])

PlotPie(df, "Outcome")
```

In [221]:
```
# Correlation
def corr_to_target(dataframe, target, title=None, file=None):
    plt.figure(figsize=(4,6))
    sns.heatmap(dataframe.corr()[[target]].sort_values(target,
                                                        ascending=False)[1:],
                                                        annot=True,
                                                        cmap='coolwarm')

    plt.title(f'\n{title}\n', fontsize=18)

    plt.show();

    return

corr_to_target(df, "Outcome", title="Outcome")
```

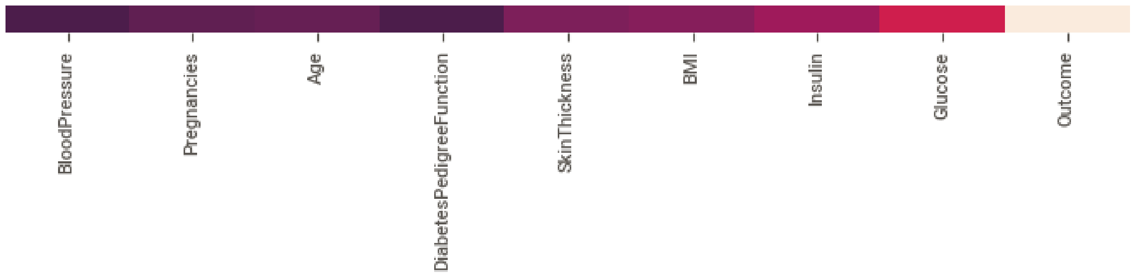


```
In [222]: # Correlation matrix of variables with each other.
In [223]: corr_matrix = df.corr()
          sns.clustermap(corr_matrix, annot=True, fmt=".2f")
          plt.title("Correlation Between Features")
```

```
Out[223]: Text(0.5, 1.0, 'Correlation Between Features')
```



1.00	0.21	0.33	-0.00	0.20	0.29	0.07	0.23	0.17	BloodPressure
0.21	1.00	0.54	-0.03	0.09	0.02	0.06	0.13	0.22	Pregnancies
0.33	0.54	1.00	0.03	0.13	0.03	0.12	0.27	0.24	Age
-0.00	-0.03	0.03	1.00	0.11	0.15	0.15	0.14	0.17	DiabetesPedigreeFunction
0.20	0.09	0.13	0.11	1.00	0.57	0.20	0.23	0.30	SkinThickness
0.29	0.02	0.03	0.15	0.57	1.00	0.24	0.24	0.32	BMI
0.07	0.06	0.12	0.15	0.20	0.24	1.00	0.49	0.38	Insulin
0.23	0.13	0.27	0.14	0.23	0.24	0.49	1.00	0.50	Glucose
0.17	0.22	0.24	0.17	0.30	0.32	0.38	0.50	1.00	Outcome



In [225]:
```
import dtale
```

In [ ]:
```
dtale.show(df, ignore_duplicate=True)
```

In [ ]:
```
import sweetviz
```

In [ ]:
```
diabetes=pd.read_csv('diabetes.csv')
# importing sweetviz
import sweetviz as sv

#analyzing the dataset
advert_report = sv.analyze(df)
#display the report
advert_report.show_html('Advertising.html')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [23]:

In [ ]:

In [26]:

In [ ]:

In [ ]: