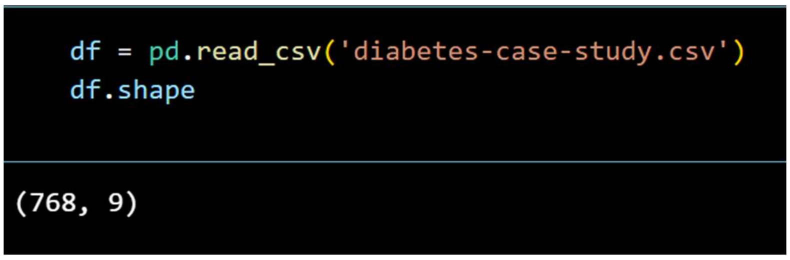
SDS DATATHON

Topic :- Case Study on Diabetes

Data Description Statistics

a. What is the structure (shape) of the dataset?

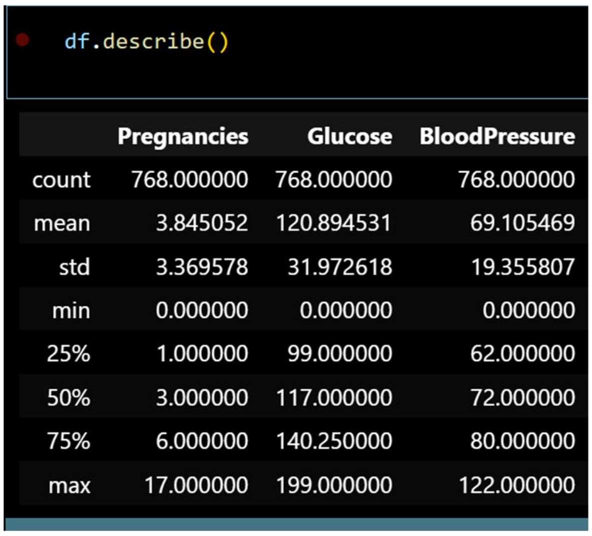


```
df = pd.read_csv('diabetes-case-study.csv')
df.shape
```

```
(768, 9)
```

The output shows that the DataFrame has 768 rows and 9 columns.

b. Show the min, max, and mean of Glucose, ...?



```
df.describe()
```

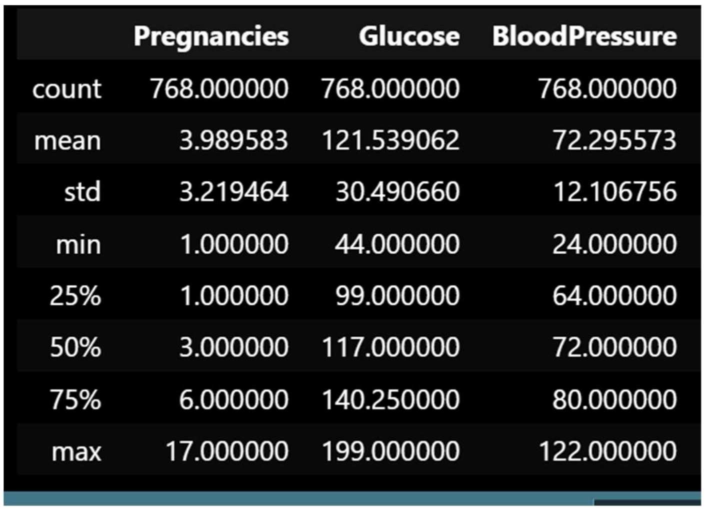
	Pregnancies	Glucose	BloodPressure
count	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469
std	3.369578	31.972618	19.355807
min	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000
50%	3.000000	117.000000	72.000000
75%	6.000000	140.250000	80.000000
max	17.000000	199.000000	122.000000

The output shows the mean of Glucose as 120.8945

Min :- 0

Max :- 199

After cleaning the data:-



	Pregnancies	Glucose	BloodPressure
count	768.000000	768.000000	768.000000
mean	3.989583	121.539062	72.295573
std	3.219464	30.490660	12.106756
min	1.000000	44.000000	24.000000
25%	1.000000	99.000000	64.000000
50%	3.000000	117.000000	72.000000
75%	6.000000	140.250000	80.000000
max	17.000000	199.000000	122.000000

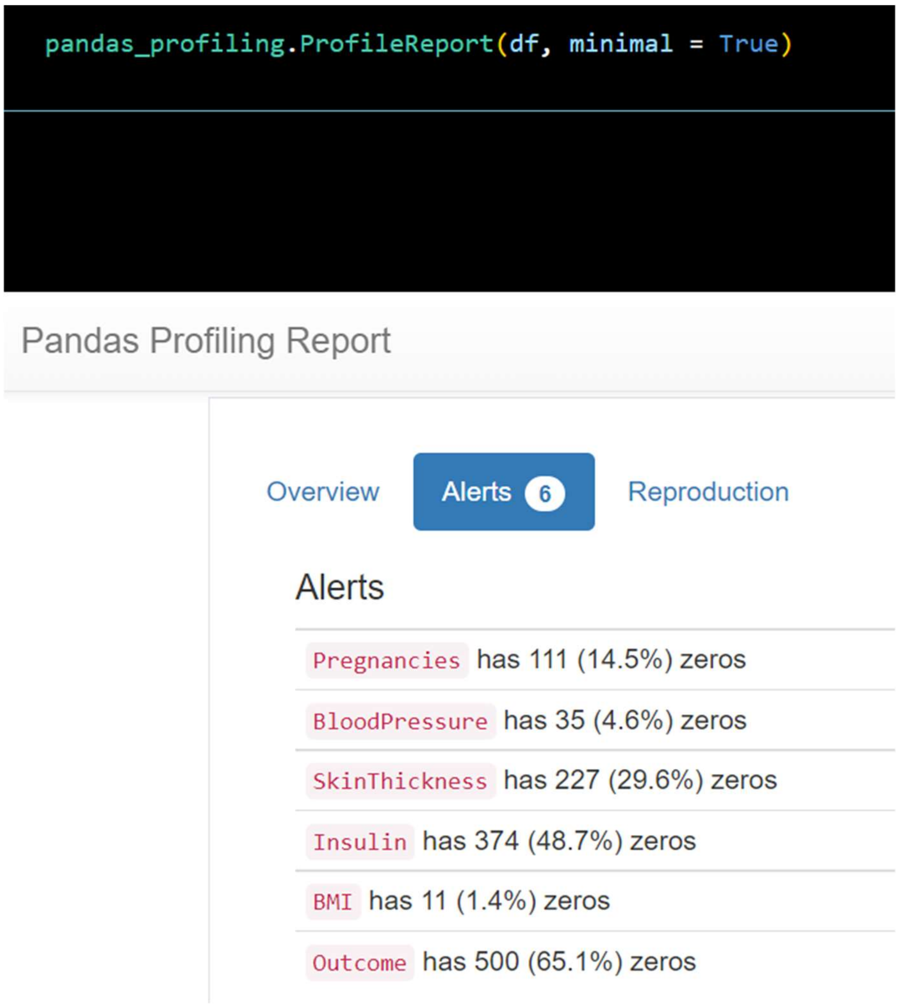
The mean of Glucose is 121.5392

Min :- 44

Max :- 199

Pre-processing

a. Check for NULLs/Duplicates. Drop attributes with more than 20% data missing.



```
pandas_profiling.ProfileReport(df, minimal = True)
```

Pandas Profiling Report

Alerts

`Pregnancies` has 111 (14.5%) zeros

`BloodPressure` has 35 (4.6%) zeros

`SkinThickness` has 227 (29.6%) zeros

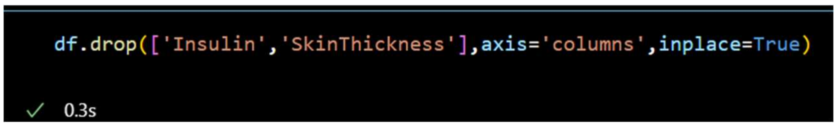
`Insulin` has 374 (48.7%) zeros

`BMI` has 11 (1.4%) zeros

`Outcome` has 500 (65.1%) zeros

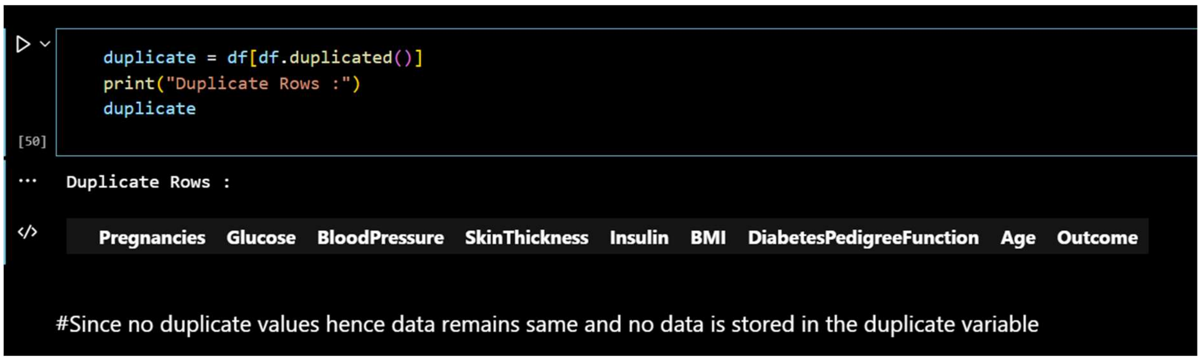
The output shows that SkinThickness, Insulin are the two columns that have more 20% of the data as 0. (not exactly a data)

So we drop those two columns from our DataFrame.



```
df.drop(['Insulin','SkinThickness'],axis='columns',inplace=True)
```

For duplicated values:-



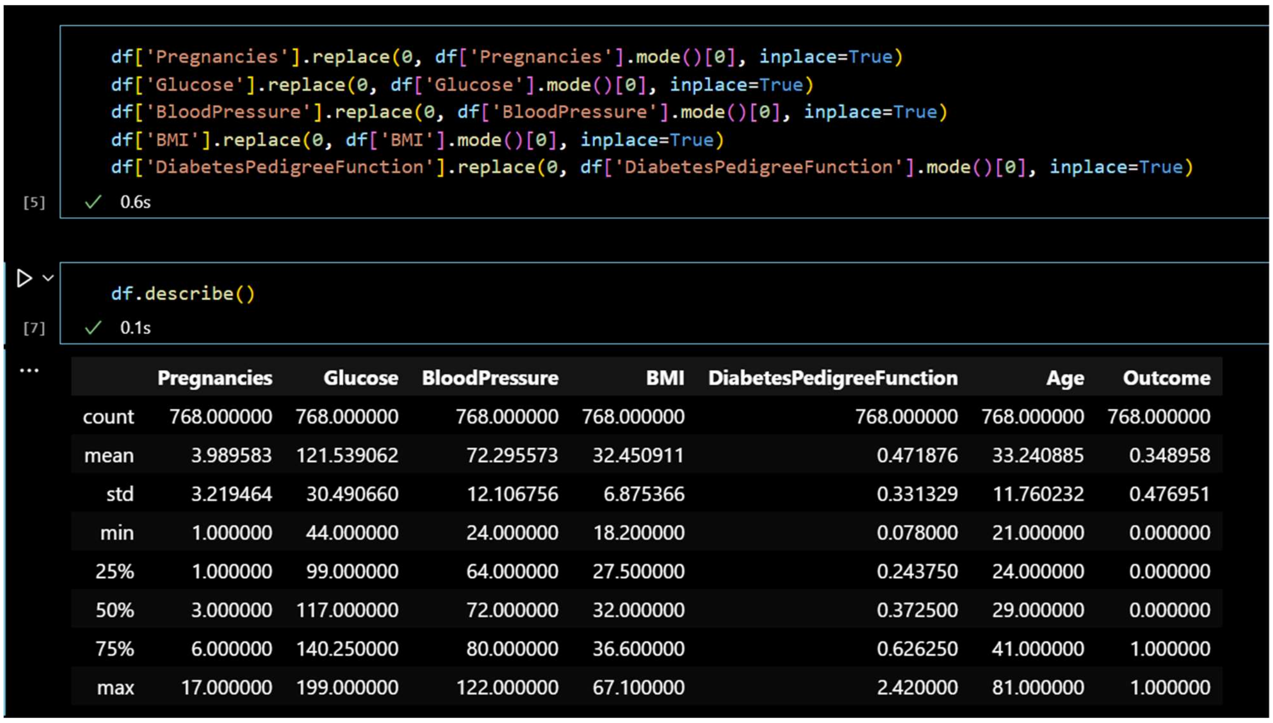
```
duplicate = df[df.duplicated()]
print("Duplicate Rows :")
duplicate
```

```
Duplicate Rows :
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

#Since no duplicate values hence data remains same and no data is stored in the duplicate variable

b. Fill remaining NULLs with mode values



```
df['Pregnancies'].replace(0, df['Pregnancies'].mode()[0], inplace=True)
df['Glucose'].replace(0, df['Glucose'].mode()[0], inplace=True)
df['BloodPressure'].replace(0, df['BloodPressure'].mode()[0], inplace=True)
df['BMI'].replace(0, df['BMI'].mode()[0], inplace=True)
df['DiabetesPedigreeFunction'].replace(0, df['DiabetesPedigreeFunction'].mode()[0], inplace=True)
```

[5] ✓ 0.6s

```
df.describe()
```

[7] ✓ 0.1s

	Pregnancies	Glucose	BloodPressure	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.989583	121.539062	72.295573	32.450911	0.471876	33.240885	0.348958
std	3.219464	30.490660	12.106756	6.875366	0.331329	11.760232	0.476951
min	1.000000	44.000000	24.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	67.100000	2.420000	81.000000	1.000000

We can see that the 0 data in new database is replaced by mode of that particular column.

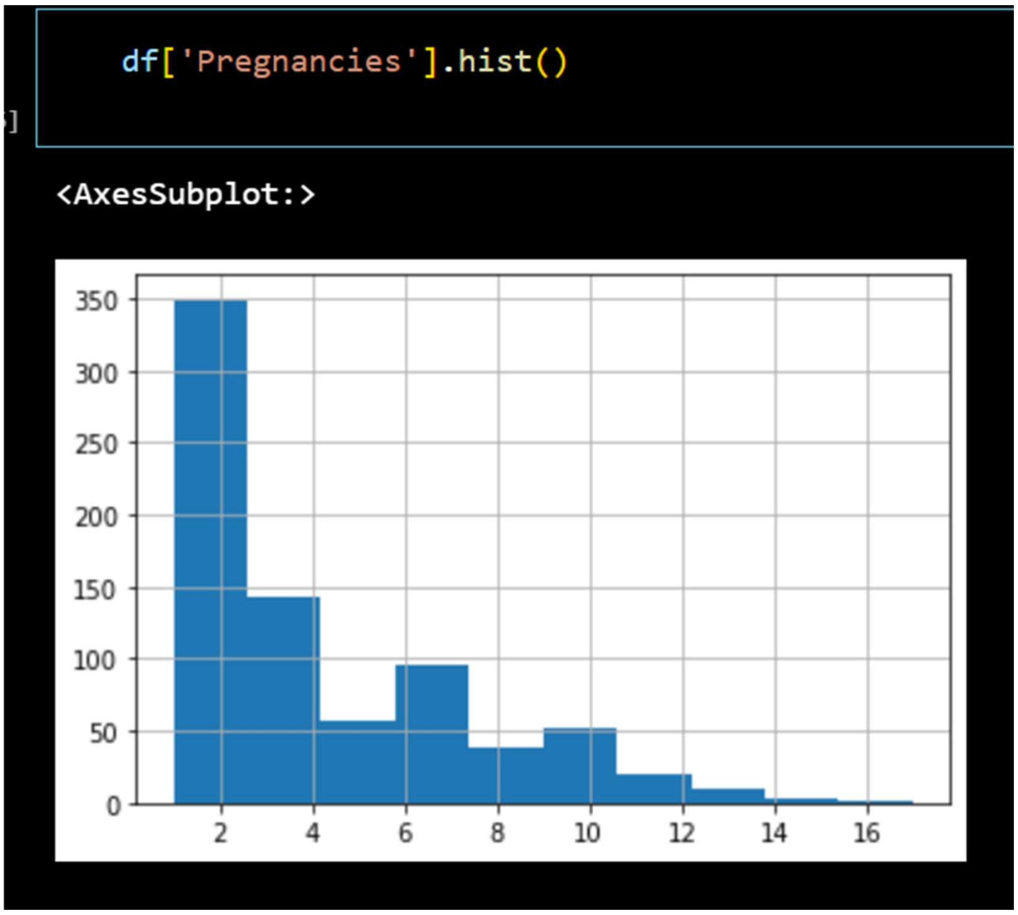
c. Are there categorical columns ?

Since all the data is in numeric values, there is no categorical data. So no need of encoding.

Data Visualization

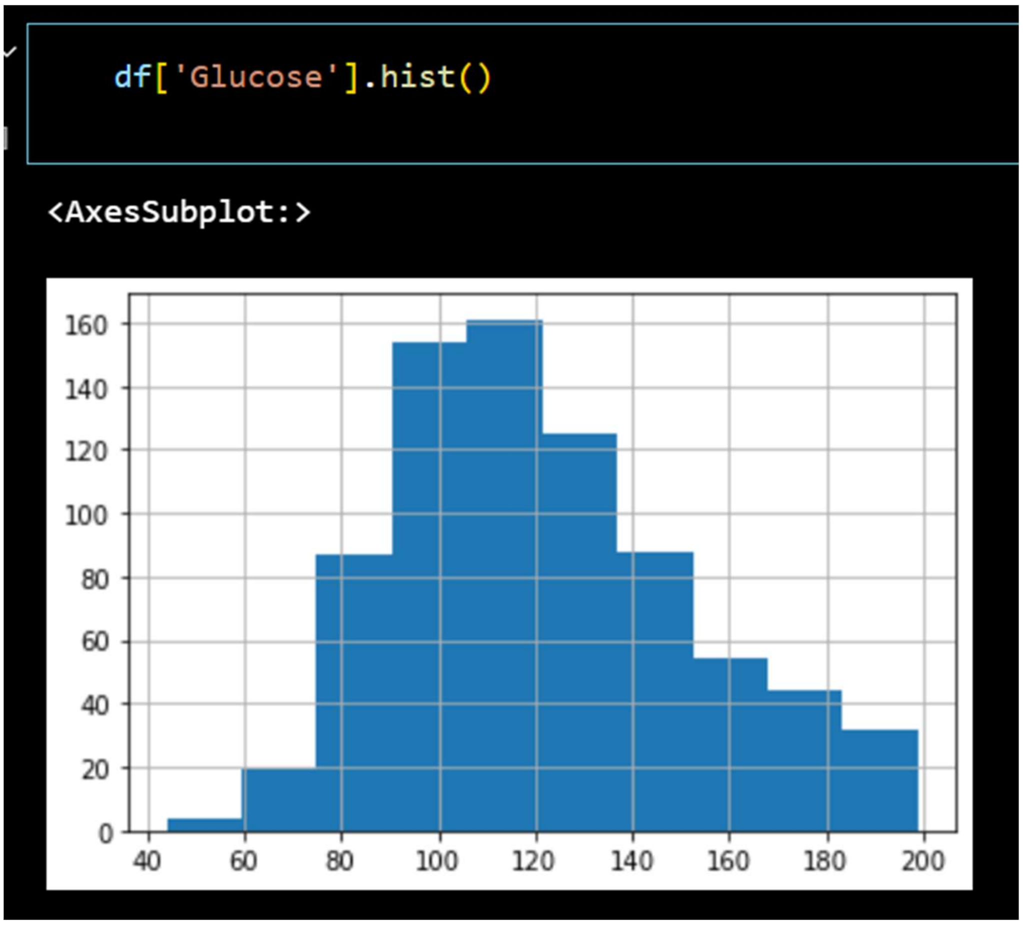
a. Make Histogram, and whisker plots to understand the meaning of the encoding.

We saw that encoding was not needed in our data. The Histogram and Whisker plots are shown below:-



```
df['Pregnancies'].hist()
```

```
<AxesSubplot:>
```

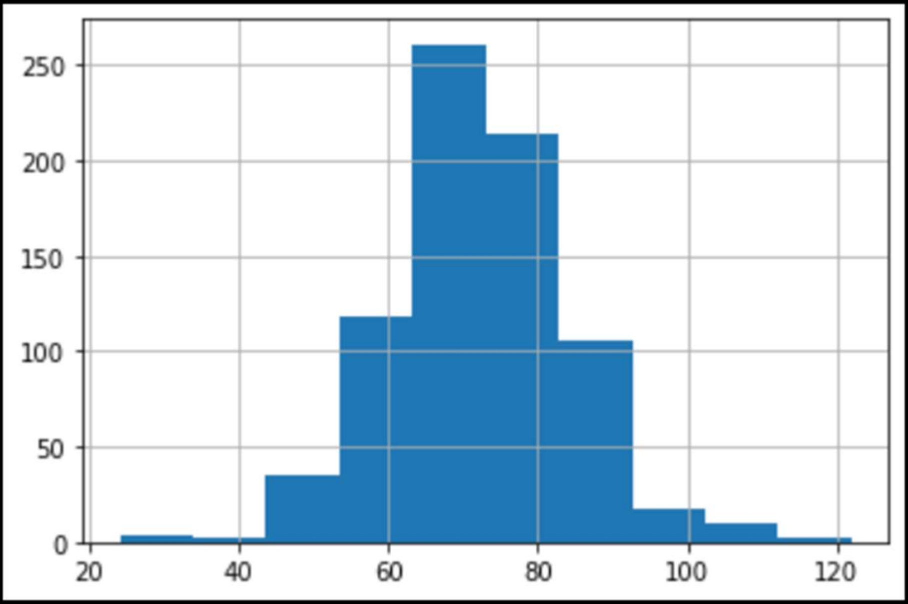


```
df['Glucose'].hist()
```

```
<AxesSubplot:>
```

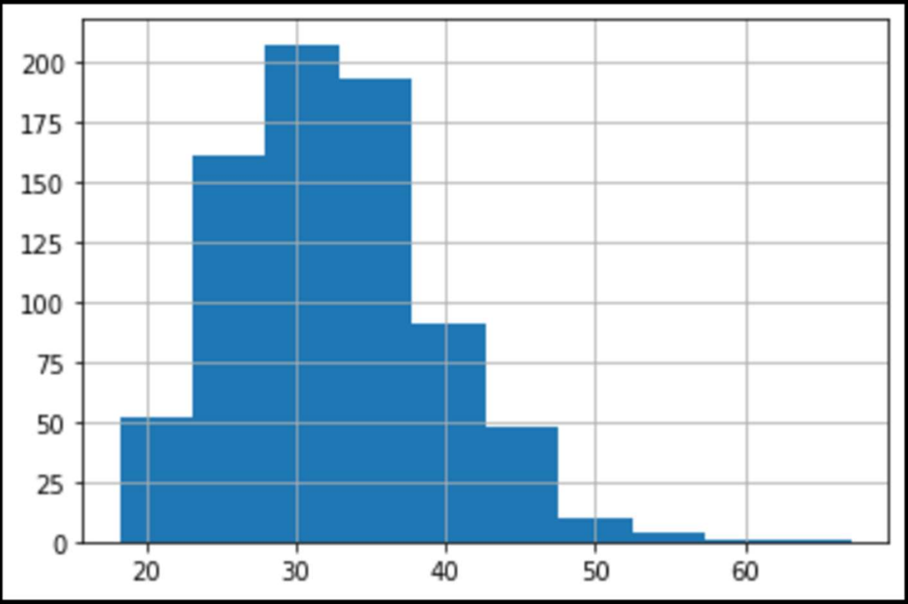
```
df['BloodPressure'].hist()
```

```
<AxesSubplot:>
```



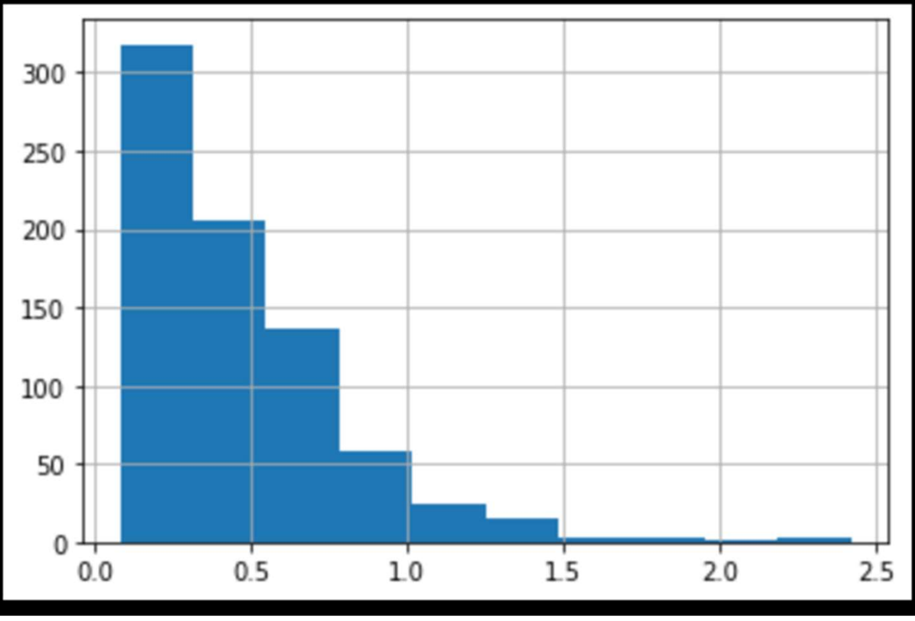
```
df['BMI'].hist()
```

```
<AxesSubplot:>
```



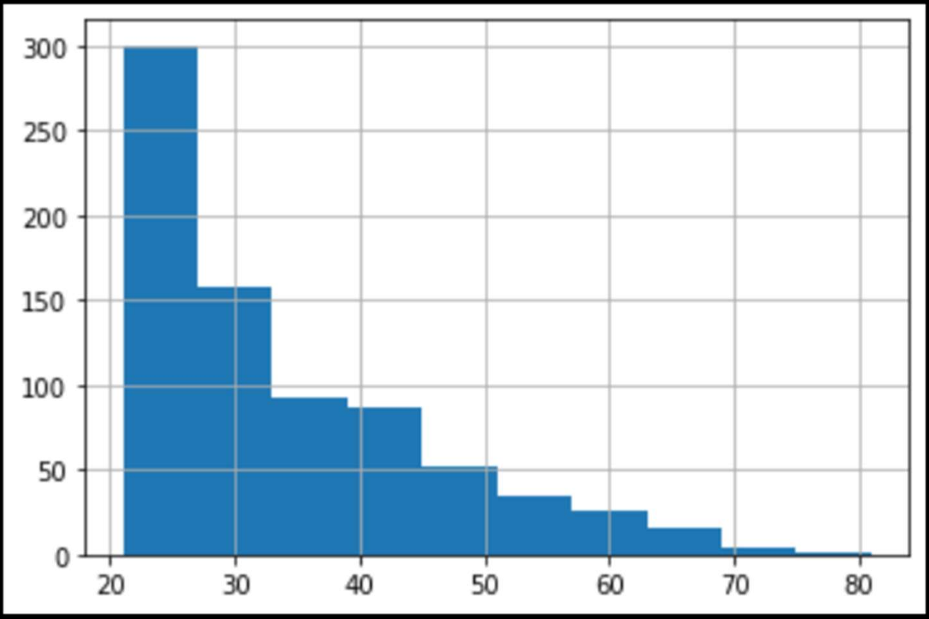
```
df['DiabetesPedigreeFunction'].hist()
```

```
<AxesSubplot:>
```



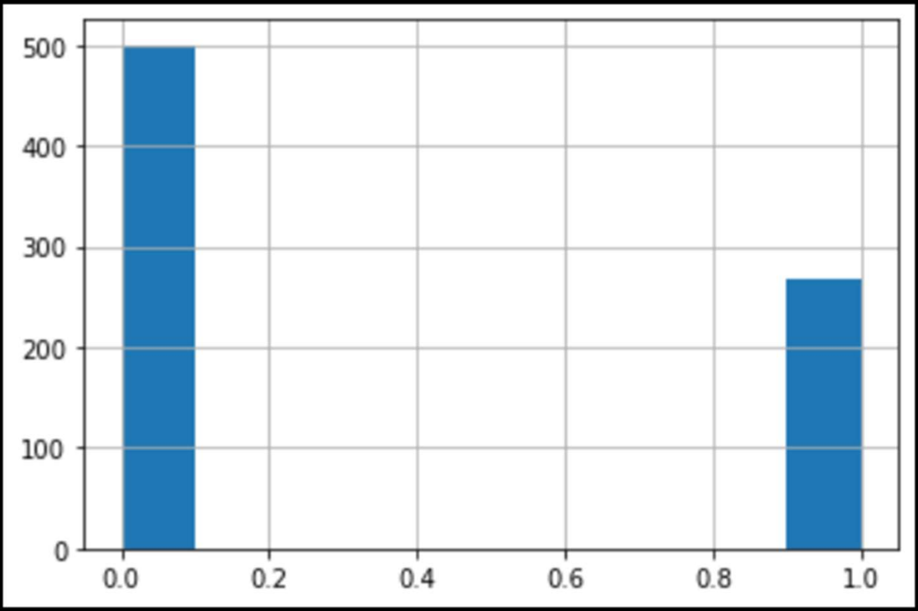
```
df['Age'].hist()
```

```
<AxesSubplot:>
```



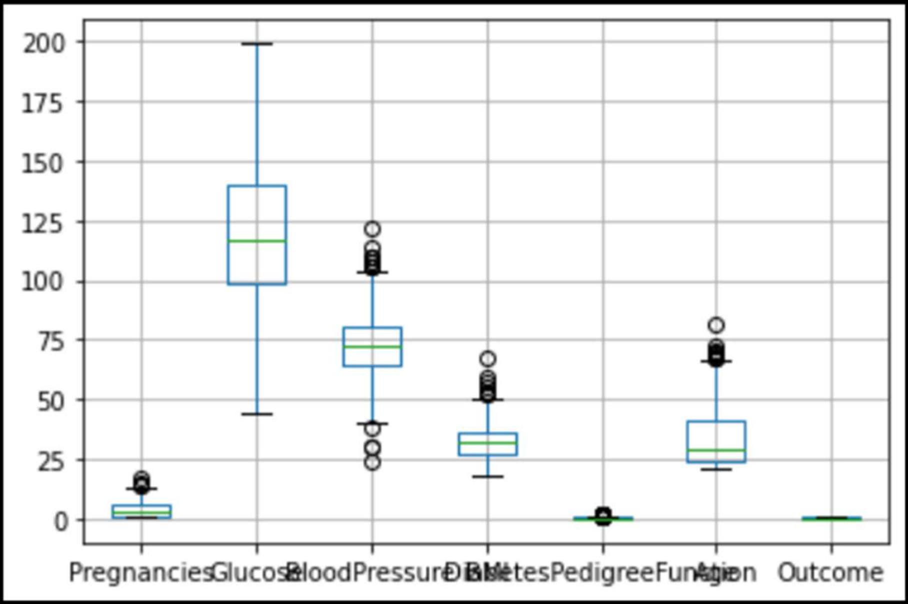
```
df['Outcome'].hist()
```

```
<AxesSubplot:>
```



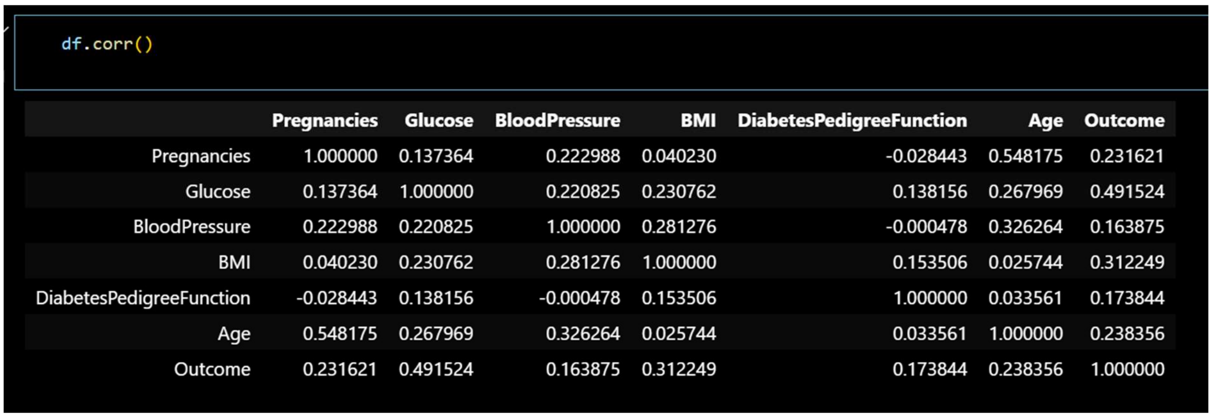
```
df.boxplot()
```

```
<AxesSubplot:>
```



Hypothesis Testing

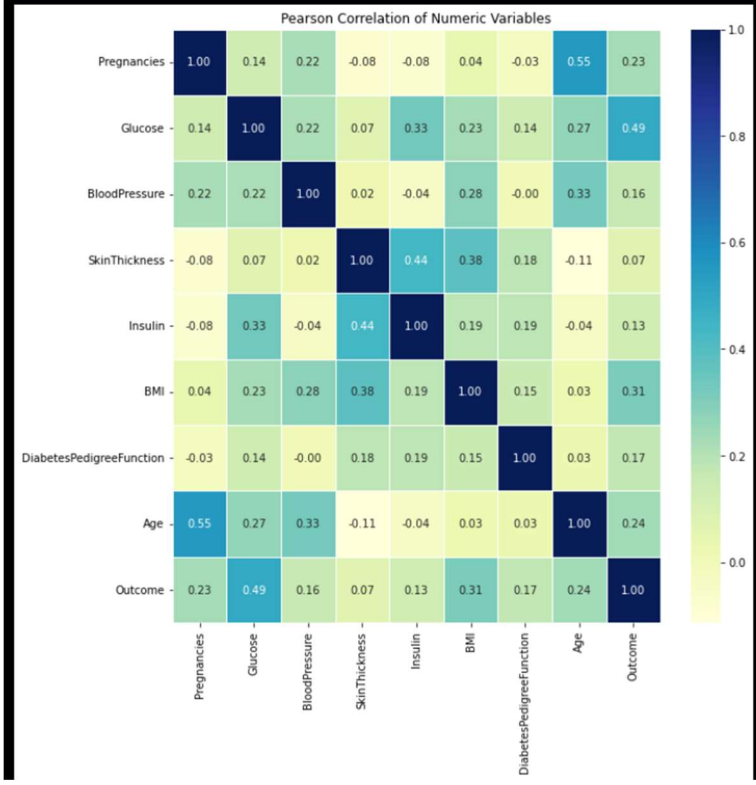
a. Perform correlation Analysis.



```python
df.corr()
```

	Pregnancies	Glucose	BloodPressure	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.137364	0.222988	0.040230	-0.028443	0.548175	0.231621
Glucose	0.137364	1.000000	0.220825	0.230762	0.138156	0.267969	0.491524
BloodPressure	0.222988	0.220825	1.000000	0.281276	-0.000478	0.326264	0.163875
BMI	0.040230	0.230762	0.281276	1.000000	0.153506	0.025744	0.312249
DiabetesPedigreeFunction	-0.028443	0.138156	-0.000478	0.153506	1.000000	0.033561	0.173844
Age	0.548175	0.267969	0.326264	0.025744	0.033561	1.000000	0.238356
Outcome	0.231621	0.491524	0.163875	0.312249	0.173844	0.238356	1.000000

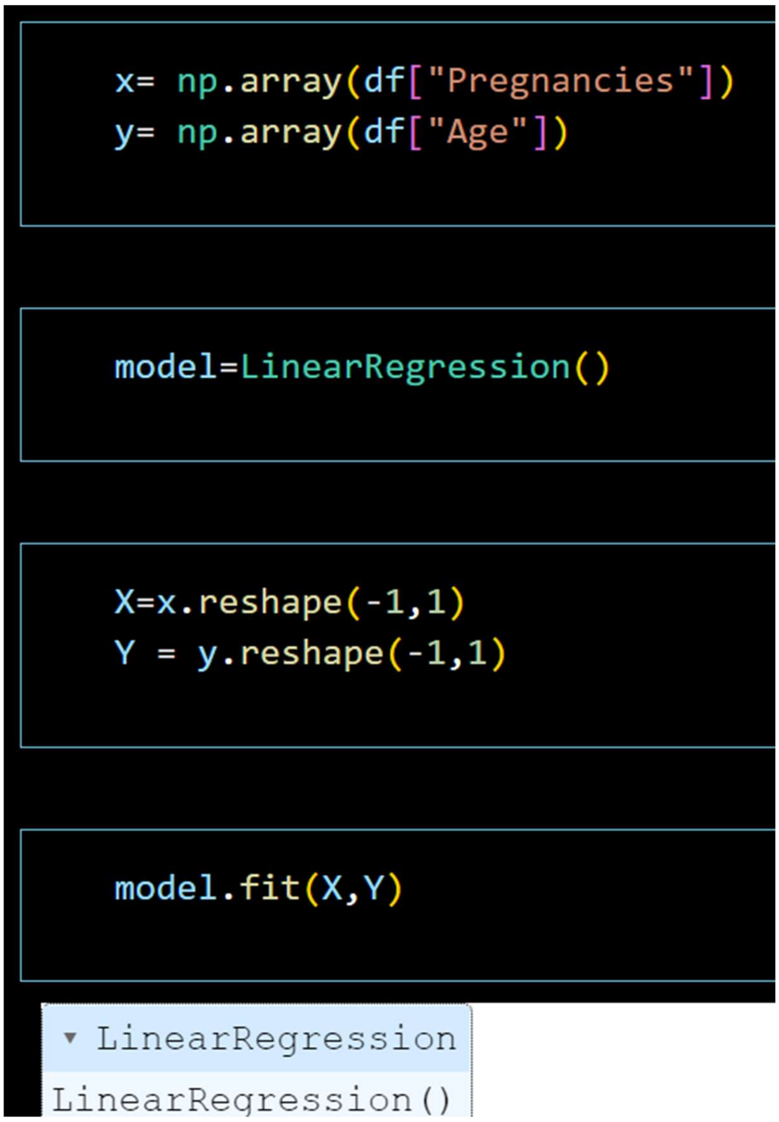
```python
plt.figure(figsize=(10,10))
num_corr = df.corr(method = 'pearson')
ax = sns.heatmap(num_corr, annot=True, cmap="YlGnBu", linewidths=.5, fmt = '.2f').set(title = 'Pearson Correlation of Numeric Variables')
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.00	0.14	0.22	-0.08	-0.08	0.04	-0.03	0.55	0.23
Glucose	0.14	1.00	0.22	0.07	0.33	0.23	0.14	0.27	0.49
BloodPressure	0.22	0.22	1.00	0.02	-0.04	0.28	-0.00	0.33	0.16
SkinThickness	-0.08	0.07	0.02	1.00	0.44	0.38	0.18	-0.11	0.07
Insulin	-0.08	0.33	-0.04	0.44	1.00	0.19	0.19	-0.04	0.13
BMI	0.04	0.23	0.28	0.38	0.19	1.00	0.15	0.03	0.31
DiabetesPedigreeFunction	-0.03	0.14	-0.00	0.18	0.19	0.15	1.00	0.03	0.17
Age	0.55	0.27	0.33	-0.11	-0.04	0.03	0.03	1.00	0.24
Outcome	0.23	0.49	0.16	0.07	0.13	0.31	0.17	0.24	1.00

Modelling

a. Build a Linear Regression Model.
1. MAE, MSE, and RMSE results.
2. Linear Regression R2 score.



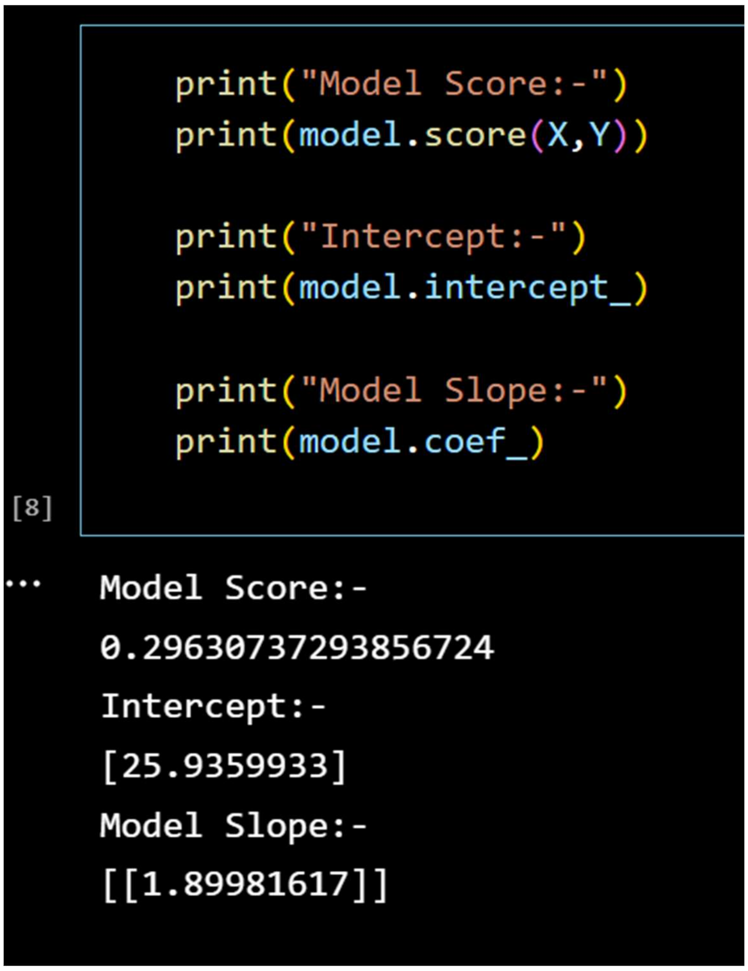
```
x= np.array(df["Pregnancies"])
y= np.array(df["Age"])
```

```
model=LinearRegression()
```

```
X=x.reshape(-1,1)
Y = y.reshape(-1,1)
```

```
model.fit(X,Y)
```

```
▾ LinearRegression
LinearRegression()
```



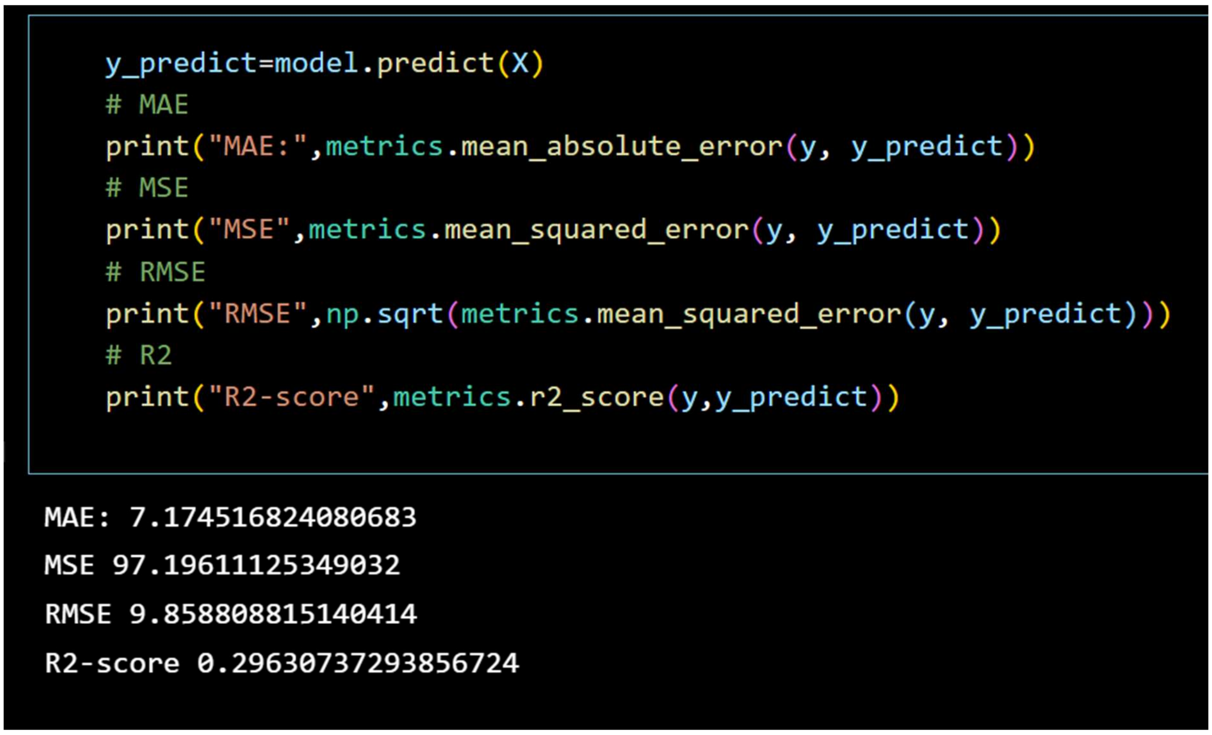
```
print("Model Score:-")
print(model.score(X,Y))

print("Intercept:-")
print(model.intercept_)

print("Model Slope:-")
print(model.coef_)
```

[8]

```
Model Score:-
0.29630737293856724
Intercept:-
[25.9359933]
Model Slope:-
[[1.89981617]]
```



```
y_predict=model.predict(X)
# MAE
print("MAE:",metrics.mean_absolute_error(y, y_predict))
# MSE
print("MSE",metrics.mean_squared_error(y, y_predict))
# RMSE
print("RMSE",np.sqrt(metrics.mean_squared_error(y, y_predict)))
# R2
print("R2-score",metrics.r2_score(y,y_predict))
```

```
MAE: 7.174516824080683
MSE 97.19611125349032
RMSE 9.858808815140414
R2-score 0.29630737293856724
```