Import Libraries ¶

In [17]:

```
import pandas as pd
import matplotlib.pyplot as pit
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Loading Dataset

In [5]:

```
data = pd.read_csv(r"C:\Users\hp\Desktop\PROJECT\diabetes.csv")
data
```

Out[5]:

	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

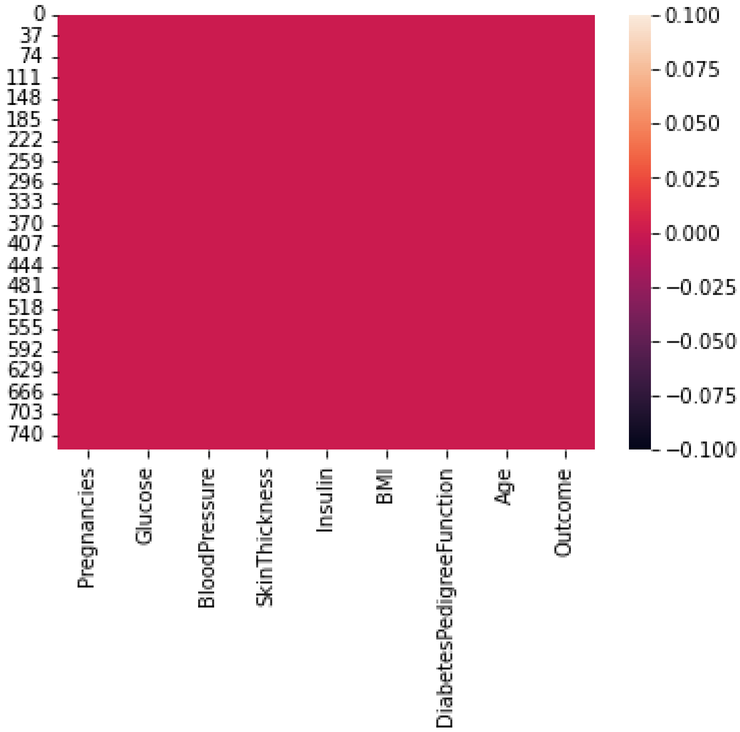
Checking for Missing Values

In [6]:

```
sns.heatmap(data.isnull())
```

Out[6]:

```
<AxesSubplot:>
```



Correlation Matrix

In [7]:

```
correlation = data.corr()
print(correlation)
```

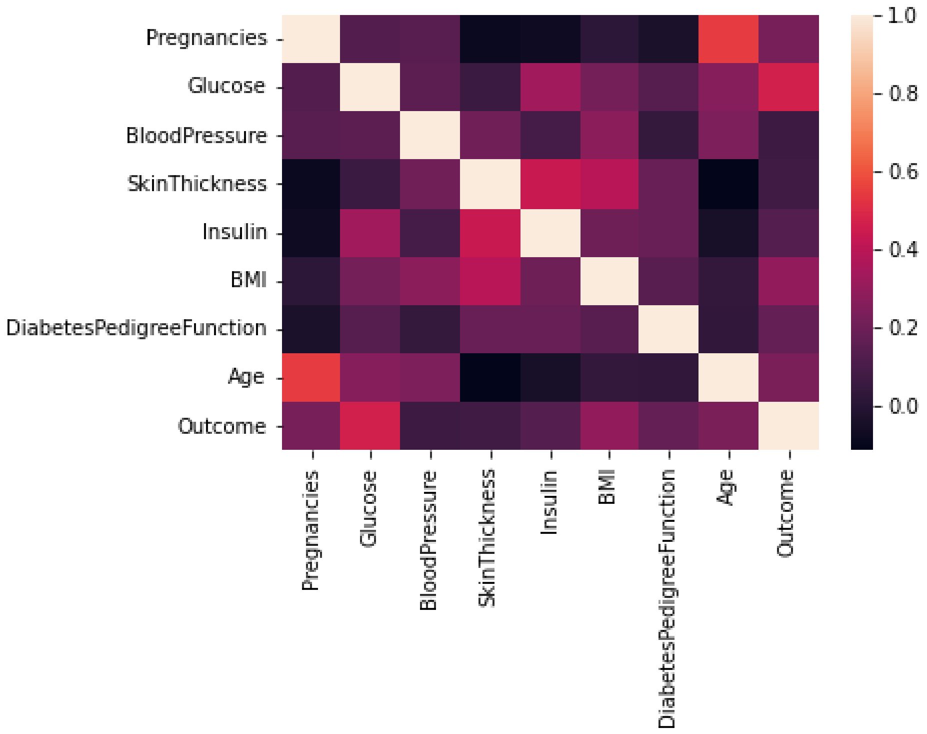
```
                          Pregnancies   Glucose  BloodPressure  SkinThickn
ess  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081
672
Glucose                      0.129459  1.000000       0.152590       0.057
328
BloodPressure                0.141282  0.152590       1.000000       0.207
371
SkinThickness               -0.081672  0.057328       0.207371       1.000
000
Insulin                     -0.073535  0.331357       0.088933       0.436
783
BMI                          0.017683  0.221071       0.281805       0.392
573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183
928
Age                          0.544341  0.263514       0.239528      -0.113
970
Outcome                      0.221898  0.466581       0.065068       0.074
752

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.073535  0.017683                 -0.033523
Glucose                    0.331357  0.221071                  0.137337
BloodPressure              0.088933  0.281805                  0.041265
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844

                                Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

In [8]:

```
sns.heatmap(correlation)
```

Out[8]:

```
<AxesSubplot:>
```



Train Test Split

In [12]:

```
X = data.drop("Outcome",axis=1)
Y = data["Outcome"]

X_train , X_test , Y_train , Y_test  = train_test_split(X,Y,test_size = 0.2)

X_train
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
430	2	99	0	0	0	22.2	
543	4	84	90	23	56	39.5	
477	7	114	76	17	110	23.8	
335	0	165	76	43	255	47.9	
127	1	118	58	36	94	33.3	
...	...	...	...	...	...	...	
261	3	141	0	0	0	30.0	
723	5	117	86	30	105	39.1	
275	2	100	70	52	57	40.5	
457	5	86	68	28	71	30.2	
265	5	96	74	18	67	33.6	

614 rows × 8 columns

Training the model

In [18]:

```
model = LogisticRegression()

model.fit(X_train,Y_train)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: 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 i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-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-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[18]:

```
LogisticRegression()
```

Making prediction

In [19]:

```
predictions = model.predict(X_test)
```

In [20]:

```
print(predictions)
```

```
[0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1
 1 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 1 0]
```

Evaluation

In [23]:

```
accuracy = accuracy_score(predictions, Y_test)
print(accuracy)
```

```
0.7792207792207793
```

In [ ]:

```

```

In [ ]:

```

```