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

In [3]:
```
df =pd.read_csv("/home/admin1/Downloads/diabetes.csv")
```

In [5]:
```
df
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
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

In [7]:
```
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
```

In [9]:
```
df.isnull().sum()
```

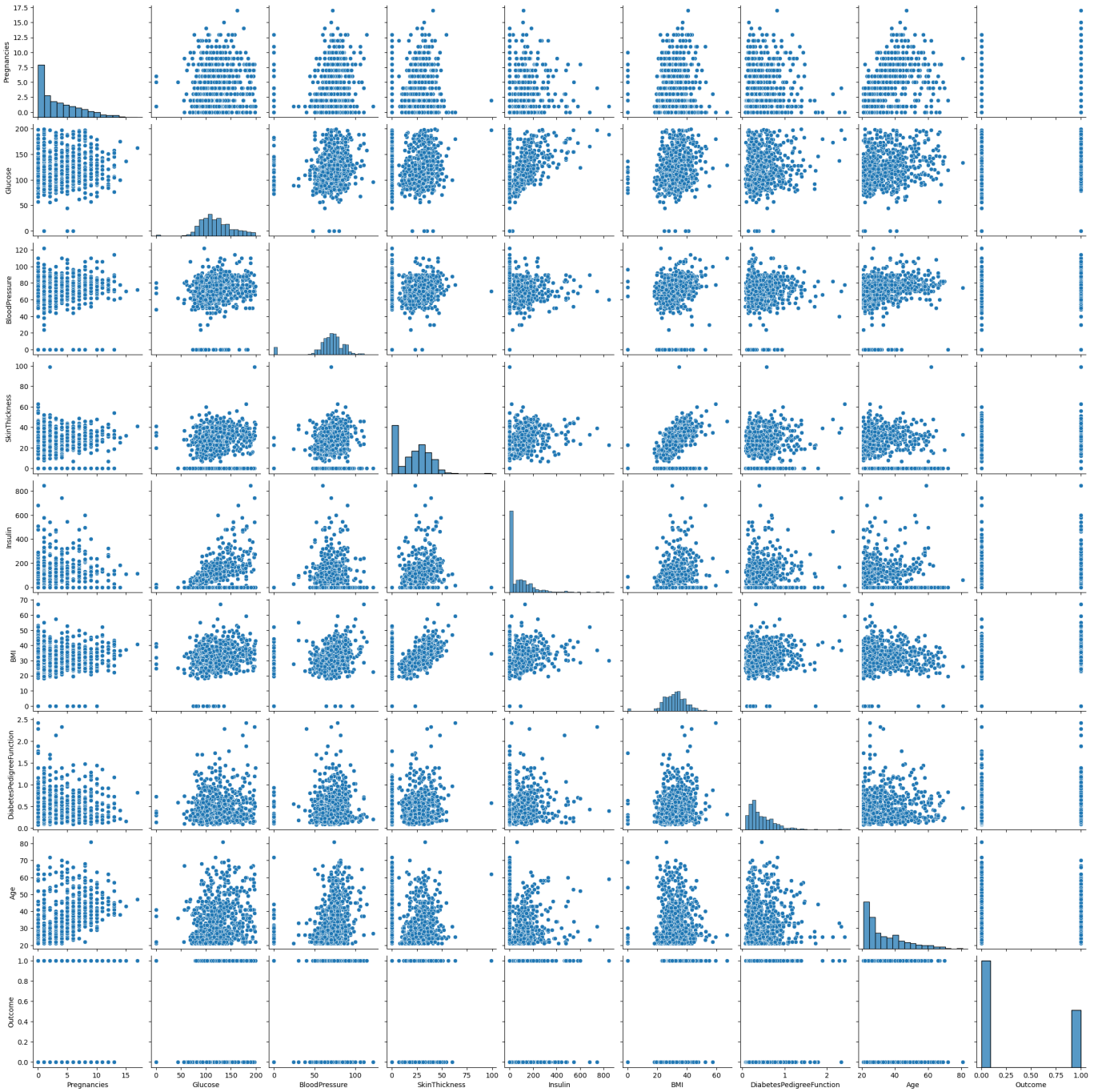
Out[9]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [11]:
```
df.isna().sum()
```

Out[11]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

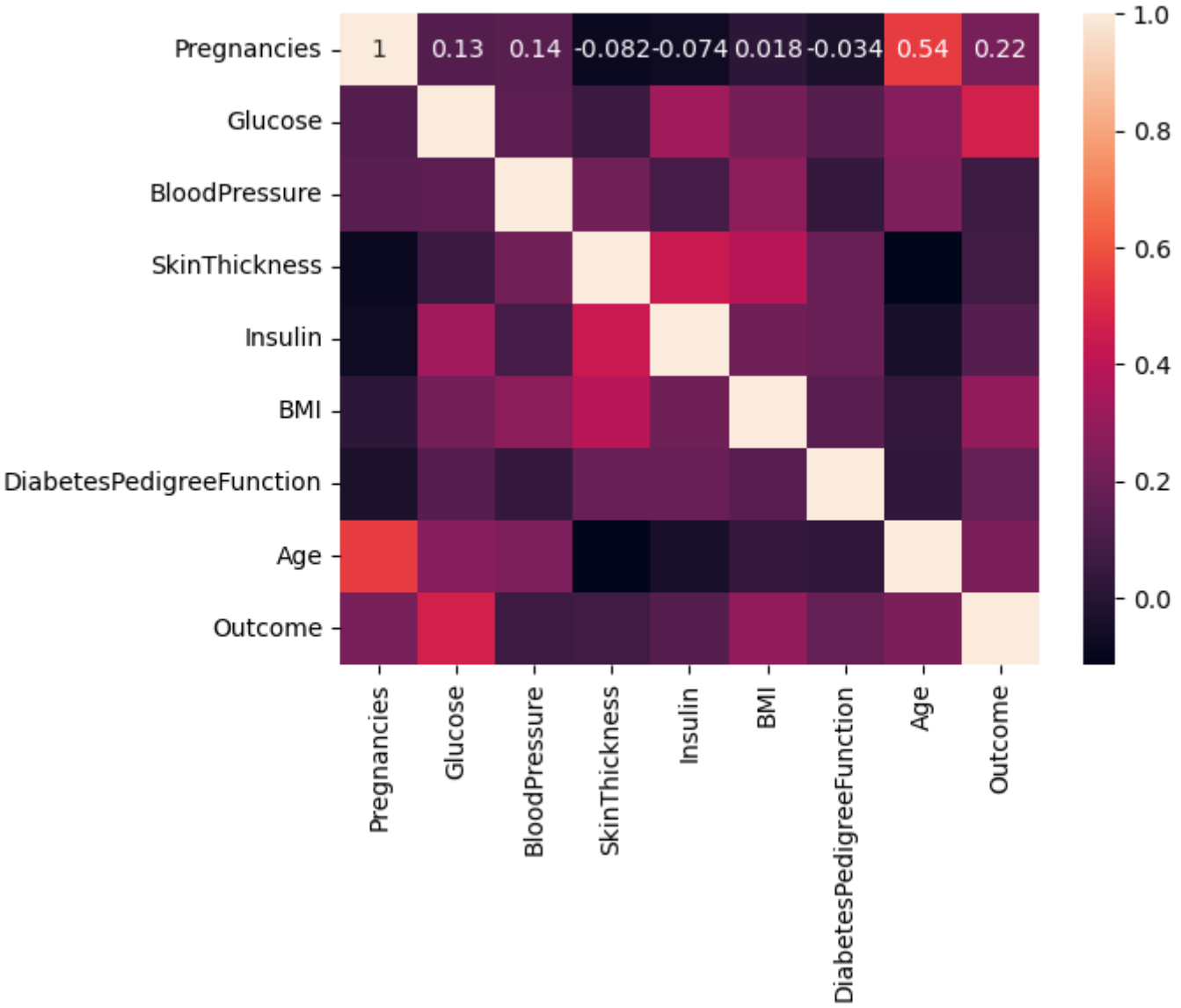
In [13]:
```
sns.pairplot(df)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x7f21569d3e20>
```



In [17]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[17]:
```
<Axes: >
```



In [19]:
```
x = df.drop("Outcome",axis=1)
y = df['Outcome']
```

In [21]:
```
x
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
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 × 8 columns

In [23]:
```
y
```

Out[23]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [58]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7)
```

In [60]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [62]:
```
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.transform(x_test)
```

Model (Logistic Regression)

In [65]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [67]:
```
model.fit(x_train_scaled,y_train)
```

Out[67]:
```
▾ LogisticRegression
LogisticRegression()
```

In [69]:
```
y_pred = model.predict(x_test_scaled)
```

In [71]:
```
model.score(x_test_scaled,y_pred)
```

Out[71]:
```
1.0
```

In [73]:
```
from sklearn.metrics import classification_report,accuracy_score,precision_score,conf
acc = accuracy_score(y_test,y_pred)
pr = precision_score(y_test,y_pred)
cr = classification_report(y_test,y_pred)
cm = confusion_matrix(y_test,y_pred)
re = recall_score(y_test,y_pred)
```

In [75]:
```
acc
```

Out[75]:
```
0.7878787878787878
```

In [77]:
```
pr
```

Out[77]:
```
0.7301587301587301
```

In [79]:
```
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       153
           1       0.73      0.59      0.65        78

    accuracy                           0.79       231
   macro avg       0.77      0.74      0.75       231
weighted avg       0.78      0.79      0.78       231
```

In [81]:
```
print(cm)
```

```
[[136  17]
 [ 32  46]]
```

In [ ]:
```

```

In [90]:
```
df = pd.read_csv("/home/admin1/Downloads/Crop_recommendation.csv")
```

In [92]:
```
df
```

Out[92]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

In [94]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

In [96]:
```
df.isnull().sum()
```

Out[96]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [100…
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [106…
```
df['label'] = le.fit_transform(df['label'])
```

In [108…
```
df
```

Out[108…

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	20
1	85	58	41	21.770462	80.319644	7.038096	226.655537	20
2	60	55	44	23.004459	82.320763	7.840207	263.964248	20
3	74	35	40	26.491096	80.158363	6.980401	242.864034	20
4	78	42	42	20.130175	81.604873	7.628473	262.717340	20
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	5
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	5
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	5
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	5
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	5

2200 rows × 8 columns

In [112…
```
x = df.drop("label",axis=1)
y = df['label']
```

In [140…
```
y.value_counts()
```

Out[140…
```
label
20    100
11    100
8     100
6     100
4     100
17    100
16    100
0     100
15    100
21    100
7     100
12    100
1     100
19    100
10    100
2     100
14    100
13    100
18    100
9     100
3     100
5     100
Name: count, dtype: int64
```

In [116…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7)
```

In [118…
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [120…
```
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.transform(x_test)
```

In [122…
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [124…
```
model.fit(x_train_scaled,y_train)
```

Out[124…
▾ LogisticRegression

LogisticRegression()

In [126…
```
y_pred = model.predict(x_test_scaled)
```

In [128…
```
model.score(x_test_scaled,y_pred)
```

Out[128…
```
1.0
```

In [132…
```
from sklearn.metrics import classification_report,accuracy_score,precision_score,conf
acc = accuracy_score(y_test,y_pred,)
pr = precision_score(y_test,y_pred,average='micro')
cr = classification_report(y_test,y_pred)
cm = confusion_matrix(y_test,y_pred)
re = recall_score(y_test,y_pred,average='micro')
```

In [134…
```
acc
```

Out[134…
```
0.9757575757575757
```

In [136…
```
pr
```

Out[136… 0.9757575757575757

In [138…
```
print(cr)
```

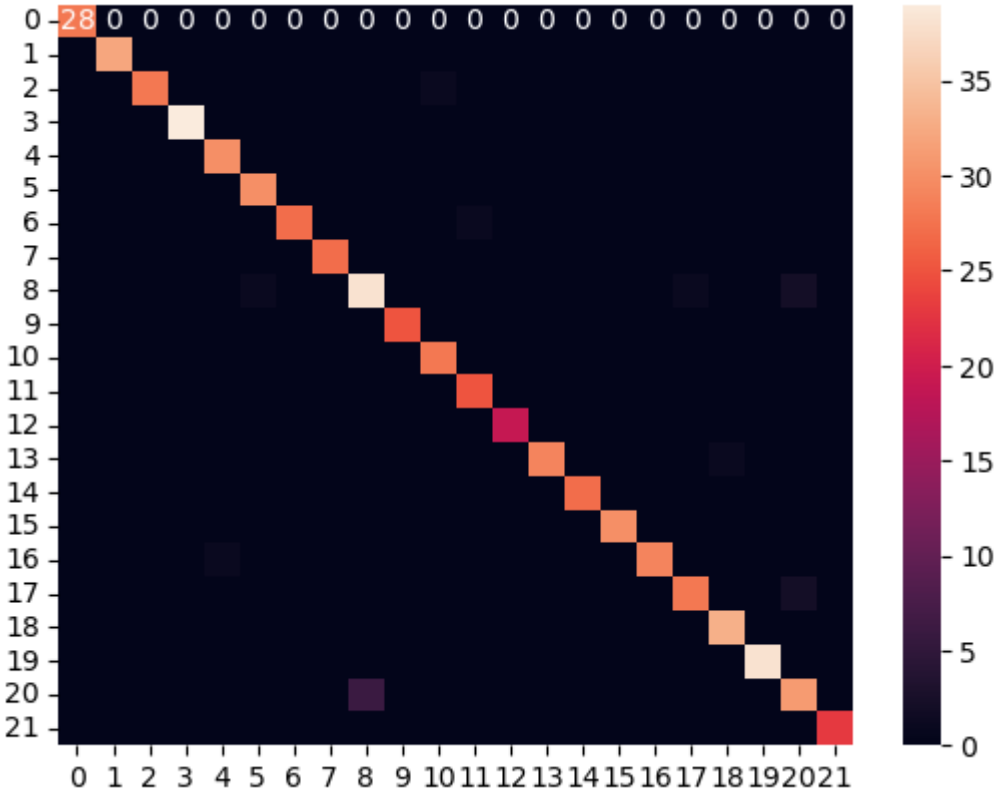
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        28
           1       1.00      1.00      1.00        32
           2       1.00      0.97      0.98        29
           3       1.00      1.00      1.00        39
           4       0.97      1.00      0.98        30
           5       0.97      1.00      0.98        30
           6       1.00      0.96      0.98        28
           7       1.00      1.00      1.00        27
           8       0.86      0.90      0.88        42
           9       1.00      1.00      1.00        25
          10       0.97      1.00      0.98        28
          11       0.96      1.00      0.98        25
          12       1.00      1.00      1.00        19
          13       1.00      0.97      0.98        30
          14       1.00      1.00      1.00        27
          15       1.00      1.00      1.00        30
          16       1.00      0.97      0.98        30
          17       0.97      0.93      0.95        30
          18       0.97      1.00      0.99        33
          19       1.00      1.00      1.00        38
          20       0.89      0.84      0.86        37
          21       1.00      1.00      1.00        23

    accuracy                           0.98       660
   macro avg       0.98      0.98      0.98       660
weighted avg       0.98      0.98      0.98       660
```

In [142…
```
print(cm)
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 28  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 39  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 27  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 27  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0 38  0  0  0  0  0  0  0  0  1  0  0  2  0]
 [ 0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 28  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 30  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 28  0  0  2  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 33  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 38  0  0]
 [ 0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0 31  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23]]
```

In [148…
```
sns.heatmap(cm,annot=True)
```

Out[148… <Axes: >


0 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
35
30
25
20
15
10
5
0

In [ ]: