In [2]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('diabetes.csv')
df.head()
```

Out[2]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesF |
|---|---|---|---|---|---|---|---|
| **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 | |

In [3]:
```
df.info()
print(df.isnull().sum())
```

```
<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
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

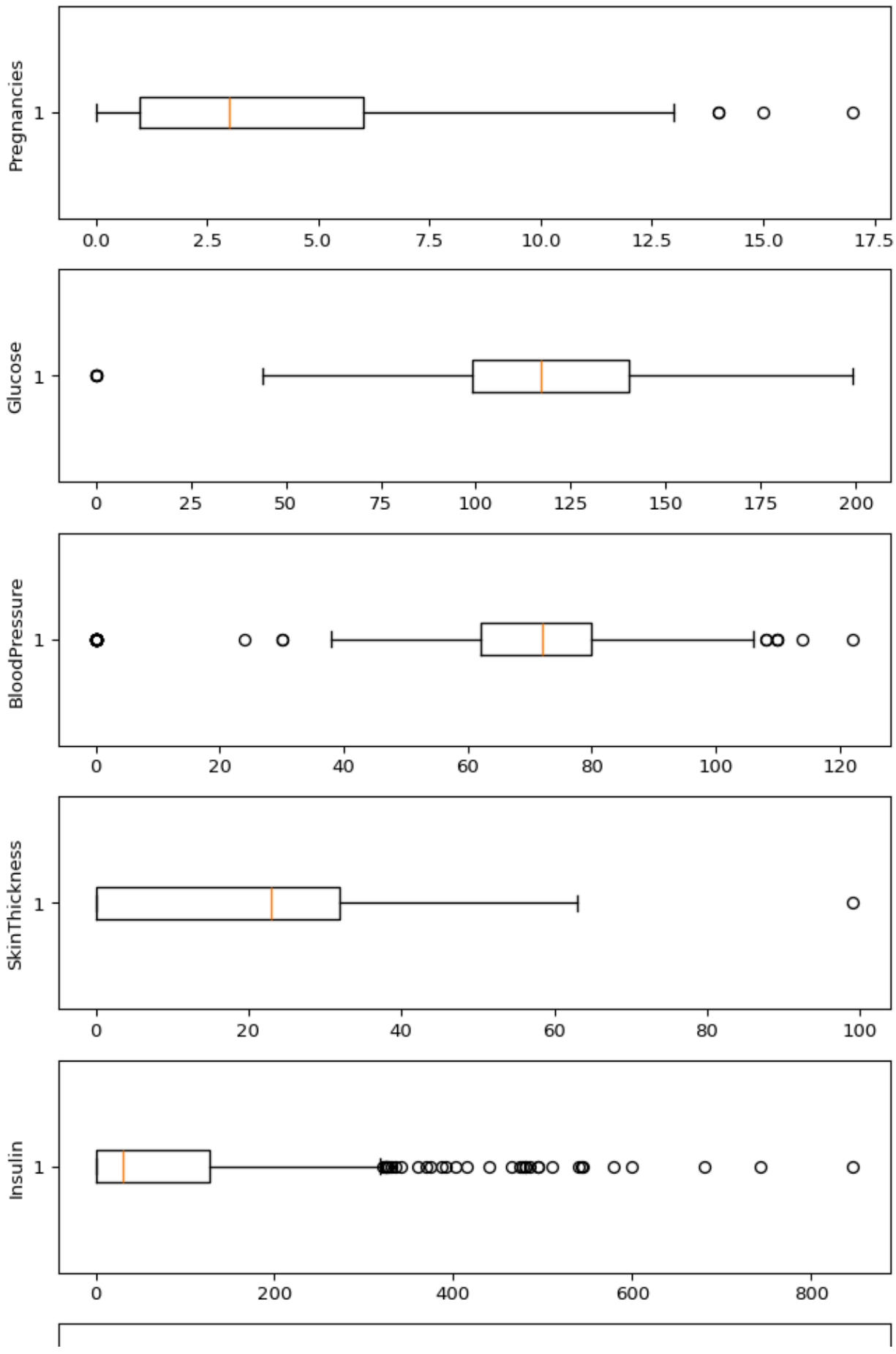
In [4]:
```
df.describe()
```

Out[4]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | |
|---|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768. |
| **mean** | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31. |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7. |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27. |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32. |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36. |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67. |

In [5]:

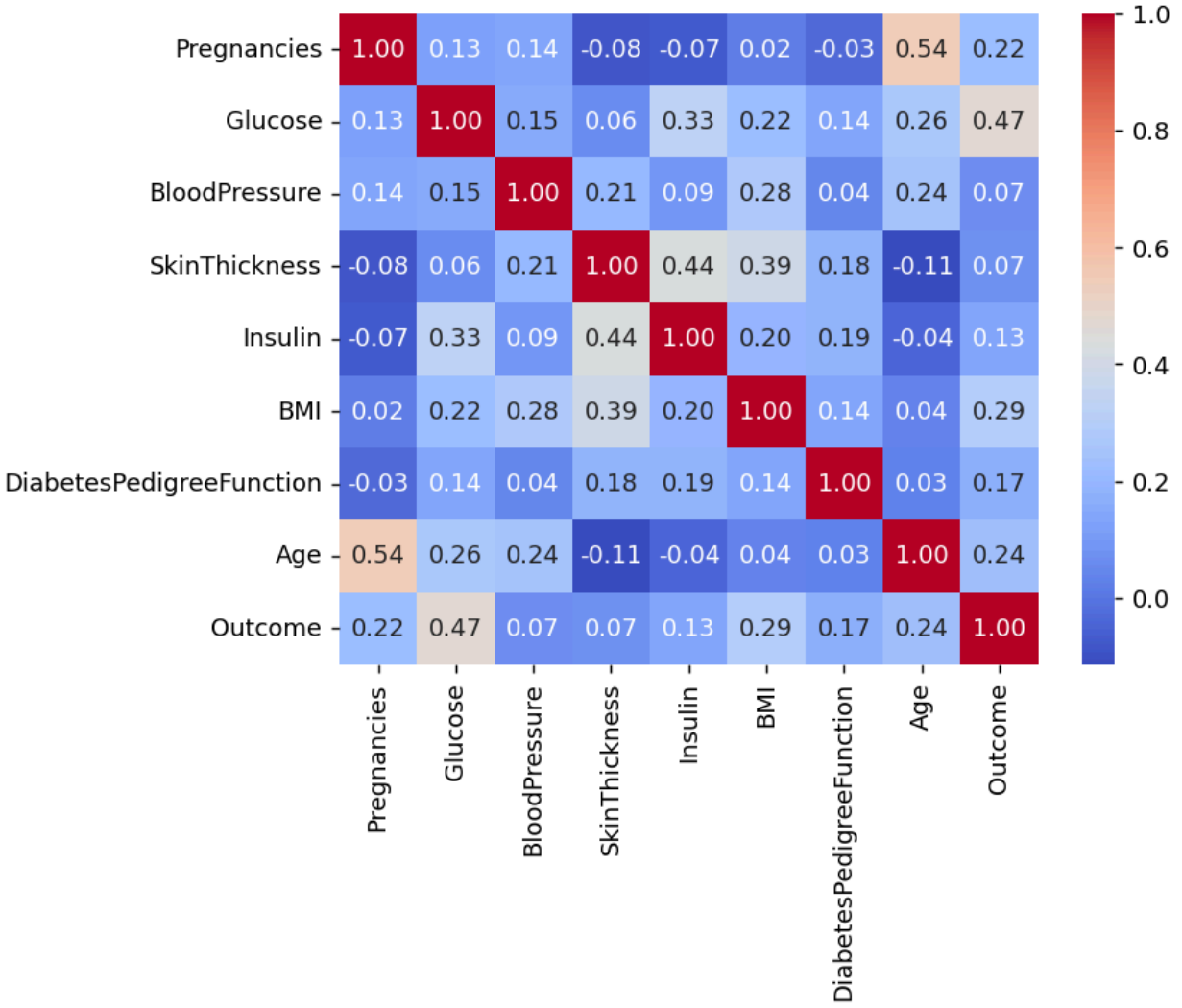
```
fig, axs = plt.subplots(len(df.columns), 1, figsize=(7, 18), dpi=95)

for i, col in enumerate(df.columns):
    axs[i].boxplot(df[col], vert=False)
    axs[i].set_ylabel(col)

plt.tight_layout()
plt.show()
```


Pregnancies
1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Glucose
1
0
25
50
75
100
125
150
175
200
BloodPressure
1
0
20
40
60
80
100
120
SkinThickness
1
0
20
40
60
80
100
Insulin
1
0
200
400
600
800

In [6]:
```
q1, q3 = np.percentile(df['Insulin'], [25, 75])
iqr = q3 - q1

lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr

clean_df = df[(df['Insulin'] >= lower) & (df['Insulin'] <= upper)]

print("Before:", df.shape)
print("After:", clean_df.shape)
```

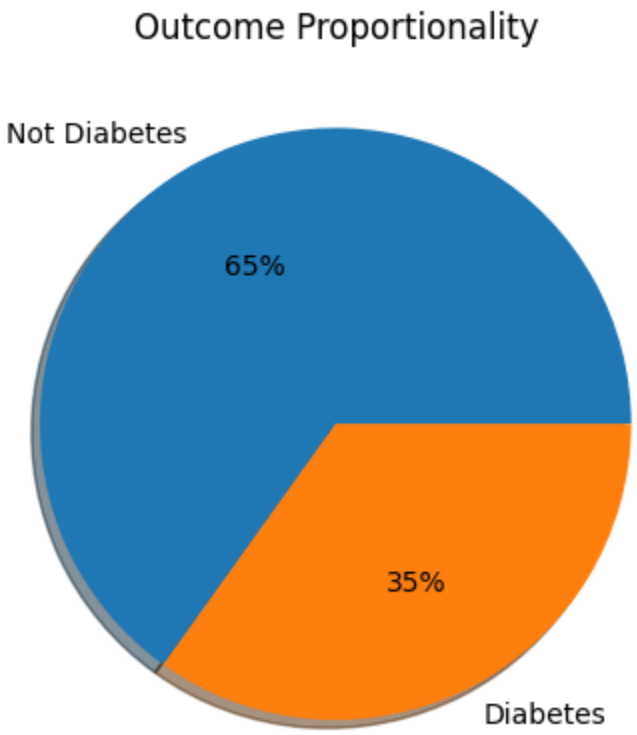
```
Before: (768, 9)
After: (734, 9)
```

In [7]:
```
corr = df.corr()

plt.figure(dpi=130)
sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')
plt.show()

print(corr['Outcome'].sort_values(ascending=False))
```



```
Outcome                     1.000000
Glucose                     0.466581
BMI                         0.292695
Age                         0.238356
Pregnancies                 0.221898
DiabetesPedigreeFunction    0.173844
Insulin                     0.130548
SkinThickness               0.074752
BloodPressure               0.065068
Name: Outcome, dtype: float64
```

In [8]:
```
plt.pie(
    df['Outcome'].value_counts(),
    labels=['Not Diabetes', 'Diabetes'],
    autopct='%.f%%',
    shadow=True
)
plt.title('Outcome Proportionality')
plt.show()
```




In [9]:
```
X = df.drop(columns=['Outcome'])
y = df['Outcome']

print(X.shape, y.shape)
```

```
(768, 8) (768,)
```

In [10]:
```
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)
```

```
print(X_normalized[:5])
```

```
[[0.35294118 0.74371859 0.59016393 0.35353535 0.         0.50074516
  0.23441503 0.48333333]
 [0.05882353 0.42713568 0.54098361 0.29292929 0.         0.39642325
  0.11656704 0.16666667]
 [0.47058824 0.91959799 0.52459016 0.         0.         0.34724292
  0.25362938 0.18333333]
 [0.05882353 0.44723618 0.54098361 0.23232323 0.11111111 0.41877794
  0.03800171 0.        ]
 [0.         0.68844221 0.32786885 0.35353535 0.19858156 0.64232489
  0.94363792 0.2       ]]
```

In [11]:
```
scaler = StandardScaler()
X_standardized = scaler.fit_transform(X)

print(X_standardized[:5])
```

```
[[ 0.63994726  0.84832379  0.14964075  0.90726993 -0.69289057  0.20401277
   0.46849198  1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575  0.53090156 -0.69289057 -0.68442195
  -0.36506078 -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 -1.28821221 -0.69289057 -1.10325546
   0.60439732 -0.10558415]
 [-0.84488505 -0.99820778 -0.16054575  0.15453319  0.12330164 -0.49404308
  -0.92076261 -1.04154944]
 [-1.14185152  0.5040552  -1.50468724  0.90726993  0.76583594  1.4097456
   5.4849091  -0.0204964 ]]
```