In [27]:
```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as p
```

In [32]:
```
df =pd.read_csv('D:\\Whatsapp\\diabetes.csv')
```

In [40]:
```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]]
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

In [49]:
```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns wh
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or consta
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolo
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

In [53]:
```
# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only numerical colum
    # Remove rows and columns that would lead to df being singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns wh
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inv
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plot
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xy
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

In [58]:
```
nRowsRead = 1000  # specify 'None' if you want to read the whole file
df = pd.read_csv('D:\\Whatsapp\\diabetes.csv', delimiter=',', nrows=nRowsRe
df.dataframeName = 'diabetes.csv'
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 768 rows and 9 columns
```

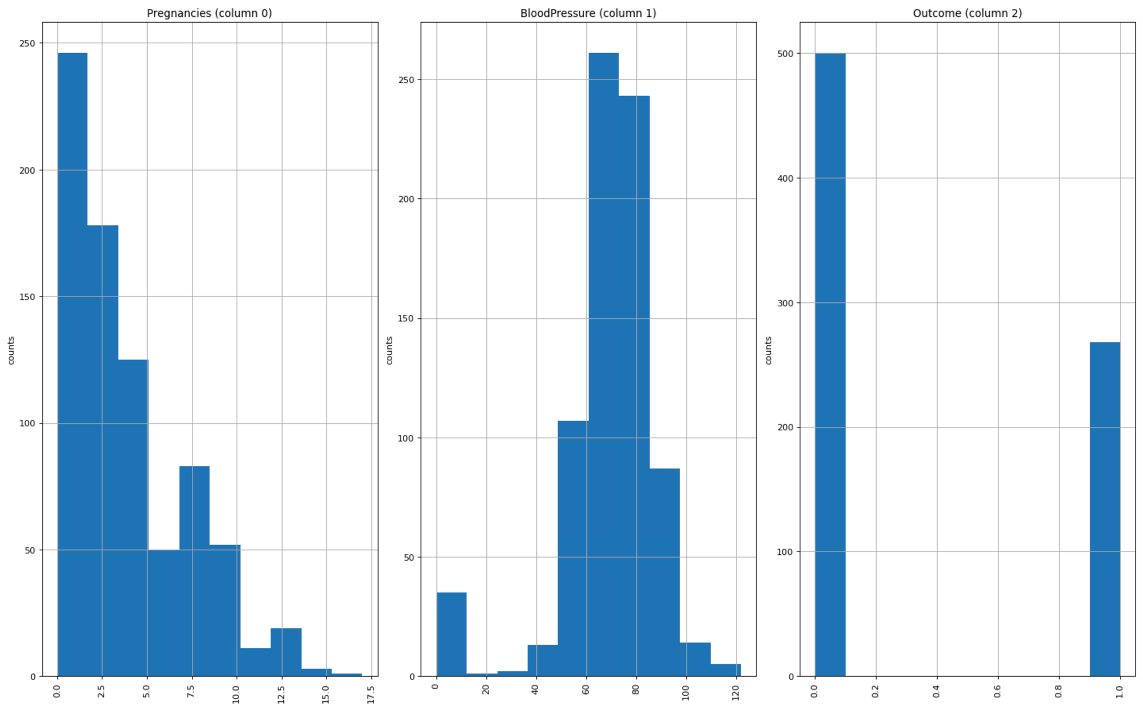
In [59]:
```
df.head(5)
```

Out[59]:

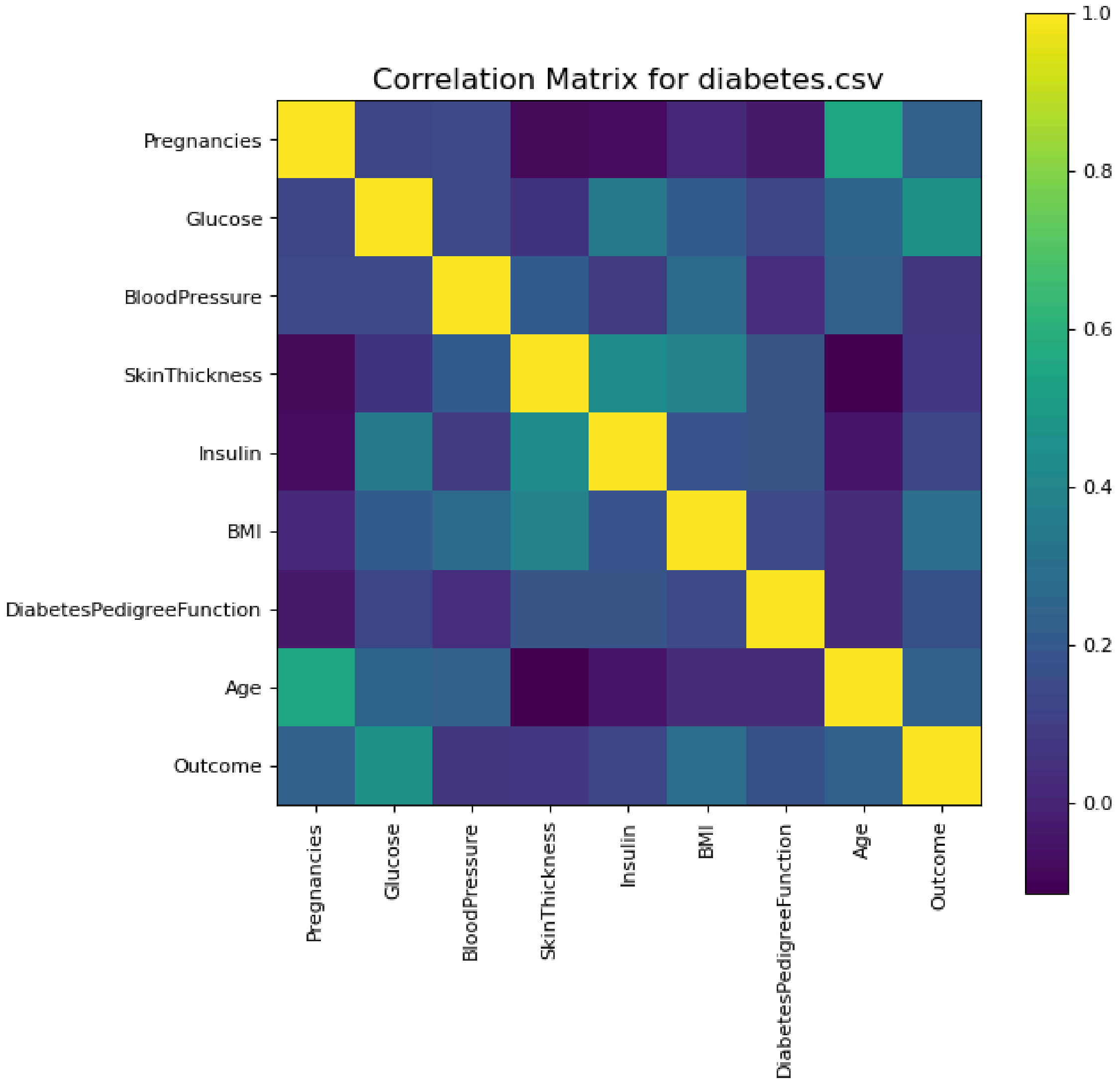
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

```
In [60]: plotPerColumnDistribution(df, 10, 5)
```

```
<ipython-input-40-0121bf3d9d74>:10: MatplotlibDeprecationWarning: Passing
non-integers as three-element position specification is deprecated since
3.3 and will be removed two minor releases later.
  plt.subplot(nGraphRow, nGraphPerRow, i + 1)
```



```
In [62]: plotCorrelationMatrix(df, 8)
```

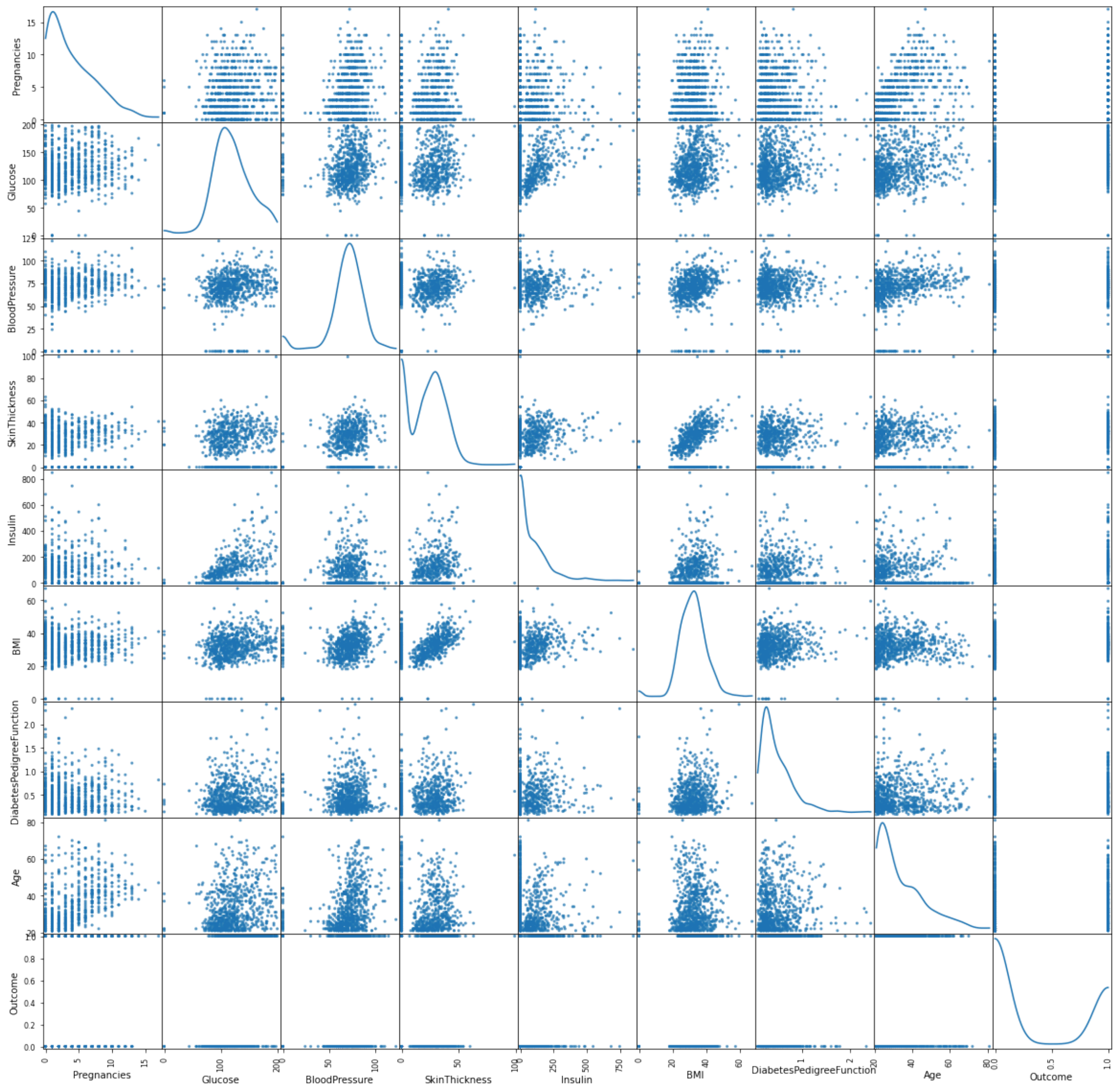


In [63]:
```
plotScatterMatrix(df, 20, 10)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-63-410071f72505> in <module>
----> 1 plotScatterMatrix(df, 20, 10)

<ipython-input-53-c769cd30a2be> in plotScatterMatrix(df, plotSize, textSize)
     12     corrs = df.corr().values
     13     for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
---> 14         a[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)
     15     plt.suptitle('Scatter and Density Plot')
     16     plt.show()

NameError: name 'a' is not defined
```



In [ ]: