In [ ]:
```
Navneeth Krishna(21047)
```

Diabetes

1.Data Frame

In [45]:
```
import numpy
import pandas
myarray = numpy.array([[1,2,3],[4,5,6]])
rownames = ['a','b']
colnames=['f1','f2','f3']
mydataframe = pandas.DataFrame(myarray, index = rownames, columns=colnames)
print(mydataframe)
```

```
   f1  f2  f3
a   1   2   3
b   4   5   6
```

Changed data

In [46]:
```
import numpy
import pandas
myarray = numpy.array([['a','sandhya',9.6],[4,'shreya',6.5]])
rownames = ['r1','r2']
colnames=['f1','f2','f3']
mydataframe = pandas.DataFrame(myarray, index = rownames, columns=colnames)
print(mydataframe)
```

```
   f1       f2   f3
r1  a  sandhya  9.6
r2  4   shreya  6.5
```

2.Loading CSV file

In [47]:
```
from pandas import read_csv
input='diabetes.csv'
data=read_csv(input)
```

3.Printing statistical summary

a.

In [48]:
```
description = data.describe()
print(description)
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

b. printing size of the matrix

In [49]:
```
print(data.shape)
```

```
(768, 9)
```

c.Peek at data

In [50]:
```
print(data.head(4))
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
```

Group on the basis of a particular attribute

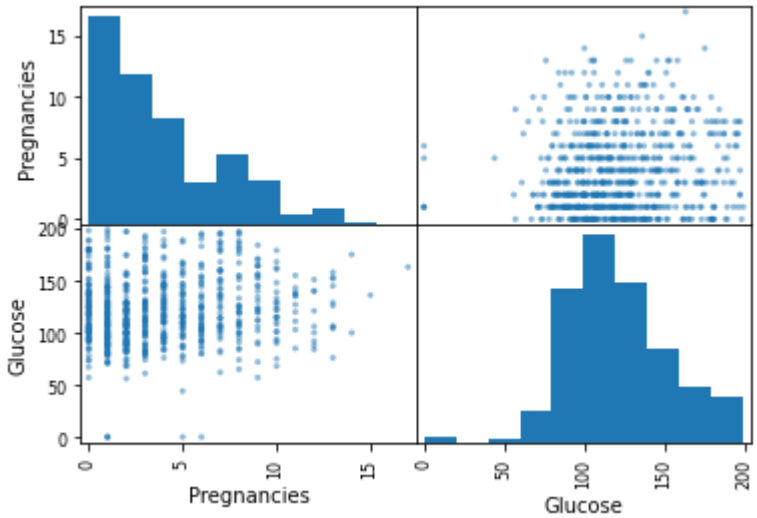
In [51]:
```
print(data.groupby('Outcome').size())
```

```
Outcome
0    500
1    268
dtype: int64
```

4. Data visualization

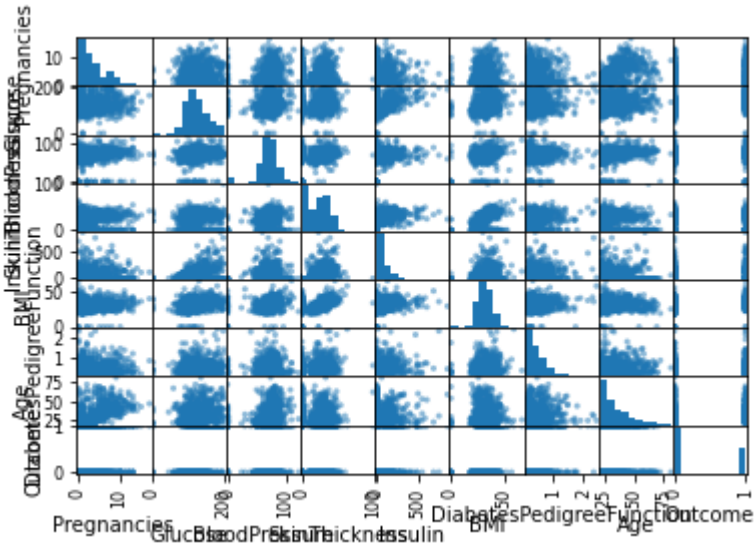
plotting certain attributes

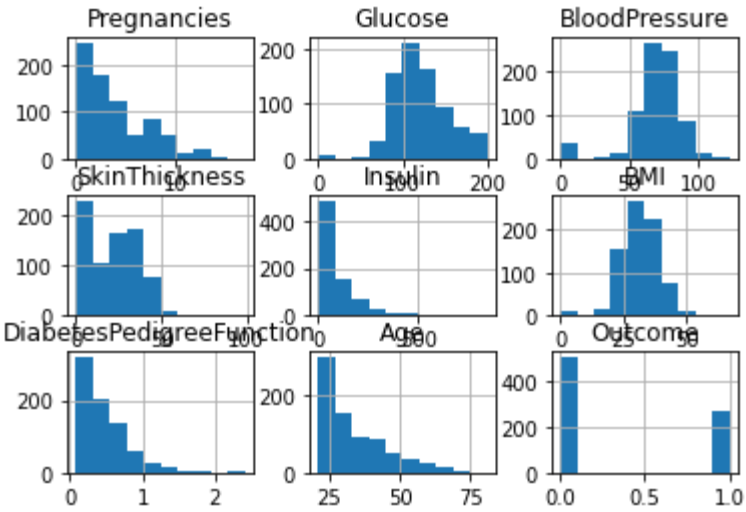
In [52]:
```
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
scatter_matrix(data[['Pregnancies','Glucose']])
plt.show()
```



plotting all attributes

In [53]:
```
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
scatter_matrix(data)      #scatter plot
plt.show()
data.hist()                        #histogram
plt.show()
```





5. Standardization of dataset

In [54]:
```
from sklearn.preprocessing import StandardScaler
import pandas
import numpy
arr=data.values                              #convert data frame to array
X=arr[:,0:8]                                 #split columnsY=arr[:,8]
scaler=StandardScaler().fit(X)               #fit data for standardizationrescaled
X=scaler.transform(X)                        #convert the data as per (x-μ)/σ
numpy.set_printoptions(precision=3)
print(rescaledX[0:2,:])
print(X[0:2,:])
```

```
[[ 1.270e+00  9.844e-01  1.568e+00  3.284e+00  2.653e+00  2.532e+00
   2.218e+00  2.256e+00  2.490e+00 -5.653e-01  2.833e+00  2.488e+00
  -2.140e-01  1.317e+00  7.240e-01  6.608e-01]
 [ 1.686e+00  1.909e+00 -8.270e-01 -4.871e-01 -2.385e-02  5.481e-01
   1.392e-03 -8.687e-01  4.993e-01 -8.762e-01  2.633e-01  7.424e-01
  -6.054e-01 -6.929e-01 -4.408e-01  2.602e-01]]
[[ 0.64   0.848  0.15   0.907 -0.693  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161  0.531 -0.693 -0.684 -0.365 -0.191]]
```

6. Normalizing a column in pandas

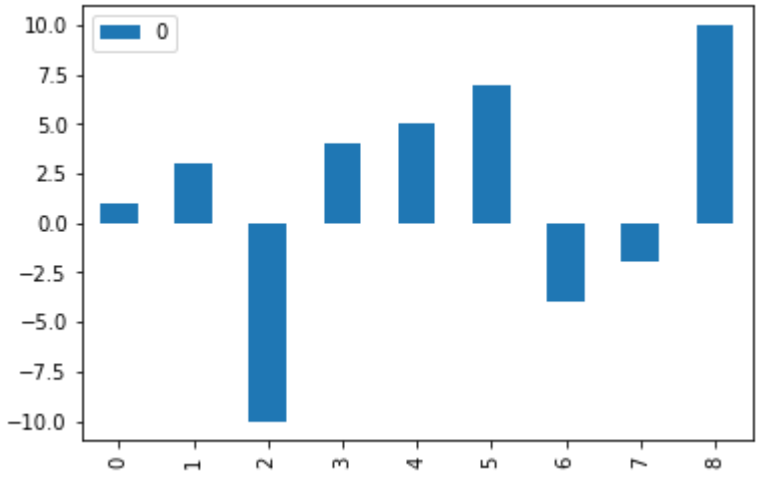
Create a dataframe for a set of values in an array

In [55]:
```
myarray=numpy.array([1,3,-10,4,5,7,-4,-2,10])
mydataframe = pandas.DataFrame(myarray)
print(mydataframe)
```

```
    0
0   1
1   3
2 -10
3   4
4   5
5   7
6  -4
7  -2
8  10
```

plot the data

In [57]:
```
mydataframe.plot(kind='bar')
plt.show()
```



Breast cancer

2.Loading CSV file

In [58]:
```
from pandas import read_csv
input='data.csv'
data=read_csv(input)
```

3.Printing statistical summary

a.

In [59]:
```
description = data.describe()
print(description)
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
count	569.000000	569.000000	569.000000	569.000000
mean	0.096360	0.104341	0.088799	0.048919
std	0.014064	0.052813	0.079720	0.038803
min	0.052630	0.019380	0.000000	0.000000
25%	0.086370	0.064920	0.029560	0.020310
50%	0.095870	0.092630	0.061540	0.033500
75%	0.105300	0.130400	0.130700	0.074000
max	0.163400	0.345400	0.426800	0.201200

	symmetry_mean	...	texture_worst	perimeter_worst	area_worst
count	569.000000	...	569.000000	569.000000	569.000000
mean	0.181162	...	25.677223	107.261213	880.583128
std	0.027414	...	6.146258	33.602542	569.356993
min	0.106000	...	12.020000	50.410000	185.200000
25%	0.161900	...	21.080000	84.110000	515.300000
50%	0.179200	...	25.410000	97.660000	686.500000
75%	0.195700	...	29.720000	125.400000	1084.000000
max	0.304000	...	49.540000	251.200000	4254.000000

	smoothness_worst	compactness_worst	concavity_worst
count	569.000000	569.000000	569.000000
mean	0.132369	0.254265	0.272188
std	0.022832	0.157336	0.208624
min	0.071170	0.027290	0.000000
25%	0.116600	0.147200	0.114500
50%	0.131300	0.211900	0.226700
75%	0.146000	0.339100	0.382900
max	0.222600	1.058000	1.252000

	concave points_worst	symmetry_worst	fractal_dimension_worst
count	569.000000	569.000000	569.000000
mean	0.114606	0.290076	0.083946
std	0.065732	0.061867	0.018061
min	0.000000	0.156500	0.055040
25%	0.064930	0.250400	0.071460
50%	0.099930	0.282200	0.080040
75%	0.161400	0.317900	0.092080
max	0.291000	0.663800	0.207500

	Unnamed: 32
count	0.0
mean	NaN
std	NaN
min	NaN
25%	NaN

```
50%            NaN
75%            NaN
max            NaN

[8 rows x 32 columns]
```

b. printing size of the matrix

In [61]:
```
print(data.shape)
```

```
(569, 33)
```

c.Printing the top 4 entries

In [62]:
```
print(data.head(4))
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520

   ...  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622
1  ...          23.41           158.80      1956.0            0.1238
2  ...          25.53           152.50      1709.0            0.1444
3  ...          26.50            98.87       567.7            0.2098

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN

[4 rows x 33 columns]
```

d.grouping of data based on attribute

In [63]:
```
print(data.groupby("symmetry_se").size())
```

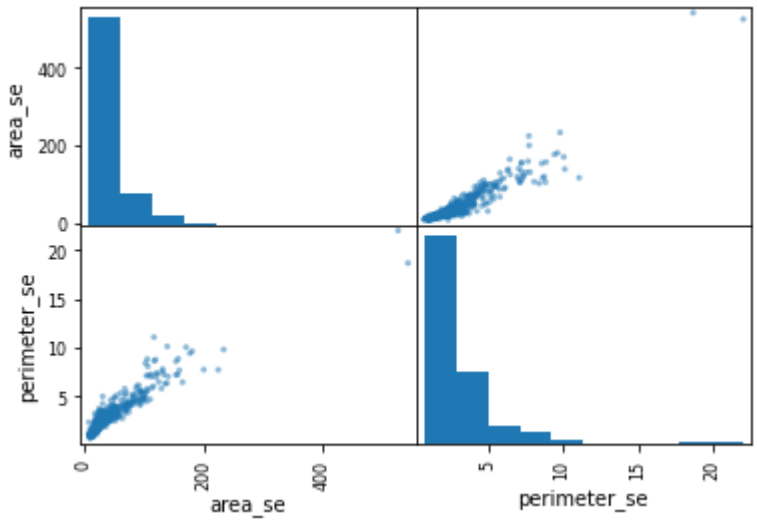
```
symmetry_se
0.007882    1
0.009539    1
0.009947    1
0.010130    1
0.010290    1
           ..
0.055430    1
0.056280    1
0.059630    1
0.061460    1
0.078950    1
Length: 498, dtype: int64
```

4.Data visualization

In [64]:
```
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
```

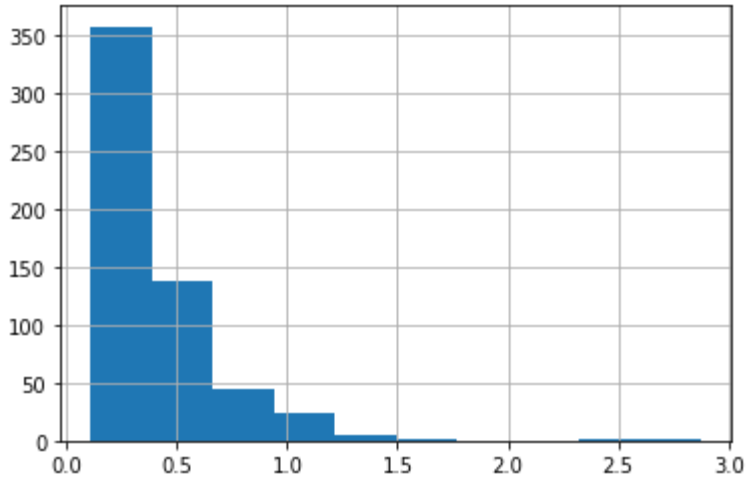
scatter plot

In [65]:
```
scatter_matrix(data[['area_se','perimeter_se']])
plt.show()
```



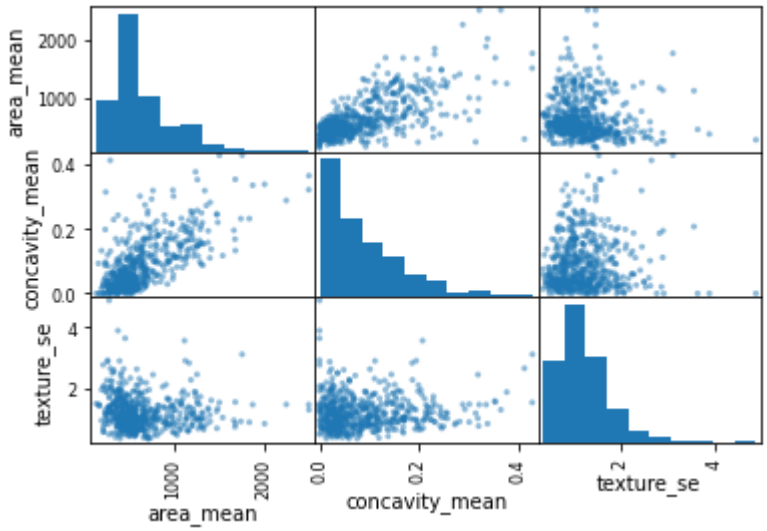
histogram

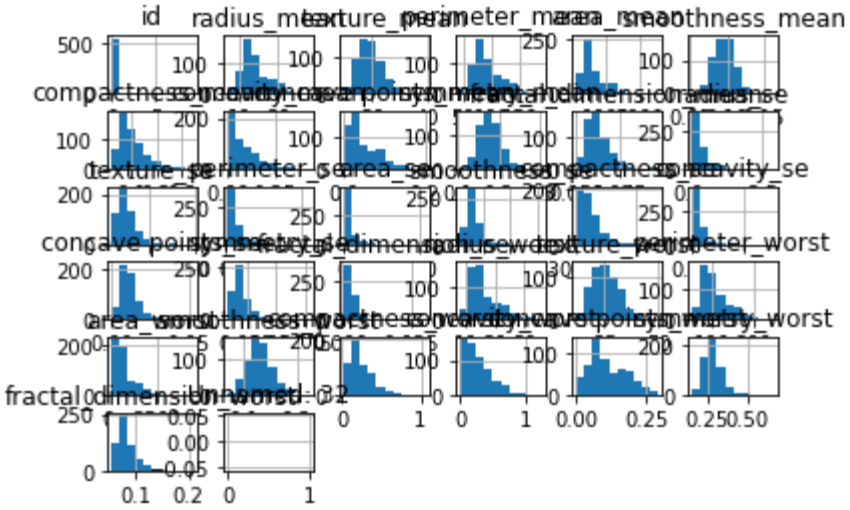
In [66]:
```
data['radius_se'].hist()
plt.show()
```



Plotting all the attributes

In [67]:
```
import seaborn as sns
#scatter_matrix(data)
#plt.show()
scatter_matrix(data[['area_mean','concavity_mean','texture_se']])
data.hist()
plt.show()
```





5.Standardizing dataset

In [68]:
```
from sklearn.preprocessing import StandardScaler

import pandas
import numpy
#converting data to array
arr=data.values
#splitting columns
X=arr[:,4:20]
#fitting data for standardisation
scaler=StandardScaler().fit(X)
#converting the data
rescaledX=scaler.transform(X)

numpy.set_printoptions(precision=3)
print(rescaledX[0:2,:])
print(X[0:2,:])
```

```
[[ 1.270e+00  9.844e-01  1.568e+00  3.284e+00  2.653e+00  2.532e+00
   2.218e+00  2.256e+00  2.490e+00 -5.653e-01  2.833e+00  2.488e+00
  -2.140e-01  1.317e+00  7.240e-01  6.608e-01]
 [ 1.686e+00  1.909e+00 -8.270e-01 -4.871e-01 -2.385e-02  5.481e-01
   1.392e-03 -8.687e-01  4.993e-01 -8.762e-01  2.633e-01  7.424e-01
  -6.054e-01 -6.929e-01 -4.408e-01  2.602e-01]]
[[122.8 1001.0 0.1184 0.2776 0.3001 0.1471 0.2419 0.07871 1.095 0.9053
  8.589 153.4 0.006399 0.04904 0.05373 0.01587]
 [132.9 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812 0.05667 0.5435
  0.7339 3.398 74.08 0.005225 0.01308 0.0186 0.0134]]
```

6.Normalizing column in pandas

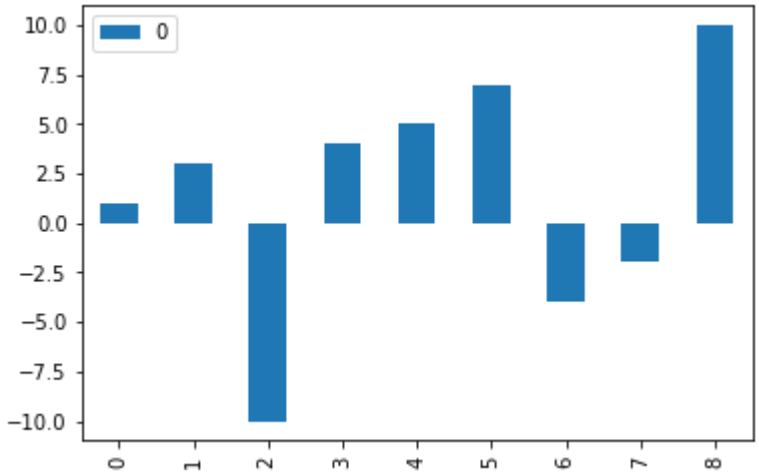
creating dataframe for a set of values in an array

In [69]:
```
myarray=numpy.array([1,3,-10,4,5,7,-4,-2,10])
mydataframe = pandas.DataFrame(myarray)
print(mydataframe)
```

```
    0
0   1
1   3
2 -10
3   4
4   5
5   7
6  -4
7  -2
8  10
```

plotting the above data

In [70]:
```
mydataframe.plot(kind='bar')
plt.show()
```



Question : Identify the difference in the standardization and normalization of data.

Normalization is a scaling technique that completely shifts the values and rescales it in such a way that its range is from 0 to 1. On the other hand, standardization is another scaling technique wherein the mean of all the data becomes 0 and the other data is shifted to such a position that they have a standard deviation of 1 unit.