In [ ]:
```
#Harsh Arora
#AE-1218
#BSc.(Hons.)Computer Science
```

In [27]:
```
import pandas as pd
#Loading the Dataset
CALORIES_DATA = pd.read_csv(r"C:\Users\MS\Downloads\Calories Set.csv")
```

In [5]:
```
# To Implement  The descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
# To Print Count
print('Count is\n',CALORIES_DATA.count(),'\n')
#To Print Mean
print('mean is\n',CALORIES_DATA.mean(),'\n')
#To Print Max
print('max is\n',CALORIES_DATA.max(),'\n')
#To Print Min
print('min is\n',CALORIES_DATA.min(),'\n')
#To Print Standard Deviation
print('standard deviation is\n',CALORIES_DATA.std(),'\n')

#print(Calories.describe()) #can also use to get the required output
```

```
Count is
 Duration    32
Date         31
Pulse        32
Maxpulse     32
Calories     30
dtype: int64

mean is
 Duration     68.4375
Pulse        103.5000
Maxpulse     128.5000
Calories     304.6800
dtype: float64

max is
 Duration    450.0
Pulse        130.0
Maxpulse     175.0
Calories     479.0
dtype: float64

min is
 Duration     30.0
Pulse         90.0
Maxpulse     101.0
Calories     195.1
dtype: float64

standard deviation is
 Duration    70.039591
Pulse         7.832933
Maxpulse     12.998759
Calories     66.003779
dtype: float64
```

In [8]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',CALORIES_DATA.isnull(),'\n')
CALORIES_DATA['Date'].fillna('2020/12/04',inplace=True)
CALORIES_DATA['Duration'].fillna('60',inplace=True)
CALORIES_DATA['Pulse'].fillna('100',inplace=True)
CALORIES_DATA['Maxpulse'].fillna('110',inplace=True)
CALORIES_DATA['Calories'].fillna('300',inplace=True)
print('updated dataset',CALORIES_DATA)
```

```
checking null values
     Duration   Date  Pulse  Maxpulse  Calories
0      False  False  False     False     False
1      False  False  False     False     False
2      False  False  False     False     False
3      False  False  False     False     False
4      False  False  False     False     False
5      False  False  False     False     False
6      False  False  False     False     False
7      False  False  False     False     False
8      False  False  False     False     False
9      False  False  False     False     False
10     False  False  False     False     False
11     False  False  False     False     False
12     False  False  False     False     False
13     False  False  False     False     False
14     False  False  False     False     False
15     False  False  False     False     False
16     False  False  False     False     False
17     False  False  False     False     False
18     False  False  False     False      True
19     False  False  False     False     False
20     False  False  False     False     False
21     False  False  False     False     False
22     False   True  False     False     False
23     False  False  False     False     False
24     False  False  False     False     False
25     False  False  False     False     False
26     False  False  False     False     False
27     False  False  False     False     False
28     False  False  False     False      True
29     False  False  False     False     False
30     False  False  False     False     False
31     False  False  False     False     False

updated dataset     Duration          Date  Pulse  Maxpulse Calories
0          60  '2020/12/01'    110       130    409.1
1          60  '2020/12/02'    117       145      479
2          60  '2020/12/03'    103       135      340
3          45  '2020/12/04'    109       175    282.4
4          45  '2020/12/05'    117       148      406
5          60  '2020/12/06'    102       127      300
6          60  '2020/12/07'    110       136      374
7         450  '2020/12/08'    104       134    253.3
8          30  '2020/12/09'    109       133    195.1
9          60  '2020/12/10'     98       124      269
10         60  '2020/12/11'    103       147    329.3
11         60  '2020/12/12'    100       120    250.7
12         60  '2020/12/12'    100       120    250.7
13         60  '2020/12/13'    106       128    345.3
14         60  '2020/12/14'    104       132    379.3
15         60  '2020/12/15'     98       123      275
16         60  '2020/12/16'     98       120    215.2
17         60  '2020/12/17'    100       120      300
18         45  '2020/12/18'     90       112      300
19         60  '2020/12/19'    103       123      323
20         45  '2020/12/20'     97       125      243
21         60  '2020/12/21'    108       131    364.2
22         45    2020/12/04    100       119      282
23         60  '2020/12/23'    130       101      300
24         45  '2020/12/24'    105       132      246
25         60  '2020/12/25'    102       126    334.5
26         60      20201226    100       120      250
27         60  '2020/12/27'     92       118      241
28         60  '2020/12/28'    103       132      300
29         60  '2020/12/29'    100       132      280
30         60  '2020/12/30'    102       129    380.3
31         60  '2020/12/31'     92       115      243
```

In [13]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',CALORIES_DATA.duplicated())
print()
CALORIES_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(CALORIES_DATA)
```

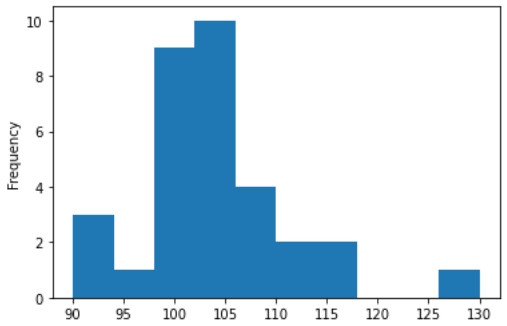
```
checking duplicate values
 0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
dtype: bool

dataset after removing duplicate values

    Duration          Date  Pulse  Maxpulse Calories
0         60  '2020/12/01'    110       130    409.1
1         60  '2020/12/02'    117       145      479
2         60  '2020/12/03'    103       135      340
3         45  '2020/12/04'    109       175    282.4
4         45  '2020/12/05'    117       148      406
5         60  '2020/12/06'    102       127      300
6         60  '2020/12/07'    110       136      374
7        450  '2020/12/08'    104       134    253.3
8         30  '2020/12/09'    109       133    195.1
9         60  '2020/12/10'     98       124      269
10        60  '2020/12/11'    103       147    329.3
11        60  '2020/12/12'    100       120    250.7
13        60  '2020/12/13'    106       128    345.3
14        60  '2020/12/14'    104       132    379.3
15        60  '2020/12/15'     98       123      275
16        60  '2020/12/16'     98       120    215.2
17        60  '2020/12/17'    100       120      300
18        45  '2020/12/18'     90       112      300
19        60  '2020/12/19'    103       123      323
20        45  '2020/12/20'     97       125      243
21        60  '2020/12/21'    108       131    364.2
22        45    2020/12/04    100       119      282
23        60  '2020/12/23'    130       101      300
24        45  '2020/12/24'    105       132      246
25        60  '2020/12/25'    102       126    334.5
26        60      20201226    100       120      250
27        60  '2020/12/27'     92       118      241
28        60  '2020/12/28'    103       132      300
29        60  '2020/12/29'    100       132      280
30        60  '2020/12/30'    102       129    380.3
31        60  '2020/12/31'     92       115      243
```

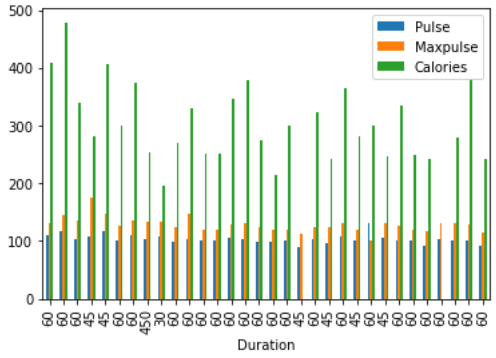
In [28]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
CALORIES_DATA.corr()
```

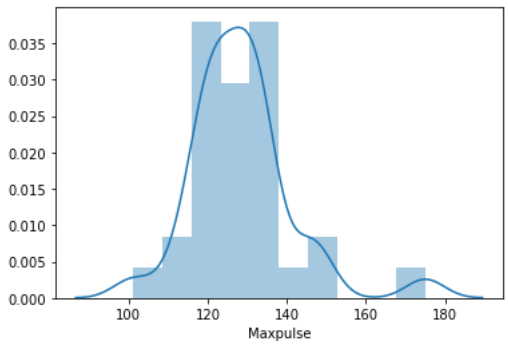
Out[28]:

	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	0.004410	0.049959	-0.114169
Pulse	0.004410	1.000000	0.276583	0.513186
Maxpulse	0.049959	0.276583	1.000000	0.357460
Calories	-0.114169	0.513186	0.357460	1.000000

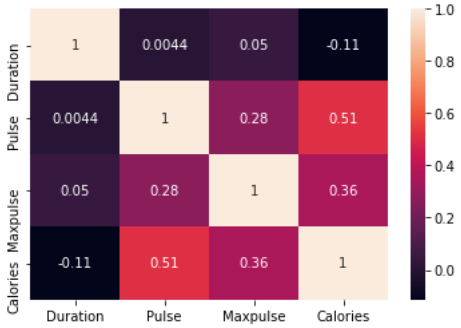
In [31]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
CALORIES_DATA['Pulse'].plot(kind = 'hist')
plt.show()
#bar
CALORIES_DATA.plot(kind = 'bar', x='Duration')
plt.show()
#distplot
import seaborn as sns
sns.distplot(CALORIES_DATA['Maxpulse'])
plt.show()
```







In [33]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(CALORIES_DATA.corr(),annot=True)
plt.show()
```



In [34]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
CALORIES_DATA.agg(['sum', 'min', 'max'])
```

Out[34]:

	Duration	Pulse	Maxpulse	Calories
sum	2190	3312	4112	9140.4
min	30	90	101	195.1
max	450	130	175	479.0

In [39]:
```
#group the dataset as per 'Duration‘ column
print('\ngrouping\n',CALORIES_DATA.groupby('Duration'))
#display the number (count) of values for each ‘Duration’
print('\nnumber (count) of values for each ‘Duration\n',CALORIES_DATA.groupby('Duration').count())
#display the sum of all the values for each ‘Duration’
print('\nsum of all the values for each ‘Duration’\n',CALORIES_DATA.groupby('Duration').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',CALORIES_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each ‘Duration’\n',CALORIES_DATA.groupby('Duration').agg(['sum', 'mean', 'median', 'min', 'max', 'p
```

```
grouping
 <pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000014F69D71850>

number (count) of values for each ‘Duration
           Date  Pulse  Maxpulse  Calories
Duration
30            1      1         1         1
45            5      6         6         5
60           24     24        24        23
450           1      1         1         1

sum of all the values for each ‘Duration’
           Pulse  Maxpulse  Calories
Duration
30           109       133     195.1
45           618       811    1459.4
60          2481      3034    7232.6
450          104       134     253.3

various Data Aggregation functions
              Duration         Pulse      Maxpulse      Calories
sum     2.190000e+03  3.312000e+03  4.112000e+03  9.140400e+03
mean    6.843750e+01  1.035000e+02  1.285000e+02  3.046800e+02
median  6.000000e+01  1.025000e+02  1.275000e+02  2.912000e+02
min     3.000000e+01  9.000000e+01  1.010000e+02  1.951000e+02
max     4.500000e+02  1.300000e+02  1.750000e+02  4.790000e+02
prod    3.664804e+18  8.222562e+18  8.700497e+18  1.701164e+74

sum of all the values for each ‘Duration’
           Pulse                                          Maxpulse              \
            sum     mean median  min  max          prod      sum        mean
Duration
30          109  109.000  109.0  109  109  1.090000e+02      133  133.000000
45          618  103.000  102.5   90  117  1.169004e+12      811  135.166667
60         2481  103.375  102.0   92  130  2.084879e+48     3034  126.416667
450         104  104.000  104.0  104  104  1.040000e+02      134  134.000000

                                      Calories                          \
          median  min  max          prod      sum       mean median    min
Duration
30         133.0  133  133  1.330000e+02    195.1  195.10000  195.1  195.1
45         128.5  112  175  5.695721e+12   1459.4  291.88000  282.0  243.0
60         126.5  101  147  2.591241e+50   7232.6  314.46087  300.0  215.2
450        134.0  134  134  1.340000e+02    253.3  253.30000  253.3  253.3


            max          prod
Duration
30        195.1  1.951000e+02
45        406.0  1.932775e+12
60        479.0  1.781036e+57
450       253.3  2.533000e+02
```

In [41]:
```
#DATASET 2 FROM HERE
```

In [2]:
```
import pandas as pd
#Loading the Dataset
DIABETES_DATA = pd.read_csv(r"C:\Users\HP\Downloads\diabetes.csv")
```

In [3]:
```
#Implement descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
#count
print('Count is\n',DIABETES_DATA.count(),'\n')
#mean
print('mean is\n',DIABETES_DATA.mean(),'\n')
#max
print('max is\n',DIABETES_DATA.max(),'\n')
#min
print('min is\n',DIABETES_DATA.min(),'\n')
#standard deviation
print('standard deviation is\n',DIABETES_DATA.std(),'\n')

#print(Calories.describe()) #can also use to get the required output
```

```
Count is
 Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64

mean is
 Pregnancies                 3.845052
Glucose                   120.894531
BloodPressure              69.105469
SkinThickness              20.536458
Insulin                    79.799479
BMI                        31.992578
DiabetesPedigreeFunction    0.471876
Age                        33.240885
Outcome                     0.348958
dtype: float64

max is
 Pregnancies                17.00
Glucose                   199.00
BloodPressure             122.00
SkinThickness              99.00
Insulin                   846.00
BMI                        67.10
DiabetesPedigreeFunction    2.42
Age                        81.00
Outcome                     1.00
dtype: float64

min is
 Pregnancies                 0.000
Glucose                    0.000
BloodPressure              0.000
SkinThickness              0.000
Insulin                    0.000
BMI                        0.000
DiabetesPedigreeFunction   0.078
Age                       21.000
Outcome                    0.000
dtype: float64

standard deviation is
 Pregnancies                 3.369578
Glucose                    31.972618
BloodPressure              19.355807
SkinThickness              15.952218
Insulin                   115.244002
BMI                         7.884160
DiabetesPedigreeFunction    0.331329
Age                        11.760232
Outcome                     0.476951
dtype: float64
```

In [46]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',DIABETES_DATA.isnull(),'\n')
DIABETES_DATA['Pregnancies'].fillna('100',inplace=True)
DIABETES_DATA['Glucose'].fillna('60',inplace=True)
DIABETES_DATA['SkinThickness'].fillna('50',inplace=True)
DIABETES_DATA['Insulin'].fillna('100',inplace=True)
DIABETES_DATA['BMI'].fillna('30',inplace=True)
DIABETES_DATA['DiabetesPedigreeFunction'].fillna('0.351',inplace=True)
DIABETES_DATA['Age'].fillna('30',inplace=True)
DIABETES_DATA['Outcome'].fillna('1',inplace=True)
print('updated dataset',DIABETES_DATA)
```

```
checking null values
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0           False    False          False          False    False  False
1           False    False          False          False    False  False
2           False    False          False          False    False  False
3           False    False          False          False    False  False
4           False    False          False          False    False  False
..            ...      ...            ...            ...      ...    ...
763         False    False          False          False    False  False
764         False    False          False          False    False  False
765         False    False          False          False    False  False
766         False    False          False          False    False  False
767         False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        ...    ...      ...
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]

updated dataset      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
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

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

In [48]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',DIABETES_DATA.duplicated())
print()
DIABETES_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(DIABETES_DATA)
```

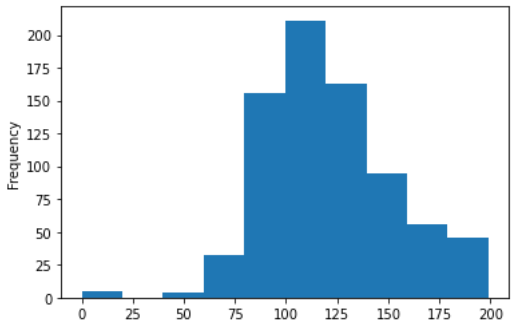
```
checking duplicate values
 0      False
1       False
2       False
3       False
4       False
        ...
763     False
764     False
765     False
766     False
767     False
Length: 768, dtype: bool

dataset after removing duplicate values

     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
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

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

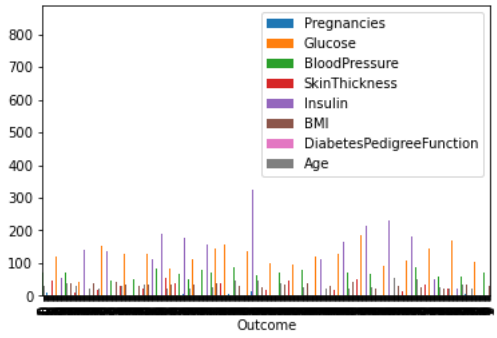
In [49]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
DIABETES_DATA.corr()
```

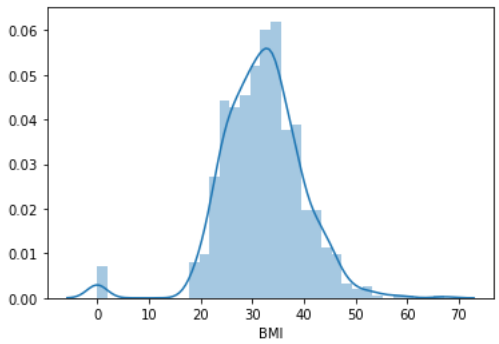
Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

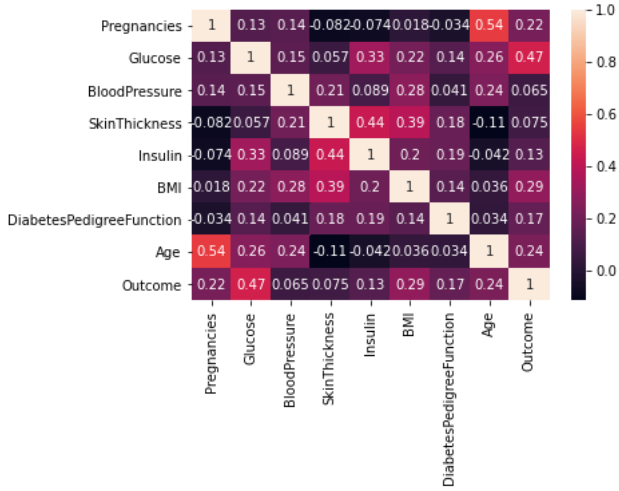
In [54]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
DIABETES_DATA['Glucose'].plot(kind = 'hist')
plt.show()
#bar
DIABETES_DATA.plot(kind = 'bar', x='Outcome')
plt.show()
#distplot
import seaborn as sns
sns.distplot(DIABETES_DATA['BMI'])
plt.show()
```







In [55]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(DIABETES_DATA.corr(),annot=True)
plt.show()
```



In [57]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
DIABETES_DATA.agg(['sum', 'min', 'max'])
```

Out[57]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
sum	2953	92847	53073	15772	61286	24570.3	362.401	25529	268
min	0	0	0	0	0	0.0	0.078	21	0
max	17	199	122	99	846	67.1	2.420	81	1

In [4]:
```
#group the dataset as per 'glucose' column
print('\ngrouping\n',DIABETES_DATA.groupby('Glucose'))
#display the number (count) of values for each 'Duration'
print('\nnumber (count) of values for each "Glucose"\n',DIABETES_DATA.groupby('Glucose').count())
#display the sum of all the values for each 'Duration'
print('\nsum of all the values for each 'Duration'\n',DIABETES_DATA.groupby('Glucose').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',DIABETES_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each 'Duration'\n',DIABETES_DATA.groupby('Glucose').agg(['sum', 'mean', 'median', 'min', 'max', 'pr
```

```
61                3  3.000000    3.0   3   3    3            82  82.000000
...             ...       ...    ...  ..  ..  ...           ...        ...
195              13  6.500000    6.5   6   7   42           140  70.000000
196              16  5.333333    7.0   1   8   56           242  80.666667
197              16  4.000000    3.0   2   8  128           284  71.000000
198               0  0.000000    0.0   0   0    0            66  66.000000
199               1  1.000000    1.0   1   1    1            76  76.000000

                  ...    Age                    Outcome                  \
        median min  ... median min max        prod     sum  mean median min
Glucose             ...
0         68.0  48  ...   22.0  21  41  15418788.0       2  0.40    0.0   0
44        62.0  62  ...   36.0  36  36        36.0       0  0.00    0.0   0
56        56.0  56  ...   22.0  22  22        22.0       0  0.00    0.0   0
57        70.0  60  ...   54.0  41  67      2747.0       0  0.00    0.0   0
61        82.0  82  ...   46.0  46  46        46.0       0  0.00    0.0   0
...        ...  ..  ...    ...  ..  ..         ...     ...   ...    ...  ..
195       70.0  70  ...   43.0  31  55      1705.0       2  1.00    1.0   1
196       76.0  76  ...   41.0  29  57     67773.0       3  1.00    1.0   1
197       70.0  70  ...   46.0  31  62   3972774.0       3  0.75    1.0   0
```

In [5]:
```
#DATASET 2 FROM HERE
```

In [10]:
```
import pandas as pd
#Loading the Dataset
HOUSING_DATA = pd.read_csv(r"C:\Users\HP\Downloads\HousingPrice-DataSet.csv")
```

In [11]:
```
#Implement descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
#count
print('Count is\n',HOUSING_DATA.count(),'\n')
#mean
print('mean is\n',HOUSING_DATA.mean(),'\n')
#max
print('max is\n',HOUSING_DATA.max(),'\n')
#min
print('min is\n',HOUSING_DATA.min(),'\n')
#standard deviation
print('standard deviation is\n',HOUSING_DATA.std(),'\n')

#print(HOUSING_DATA.describe()) #can also use to get the required output
```

```
Count is
 CRIM       486
ZN         486
INDUS      486
CHAS       486
NOX        506
RM         506
AGE        486
DIS        506
RAD        506
TAX        506
PTRATIO    506
B          506
LSTAT      486
MEDV       506
dtype: int64

mean is
 CRIM         3.611874
ZN           11.211934
INDUS        11.083992
CHAS          0.069959
NOX           0.554695
RM            6.284634
AGE          68.518519
DIS           3.795043
RAD           9.549407
TAX         408.237154
PTRATIO      18.455534
B           356.674032
LSTAT        12.715432
MEDV         22.532806
dtype: float64

max is
 CRIM        88.9762
ZN         100.0000
INDUS       27.7400
CHAS         1.0000
NOX          0.8710
RM           8.7800
AGE        100.0000
DIS          12.1265
RAD          24.0000
TAX         711.0000
PTRATIO      22.0000
B           396.9000
LSTAT        37.9700
MEDV         50.0000
dtype: float64

min is
 CRIM         0.00632
ZN           0.00000
INDUS        0.46000
CHAS         0.00000
NOX          0.38500
RM           3.56100
AGE          2.90000
DIS          1.12960
RAD          1.00000
TAX        187.00000
PTRATIO     12.60000
B            0.32000
LSTAT        1.73000
MEDV         5.00000
dtype: float64

standard deviation is
 CRIM         8.720192
ZN          23.388876
INDUS        6.835896
CHAS         0.255340
NOX          0.115878
RM           0.702617
AGE         27.999513
DIS          2.105710
RAD          8.707259
TAX        168.537116
PTRATIO      2.164946
B           91.294864
LSTAT        7.155871
MEDV         9.197104
dtype: float64
```

In [14]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',HOUSING_DATA.isnull(),'\n')
HOUSING_DATA['CRIM'].fillna('0.654',inplace=True)
HOUSING_DATA['ZN'].fillna('10',inplace=True)
HOUSING_DATA['INDUS'].fillna('7.87',inplace=True)
HOUSING_DATA['CHAS'].fillna('0',inplace=True)
HOUSING_DATA['NOX'].fillna('0.524',inplace=True)
HOUSING_DATA['RM'].fillna('6.575',inplace=True)
HOUSING_DATA['AGE'].fillna('30',inplace=True)
print('updated dataset',HOUSING_DATA)
```

```
checking null values
       CRIM     ZN  INDUS   CHAS    NOX     RM    AGE    DIS    RAD    TAX  \
0    False  False  False  False  False  False  False  False  False  False
1    False  False  False  False  False  False  False  False  False  False
2    False  False  False  False  False  False  False  False  False  False
3    False  False  False  False  False  False  False  False  False  False
4    False  False  False  False  False  False  False  False  False  False
..     ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
501  False  False  False  False  False  False  False  False  False  False
502  False  False  False  False  False  False  False  False  False  False
503  False  False  False  False  False  False  False  False  False  False
504  False  False  False  False  False  False  False  False  False  False
505  False  False  False  False  False  False  False  False  False  False

     PTRATIO      B  LSTAT   MEDV
0      False  False  False  False
1      False  False  False  False
2      False  False  False  False
3      False  False  False  False
4      False  False   True  False
..       ...    ...    ...    ...
501    False  False   True  False
502    False  False  False  False
503    False  False  False  False
504    False  False  False  False
505    False  False  False  False

[506 rows x 14 columns]

updated dataset         CRIM  ZN  INDUS CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0    0.00632  18   2.31    0  0.538  6.575  65.2  4.0900    1  296     15.3
1    0.02731   0   7.07    0  0.469  6.421  78.9  4.9671    2  242     17.8
2    0.02729   0   7.07    0  0.469  7.185  61.1  4.9671    2  242     17.8
3    0.03237   0   2.18    0  0.458  6.998  45.8  6.0622    3  222     18.7
4    0.06905   0   2.18    0  0.458  7.147  54.2  6.0622    3  222     18.7
..       ...  ..    ...  ...    ...    ...   ...     ...  ...  ...      ...
501  0.06263   0  11.93    0  0.573  6.593  69.1  2.4786    1  273     21.0
502  0.04527   0  11.93    0  0.573  6.120  76.7  2.2875    1  273     21.0
503  0.06076   0  11.93    0  0.573  6.976    91  2.1675    1  273     21.0
504  0.10959   0  11.93    0  0.573  6.794  89.3  2.3889    1  273     21.0
505  0.04741   0  11.93    0  0.573  6.030    30  2.5050    1  273     21.0

          B  LSTAT  MEDV
0    396.90   4.98  24.0
1    396.90   9.14  21.6
2    392.83   4.03  34.7
3    394.63   2.94  33.4
4    396.90    NaN  36.2
..      ...    ...   ...
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9

[506 rows x 14 columns]
```

In [15]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',HOUSING_DATA.duplicated())
print()
HOUSING_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(HOUSING_DATA)
```

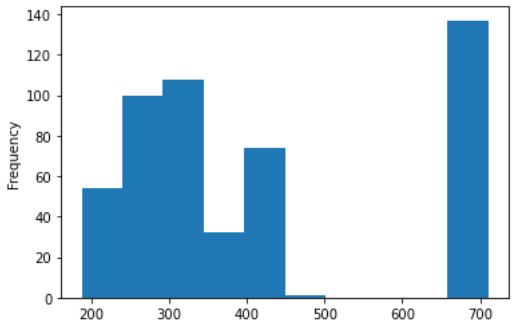
```
checking duplicate values
 0      False
1       False
2       False
3       False
4       False
        ...
501     False
502     False
503     False
504     False
505     False
Length: 506, dtype: bool

dataset after removing duplicate values

        CRIM  ZN  INDUS CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0    0.00632  18   2.31    0  0.538  6.575  65.2  4.0900    1  296     15.3
1    0.02731   0   7.07    0  0.469  6.421  78.9  4.9671    2  242     17.8
2    0.02729   0   7.07    0  0.469  7.185  61.1  4.9671    2  242     17.8
3    0.03237   0   2.18    0  0.458  6.998  45.8  6.0622    3  222     18.7
4    0.06905   0   2.18    0  0.458  7.147  54.2  6.0622    3  222     18.7
..       ...  ..    ...  ...    ...    ...   ...     ...  ...  ...      ...
501  0.06263   0  11.93    0  0.573  6.593  69.1  2.4786    1  273     21.0
502  0.04527   0  11.93    0  0.573  6.120  76.7  2.2875    1  273     21.0
503  0.06076   0  11.93    0  0.573  6.976    91  2.1675    1  273     21.0
504  0.10959   0  11.93    0  0.573  6.794  89.3  2.3889    1  273     21.0
505  0.04741   0  11.93    0  0.573  6.030    30  2.5050    1  273     21.0

          B  LSTAT  MEDV
0    396.90   4.98  24.0
1    396.90   9.14  21.6
2    392.83   4.03  34.7
3    394.63   2.94  33.4
4    396.90    NaN  36.2
..      ...    ...   ...
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9

[506 rows x 14 columns]
```

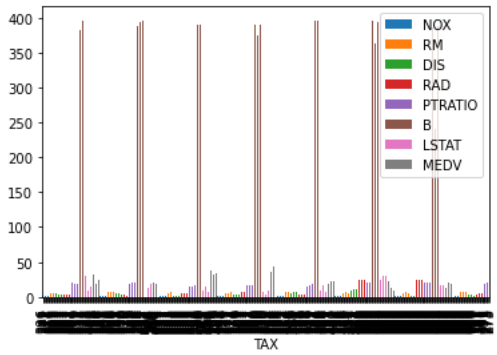
In [16]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
HOUSING_DATA.corr()
```

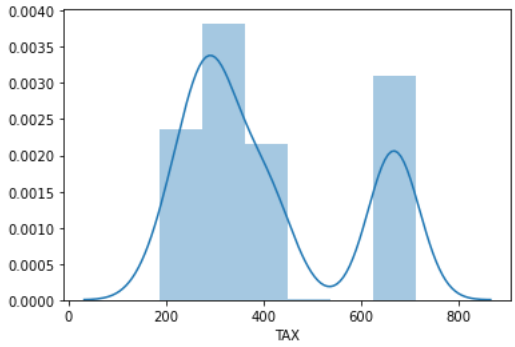
Out[16]:

	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
NOX	1.000000	-0.302188	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.582641	-0.427321
RM	-0.302188	1.000000	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.614339	0.695360
DIS	-0.769230	0.205246	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.493328	0.249929
RAD	0.611441	-0.209847	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.479541	-0.381626
TAX	0.668023	-0.292048	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.536110	-0.468536
PTRATIO	0.188933	-0.355501	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.375966	-0.507787
B	-0.380051	0.128069	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.369889	0.333461
LSTAT	0.582641	-0.614339	-0.493328	0.479541	0.536110	0.375966	-0.369889	1.000000	-0.735822
MEDV	-0.427321	0.695360	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.735822	1.000000

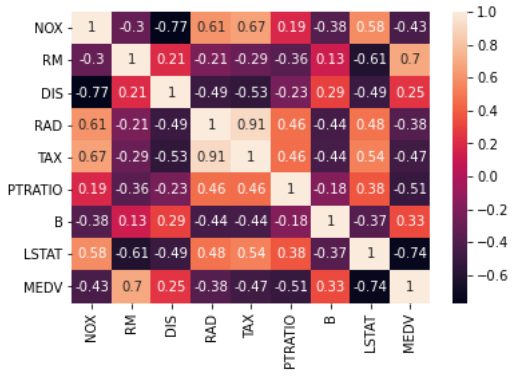
In [21]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
HOUSING_DATA['TAX'].plot(kind = 'hist')
plt.show()
#bar
HOUSING_DATA.plot(kind = 'bar', x='TAX')
plt.show()
#distplot
import seaborn as sns
sns.distplot(HOUSING_DATA['TAX'])
plt.show()
```







In [22]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(HOUSING_DATA.corr(),annot=True)
plt.show()
```



In [23]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
HOUSING_DATA.agg(['sum', 'min', 'max'])
```

Out[23]:

	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
sum	280.6757	3180.025	1920.2916	4832	206568	9338.5	180477.06	6179.70	11401.6
min	0.3850	3.561	1.1296	1	187	12.6	0.32	1.73	5.0
max	0.8710	8.780	12.1265	24	711	22.0	396.90	37.97	50.0

In [24]:
```
#group the dataset as per 'TAX‘ column
print('\ngrouping\n',HOUSING_DATA.groupby('TAX'))
#display the number (count) of values for each ‘TAX’
print('\nnumber (count) of values for each "TAX"\n',HOUSING_DATA.groupby('TAX').count())
#display the sum of all the values for each ‘TAX’
print('\nsum of all the values for each ‘TAX’\n',HOUSING_DATA.groupby('TAX').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',HOUSING_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each ‘TAX’\n',HOUSING_DATA.groupby('TAX').agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
```

```
grouping
 <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002765CE8B6D0>

number (count) of values for each "TAX"
      CRIM   ZN  INDUS  CHAS  NOX   RM  AGE  DIS  RAD  PTRATIO    B  LSTAT  \
TAX
187      1    1      1     1    1    1    1    1    1        1    1      1
188      7    7      7     7    7    7    7    7    7        7    7      7
193      8    8      8     8    8    8    8    8    8        8    8      8
198      1    1      1     1    1    1    1    1    1        1    1      1
216      5    5      5     5    5    5    5    5    5        5    5      5
..     ...  ...    ...   ...  ...  ...  ...  ...  ...      ...  ...    ...
432      9    9      9     9    9    9    9    9    9        9    9      8
437     15   15     15    15   15   15   15   15   15       15   15     15
469      1    1      1     1    1    1    1    1    1        1    1      1
666    132  132    132   132  132  132  132  132  132      132  132    126
711      5    5      5     5    5    5    5    5    5        5    5      5

     MEDV
```

In [ ]:
```
```