Importing the required packages for the models

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv('base.csv') #Read the datafiles and loading the datasets,trying to under
stand the data.
df
```

Out[2]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35
5	93	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40
10	88	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45
11	86	729	5459	1363	278	26	230	486	1392	121	618	572	3.57	5	13	39
12	85	661	5417	1331	243	21	176	435	1150	52	675	630	3.94	2	12	46
13	76	656	5544	1379	262	22	198	478	1336	69	726	677	4.16	6	12	45
14	68	694	5600	1405	277	46	146	475	1119	78	729	664	4.14	5	15	28
15	100	647	5484	1386	288	39	137	506	1267	69	525	478	2.94	1	15	62
16	98	697	5631	1462	292	27	140	461	1322	98	596	532	3.21	0	13	54
17	97	689	5491	1341	272	30	171	567	1518	95	608	546	3.36	6	21	48
18	68	655	5480	1378	274	34	145	412	1299	84	737	682	4.28	1	7	40
19	64	640	5571	1382	257	27	167	496	1255	134	754	700	4.33	2	8	35
20	90	683	5527	1351	295	17	177	488	1290	51	613	557	3.43	1	14	50
21	83	703	5428	1363	265	13	177	539	1344	57	635	577	3.62	4	13	41
22	71	613	5463	1420	236	40	120	375	1150	112	678	638	4.02	0	12	35
23	67	573	5420	1361	251	18	100	471	1107	69	760	698	4.41	3	10	44
24	63	626	5529	1374	272	37	130	387	1274	88	809	749	4.69	1	7	35
25	92	667	5385	1346	263	26	187	563	1258	59	595	553	3.44	6	21	47
26	84	696	5565	1486	288	39	136	457	1159	93	627	597	3.72	7	18	41
27	79	720	5649	1494	289	48	154	490	1312	132	713	659	4.04	1	12	44
28	74	650	5457	1324	260	36	148	426	1327	82	731	655	4.09	1	6	41
29	68	737	5572	1479	274	49	186	388	1283	97	844	799	5.04	4	4	36

In [3]:

```
df.shape #knowing the shape of the datasets
```

Out[3]:

```
(30, 17)
```

In [4]:

```
df.dtypes #finding the datatypes of each of the columns.
```

Out[4]:

```
W        int64
R        int64
AB       int64
H        int64
2B       int64
3B       int64
HR       int64
BB       int64
SO       int64
SB       int64
RA       int64
ER       int64
ERA    float64
CG       int64
SHO      int64
SV       int64
E        int64
dtype: object
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   W       30 non-null     int64  
 1   R       30 non-null     int64  
 2   AB      30 non-null     int64  
 3   H       30 non-null     int64  
 4   2B      30 non-null     int64  
 5   3B      30 non-null     int64  
 6   HR      30 non-null     int64  
 7   BB      30 non-null     int64  
 8   SO      30 non-null     int64  
 9   SB      30 non-null     int64  
 10  RA      30 non-null     int64  
 11  ER      30 non-null     int64  
 12  ERA     30 non-null     float64
 13  CG      30 non-null     int64  
 14  SHO     30 non-null     int64  
 15  SV      30 non-null     int64  
 16  E       30 non-null     int64  
dtypes: float64(1), int64(16)
memory usage: 4.1 KB
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['W', 'R', 'AB', 'H', '2B', '3B', 'HR', 'BB', 'SO', 'SB', 'RA', 'E
R',
       'ERA', 'CG', 'SHO', 'SV', 'E'],
      dtype='object')
```

In [7]:

```
df.describe() #understanding the datasets
```

Out[7]:

	W	R	AB	H	2B	3B	HR
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000
mean	80.966667	688.233333	5516.266667	1403.533333	274.733333	31.300000	163.633333
std	10.453455	58.761754	70.467372	57.140923	18.095405	10.452355	31.823309
min	63.000000	573.000000	5385.000000	1324.000000	236.000000	13.000000	100.000000
25%	74.000000	651.250000	5464.000000	1363.000000	262.250000	23.000000	140.250000
50%	81.000000	689.000000	5510.000000	1382.500000	275.500000	31.000000	158.500000
75%	87.750000	718.250000	5570.000000	1451.500000	288.750000	39.000000	177.000000
max	100.000000	891.000000	5649.000000	1515.000000	308.000000	49.000000	232.000000

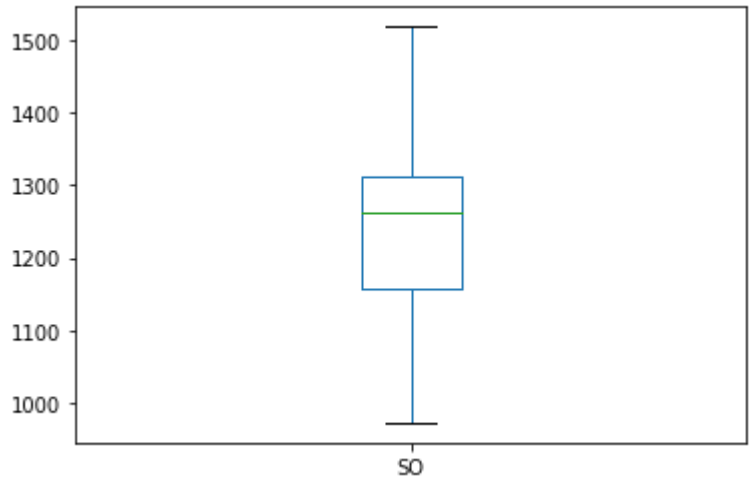
Data Visualization Process

In [8]:

```
df['SO'].plot.box()
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1437124a2c8>
```

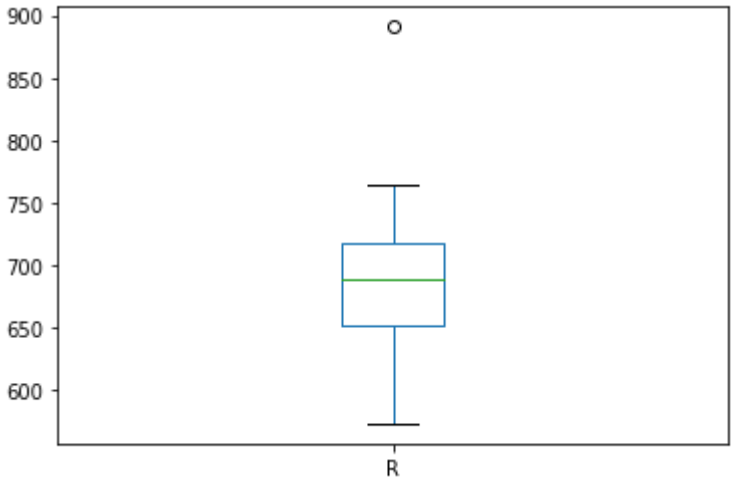


In [9]:

```
df['R'].plot.box()
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14371975dc8>
```

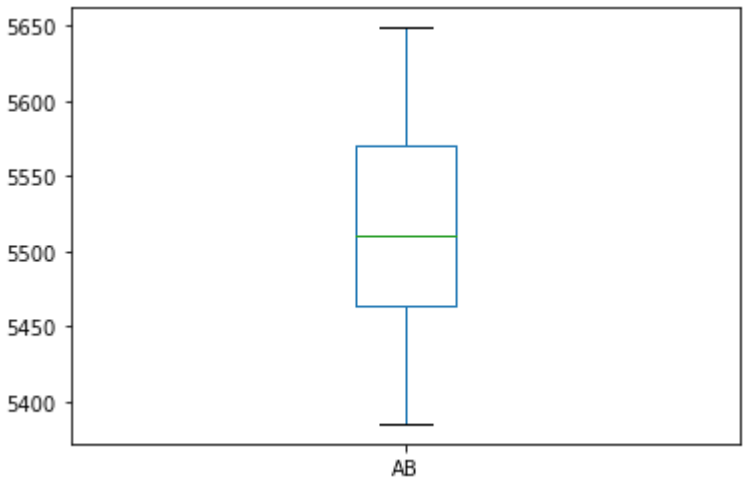


In [10]:

```
df['AB'].plot.box()
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14371a140c8>
```

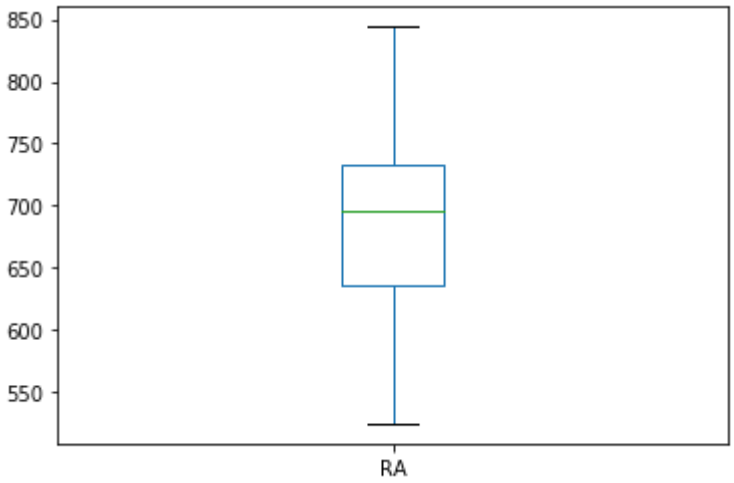


In [11]:

```
df['RA'].plot.box()
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14371a81588>
```

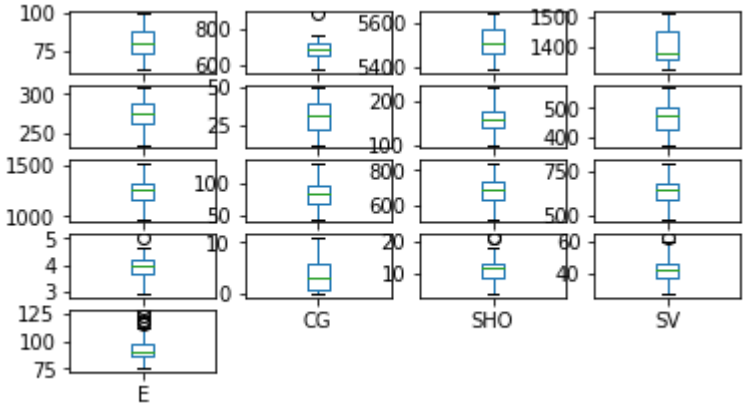


In [12]:

```
df.plot(kind='box',subplots=True,layout=(6,4))
```

Out[12]:

```
W          AxesSubplot(0.125,0.772143;0.168478x0.107857)
R       AxesSubplot(0.327174,0.772143;0.168478x0.107857)
AB      AxesSubplot(0.529348,0.772143;0.168478x0.107857)
H       AxesSubplot(0.731522,0.772143;0.168478x0.107857)
2B         AxesSubplot(0.125,0.642714;0.168478x0.107857)
3B      AxesSubplot(0.327174,0.642714;0.168478x0.107857)
HR      AxesSubplot(0.529348,0.642714;0.168478x0.107857)
BB      AxesSubplot(0.731522,0.642714;0.168478x0.107857)
SO         AxesSubplot(0.125,0.513286;0.168478x0.107857)
SB      AxesSubplot(0.327174,0.513286;0.168478x0.107857)
RA      AxesSubplot(0.529348,0.513286;0.168478x0.107857)
ER      AxesSubplot(0.731522,0.513286;0.168478x0.107857)
ERA        AxesSubplot(0.125,0.383857;0.168478x0.107857)
CG      AxesSubplot(0.327174,0.383857;0.168478x0.107857)
SHO     AxesSubplot(0.529348,0.383857;0.168478x0.107857)
SV      AxesSubplot(0.731522,0.383857;0.168478x0.107857)
E          AxesSubplot(0.125,0.254429;0.168478x0.107857)
dtype: object
```

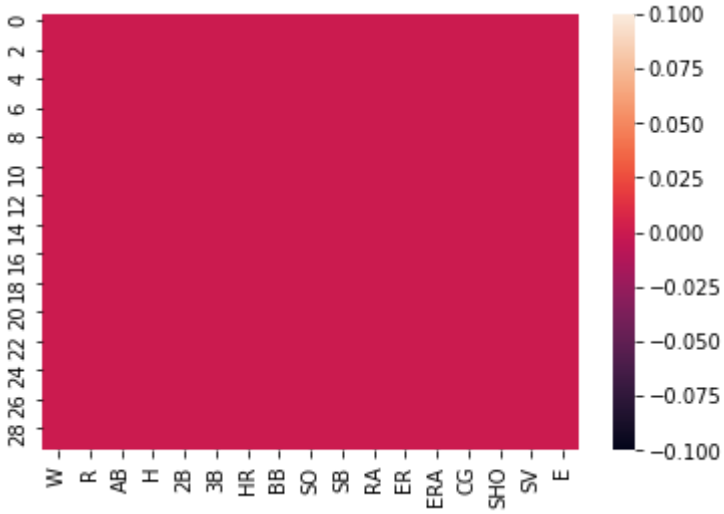


In [13]:

```
sns.heatmap(df.isnull()) #finding out if their any null value graphically.
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x143722ca288>
```



In [14]:

```
dfcor=df.corr()  #finding out the correlation between each variable in the dataset.
dfcor
```

Out[14]:

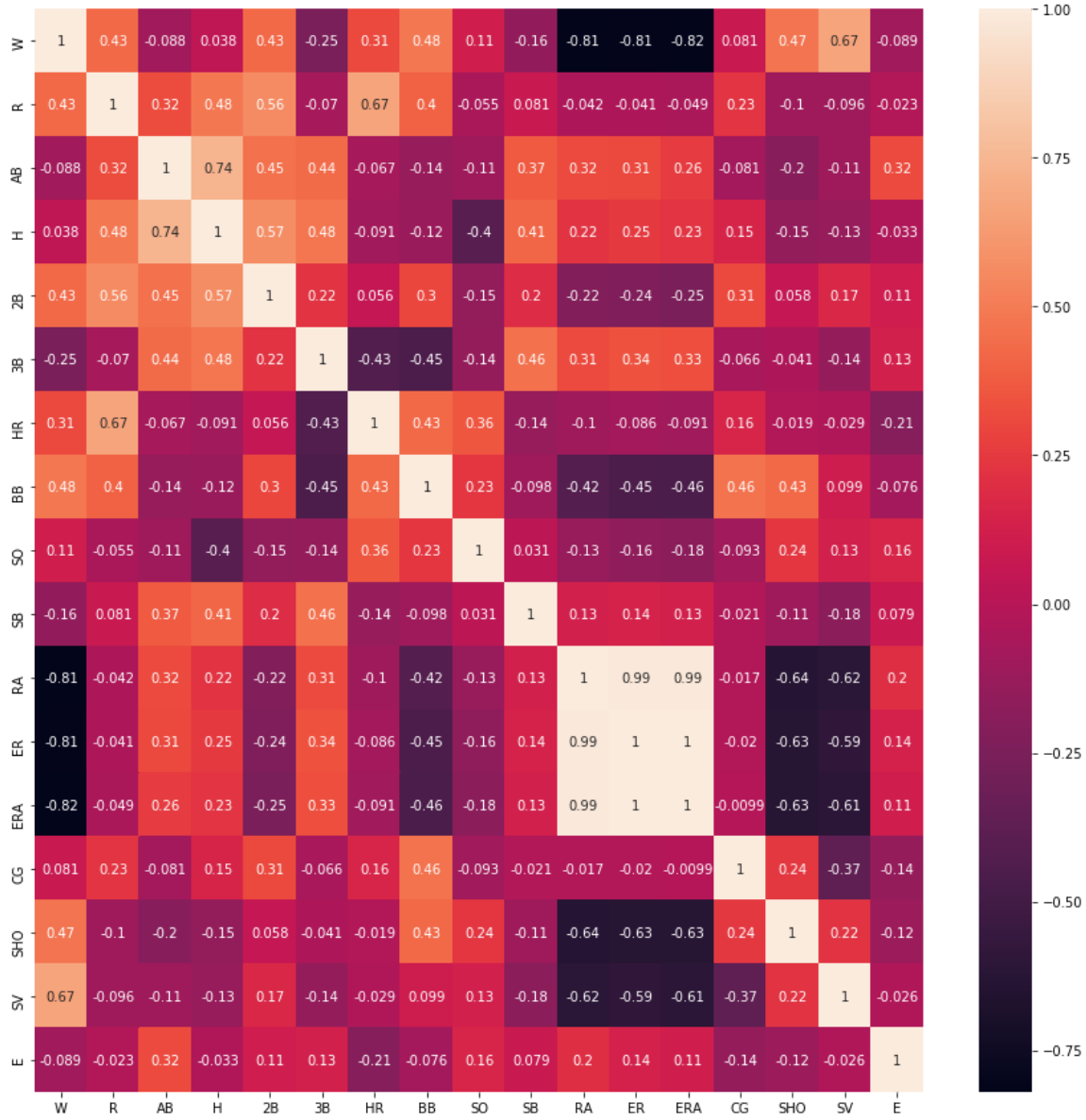
	W	R	AB	H	2B	3B	HR	BB
W	1.000000	0.430751	-0.087947	0.037612	0.427797	-0.251118	0.307407	0.484342
R	0.430751	1.000000	0.319464	0.482856	0.560084	-0.070072	0.671283	0.402452
AB	-0.087947	0.319464	1.000000	0.739122	0.453370	0.435422	-0.066983	-0.136414
H	0.037612	0.482856	0.739122	1.000000	0.566847	0.478694	-0.090855	-0.118281
2B	0.427797	0.560084	0.453370	0.566847	1.000000	0.220490	0.056292	0.302700
3B	-0.251118	-0.070072	0.435422	0.478694	0.220490	1.000000	-0.430915	-0.454949
HR	0.307407	0.671283	-0.066983	-0.090855	0.056292	-0.430915	1.000000	0.425691
BB	0.484342	0.402452	-0.136414	-0.118281	0.302700	-0.454949	0.425691	1.000000
SO	0.111850	-0.054726	-0.106022	-0.398830	-0.150752	-0.141196	0.359923	0.233652
SB	-0.157234	0.081367	0.372618	0.413444	0.195027	0.457437	-0.136567	-0.098347
RA	-0.812952	-0.041623	0.316010	0.224324	-0.218160	0.314125	-0.103903	-0.416445
ER	-0.809435	-0.041245	0.309686	0.252489	-0.235531	0.340225	-0.085922	-0.452663
ERA	-0.819600	-0.049281	0.255551	0.231172	-0.254854	0.330951	-0.090917	-0.459832
CG	0.080533	0.232042	-0.080876	0.147955	0.306675	-0.065898	0.156502	0.462478
SHO	0.471805	-0.103274	-0.197321	-0.145559	0.057998	-0.041396	-0.019119	0.426004
SV	0.666530	-0.096380	-0.106367	-0.130371	0.171576	-0.142370	-0.028540	0.099445
E	-0.089485	-0.023262	0.316743	-0.033173	0.105754	0.126678	-0.207597	-0.075685

In [15]:

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14372393fc8>
```

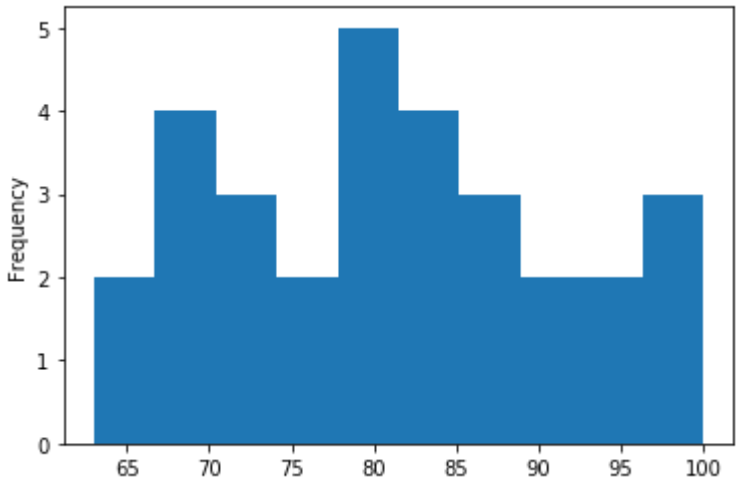


In [16]:

```
df['W'].plot.hist()
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14372922608>
```

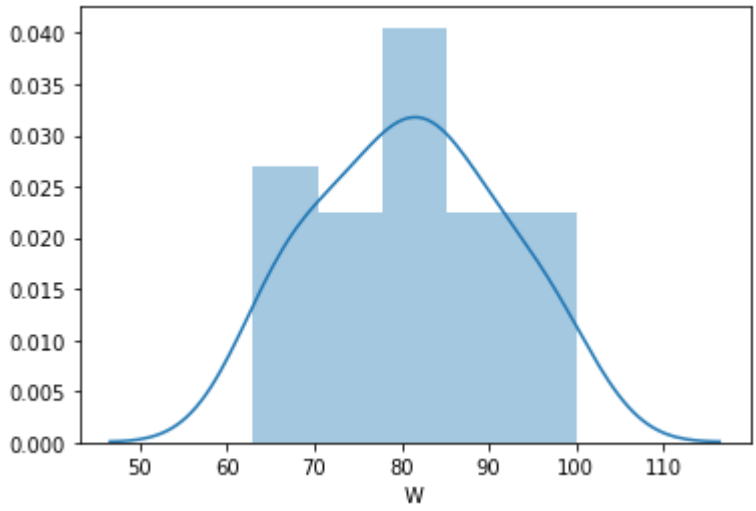


In [17]:

```
sns.distplot(df['W'])
```

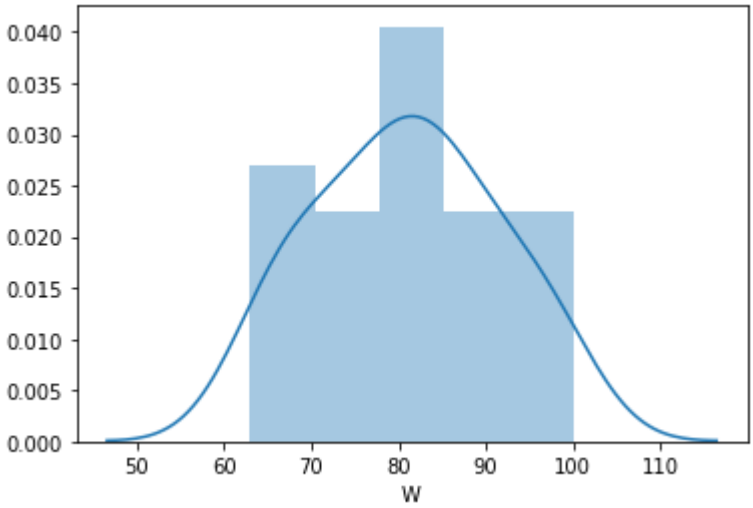
Out[17]:

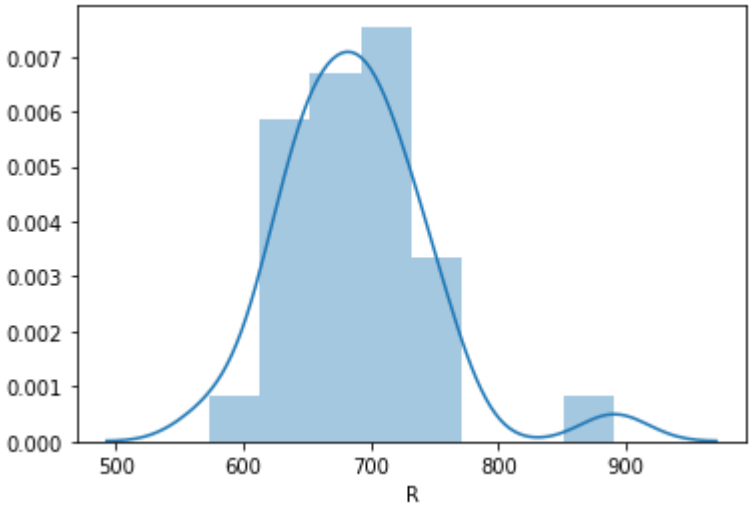
```
<matplotlib.axes._subplots.AxesSubplot at 0x14372bda0c8>
```

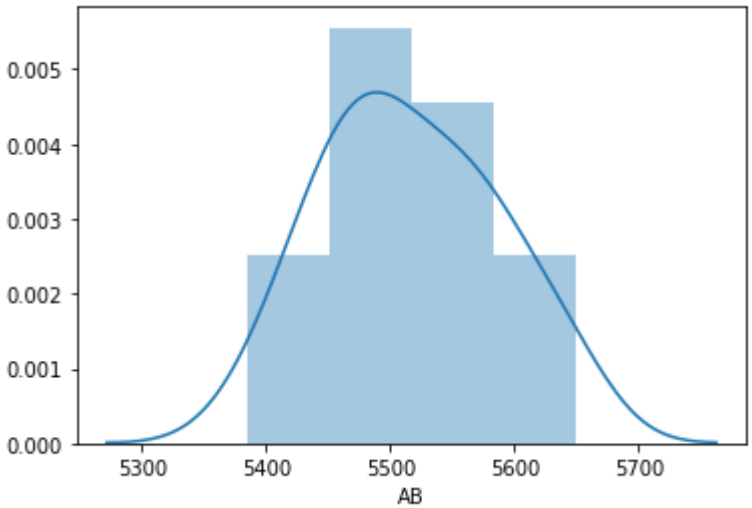


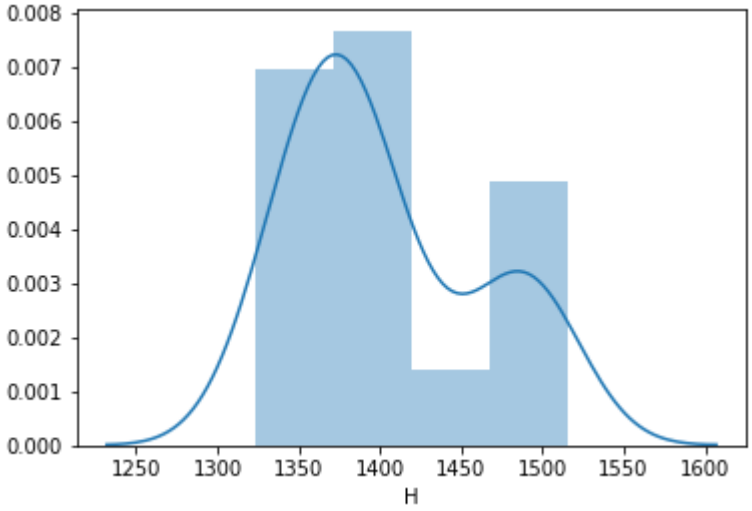
In [18]:

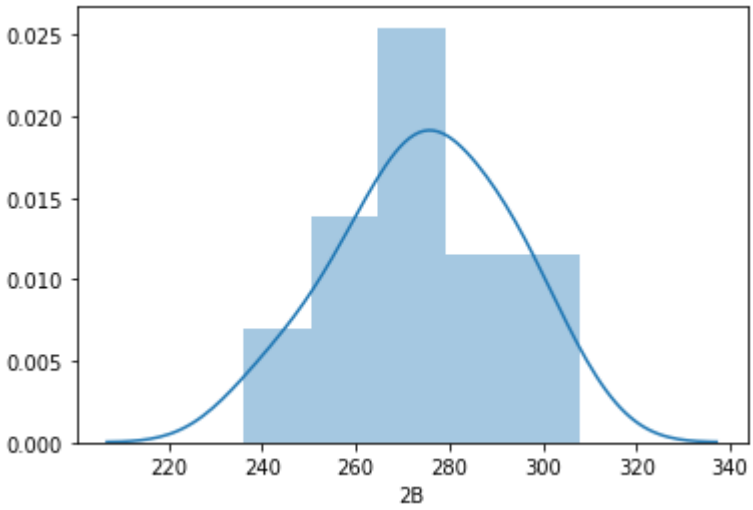
```
for i in df.columns:
    plt.figure()
    sns.distplot(df[i])
```

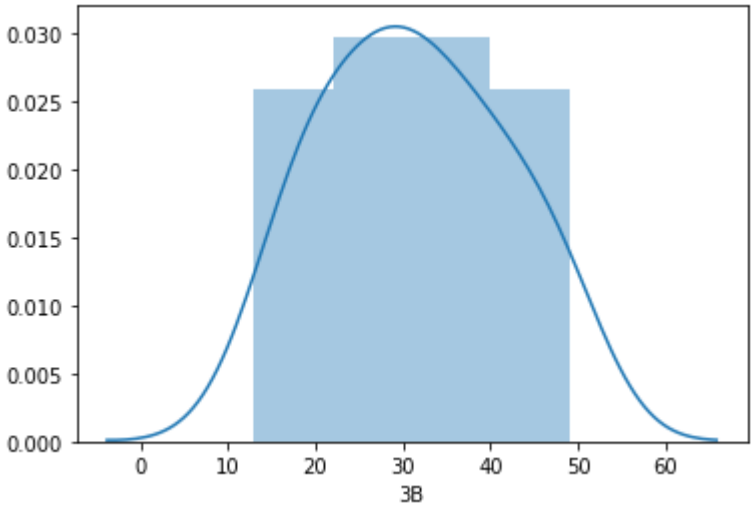

0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
50
60
70
80
90
100
110
W

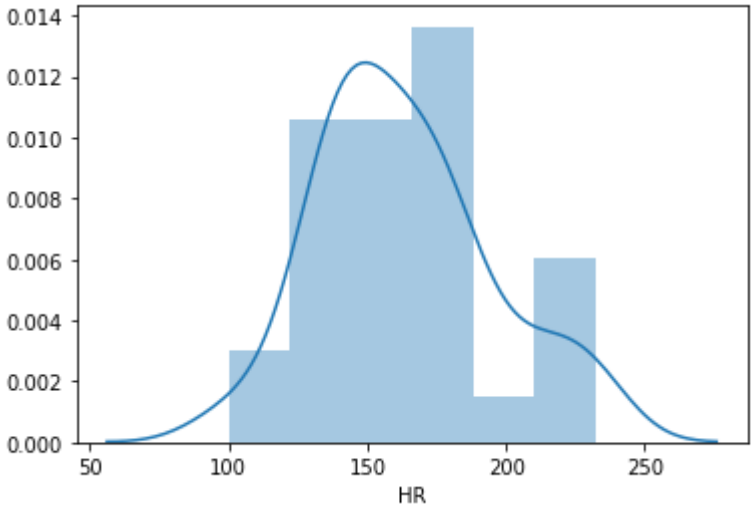

0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
500
600
700
800
900
R

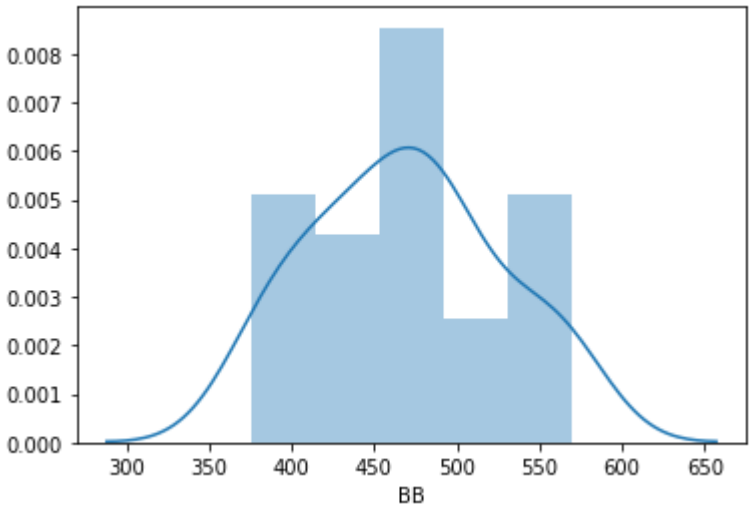

0.005
0.004
0.003
0.002
0.001
0.000
5300
5400
5500
5600
5700
AB

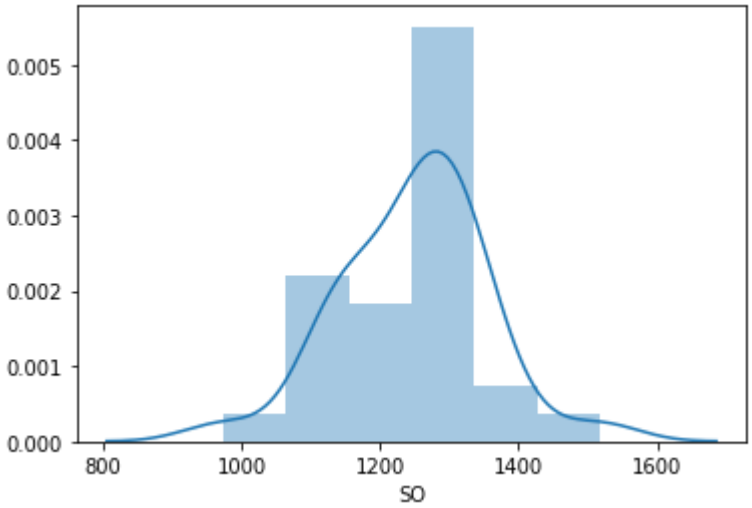

0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
1250
1300
1350
1400
1450
1500
1550
1600
H

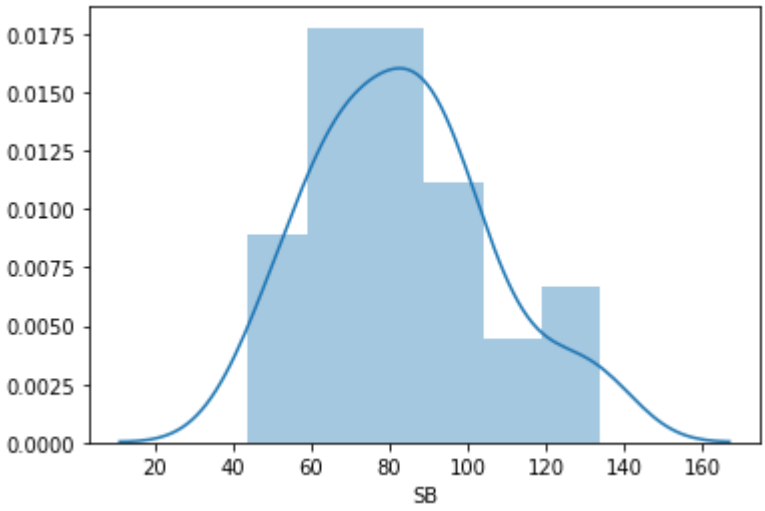

0.025
0.020
0.015
0.010
0.005
0.000
220
240
260
280
300
320
340
2B

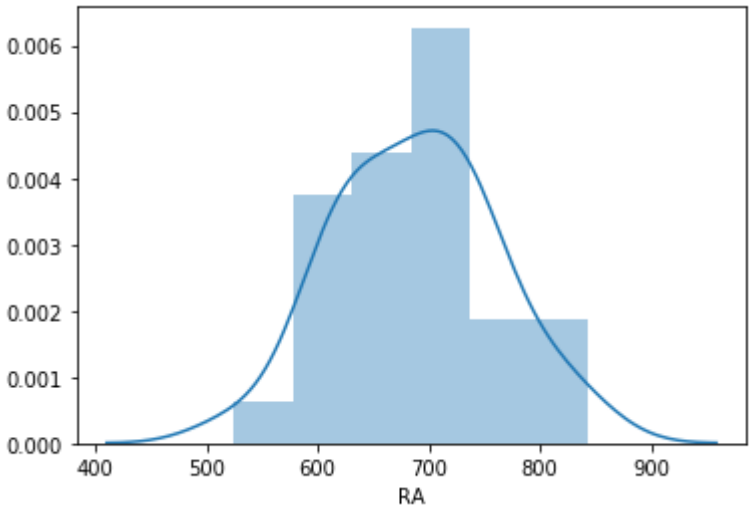

0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60
3B

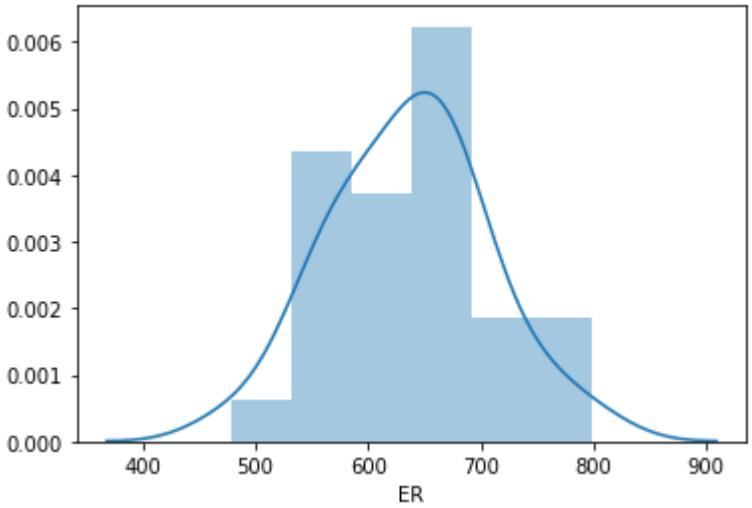

0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
50
100
150
200
250
HR

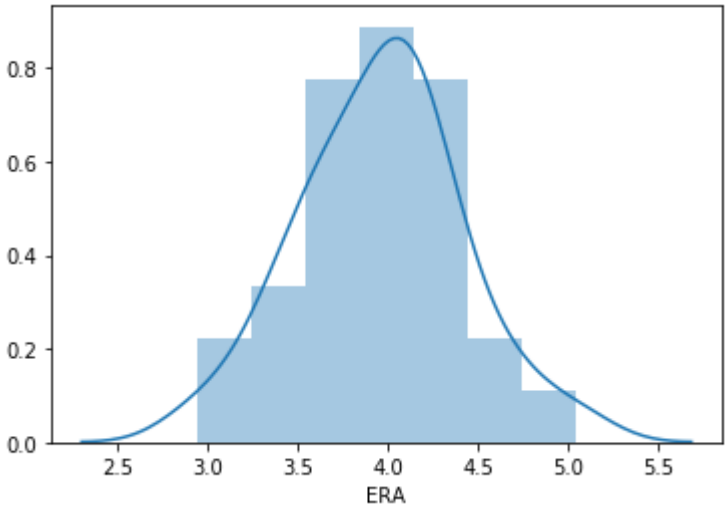

0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
300
350
400
450
500
550
600
650
BB

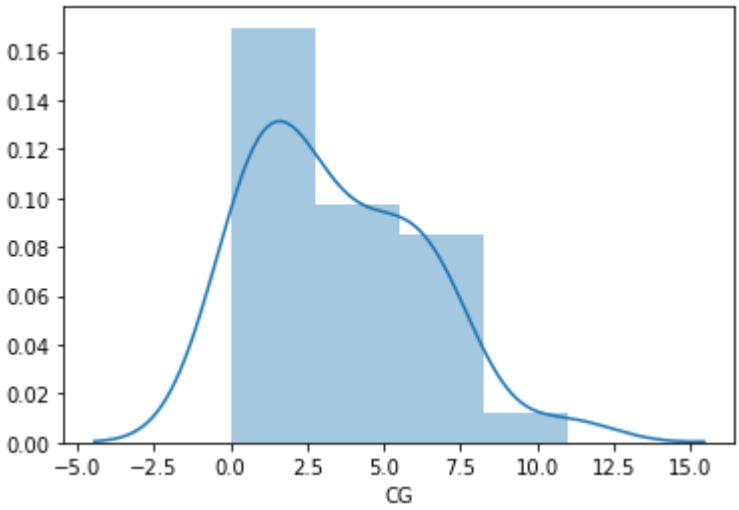

0.005
0.004
0.003
0.002
0.001
0.000
800
1000
1200
1400
1600
SO

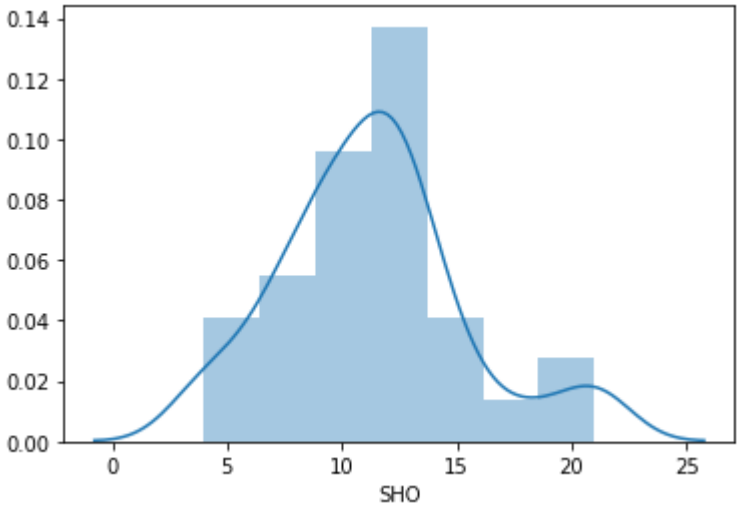

0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
20
40
60
80
100
120
140
160
SB

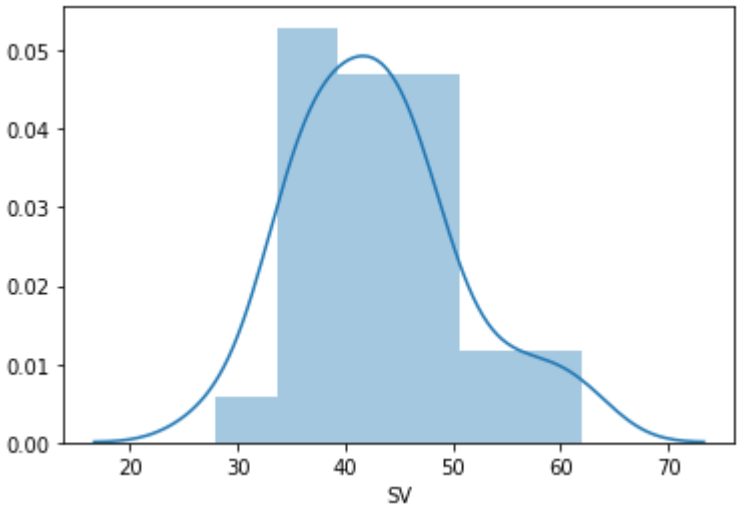

0.006
0.005
0.004
0.003
0.002
0.001
0.000
400
500
600
700
800
900
RA

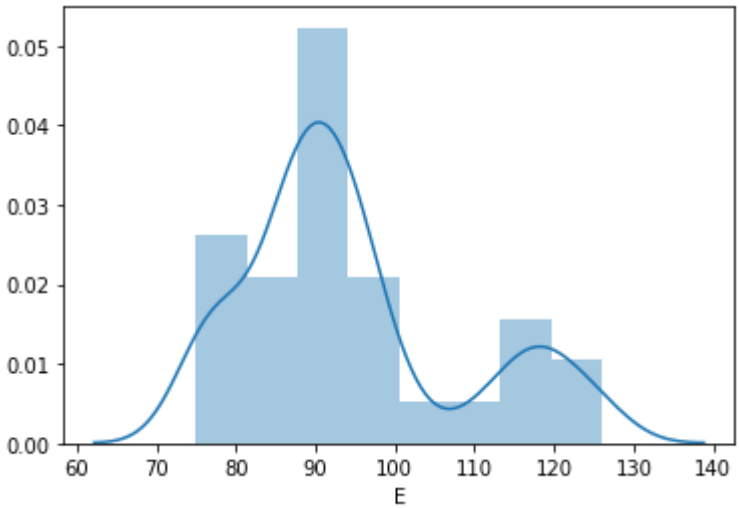

0.006
0.005
0.004
0.003
0.002
0.001
0.000
400
500
600
700
800
900
ER


0.8
0.6
0.4
0.2
0.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
ERA


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
CG


0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
SHO


0.05
0.04
0.03
0.02
0.01
0.00
20
30
40
50
60
70
SV

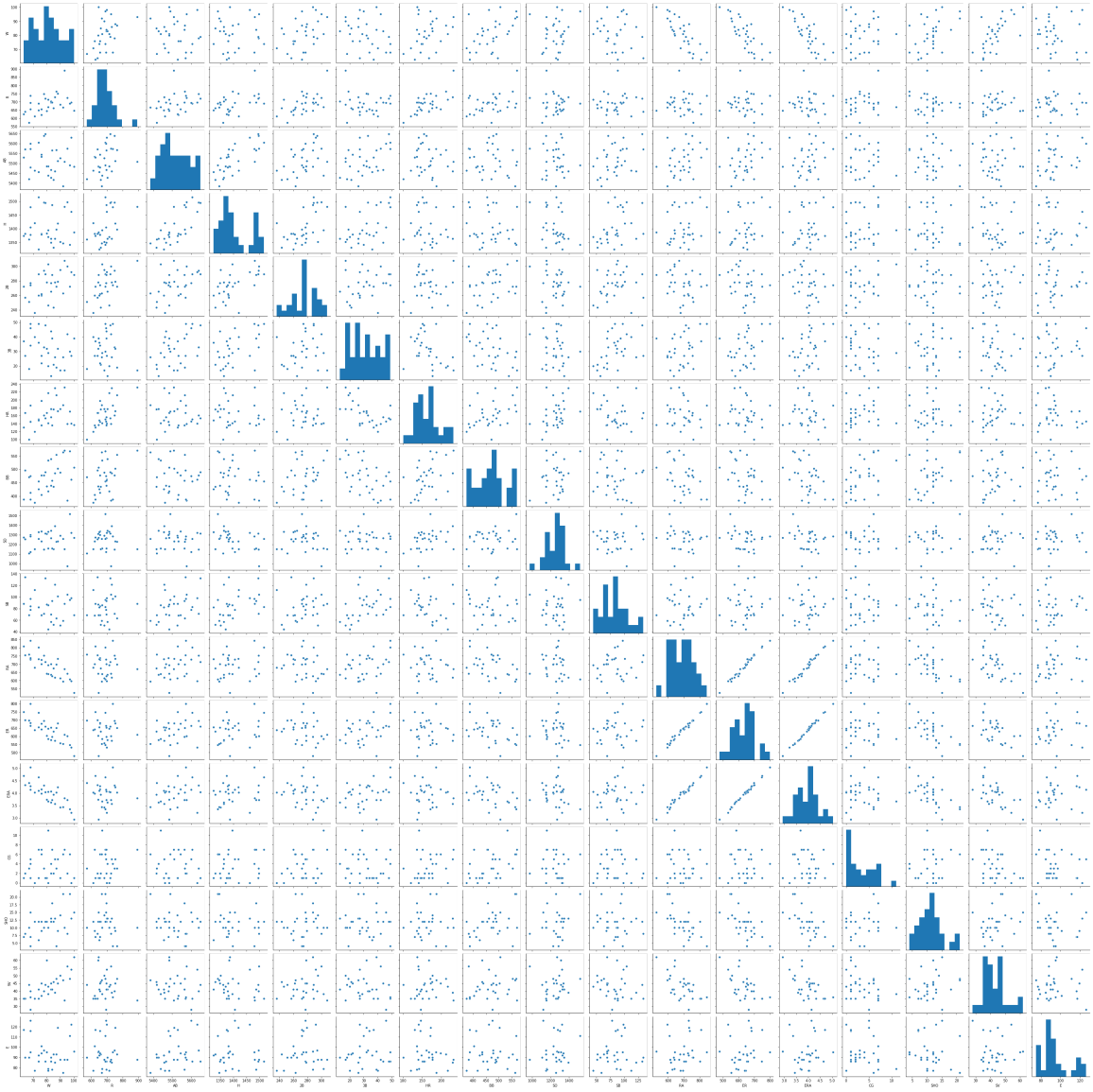

0.05
0.04
0.03
0.02
0.01
0.00
60
70
80
90
100
110
120
130
140
E

In [19]:

```
sns.pairplot(df)
```

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x14374283548>
```



Trying to find out best model for the dataset

In [20]:

```
from sklearn.impute import SimpleImputer
```

In [21]:

```
imputer = SimpleImputer(missing_values = np.nan , strategy = 'mean')
```

In [22]:

```
imputer = imputer.fit(df)
```

In [23]:

```
df = imputer.transform(df)
```

In [24]:

```
df
```

Out[24]:

```
array([[9.500e+01, 7.240e+02, 5.575e+03, 1.497e+03, 3.000e+02, 4.200e+01,
        1.390e+02, 3.830e+02, 9.730e+02, 1.040e+02, 6.410e+02, 6.010e+02,
        3.730e+00, 2.000e+00, 8.000e+00, 5.600e+01, 8.800e+01],
       [8.300e+01, 6.960e+02, 5.467e+03, 1.349e+03, 2.770e+02, 4.400e+01,
        1.560e+02, 4.390e+02, 1.264e+03, 7.000e+01, 7.000e+02, 6.530e+02,
        4.070e+00, 2.000e+00, 1.200e+01, 4.500e+01, 8.600e+01],
       [8.100e+01, 6.690e+02, 5.439e+03, 1.395e+03, 3.030e+02, 2.900e+01,
        1.410e+02, 5.330e+02, 1.157e+03, 8.600e+01, 6.400e+02, 5.840e+02,
        3.670e+00, 1.100e+01, 1.000e+01, 3.800e+01, 7.900e+01],
       [7.600e+01, 6.220e+02, 5.533e+03, 1.381e+03, 2.600e+02, 2.700e+01,
        1.360e+02, 4.040e+02, 1.231e+03, 6.800e+01, 7.010e+02, 6.430e+02,
        3.980e+00, 7.000e+00, 9.000e+00, 3.700e+01, 1.010e+02],
       [7.400e+01, 6.890e+02, 5.605e+03, 1.515e+03, 2.890e+02, 4.900e+01,
        1.510e+02, 4.550e+02, 1.259e+03, 8.300e+01, 8.030e+02, 7.460e+02,
        4.640e+00, 7.000e+00, 1.200e+01, 3.500e+01, 8.600e+01],
       [9.300e+01, 8.910e+02, 5.509e+03, 1.480e+03, 3.080e+02, 1.700e+01,
        2.320e+02, 5.700e+02, 1.151e+03, 8.800e+01, 6.700e+02, 6.090e+02,
        3.800e+00, 7.000e+00, 1.000e+01, 3.400e+01, 8.800e+01],
       [8.700e+01, 7.640e+02, 5.567e+03, 1.397e+03, 2.720e+02, 1.900e+01,
        2.120e+02, 5.540e+02, 1.227e+03, 6.300e+01, 6.980e+02, 6.520e+02,
        4.030e+00, 3.000e+00, 4.000e+00, 4.800e+01, 9.300e+01],
       [8.100e+01, 7.130e+02, 5.485e+03, 1.370e+03, 2.460e+02, 2.000e+01,
        2.170e+02, 4.180e+02, 1.331e+03, 4.400e+01, 6.930e+02, 6.460e+02,
        4.050e+00, 0.000e+00, 1.000e+01, 4.300e+01, 7.700e+01],
       [8.000e+01, 6.440e+02, 5.485e+03, 1.383e+03, 2.780e+02, 3.200e+01,
        1.670e+02, 4.360e+02, 1.310e+03, 8.700e+01, 6.420e+02, 6.040e+02,
        3.740e+00, 1.000e+00, 1.200e+01, 6.000e+01, 9.500e+01],
       [7.800e+01, 7.480e+02, 5.640e+03, 1.495e+03, 2.940e+02, 3.300e+01,
        1.610e+02, 4.780e+02, 1.148e+03, 7.100e+01, 7.530e+02, 6.940e+02,
        4.310e+00, 3.000e+00, 1.000e+01, 4.000e+01, 9.700e+01],
       [8.800e+01, 7.510e+02, 5.511e+03, 1.419e+03, 2.790e+02, 3.200e+01,
        1.720e+02, 5.030e+02, 1.233e+03, 1.010e+02, 7.330e+02, 6.800e+02,
        4.240e+00, 5.000e+00, 9.000e+00, 4.500e+01, 1.190e+02],
       [8.600e+01, 7.290e+02, 5.459e+03, 1.363e+03, 2.780e+02, 2.600e+01,
        2.300e+02, 4.860e+02, 1.392e+03, 1.210e+02, 6.180e+02, 5.720e+02,
        3.570e+00, 5.000e+00, 1.300e+01, 3.900e+01, 8.500e+01],
       [8.500e+01, 6.610e+02, 5.417e+03, 1.331e+03, 2.430e+02, 2.100e+01,
        1.760e+02, 4.350e+02, 1.150e+03, 5.200e+01, 6.750e+02, 6.300e+02,
        3.940e+00, 2.000e+00, 1.200e+01, 4.600e+01, 9.300e+01],
       [7.600e+01, 6.560e+02, 5.544e+03, 1.379e+03, 2.620e+02, 2.200e+01,
        1.980e+02, 4.780e+02, 1.336e+03, 6.900e+01, 7.260e+02, 6.770e+02,
        4.160e+00, 6.000e+00, 1.200e+01, 4.500e+01, 9.400e+01],
       [6.800e+01, 6.940e+02, 5.600e+03, 1.405e+03, 2.770e+02, 4.600e+01,
        1.460e+02, 4.750e+02, 1.119e+03, 7.800e+01, 7.290e+02, 6.640e+02,
        4.140e+00, 5.000e+00, 1.500e+01, 2.800e+01, 1.260e+02],
       [1.000e+02, 6.470e+02, 5.484e+03, 1.386e+03, 2.880e+02, 3.900e+01,
        1.370e+02, 5.060e+02, 1.267e+03, 6.900e+01, 5.250e+02, 4.780e+02,
        2.940e+00, 1.000e+00, 1.500e+01, 6.200e+01, 9.600e+01],
       [9.800e+01, 6.970e+02, 5.631e+03, 1.462e+03, 2.920e+02, 2.700e+01,
        1.400e+02, 4.610e+02, 1.322e+03, 9.800e+01, 5.960e+02, 5.320e+02,
        3.210e+00, 0.000e+00, 1.300e+01, 5.400e+01, 1.220e+02],
       [9.700e+01, 6.890e+02, 5.491e+03, 1.341e+03, 2.720e+02, 3.000e+01,
        1.710e+02, 5.670e+02, 1.518e+03, 9.500e+01, 6.080e+02, 5.460e+02,
        3.360e+00, 6.000e+00, 2.100e+01, 4.800e+01, 1.110e+02],
       [6.800e+01, 6.550e+02, 5.480e+03, 1.378e+03, 2.740e+02, 3.400e+01,
        1.450e+02, 4.120e+02, 1.299e+03, 8.400e+01, 7.370e+02, 6.820e+02,
        4.280e+00, 1.000e+00, 7.000e+00, 4.000e+01, 1.160e+02],
       [6.400e+01, 6.400e+02, 5.571e+03, 1.382e+03, 2.570e+02, 2.700e+01,
        1.670e+02, 4.960e+02, 1.255e+03, 1.340e+02, 7.540e+02, 7.000e+02,
```

```
 4.330e+00, 2.000e+00, 8.000e+00, 3.500e+01, 9.000e+01],
[9.000e+01, 6.830e+02, 5.527e+03, 1.351e+03, 2.950e+02, 1.700e+01,
 1.770e+02, 4.880e+02, 1.290e+03, 5.100e+01, 6.130e+02, 5.570e+02,
 3.430e+00, 1.000e+00, 1.400e+01, 5.000e+01, 8.800e+01],
[8.300e+01, 7.030e+02, 5.428e+03, 1.363e+03, 2.650e+02, 1.300e+01,
 1.770e+02, 5.390e+02, 1.344e+03, 5.700e+01, 6.350e+02, 5.770e+02,
 3.620e+00, 4.000e+00, 1.300e+01, 4.100e+01, 9.000e+01],
[7.100e+01, 6.130e+02, 5.463e+03, 1.420e+03, 2.360e+02, 4.000e+01,
 1.200e+02, 3.750e+02, 1.150e+03, 1.120e+02, 6.780e+02, 6.380e+02,
 4.020e+00, 0.000e+00, 1.200e+01, 3.500e+01, 7.700e+01],
[6.700e+01, 5.730e+02, 5.420e+03, 1.361e+03, 2.510e+02, 1.800e+01,
 1.000e+02, 4.710e+02, 1.107e+03, 6.900e+01, 7.600e+02, 6.980e+02,
 4.410e+00, 3.000e+00, 1.000e+01, 4.400e+01, 9.000e+01],
[6.300e+01, 6.260e+02, 5.529e+03, 1.374e+03, 2.720e+02, 3.700e+01,
 1.300e+02, 3.870e+02, 1.274e+03, 8.800e+01, 8.090e+02, 7.490e+02,
 4.690e+00, 1.000e+00, 7.000e+00, 3.500e+01, 1.170e+02],
[9.200e+01, 6.670e+02, 5.385e+03, 1.346e+03, 2.630e+02, 2.600e+01,
 1.870e+02, 5.630e+02, 1.258e+03, 5.900e+01, 5.950e+02, 5.530e+02,
 3.440e+00, 6.000e+00, 2.100e+01, 4.700e+01, 7.500e+01],
[8.400e+01, 6.960e+02, 5.565e+03, 1.486e+03, 2.880e+02, 3.900e+01,
 1.360e+02, 4.570e+02, 1.159e+03, 9.300e+01, 6.270e+02, 5.970e+02,
 3.720e+00, 7.000e+00, 1.800e+01, 4.100e+01, 7.800e+01],
[7.900e+01, 7.200e+02, 5.649e+03, 1.494e+03, 2.890e+02, 4.800e+01,
 1.540e+02, 4.900e+02, 1.312e+03, 1.320e+02, 7.130e+02, 6.590e+02,
 4.040e+00, 1.000e+00, 1.200e+01, 4.400e+01, 8.600e+01],
[7.400e+01, 6.500e+02, 5.457e+03, 1.324e+03, 2.600e+02, 3.600e+01,
 1.480e+02, 4.260e+02, 1.327e+03, 8.200e+01, 7.310e+02, 6.550e+02,
 4.090e+00, 1.000e+00, 6.000e+00, 4.100e+01, 9.200e+01],
[6.800e+01, 7.370e+02, 5.572e+03, 1.479e+03, 2.740e+02, 4.900e+01,
 1.860e+02, 3.880e+02, 1.283e+03, 9.700e+01, 8.440e+02, 7.990e+02,
 5.040e+00, 4.000e+00, 4.000e+00, 3.600e+01, 9.500e+01]])
```

In [25]:

```
df=pd.DataFrame(df)
df
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	95.0	724.0	5575.0	1497.0	300.0	42.0	139.0	383.0	973.0	104.0	641.0	601.0	3.73
1	83.0	696.0	5467.0	1349.0	277.0	44.0	156.0	439.0	1264.0	70.0	700.0	653.0	4.07
2	81.0	669.0	5439.0	1395.0	303.0	29.0	141.0	533.0	1157.0	86.0	640.0	584.0	3.67
3	76.0	622.0	5533.0	1381.0	260.0	27.0	136.0	404.0	1231.0	68.0	701.0	643.0	3.98
4	74.0	689.0	5605.0	1515.0	289.0	49.0	151.0	455.0	1259.0	83.0	803.0	746.0	4.64
5	93.0	891.0	5509.0	1480.0	308.0	17.0	232.0	570.0	1151.0	88.0	670.0	609.0	3.80
6	87.0	764.0	5567.0	1397.0	272.0	19.0	212.0	554.0	1227.0	63.0	698.0	652.0	4.03
7	81.0	713.0	5485.0	1370.0	246.0	20.0	217.0	418.0	1331.0	44.0	693.0	646.0	4.05
8	80.0	644.0	5485.0	1383.0	278.0	32.0	167.0	436.0	1310.0	87.0	642.0	604.0	3.74
9	78.0	748.0	5640.0	1495.0	294.0	33.0	161.0	478.0	1148.0	71.0	753.0	694.0	4.31
10	88.0	751.0	5511.0	1419.0	279.0	32.0	172.0	503.0	1233.0	101.0	733.0	680.0	4.24
11	86.0	729.0	5459.0	1363.0	278.0	26.0	230.0	486.0	1392.0	121.0	618.0	572.0	3.57
12	85.0	661.0	5417.0	1331.0	243.0	21.0	176.0	435.0	1150.0	52.0	675.0	630.0	3.94
13	76.0	656.0	5544.0	1379.0	262.0	22.0	198.0	478.0	1336.0	69.0	726.0	677.0	4.16
14	68.0	694.0	5600.0	1405.0	277.0	46.0	146.0	475.0	1119.0	78.0	729.0	664.0	4.14
15	100.0	647.0	5484.0	1386.0	288.0	39.0	137.0	506.0	1267.0	69.0	525.0	478.0	2.94
16	98.0	697.0	5631.0	1462.0	292.0	27.0	140.0	461.0	1322.0	98.0	596.0	532.0	3.21
17	97.0	689.0	5491.0	1341.0	272.0	30.0	171.0	567.0	1518.0	95.0	608.0	546.0	3.36
18	68.0	655.0	5480.0	1378.0	274.0	34.0	145.0	412.0	1299.0	84.0	737.0	682.0	4.28
19	64.0	640.0	5571.0	1382.0	257.0	27.0	167.0	496.0	1255.0	134.0	754.0	700.0	4.33
20	90.0	683.0	5527.0	1351.0	295.0	17.0	177.0	488.0	1290.0	51.0	613.0	557.0	3.43
21	83.0	703.0	5428.0	1363.0	265.0	13.0	177.0	539.0	1344.0	57.0	635.0	577.0	3.62
22	71.0	613.0	5463.0	1420.0	236.0	40.0	120.0	375.0	1150.0	112.0	678.0	638.0	4.02
23	67.0	573.0	5420.0	1361.0	251.0	18.0	100.0	471.0	1107.0	69.0	760.0	698.0	4.41
24	63.0	626.0	5529.0	1374.0	272.0	37.0	130.0	387.0	1274.0	88.0	809.0	749.0	4.69
25	92.0	667.0	5385.0	1346.0	263.0	26.0	187.0	563.0	1258.0	59.0	595.0	553.0	3.44
26	84.0	696.0	5565.0	1486.0	288.0	39.0	136.0	457.0	1159.0	93.0	627.0	597.0	3.72
27	79.0	720.0	5649.0	1494.0	289.0	48.0	154.0	490.0	1312.0	132.0	713.0	659.0	4.04
28	74.0	650.0	5457.0	1324.0	260.0	36.0	148.0	426.0	1327.0	82.0	731.0	655.0	4.09
29	68.0	737.0	5572.0	1479.0	274.0	49.0	186.0	388.0	1283.0	97.0	844.0	799.0	5.04

In [26]:

```
y=df.iloc[:,-1]
```

In [27]:

```
y.head()
```

Out[27]:

```
0     88.0
1     86.0
2     79.0
3    101.0
4     86.0
Name: 16, dtype: float64
```

In [28]:

```
y.shape
```

Out[28]:

```
(30,)
```

In [29]:

```
x=df.iloc[:,1:-1]
```

In [30]:

```
x.head()
```

Out[30]:

	1	2	3	4	5	6	7	8	9	10	11	12	13
0	724.0	5575.0	1497.0	300.0	42.0	139.0	383.0	973.0	104.0	641.0	601.0	3.73	2.0
1	696.0	5467.0	1349.0	277.0	44.0	156.0	439.0	1264.0	70.0	700.0	653.0	4.07	2.0
2	669.0	5439.0	1395.0	303.0	29.0	141.0	533.0	1157.0	86.0	640.0	584.0	3.67	11.0
3	622.0	5533.0	1381.0	260.0	27.0	136.0	404.0	1231.0	68.0	701.0	643.0	3.98	7.0
4	689.0	5605.0	1515.0	289.0	49.0	151.0	455.0	1259.0	83.0	803.0	746.0	4.64	7.0

In [31]:

```
x.shape
```

Out[31]:

```
(30, 15)
```

In [32]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=3)
```

In [33]:

```
x_train.shape
```

Out[33]:

```
(24, 15)
```

In [34]:

```
x_test.shape
```

Out[34]:

```
(6, 15)
```

In [35]:

```
y_train.shape
```

Out[35]:

```
(24,)
```

In [36]:

```
y_test.shape
```

Out[36]:

```
(6,)
```

In [37]:

```
lm=LinearRegression()
```

In [38]:

```
lm.fit(x_train,y_train)
```

Out[38]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [39]:

```
lm.score(x_train,y_train)
```

Out[39]:

```
0.7920203990990184
```

In [40]:

```
lm.coef_
```

Out[40]:

```
array([ 4.28881903e-01, -8.43590302e-02, -1.13517298e-01,  7.17414127e-02,
       -1.58776816e+00, -6.53256934e-01, -2.06097542e-01,  8.04403548e-02,
        2.76358079e-01, -6.93365763e-02,  1.83999038e+00, -2.44596021e+02,
        2.05121201e+00,  1.14041700e+00,  7.25432404e-01])
```

In [41]:

```
pred=lm.predict(x_test)
```

In [42]:

```
print('Predicted value and actual value is',pred,y_test)
```

```
Predicted value and actual value is [ 56.67756504 112.50638501  69.8323567
5  89.80111037  96.97239905
  62.19923832] 15     96.0
5      88.0
22     77.0
26     78.0
18    116.0
14    126.0
Name: 16, dtype: float64
```

In [46]:

```
from sklearn.metrics import mean_absolute_error
```

In [47]:

```
print('Mean absolute error:',mean_absolute_error(y_test,pred))
```

```
Mean absolute error: 27.604322703872015
```

In [48]:

```
print('Root mean squared error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Root mean squared error: 33.59078470339915
```

Conclusion: We have used LinearRegression in this dataset and achieve the accuracy score.it is also used for predicting the number of wins for a baseball team.