In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data =pd.read_csv("baseball.csv")
```

In [3]:
```
data
```

Out[3]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86
5	93	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34	88
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97
10	88	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45	119
11	86	729	5459	1363	278	26	230	486	1392	121	618	572	3.57	5	13	39	85
12	85	661	5417	1331	243	21	176	435	1150	52	675	630	3.94	2	12	46	93
13	76	656	5544	1379	262	22	198	478	1336	69	726	677	4.16	6	12	45	94
14	68	694	5600	1405	277	46	146	475	1119	78	729	664	4.14	5	15	28	126
15	100	647	5484	1386	288	39	137	506	1267	69	525	478	2.94	1	15	62	96
16	98	697	5631	1462	292	27	140	461	1322	98	596	532	3.21	0	13	54	122
17	97	689	5491	1341	272	30	171	567	1518	95	608	546	3.36	6	21	48	111
18	68	655	5480	1378	274	34	145	412	1299	84	737	682	4.28	1	7	40	116
19	64	640	5571	1382	257	27	167	496	1255	134	754	700	4.33	2	8	35	90
20	90	683	5527	1351	296	17	177	488	1290	51	613	557	3.43	1	14	50	88
21	83	703	5428	1363	265	13	177	539	1344	57	635	577	3.62	4	13	41	90
22	71	613	5463	1420	236	40	120	375	1150	112	678	638	4.02	0	12	35	77
23	67	573	5420	1361	251	18	100	471	1107	69	760	698	4.41	3	10	44	90
24	63	626	5529	1374	272	37	130	387	1274	88	809	749	4.69	1	7	35	117
25	92	687	5385	1346	263	26	187	563	1258	59	595	553	3.44	6	21	47	75
26	84	696	5565	1486	288	39	136	457	1159	93	627	597	3.72	7	18	41	78
27	79	720	5649	1494	289	48	154	490	1312	132	713	659	4.04	1	12	44	86
28	74	650	5457	1324	260	36	148	426	1327	82	731	655	4.09	1	6	41	92
29	68	737	5572	1479	274	49	186	388	1283	97	844	799	5.04	4	4	36	95

In [4]:
```
data.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 [5]:
```
data.isnull().sum()
```

Out[5]:
```
W      0
R      0
AB     0
H      0
2B     0
3B     0
HR     0
BB     0
SO     0
SB     0
RA     0
ER     0
ERA    0
CG     0
SHO    0
SV     0
E      0
dtype: int64
```

In [6]:
```
data.describe()
```

Out[6]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	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	469.100000	1248.200000	83.500000	688.233333	635.833333	3.956333	5.466667	11.300000	43.066667	94.333333
std	10.453455	58.761754	70.467372	57.140923	18.096405	10.452355	31.823309	57.053725	103.75947	22.815225	72.106005	70.140786	0.454089	2.763473	4.120177	7.669335	13.958889
min	63.000000	573.000000	5385.000000	1324.000000	236.000000	13.000000	100.000000	375.000000	973.000000	44.000000	525.000000	478.000000	2.944000	0.000000	4.000000	28.000000	75.000000
25%	74.000000	651.250000	5464.000000	1363.000000	262.250000	23.000000	140.250000	429.250000	1157.500000	69.000000	636.250000	587.250000	3.692500	1.000000	9.000000	37.250000	86.000000
50%	81.000000	689.000000	5510.000000	1382.500000	275.500000	31.000000	158.500000	473.000000	1261.500000	83.500000	695.500000	644.500000	4.025000	3.000000	12.000000	42.000000	91.000000
75%	87.750000	718.250000	5570.000000	1451.500000	288.750000	39.000000	177.000000	501.250000	1311.500000	96.500000	732.500000	679.250000	4.220000	5.750000	13.000000	46.750000	96.750000
max	100.000000	891.000000	5649.000000	1515.000000	308.000000	49.000000	232.000000	570.000000	1518.000000	134.000000	844.000000	799.000000	5.040000	11.000000	21.000000	62.000000	126.000000

In [7]:
```
data.columns
```

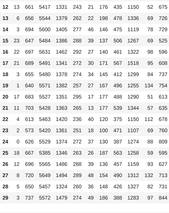
Out[7]:
```
Index(['W', 'R', 'AB', 'H', '2B', '3B', 'HR', 'BB', 'SO', 'SB', 'RA', 'ER',
       'ERA', 'CG', 'SHO', 'SV', 'E'],
      dtype='object')
```

In [8]:
```
data.shape
```

Out[8]:
```
(30, 17)
```

In [9]:
```
data.plot.bar()
plt.show()
```

In [10]:
```
sns.heatmap(data)
plt.show()
```



In [11]:
```
# Plotting distribution of wins
plt.hist(data['W'])
plt.xlabel('Wins')
plt.title('Distribution of Wins')

plt.show()
```

In [12]:
```
# Create scatter plots for runs per game vs. wins and runs allowed per game vs. wins
fig = plt.figure(figsize=(12, 6))

ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

ax1.scatter(data['R'], data['W'], c='pink')
ax1.set_title('Runs per Game vs. Wins')
ax1.set_ylabel('Wins')
ax1.set_xlabel('Runs per Game')

ax2.scatter(data['RA'], data['W'], c='purple')
ax2.set_title('Runs Allowed per Game vs. Wins')
ax2.set_xlabel('Runs Allowed per Game')

plt.show()
```


In [13]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
data['W']= label_encoder.fit_transform(data['W'])
data
```

Out[13]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	20	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	11	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	10	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	6	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	5	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86
5	19	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34	88
6	15	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	10	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	9	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	7	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97
10	16	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45	119
11	14	729	5459	1363	278	26	230	486	1392	121	618	572	3.57	5	13	39	85
12	13	661	5417	1331	243	21	176	435	1150	52	675	630	3.94	2	12	46	93
13	6	656	5544	1379	262	22	198	478	1336	69	726	677	4.16	6	12	45	94
14	3	694	5600	1405	277	46	146	475	1119	78	729	664	4.14	5	15	28	126
15	23	647	5484	1386	288	39	137	506	1267	69	525	478	2.94	1	15	62	96
16	22	697	5631	1462	292	27	140	461	1322	98	596	532	3.21	0	13	54	122
17	21	689	5491	1341	272	30	171	567	1518	95	608	546	3.36	6	21	48	111
18	3	655	5480	1378	274	34	145	412	1299	84	737	682	4.28	1	7	40	116
19	1	640	5571	1382	257	27	167	496	1255	134	754	700	4.33	2	8	35	90
20	17	683	5527	1351	295	17	177	488	1290	51	613	557	3.43	1	14	50	88
21	11	703	5428	1363	265	13	177	539	1344	57	635	577	3.62	4	13	41	90
22	4	613	5463	1420	236	40	120	375	1150	112	678	638	4.02	0	12	35	77
23	2	573	5420	1361	251	18	100	471	1107	69	760	698	4.41	3	10	44	90
24	0	626	5529	1374	272	37	130	387	1274	88	809	749	4.69	1	7	35	117
25	18	687	5385	1346	263	26	187	563	1258	59	595	553	3.44	6	21	47	75
26	12	696	5565	1486	288	39	136	457	1159	93	627	597	3.72	7	18	41	78
27	8	720	5649	1494	289	48	154	490	1312	132	713	659	4.04	1	12	44	86
28	5	650	5457	1324	260	36	148	426	1327	82	731	655	4.09	1	6	41	92
29	3	737	5572	1479	274	49	186	388	1283	97	844	799	5.04	4	4	36	95

In [17]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
import sklearn.metrics as metrics
import statsmodels.api as sm
from sklearn.model_selection import cross_val_score
```

In [15]:
```
# model fitting
from sklearn.model_selection import train_test_split

X=data.iloc[:,1:78]
y=data['W']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=50)
y_test = np.array(y_test,dtype = float)
```

In [18]:
```
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [19]:
```
def model_accuracy(model,X_train=X_train,y_trainy_train):
    accuracies = cross_val_score(estimator = model, X = X_train, y = y_train, cv = 10)
    print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
    print('Standard Deviation: {:.2f} %'.format(accuracies.std()*100))
```

In [22]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [25]:
```
rand_regressor = RandomForestRegressor()
rand_regressor.fit(X_train, y_train)
y_pred_rf = rand_regressor.predict(X_test)
LinearRegression(y_test, y_pred_rf)
model_accuracy(rand_regressor)
```

```
Accuracy: -887.66 %
Standard Deviation: 2345.01 %
```

In [ ]:
```

```