In [184]:
```python
print("Question:2\nDiscuss about the methods to handle missing values in data. In Table D2, there is a missing value indicated by ‘$’. How can you handle this missing value? Does Table D2 contain any noisy data? Explain your answer.")
```

```
Question:2
Discuss about the methods to handle missing values in data. In Table D2, there is a missing value indicated by ‘$’. How can you handle this missing value? Does Table D2 contain any noisy data? Explain your answer.
```

In [115]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
import scipy.stats as stats
sleep = pd.read_csv('sleep.csv')
```

In [116]:
```python
sleep
```

Out[116]:

	Sleep minute	Mood
0	40	22.0
1	50	20.0
2	60	35.0
3	70	30.0
4	70	45.0
5	75	30.0
6	80	45.0
7	80	35.0
8	80	5.0
9	90	30.0
10	30	10.0
11	50	50.0
12	55	35.0
13	80	40.0
14	65	30.0
15	95	60.0
16	65	40.0
17	90	50.0
18	50	30.0
19	60	30.0
20	90	80.0
21	90	80.0
22	95	30.0
23	80	80.0
24	75	35.0
25	95	70.0
26	50	NaN
27	55	30.0
28	75	60.0
29	80	65.0
30	70	30.0
31	30	20.0
32	70	40.0

In [117]:
```python
#finding out the missing values
sleep.isnull().sum()
```

Out[117]:
```
Sleep minute    0
Mood            1
dtype: int64
```

In [118]:
```python
#handling missing values(replacing with $ sign)
sleep[sleep['Mood'].isnull()]
```

Out[118]:

	Sleep minute	Mood
26	50	NaN

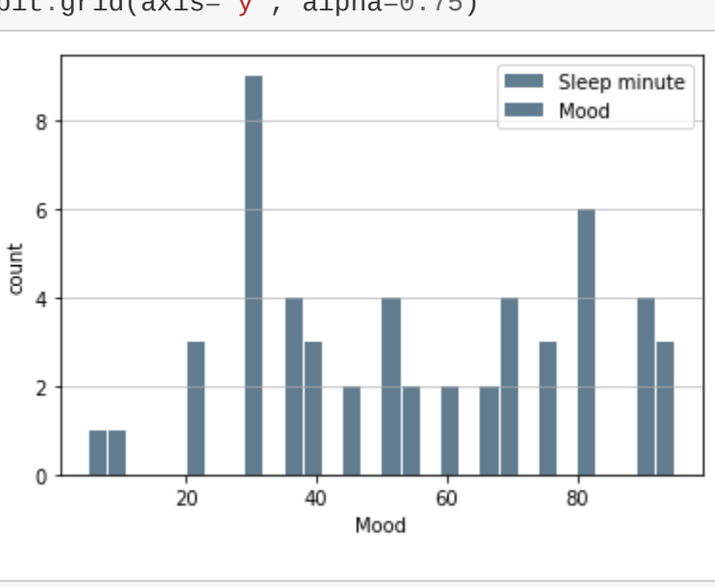
In [119]:
```python
dollar=sleep.fillna(value='$')
dollar
```

Out[119]:

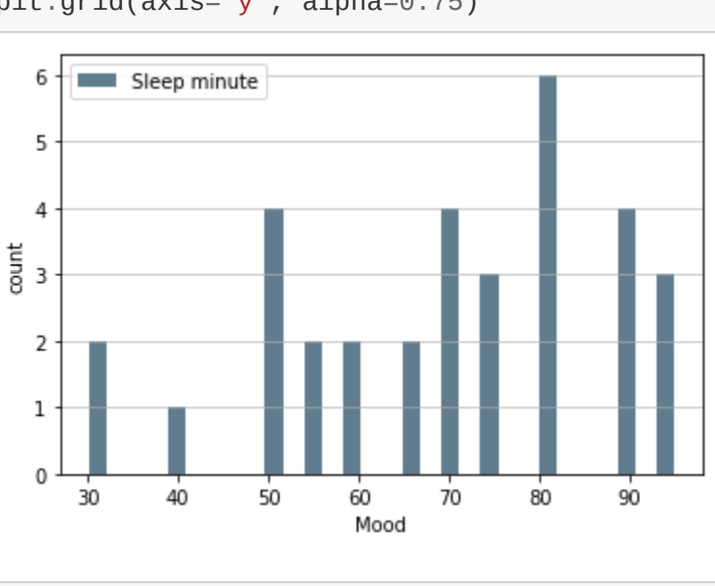
	Sleep minute	Mood
0	40	22
1	50	20
2	60	35
3	70	30
4	70	45
5	75	30
6	80	45
7	80	35
8	80	5
9	90	30
10	30	10
11	50	50
12	55	35
13	80	40
14	65	30
15	95	60
16	65	40
17	90	50
18	50	30
19	60	30
20	90	80
21	90	80
22	95	30
23	80	80
24	75	35
25	95	70
26	50	$
27	55	30
28	75	60
29	80	65
30	70	30
31	30	20
32	70	40

In [95]:
```python
#dollar = pd.Series(np.random.gamma(scale, size=size) ** 1.5)

sleep.plot.hist(grid=False, bins=30, rwidth=0.9,
                   color='#607c8e')
#plt.title('Commute Times for 1,000 Commuters')
plt.xlabel('Mood')
plt.ylabel('count')
plt.grid(axis='y', alpha=0.75)
```



In [96]:
```python
dollar.plot.hist(grid=False, bins=30, rwidth=0.9,
                   color='#607c8e')
#plt.title('Commute Times for 1,000 Commuters')
plt.xlabel('Mood')
plt.ylabel('count')
plt.grid(axis='y', alpha=0.75)
```



In [103]:
```python
#So imputing with the value 30 (mode) is the best choice in this case and final data
finalSleep=sleep.fillna(value='30')
```

In [124]:
```python
finalSleep=finalSleep.drop_duplicates()
finalSleep
```

Out[124]:

	Sleep minute	Mood
0	40	22.0
1	50	20.0
2	60	35.0
3	70	30.0
4	70	45.0
5	75	30.0
6	80	45.0
7	80	35.0
8	80	5.0
9	90	30.0
10	30	10.0
11	50	50.0
12	55	35.0
13	80	40.0
14	65	30.0
15	95	60.0
16	65	40.0
17	90	50.0
18	50	30.0
19	60	30.0
20	90	80.0
22	95	30.0
23	80	80.0
24	75	35.0
25	95	70.0
27	55	30.0
28	75	60.0
29	80	65.0
31	30	20.0
32	70	40.0

In [125]:
```python
#Handling noisy data

finalSleep.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30 entries, 0 to 32
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Sleep minute  30 non-null     int64
 1   Mood          30 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 720.0 bytes
```

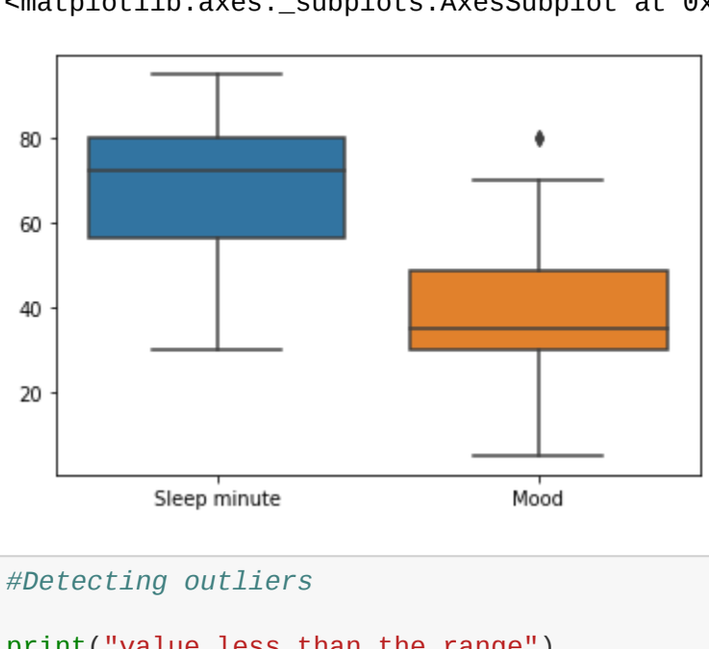
In [126]:
```python
finalSleep['Mood']=pd.to_numeric(finalSleep['Mood'])
```

In [108]:
```python
finalSleep['Mood']
```

Out[108]:
```
0     22.0
1     20.0
2     35.0
3     30.0
4     45.0
5     30.0
6     45.0
7     35.0
8      5.0
9     30.0
10    10.0
11    50.0
12    35.0
13    40.0
14    30.0
15    60.0
16    40.0
17    50.0
18    30.0
19    30.0
20    80.0
21    80.0
22    30.0
23    80.0
24    35.0
25    70.0
26    30.0
27    30.0
28    60.0
29    65.0
30    30.0
31    20.0
32    40.0
Name: Mood, dtype: float64
```

In [127]:
```python
sns.boxplot(data =finalSleep)
```

Out[127]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2308ebc24c8>
```



In [129]:
```python
#Detecting outliers

print("value less than the range")

finalSleep[finalSleep['Mood']<10]
```

```
value less than the range
```

Out[129]:

	Sleep minute	Mood
8	80	5.0

In [132]:
```python
print("value greater than the range")

finalSleep[finalSleep['Mood']>70]
```

```
value greater than the range
```

Out[132]:

	Sleep minute	Mood
20	90	80.0
23	80	80.0

In [134]:
```python
print("3 noisy instances found: (80,5), (90,80), (80,80)")
```

```
3 noisy instances found: (80,5), (90,80), (80,80)
```

In [136]:
```python
finalSleep = finalSleep[(finalSleep['Mood']>=10) & (finalSleep['Mood']<=70)]
finalSleep.shape
```

Out[136]:
```
(27, 2)
```

In [137]:
```python
print("Final DataSet after handling Noisy Data, dropping duplicates and missing value solving")
```

```
Final DataSet after handling Noisy Data, dropping duplicates and missing value solving
```

In [138]:
```python
finalSleep
```

Out[138]:

	Sleep minute	Mood
0	40	22.0
1	50	20.0
2	60	35.0
3	70	30.0
4	70	45.0
5	75	30.0
6	80	45.0
7	80	35.0
9	90	30.0
10	30	10.0
11	50	50.0
12	55	35.0
13	80	40.0
14	65	30.0
15	95	60.0
16	65	40.0
17	90	50.0
18	50	30.0
19	60	30.0
22	95	30.0
24	75	35.0
25	95	70.0
27	55	30.0
28	75	60.0
29	80	65.0
31	30	20.0
32	70	40.0

In [139]:
```python
#Question 4
#In Table D3 data, you want to apply K-NN to find out whether a customer has a car or not based on data ‘age’ and ‘income’. For this, you have to apply normalization techniques on these attributes. Apply ‘min-max’ and ‘z-score’ normalization on ‘age’ and ‘income’
```

In [142]:
```python
print("Question 4: ")
print("In Table D3 data, you want to apply K-NN to find out whether a customer has a car or not based on data ‘age’ and ‘income’. For this, you have to apply normalization techniques on these attributes. Apply ‘min-max’ and ‘z-score’ normalization on ‘age’ and ‘income’")
```

```
Question 4:
In Table D3 data, you want to apply K-NN to find out whether a customer has a car or not based on data ‘age’ and ‘income’. For this, you have to apply normalization techniques on these attributes. Apply ‘min-max’ and ‘z-score’ normalization on ‘age’ and ‘income’
```

In [143]:
```python
bank = pd.read_csv('bank.csv')
```

In [144]:
```python
bank
```

Out[144]:

	Age	Gender	Income	Married	Car
0	40	M	160	Y	Y
1	50	M	160	Y	N
2	47	F	260	Y	Y
3	66	F	590	Y	N
4	39	F	290	N	Y
5	51	F	160	Y	Y
6	54	M	380	Y	N
7	20	M	130	N	Y
8	52	F	260	N	Y
9	45	M	230	Y	Y
10	58	M	240	Y	Y
11	22	M	120	N	Y
12	47	F	170	Y	Y
13	36	M	190	Y	N
14	33	F	290	N	Y
15	37	F	250	Y	Y
16	56	M	410	Y	Y
17	28	M	230	Y	N
18	38	F	220	Y	Y
19	19	M	100	Y	Y
20	54	M	240	Y	Y
21	45	M	200	Y	N
22	47	M	270	Y	N
23	31	M	220	Y	Y
24	43	F	180	Y	Y

In [145]:
```python
#performing feature selection
bank = bank.drop(['Gender','Married'],axis =1)
```

In [146]:
```python
bank
```

Out[146]:

	Age	Income	Car
0	40	160	Y
1	50	160	N
2	47	260	Y
3	66	590	N
4	39	290	Y
5	51	160	Y
6	54	380	N
7	20	130	Y
8	52	260	Y
9	45	230	Y
10	58	240	Y
11	22	120	Y
12	47	170	Y
13	36	190	N
14	33	290	Y
15	37	250	Y
16	56	410	Y
17	28	230	N
18	38	220	Y
19	19	100	Y
20	54	240	Y
21	45	200	N
22	47	270	N
23	31	220	Y
24	43	180	Y

In [154]:
```python
#MIN-MAX Normalization

print("MIN-MAX Normalization")
```

```
MIN-MAX Normalization
```

In [148]:
```python
def normalize(feature):
    min = np.min(feature)
    max = np.max(feature)
    range = max - min

    return [(a - min) / range for a in feature]
```

In [149]:
```python
#copy the dataframe
bank_min_max_norm = bank.copy()
```

In [151]:
```python
bank_min_max_norm['Age'] = normalize(bank_min_max_norm['Age'])
```

In [152]:
```python
bank_min_max_norm['Income'] = normalize(bank_min_max_norm['Income'])
```

In [153]:
```python
#Min Max Normalized dataset
bank_min_max_norm.head()
```

Out[153]:

	Age	Income	Car
0	0.446809	0.122449	Y
1	0.659574	0.122449	N
2	0.595745	0.326531	Y
3	1.000000	1.000000	N
4	0.425532	0.387755	Y

In [155]:
```python
print("Z-Score Normalization")
```

```
Z-Score Normalization
```

In [157]:
```python
# we will use the scipy.stats package for the z-score calculation
bank_zscore_norm = bank.copy()
```

In [158]:
```python
bank_zscore_norm['Age'] = stats.zscore(bank_zscore_norm['Age'])
bank_zscore_norm['Income'] = stats.zscore(bank_zscore_norm['Income'])
```

In [159]:
```python
#Z-score normalized dataset

bank_zscore_norm.head()
```

Out[159]:

	Age	Income	Car
0	-0.194596	-0.769903	Y
1	0.644179	-0.769903	N
2	0.392547	0.217152	Y
3	1.986219	3.474436	N
4	-0.278473	0.513269	Y

In [160]:
```python
#Question 6
#You decide to survey 30 randomly selected NSTU students about their sleep habits (in minutes) and their mood. Their mood is rated on a scale from 1 to 100, with 1 being very sad and 100 being very happy. The data is shown in Table D2. Build a ‘statistical learning’ model to predict someone’s mood from their sleeping habits. Using that model you have to predict a student’s mood with sleeping habits of 60.

#We've already analyzed the Table D2 data in question 2. So, we'll use the cleaned data from question 2

print("Question 6\nYou decide to survey 30 randomly selected NSTU students about their sleep habits (in minutes) and their mood. Their mood is rated on a scale from 1 to 100, with 1 being very sad and 100 being very happy. The data is shown in Table D2. Build a ‘statistical learning’ model to predict someone’s mood from their sleeping habits. Using that model you have to predict a student’s mood with sleeping habits of 60.\nWe've already analyzed the Table D2 data in question 2. So, we'll use the cleaned data from question 2")
```

```
Question 6
You decide to survey 30 randomly selected NSTU students about their sleep habits (in minutes) and their mood. Their mood is rated on a scale from 1 to 100, with 1 being very sad and 100 being very happy. The data is shown in Table D2. Build a ‘statistical learning’ model to predict someone’s mood from their sleeping habits. Using that model you have to predict a student’s mood with sleeping habits of 60.
We've already analyzed the Table D2 data in question 2. So, we'll use the cleaned data from question 2
```

In [161]:
```python
sleep
```

Out[161]:

	Sleep minute	Mood
0	40	22.0
1	50	20.0
2	60	35.0
3	70	30.0
4	70	45.0
5	75	30.0
6	80	45.0
7	80	35.0
8	80	5.0
9	90	30.0
10	30	10.0
11	50	50.0
12	55	35.0
13	80	40.0
14	65	30.0
15	95	60.0
16	65	40.0
17	90	50.0
18	50	30.0
19	60	30.0
20	90	80.0
21	90	80.0
22	95	30.0
23	80	80.0
24	75	35.0
25	95	70.0
26	50	NaN
27	55	30.0
28	75	60.0
29	80	65.0
30	70	30.0
31	30	20.0
32	70	40.0

In [164]:
```python
#Handle missing instances using constant 30. Details are in the Answer of Question 2
sleep.fillna(value='30')
```

Out[164]:

	Sleep minute	Mood
0	40	22
1	50	20
2	60	35
3	70	30
4	70	45
5	75	30
6	80	45
7	80	35
8	80	5
9	90	30
10	30	10
11	50	50
12	55	35
13	80	40
14	65	30
15	95	60
16	65	40
17	90	50
18	50	30
19	60	30
20	90	80
21	90	80
22	95	30
23	80	80
24	75	35
25	95	70
26	50	30
27	55	30
28	75	60
29	80	65
30	70	30
31	30	20
32	70	40

In [168]:
```python
sleep = sleep.sample(n=30, replace = True)
```

In [167]:
```python
#randomly select 30 data
```

In [169]:
```python
sleep.shape
```

Out[169]:
```
(30, 2)
```

In [171]:
```python
#  x and y
X = sleep[['Sleep minute']]
y = sleep['Mood']
```

In [172]:
```python
#Linear Regression Application
LR = LinearRegression()
```

In [176]:
```python
LR.fit(X,y)
```

Out[176]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [177]:
```python
LR.score(X,y)
```

Out[177]:
```
0.7270831920287949
```

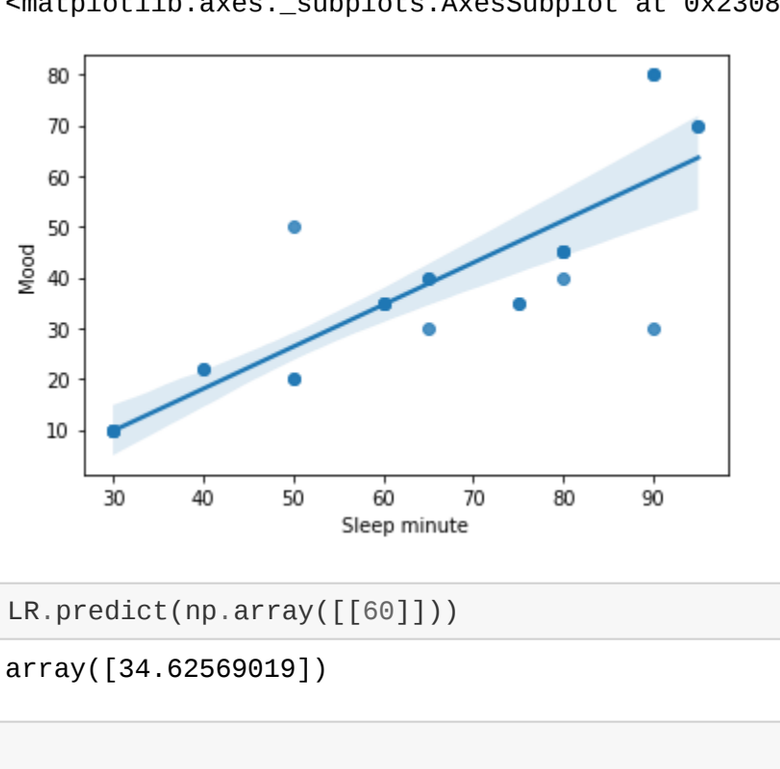
In [178]:
```python
LR.coef_
```

Out[178]:
```
array([0.82808082])
```

In [179]:
```python
#regression plot
```

In [181]:
```python
sns.regplot(x = 'Sleep minute',y = 'Mood', data =sleep )
```

Out[181]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2308ed9b288>
```



In [182]:
```python
LR.predict(np.array([[60]]))
```

Out[182]:
```
array([34.62569019])
```

In [ ]: