In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import seaborn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Read and describing dataset

In [43]:

```
dataset= pd.read_csv('iotprojectdataset.csv')
print(dataset.shape)
```

```
(366, 3)
```

In [44]:

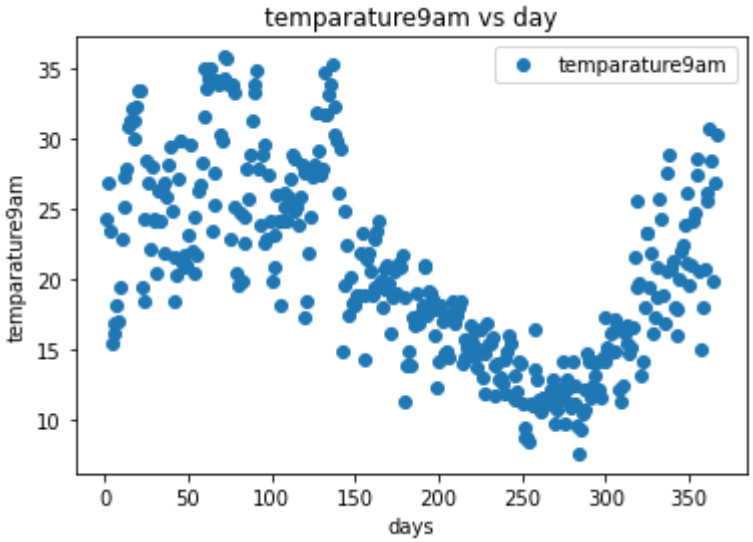
```
print(dataset.describe())
```

```
              day  temparature9am  Humidity9am
count  366.000000      366.000000   366.000000
mean   183.500000       20.550273    72.035519
std    105.799338        6.690516    13.137058
min      1.000000        7.600000    36.000000
25%     92.250000       15.025000    64.000000
50%    183.500000       19.650000    72.000000
75%    274.750000       25.500000    81.000000
max    366.000000       35.800000    99.000000
```

Temparature vs Day plot

In [45]:

```
dataset.plot(x='day', y='temparature9am' , style='o')
plt.title('temparature9am vs day')
plt.xlabel('days')
plt.ylabel('temparature9am')
plt.show()
```



In [47]:

```
X= dataset['day'].values.reshape(-1,1)
y= dataset['temparature9am'].values.reshape(-1,1)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, random_state=0)
```

Temperature Prediction using Linear Regression

In [48]:

```
model =LinearRegression()
model.fit(X_train,y_train)
```

Out[48]:

```
LinearRegression()
```

In [49]:

```
print('Intercept is :',model.intercept_)
```

```
Intercept is : [26.58996609]
```

In [50]:

```
print('Coefficient is :' ,model.coef_)
```

```
Coefficient is : [[-0.03263307]]
```

In [51]:

```
y_pred= model.predict(X_test)
```

In [52]:

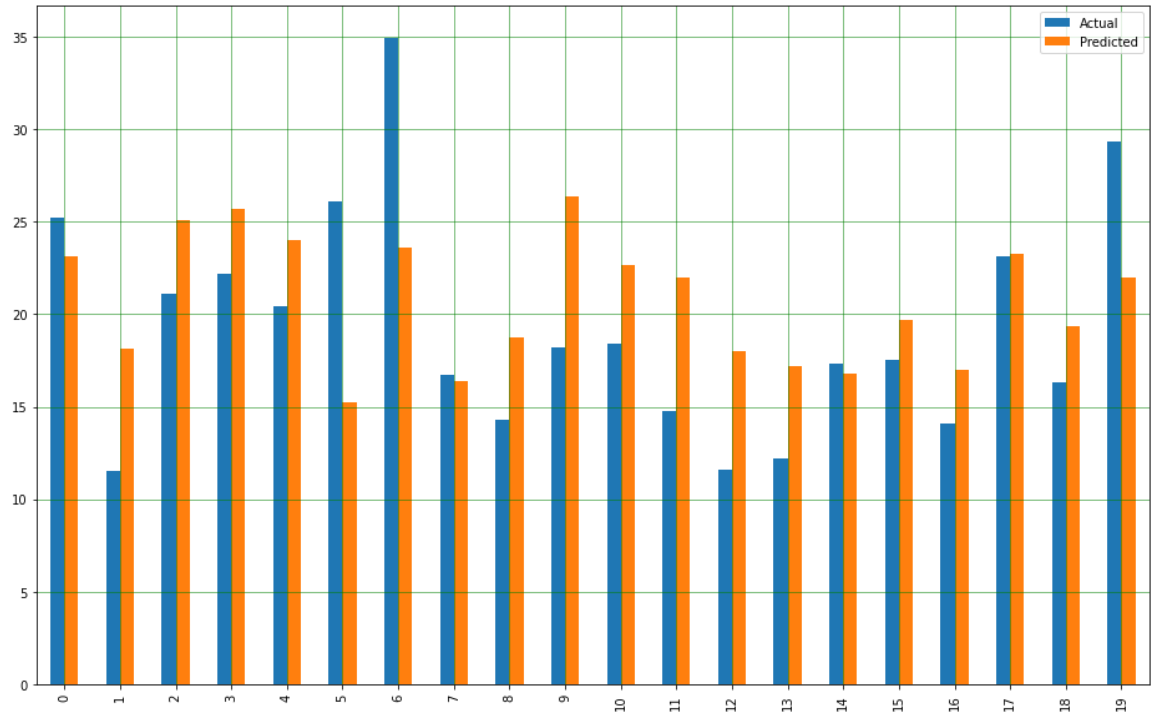
```
df=  pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
print(df)
```

```
    Actual  Predicted
0     25.2  23.098228
1     11.5  18.105368
2     21.1  25.088845
3     22.2  25.708873
4     20.4  24.011954
..     ...        ...
69    18.9  21.368675
70    22.8  24.142486
71    16.1  21.009711
72    25.1  22.837163
73    12.2  16.930577

[74 rows x 2 columns]
```

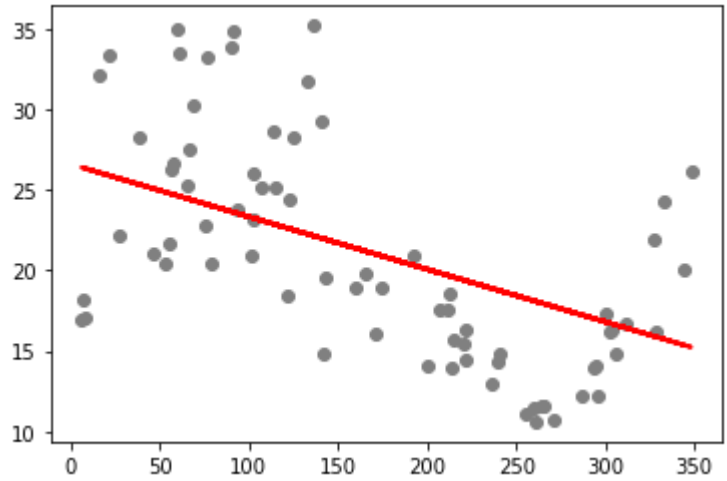
In [53]:

```
df1= df.head(20)
df1.plot(kind='bar', figsize=(16,10))
plt.grid(which='major', linestyle='-',linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':',linewidth='0.5', color='black')
plt.show()
```



In [54]:

```
plt.scatter(X_test,y_test,color='gray')
plt.plot(X_test,y_pred,color='red',linewidth=2)
plt.show()
```



In [55]:

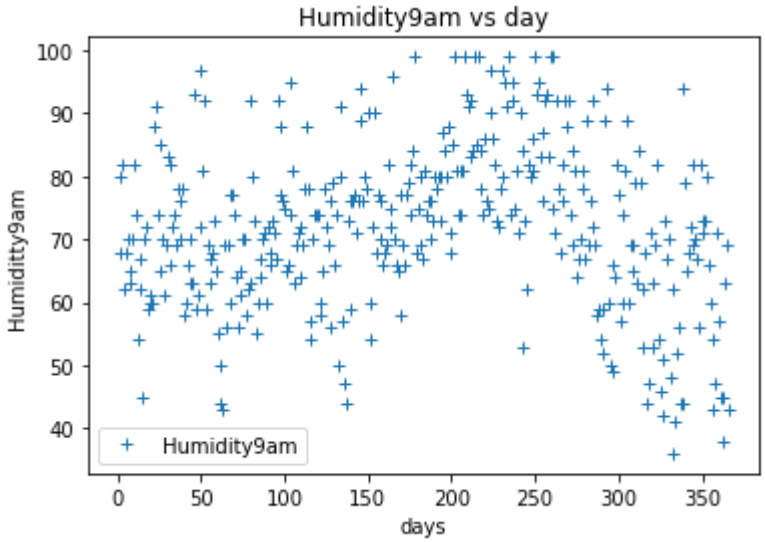
```
print('Mean abolute error is:', metrics.mean_absolute_error(y_test,y_pred))
print('Mean squared error is:', metrics.mean_squared_error(y_test,y_pred))
print('Root mean squared error is:', np.sqrt(metrics.mean_squared_error(y_test,y_pred
)))
```

```
Mean abolute error is: 4.628743431864956
Mean squared error is: 31.068282617526176
Root mean squared error is: 5.573892949952141
```

Humidity vs days plot

In [56]:

```
dataset.plot(x='day', y='Humidity9am' , style='+')
plt.title('Humidity9am vs day')
plt.xlabel('days')
plt.ylabel('Humiditty9am')
plt.show()
```



In [58]:

```
X1= dataset['day'].values.reshape(-1,1)
y1= dataset['Humidity9am'].values.reshape(-1,1)
X1_train,X1_test,y1_train,y1_test = train_test_split(X1,y1,test_size=0.2, random_state=
0)
```

Humidity Prediction using Linear Regression

In [59]:

```
model1 =LinearRegression()
model1.fit(X1_train,y1_train)
```

Out[59]:

```
LinearRegression()
```

In [60]:

```
print('Intercept is :',model1.intercept_)
```

```
Intercept is : [73.90598946]
```

In [61]:

```
print('Coefficient is :' ,model1.coef_)
```

```
Coefficient is : [[-0.00857091]]
```

In [62]:

```
y1_pred= model1.predict(X1_test)
```

In [63]:

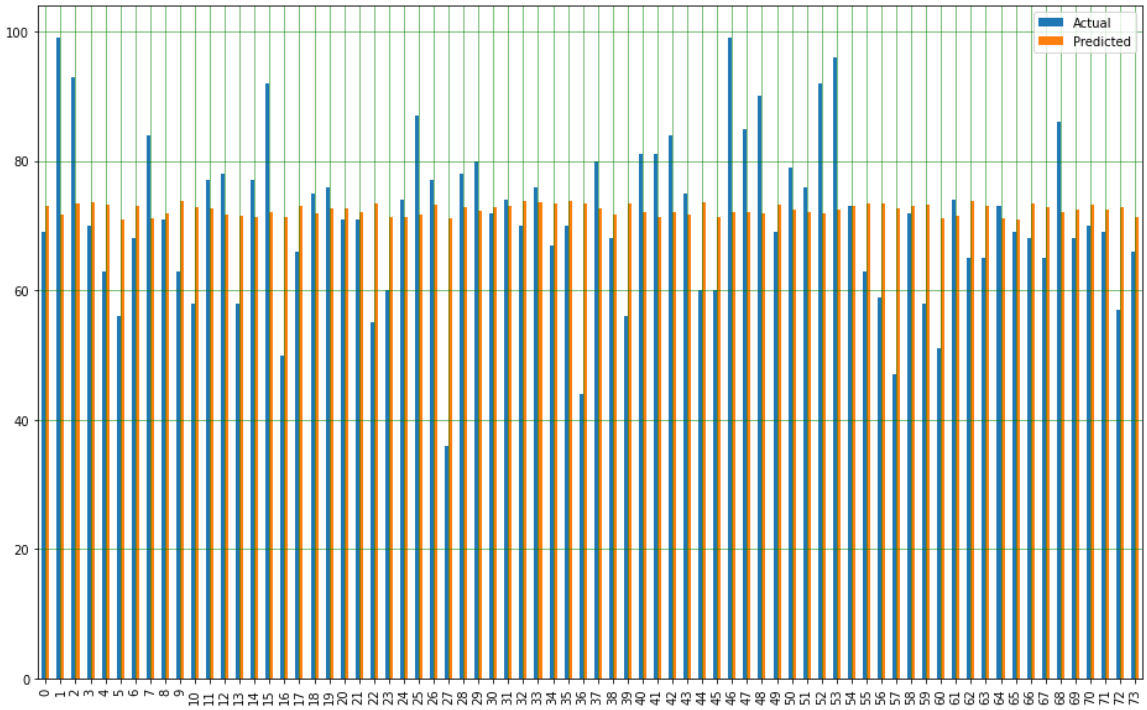
```
df2=  pd.DataFrame({'Actual': y1_test.flatten(), 'Predicted': y1_pred.flatten()})
print(df2)
```

```
    Actual  Predicted
0       69  72.988902
1       99  71.677552
2       93  73.511727
3       70  73.674575
4       63  73.228887
..     ...        ...
69      68  72.534643
70      70  73.263171
71      69  72.440363
72      57  72.920334
73      66  71.368999

[74 rows x 2 columns]
```

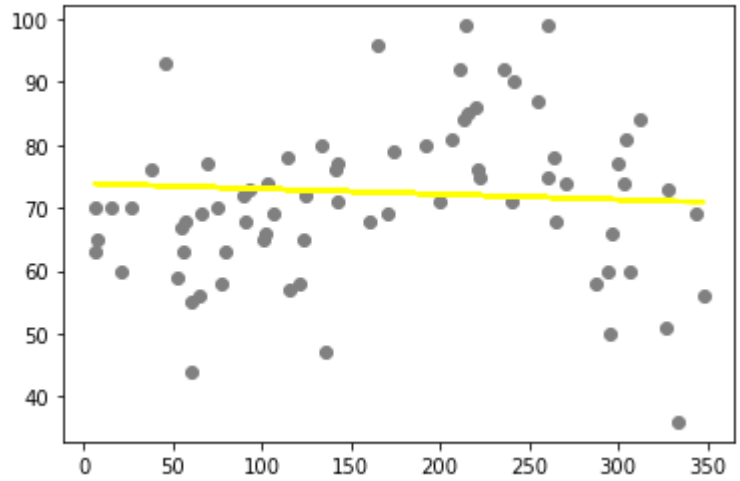
In [65]:

```
df3= df2.head(20)
df2.plot(kind='bar', figsize=(16,10))
plt.grid(which='major', linestyle='-',linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':',linewidth='0.5', color='black')
plt.show()
```



In [66]:

```
plt.scatter(X1_test,y1_test,color='gray')
plt.plot(X1_test,y1_pred,color='yellow',linewidth=2)
plt.show()
```



In [67]:

```
print('Mean abolute error is:', metrics.mean_absolute_error(y1_test,y1_pred))
print('Mean squared error is:', metrics.mean_squared_error(y1_test,y1_pred))
print('Root mean squared error is:', np.sqrt(metrics.mean_squared_error(y1_test,y1_pred
)))
```

```
Mean abolute error is: 9.892821949213232
Mean squared error is: 159.99725178076565
Root mean squared error is: 12.649002007303409
```