In [20]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
```

In [2]:
```python
dfgpa = pd.read_csv("D:\\24 - Machine_Learning\\download files\\LR1.csv")
dfgpa
```

Out[2]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83
...	...	...
79	1936	3.71
80	1810	3.71
81	1987	3.73
82	1962	3.76
83	2050	3.81

84 rows × 2 columns

In [4]:
```python
dfgpa.head(10)
```

Out[4]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83
5	1670	2.91
6	1764	3.00
7	1764	3.00
8	1792	3.01
9	1850	3.01

In [5]:
```python
dfgpa.describe()
```

Out[5]:

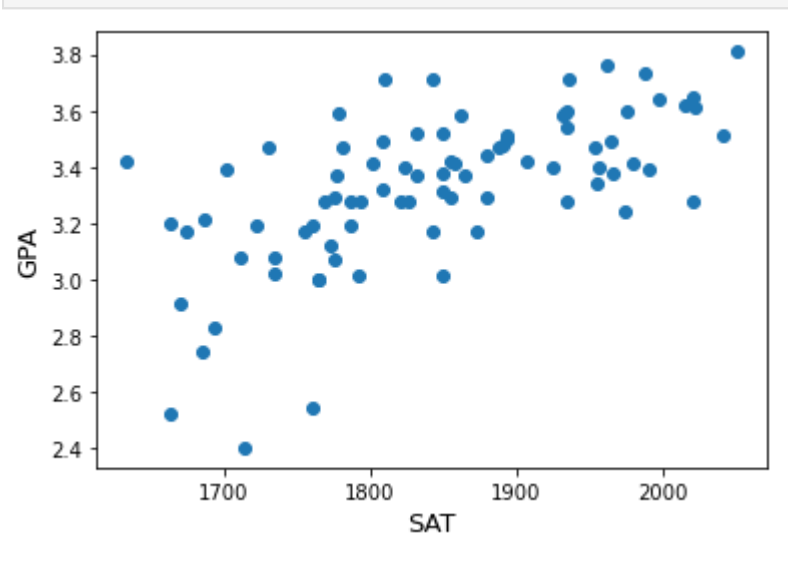
	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

In [7]:
```python
dfgpa.shape
```

Out[7]:
```
(84, 2)
```

In [8]:
```python
x = dfgpa['SAT']
y = dfgpa['GPA']
```

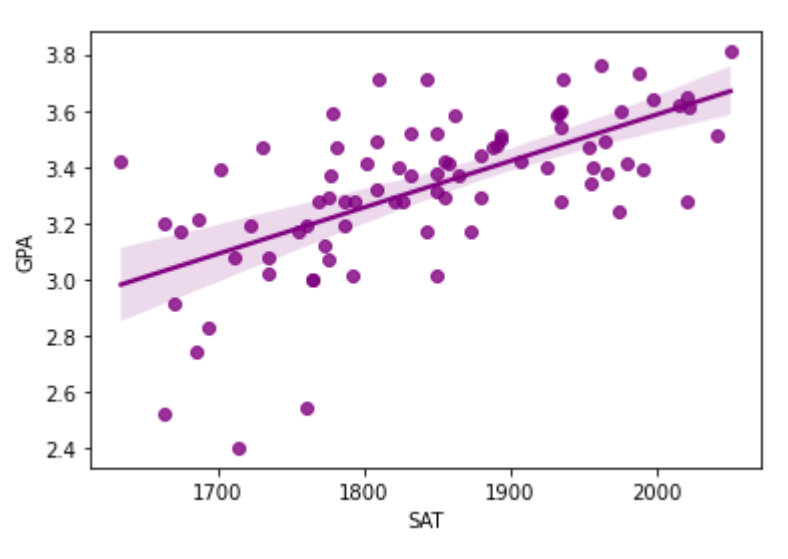
In [9]:
```python
plt.scatter(x, y)
plt.xlabel('SAT', fontsize = 13)
plt.ylabel('GPA', fontsize = 13)
plt.show()
```



In [11]:
```python
sns.regplot(x,y,color='purple')
```

```
D:\24-Annaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

Out[11]:
```
<AxesSubplot:xlabel='SAT', ylabel='GPA'>
```



In [15]:
```python
x.shape
```

Out[15]:
```
(84,)
```

In [12]:
```python
x.head()
```

Out[12]:
```
0    1714
1    1664
2    1760
3    1685
4    1693
Name: SAT, dtype: int64
```

In [13]:
```python
X=x.values.reshape(-1, 1)
# to 1D to 2D
```

In [14]:
```python
X.shape
```

Out[14]:
```
(84, 1)
```

In [16]:
```python
x
```

Out[16]:
```
0     1714
1     1664
2     1760
3     1685
4     1693
      ...
79    1936
80    1810
81    1987
82    1962
83    2050
Name: SAT, Length: 84, dtype: int64
```

In [17]:
```python
X
```

Out[17]:
```
array([[1714],
       [1664],
       [1760],
       [1685],
       [1693],
       [1670],
       [1764],
       [1764],
       [1792],
       [1850],
       [1735],
       [1775],
       [1735],
       [1712],
       [1773],
       [1872],
       [1755],
       [1674],
       [1842],
       [1786],
       [1761],
       [1722],
       [1663],
       [1687],
       [1974],
       [1826],
       [1787],
       [1821],
       [2020],
       [1794],
       [1769],
       [1934],
       [1775],
       [1855],
       [1880],
       [1849],
       [1808],
       [1954],
       [1777],
       [1831],
       [1865],
       [1850],
       [1966],
       [1702],
       [1990],
       [1925],
       [1824],
       [1956],
       [1857],
       [1979],
       [1802],
       [1855],
       [1907],
       [1634],
       [1879],
       [1887],
       [1730],
       [1953],
       [1781],
       [1891],
       [1964],
       [1808],
       [1893],
       [2041],
       [1893],
       [1832],
       [1850],
       [1934],
       [1861],
       [1931],
       [1933],
       [1778],
       [1975],
       [1934],
       [2021],
       [2015],
       [1997],
       [2020],
       [1843],
       [1936],
       [1810],
       [1987],
       [1962],
       [2050]], dtype=int64)
```

In [22]:
```python
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=41)
```

In [23]:
```python
X_train.shape
```

Out[23]:
```
(67, 1)
```

In [24]:
```python
X_test.shape
```

Out[24]:
```
(17, 1)
```

In [25]:
```python
LR = LinearRegression()
LR.fit(X_train, y_train)
# fit is used for training the data and it is supervised learning
```

Out[25]:
```
LinearRegression()
```

In [ ]:
```python
# So here our MODEL is created and ready for testing
```

In [26]:
```python
y_pred = LR.predict(X_test)
# predict function is used for testing
```

In [27]:
```python
y_test
```

Out[27]:
```
75    3.62
24    3.24
67    3.54
54    3.44
32    3.29
6     3.00
42    3.38
78    3.71
43    3.39
0     2.40
2     2.54
69    3.58
45    3.40
37    3.34
38    3.37
27    3.28
48    3.41
Name: GPA, dtype: float64
```

In [28]:
```python
y_pred
# used for getting the predicted value
```

Out[28]:
```
array([3.61756384, 3.55190026, 3.48783823, 3.39975294, 3.23319167,
       3.21557461, 3.53908785, 3.34209712, 3.11627847, 3.13549708,
       3.20916841, 3.48303358, 3.47342428, 3.51986925, 3.23639477,
       3.306863  , 3.36451883])
```

In [29]:
```python
acc = mean_squared_error(y_test, y_pred)
acc
# here we calulcating the error i.e. difference between actual value and predicted value
```

Out[29]:
```
0.08490837191499061
```

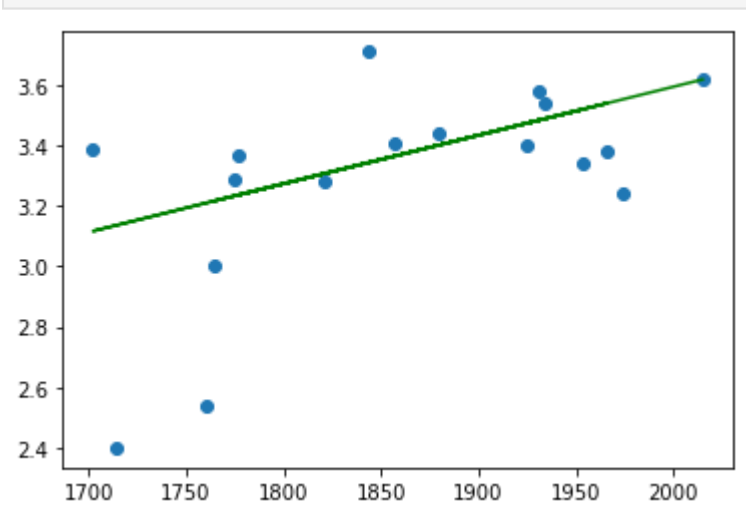
In [30]:
```python
weights = LR.coef_
intercept = LR.intercept_
print(weights, intercept)
```

```
[0.00160155] 0.3904391936980436
```

In [31]:
```python
LR.score(X_test, y_test)
```

Out[31]:
```
0.2594693604787399
```

In [32]:
```python
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred, color='green')
plt.show()
```

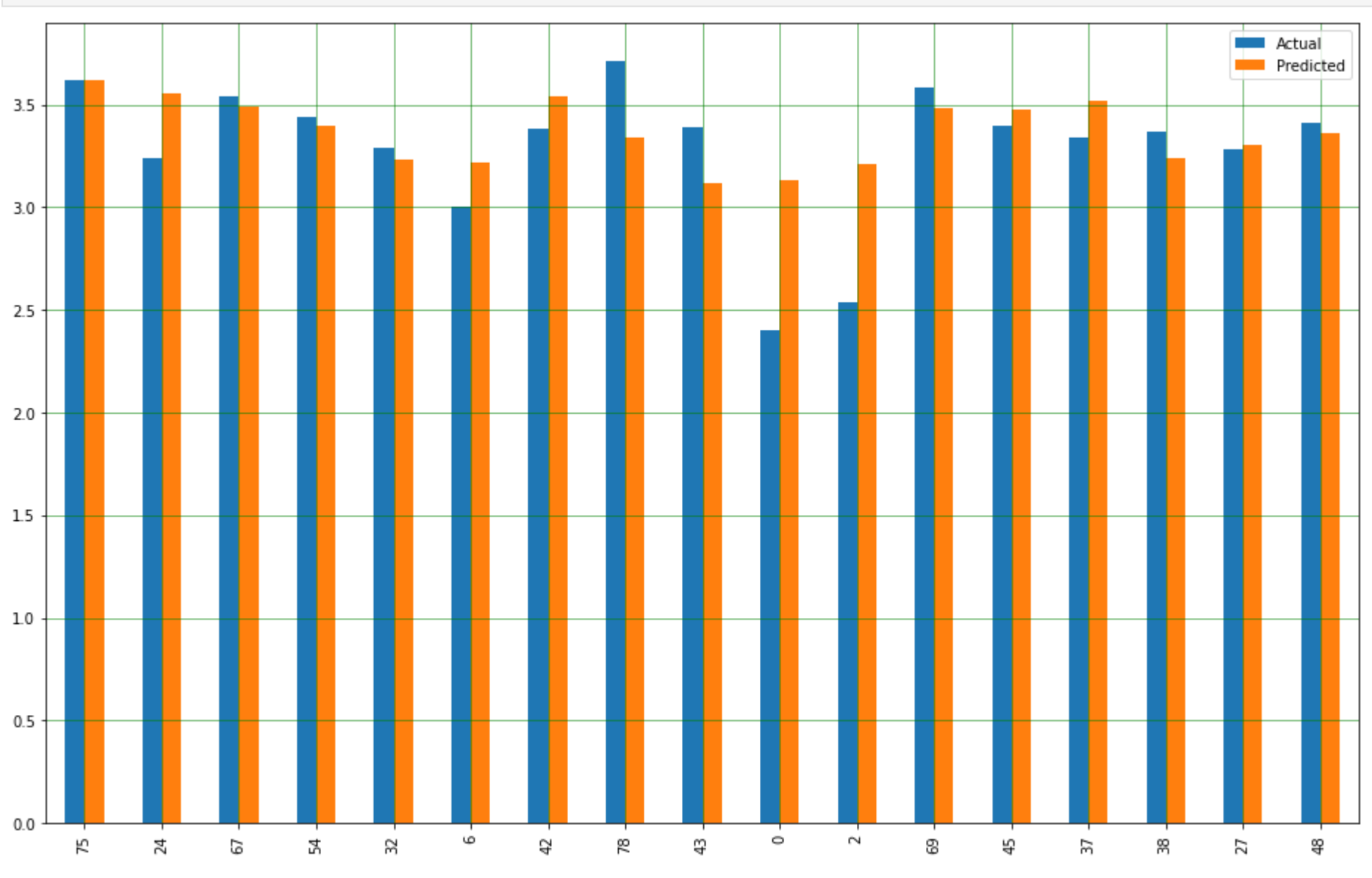


In [33]:
```python
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[33]:

	Actual	Predicted
75	3.62	3.617564
24	3.24	3.551900
67	3.54	3.487838
54	3.44	3.399753
32	3.29	3.233192
6	3.00	3.215575
42	3.38	3.539088
78	3.71	3.342097
43	3.39	3.116278
0	2.40	3.135497
2	2.54	3.209168
69	3.58	3.483034
45	3.40	3.473424
37	3.34	3.519869
38	3.37	3.236395
27	3.28	3.306863
48	3.41	3.364519

In [34]:
```python
df1 = df.head(25)
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 [35]:
```python
new_data = pd.DataFrame([2115,900])
new_data
```

Out[35]:

	0
0	2115
1	900

In [36]:
```python
LR.predict(new_data)
```

Out[36]:
```
array([3.77771891, 1.83183482])
```

In [37]:
```python
LR.predict([[4562]])
```

Out[37]:
```
array([7.69671345])
```

In [38]:
```python
print('Mean Absolute Error: ',mean_absolute_error(y_test,y_pred))
print('Mean Squared Error: ',mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error: ',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error:  0.20239501683340122
Mean Squared Error:  0.08490837191499061
Root Mean Squared Error:  0.2913904115014607
```

In [39]:
```python
dfgpa['GPA'].mean()
```

Out[39]:
```
3.330238095238094
```

In [41]:
```python
plt.scatter(x,y)
yhat = LR.coef_*x+LR.intercept_
fig = plt.plot(x, yhat, lw=2, c='orange', label='Regression Line')
plt.xlabel('SAT', fontsize='12')
plt.ylabel('GPA', fontsize='12')
plt.show()
```