In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
student_data = pd.read_csv("student_clustering.csv")
```

In [3]:

```
student_data.head()
```

Out[3]:

	cgpa	iq
0	5.13	88
1	5.90	113
2	8.36	93
3	8.27	97
4	5.45	110

In [4]:

```
student_data.tail()
```

Out[4]:

	cgpa	iq
195	4.68	89
196	8.57	118
197	5.85	112
198	6.23	108
199	8.82	117

In [5]:

```
student_data.shape
```

Out[5]:

```
(200, 2)
```

In [6]:

```
student_data.columns
```

Out[6]:

```
Index(['cgpa', 'iq'], dtype='object')
```

In [7]:

```
student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   cgpa    200 non-null    float64
 1   iq      200 non-null    int64  
dtypes: float64(1), int64(1)
memory usage: 3.2 KB
```

In [8]:

```
student_data.describe()
```

Out[8]:

	cgpa	iq
count	200.000000	200.000000
mean	6.983400	101.995000
std	1.624101	12.161599
min	4.600000	83.000000
25%	5.407500	91.000000
50%	7.040000	102.000000
75%	8.585000	113.000000
max	9.300000	121.000000

In [9]:

```
student_data.isnull().sum()
```

Out[9]:

```
cgpa    0
iq      0
dtype: int64
```

In [15]:

```
student_data.iq.value_counts()
```

Out[15]:

```
117    15
118    13
88     12
109    11
86     11
108    10
111    10
96      9
87      9
93      8
85      8
94      8
116     7
110     7
95      7
119     6
91      5
92      5
115     5
98      5
113     4
112     4
89      4
97      3
83      3
107     2
106     2
121     1
114     1
104     1
90      1
100     1
84      1
120     1
Name: iq, dtype: int64
```

In [18]:

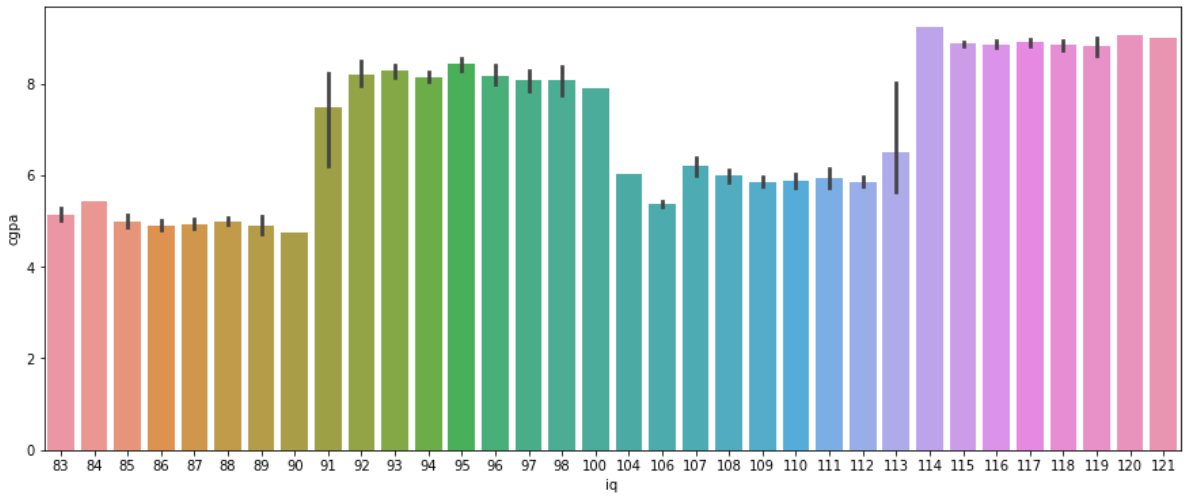
```
student_data.cgpa.value_counts()
```

Out[18]:

```
8.91    4
5.01    3
4.86    3
4.78    3
8.97    3
       ..
8.88    1
8.12    1
5.14    1
8.45    1
8.82    1
Name: cgpa, Length: 152, dtype: int64
```

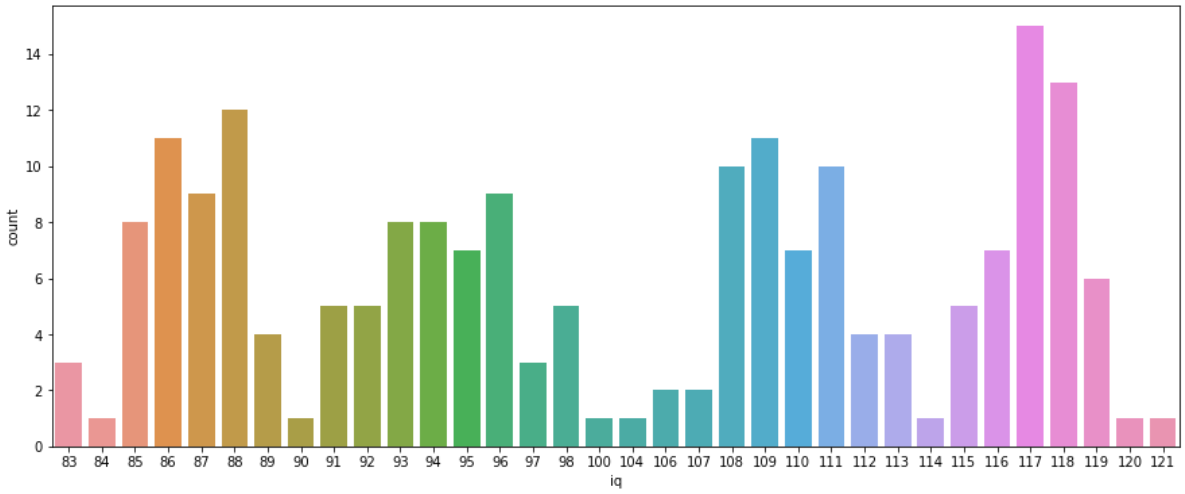
In [21]:

```
plt.figure(figsize=(15,6))
sns.barplot(x = 'iq', y = 'cgpa', data = student_data)
plt.xticks(rotation = 0)
plt.show()
```



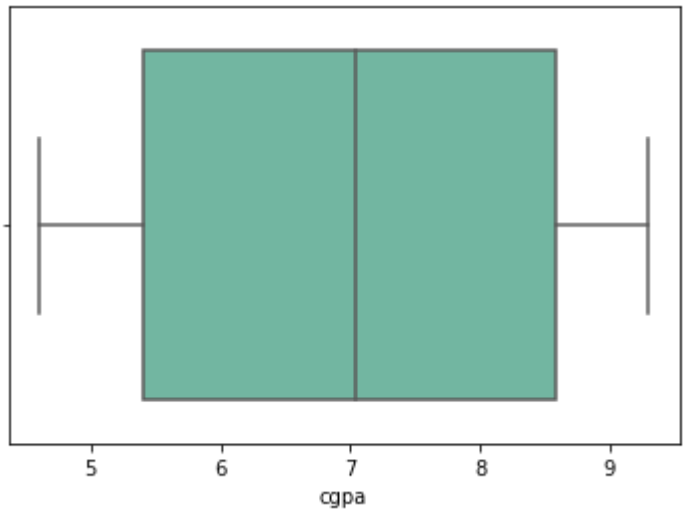
In [11]:

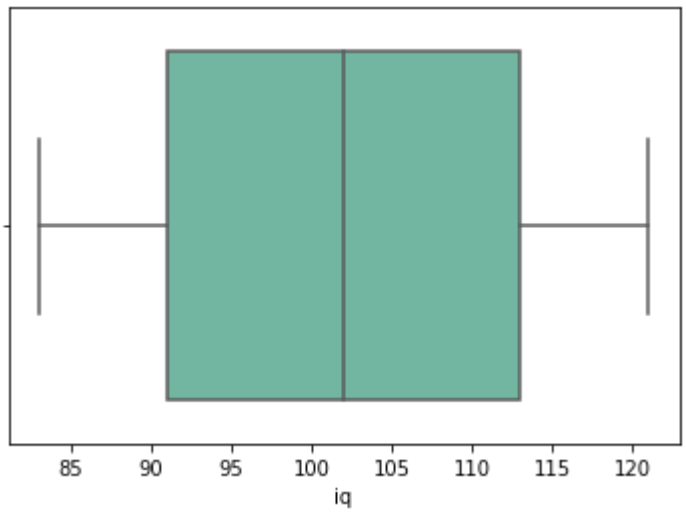
```
plt.figure(figsize=(15,6))
sns.countplot(x = 'iq', data = student_data)
plt.xticks(rotation = 0)
plt.show()
```



In [19]:

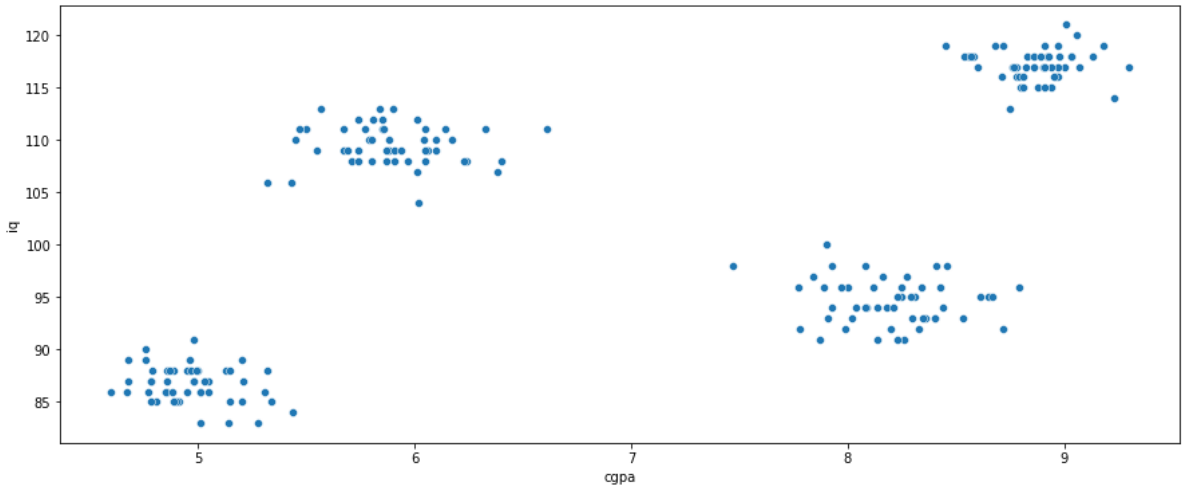
```
for i in student_data.columns:
    sns.boxplot(x=student_data[i], orient = 'h', palette = 'Set2')
    plt.show()
```





In [22]:

```
plt.figure(figsize=(15,6))
sns.scatterplot(x = 'cgpa', y = 'iq', data = student_data)
plt.xticks(rotation = 0)
plt.show()
```



In [23]:

```
from sklearn.cluster import KMeans
```

In [24]:

```
data = []
for i in range(1,11):
    km = KMeans(n_clusters=i)
    km.fit_predict(student_data)
    data.append(km.inertia_)
```

In [25]:

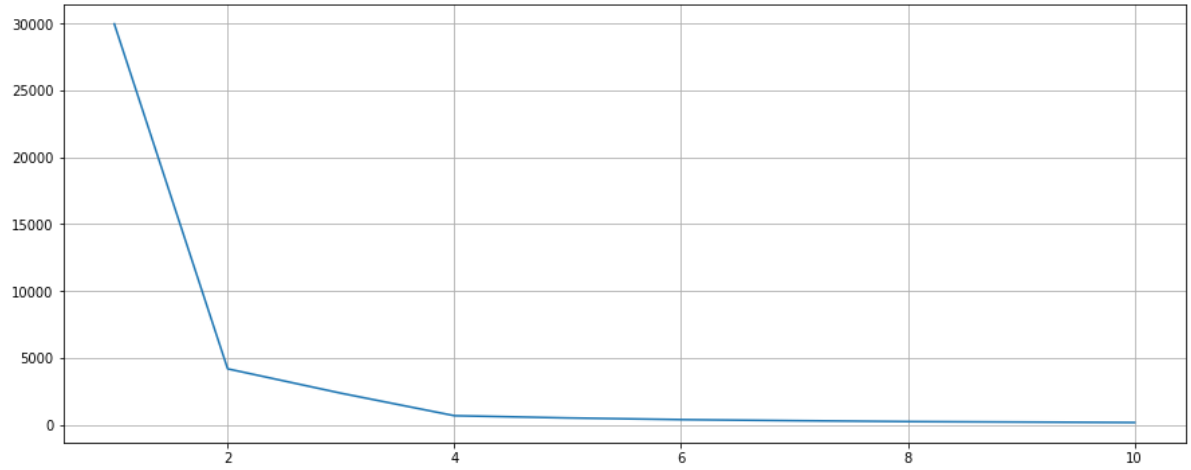
```
data
```

Out[25]:

```
[29957.898288,
 4184.14127,
 2362.7133489999997,
 681.96966,
 514.1616803171115,
 388.8524026875981,
 310.6394834520718,
 243.3997975201465,
 206.93812936151184,
 171.56716356743664]
```

In [32]:

```
plt.figure(figsize=(15,6))
plt.plot(range(1,11),data)
plt.grid()
plt.xticks(rotation = 0)
plt.show()
```



In [27]:

```
x = student_data.iloc[:,:].values
km = KMeans(n_clusters = 5)
y_means = km.fit_predict(x)
```

In [28]:

```
y_means
```

Out[28]:

```
array([2, 3, 4, 1, 3, 3, 1, 0, 3, 4, 2, 3, 1, 2, 3, 1, 3, 4, 3, 3, 1, 2,
       4, 2, 2, 1, 2, 0, 4, 3, 0, 3, 0, 3, 1, 1, 0, 3, 2, 3, 2, 1, 4, 2,
       0, 0, 4, 3, 0, 3, 2, 2, 0, 1, 0, 3, 3, 0, 3, 0, 3, 1, 1, 0, 2, 0,
       4, 2, 3, 1, 3, 0, 4, 2, 3, 0, 3, 0, 2, 4, 4, 0, 3, 2, 0, 2, 0, 3,
       0, 3, 0, 0, 4, 2, 1, 4, 0, 1, 4, 0, 3, 2, 2, 0, 2, 2, 4, 2, 0, 0,
       4, 0, 3, 3, 4, 0, 1, 3, 0, 2, 2, 3, 4, 0, 1, 2, 1, 3, 2, 1, 4, 3,
       2, 2, 3, 0, 3, 2, 1, 1, 4, 2, 3, 2, 2, 0, 2, 0, 3, 2, 0, 2, 0, 0,
       2, 1, 3, 0, 3, 4, 2, 0, 3, 4, 0, 2, 3, 2, 2, 0, 0, 3, 0, 2, 2, 1,
       0, 3, 2, 0, 0, 3, 3, 3, 1, 2, 4, 4, 0, 3, 4, 4, 2, 2, 1, 2, 0, 3,
       3, 0])
```

In [29]:

```
x[y_means==0]
```

Out[29]:

```
array([[  8.8 , 115.  ],
       [  9.18, 119.  ],
       [  8.86, 117.  ],
       [  8.83, 118.  ],
       [  8.56, 118.  ],
       [  8.96, 116.  ],
       [  8.78, 116.  ],
       [  8.45, 119.  ],
       [  8.79, 116.  ],
       [  8.81, 115.  ],
       [  8.88, 115.  ],
       [  9.07, 117.  ],
       [  8.92, 118.  ],
       [  8.75, 113.  ],
       [  8.71, 116.  ],
       [  8.86, 118.  ],
       [  9.3 , 117.  ],
       [  9.01, 121.  ],
       [  8.97, 116.  ],
       [  9.  , 117.  ],
       [  8.76, 117.  ],
       [  8.78, 117.  ],
       [  9.23, 114.  ],
       [  9.03, 118.  ],
       [  9.13, 118.  ],
       [  8.91, 119.  ],
       [  8.98, 118.  ],
       [  9.03, 118.  ],
       [  8.86, 117.  ],
       [  8.89, 118.  ],
       [  8.97, 117.  ],
       [  8.72, 119.  ],
       [  8.93, 118.  ],
       [  8.58, 118.  ],
       [  8.94, 117.  ],
       [  8.6 , 117.  ],
       [  8.77, 117.  ],
       [  8.81, 116.  ],
       [  8.54, 118.  ],
       [  8.97, 119.  ],
       [  8.91, 117.  ],
       [  8.68, 119.  ],
       [  9.06, 120.  ],
       [  8.9 , 117.  ],
       [  8.94, 115.  ],
       [  8.91, 115.  ],
       [  8.91, 117.  ],
       [  8.95, 116.  ],
       [  8.57, 118.  ],
       [  8.82, 117.  ]])
```

In [30]:

```
x[y_means==0,0],x[y_means==0,1]
```

Out[30]:

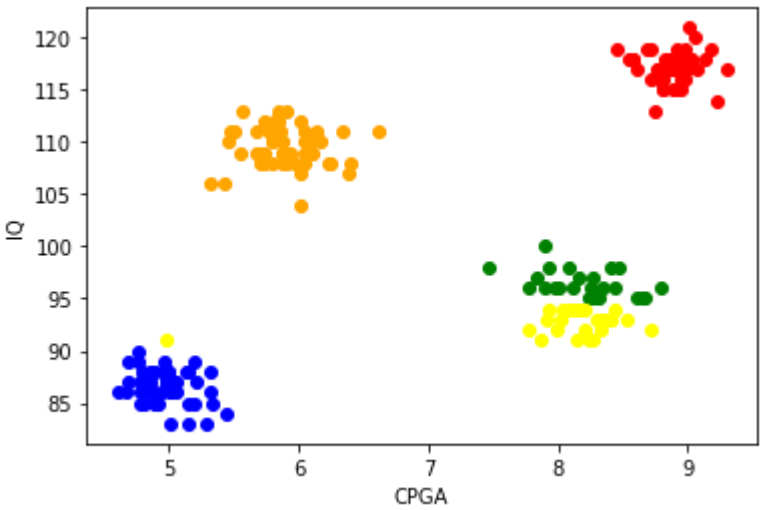
```
(array([8.8 , 9.18, 8.86, 8.83, 8.56, 8.96, 8.78, 8.45, 8.79, 8.81, 8.88,
        9.07, 8.92, 8.75, 8.71, 8.86, 9.3 , 9.01, 8.97, 9.  , 8.76, 8.78,
        9.23, 9.03, 9.13, 8.91, 8.98, 9.03, 8.86, 8.89, 8.97, 8.72, 8.93,
        8.58, 8.94, 8.6 , 8.77, 8.81, 8.54, 8.97, 8.91, 8.68, 9.06, 8.9 ,
        8.94, 8.91, 8.91, 8.95, 8.57, 8.82]),
 array([115., 119., 117., 118., 118., 116., 116., 119., 116., 115., 115.,
        117., 118., 113., 116., 118., 117., 121., 116., 117., 117., 117.,
        114., 118., 118., 119., 118., 118., 117., 118., 117., 119., 118.,
        118., 117., 117., 117., 116., 118., 119., 117., 119., 120., 117.,
        115., 115., 117., 116., 118., 117.]))
```

In [31]:

```
plt.scatter(x[y_means==0,0],x[y_means==0,1],color='red')
plt.scatter(x[y_means==1,0],x[y_means==1,1],color='green')
plt.scatter(x[y_means==2,0],x[y_means==2,1],color='blue')
plt.scatter(x[y_means==3,0],x[y_means==3,1],color='orange')
plt.scatter(x[y_means==4,0],x[y_means==4,1],color='yellow')
plt.xlabel('CPGA')
plt.ylabel('IQ')
```

Out[31]:

```
Text(0, 0.5, 'IQ')
```



In [39]:

```
X = student_data.drop(['cgpa'], axis = 1)
Y = student_data.cgpa
```

In [40]:

```
X.shape
```

Out[40]:

```
(200, 1)
```

In [41]:

```
Y.shape
```

Out[41]:

```
(200,)
```

In [43]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.33)
```

In [44]:

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

Out[44]:

```
LinearRegression()
```

In [45]:

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

In [46]:

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.2997403768574458
Testing Accuracy : 0.251699362242886
```