In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
#from sklearn import cluster
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
students = pd.read_csv('student_clustering.csv')
students
```

Out[2]:

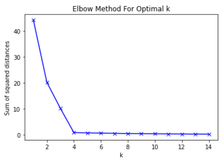
	cgpa	iq
0	5.13	88
1	5.90	113
2	8.36	93
3	8.27	97
4	5.45	110
...	...	...
195	4.68	89
196	8.57	118
197	5.85	112
198	6.23	108
199	8.82	117

200 rows × 2 columns

In [10]:
```
mms = MinMaxScaler()
mms.fit(students)
data_transformed = mms.transform(students)

Sum_of_squared_distances = []
K = range(1,15)
for k in K:
    km= KMeans(n_clusters=k, n_init = 10)
    km = km.fit(data_transformed)
    Sum_of_squared_distances.append(km.inertia_)
```

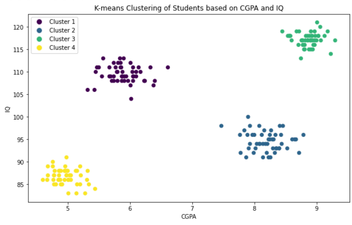
```
C:\Users\ADMIN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [11]:
```
plt.plot(K,Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum of squared distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In [22]:
```
kmeans = KMeans(4,n_init=10)
kmeans.fit(students)
labels = kmeans.labels_
print(len(labels))
print(labels)

plt.figure(figsize=(10, 6))
scatter = plt.scatter(students['cgpa'], students['iq'], c=labels, cmap='viridis')

legend_labels = [f'Cluster {i+1}' for i in range(kmeans.n_clusters)]
handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=scatter.cmap(scatter.norm(i)), markersize=10) for i in range(kmeans.n_clusters)]
plt.legend(handles, legend_labels)

plt.xlabel('CGPA')
plt.ylabel('IQ')
plt.title('K-means Clustering of Students based on CGPA and IQ')

plt.show()
```

```
200
[3 0 1 1 0 0 1 2 0 1 3 0 1 3 0 1 0 1 0 0 1 3 1 3 3 1 3 2 1 0 2 0 2 0 1 1 2
 0 3 0 3 1 1 3 2 2 1 0 2 0 3 3 2 1 2 0 0 2 0 2 0 1 1 2 3 2 1 3 0 1 0 2 1 3
 0 2 0 2 3 1 1 2 0 3 2 3 2 0 2 0 2 2 1 3 1 1 2 1 3 2 0 3 3 2 3 3 1 3 2 2 1
 2 0 0 1 2 1 0 2 3 3 0 1 2 1 3 1 0 3 1 1 0 3 3 0 2 0 3 1 1 1 3 0 3 3 2 3 2
 0 3 2 3 2 2 3 1 0 2 0 1 3 2 0 1 2 3 0 3 3 2 2 0 2 3 3 1 2 0 3 2 2 0 0 0 1
 3 1 1 2 0 1 1 3 3 1 3 2 0 0 2]
```



In [16]:
```
students['cluster'] = labels
students.sort_values(by='cluster')
```

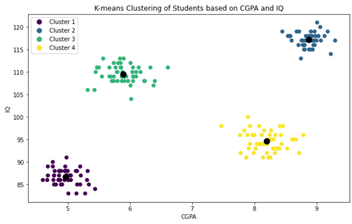
Out[16]:

	cgpa	iq	cluster
166	6.10	109	0
68	5.77	111	0
162	6.24	108	0
60	6.02	104	0
117	5.74	108	0
...	...	...	...
67	4.91	85	3
64	5.21	87	3
165	4.67	86	3
144	4.98	87	3
0	5.13	88	3

200 rows × 3 columns

In [26]:
```
# Now with centroids
kmeans = KMeans(4,n_init=10)
kmeans.fit(students)
labels = kmeans.labels_
print(len(labels))
print(labels)

plt.figure(figsize=(10, 6))
scatter = plt.scatter(students['cgpa'], students['iq'], c=labels, cmap='viridis')

plt.scatter(centroids[:, 0], centroids[:, 1], marker='o', c='black', s=100)

legend_labels = [f'Cluster {i+1}' for i in range(kmeans.n_clusters)]
handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=scatter.cmap(scatter.norm(i)), markersize=10) for i in range(kmeans.n_clusters)]
plt.legend(handles, legend_labels)

plt.xlabel('CGPA')
plt.ylabel('IQ')
plt.title('K-means Clustering of Students based on CGPA and IQ')

plt.show()
```

```
200
[0 2 3 3 2 2 3 1 2 3 0 2 3 0 2 3 3 2 2 3 0 3 0 0 3 0 1 3 2 1 2 1 2 3 3 1
 2 0 2 0 3 3 0 1 1 3 2 1 2 0 0 1 3 1 2 2 1 2 1 2 3 3 1 0 1 3 0 2 3 2 1 3 0
 2 1 2 1 0 3 3 1 2 0 1 0 1 2 1 2 1 1 3 0 3 3 1 3 0 1 2 0 0 1 0 0 3 0 1 1 3
 1 2 2 3 1 3 2 1 0 0 2 3 1 3 0 3 2 0 0 3 3 2 0 0 2 1 2 0 3 3 3 0 2 0 0 1 0 1
 2 0 1 0 1 1 0 3 2 1 2 3 0 1 2 3 1 0 2 0 0 1 1 2 1 0 0 3 1 2 0 1 1 2 2 2 3
 0 3 3 1 2 3 3 0 0 3 0 1 2 2 1]
```



Learning Reflection

Implementing K-Means Clustering

I learned that implementing the K-Means clustering algorithm provides valuable insights into the machine learning process, particularly in unsupervised learning. From data preparation to model evaluation, each step contributed to a deeper understanding of the algorithm's mechanics and its application in real-world scenarios.

Data Preparation

The journey began with loading and scaling the dataset, ensuring that it was suitable for clustering. This step emphasized the importance of data normalization and its direct impact on model performance. Using MinMaxScaler to transform the data laid a solid foundation for subsequent analysis by ensuring that all features contributed equally to the distance calculations used in K-Means clustering.

Feature Engineering

While the code primarily focused on scaling the features, it underscored the significance of transforming raw data into a suitable format for clustering. Proper feature engineering can enhance the clustering performance by capturing relevant patterns and relationships within the data. In this implementation, basic feature scaling laid the groundwork for more advanced transformations and preprocessing steps in future projects.

Modeling

Creating K-Means clusters involved determining the optimal number of clusters using the Elbow method. Plotting the sum of squared distances for different values of kk highlighted the point where adding more clusters yielded diminishing returns, known as the elbow point. Understanding the parameters and their effects on the model allowed for fine-tuning and optimization. K-Means' simplicity and effectiveness made it an ideal choice for initial clustering tasks, providing clear groupings of the data.

Model Evaluation

Evaluating model performance involved visualizing the clusters and their centroids. This step provided a comprehensive assessment of the model's strengths and weaknesses. Visualizing the clusters on a scatter plot of CGPA versus IQ offered intuitive insights into the groupings. Beyond visual inspection, considering additional metrics like inertia helped quantify the clustering performance.

Handling Cluster Interpretation

Interpreting the resulting clusters and assigning meaningful labels such as 'Diligent Students', 'Underachievers', 'Exceptional Students', and 'Struggling Students' added context to the raw clusters. This step emphasized the importance of understanding the data and domain knowledge in making sense of the clusters.

Visualization

Visualizing the clusters and their centroids played a crucial role in understanding the model's decision-making process. Plotting the data points along with the cluster centroids enhanced interpretability and trust in the model's groupings. Visualization facilitated communication of results to stakeholders, making the insights derived from clustering more accessible and actionable.

Summary

In conclusion, implementing the K-Means clustering algorithm was a rewarding learning experience that highlighted the importance of data quality, feature engineering, model selection, evaluation metrics, handling cluster interpretation, and visualization. Each step contributed to a deeper understanding of the machine learning workflow and its practical applications. Moving forward, these insights will inform future projects, guiding the development of robust and reliable clustering solutions.