In [1]:
```
import pandas as pd
import numpy as np
from sklearn. datasets import load_iris
```

In [2]:
```
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target
```

In [3]:
```
print(type(iris))
```

```
<class 'sklearn.utils._bunch.Bunch'>
```

In [4]:
```
iris.keys()
```

Out[4]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'fil
ename', 'data_module'])
```

In [5]:
```
iris['target_names']
```

Out[5]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [6]:
```
df['target']=iris.target
```

In [7]:
```
df.head()
```

Out[7]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	target
0	5.1	3.5	1.4	0.2	0	0
1	4.9	3.0	1.4	0.2	0	0
2	4.7	3.2	1.3	0.2	0	0
3	4.6	3.1	1.5	0.2	0	0
4	5.0	3.6	1.4	0.2	0	0

In [8]:
```
dict_target_name={0:'setosa',1:'versicolor',2:'virginica'}
```

In [9]:
```
df['target_names']=df['target'].map(dict_target_name)
```

In [10]:
```
df.head()
```

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	target	target_names
0	5.1	3.5	1.4	0.2	0	0	setosa
1	4.9	3.0	1.4	0.2	0	0	setosa
2	4.7	3.2	1.3	0.2	0	0	setosa
3	4.6	3.1	1.5	0.2	0	0	setosa
4	5.0	3.6	1.4	0.2	0	0	setosa

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int32  
 5   target             150 non-null    int32  
 6   target_names       150 non-null    object 
dtypes: float64(4), int32(2), object(1)
memory usage: 7.2+ KB
```

In [12]:
```
df.describe()
```

Out[12]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	target
count	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000	2.000000

Data visualization

In [13]:
```
df.columns
```

Out[13]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species', 'target', 'target_names'],
      dtype='object')
```

In [14]:
```
df.shape
```

Out[14]:
```
(150, 7)
```

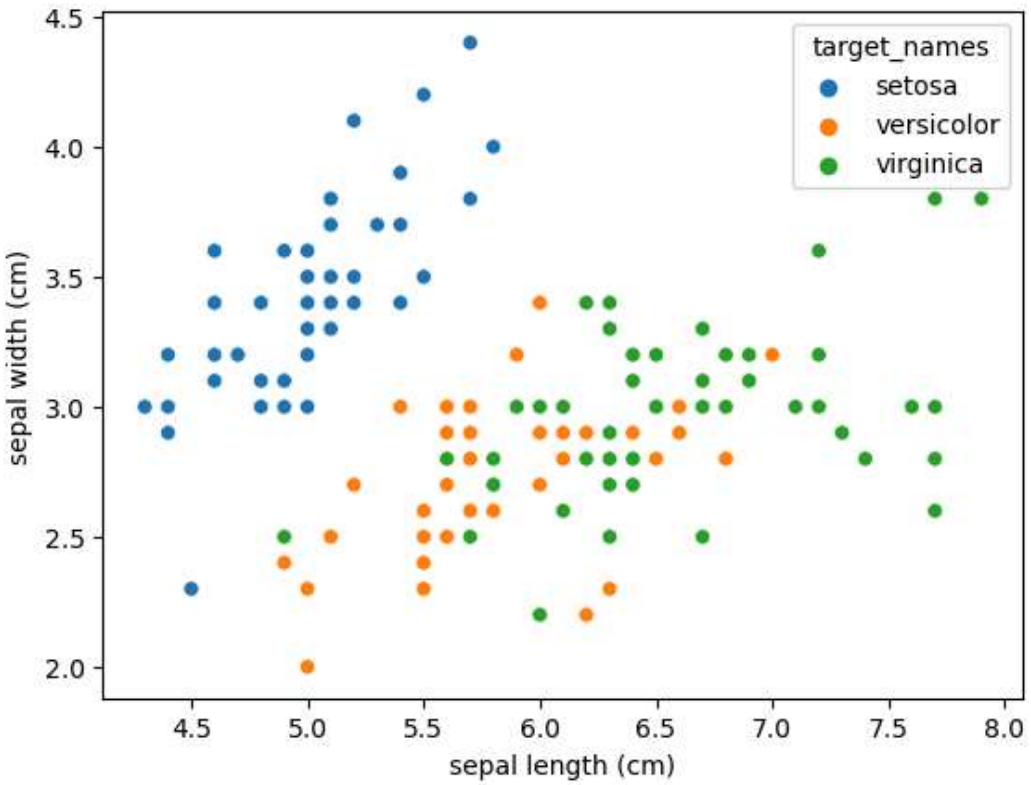
In [15]:
```
df.isnull().sum()
```

Out[15]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
target               0
target_names         0
dtype: int64
```

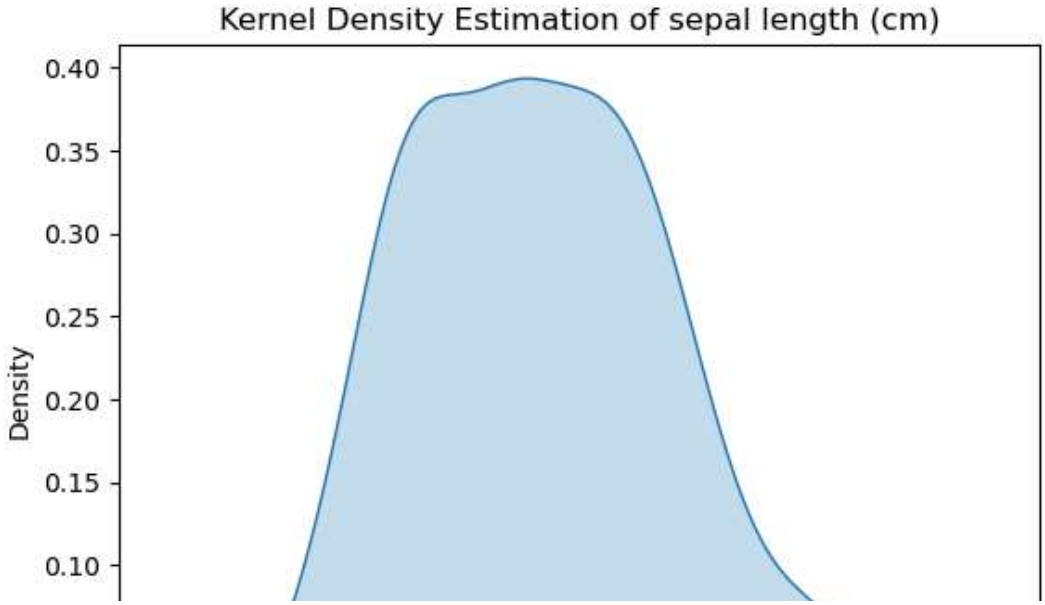
In [16]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns
```

In [17]:
```
sns.scatterplot(data=df,x='sepal length (cm)',y='sepal width (cm)',hue='target_names'
```

Out[17]:
```
<Axes: xlabel='sepal length (cm)', ylabel='sepal width (cm)'>
```

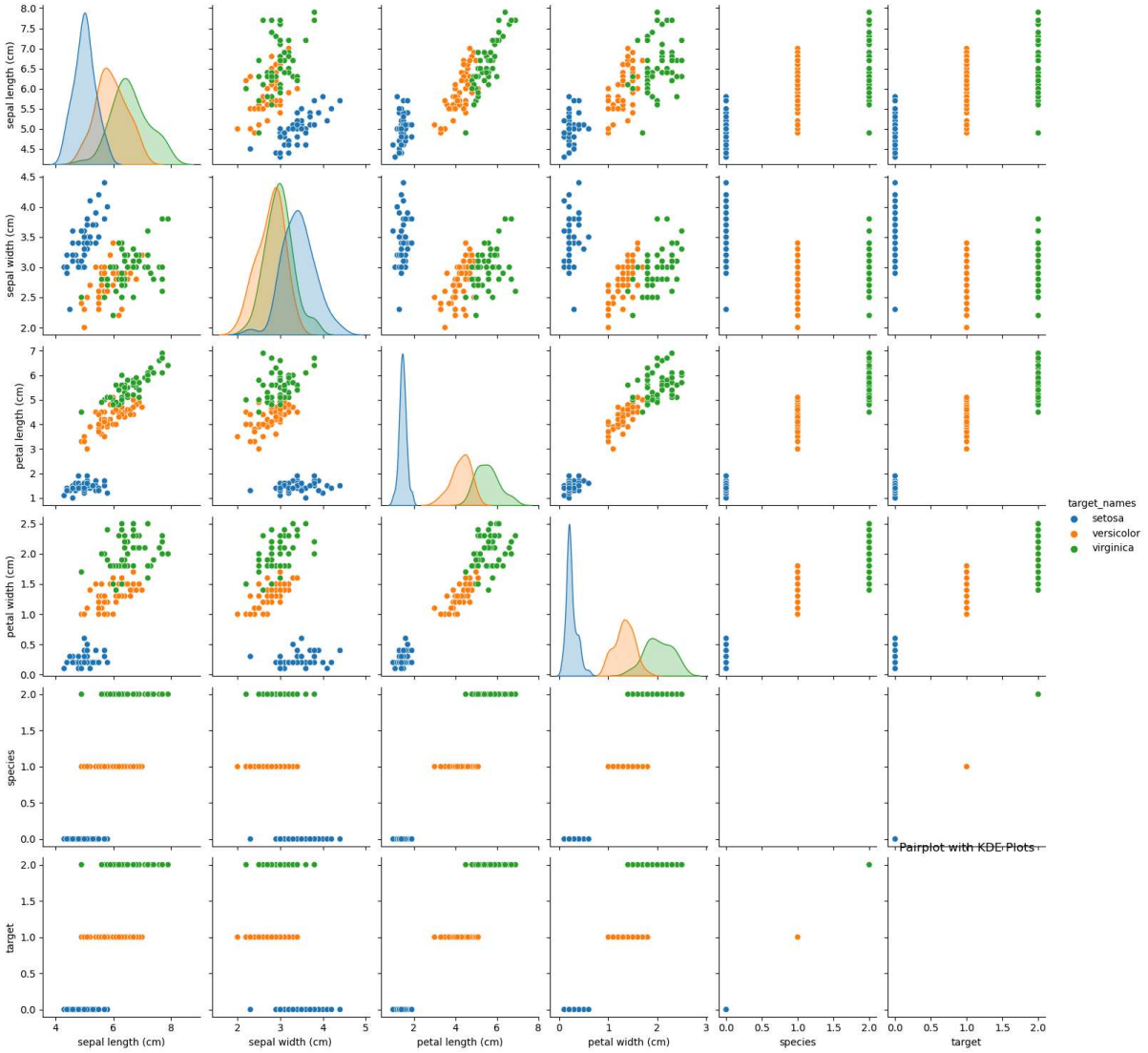


In [18]:
```
for feature in df.columns[:-2]:
    sns.kdeplot(data=df, x=feature, fill=True)
    plt.xlabel(feature)
    plt.ylabel('Density')
    plt.title(f'Kernel Density Estimation of {feature}')
    plt.show()
```



In [19]:
```
sns.pairplot(data=df, hue='target_names', diag_kind='kde')
plt.title('Pairplot with KDE Plots')
plt.show()
```

```
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\seaborn\axisgrid.py:1
18: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Features

In [20]:
```
features = df.drop(columns=["target",'target_names'],axis=1)
features
```

Out[20]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

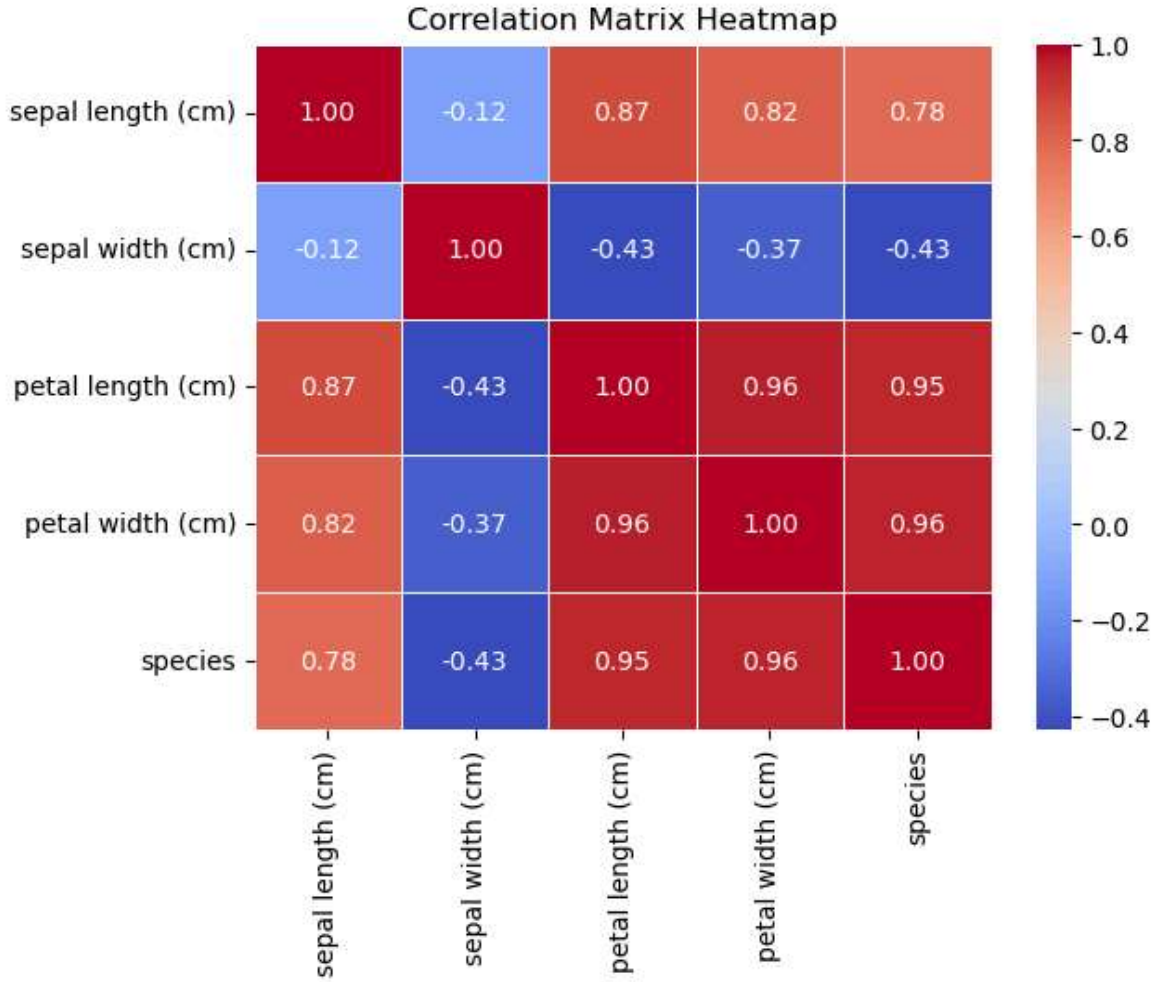
Scaling

In [21]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [22]:
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

In [23]:

```
correlation_matrix = features.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f
plt.title("Correlation Matrix Heatmap")
plt.show()
```



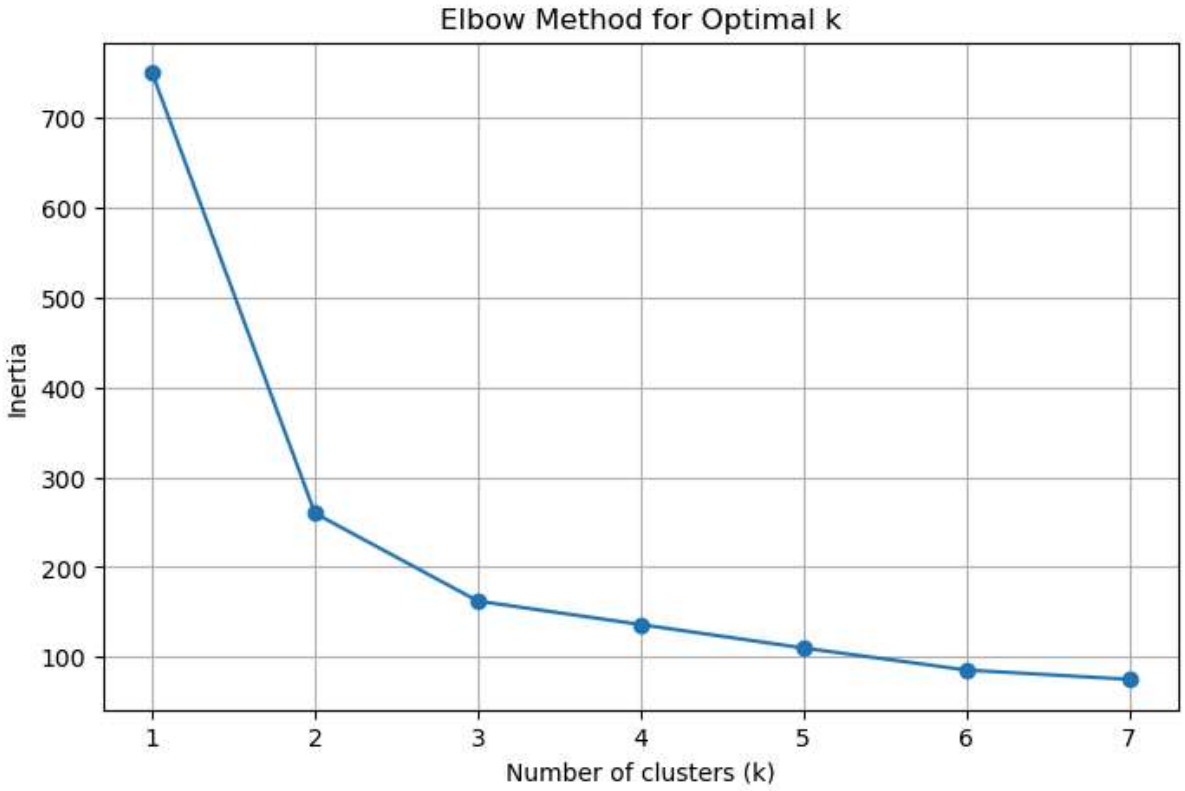
Elbow method

Using the elbow method to determine optimal number of clusters for k- means clustering

In [24]:
```
inertia=[]
k_value=range(1,8)
for k in k_value:
  kmeans=KMeans(n_clusters=k)
  kmeans.fit(scaled_features)
  inertia.append(kmeans.inertia_)
```

```
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'aut
o' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmea
ns.py:1436: 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 e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [25]:
```
plt.figure(figsize=(8, 5))
plt.plot(k_value, inertia, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.grid(True)
plt.show()
```



KMeans Clustering

K-means is an algorithm that groups data points into clusters based on their similarities.

How It Works

Initialization: Randomly choose (K) centroids (cluster centers).

Assignment: Assign each data point to the nearest centroid, forming clusters.

Update: Recalculate the centroids as the average of all points in each cluster.

Repeat: Continue assigning and updating until the centroids don't change much.

Why It's Suitable for the Iris Dataset

Natural Grouping: The Iris dataset has natural clusters (species of flowers), which K-means can identify.

Feature Space: Uses the numeric features (sepal length, sepal width, petal length, petal width) to form clusters.

Low Dimensionality: With only four features, it's computationally efficient.

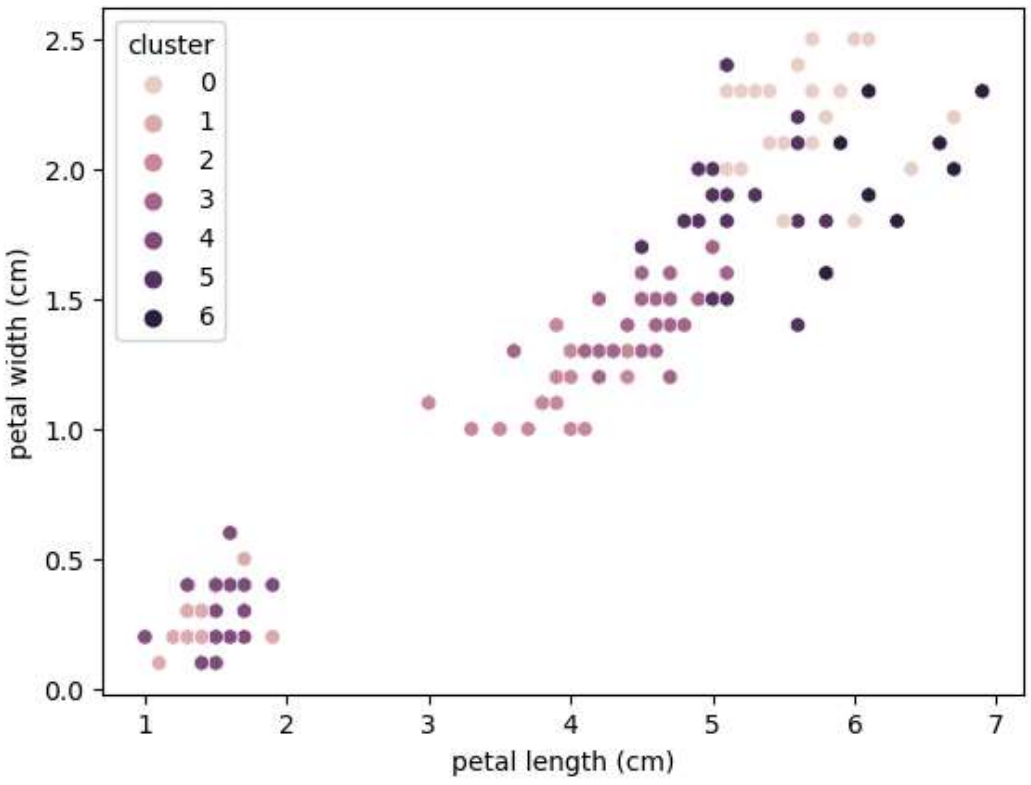
Known Classes: The dataset has three species, so setting (K = 3) aligns with the actual number of clusters.

In [26]:
```
means=KMeans(n_clusters=3)
df['cluster'] = kmeans.fit_predict(scaled_features)


sns.scatterplot(data=df,x='petal length (cm)',y='petal width (cm)',hue='cluster')
```

```
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: 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(
```

Out[26]: <Axes: xlabel='petal length (cm)', ylabel='petal width (cm)'>



In [27]:
```
kmeans=KMeans(n_clusters=3)
df['cluster'] = kmeans.fit_predict(scaled_features)
df[['target','cluster']]
```

```
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\FamiAmal\.anaconda\annaconda for me\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: 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(
```

Out[27]:

	target	cluster
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
...	...	...
145	2	0
146	2	0
147	2	0
148	2	0
149	2	0

150 rows × 2 columns

Hierarchical Cluster

Hierarchical clustering is an unsupervised machine learning algorithm that builds a hierarchy of clusters using two main approaches:

Agglomerative (Bottom-up): Starts with each data point as its own cluster and merges them step by step based on a distance metric until a single cluster or desired number of clusters is formed.

Divisive (Top-down): Starts with all data points in one cluster and splits them recursively until each data point is its own cluster or a desired number of clusters is reached.A dendrogram is often used to visualize the hierarchy, showing how clusters merge or split at different distance levels.

Suitability for the Iris Dataset

Flexible Cluster Formation: No need to predefine the number of clusters, which is useful for the Iris dataset's potential nested structure.

Visual Insight via Dendrogram: Helps visualize relationships and subclusters within species.

Small Dataset: The Iris dataset's 150 samples make hierarchical clustering computationally feasible.

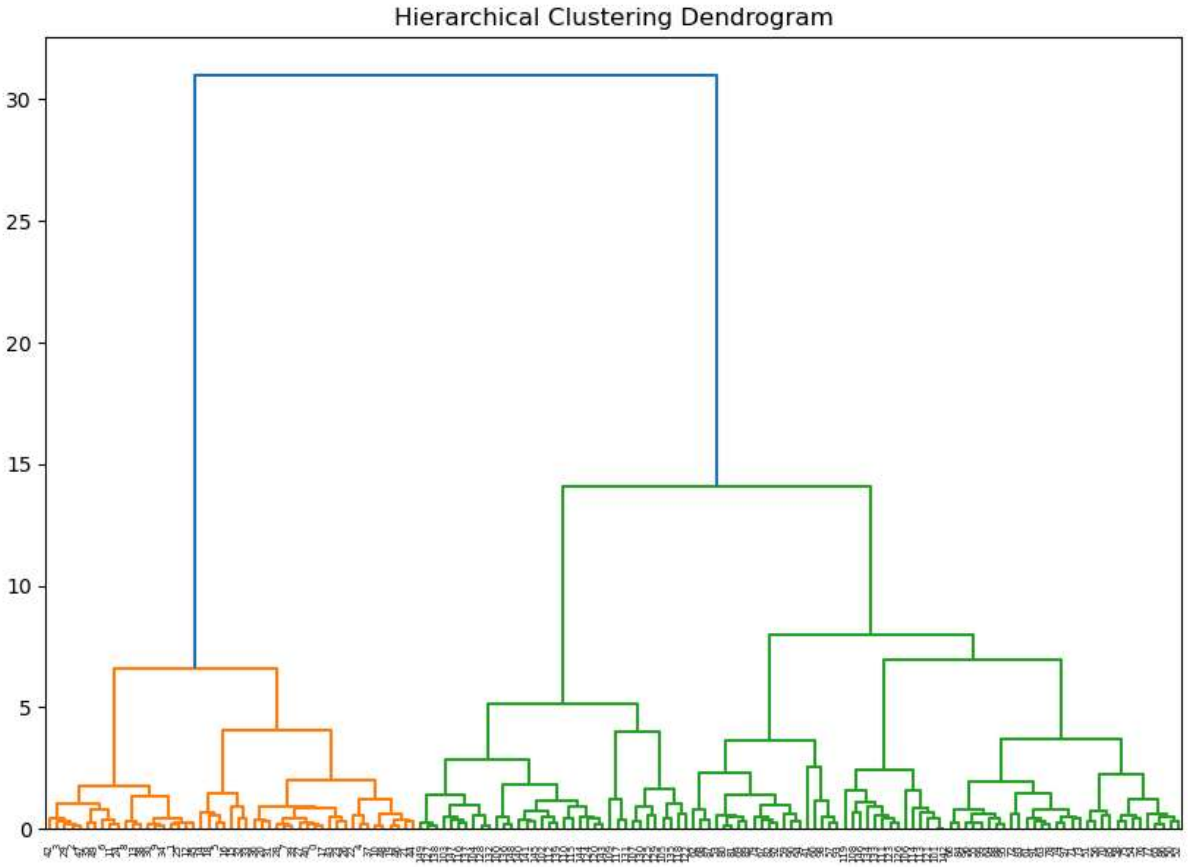
Natural Clustering: Can detect natural groupings corresponding to the species without prior knowledge of the number of species.

In [28]:
```
aggclust=AgglomerativeClustering(n_clusters=3,linkage='ward')
df['cluster']=aggclust.fit_predict(scaled_features)
```

In [29]:
```
from scipy.cluster.hierarchy import dendrogram, linkage


Z = linkage(scaled_features, 'ward')


plt.figure(figsize=(10, 7))
dendrogram(Z, leaf_rotation=90)
plt.title("Hierarchical Clustering Dendrogram")
plt.show()
```



In [ ]:

In [ ]: