In [7]:
```
#Step 1: Load Data; Select all four features (sepal length, sepal width, petal le
# of the dataset in a variable called x so that we can train our model with these
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv('OneDrive\Desktop\iris.csv')
#df.head()

x = df.drop(["species"], axis=1)
x.head()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

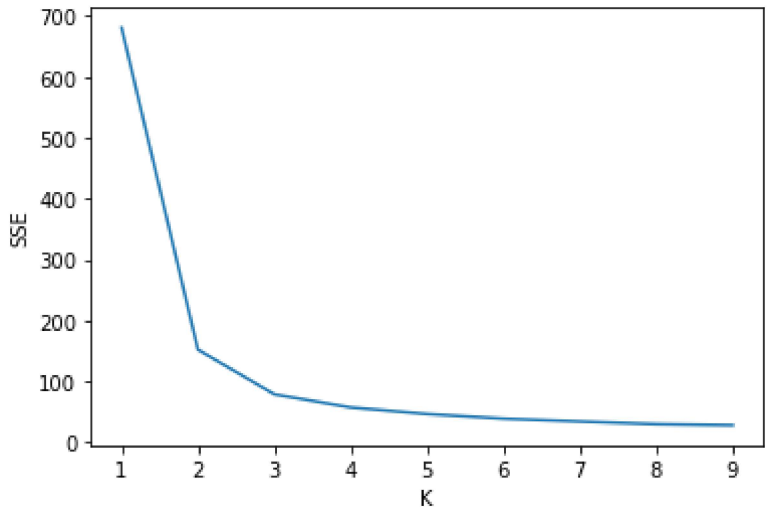
In [11]:
```
#Step 2: Using Elbow method to determine how many clusters you will choose.(30 pc
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k)
    km.fit(x)
    sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('SSE')
plt.plot(k_range, sse)
```

```
C:\Users\andre\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: Use
rWarning: 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 environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

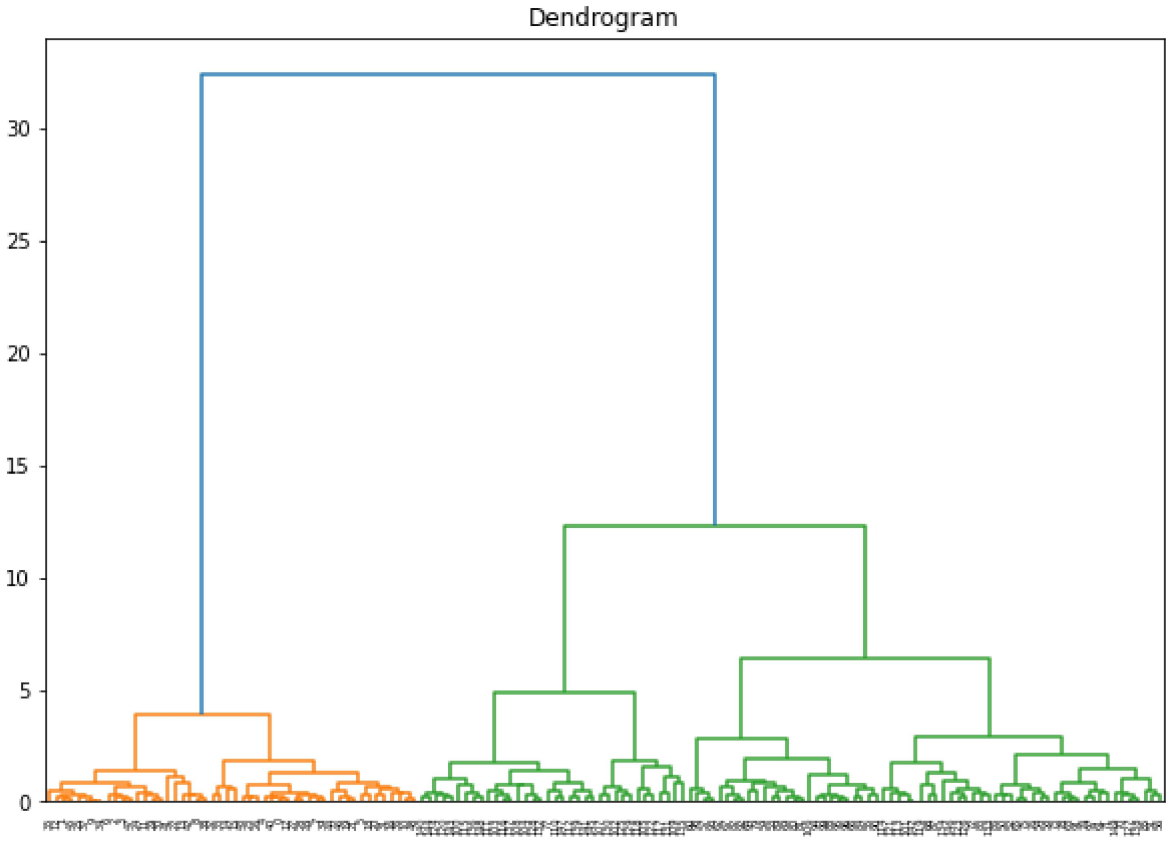
Out[11]:
```
[<matplotlib.lines.Line2D at 0x2dc1854ee50>]
```



In [13]:
```
#Step 3: Build your model and import your k-means cluster center .(40 points)
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Dendrogram")

#comparing elbow method to dendrogram
clusters = shc.linkage(x,
            method='ward',
            metric="euclidean")
shc.dendrogram(Z=clusters)
plt.show()
```



In [60]:
```
kmeans = KMeans(n_clusters=3)
y_pred = kmeans.fit_predict(x)
center = kmeans.cluster_centers_
center
```

Out[60]:
```
array([[ 5.00600000e+00,  3.41800000e+00,  1.46400000e+00,
         2.44000000e-01, -1.11022302e-15],
       [ 6.85000000e+00,  3.07368421e+00,  5.74210526e+00,
         2.07105263e+00,  2.00000000e+00],
       [ 5.90161290e+00,  2.74838710e+00,  4.39354839e+00,
         1.43387097e+00,  1.00000000e+00]])
```

In [61]:
```
kmeans.fit(x)
classification = model.labels_
classification
```

Out[61]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

In [ ]:
```

```