In [17]:

```
# Importing libraries to be used.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn import datasets

#  Loading the dataset
df = pd.read_csv("socialMedia_Influencers_TiktokSep2022.csv")

print(df.head())
print(df.info())
```

```
   S.no   Tiktoker name   Tiktok name Subscribers Views avg. Likes avg.  \
0     1   jypestraykids    Stray Kids       13.8M       6.4M       2.3M
1     2      khaby.lame  Khabane lame      149.2M      17.3M       2.3M
2     3  scarlettsspam2      scarlett        2.1M      17.9M     845.8K
3     4       addisonre   Addison Rae       88.7M        22M     906.6K
4     5      belindatok       Belinda        4.8M      14.2M       1.5M

  Comments avg. Shares avg.
0         50.2K       34.2K
1         15.2K        8.7K
2         53.9K        6.3K
3          7.6K       26.2K
4         14.5K       15.3K
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   S.no            1000 non-null   int64
 1   Tiktoker name   1000 non-null   object
 2   Tiktok name     999 non-null    object
 3   Subscribers     1000 non-null   object
 4   Views avg.      1000 non-null   object
 5   Likes avg.      1000 non-null   object
 6   Comments avg.   1000 non-null   object
 7   Shares avg.     1000 non-null   object
dtypes: int64(1), object(7)
memory usage: 62.6+ KB
None
```

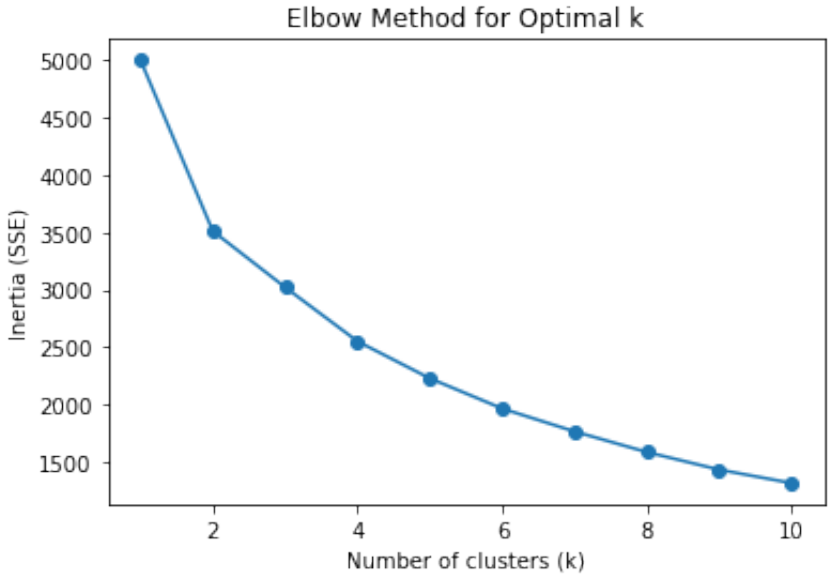
In [18]:
```
# Helper function to convert values like '13.8M', '50.2K' into numbers
def convert_to_number(x):
    if isinstance(x, str):
        x = x.replace(",", "").strip()
        if x.endswith("K"):
            return float(x[:-1]) * 1_000
        elif x.endswith("M"):
            return float(x[:-1]) * 1_000_000
        elif x.endswith("B"):
            return float(x[:-1]) * 1_000_000_000
        else:
            return float(x)
    return x

# Apply conversion to selected columns
num_cols = ["Subscribers", "Views avg.", "Likes avg.", "Comments avg.", "Sl
for col in num_cols:
    df[col] = df[col].apply(convert_to_number)

print(df[num_cols].head())
print(df.dtypes)
```

```
   Subscribers  Views avg.  Likes avg.  Comments avg.  Shares avg.
0   13800000.0   6400000.0   2300000.0        50200.0      34200.0
1  149200000.0  17300000.0   2300000.0        15200.0       8700.0
2    2100000.0  17900000.0    845800.0        53900.0       6300.0
3   88700000.0  22000000.0    906600.0         7600.0      26200.0
4    4800000.0  14200000.0   1500000.0        14500.0      15300.0
S.no               int64
Tiktoker name     object
Tiktok name       object
Subscribers      float64
Views avg.       float64
Likes avg.       float64
Comments avg.    float64
Shares avg.      float64
dtype: object
```

In [19]:
```
# Selecting numerical features for clustering
features = num_cols
X = df[features]
```

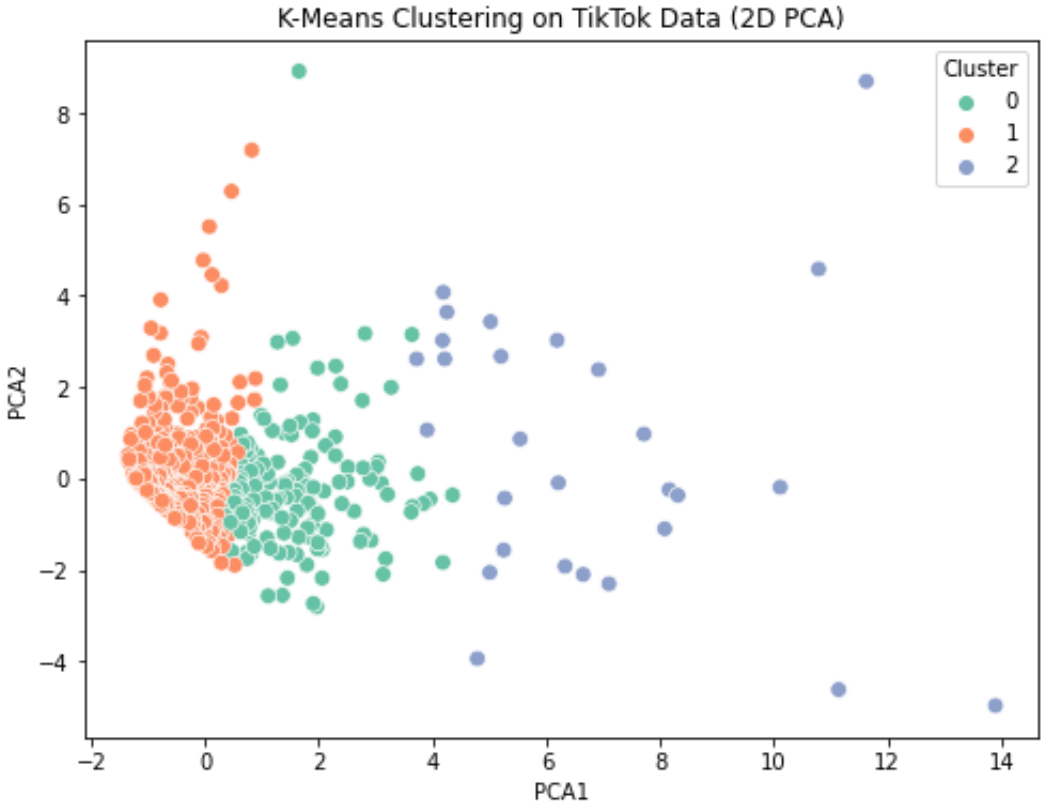
In [20]:
```
# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [21]:
```
# using Elbow method
inertia = []
K = range(1, 11)
for k in K:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    km.fit(X_scaled)
    inertia.append(km.inertia_)

plt.plot(K, inertia, marker='o')
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia (SSE)")
plt.title("Elbow Method for Optimal k")
plt.show()
```



In [22]:
```
# Training KMeans with chosen k
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

In [23]:
```
# PCA visualization by clustering K-means
pca = PCA(n_components=2)
pca_result = pca.fit_transform(X_scaled)
df['PCA1'] = pca_result[:,0]
df['PCA2'] = pca_result[:,1]

plt.figure(figsize=(8,6))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=df, palette='Set2'
plt.title("K-Means Clustering on TikTok Data (2D PCA)")
plt.show()
```



In [24]:

```
# Inspecting the  cluster characteristics
print(df.groupby('Cluster')[features].mean())
```

```
           Subscribers    Views avg.    Likes avg.  Comments avg.   Shares a
vg.
Cluster
0         1.363303e+07  5.167665e+06  6.404108e+05    3572.365269   3827.215
569
1         4.774087e+06  2.105505e+06  2.562955e+05    1851.980124   2911.600
000
2         3.441987e+07  1.059286e+07  1.398857e+06   16167.857143  10017.857
143
```

In [25]:

```
#In this project, we worked with a TikTok dataset containing information l
Since the data was not directly usable (with values such as 13.8M or 50.2K
After that, we standardized the features so that large values, like subscr
Using the Elbow Method, we determined the best number of clusters and then
To better visualize the results, we used PCA to reduce the data into two d
Finally, by analyzing the average values in each cluster, we were able to
such as mega influencers with massive reach, mid-level creators with steady
Through this process, we learned not only how to prepare and clean real-wo
```

```
  File "/var/folders/v_/7dgqf6xx1y30mhhjz64drcqm0000gn/T/ipykernel_48467/20
37120222.py", line 2
    Since the data was not directly usable (with values such as 13.8M or 5
0.2K), we first cleaned and converted everything into proper numbers.
          ^
SyntaxError: invalid syntax
```

In [26]:
```
#activity C and D
```

In [27]:
```
#preparing the data#  Loading the dataset
airquality = pd.read_csv('air_quality_no2_long.csv')
```

In [42]:
```
x = airquality.value
y = airquality.parameter
```

In [44]:
```
print(airquality.value.shape)
print(airquality.parameter.shape)
```

```
(2068,)
(2068,)
```

In [45]:
```
print(airquality.value)
```

```
0       20.0
1       21.8
2       26.5
3       24.9
4       21.4
        ...
2063    26.0
2064    16.0
2065    19.0
2066    19.0
2067    23.0
Name: value, Length: 2068, dtype: float64
```

In [46]:
```
np.bincount(airquality.value)
```

Out[46]:
```
array([19,  8,  6,  9,  9, 17, 12, 13, 23, 31, 46, 54, 36, 47, 45, 60, 56,
       65, 51, 75, 72, 61, 71, 74, 65, 70, 87, 61, 59, 62, 54, 59, 56, 64,
       37, 44, 43, 27, 29, 21, 23, 17, 17, 19, 18, 14, 16, 11,  6, 14,  8,
       10, 10,  8,  6,  4,  3,  3,  5,  6,  7,  2,  4,  2,  1,  1,  3,  5,
        4,  1,  2,  1,  1,  1,  3,  3,  1,  2,  1,  2,  0,  1,  0,  0,  1,
        0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  2])
```

In [51]:
```
#print target names
print(airquality['country'].head())
```

```
0    FR
1    FR
2    FR
3    FR
4    FR
Name: country, dtype: object
```

In [52]:
```
print(airquality[['country', 'value']].head())
```

```
  country  value
0      FR   20.0
1      FR   21.8
2      FR   26.5
3      FR   24.9
4      FR   21.4
```

In [55]:
```
print(X.shape)  # (n_samples, n_features)
print(y.shape)  # (n_samples,)
```

```
(1000, 5)
(2068,)
```

In [56]:
```
# Suppose you want to predict 'country' from 'value'
X = airquality[['value']]          # features
y = airquality['country']          # target
```

In [57]:
```
data = airquality.dropna(subset=['value', 'country'])
X = data[['value']]
y = data['country']
```

In [59]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=1234)
model = clf.fit(X, y)
```

In [61]:
```
from sklearn import tree

text_representation = tree.export_text(clf)
print(text_representation)
```

```
|--- feature_0 <= 36.05
|   |--- feature_0 <= 19.95
|   |   |--- feature_0 <= 8.05
|   |   |   |--- feature_0 <= 0.50
|   |   |   |   |--- class: GB
|   |   |   |--- feature_0 >  0.50
|   |   |   |   |--- feature_0 <= 6.05
|   |   |   |   |   |--- feature_0 <= 4.40
|   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |--- feature_0 >  4.40
|   |   |   |   |   |   |--- feature_0 <= 4.90
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |--- feature_0 >  4.90
|   |   |   |   |   |   |   |--- feature_0 <= 5.45
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |--- feature_0 >  5.45
|   |   |   |   |   |   |   |   |--- feature_0 <= 5.95
```

```
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |--- feature_0 >  5.95
|   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |--- feature_0 >  6.05
|   |   |   |   |   |--- feature_0 <= 6.75
|   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  6.75
|   |   |   |   |   |   |--- feature_0 <= 7.95
|   |   |   |   |   |   |   |--- feature_0 <= 7.05
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |--- feature_0 >  7.05
|   |   |   |   |   |   |   |   |--- feature_0 <= 7.35
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |--- feature_0 >  7.35
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 7.70
|   |   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |   |--- feature_0 >  7.70
|   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  7.95
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |--- feature_0 >  8.05
|   |   |   |   |--- feature_0 <= 19.05
|   |   |   |   |   |--- feature_0 <= 18.95
|   |   |   |   |   |   |--- feature_0 <= 18.05
|   |   |   |   |   |   |   |--- feature_0 <= 17.95
|   |   |   |   |   |   |   |   |--- feature_0 <= 17.05
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 16.95
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 16.05
|   |   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 15.95
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
15
|   |   |   |   |   |   |   |   |   |   |   |--- feature_0 >  15.95
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  16.05
|   |   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 16.55
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
2
|   |   |   |   |   |   |   |   |   |   |   |--- feature_0 >  16.55
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  16.95
|   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |--- feature_0 >  17.05
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 17.55
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 17.45
|   |   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  17.45
|   |   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |--- feature_0 >  17.55
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  17.95
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |--- feature_0 >  18.05
|   |   |   |   |   |   |--- feature_0 <= 18.55
|   |   |   |   |   |   |   |--- feature_0 <= 18.45
|   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |--- feature_0 >  18.45
|   |   |   |   |   |   |   |--- class: FR
```

```
|   |   |   |   |   |   |--- feature_0 >  18.55
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |--- feature_0 >  18.95
|   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |--- feature_0 >  19.05
|   |   |   |   |   |--- class: FR
|   |--- feature_0 >  19.95
|   |   |--- feature_0 <= 20.05
|   |   |   |--- class: GB
|   |   |--- feature_0 >  20.05
|   |   |   |--- feature_0 <= 20.95
|   |   |   |   |--- feature_0 <= 20.45
|   |   |   |   |   |--- class: FR
|   |   |   |   |--- feature_0 >  20.45
|   |   |   |   |   |--- feature_0 <= 20.55
|   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  20.55
|   |   |   |   |   |   |--- class: FR
|   |   |   |--- feature_0 >  20.95
|   |   |   |   |--- feature_0 <= 21.05
|   |   |   |   |   |--- class: GB
|   |   |   |   |--- feature_0 >  21.05
|   |   |   |   |   |--- feature_0 <= 21.95
|   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  21.95
|   |   |   |   |   |   |--- feature_0 <= 22.05
|   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  22.05
|   |   |   |   |   |   |   |--- feature_0 <= 22.95
|   |   |   |   |   |   |   |   |--- feature_0 <= 22.55
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 22.40
|   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  22.40
|   |   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |--- feature_0 >  22.55
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  22.95
|   |   |   |   |   |   |   |   |--- feature_0 <= 23.05
|   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |--- feature_0 >  23.05
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 23.95
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 23.45
|   |   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  23.45
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
2
|   |   |   |   |   |   |   |   |   |--- feature_0 >  23.95
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 24.05
|   |   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |   |--- feature_0 >  24.05
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
26
|--- feature_0 >  36.05
|   |--- feature_0 <= 49.05
|   |   |--- feature_0 <= 36.95
|   |   |   |--- feature_0 <= 36.55
|   |   |   |   |--- feature_0 <= 36.45
```

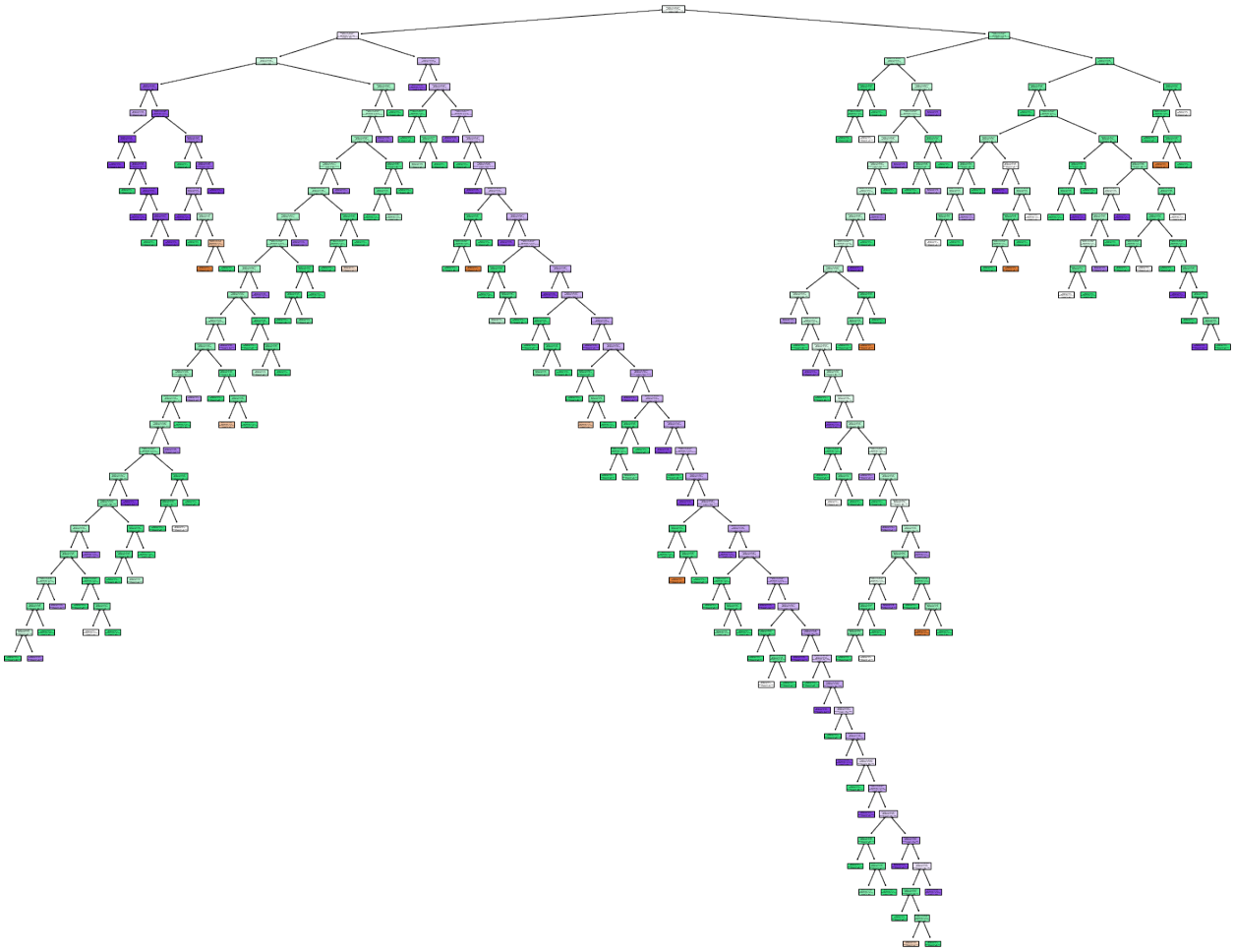
```
|   |   |   |   |   |--- class: FR
|   |   |   |   |--- feature_0 >  36.45
|   |   |   |   |   |--- class: BE
|   |   |   |--- feature_0 >  36.55
|   |   |   |   |--- class: FR
|   |   |--- feature_0 >  36.95
|   |   |   |--- feature_0 <= 48.95
|   |   |   |   |--- feature_0 <= 46.05
|   |   |   |   |   |--- feature_0 <= 45.95
|   |   |   |   |   |   |--- feature_0 <= 45.05
|   |   |   |   |   |   |   |--- feature_0 <= 44.95
|   |   |   |   |   |   |   |   |--- feature_0 <= 44.05
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 43.95
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 43.05
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
15
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  43.05
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
3
|   |   |   |   |   |   |   |   |   |--- feature_0 >  43.95
|   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |--- feature_0 >  44.05
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  44.95
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  45.05
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  45.95
|   |   |   |   |   |   |--- class: GB
|   |   |   |   |--- feature_0 >  46.05
|   |   |   |   |   |--- feature_0 <= 47.10
|   |   |   |   |   |   |--- feature_0 <= 46.95
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |--- feature_0 >  46.95
|   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |--- feature_0 >  47.10
|   |   |   |   |   |   |--- class: FR
|   |   |   |--- feature_0 >  48.95
|   |   |   |   |--- class: GB
|   |--- feature_0 >  49.05
|   |   |--- feature_0 <= 67.15
|   |   |   |--- feature_0 <= 50.45
|   |   |   |   |--- class: FR
|   |   |   |--- feature_0 >  50.45
|   |   |   |   |--- feature_0 <= 53.05
|   |   |   |   |   |--- feature_0 <= 51.80
|   |   |   |   |   |   |--- feature_0 <= 51.10
|   |   |   |   |   |   |   |--- feature_0 <= 50.90
|   |   |   |   |   |   |   |   |--- feature_0 <= 50.60
|   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |--- feature_0 >  50.60
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  50.90
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  51.10
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  51.80
```

```
|   |   |   |   |   |   |--- feature_0 <= 52.15
|   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  52.15
|   |   |   |   |   |   |   |--- feature_0 <= 52.85
|   |   |   |   |   |   |   |   |--- feature_0 <= 52.55
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 52.45
|   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  52.45
|   |   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |--- feature_0 >  52.55
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  52.85
|   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |--- feature_0 >  53.05
|   |   |   |   |   |--- feature_0 <= 57.90
|   |   |   |   |   |   |--- feature_0 <= 54.05
|   |   |   |   |   |   |   |--- feature_0 <= 53.75
|   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  53.75
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  54.05
|   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |--- feature_0 >  57.90
|   |   |   |   |   |   |--- feature_0 <= 60.05
|   |   |   |   |   |   |   |--- feature_0 <= 59.70
|   |   |   |   |   |   |   |   |--- feature_0 <= 59.15
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 58.90
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 58.35
|   |   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  58.35
|   |   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  58.90
|   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |--- feature_0 >  59.15
|   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |--- feature_0 >  59.70
|   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |--- feature_0 >  60.05
|   |   |   |   |   |   |   |--- feature_0 <= 66.80
|   |   |   |   |   |   |   |   |--- feature_0 <= 60.60
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 60.35
|   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  60.35
|   |   |   |   |   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |   |   |   |--- feature_0 >  60.60
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 61.95
|   |   |   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |   |   |   |   |   |   |--- feature_0 >  61.95
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 62.05
|   |   |   |   |   |   |   |   |   |   |   |--- class: GB
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  62.05
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth
3
|   |   |   |   |   |   |   |--- feature_0 >  66.80
|   |   |   |   |   |   |   |   |--- class: FR
|   |   |   |--- feature_0 >  67.15
|   |   |   |   |--- feature_0 <= 94.40
```

```
|   |   |   |   |--- feature_0 <= 74.35
|   |   |   |   |   |--- class: FR
|   |   |   |   |--- feature_0 >  74.35
|   |   |   |   |   |--- feature_0 <= 74.65
|   |   |   |   |   |   |--- class: BE
|   |   |   |   |   |--- feature_0 >  74.65
|   |   |   |   |   |   |--- class: FR
|   |   |   |--- feature_0 >  94.40
|   |   |   |   |--- class: FR
```

In [63]:

```
import matplotlib.pyplot as plt
from sklearn import tree

plt.figure(figsize=(25, 20))
tree.plot_tree(
    clf,
    feature_names=X.columns,           # ['value']
    class_names=y.unique(),            # unique country names
    filled=True
)
plt.show()
```



In [ ]:

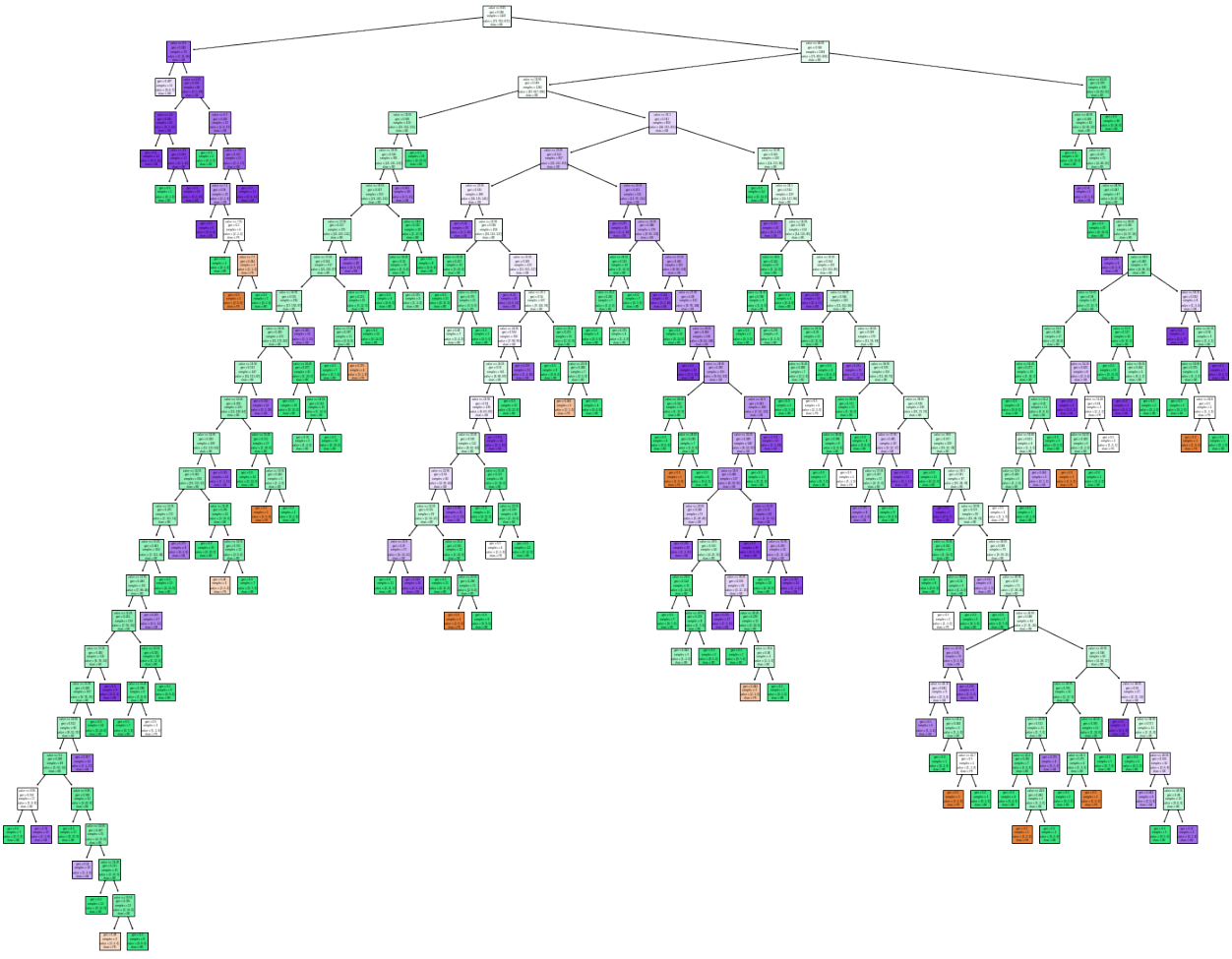
```
#predicting the data using classification model
```

In [71]:
```
#splitting the data
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.3
print("Training split input- ", X_train.shape)
print("Testing split input- ",X_test.shape)
```

```
Training split input-  (1447, 1)
Testing split input-  (621, 1)
```

In [72]:
```
#dtree
dTree = DecisionTreeClassifier(random_state=1234)
model =dTree.fit(X_train,y_train)
```

In [73]:
```
plt.figure(figsize=(25, 20))
tree.plot_tree(
    dTree,
    feature_names=X.columns,           # ['value']
    class_names=y.unique(),            # unique country names
    filled=True
)
plt.show()
```



In [74]:
```
#predicting the values
from sklearn.metrics import classification_report
y_pred = dTree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
              precision    recall  f1-score   support

          BE       0.20      0.23      0.21        22
          FR       0.96      0.80      0.87       302
          GB       0.85      0.98      0.91       297

    accuracy                           0.87       621
   macro avg       0.67      0.67      0.67       621
weighted avg       0.88      0.87      0.87       621
```

In [75]:
```
#In this work we learned about training decission tree by classification,
#first I took the csv format from the database given to us called airquali
#i classified the data into value and country as X and Y and import as dec
#now changing it into text classification and splitting the data into tree
# then testing the training and testing sets with 80 percent data but usin
#finally prediction model was created with 0.87 percent accuracy.
```

In [ ]:
```

```