In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler , OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.metrics import silhouette_score
import seaborn as sns
```

In [2]:
```
pd.set_option('display.max_columns', None)
```

# HW:

The data set includes the churn of customers of a telecommunications company. The task is to create segments from customers based on their characteristics using the KMeans algorithm.

Do not use the following variables for grouping:

- churn?: has the customer dropped out?
- Contract_date: contract conclusion time
- Cust_ID: customer ID

In [3]:
```
file_path = "/content/telco_sampled.csv"
df = pd.read_csv(file_path, sep = ';')
```

In [4]:
```
df.head()
```

Out[4]:

| | Contract_date | Package | Gender | Age | Marital_Status | Living_Condition | Graduation | Job_Type | Income | Peak_minute_09 | Week |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9/20/04 12:00 AM | PACK_B | Male | 42.0 | Married | Owner | University | Leader | 15_30k | 0.55 | |
| **1** | 2/12/05 12:00 AM | PACK_B | Female | 53.0 | Married | Owner | University | Public_Employee | Below_15k | 11.32 | |
| **2** | 10/19/04 12:00 AM | PACK_X | Male | 43.0 | Married | Owner | Highschool | Executive | 30_60k | 78.05 | |
| **3** | 10/31/04 12:00 AM | PACK_B | Male | 32.0 | Married | Owner | Highschool | Labourer | 15_30k | 0.08 | |
| **4** | 11/19/04 12:00 AM | PACK_B | Female | 31.0 | Married | Owner | Highschool | Public_Employee | 30_60k | 20.68 | |

In [5]:
```
df['churn?'].value_counts()
```

Out[5]:

| churn? | count |
|---|---|
| **0** | 1224 |
| **1** | 341 |

**dtype:** int64

# 1. Subtask: (data preparation)

Use all variables except for the three variables above when creating the clusters. Perform data preparation so that the variables are input to the model in the appropriate form.

(hint: categorical variables, missing values, scaling, etc.)

In [6]:
```
# Dropping the columns not required for clustering
df = df.drop(columns=['churn?', 'Contract_date', 'Cust_ID'])
```

In [7]:
```
# Temporarily convert 'Age' to object type to avoid scaling
df['Age'] = df['Age'].astype('object')
```

In [8]:
```
# Handling categorical variables with OneHotEncoding and missing values
categorical_features = df.select_dtypes(include=['object']).columns.tolist()
numerical_features = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
```

In [9]:
```
# Now 'Age' will be in categorical features
print("Categorical Features:", categorical_features)
print("Numerical Features:", numerical_features)
```

```
Categorical Features: ['Package', 'Gender', 'Age', 'Marital_Status', 'Living_Condition', 'Graduation', 'Job_Type', 'Income']
Numerical Features: ['Peak_minute_09', 'Weekend_minute_09', 'Offpeak_minute_09', 'Offpeak_nr_09', 'Peak_nr_09', 'Weekend_nr_09', 'Selfnet_minute_09', 'Fixed_minute_09', 'Othermob_minute_09', 'Voicemail_nr_09', 'Voicemail_minute_09', 'SMS_09', 'Peak_minute_10', 'Weekend_minute_10', 'Offpeak_minute_10', 'Offpeak_nr_10', 'Peak_nr_10', 'Weekend_nr_10', 'Selfnet_minute_10', 'Fixed_minute_10', 'Othermob_minute_10', 'Voicemail_nr_10', 'Voicemail_minute_10', 'SMS_10', 'Peak_minute_11', 'Weekend_minute_11', 'Offpeak_minute_11', 'Offpeak_nr_11', 'Peak_nr_11', 'Weekend_nr_11', 'Selfnet_minute_11', 'Fixed_minute_11', 'Othermob_minute_11', 'Voicemail_nr_11', 'Voicemail_minute_11', 'SMS_11', 'Peak_minute_12', 'Weekend_minute_12', 'Offpeak_minute_12', 'Offpeak_nr_12', 'Peak_nr_12', 'Weekend_nr_12', 'Selfnet_minute_12', 'Fixed_minute_12', 'Othermob_minute_12', 'Voicemail_nr_12', 'Voicemail_minute_12', 'SMS_12']
```

In [10]:
```
# Apply scaling only to numerical columns
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

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

Out[12]:

| | Package | Gender | Age | Marital_Status | Living_Condition | Graduation | Job_Type | Income | Peak_minute_09 | Weekend_minute_09 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | PACK_B | Male | 42.0 | Married | Owner | University | Leader | 15_30k | -0.490696 | -0.521675 |
| **1** | PACK_B | Female | 53.0 | Married | Owner | University | Public_Employee | Below_15k | -0.290959 | -0.302073 |
| **2** | PACK_X | Male | 43.0 | Married | Owner | Highschool | Executive | 30_60k | 0.946595 | -0.394482 |
| **3** | PACK_B | Male | 32.0 | Married | Owner | Highschool | Labourer | 15_30k | -0.499413 | -0.531513 |
| **4** | PACK_B | Female | 31.0 | Married | Owner | Highschool | Public_Employee | 30_60k | -0.117372 | -0.044174 |

In [11]:
```
# Pipelines for numerical and categorical data processing
numerical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(drop='first'))
])

# Combine pipelines using ColumnTransformer
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_pipeline, numerical_features),
    ('cat', categorical_pipeline, categorical_features)
])
```

In [13]:
```
# Preprocess the data
X_preprocessed = preprocessor.fit_transform(df)
```

In [14]:
```
df['Age'] = df['Age'].astype('int64')
```

In [16]:
```
X_preprocessed_with_age = np.hstack((X_preprocessed, df[['Age']].values))
```

In [17]:
```
# Display shape and preview of preprocessed data
print("Preprocessed Data Shape:", X_preprocessed.shape)
print(pd.DataFrame(X_preprocessed).head())
```

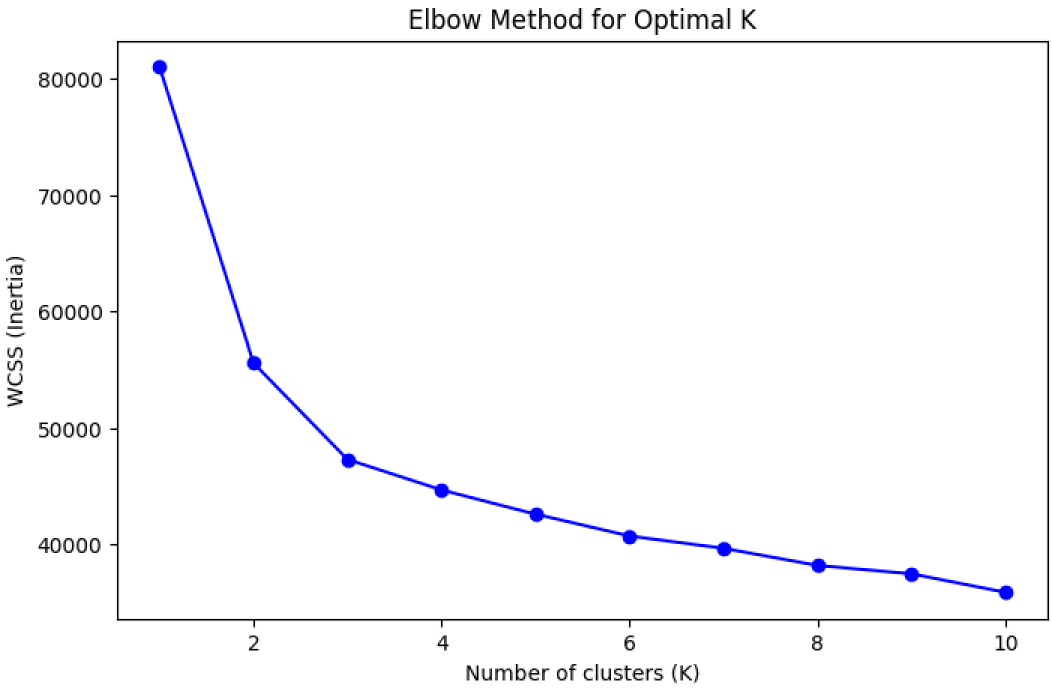
```
Preprocessed Data Shape: (1565, 130)
          0         1         2         3         4         5         6    \
0 -0.490696 -0.521675 -0.459272 -0.590807 -0.567464 -0.593901 -0.439902
1 -0.290959 -0.302073 -0.312966  0.204770  0.057757  0.073678 -0.337945
2  0.946595 -0.394482 -0.282573 -0.437812  1.236744 -0.297199 -0.112668
3 -0.499413 -0.531513 -0.459272 -0.590807 -0.585328 -0.630988 -0.452929
4 -0.117372 -0.044174  0.238093  0.908549 -0.067288  0.333292  0.203970

          7         8         9         10        11        12        13   \
0 -0.392252 -0.455270 -0.587854 -0.566961 -0.400519 -0.499263 -0.523609
1  0.339403 -0.318874 -0.375662 -0.379413  0.822817  0.041093 -0.363092
2 -0.134075  0.459603  0.744239  0.230140 -0.379427  0.011658 -0.051483
3 -0.392252 -0.455270 -0.611430 -0.573637 -0.400519 -0.499263 -0.523609
4 -0.343901 -0.145011  0.308067  0.029330  0.147873  0.385491  0.220606

          14        15        16        17        18        19        20   \
0 -0.452379 -0.565336 -0.591148 -0.608457 -0.487639 -0.277618 -0.448946
1 -0.071564  1.266374  0.970777  0.455122 -0.266674  0.493279 -0.131202
2 -0.387259 -0.248309  0.220242 -0.076668 -0.145313 -0.176438 -0.119198
3 -0.452379 -0.565336 -0.591148 -0.608457 -0.487639 -0.277618 -0.448946
4  0.317670  0.808446  0.544798  0.322174  0.821386 -0.185472 -0.080363

          21        22        23        24        25        26        27   \
0 -0.601178 -0.568592 -0.392033 -0.473401 -0.477615 -0.442899 -0.550587
1 -0.222053 -0.223253  2.288882  0.023522 -0.477615 -0.247176  1.402267
2  0.220259 -0.175703 -0.392033  0.610884 -0.436604 -0.397831 -0.282548
3 -0.601178 -0.568592 -0.392033 -0.473401 -0.477615 -0.442899 -0.550587
4  0.662572  0.334952  0.089694  0.144957  0.435284  0.287716  0.713025

          28        29        30        31        32        33        34   \
0 -0.534268 -0.551492 -0.452852 -0.309751 -0.433061 -0.575227 -0.540116
1  0.371554  0.154393 -0.417322  0.743390 -0.180469 -0.089204 -0.222023
2  0.648847 -0.410315 -0.324128  0.810843  0.118437  0.423819  0.022850
3 -0.534268 -0.551492 -0.452852 -0.309751 -0.433061 -0.575227 -0.540116
4  0.353068  1.072043  0.533448 -0.264782  0.041806  0.815338  0.297378

          35        36        37        38        39        40        41   \
0 -0.353420 -0.472188 -0.361515 -0.449857 -0.577982 -0.609884 -0.596913
1  1.391981 -0.250612 -0.272427 -0.339479  0.304567  0.210353  0.322726
2 -0.332392 -0.287325 -0.147455 -0.340768 -0.472076 -0.158754  0.065227
3 -0.353420 -0.472188 -0.361515 -0.449857 -0.577982 -0.609884 -0.596913
4  0.088187  0.448977  0.526895  0.309636  1.857853  0.989578  1.132007

          42        43        44        45        46        47   48   49   50   \
0 -0.429383 -0.245480 -0.454958 -0.588222 -0.543702 -0.435443  1.0  0.0  0.0
1 -0.412159  0.112460 -0.266448 -0.311383 -0.357744  0.975686  1.0  0.0  0.0
2 -0.374192 -0.223542 -0.315431 -0.179556 -0.336739 -0.178874  0.0  0.0  0.0
3 -0.429383 -0.245480 -0.454958 -0.588222 -0.543702 -0.435443  1.0  0.0  0.0
4  0.888704 -0.122510  0.005926  1.006896  0.500963  0.932924  1.0  0.0  0.0

   51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   \
0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80   \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
1  0.0  0.0  0.0  0.0  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  0.0  0.0  0.0  0.0  1.0
3  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  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  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

   96   97   98   99   100  101  102  103  104  105  106  107  108  109  110  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  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  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

   111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  \
0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  1.0  0.0  0.0
1  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0
2  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
4  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

```
   126  127  128  129
0  0.0  0.0  0.0  0.0
1  0.0  0.0  1.0  0.0
2  0.0  1.0  0.0  0.0
3  0.0  0.0  0.0  0.0
4  0.0  1.0  0.0  0.0
```

# 2. Subtask: (clustering)

Find the optimal k value for the KMeans algorithm using the variables prepared in the previous task. Then group the customers.

In [18]:
```
# Function to apply only the Elbow Method
def elbow_method(X):
    wcss = []  # Within-cluster sum of squares (WCSS) for the elbow method
    K_values = range(1, 11)  # Trying out K values from 1 to 10

    # Loop to calculate WCSS for each K value
    for k in K_values:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)

    # Plotting the elbow graph
    plt.figure(figsize=(8, 5))
    plt.plot(K_values, wcss, 'bo-')
    plt.title('Elbow Method for Optimal K')
    plt.xlabel('Number of clusters (K)')
    plt.ylabel('WCSS (Inertia)')
    plt.show()

# Assuming X_preprocessed is already available from previous steps
elbow_method(X_preprocessed)
```

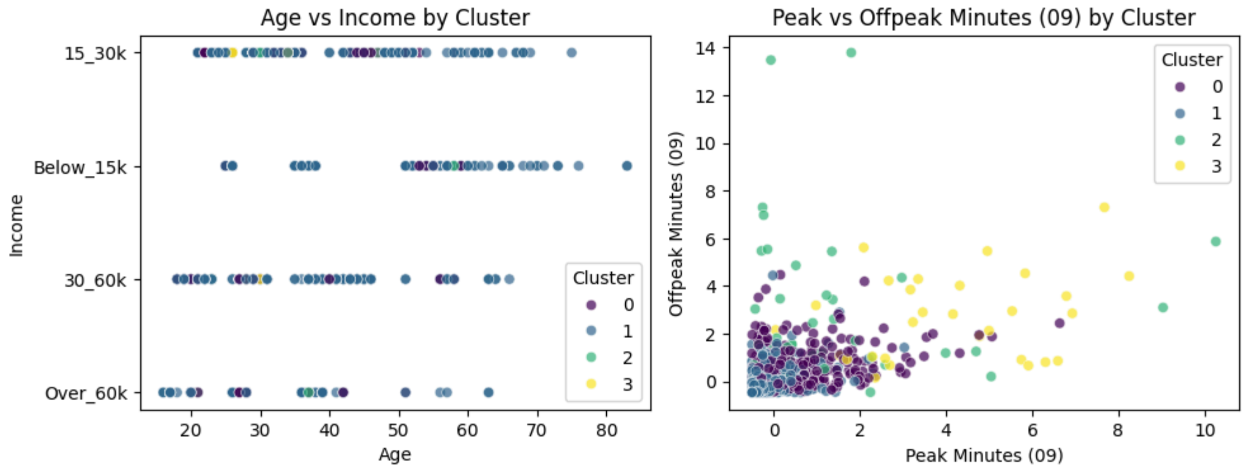



In [19]:
```
# The optmal k is 4
# Apply KMeans clustering with the optimal number of clusters
k = 4
kmeans = KMeans(n_clusters=k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_preprocessed)
```

Let's generate graphs for different customer segments based on their characteristics.

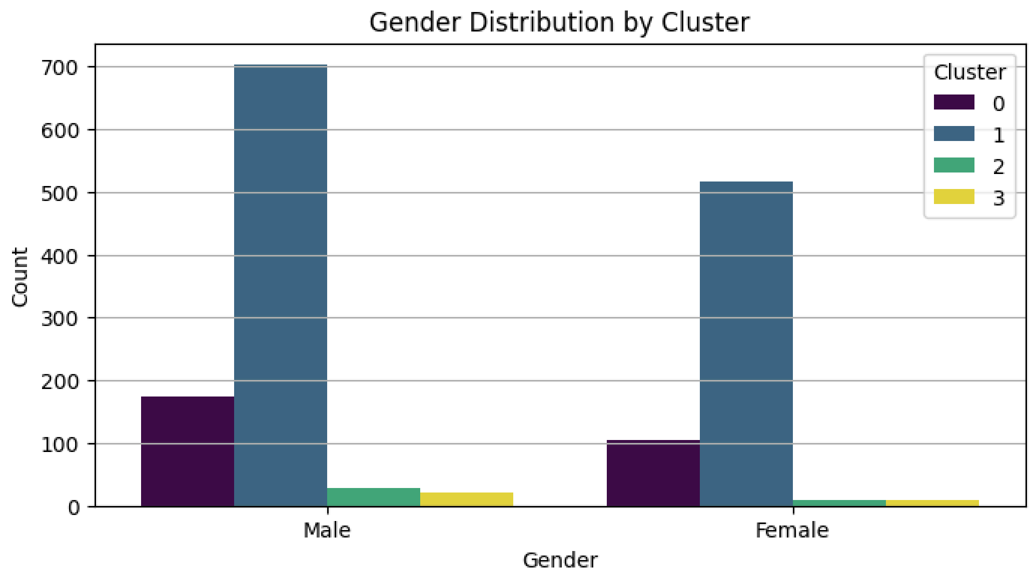
In [20]:
```
# Create scatter plots for key feature relationships
plt.figure(figsize=(10, 7))

# Scatter plot of Age vs Income colored by Cluster
plt.subplot(2, 2, 1)
sns.scatterplot(data=df, x='Age', y='Income', hue='Cluster', palette='viridis', alpha=0.7)
plt.title('Age vs Income by Cluster')
plt.xlabel('Age')
plt.ylabel('Income')

# Scatter plot of Peak_minute_09 vs Offpeak_minute_09 colored by Cluster
```

```python
plt.subplot(2, 2, 2)
sns.scatterplot(data=df, x='Peak_minute_09', y='Offpeak_minute_09', hue='Cluster', palette='viridis', alpha=0.7)
plt.title('Peak vs Offpeak Minutes (09) by Cluster')
plt.xlabel('Peak Minutes (09)')
plt.ylabel('Offpeak Minutes (09)')

plt.tight_layout()
plt.show()
```




In [22]:
```python
# Count plot for Gender distribution across clusters
plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='Gender', hue='Cluster', palette='viridis')
plt.title('Gender Distribution by Cluster')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Cluster')
plt.grid(axis='y')
plt.show()
```

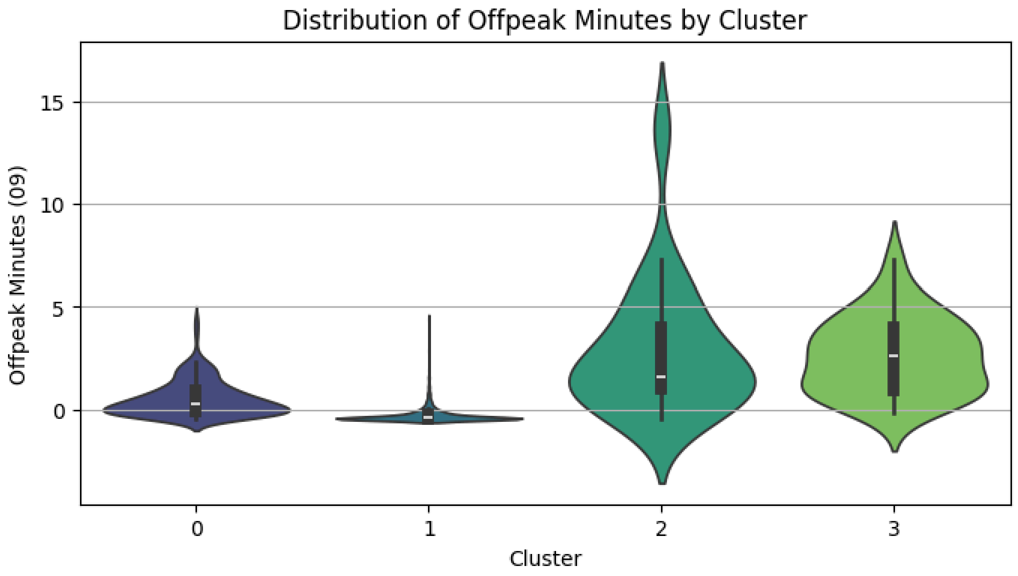
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-lik
e, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `nam
e` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-lik
e, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `nam
e` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-lik
e, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `nam
e` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-lik
e, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `nam
e` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```




In [23]:
```python
# Violin plot for Offpeak Minutes by Cluster
plt.figure(figsize=(8, 4))
sns.violinplot(data=df, x='Cluster', y='Offpeak_minute_09', palette='viridis')
plt.title('Distribution of Offpeak Minutes by Cluster')
```

```
plt.xlabel('Cluster')
plt.ylabel('Offpeak Minutes (09)')
plt.grid(axis='y')
plt.show()
```

```
<ipython-input-23-368e32be3b7c>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(data=df, x='Cluster', y='Offpeak_minute_09', palette='viridis')
```




# 3. Subtask: (explaination of clusters / conclusions)

Try to find an explanation of what characterizes each group and what characteristics caused each customer to be in the given cluster.

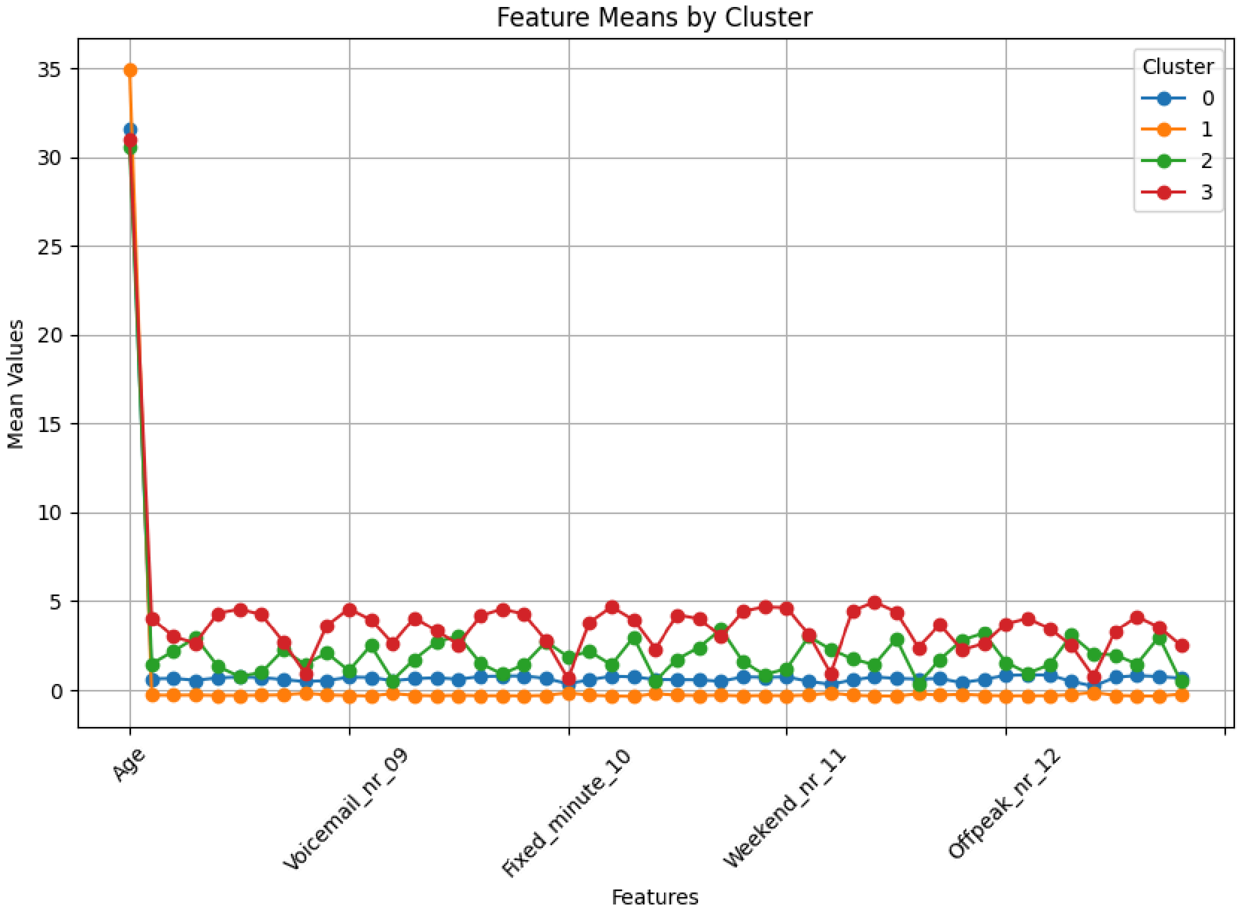
In [24]:
```
# Analyze cluster characteristics
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
cluster_summary = df.groupby('Cluster')[numerical_columns].mean()
# Display cluster summary
print(cluster_summary)
```

```
               Age  Peak_minute_09  Weekend_minute_09  Offpeak_minute_09  \
Cluster
0        31.557554        0.623515           0.648169           0.542985
1        34.912151       -0.288483          -0.291217          -0.282465
2        30.564103        1.466402           2.179957           2.933920
3        31.000000        4.028172           2.983102           2.622342

         Offpeak_nr_09  Peak_nr_09  Weekend_nr_09  Selfnet_minute_09  \
Cluster
0             0.679826    0.728985       0.697765           0.583021
1            -0.304110   -0.302754      -0.295190          -0.271121
2             1.330660    0.761302       0.971390           2.240069
3             4.317289    4.546841       4.255933           2.692765

         Fixed_minute_09  Othermob_minute_09  Voicemail_nr_09  \
Cluster
0               0.469722            0.529366         0.713479
1              -0.175707           -0.276530        -0.306159
2               1.428487            2.072138         1.078806
3               0.923905            3.627901         4.568326

         Voicemail_minute_09    SMS_09  Peak_minute_10  Weekend_minute_10  \
Cluster
0                   0.709356  0.513341        0.631212           0.686603
1                  -0.337058 -0.198136       -0.299180          -0.324594
2                   2.491617  0.534560        1.717162           2.685880
3                   3.955140  2.592436        4.065163           3.324369

         Offpeak_minute_10  Offpeak_nr_10  Peak_nr_10  Weekend_nr_10  \
Cluster
0                 0.601369       0.754595    0.776540       0.788327
1                -0.296855      -0.323040   -0.318153      -0.330692
2                 3.020337       1.501208    0.895880       1.416433
3                 2.553179       4.171277    4.556444       4.279573

         Selfnet_minute_10  Fixed_minute_10  Othermob_minute_10  \
Cluster
0                 0.649073         0.346518            0.567723
1                -0.302837        -0.154118           -0.292401
2                 2.708910         1.857020            2.152563
3                 2.758863         0.631981            3.812239

         Voicemail_nr_10  Voicemail_minute_10    SMS_10  Peak_minute_11  \
Cluster
0               0.773335             0.732817  0.585660        0.577917
1              -0.332966            -0.359185 -0.207297       -0.291288
2               1.405467             2.959405  0.525772        1.718338
3               4.681107             3.944900  2.305638        4.237088

         Weekend_minute_11  Offpeak_minute_11  Offpeak_nr_11  Peak_nr_11  \
Cluster
0                 0.569108           0.484439       0.737818    0.710622
1                -0.305210          -0.293397      -0.327976   -0.305103
2                 2.390085           3.406594       1.565251    0.866889
3                 4.010681           2.994212       4.443870    4.675133

         Weekend_nr_11  Selfnet_minute_11  Fixed_minute_11  \
Cluster
0             0.734462           0.479425         0.320243
1            -0.320051          -0.281590        -0.168695
2             1.188785           2.990417         2.268665
3             4.642643           3.102333         0.932162

         Othermob_minute_11  Voicemail_nr_11  Voicemail_minute_11    SMS_11  \
Cluster
0                  0.555094         0.729924             0.649074  0.611566
1                 -0.292224        -0.329638            -0.344725 -0.210085
2                  1.747592         1.410299             2.865366  0.363181
3                  4.448558         4.950998             4.402848  2.390154

         Peak_minute_12  Weekend_minute_12  Offpeak_minute_12  Offpeak_nr_12  \
Cluster
0              0.640959           0.407556           0.595196       0.817334
1             -0.291453          -0.237706          -0.299381      -0.327767
2              1.704663           2.791620           3.186739       1.550092
3              3.677387           2.245081           2.602870       3.718265

         Peak_nr_12  Weekend_nr_12  Selfnet_minute_12  Fixed_minute_12  \
Cluster
0          0.845152       0.839311           0.483180         0.208477
1         -0.322296      -0.321867          -0.267422        -0.130949
2          0.935947       1.408371           3.068274         2.024555
3          4.036760       3.459306           2.493391         0.752731

         Othermob_minute_12  Voicemail_nr_12  Voicemail_minute_12    SMS_12
```

```
Cluster
0                 0.712576         0.803938            0.750995  0.664358
1                -0.304043        -0.331731           -0.345962 -0.228043
2                 1.965588         1.454773            2.977059  0.465837
3                 3.295514         4.127263            3.532892  2.496569
```

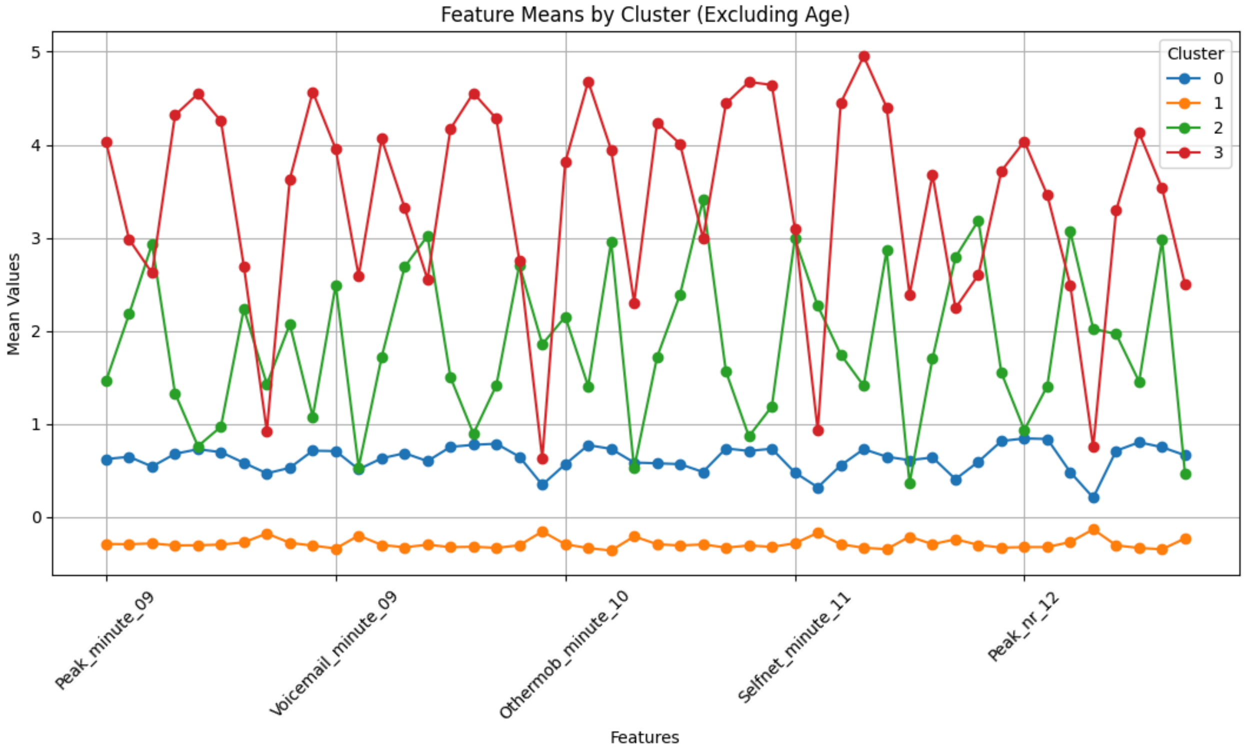
In [25]:

```python
# Visualizing the mean values for each feature across clusters
cluster_summary.T.plot(marker='o', figsize=(10, 6))
plt.xticks(rotation=45)
plt.title('Feature Means by Cluster')
plt.xlabel('Features')
plt.ylabel('Mean Values')
plt.legend(title='Cluster')
plt.grid()
plt.show()
```




In [36]:

```python
# Exclude the Age feature for better visualization
cluster_summary_excluding_age = cluster_summary.drop(columns=['Age'])

# Visualizing the mean values for each feature across clusters (excluding Age)
cluster_summary_excluding_age.T.plot(marker='o', figsize=(13, 6))
plt.xticks(rotation=45)
plt.title('Feature Means by Cluster (Excluding Age)')
plt.xlabel('Features')
plt.ylabel('Mean Values')
plt.legend(title='Cluster')
plt.grid()
plt.show()
```




In [33]:
```python
# Generate characteristics for each cluster
for cluster in range(k):
    print(f"\nCluster {cluster} Characteristics:")
    characteristics = cluster_summary.loc[cluster]

    # Print Age and other characteristics
    print(f"- Average Age: {characteristics['Age']:.1f} years")
    print(f"- Peak Minutes: {characteristics['Peak_minute_09']:.2f}")
    print(f"- Off-Peak Minutes: {characteristics['Offpeak_minute_09']:.2f}")
    print(f"- Weekend Minutes: {characteristics['Weekend_minute_09']:.2f}")
    print(f"- Selfnet Minutes: {characteristics['Selfnet_minute_09']:.2f}")
    print(f"- Othermob Minutes: {characteristics['Othermob_minute_09']:.2f}")
```

```
Cluster 0 Characteristics:
- Average Age: 31.6 years
- Peak Minutes: 0.62
- Off-Peak Minutes: 0.54
- Weekend Minutes: 0.65
- Selfnet Minutes: 0.58
- Othermob Minutes: 0.53

Cluster 1 Characteristics:
- Average Age: 34.9 years
- Peak Minutes: -0.29
- Off-Peak Minutes: -0.28
- Weekend Minutes: -0.29
- Selfnet Minutes: -0.27
- Othermob Minutes: -0.28

Cluster 2 Characteristics:
- Average Age: 30.6 years
- Peak Minutes: 1.47
- Off-Peak Minutes: 2.93
- Weekend Minutes: 2.18
- Selfnet Minutes: 2.24
- Othermob Minutes: 2.07

Cluster 3 Characteristics:
- Average Age: 31.0 years
- Peak Minutes: 4.03
- Off-Peak Minutes: 2.62
- Weekend Minutes: 2.98
- Selfnet Minutes: 2.69
- Othermob Minutes: 3.63
```

- Cluster 0: Moderately active users with balanced usage of peak and off-peak minutes. Average age is 31.6 years, likely representing young professionals with steady communication needs.
- Cluster 1: Low usage of services, indicated by negative values for peak and off-peak minutes. Average age is 34.9 years, suggesting infrequent users who may prefer alternative communication methods. Marketing efforts should aim to boost engagement.

- Cluster 2: Heavy users of mobile services, with high usage across all metrics. Average age is 30.6 years, indicating active individuals, possibly business users. Loyalty programs could help retain this segment.
- Cluster 3: High usage, particularly during peak hours, with an average age of 31.0 years. This group may consist of professionals who rely on their devices during work hours. Targeted promotions for high usage plans could be beneficial.