In [95]:

```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from scipy import stats
```

Data Preparation

In [96]:

```python
# Read the data
august_data = pd.read_excel('August2015.xlsx')
september_data = pd.read_excel('September2015.xlsx')

# Merge the data
combined_data = pd.merge(august_data, september_data, on='ID', how='outer', suffixes=('_aug', '_sep'))
combined_data.head(5)
```

Out[96]:

	ID	SUBS_ACTIVATION_DATE_KEY_aug	REVENUE_TOTAL_aug	REVENUE_VOICE_aug	REVENUE_DATA_aug	TRAFFIC_DATA_aug	MARKET_KEY_aug	MOU_aug	SUBS_ACTIVATION_DATE_KEY_sep	REVENUE_TOTAL_sep	REVENUE_VOICE_sep	REVENUE_DATA_sep	TRAFFIC_DATA_sep	MA
0	1	2011-12-02	11.075	6.562	4.513	0.247	KZT	40.250	2011-12-02	0.000	0.000	0.000	0.0	
1	2	2012-12-27	1940.036	1090.420	1.402	0.077	KZT	445.882	2012-12-27	1931.732	503.161	0.000	0.0	
2	3	2010-11-21	1127.000	631.242	0.000	0.000	KOS	333.233	2010-11-21	549.937	315.117	0.000	0.0	
3	4	2012-02-15	142.286	142.286	0.000	0.000	KZO	479.000	2012-02-15	683.527	652.777	0.000	0.0	
4	5	2006-08-01	473.233	169.661	44.643	0.000	KOS	131.884	2006-08-01	666.732	357.911	44.643	0.0	

Variables Descriptions:

The text at the bottom seems to describe the variables in the original dataset before PCA was applied. ID is a unique identifier for a subscriber. SUBS_ACTIVATION_DATE_KEY is the date when the SIM card was activated. REVENUE_TOTAL is the total monthly expenditure of a subscriber in tenge (currency of Kazakhstan). REVENUE_VOICE is the monthly expenditure on voice services. REVENUE_DATA is the monthly expenditure on data transfer services. TRAFFIC_DATA is the data traffic in megabytes per month. MOU is the monthly voice traffic in minutes. MARKET_KEY is a code representing the subscriber's region, with a list of codes corresponding to various regions in Kazakhstan.

In [98]:

```python
# Data preparation
# Separate numerical and non-numerical data
numerical_data = combined_data.select_dtypes(include=[np.number])
non_numerical_data = combined_data.select_dtypes(exclude=[np.number])

# Handle missing values in numerical data
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
numerical_data_imputed = pd.DataFrame(imputer.fit_transform(numerical_data), columns=numerical_data.columns)

# Calculate Z-scores for outlier detection
z_scores = np.abs(stats.zscore(numerical_data_imputed))

# Filter out the outliers
numerical_data_no_outliers = numerical_data_imputed[(z_scores < 3).all(axis=1)]

# Standardize the numerical data without outliers
scaler = StandardScaler()
scaled_features_no_outliers = scaler.fit_transform(numerical_data_no_outliers)

combined_data_no_outliers = pd.concat([non_numerical_data.reset_index(drop=True),
                                       pd.DataFrame(scaled_features_no_outliers, columns=numerical_data_no_outliers.columns)],
                                      axis=1)
numerical_data_no_outliers.head(5)
```

Out[98]:

	ID	REVENUE_TOTAL_aug	REVENUE_VOICE_aug	REVENUE_DATA_aug	TRAFFIC_DATA_aug	MOU_aug	REVENUE_TOTAL_sep	REVENUE_VOICE_sep	REVENUE_DATA_sep	TRAFFIC_DATA_sep	MOU_sep
0	1.0	11.075	6.562	4.513	0.247	40.250	0.000	0.000	0.000	0.0	0.600
1	2.0	1940.036	1090.420	1.402	0.077	445.882	1931.732	503.161	0.000	0.0	495.283
2	3.0	1127.000	631.242	0.000	0.000	333.233	549.937	315.117	0.000	0.0	471.883
3	4.0	142.286	142.286	0.000	0.000	479.000	683.527	652.777	0.000	0.0	666.750
4	5.0	473.233	169.661	44.643	0.000	131.884	666.732	357.911	44.643	0.0	112.383

Principal Component Analysis

In [114…

```python
# Here we'll start by choosing a number that retains significant variance 95%.
pca = PCA(n_components=0.95)
pca.fit(scaled_features_no_outliers)

# Transform the scaled features using PCA
pca_features = pca.fit_transform(scaled_features_no_outliers)

# Check how many components were selected
n_components = pca.n_components_
explained_variance = pca.explained_variance_ratio_

# Now we can look at the reduced features
pca_features_df = pd.DataFrame(pca_features, columns=[f'PC{i+1}' for i in range(n_components)])

# Output the PCA results
n_components, explained_variance
```

Out[114…

```
(8,
 array([0.34891664, 0.18897505, 0.10210877, 0.0897073 , 0.08542494,
        0.06420075, 0.04577503, 0.03181258]))
```

In [112…

```python
pca_features_df.head(5)
```

Out[112…

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	-2.301654	0.053713	-0.404847	-1.494648	-0.011128	0.319428	-0.045730	0.040415
1	2.724159	-1.058107	-0.303048	-1.824205	0.113806	0.737545	0.287865	0.189906
2	0.536623	-0.677134	-0.855250	-1.566082	0.568733	0.369428	0.237290	0.097291
3	0.306931	-0.946965	-1.981642	-1.330540	0.188363	-0.133498	-1.067075	0.328922
4	-0.795418	-0.167563	-0.246449	-1.649674	-0.311539	0.328273	-0.345036	0.179601

Number of Principal Components (PCs): 8 indicates that seven principal components were retained after applying PCA. This number of components is enough to explain 95% of the variance in your data, given the cumulative sum of the variance explained by these components.

Explained Variance Ratio: The array of values represents the proportion of the dataset's variance that each principal component accounts for. For example, the first principal component (PC1) explains approximately 34.89% of the variance in the dataset, and the second (PC2) explains about 18.90%, and so on. The sum of these values should be close to 0.95 (or 95%), which is the amount of total variance you aimed to capture with PCA.

Cluster analysis

In [103…

```python
# Loadings are the weights used in the PCA linear combination, available in the components_ attribute
loadings = pca.components_

# Create a DataFrame of the loadings
loadings_df = pd.DataFrame(loadings.T, columns=[f'PC{i+1}' for i in range(n_components)],
                           index=numerical_data_no_outliers.columns)
loadings_df
```

Out[103…

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
ID	0.062456	-0.068494	0.291758	0.924627	-0.058192	-0.214449	0.007596	0.000156
REVENUE_TOTAL_aug	0.428947	0.095483	0.269759	-0.065257	0.249003	0.094281	0.267260	0.119761
REVENUE_VOICE_aug	0.395241	-0.217719	0.250588	-0.088195	0.051672	0.236106	0.455409	-0.284214
REVENUE_DATA_aug	0.174281	0.472716	0.322146	-0.126223	0.338066	-0.285707	-0.072604	0.495267
TRAFFIC_DATA_aug	0.105222	0.503541	-0.026031	0.167150	0.282239	0.470896	-0.443419	-0.450022
MOU_aug	0.344243	-0.122480	-0.434464	0.099931	0.398573	-0.225202	0.195887	-0.241995
REVENUE_TOTAL_sep	0.432652	-0.041705	0.062232	-0.048677	-0.394174	0.021784	-0.284324	0.137769
REVENUE_VOICE_sep	0.377317	-0.286159	0.177030	-0.071438	-0.257582	0.196375	-0.376219	0.049369
REVENUE_DATA_sep	0.183415	0.421971	0.013615	-0.147098	-0.439428	-0.548649	0.071952	-0.470838
TRAFFIC_DATA_sep	0.111825	0.413296	-0.390402	0.210177	-0.386700	0.393754	0.416388	0.325733
MOU_sep	0.346754	-0.119480	-0.546157	0.076720	0.121560	-0.200486	-0.282695	0.222040

Hypotheses and Conclusions: Based on the loadings table, here are some potential hypotheses and conclusions:

High Spending on Voice and Data: The strong loadings of REVENUE_TOTAL_aug, REVENUE_VOICE_aug, and REVENUE_DATA_aug on PC1 suggest that these features are closely related. A hypothesis could be that subscribers with high total revenue also tend to spend significantly on voice and data services.

Different Usage Patterns: The contrast between the loadings of TRAFFIC_DATA_aug and MOU_aug on PC3 indicates different usage patterns. For instance, one might hypothesize that subscribers with high data traffic have lower minutes of use (MOU), suggesting a preference for data over voice services.

Segmentation of Subscribers: The PCA results could be used to segment subscribers into distinct groups based on spending and usage patterns. For example, one segment may be characterized by high revenue and voice usage, while another by high data usage.

Change in Behavior Over Time: By comparing aug (August) and sep (September) features, you could analyze changes in subscriber behavior over time. For instance, if REVENUE_TOTAL_sep has a different loading pattern compared to REVENUE_TOTAL_aug, it could suggest a change in spending behavior from August to September.

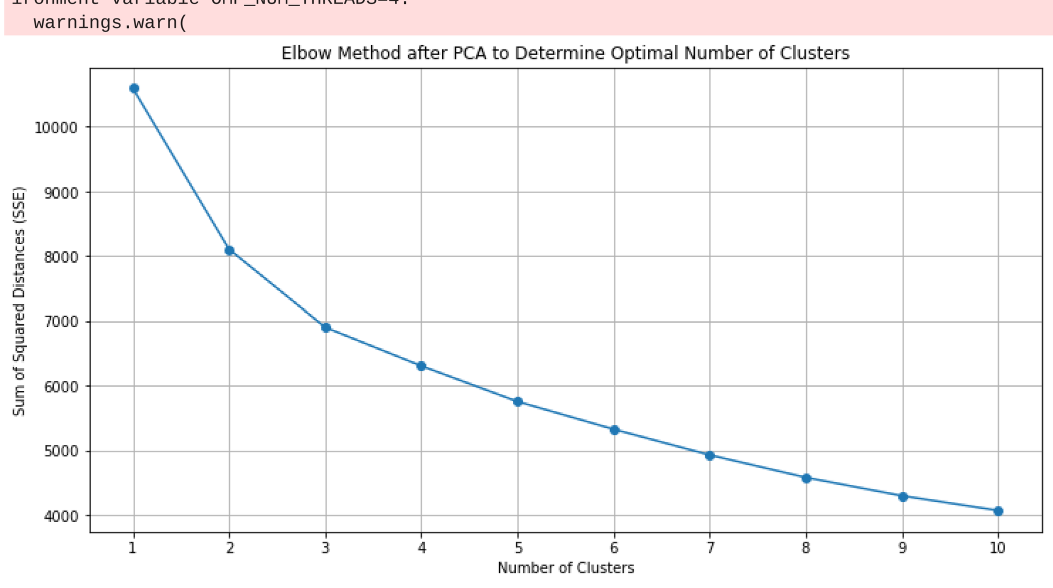
Impact of New Services: If the dataset includes information before and after the introduction of a new service (like the international calling service mentioned in the task), one could examine how the loadings change for related features. This could indicate whether the new service is influencing spending or traffic patterns.

In [58]:

```python
sse_pca = []
for k in range(1, 11):
    kmeans_pca = KMeans(n_clusters=k, random_state=42)
    kmeans_pca.fit(scaled_features_no_outliers)
    sse_pca.append(kmeans_pca.inertia_)

# Plot SSE for each k after PCA
plt.figure(figsize=(12, 6))
plt.plot(range(1, 11), sse_pca, marker='o')
plt.title('Elbow Method after PCA to Determine Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of Squared Distances (SSE)')
plt.xticks(range(1, 11))
plt.grid(True)
plt.show()
```

```
C:\Users\1\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: 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 env
ironment variable OMP_NUM_THREADS=4.
  warnings.warn(
```



In the Elbow Method graph for determining the optimal number of clusters, the point where the sum of squared distances (SSE) starts to decrease at a diminishing rate is typically chosen as the right number of clusters. This point is referred to as the "elbow" because the SSE plot resembles an arm with an elbow bend.

Upon analyzing the graph, the SSE begins to level off after **3** clusters, indicating that additional clusters do not significantly improve the compactness of the clustering. Consequently, 3 clusters seem to be the most reasonable choice for this dataset, balancing the complexity of the model with the distinctness of the clusters. This selection is based on visual inspection and should be complemented by domain knowledge and additional cluster validation metrics, such as the Silhouette Score, for a more comprehensive evaluation.

In [77]:

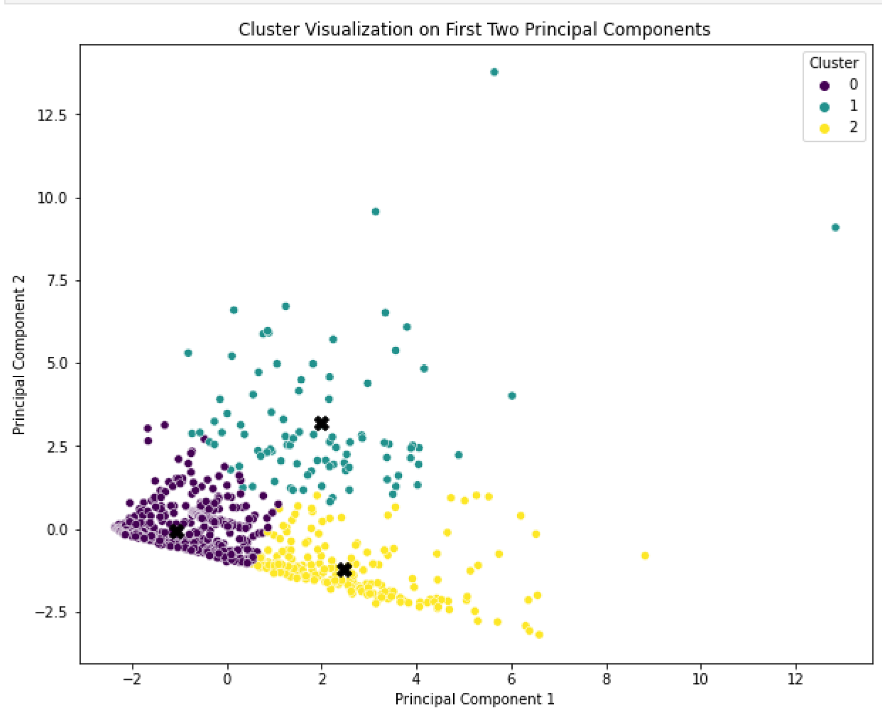
```python
# Apply K-Means using the optimal number of clusters found from the elbow method
optimal_k = 3
kmeans_optimal_pca = KMeans(n_clusters=optimal_k, random_state=42)
clusters_pca = kmeans_optimal_pca.fit_predict(pca_features)

# Adding the cluster assignments to the PCA features dataframe
pca_features_df['Cluster'] = clusters_pca

# Visualize the clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(x='PC1', y='PC2', data=pca_features_df, hue='Cluster', palette='viridis')

# Plot the centroids
centroids = kmeans_optimal_pca.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], s=100, c='black', marker='X')

plt.title('Cluster Visualization on First Two Principal Components')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.show()
```



In [104…

```python
# Now let's map the cluster centroids back to the original feature space
centroids_original_space = pca.inverse_transform(centroids)

# Create a DataFrame of the centroids in the original feature space
centroids_df = pd.DataFrame(centroids_original_space,
                            columns=numerical_data_no_outliers.columns)
centroids_df
```

Out[104…

	ID	REVENUE_TOTAL_aug	REVENUE_VOICE_aug	REVENUE_DATA_aug	TRAFFIC_DATA_aug	MOU_aug	REVENUE_TOTAL_sep	REVENUE_VOICE_sep	REVENUE_DATA_sep	TRAFFIC_DATA_sep	MOU_sep
0	-0.030704	-0.424315	-0.376559	-0.175704	-0.147309	-0.352930	-0.480910	-0.393030	-0.240341	-0.195340	-0.391547
1	-0.129928	0.987907	0.023511	1.644168	1.593673	0.279961	0.862182	-0.156038	2.017600	1.801834	0.389131
2	0.155815	0.908754	1.187961	-0.175920	-0.243717	0.998091	1.145094	1.320661	-0.137170	-0.183919	1.072186

Cluster 0 (Low Utilization Cluster):

Characteristics: This cluster has negative mean values for almost all features, indicating lower usage and expenditure across the board compared to the other clusters. Hypothesis: Subscribers in this cluster might represent a segment with minimal usage or lower engagement with services. They might be more cost-conscious or occasional users. Conclusion: Targeting this group could involve strategies to increase engagement or upsell additional services.

Cluster 1 (High Spending on Voice Services Cluster):

Characteristics: This cluster shows particularly high mean values for REVENUE_VOICE_sep, REVENUE_TOTAL_aug, and TRAFFIC_DATA_aug, indicating significant spending on voice services and high data traffic in August. Hypothesis: Subscribers in this cluster are likely heavy users of voice services and possibly have a higher willingness to pay for these services. Conclusion: This segment could be targeted with premium voice service packages or loyalty rewards to maintain their high usage levels.

Cluster 2 (High Data Usage Cluster):

Characteristics: Exhibits high mean values for REVENUE_DATA_aug, TRAFFIC_DATA_sep, and MOU_sep, suggesting high expenditure on data services and significant data traffic. Hypothesis: These subscribers are possibly heavy internet users, likely utilizing data-intensive applications. Conclusion: Offering data-centric plans or promotions, especially for high-speed or unlimited data, could appeal to this segment.