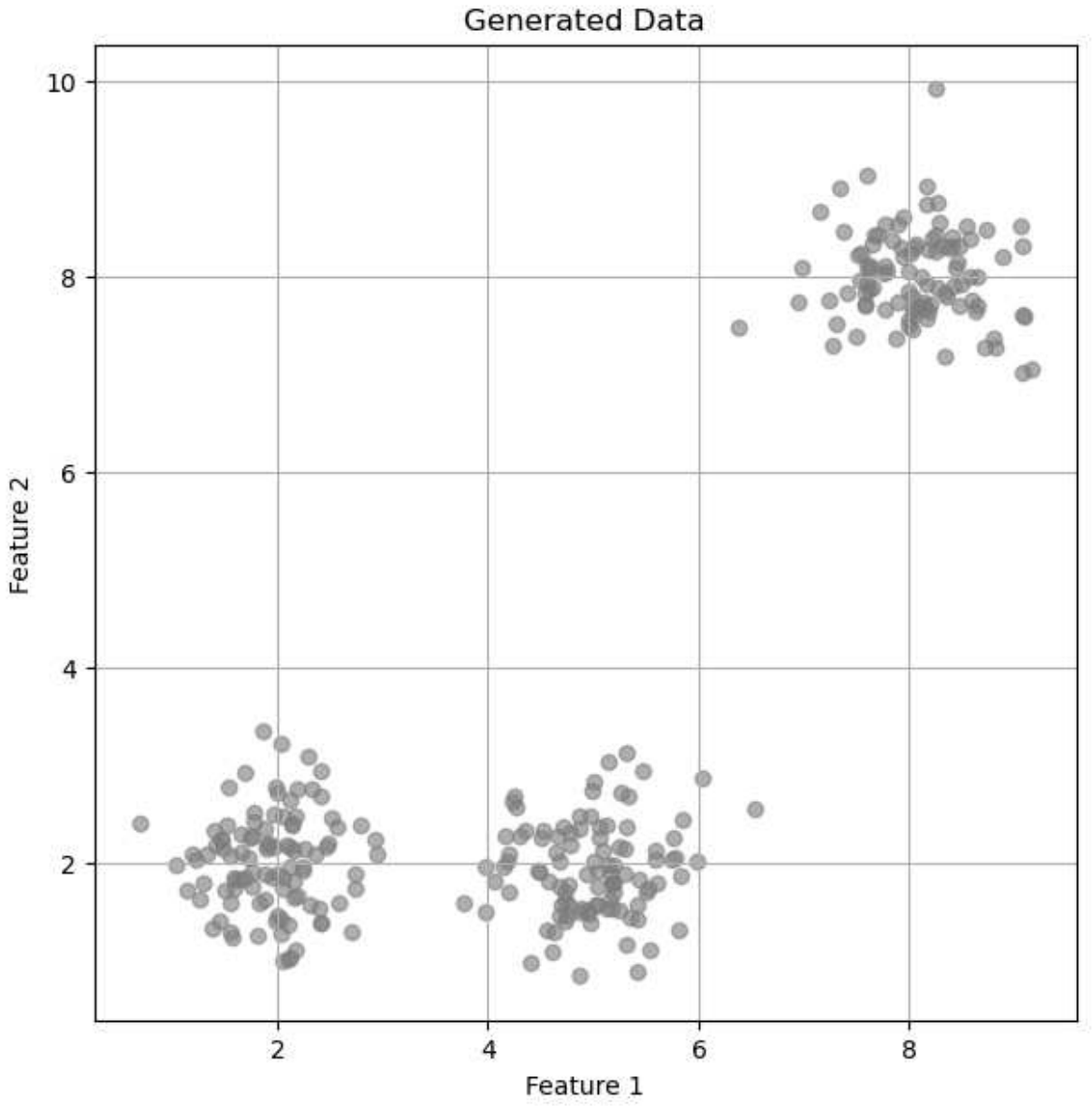
In [2]:
```python
# 라이브러리 불러오기
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# 임의의 데이터 생성
np.random.seed(42)  # 재현성을 위한 시드 고정
data_1 = np.random.normal(loc=[2, 2], scale=0.5, size=(100, 2))  # 첫 번째 군집
data_2 = np.random.normal(loc=[8, 8], scale=0.5, size=(100, 2))  # 두 번째 군집
data_3 = np.random.normal(loc=[5, 2], scale=0.5, size=(100, 2))  # 세 번째 군집

# 데이터를 하나로 합치기
data = np.vstack([data_1, data_2, data_3])

# 데이터 시각화
plt.figure(figsize=(7, 7))
plt.scatter(data[:, 0], data[:, 1], color='gray', alpha=0.6)
plt.title('Generated Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.show()
```



In [24]:
```
# K-Means 모델 생성 및 학습
kmeans = KMeans(n_clusters=3, random_state=42)  # 군집 수 설정
kmeans.fit(data)  # 데이터 학습

# 군집화 결과
labels = kmeans.labels_  # 각 데이터 포인트의 군집 할당 결과
centroids = kmeans.cluster_centers_  # 각 군집의 중심점
```

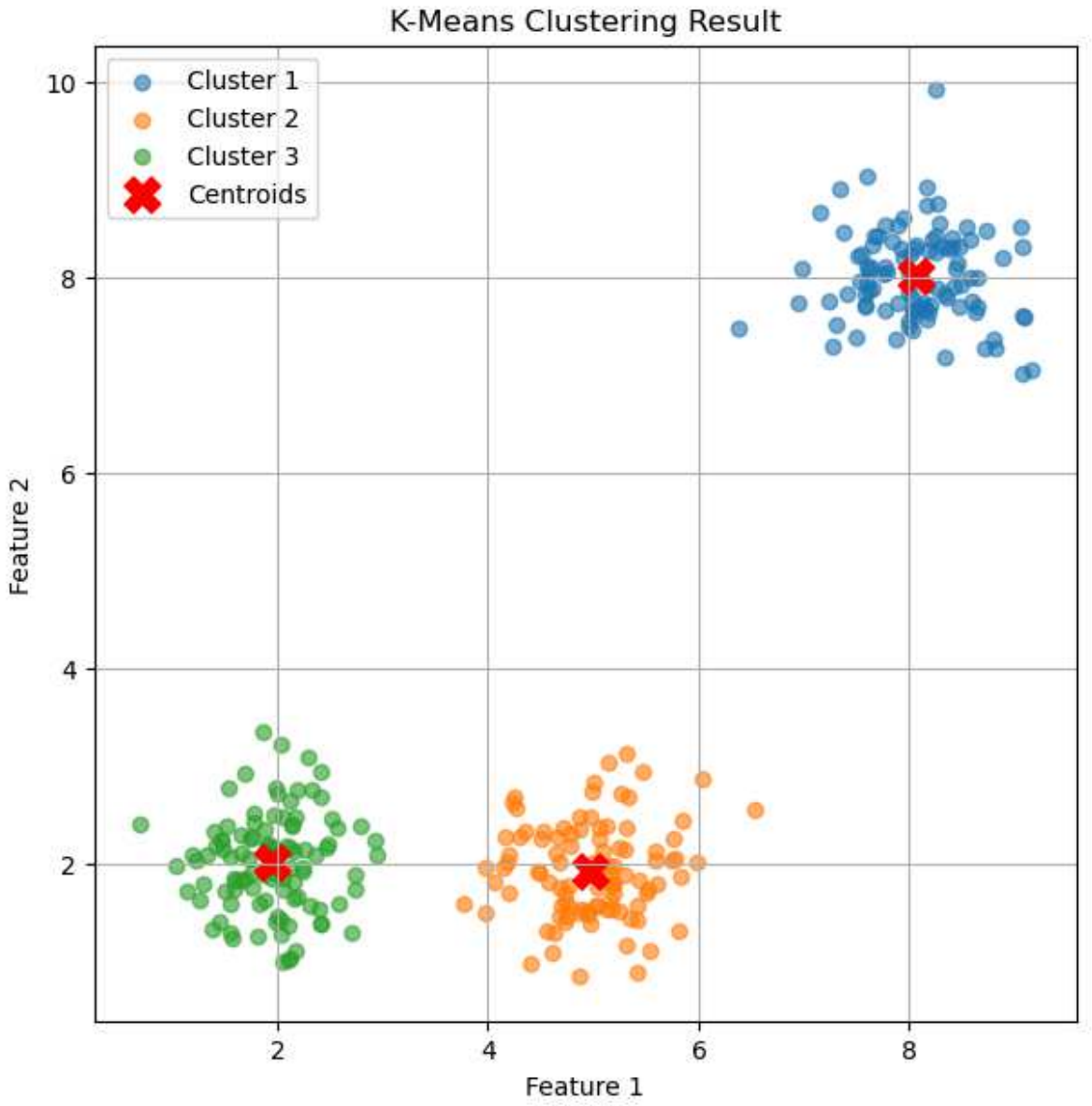
In [26]:
```
# 각 데이터 포인트의 군집 할당 결과
labels = kmeans.labels_
labels
```

Out[26]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 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, 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, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 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, 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, 1], dtype=int32)
```

In [28]:
```
# 각 군집의 중심점
centroids = kmeans.cluster_centers_
centroids
```

Out[28]:
```
array([[8.06412436, 8.02174382],
       [4.97748131, 1.93686366],
       [1.94221787, 2.01701116]])
```

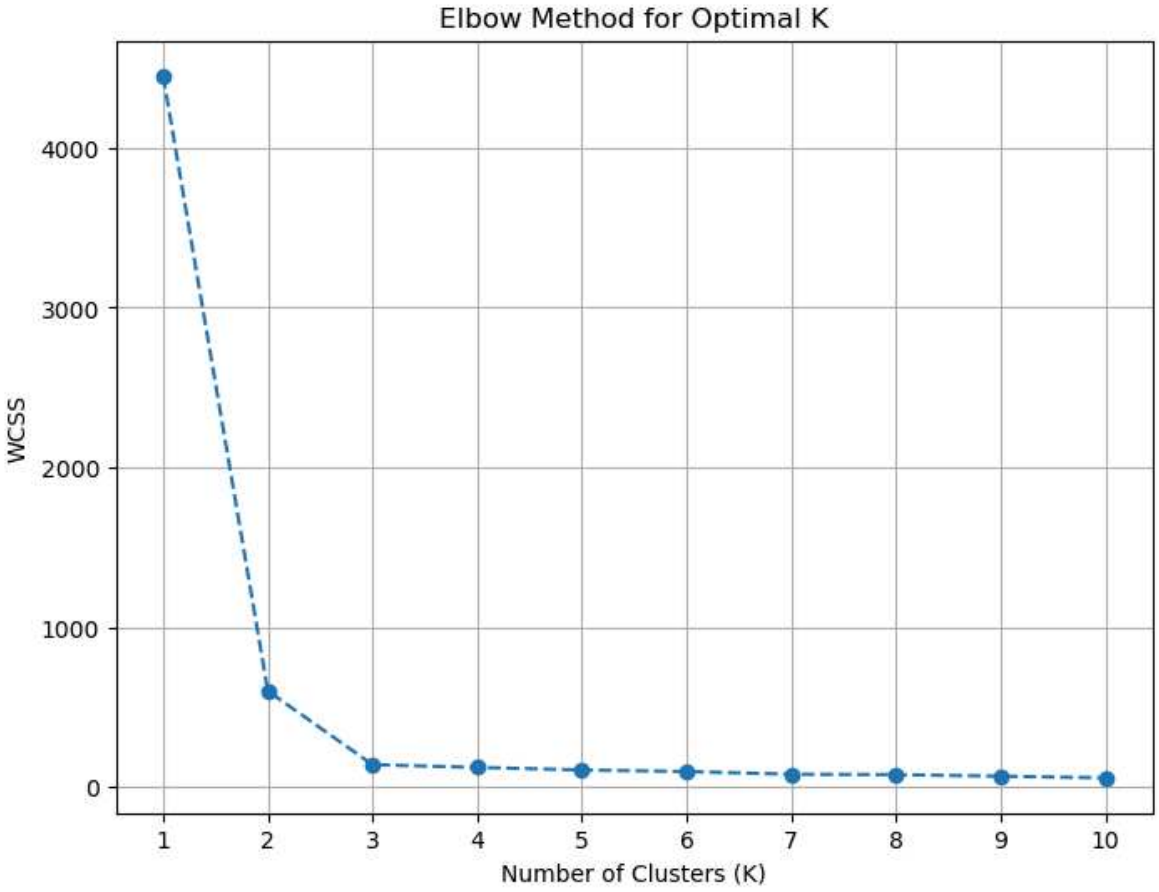
In [30]:
```
# 군집화 결과 시각화
plt.figure(figsize=(7, 7))
for cluster in np.unique(labels):
    plt.scatter(
        data[labels == cluster, 0],  # 군집에 속한 데이터의 x축 값
        data[labels == cluster, 1],  # 군집에 속한 데이터의 y축 값
        label=f'Cluster {cluster + 1}',  # 군집 번호
        alpha=0.6
    )
# 중심점 시각화
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label=
plt.title('K-Means Clustering Result')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid(True)
plt.show()
```



In [34]:
```
#엘보우 기법
# WCSS(Within-Cluster Sum of Squares) 저장 리스트
wcss = []

# 다양한 K 값에 대해 K-Means 모델 실행
for k in range(1, 11):  # K 값을 1부터 10까지 테스트
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data)  # 데이터 학습
    wcss.append(kmeans.inertia_)  # WCSS 값을 저장

# 엘보우 기법 시각화
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.xticks(range(1, 11))  # x축에 1~10의 값을 표시
plt.grid(True)
plt.show()
```


Elbow Method for Optimal K
WCSS
Number of Clusters (K)
0
1000
2000
3000
4000
1
2
3
4
5
6
7
8
9
10

In [ ]: