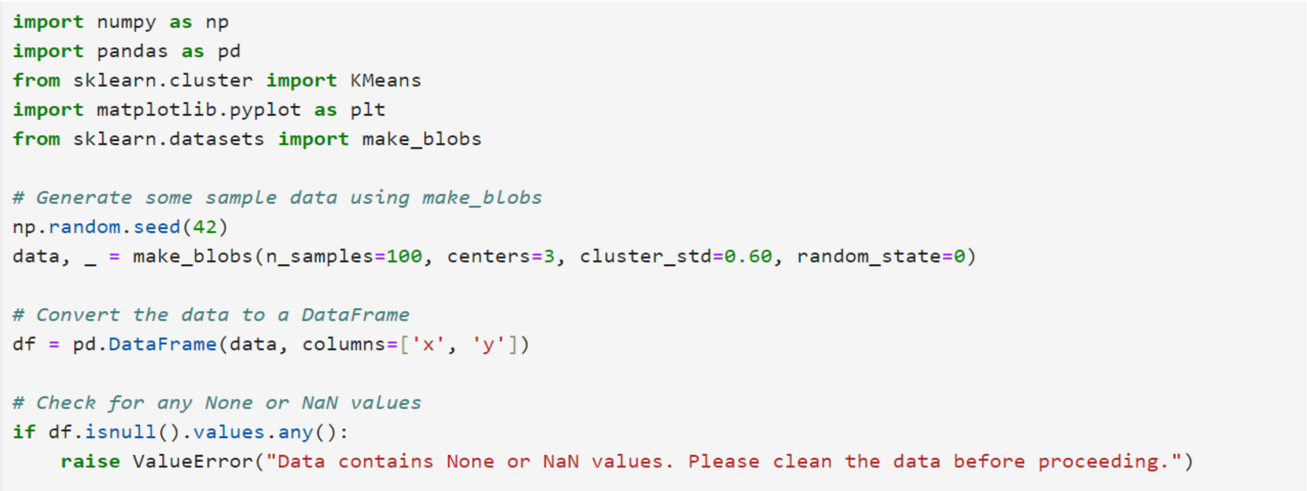
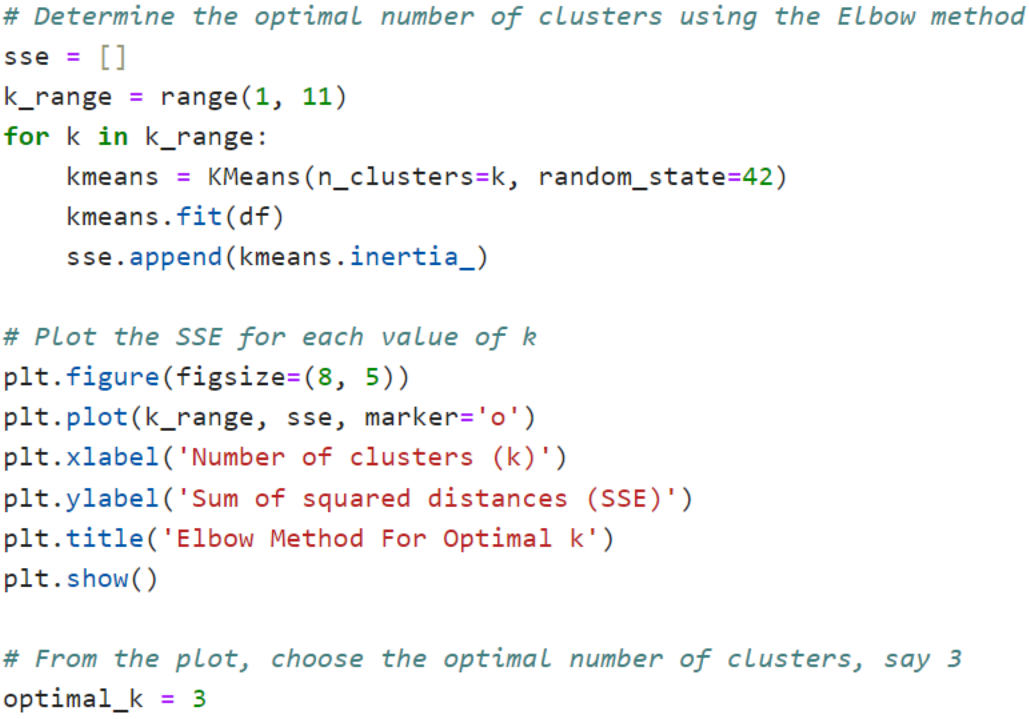
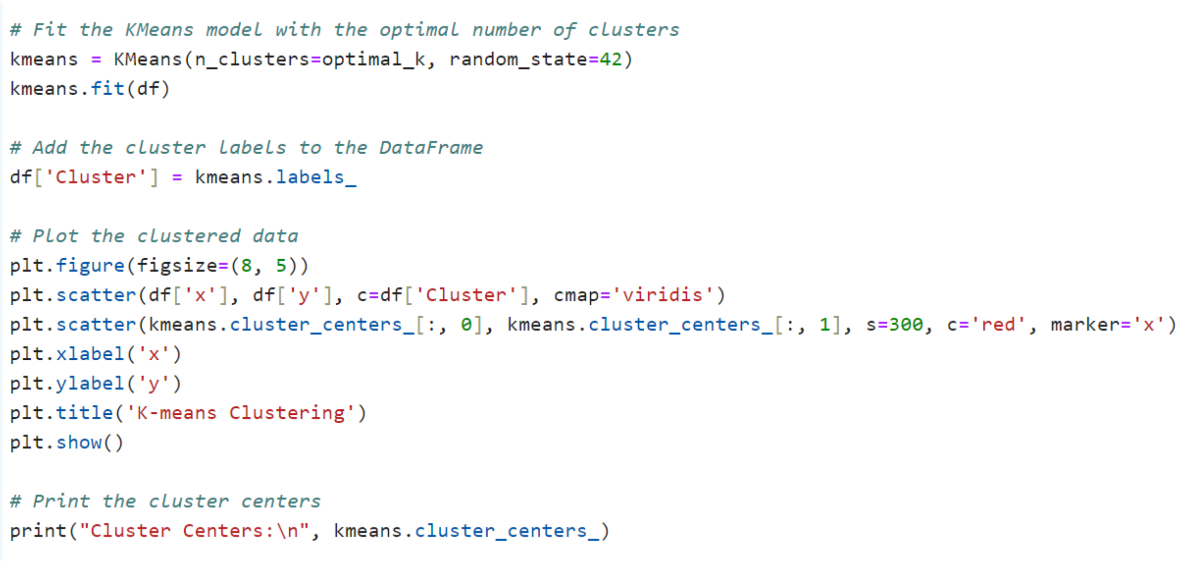


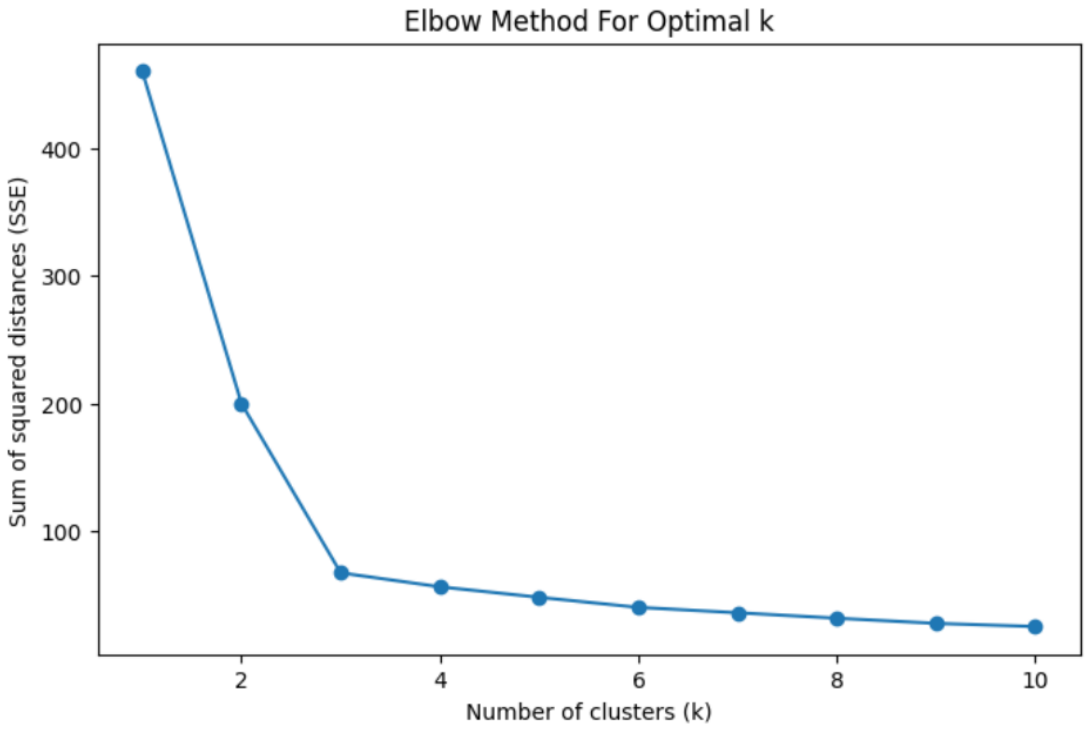
```python
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# Generate some sample data using make_blobs
np.random.seed(42)
data, _ = make_blobs(n_samples=100, centers=3, cluster_std=0.60, random_state=0)

# Convert the data to a DataFrame
df = pd.DataFrame(data, columns=['x', 'y'])

# Check for any None or NaN values
if df.isnull().values.any():
    raise ValueError("Data contains None or NaN values. Please clean the data before proceeding.")
```

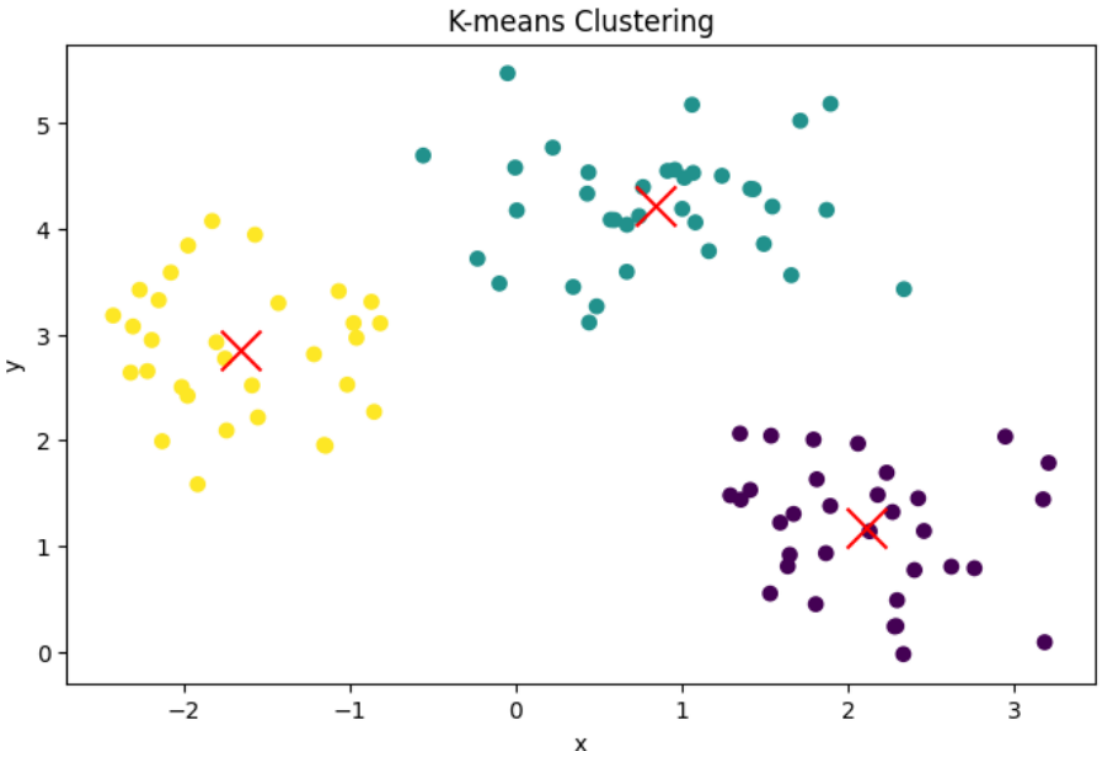


```python
# Determine the optimal number of clusters using the Elbow method
sse = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df)
    sse.append(kmeans.inertia_)

# Plot the SSE for each value of k
plt.figure(figsize=(8, 5))
plt.plot(k_range, sse, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Sum of squared distances (SSE)')
plt.title('Elbow Method For Optimal k')
plt.show()

# From the plot, choose the optimal number of clusters, say 3
optimal_k = 3
```



```python
# Fit the KMeans model with the optimal number of clusters
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(df)

# Add the cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Plot the clustered data
plt.figure(figsize=(8, 5))
plt.scatter(df['x'], df['y'], c=df['Cluster'], cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red', marker='x')
plt.xlabel('x')
plt.ylabel('y')
plt.title('K-means Clustering')
plt.show()

# Print the cluster centers
print("Cluster Centers:\n", kmeans.cluster_centers_)
```





```
Cluster Centers:
 [[ 2.10570255  1.17012424]
 [ 0.84321619  4.22240438]
 [-1.65213203  2.85383679]]
```