1. Imports y Configuración de Plots

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min

plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

`import pandas as pd` : Importa pandas, la librería para manipulación de datos
`import numpy as np` : Importa Numpy, para arrays y operaciones numericas
`import matplotlib.pyplot as plt` : Importa pyplot para graficar con matplotlib
`import seaborn as sb` : Importa seaborn para gráficos estadísticos
`from sklearn.cluster import KMeans` : Importa el algoritmo de KMeans de scikit-learn
`from sklearn.metrics import pairwise_distances_argmin_min` : Función que devuelve, para cada punto de un conjunto A, el índice del punto más cercano en otro conjunto B y la distancia; la usas después para encontrar el usuario más cercano a cada centroide
`plt.rcParams['figure.figize] = (16, 9)'` : Ajusta el tamaño de las figuras a 16 x 9 pulgadas
`plt.style.use('ggplot')` : Aplica el estilo de 'ggplot' a las figuras (cambia colores, grillas, etc.)

2. Carga del CSV y Primeras Impresiones

```
dataframe = pd.read_csv(r'analisis.csv')
print(dataframe.head())
print('\n---------------------------------------------------------------------
-\n')
print(dataframe.describe())
print('\n---------------------------------------------------------------------
-\n')
print(dataframe.groupby('categoria').size())
```

`pd.read_csv(r'analisi.csv')` : Lee el archivo CSV 'análisis.csv' y devuelve un DataFrame
`print(detaframe.head())` : Muestra las primeras 5 filas
`print('\n---...')` : Son separadores para hacer la salida más legible
`print(dataframe.describe())` : Muestra estadísticas descriptivas para las columnas

numéricas
`print(dataframe.groupby('categoria').size())` : Agrupa por la columna 'categoria' y cuenta cuántas filas hay en cada categoría

3. Visualización Rápida

```
dataframe.drop(['categoria'], axis = 1).hist()
plt.show()
```

`dataframe.drop(['categoria'], axis = 1)` : Crea un DataFrame sin la columna 'categoria' (axis = 1 indica columnas)
`.hist()` : Dibuja histogramas por cada columna numérica restante (cada columna una subgráfica)
`plt.show()` : fuerza la visualización de las figuras

4. Pairplot de Seaborn

```
sb.pairplot(dataframe.dropna(), hue = 'categoria', size = 4, vars = ['op',
'ex', 'ag'], kind = 'scatter')
```

`dataframe.dropna()` : Elimina filas con NA para que seaborn no falle por datos faltantes
`sb.pairplot(..., hue = 'categoria')` : Crea una matriz de scatterplots y distribuciones marginales coloreadas por 'categoria'. Es muy útil para ver relaciones entre pares de variables según la categoría
`vars = ['op', 'ex', 'ag']` : Limita la pairplot a esas tres columnas
`kind = 'scatter'` : Indica que las gráficas de pares serán scatter plots
`size = 4` : Parámetro histórico para controlar el tamaño de los subplots

5. Preparación de Datos para Clustering

```
X = np.array(dataframe[['op', 'ex', 'ag']])
y = np.array(dataframe['categoria'])
print('\n', X.shape)
```

`dataframe[['op', 'ex', 'ag']]` : Selecciona esas tres columnas en ese orden
`np.array(...)` : Transforma esa selección a un array de Numpy 'x' con forma '(n_samples, 3)'
`y = np.array(dataframe['categoria'])` : Guarda la columna de categoría en 'y'
`print(X.shape)` : Imprime la forma del array (140, 3), te confirma número de filas y columnas

6. Gráfico 3D de los Puntos Colorados según 'Categoria'

```
fig = plt.figure()
ax = fig.add_subplot(projection = '3d')
colores = ['white', 'red', 'green', 'blue', 'cyan', 'yellow', 'orange',
'black', 'pink', 'brown', 'purple']
# NOTA: Asignamos la posición cero del array repetida pues la cateogrías
comienzan en 1
asignar = []
for row in y:
    asignar.append(colores[row])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60)
```

`fig = plt.figure() / ax = fig.add_subplot(projection = '3d')` : Crea figura y eje 3D
`colores = [...]` : Lista de strings con nombres de color. Aquí definimos una paleta manual
`asignar = [] / for row in y: asignar.append(colores[row])` : Por cada valor de 'y' (categoría) buscas el color en la lista 'colores' usando el valor de categoría como índice

7. Bucle para Calcular "elbow" con Distancias K

```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters = i) for i in Nc]
print('\n----------------------------------------------------------------------
-\n')
print(kmeans)
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
print('\n----------------------------------------------------------------------
-\n')
print(score)

plt.figure()
plt.plot(Nc, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

`Nc = range(1, 20)` : Crea un iterador con K desde 1 hasta 19
`kmeans = [KMeans(n_clusters = i) for i in Nc]` : Crea una lista de objetos de KMeans con distintos números de clusters (Sin 'fit' todavía)
`score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]` : Para cada K:
`.fit(X)` : Ajusta Kmeans a 'X'
`.score(X)` : Devuelve el 'score' del modelo sobre 'X'
`plt.plot(Nc, score)` : Dibuja la curva (Number of clusters vs score)
`plt.xlabel / ylabel / title / plt.show()` : Etiquetas y muestra de la figura

8. Ajuste final de KMeans y Centroides

```
kmeans = KMeans(n_clusters = 5).fit(X)
centroids = kmeans.cluster_centers_
print('\n----------------------------------------------------------------------
-\n')
print(centroids)
```

`KMeans(n_clusters = 5).fit(X)` : Crea y ajusta KMeans con 5 clusters sobre 'X'
`centroids = kmeans.cluster_ceners_` : Array (5, 3) con las coordenadas de cada centroide
`print(centroids)` : Imprime los centros

9. Etiquetas, Coloreado por Cluster y Dibujo de Centroides

```
labels = kmeans.predict(X)
C = kmeans.cluster_centers_
colores = ['red', 'green', 'blue', 'cyan', 'yellow']
asignar = []
for row in labels:
    asignar.append(colores[row])

fig = plt.figure()
ax = fig.add_subplot(projection = '3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60)
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker = '*', c = colores, s = 1000)
```

`labels = kmeans.predict(X)` : Predice el cluster para cada punto 'X'
`C = kmeans.cluster_centers_` : Asigna a 'C' los centroides
`colores = ['red', 'green'...]` : Nueva paleta para clusters
`asignar = []` y `for` : Se generan lista de colores por etiqueta de cluster