```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
df = pd.read_csv("Supabase Snippet Country COVID-19 Summary with Demographics.csv")

print(df.shape)
df.head()
```

```
(100, 14)
```

	country_id	location	continent	gdp_per_capita	median_age	human_development_index	population_density	life_
0	1	Afghanistan	Asia	1803.99	18.6	0.51	54.42	
1	2	Albania	Europe	11803.43	38.0	0.80	104.87	
2	3	Algeria	Africa	13913.84	29.1	0.75	17.35	
3	4	American Samoa	Oceania	NaN	NaN	NaN	278.20	
4	5	Andorra	Europe	NaN	NaN	0.87	163.76	

Next steps: Generate code with df | New interactive sheet

```
# Drop rows where core pandemic info missing
df = df.dropna(subset=["total_cases", "total_deaths"])

# Reset index
df = df.reset_index(drop=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98 entries, 0 to 97
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   country_id              98 non-null     int64
 1   location                98 non-null     object
 2   continent               98 non-null     object
 3   gdp_per_capita          81 non-null     float64
 4   median_age              84 non-null     float64
 5   human_development_index  80 non-null     float64
 6   population_density      91 non-null     float64
 7   life_expectancy         97 non-null     float64
 8   avg_reproduction        80 non-null     float64
 9   avg_positive_rate       79 non-null     float64
 10  avg_stringency          79 non-null     float64
 11  total_cases             98 non-null     float64
 12  total_deaths            98 non-null     float64
 13  avg_vaccinations        78 non-null     float64
dtypes: float64(11), int64(1), object(2)
memory usage: 10.8+ KB
```

```
# Death rate
df["death_rate"] = df["total_deaths"] / df["total_cases"]

# Replace infinite values if any
df.replace([np.inf, -np.inf], np.nan, inplace=True)
```

```
features = [
    "death_rate",
    "avg_reproduction",
    "avg_positive_rate",
    "avg_vaccinations",
    "gdp_per_capita",
    "median_age",
    "human_development_index",
    "life_expectancy",
    "avg_stringency"
]

X = df[features]
```

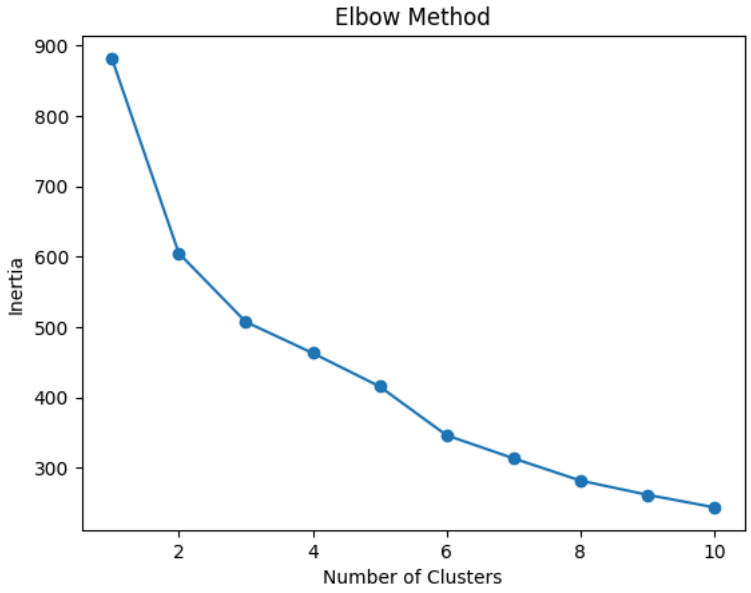
```
X = X.fillna(X.median())
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
inertia = []

for k in range(1, 11):
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(X_scaled)
    inertia.append(model.inertia_)

plt.figure()
plt.plot(range(1,11), inertia, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.title("Elbow Method")
plt.show()
```




```
kmeans = KMeans(n_clusters=4, random_state=42)

df["cluster"] = kmeans.fit_predict(X_scaled)

df.head()
```

	country_id	location	continent	gdp_per_capita	median_age	human_development_index	population_density	life_
0	1	Afghanistan	Asia	1803.99	18.6	0.51	54.42	
1	2	Albania	Europe	11803.43	38.0	0.80	104.87	
2	3	Algeria	Africa	13913.84	29.1	0.75	17.35	
3	4	American Samoa	Oceania	NaN	NaN	NaN	278.20	
4	5	Andorra	Europe	NaN	NaN	0.87	163.76	

Next steps: Generate code with df | New interactive sheet

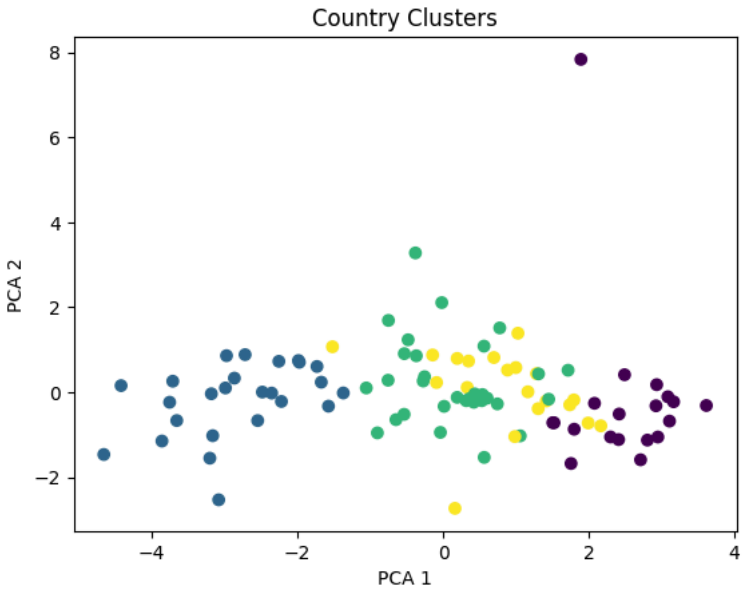
```
cluster_summary = df.groupby("cluster")[features].mean()
cluster_summary
```

cluster	death_rate	avg_reproduction	avg_positive_rate	avg_vaccinations	gdp_per_capita	median_age	human_deve
0	0.004636	1.024418	0.081677	366304.262811	41072.586316	40.576471	
1	0.015623	0.736723	0.046618	22354.638531	3735.170800	19.964000	
2	0.008546	0.888072	0.088914	154408.589525	11796.726000	30.734783	
3	0.019116	0.986470	0.186131	64139.242270	15362.565294	34.547368	

Next steps: Generate code with `cluster_summary` | New interactive sheet

```
pca = PCA(n_components=2)
reduced = pca.fit_transform(X_scaled)

plt.figure()
plt.scatter(reduced[:,0], reduced[:,1], c=df["cluster"])
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.title("Country Clusters")
plt.show()
```




```
for i in sorted(df["cluster"].unique()):
    print(f"\nCluster {i}")
    print(df[df["cluster"] == i]["location"].values)
```

```
Cluster 0
['Aruba' 'Australia' 'Austria' 'Bahrain' 'Belgium' 'Bermuda' 'Brunei'
 'Canada' 'Cayman Islands' 'China' 'Cyprus' 'Czechia' 'Denmark' 'Estonia'
 'Finland' 'France' 'Germany' 'Greece' 'Iceland']

Cluster 1
['Afghanistan' 'Angola' 'Benin' 'Burkina Faso' 'Burundi' 'Cambodia'
 'Cameroon' 'Central African Republic' 'Chad' 'Comoros' 'Congo'
 "Cote d'Ivoire" 'Democratic Republic of Congo' 'Djibouti' 'East Timor'
 'El Salvador' 'Equatorial Guinea' 'Eritrea' 'Eswatini' 'Ethiopia'
 'Gambia' 'Ghana' 'Guinea' 'Guinea-Bissau' 'Haiti']

Cluster 2
['American Samoa' 'Andorra' 'Anguilla' 'Antigua and Barbuda' 'Azerbaijan'
 'Bangladesh' 'Belarus' 'Belize' 'Bhutan'
 'Bonaire Sint Eustatius and Saba' 'Botswana' 'Brazil'
 'British Virgin Islands' 'Cape Verde' 'Chile' 'Cook Islands' 'Cuba'
 'Dominica' 'Dominican Republic' 'Falkland Islands' 'Fiji' 'French Guiana'
 'French Polynesia' 'Gabon' 'Georgia' 'Gibraltar' 'Greenland' 'Grenada'
 'Guadeloupe' 'Guernsey' 'Guyana' 'Honduras' 'India' 'Indonesia']

Cluster 3
['Albania' 'Algeria' 'Argentina' 'Armenia' 'Bahamas' 'Barbados' 'Bolivia'
 'Bosnia and Herzegovina' 'Bulgaria' 'Colombia' 'Costa Rica' 'Croatia'
 'Curacao' 'Ecuador' 'Egypt' 'Faroe Islands' 'Guam' 'Guatemala' 'Hungary'
 'Iran']
```

```
df.to_csv("covid_clustered_output.csv", index=False)
```