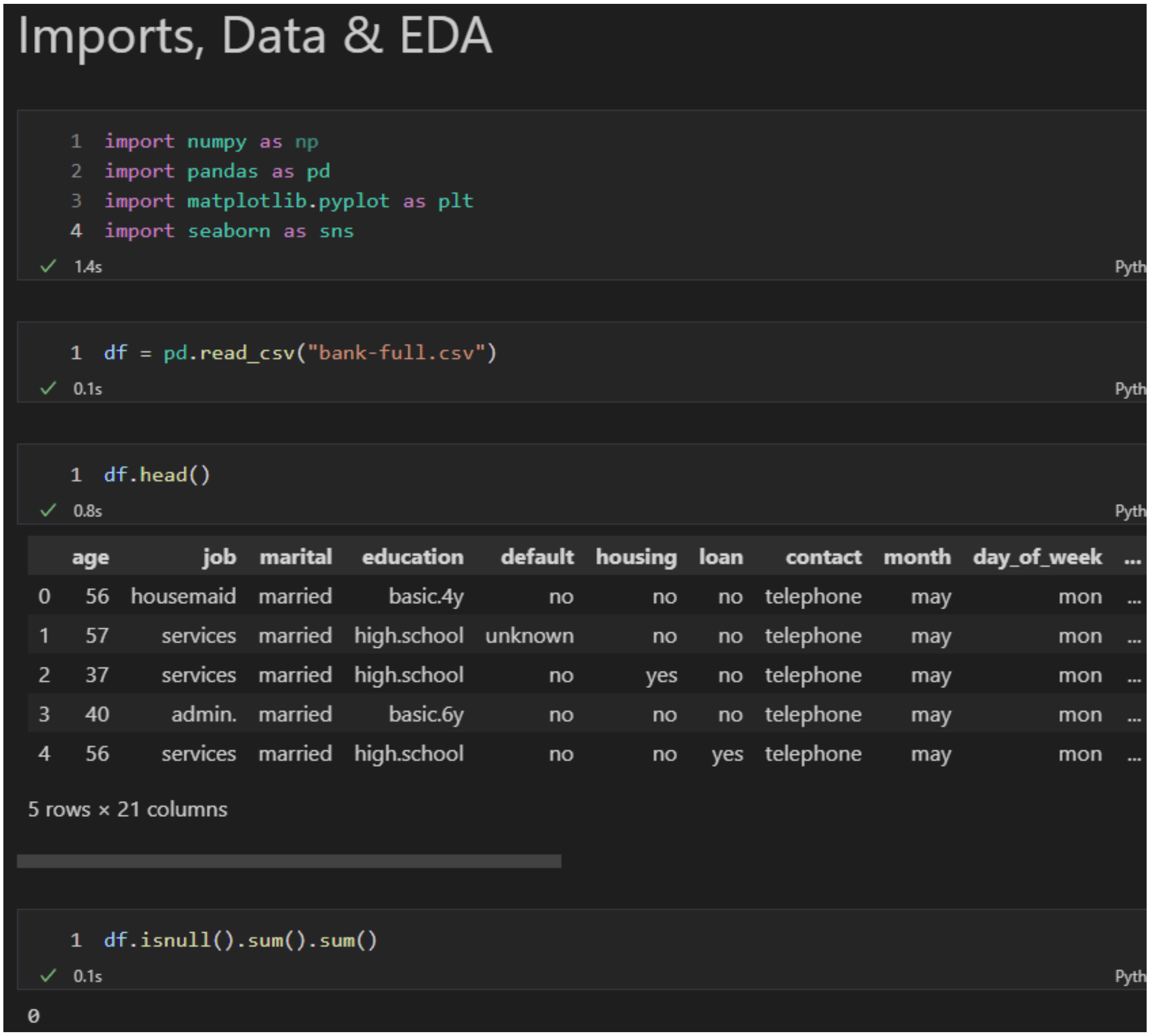
K-MEANS CLUSTERING

▼ Bank Marketing



Imports, Data & EDA

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

✓ 1.4s

```
df = pd.read_csv("bank-full.csv")
```

✓ 0.1s

```
df.head()
```

✓ 0.8s

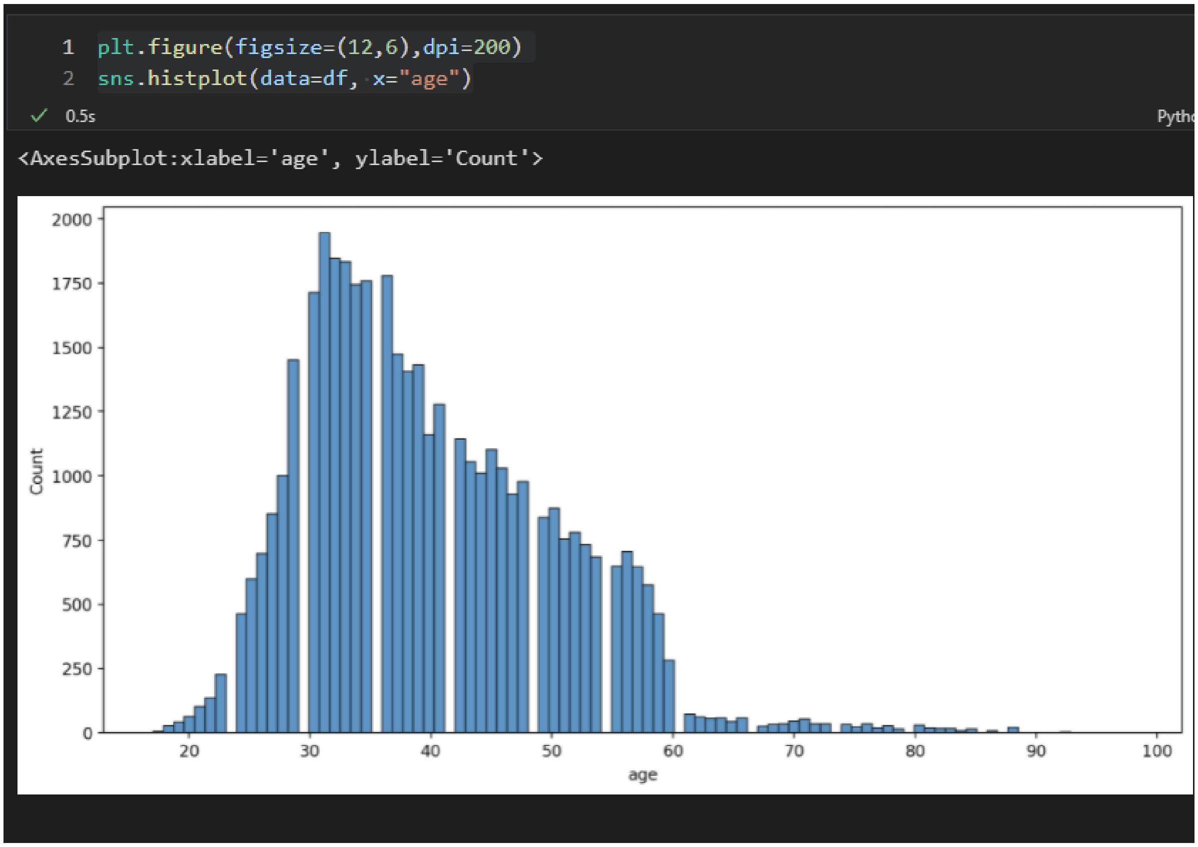
	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...

5 rows × 21 columns

```
df.isnull().sum().sum()
```

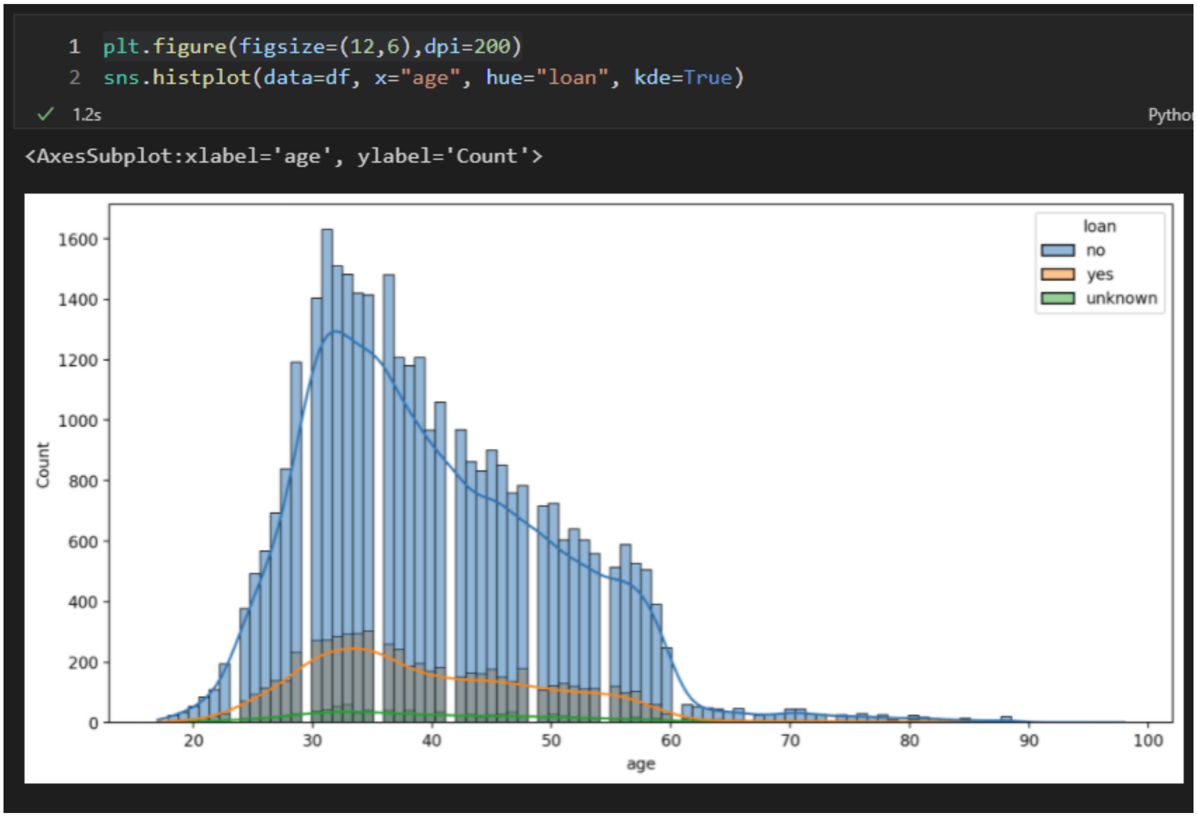
✓ 0.1s

```
0
```



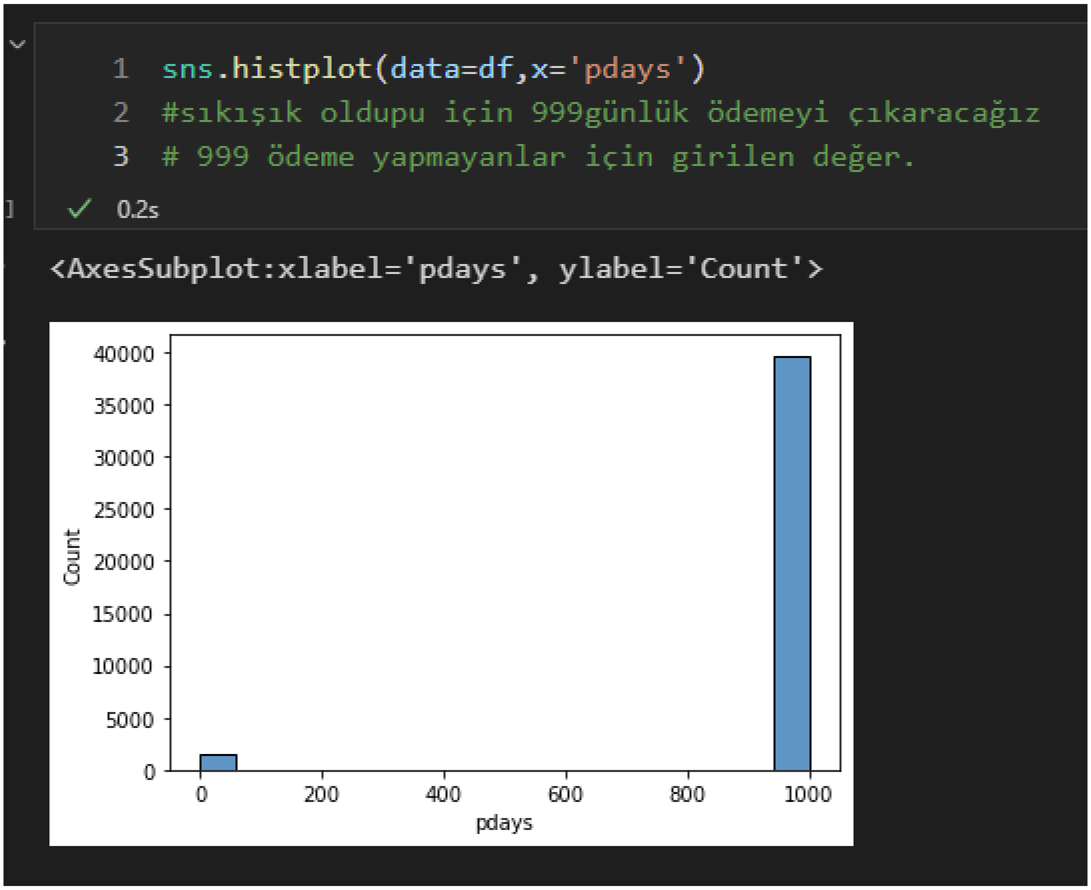
```python
plt.figure(figsize=(12,6),dpi=200)
sns.histplot(data=df, x="age")
```

```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



```python
plt.figure(figsize=(12,6),dpi=200)
sns.histplot(data=df, x="age", hue="loan", kde=True)
```

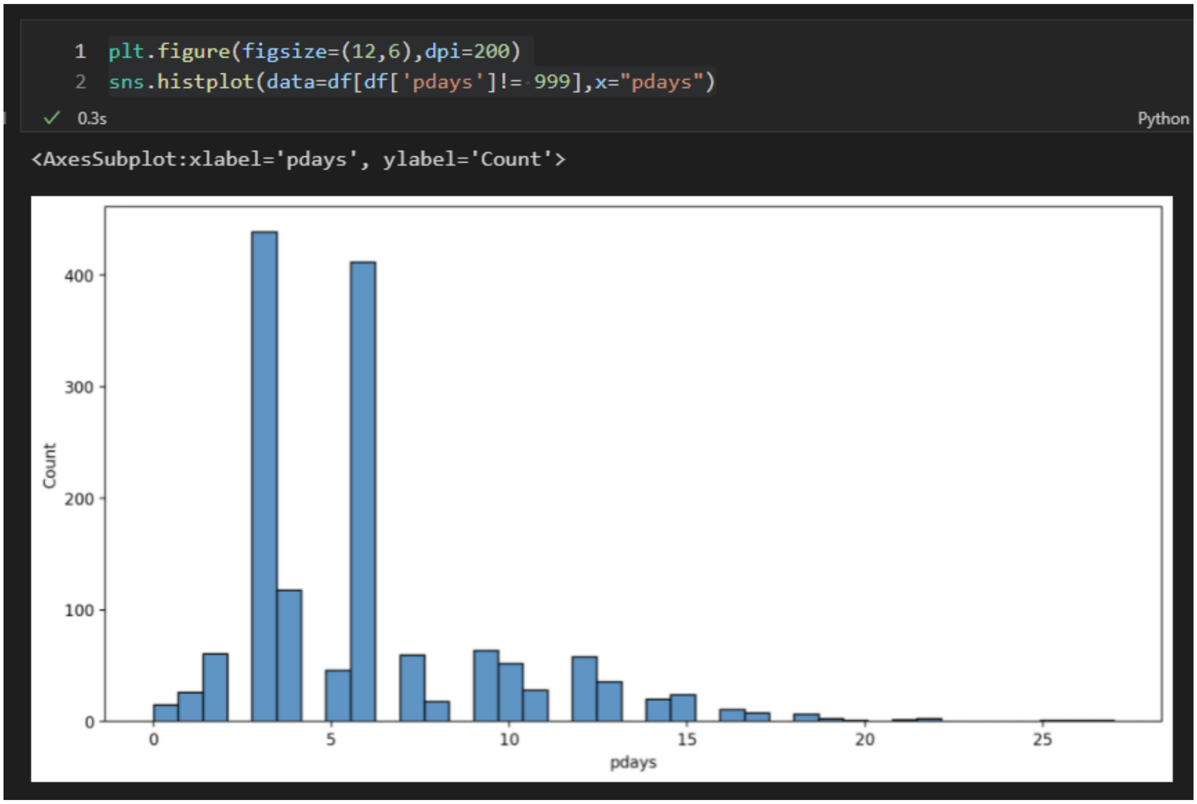
```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



```
sns.histplot(data=df,x='pdays')
#sıkışık oldupu için 999günlük ödemeyi çıkaracağız
# 999 ödeme yapmayanlar için girilen değer.
```

✓ 0.2s

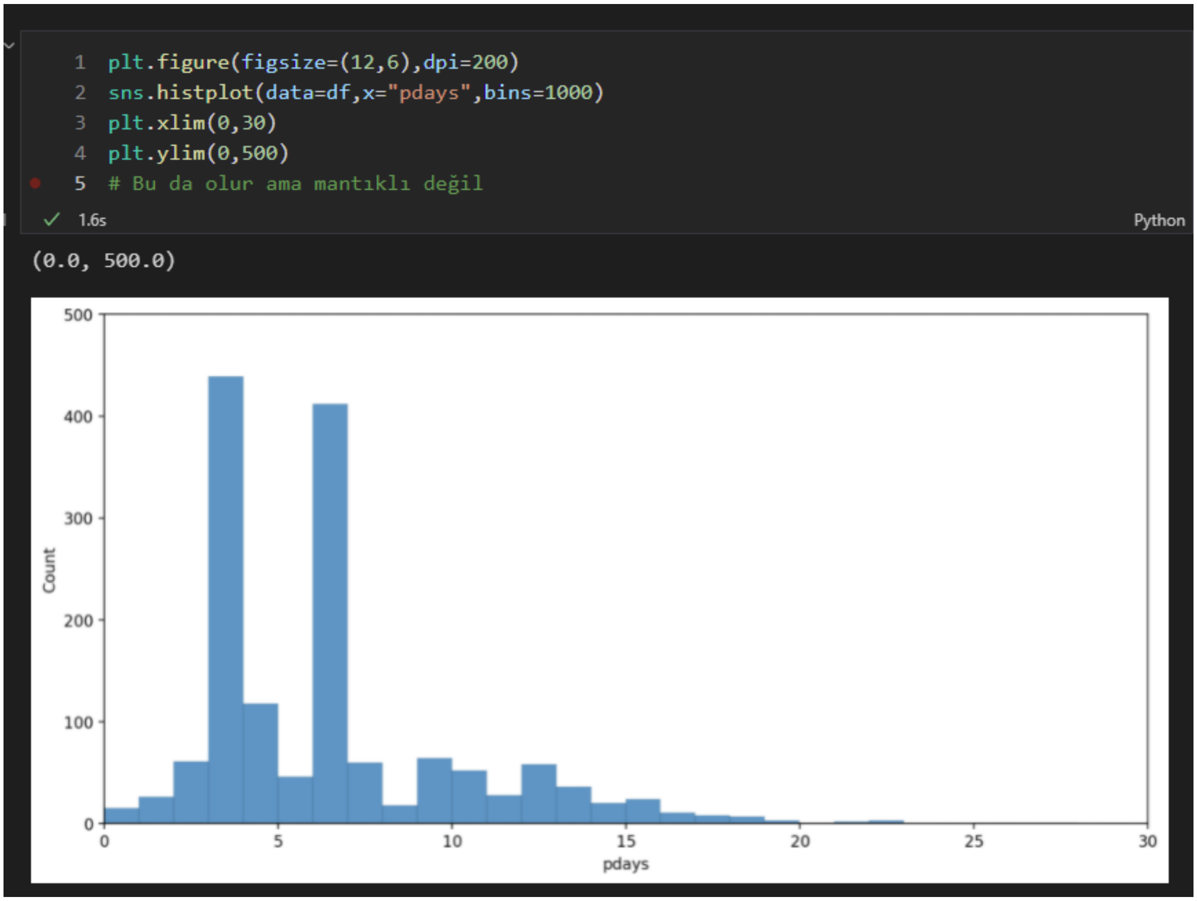
```
<AxesSubplot:xlabel='pdays', ylabel='Count'>
```



```
plt.figure(figsize=(12,6),dpi=200)
sns.histplot(data=df[df['pdays']!= 999],x="pdays")
```

✓ 0.3s

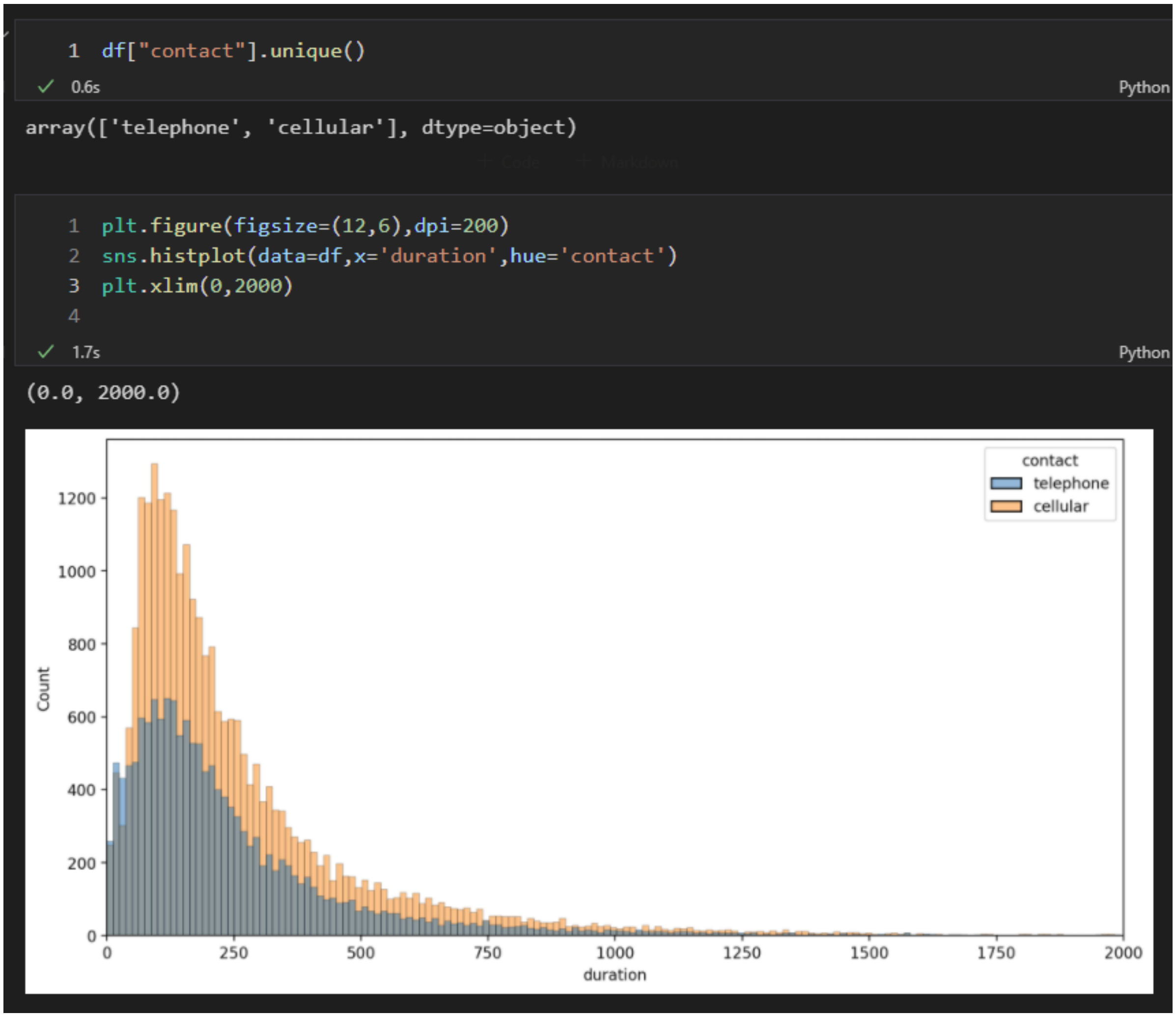
```
<AxesSubplot:xlabel='pdays', ylabel='Count'>
```



```python
plt.figure(figsize=(12,6),dpi=200)
sns.histplot(data=df,x="pdays",bins=1000)
plt.xlim(0,30)
plt.ylim(0,500)
# Bu da olur ama mantıklı değil
```

✓ 1.6s

```
(0.0, 500.0)
```



```
df["contact"].unique()
```

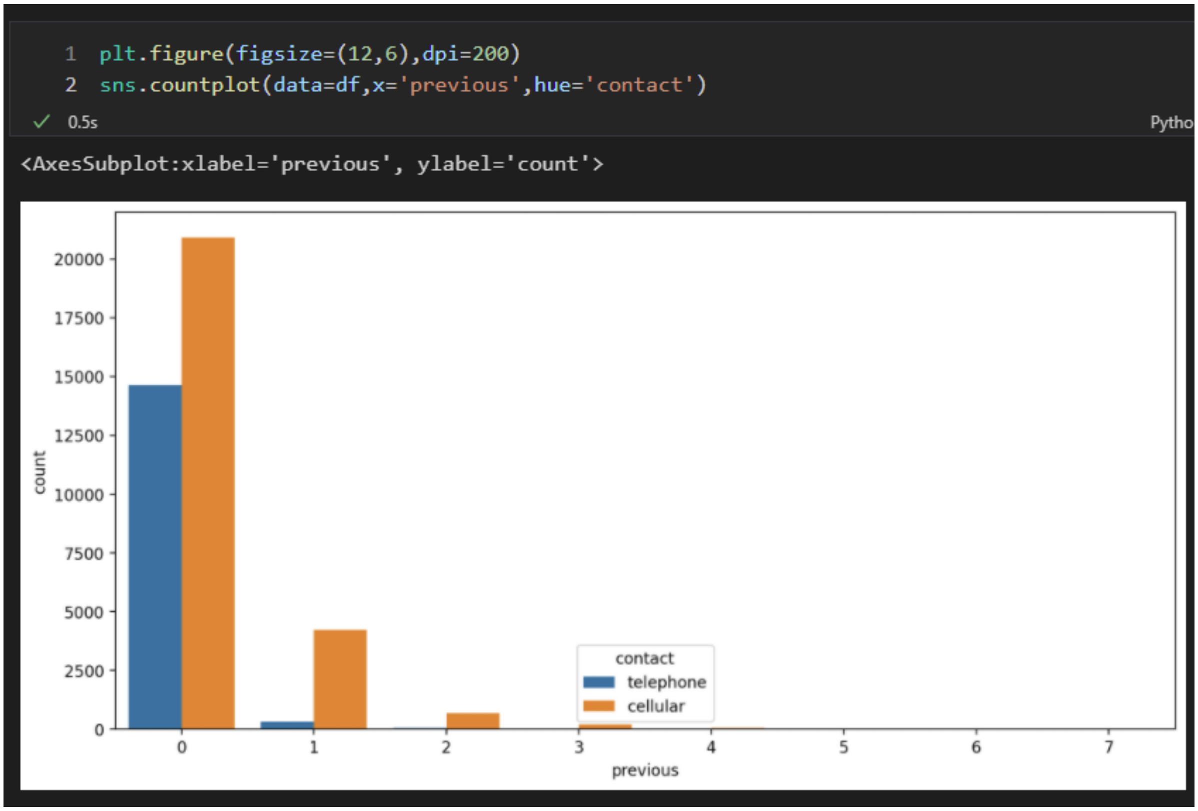
✓ 0.6s Python

```
array(['telephone', 'cellular'], dtype=object)
```

```
plt.figure(figsize=(12,6),dpi=200)
sns.histplot(data=df,x='duration',hue='contact')
plt.xlim(0,2000)

```

✓ 1.7s Python

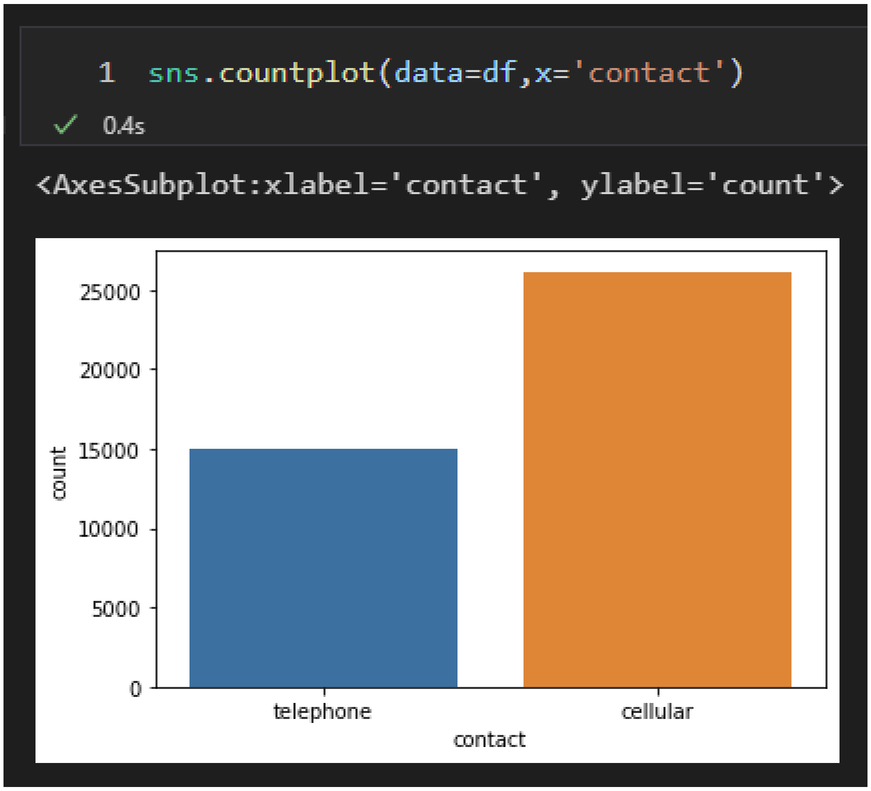
```
(0.0, 2000.0)
```



```
plt.figure(figsize=(12,6),dpi=200)
sns.countplot(data=df,x='previous',hue='contact')
```

✓ 0.5s

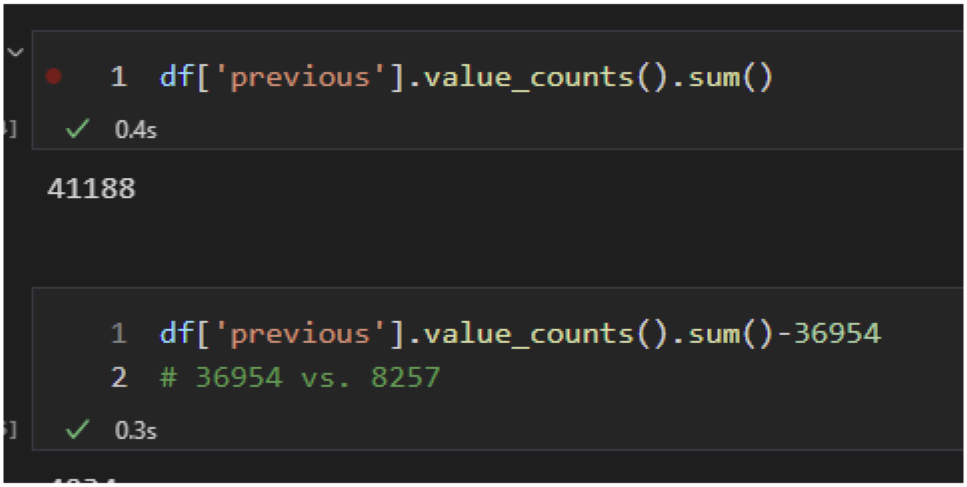
```
<AxesSubplot:xlabel='previous', ylabel='count'>
```



```
sns.countplot(data=df,x='contact')
```

✓ 0.4s

```
<AxesSubplot:xlabel='contact', ylabel='count'>
```



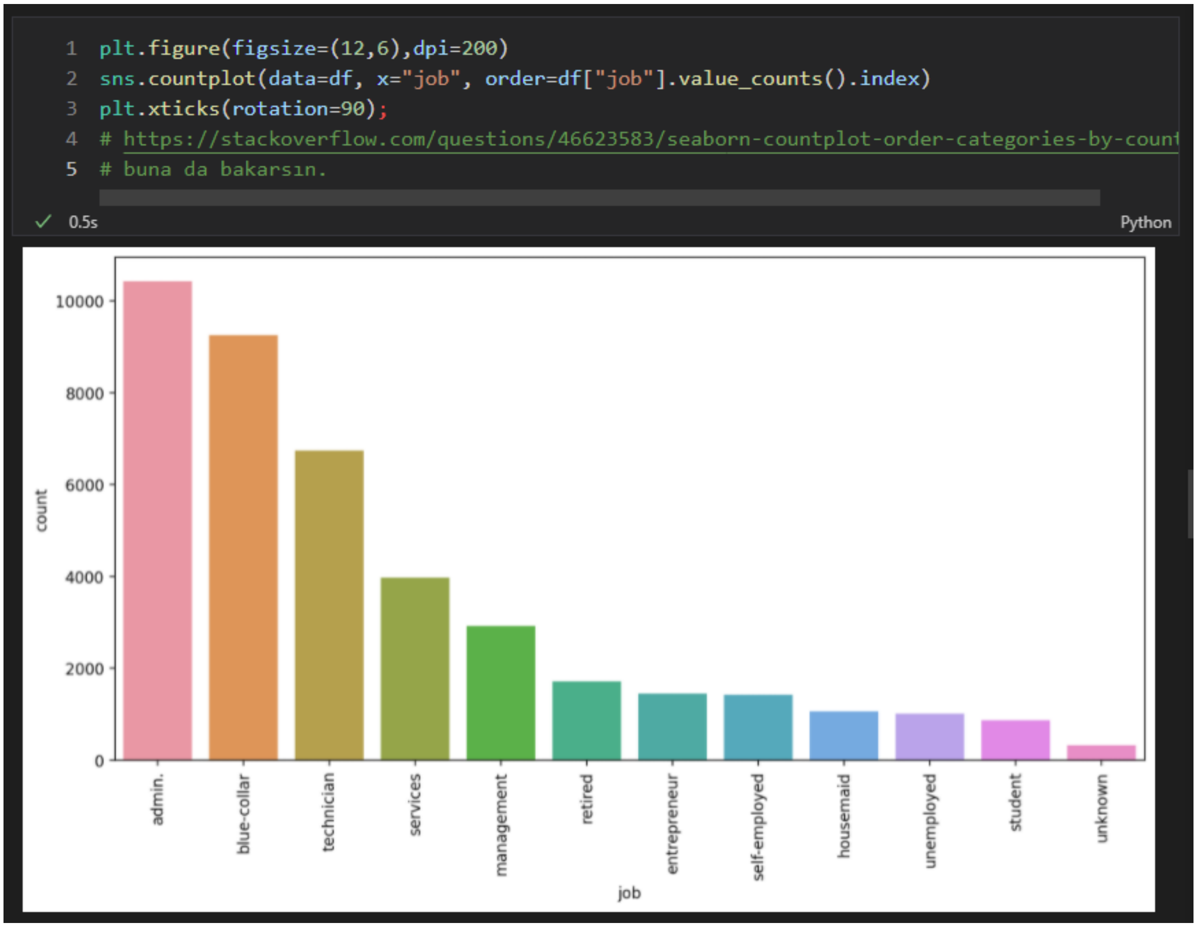
```
df['previous'].value_counts().sum()
```

✓ 0.4s

```
41188
```

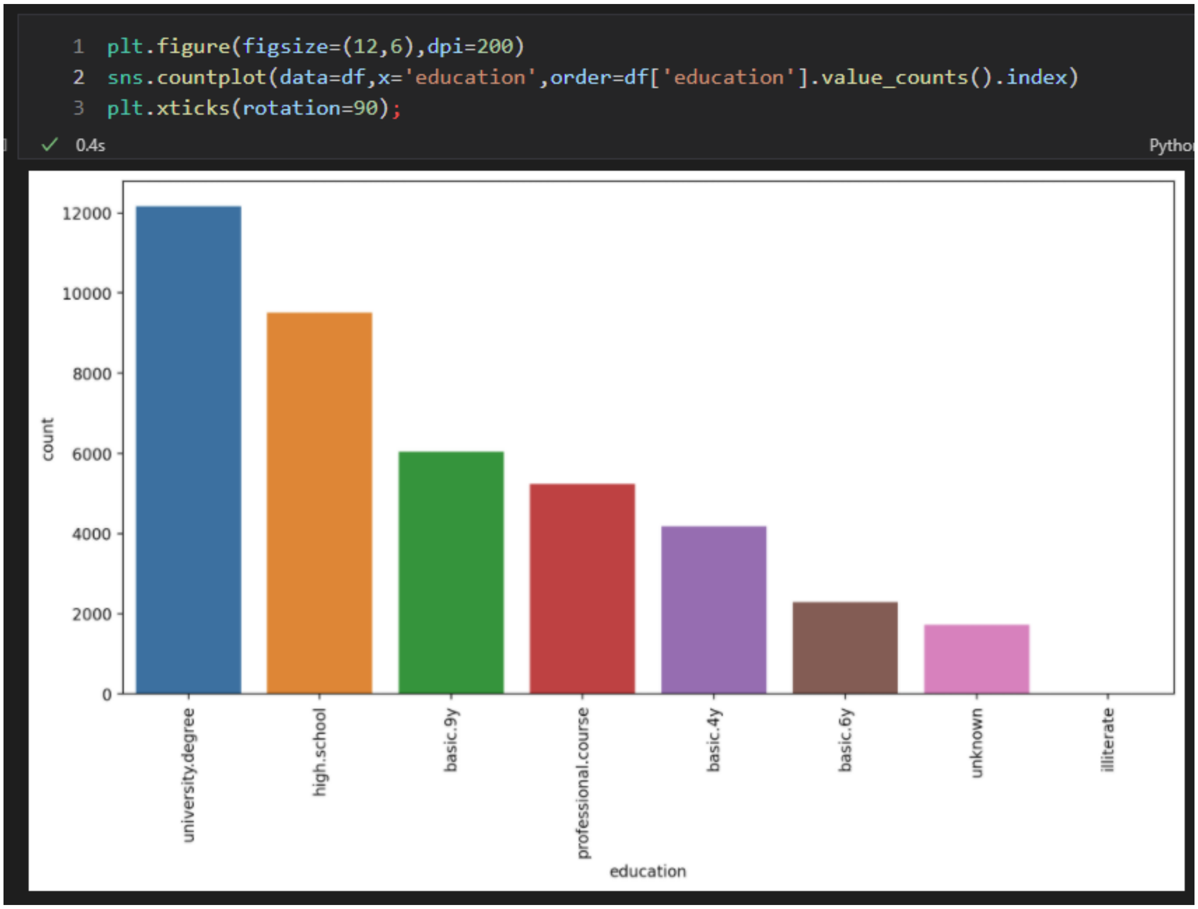
```
df['previous'].value_counts().sum()-36954
# 36954 vs. 8257
```

✓ 0.3s

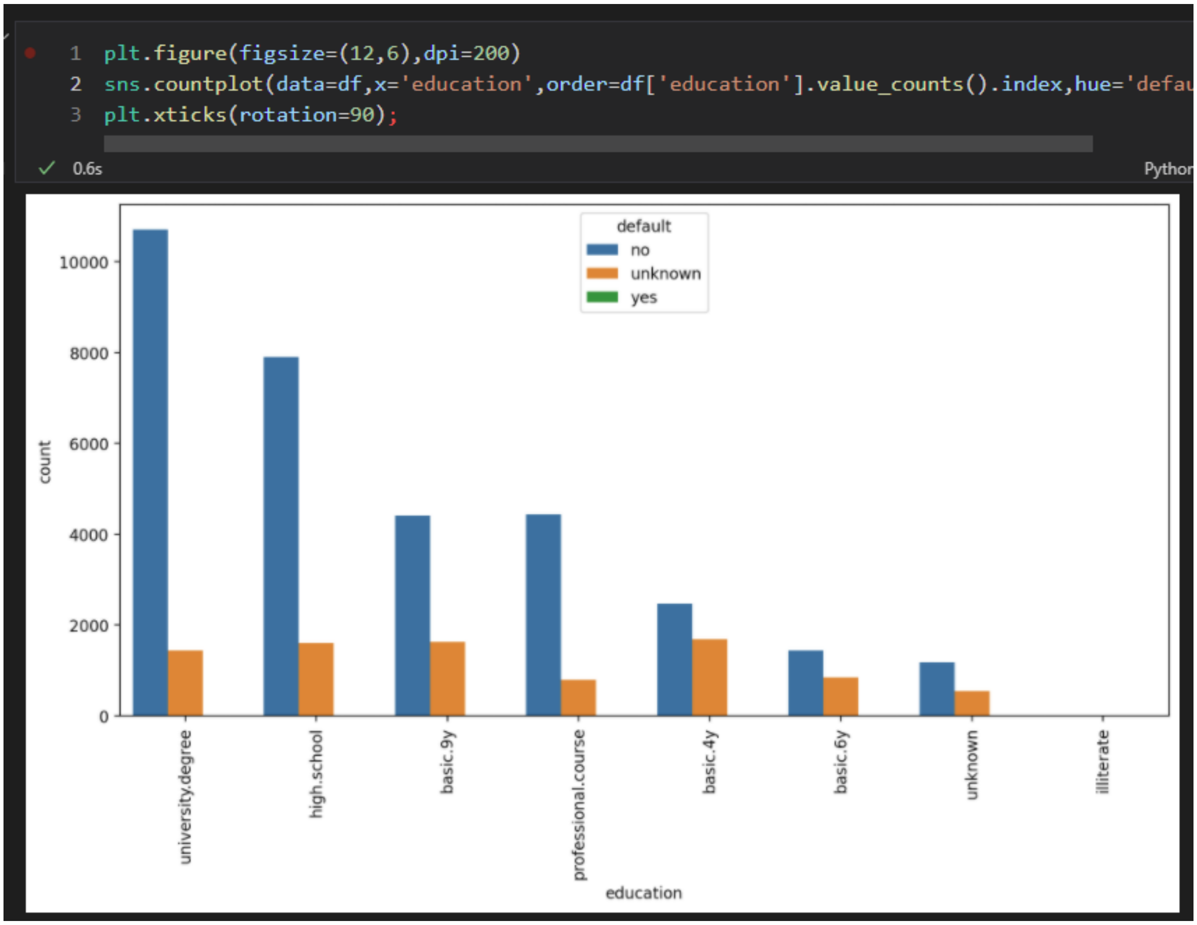


```
plt.figure(figsize=(12,6),dpi=200)
sns.countplot(data=df, x="job", order=df["job"].value_counts().index)
plt.xticks(rotation=90);
# https://stackoverflow.com/questions/46623583/seaborn-countplot-order-categories-by-count
# buna da bakarsın.
```

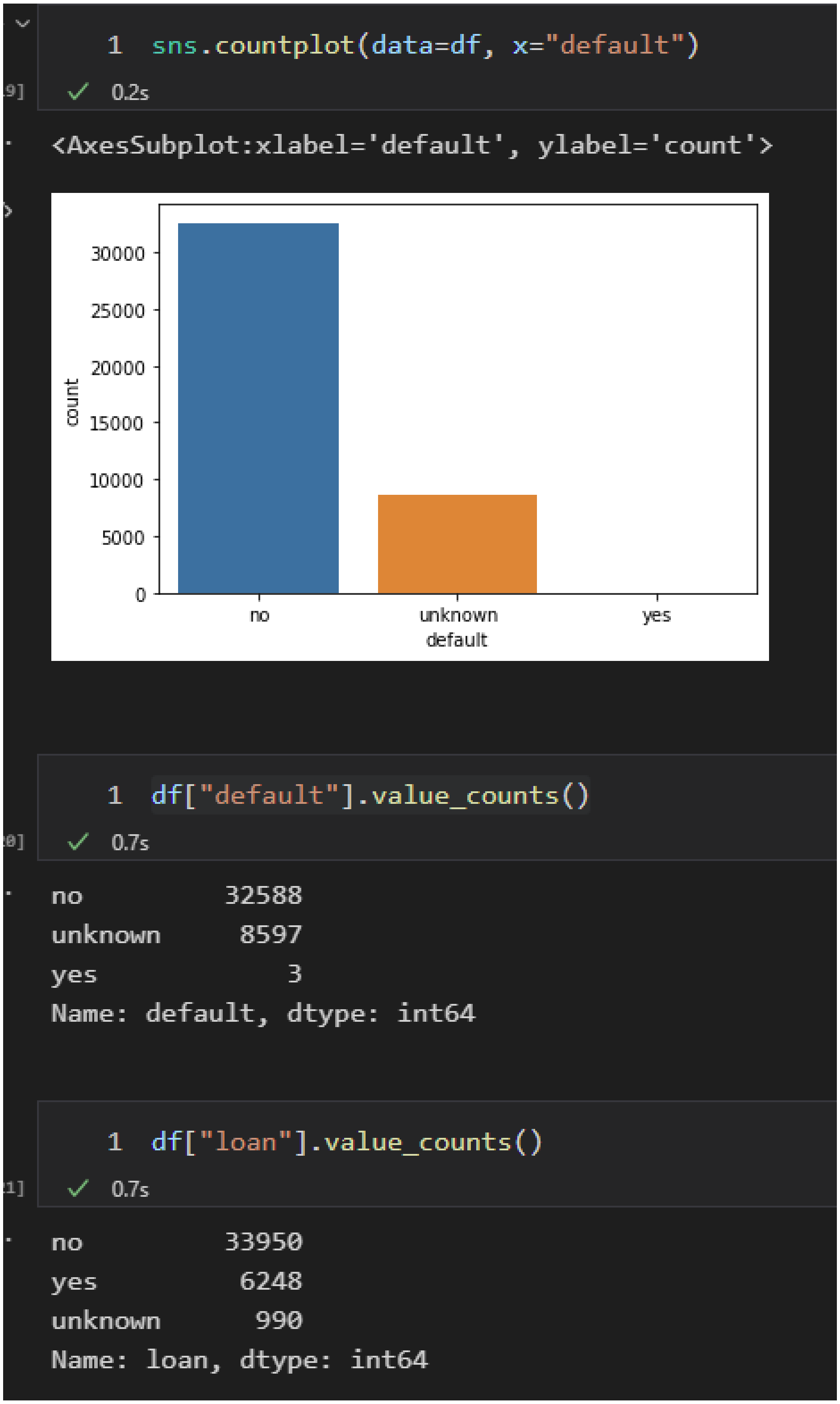
✓ 0.5s Python



```
plt.figure(figsize=(12,6),dpi=200)
sns.countplot(data=df,x='education',order=df['education'].value_counts().index)
plt.xticks(rotation=90);
```



```
plt.figure(figsize=(12,6),dpi=200)
sns.countplot(data=df,x='education',order=df['education'].value_counts().index,hue='defau
plt.xticks(rotation=90);
```



```
sns.countplot(data=df, x="default")
```

✓ 0.2s

```
<AxesSubplot:xlabel='default', ylabel='count'>
```

```
df["default"].value_counts()
```

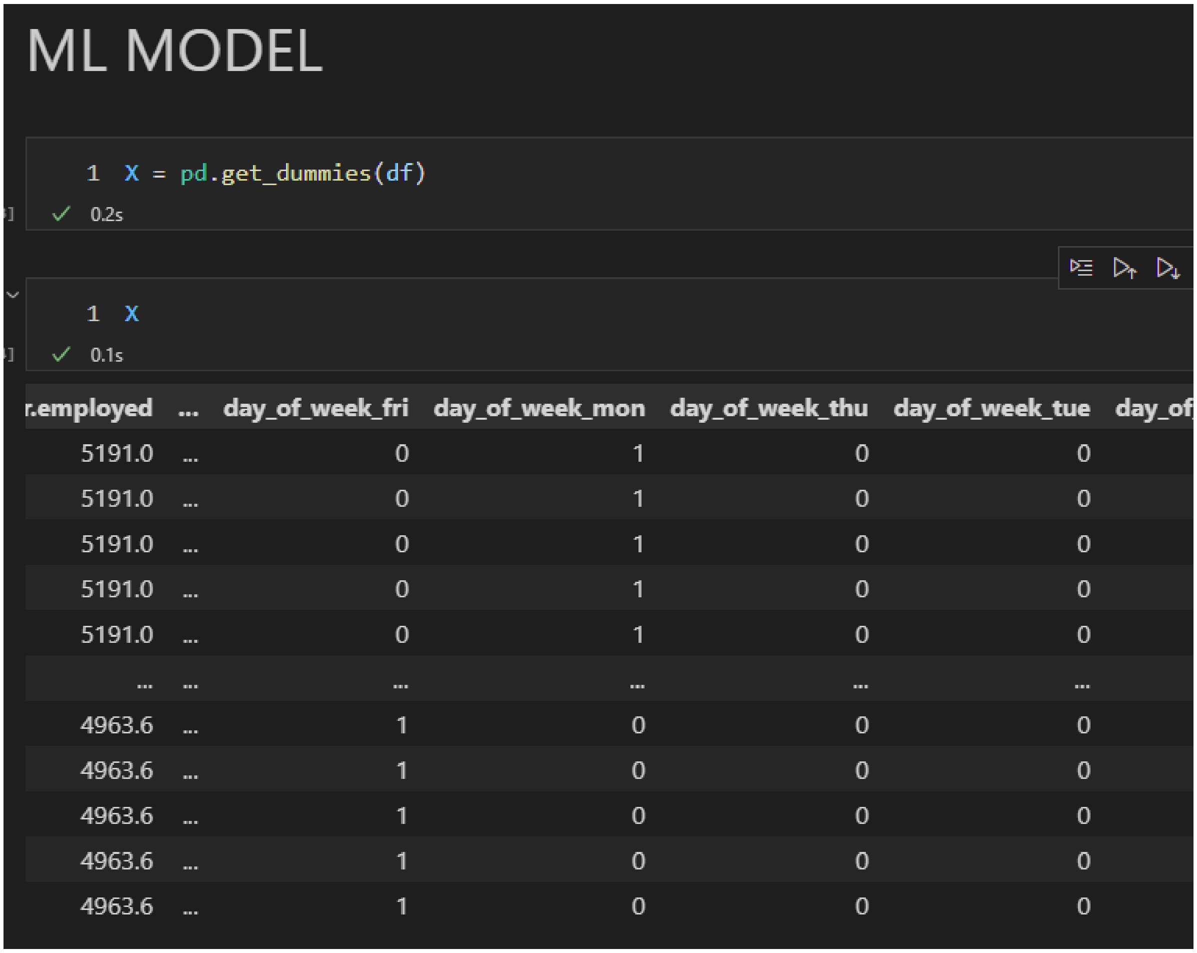
✓ 0.7s

```
no         32588
unknown     8597
yes            3
Name: default, dtype: int64
```

```
df["loan"].value_counts()
```

✓ 0.7s

```
no         33950
yes         6248
unknown      990
Name: loan, dtype: int64
```



ML MODEL

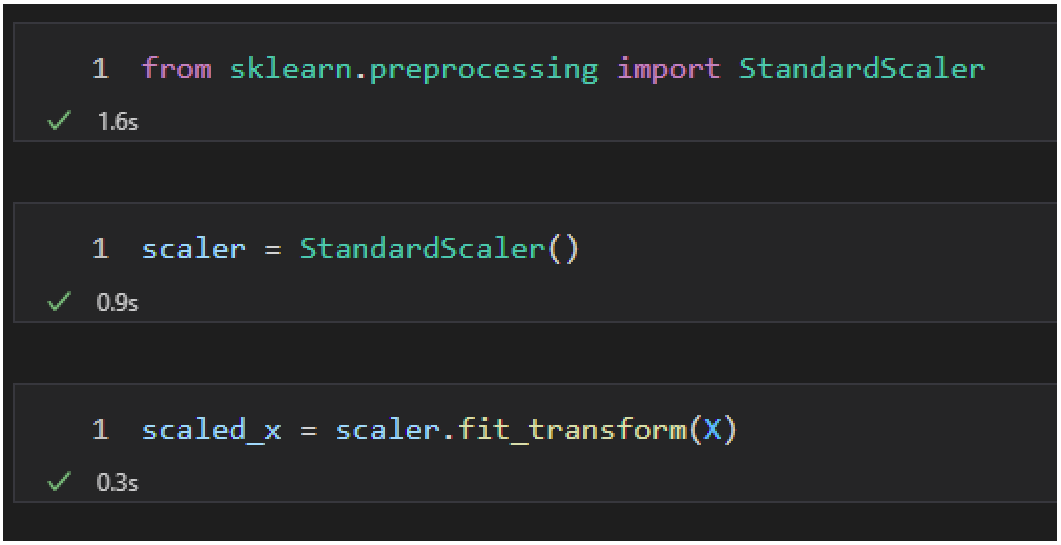
```python
X = pd.get_dummies(df)
```

✓ 0.2s

```python
X
```

✓ 0.1s

```
r.employed  ...  day_of_week_fri  day_of_week_mon  day_of_week_thu  day_of_week_tue  day_of
    5191.0  ...                0                1                0                0
    5191.0  ...                0                1                0                0
    5191.0  ...                0                1                0                0
    5191.0  ...                0                1                0                0
    5191.0  ...                0                1                0                0
       ...  ...              ...              ...              ...              ...
    4963.6  ...                1                0                0                0
    4963.6  ...                1                0                0                0
    4963.6  ...                1                0                0                0
    4963.6  ...                1                0                0                0
    4963.6  ...                1                0                0                0
```



```python
from sklearn.preprocessing import StandardScaler
```

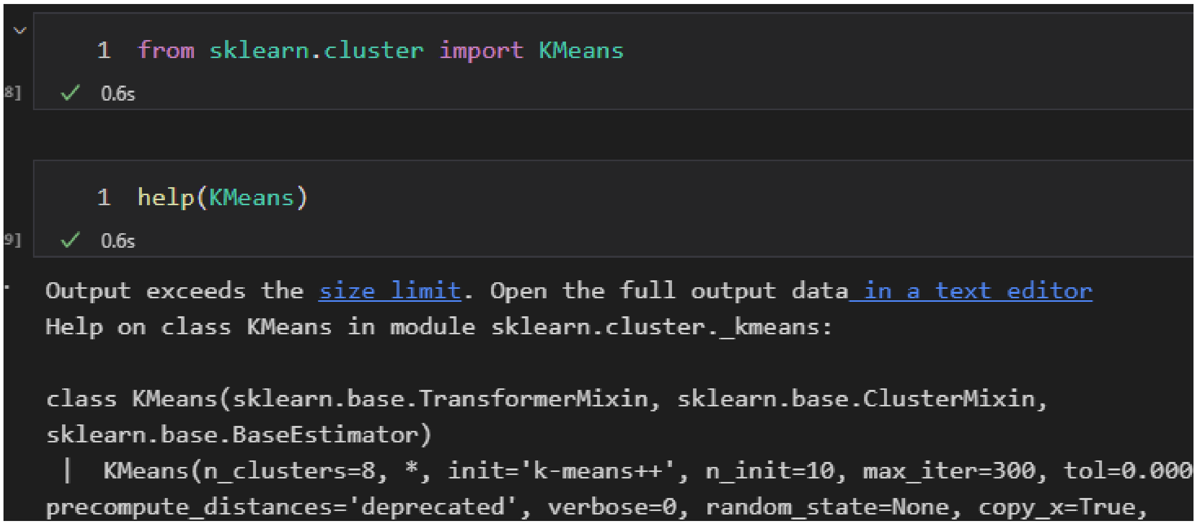
✓ 1.6s

```python
scaler = StandardScaler()
```

✓ 0.9s

```python
scaled_x = scaler.fit_transform(X)
```

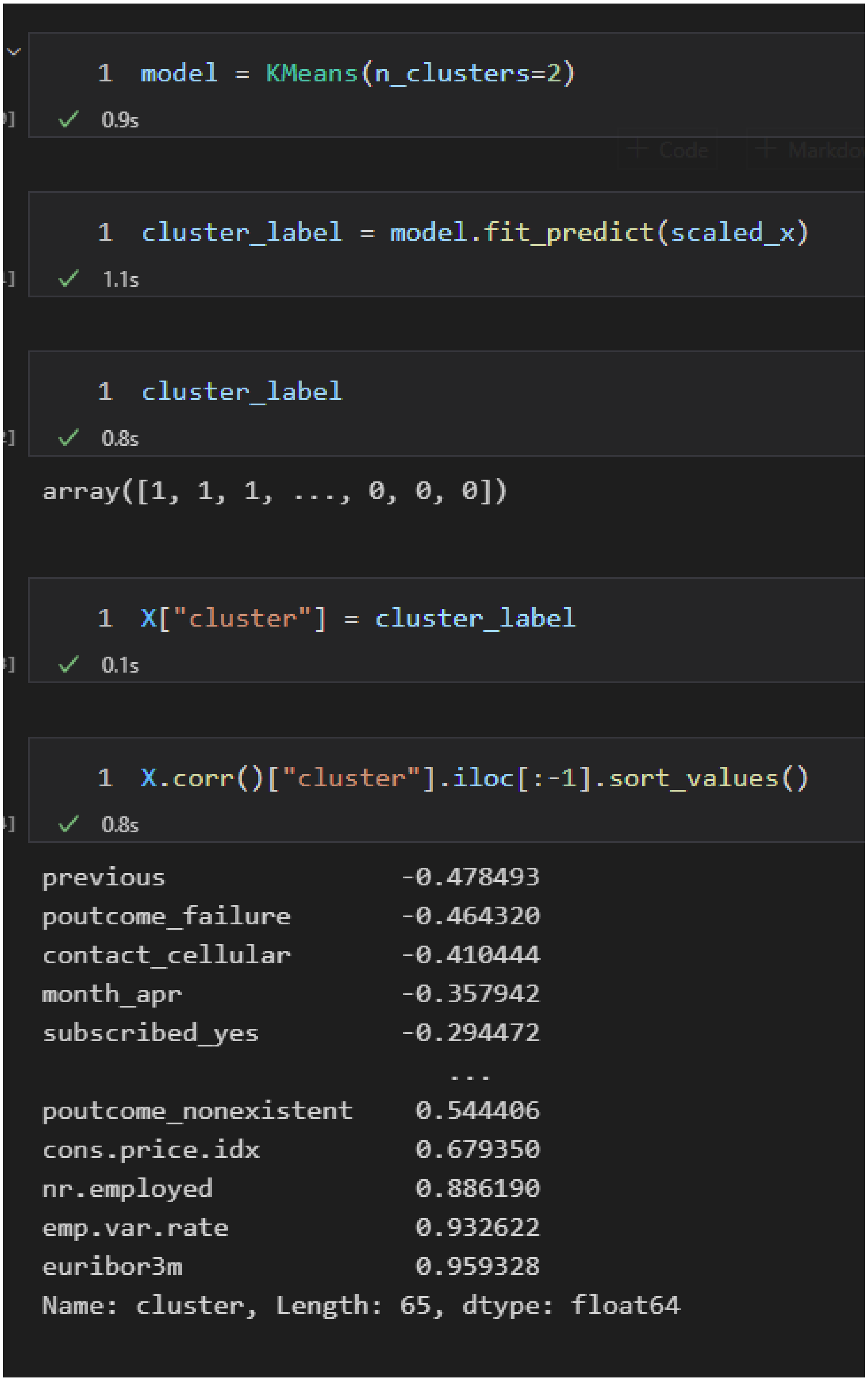
✓ 0.3s



```python
from sklearn.cluster import KMeans
```

```python
help(KMeans)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Help on class KMeans in module sklearn.cluster._kmeans:

class KMeans(sklearn.base.TransformerMixin, sklearn.base.ClusterMixin,
sklearn.base.BaseEstimator)
 |  KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.000
precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True,
```



```python
model = KMeans(n_clusters=2)
```

```python
cluster_label = model.fit_predict(scaled_x)
```

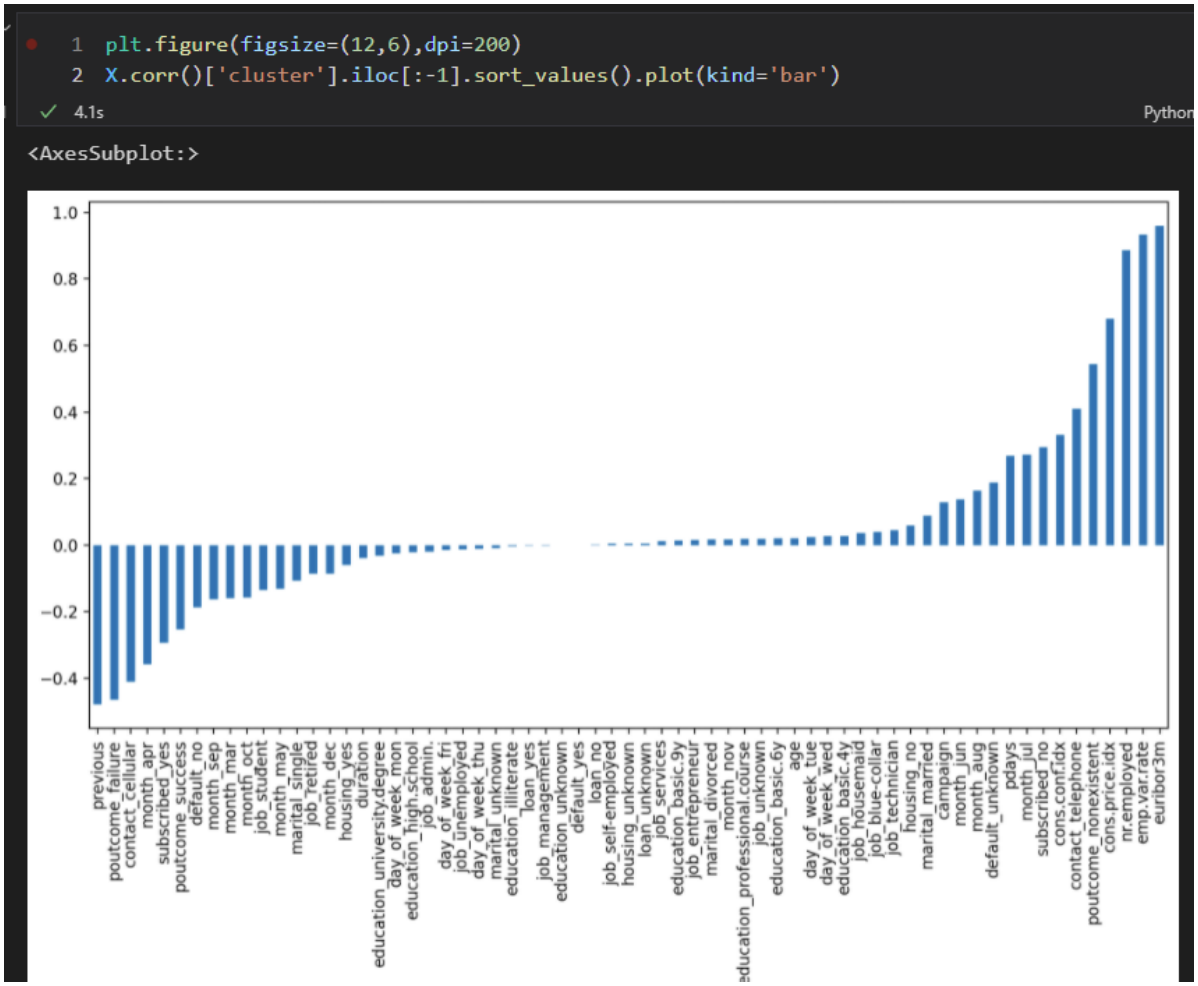
```python
cluster_label
```

```
array([1, 1, 1, ..., 0, 0, 0])
```

```python
X["cluster"] = cluster_label
```

```python
X.corr()["cluster"].iloc[:-1].sort_values()
```

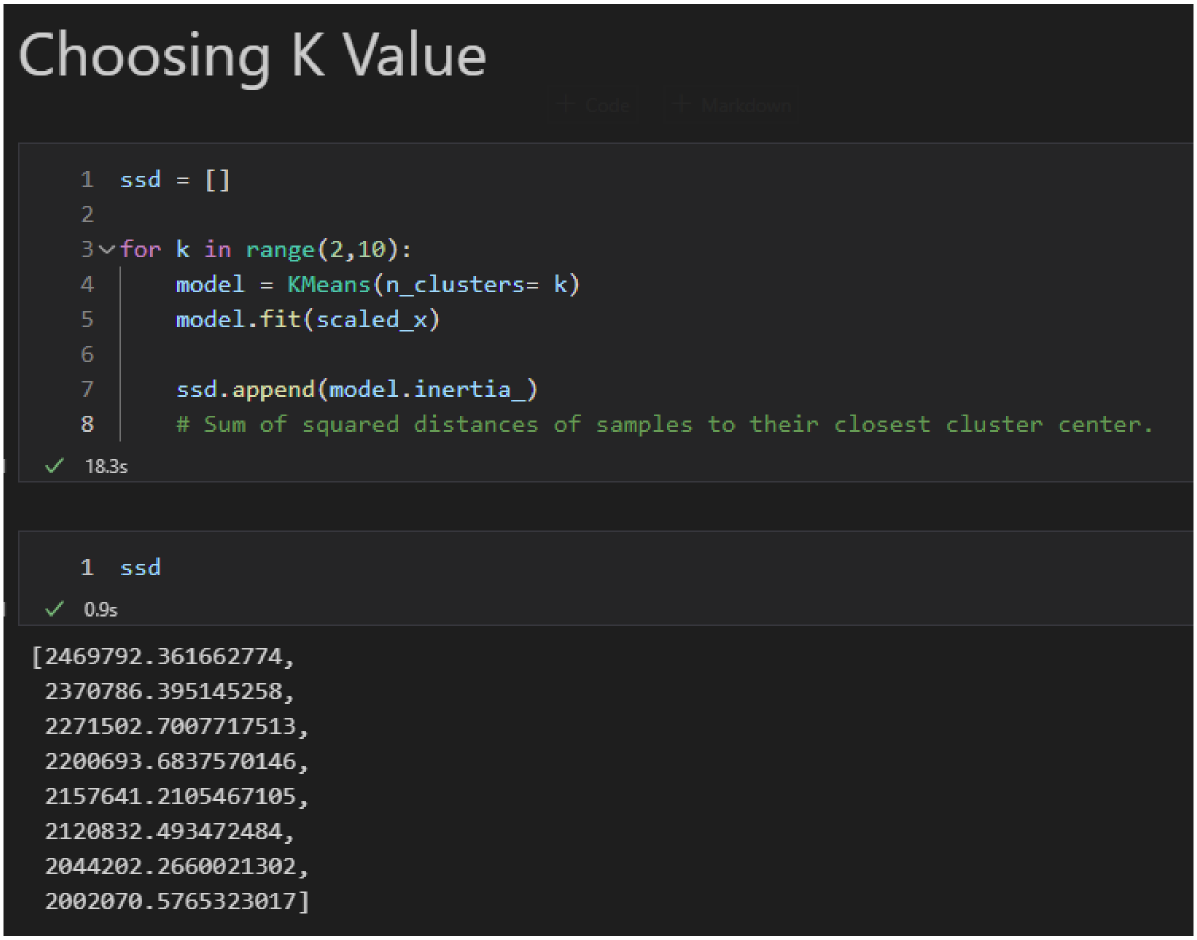
```
previous                -0.478493
poutcome_failure        -0.464320
contact_cellular        -0.410444
month_apr               -0.357942
subscribed_yes          -0.294472
                           ...
poutcome_nonexistent     0.544406
cons.price.idx           0.679350
nr.employed              0.886190
emp.var.rate             0.932622
euribor3m                0.959328
Name: cluster, Length: 65, dtype: float64
```



```
plt.figure(figsize=(12,6),dpi=200)
X.corr()['cluster'].iloc[:-1].sort_values().plot(kind='bar')
```

✓ 4.1s

```
<AxesSubplot:>
```

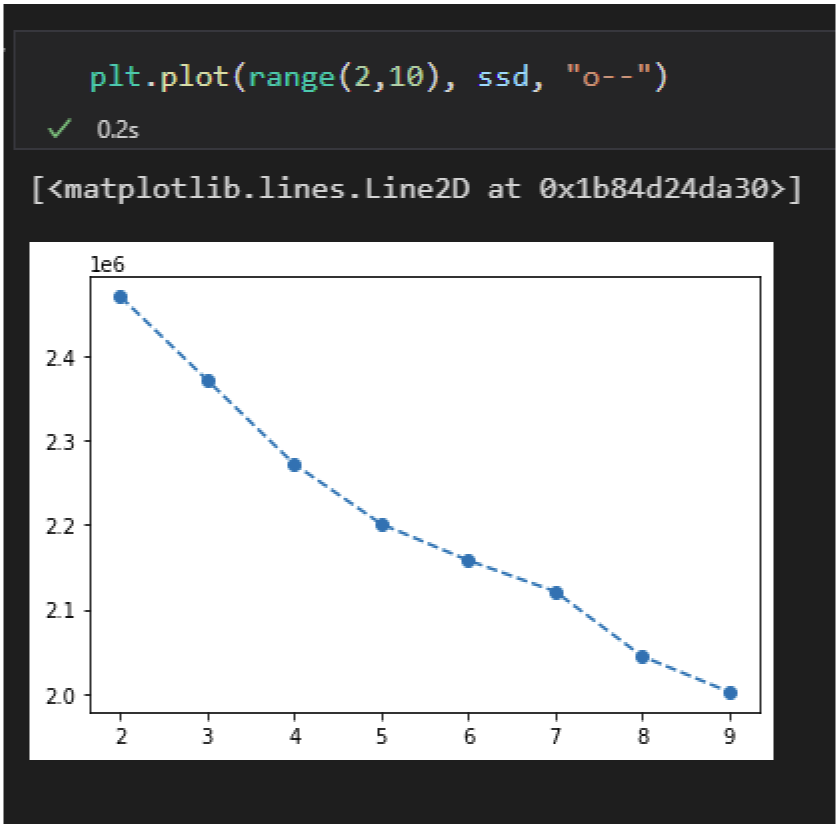


Choosing K Value

```python
ssd = []

for k in range(2,10):
    model = KMeans(n_clusters= k)
    model.fit(scaled_x)

    ssd.append(model.inertia_)
    # Sum of squared distances of samples to their closest cluster center.
```

```python
ssd
```

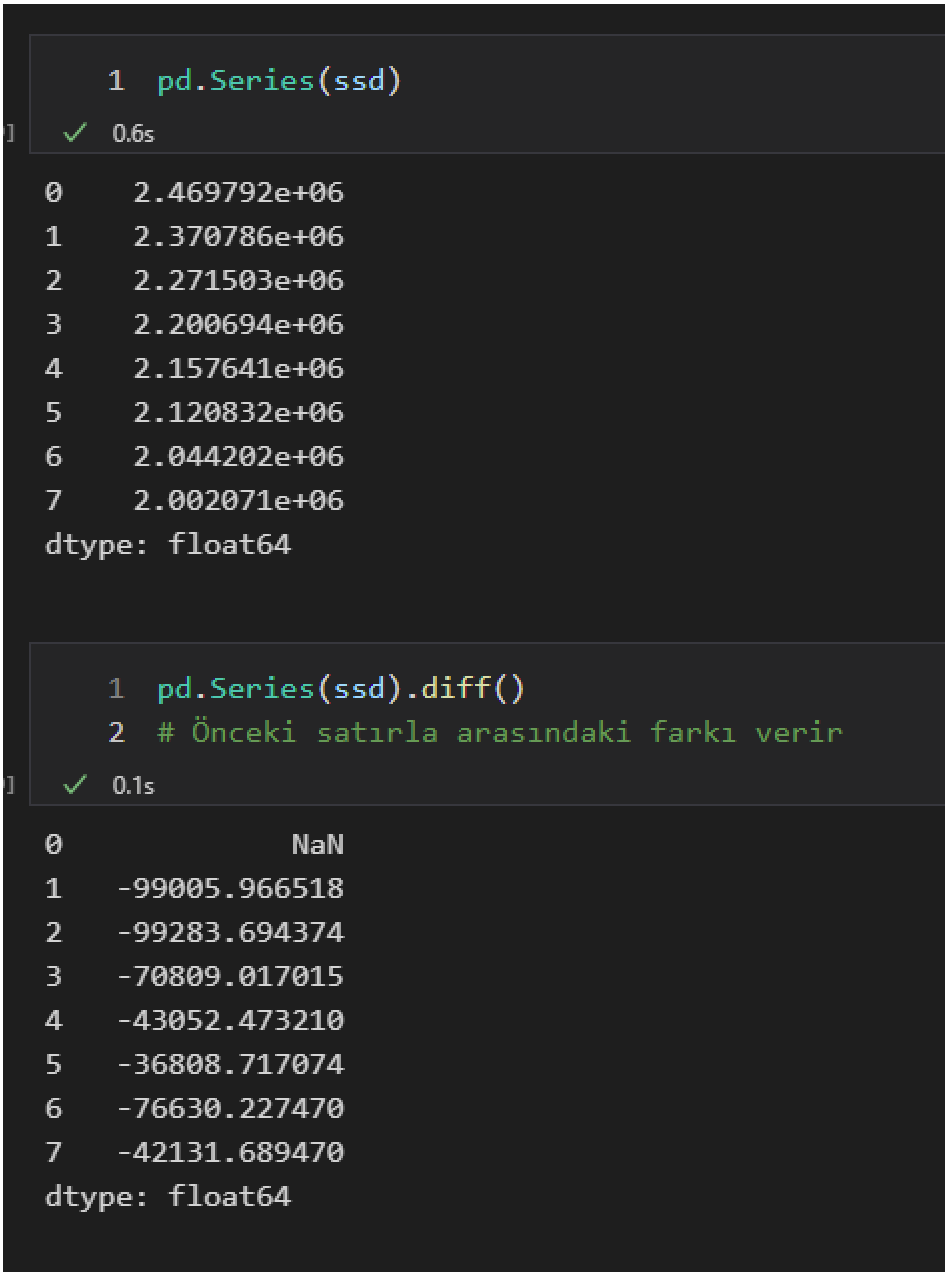
```
[2469792.361662774,
 2370786.395145258,
 2271502.7007717513,
 2200693.6837570146,
 2157641.2105467105,
 2120832.493472484,
 2044202.2660021302,
 2002070.5765323017]
```



```
plt.plot(range(2,10), ssd, "o--")
```

✓ 0.2s

```
[<matplotlib.lines.Line2D at 0x1b84d24da30>]
```

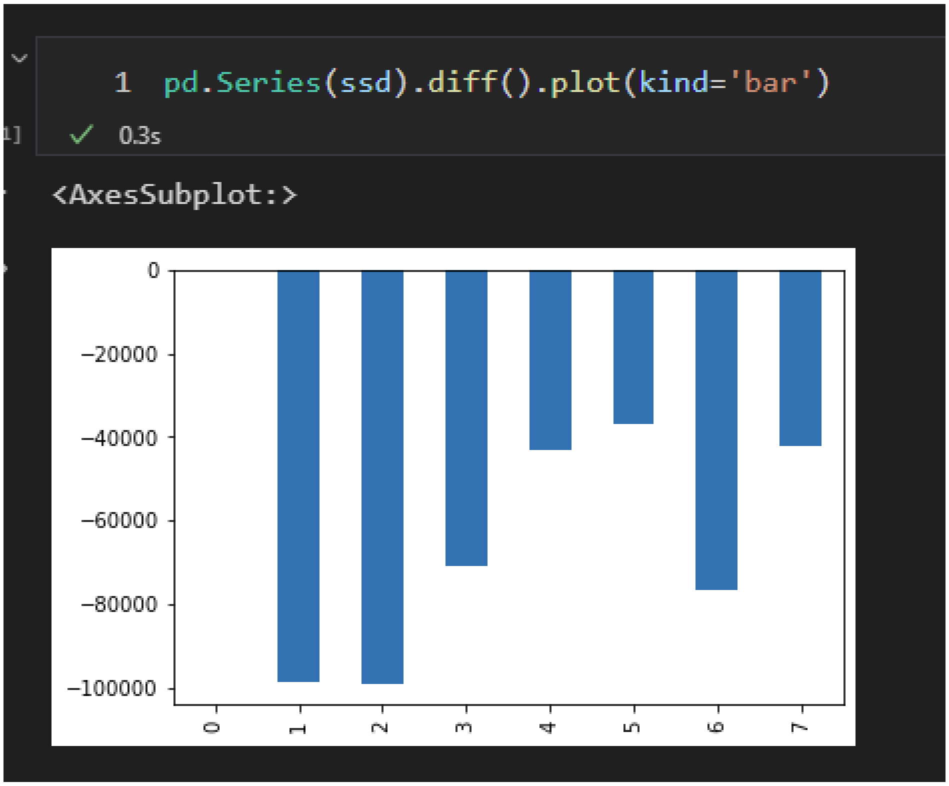


```python
pd.Series(ssd)
```

```
0    2.469792e+06
1    2.370786e+06
2    2.271503e+06
3    2.200694e+06
4    2.157641e+06
5    2.120832e+06
6    2.044202e+06
7    2.002071e+06
dtype: float64
```

```python
pd.Series(ssd).diff()
# Önceki satırla arasındaki farkı verir
```

```
0             NaN
1   -99005.966518
2   -99283.694374
3   -70809.017015
4   -43052.473210
5   -36808.717074
6   -76630.227470
7   -42131.689470
dtype: float64
```

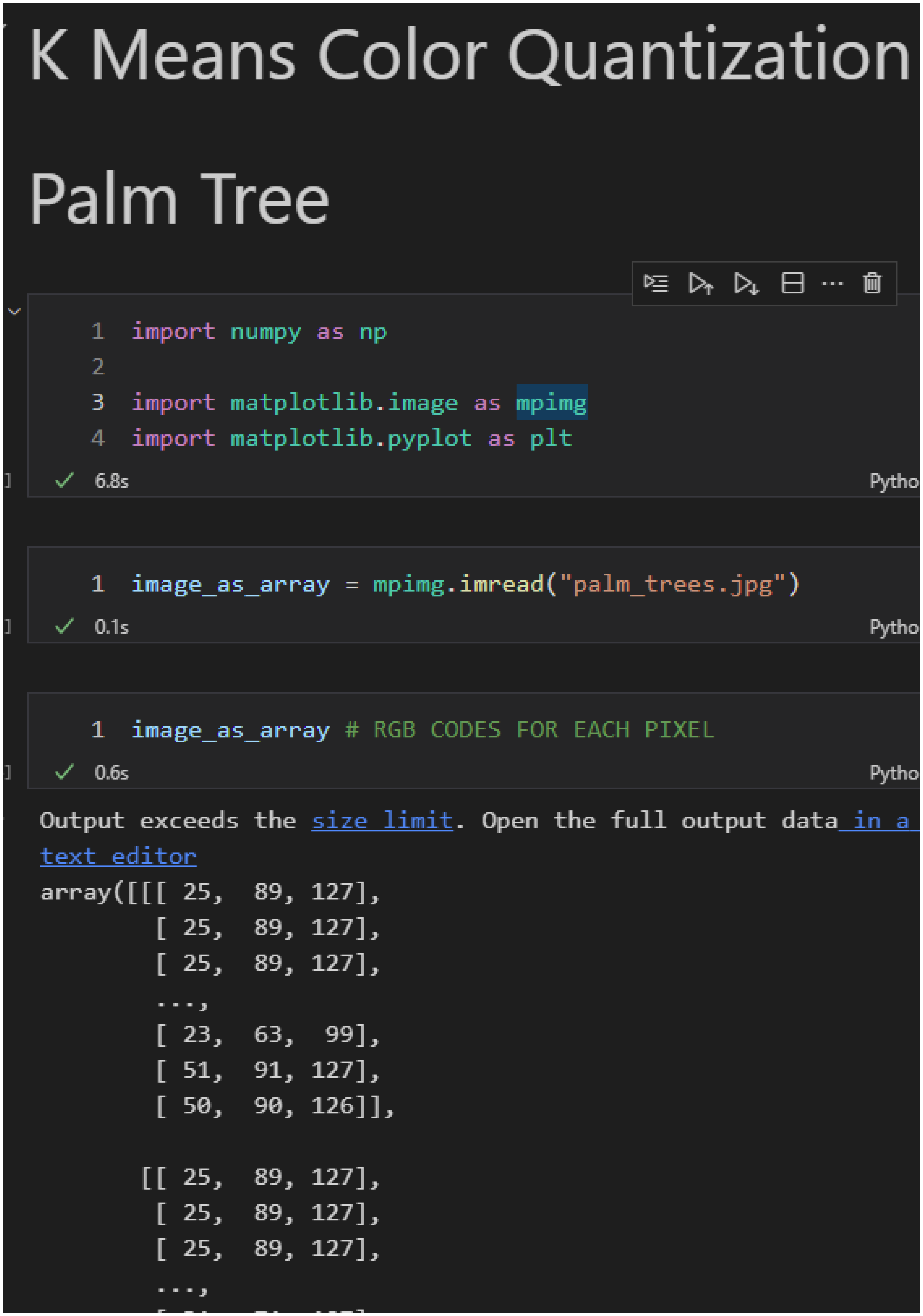


```python
pd.Series(ssd).diff().plot(kind='bar')
```

✓ 0.3s

```
<AxesSubplot:>
```

▼ K Means Color Quantization



K Means Color Quantization

Palm Tree

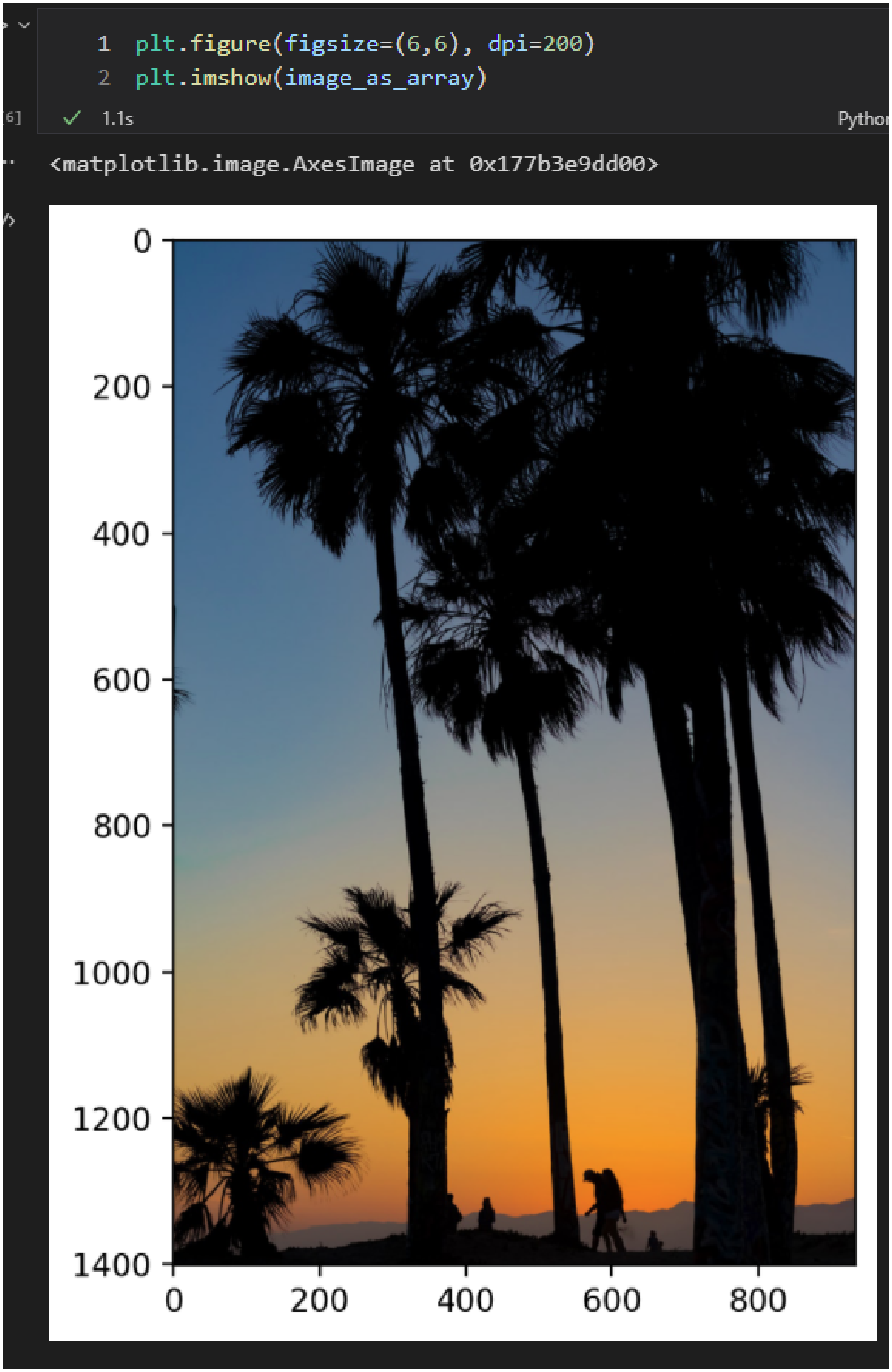
```python
import numpy as np

import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

```python
image_as_array = mpimg.imread("palm_trees.jpg")
```

```python
image_as_array # RGB CODES FOR EACH PIXEL
```

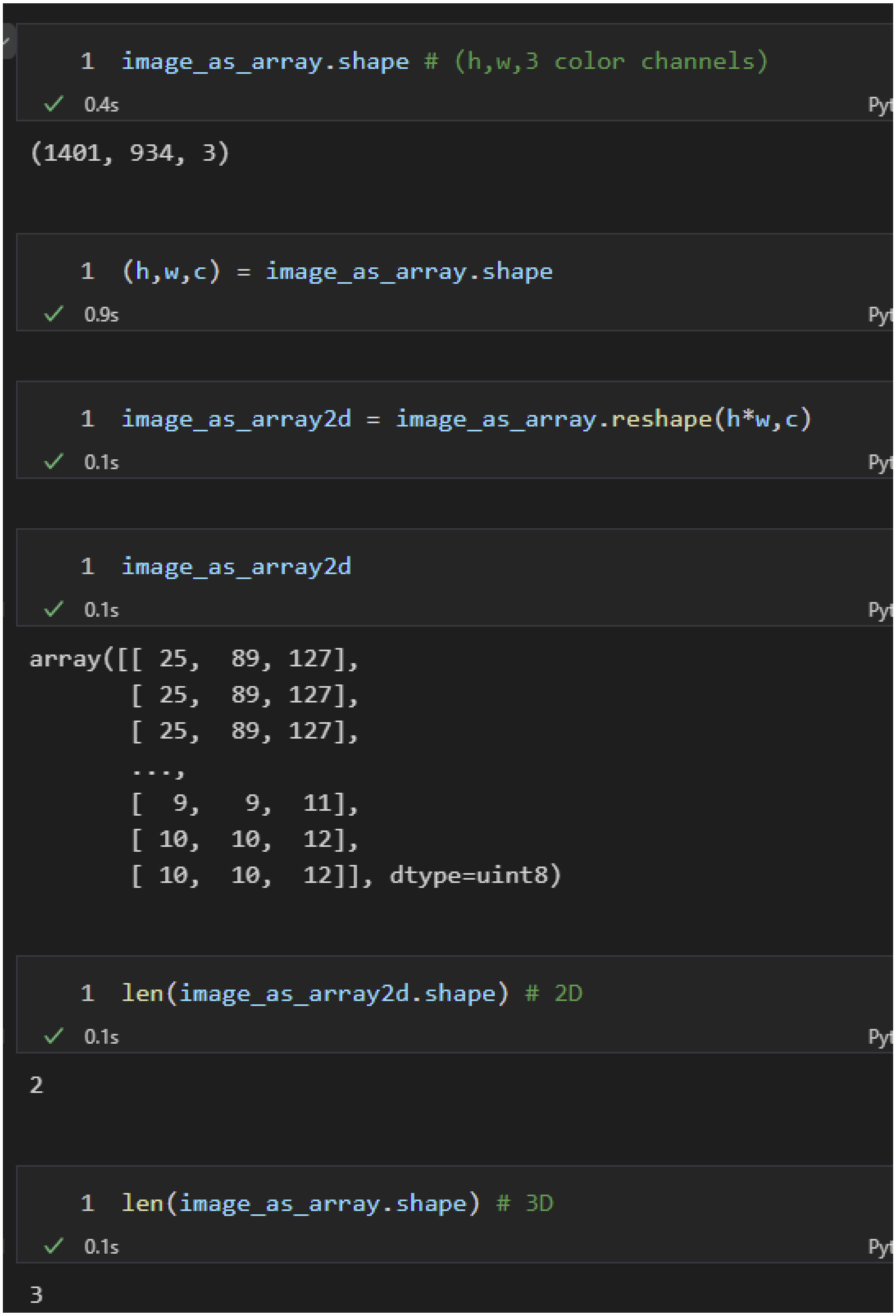
Output exceeds the size limit. Open the full output data in a text editor

```
array([[[ 25,  89, 127],
        [ 25,  89, 127],
        [ 25,  89, 127],
        ...,
        [ 23,  63,  99],
        [ 51,  91, 127],
        [ 50,  90, 126]],

       [[ 25,  89, 127],
        [ 25,  89, 127],
        [ 25,  89, 127],
        ...,
```



```
plt.figure(figsize=(6,6), dpi=200)
plt.imshow(image_as_array)
```

```
<matplotlib.image.AxesImage at 0x177b3e9dd00>
```



```python
image_as_array.shape # (h,w,3 color channels)
```

```
(1401, 934, 3)
```

```python
(h,w,c) = image_as_array.shape
```

```python
image_as_array2d = image_as_array.reshape(h*w,c)
```

```python
image_as_array2d
```

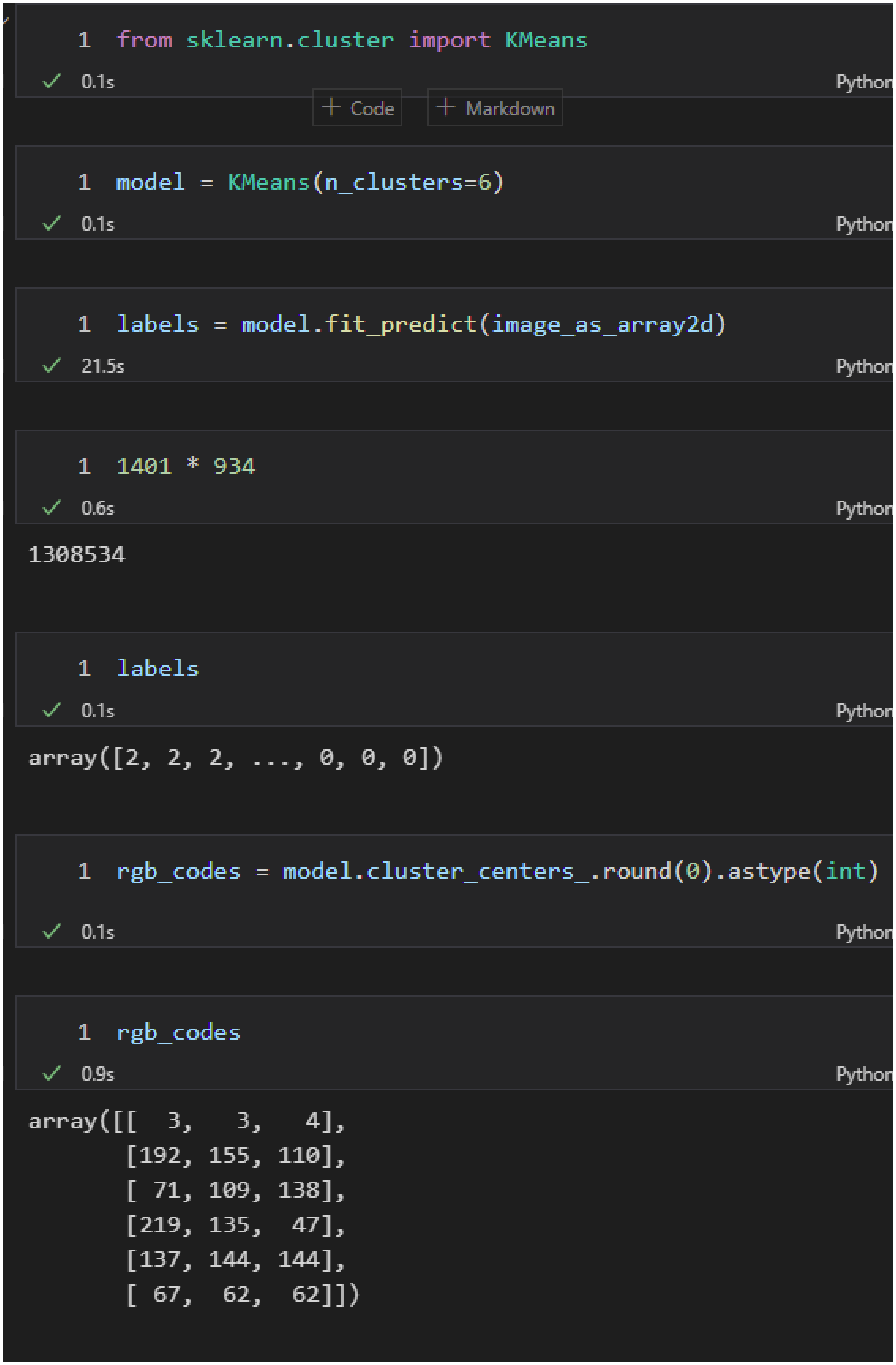
```
array([[ 25,  89, 127],
       [ 25,  89, 127],
       [ 25,  89, 127],
       ...,
       [  9,   9,  11],
       [ 10,  10,  12],
       [ 10,  10,  12]], dtype=uint8)
```

```python
len(image_as_array2d.shape) # 2D
```

```
2
```

```python
len(image_as_array.shape) # 3D
```

```
3
```



```python
from sklearn.cluster import KMeans
```

```python
model = KMeans(n_clusters=6)
```

```python
labels = model.fit_predict(image_as_array2d)
```

```python
1401 * 934
```

```
1308534
```

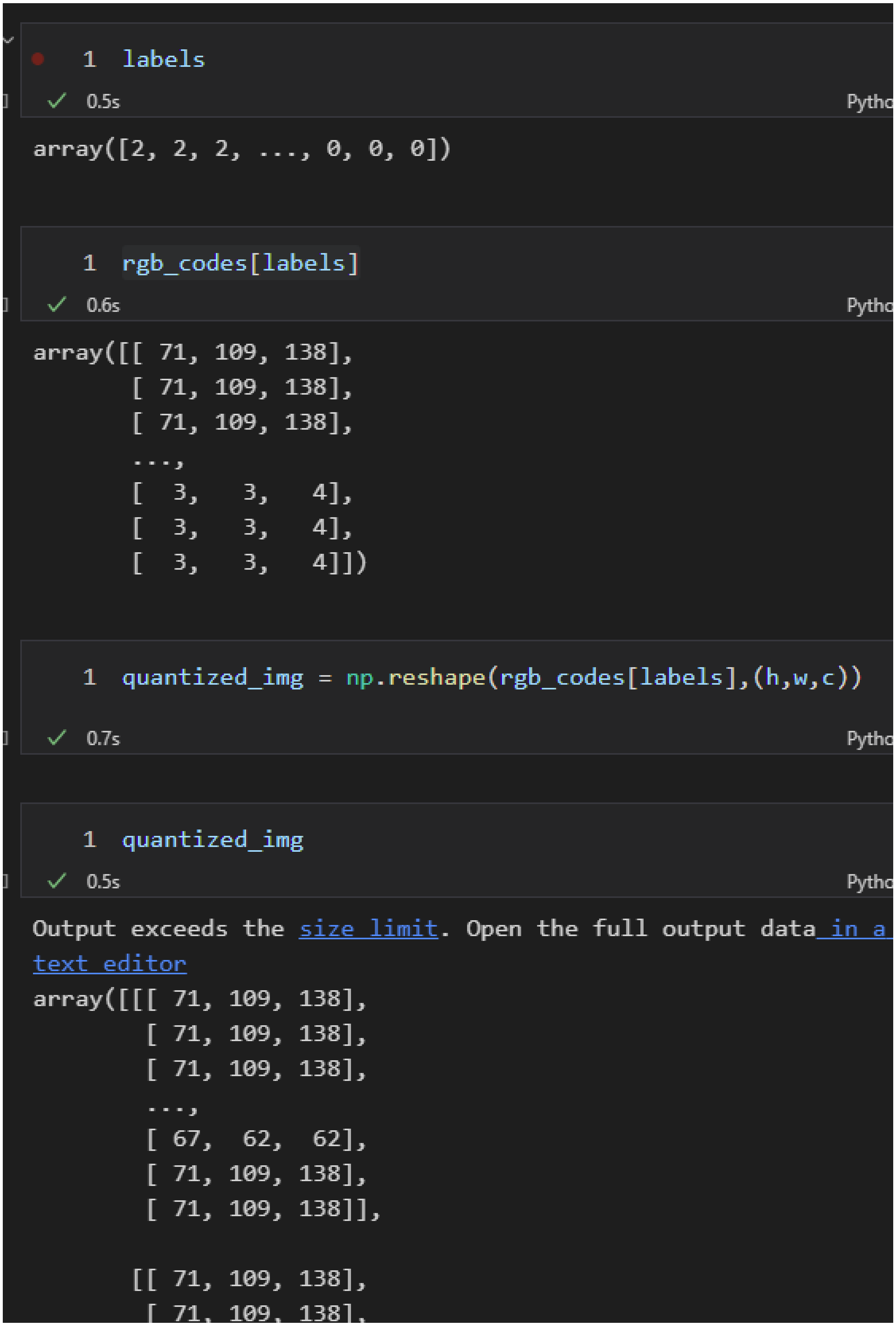
```python
labels
```

```
array([2, 2, 2, ..., 0, 0, 0])
```

```python
rgb_codes = model.cluster_centers_.round(0).astype(int)
```

```python
rgb_codes
```

```
array([[  3,   3,   4],
       [192, 155, 110],
       [ 71, 109, 138],
       [219, 135,  47],
       [137, 144, 144],
       [ 67,  62,  62]])
```



```
labels
```

```
array([2, 2, 2, ..., 0, 0, 0])
```

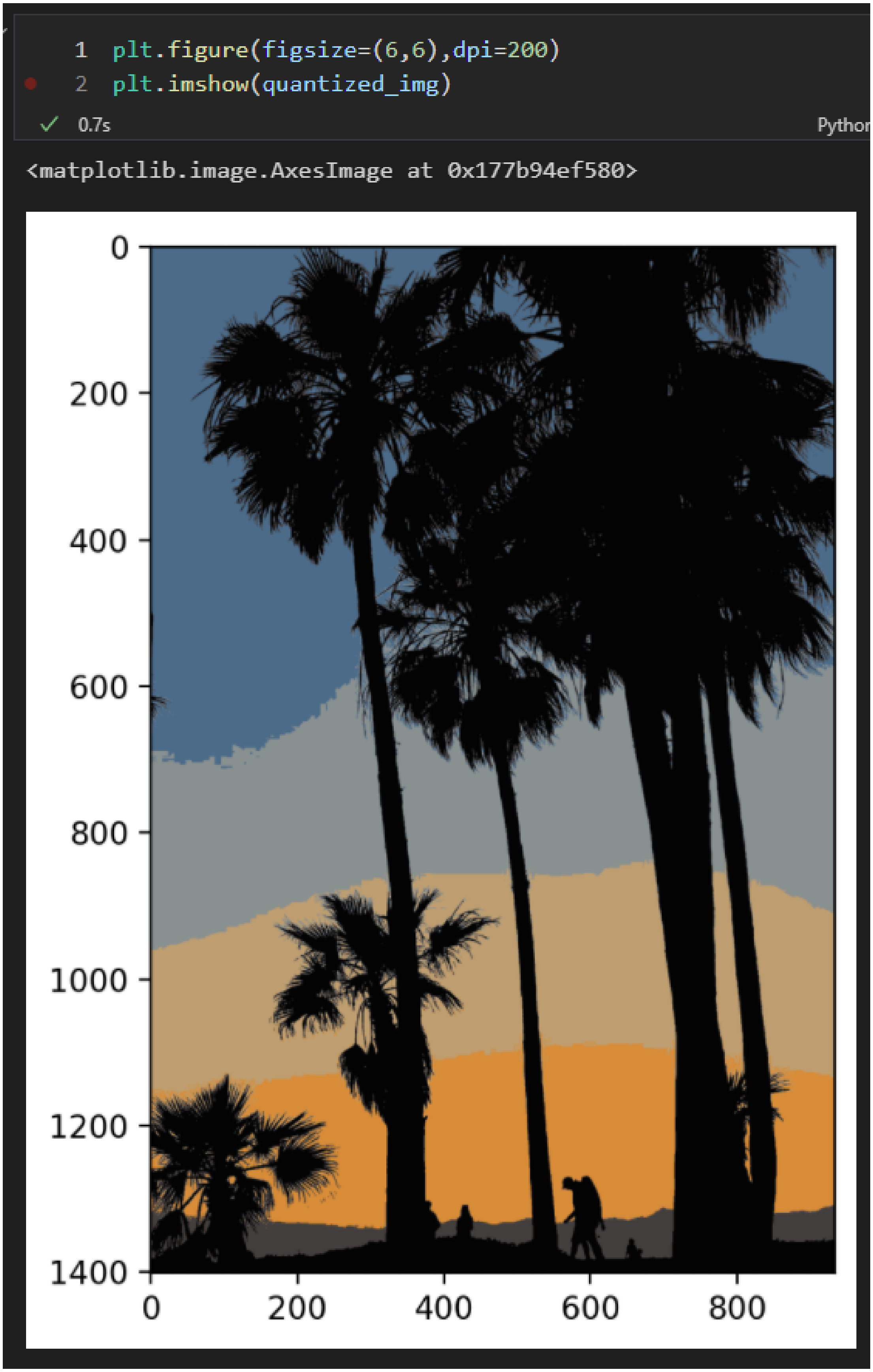
```
rgb_codes[labels]
```

```
array([[ 71, 109, 138],
       [ 71, 109, 138],
       [ 71, 109, 138],
       ...,
       [  3,   3,   4],
       [  3,   3,   4],
       [  3,   3,   4]])
```

```
quantized_img = np.reshape(rgb_codes[labels],(h,w,c))
```

```
quantized_img
```

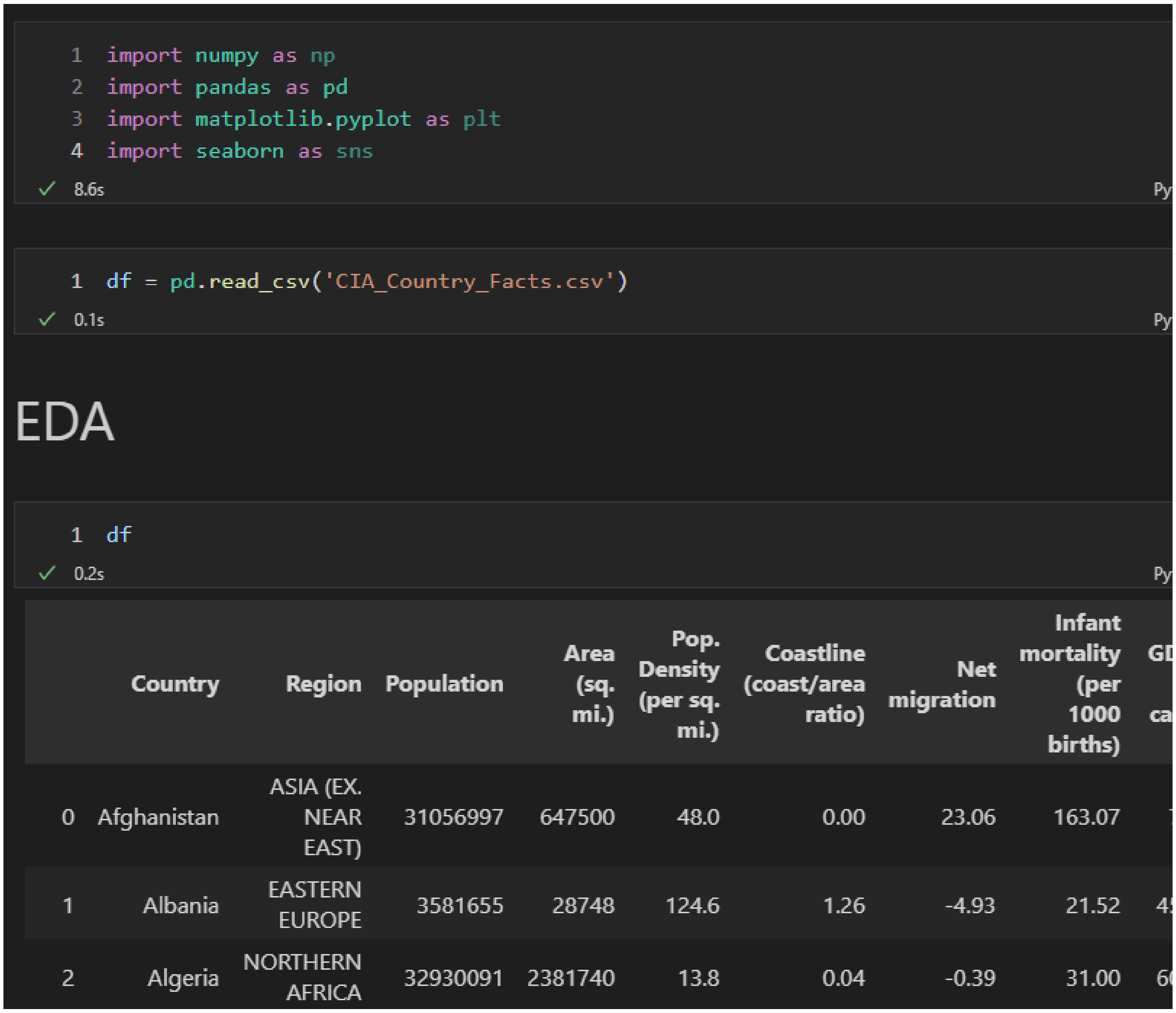
```
Output exceeds the size limit. Open the full output data in a
text editor
array([[[ 71, 109, 138],
        [ 71, 109, 138],
        [ 71, 109, 138],
        ...,
        [ 67,  62,  62],
        [ 71, 109, 138],
        [ 71, 109, 138]],

       [[ 71, 109, 138],
        [ 71, 109, 138],
```



```
plt.figure(figsize=(6,6),dpi=200)
plt.imshow(quantized_img)
```

```
<matplotlib.image.AxesImage at 0x177b94ef580>
```

▼ CIA Country Analysis



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

✓ 8.6s Py

```
df = pd.read_csv('CIA_Country_Facts.csv')
```

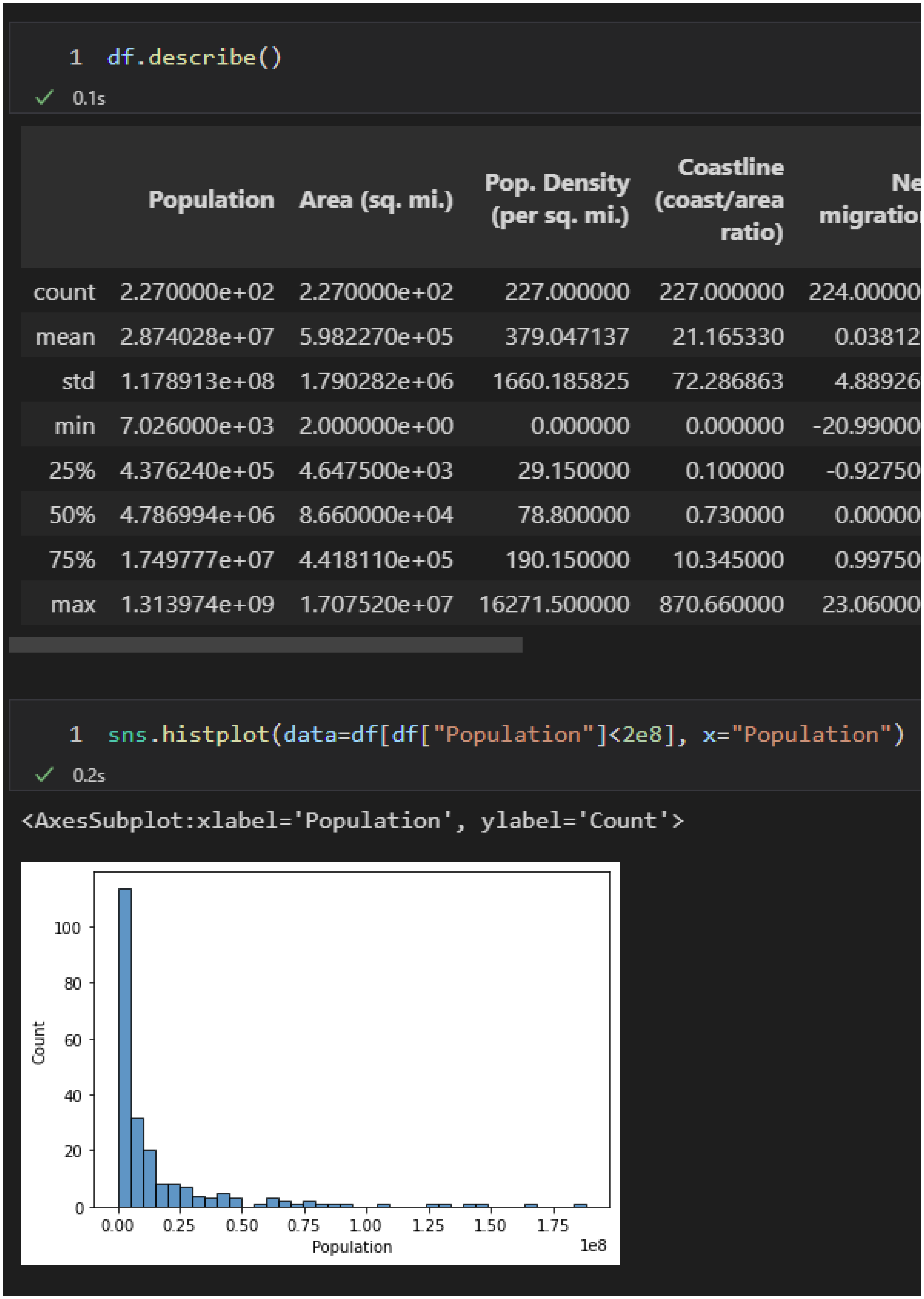
✓ 0.1s Py

EDA

```
df
```

✓ 0.2s Py

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GD ca
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	[illegible]
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	45
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	60



```
df.describe()
```

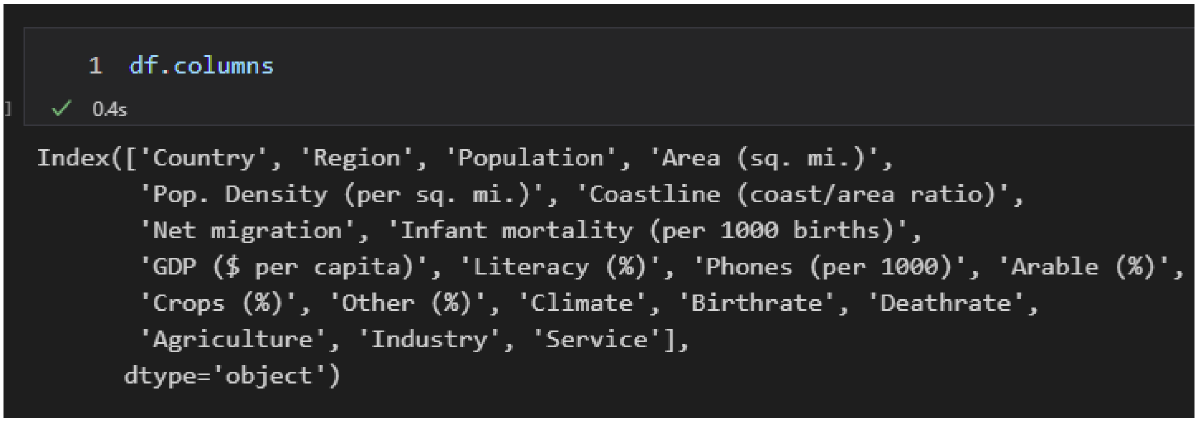
✓ 0.1s

	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Ne migratio
count	2.270000e+02	2.270000e+02	227.000000	227.000000	224.00000
mean	2.874028e+07	5.982270e+05	379.047137	21.165330	0.03812
std	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.88926
min	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.99000
25%	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.92750
50%	4.786994e+06	8.660000e+04	78.800000	0.730000	0.00000
75%	1.749777e+07	4.418110e+05	190.150000	10.345000	0.99750
max	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.06000

```
sns.histplot(data=df[df["Population"]<2e8], x="Population")
```

✓ 0.2s

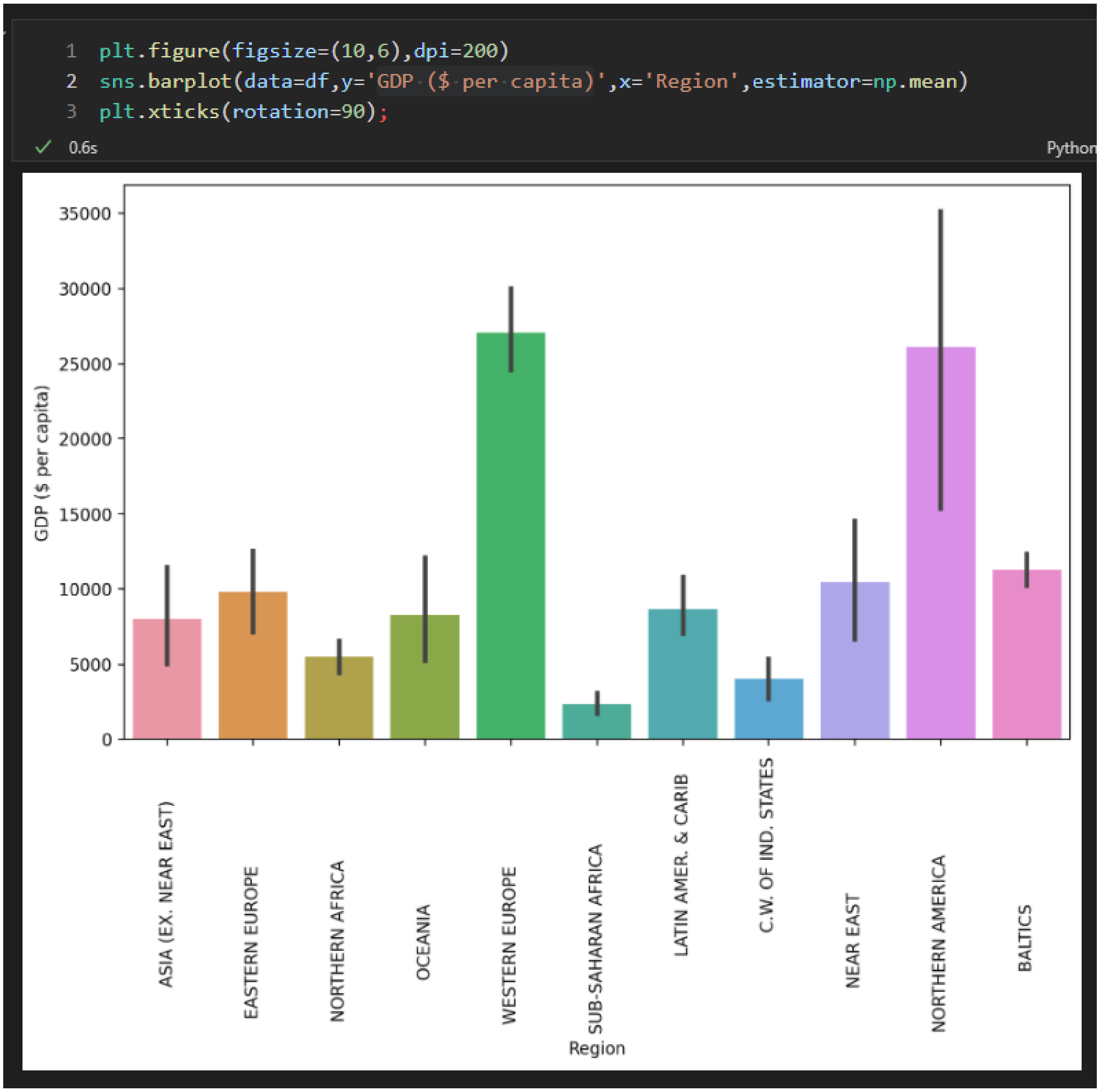
```
<AxesSubplot:xlabel='Population', ylabel='Count'>
```



```
df.columns
```

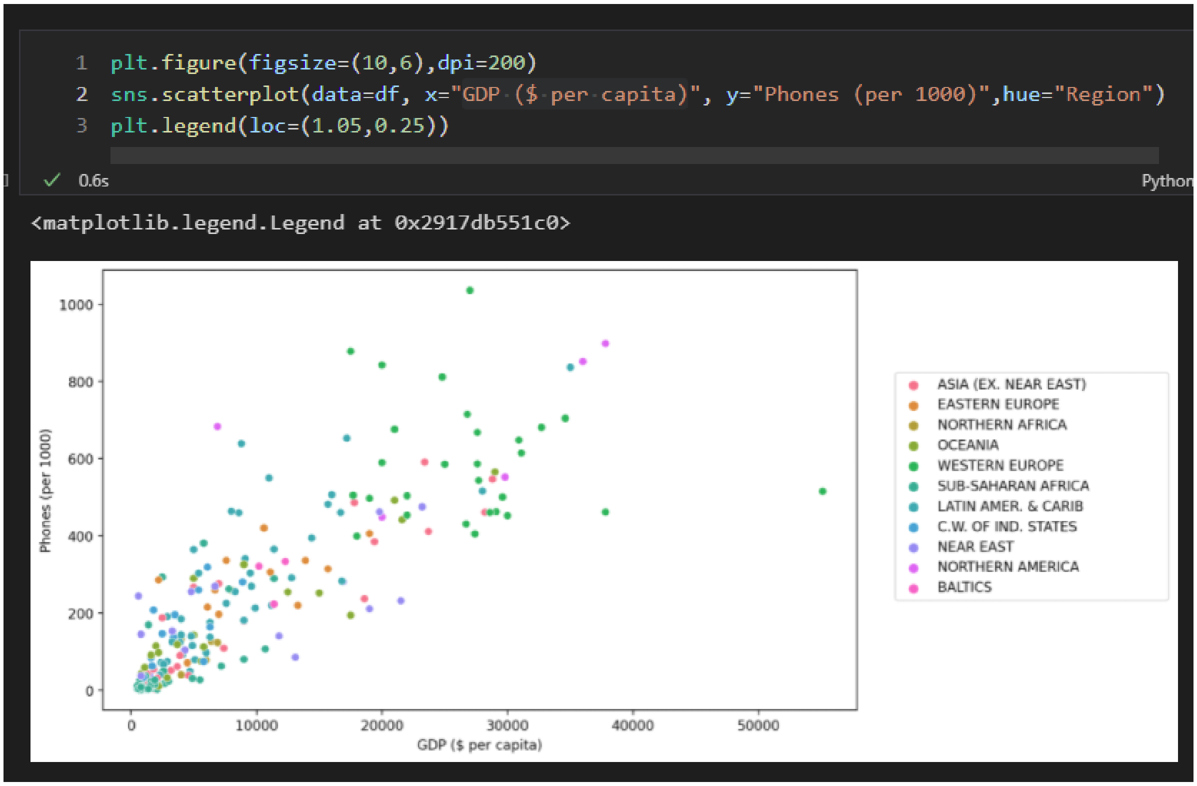
✓ 0.4s

```
Index(['Country', 'Region', 'Population', 'Area (sq. mi.)',
       'Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)',
       'Net migration', 'Infant mortality (per 1000 births)',
       'GDP ($ per capita)', 'Literacy (%)', 'Phones (per 1000)', 'Arable (%)',
       'Crops (%)', 'Other (%)', 'Climate', 'Birthrate', 'Deathrate',
       'Agriculture', 'Industry', 'Service'],
      dtype='object')
```



```
plt.figure(figsize=(10,6),dpi=200)
sns.barplot(data=df,y='GDP ($ per capita)',x='Region',estimator=np.mean)
plt.xticks(rotation=90);
```

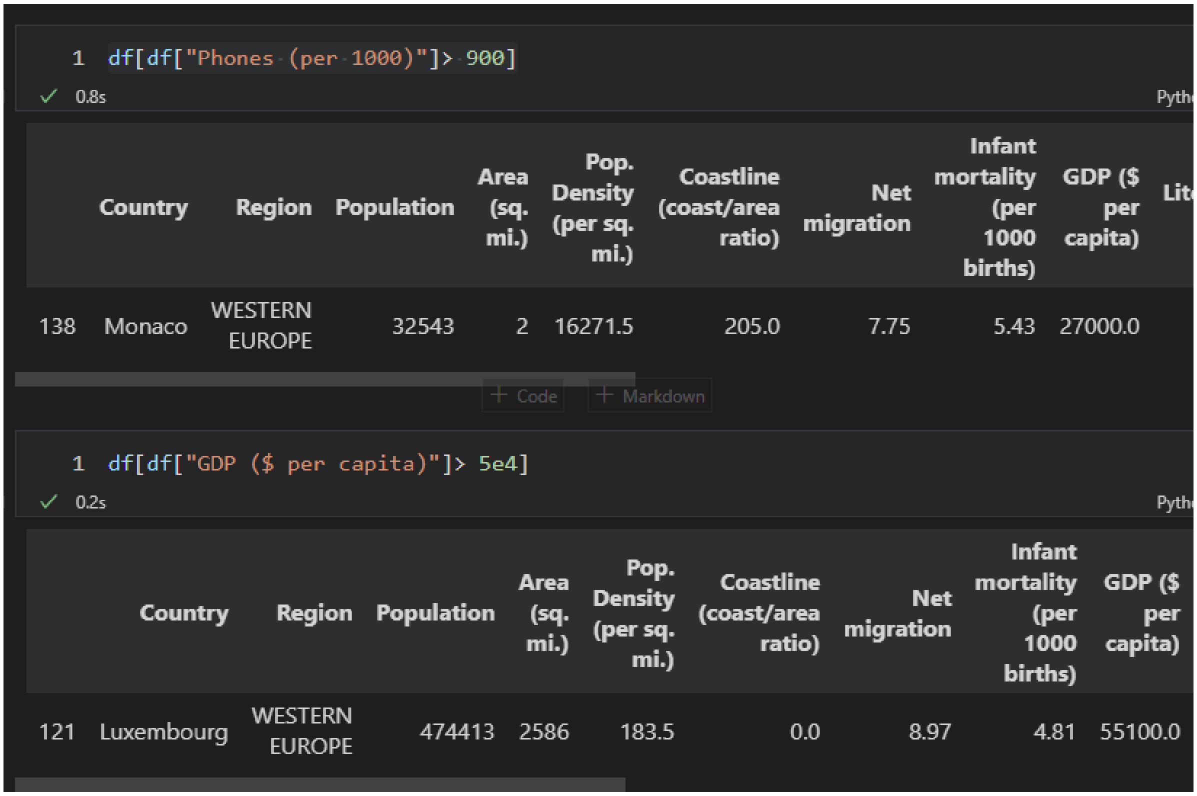
✓ 0.6s



```
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df, x="GDP ($ per capita)", y="Phones (per 1000)",hue="Region")
plt.legend(loc=(1.05,0.25))
```

✓ 0.6s

```
<matplotlib.legend.Legend at 0x2917db551c0>
```



```
df[df["Phones (per 1000)"]> 900]
```

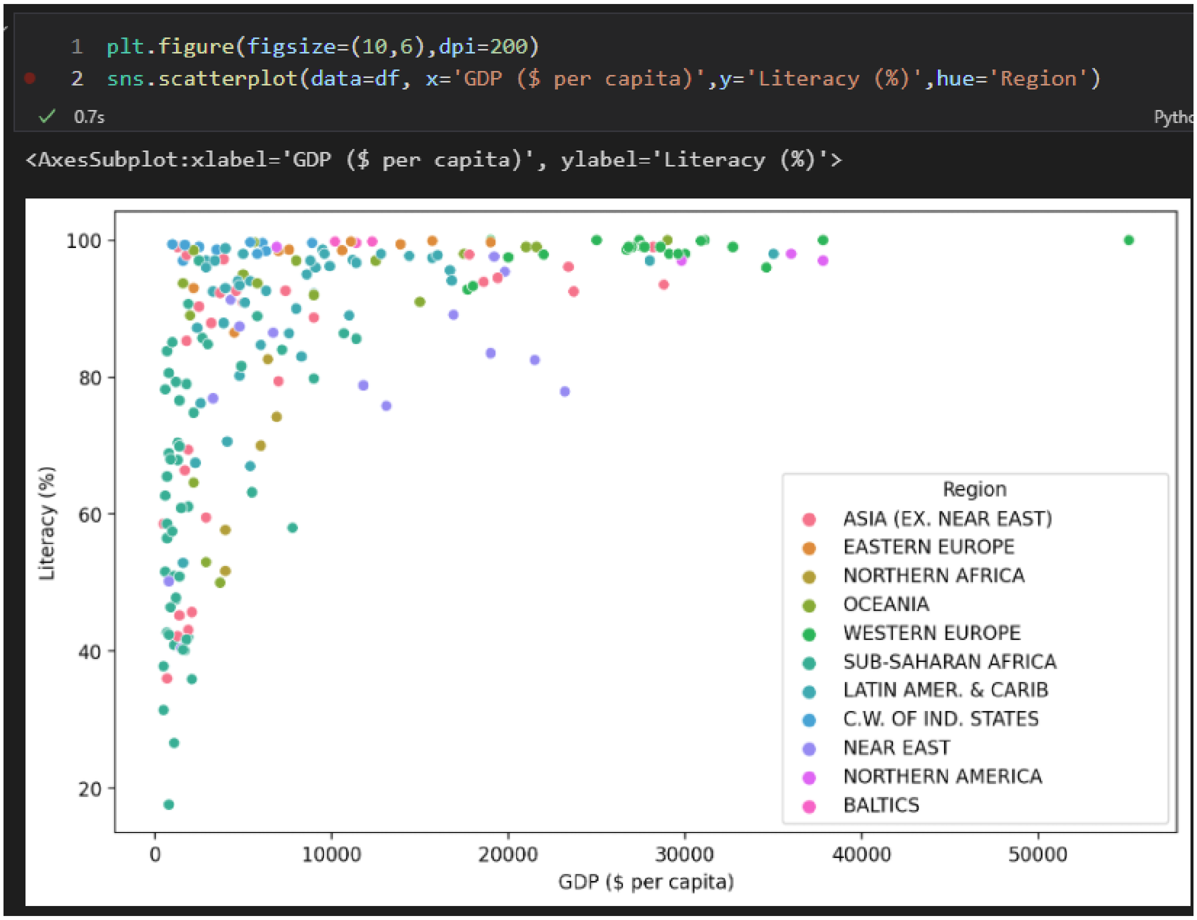
✓ 0.8s

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Lit
138	Monaco	WESTERN EUROPE	32543	2	16271.5	205.0	7.75	5.43	27000.0	

```
df[df["GDP ($ per capita)"]> 5e4]
```

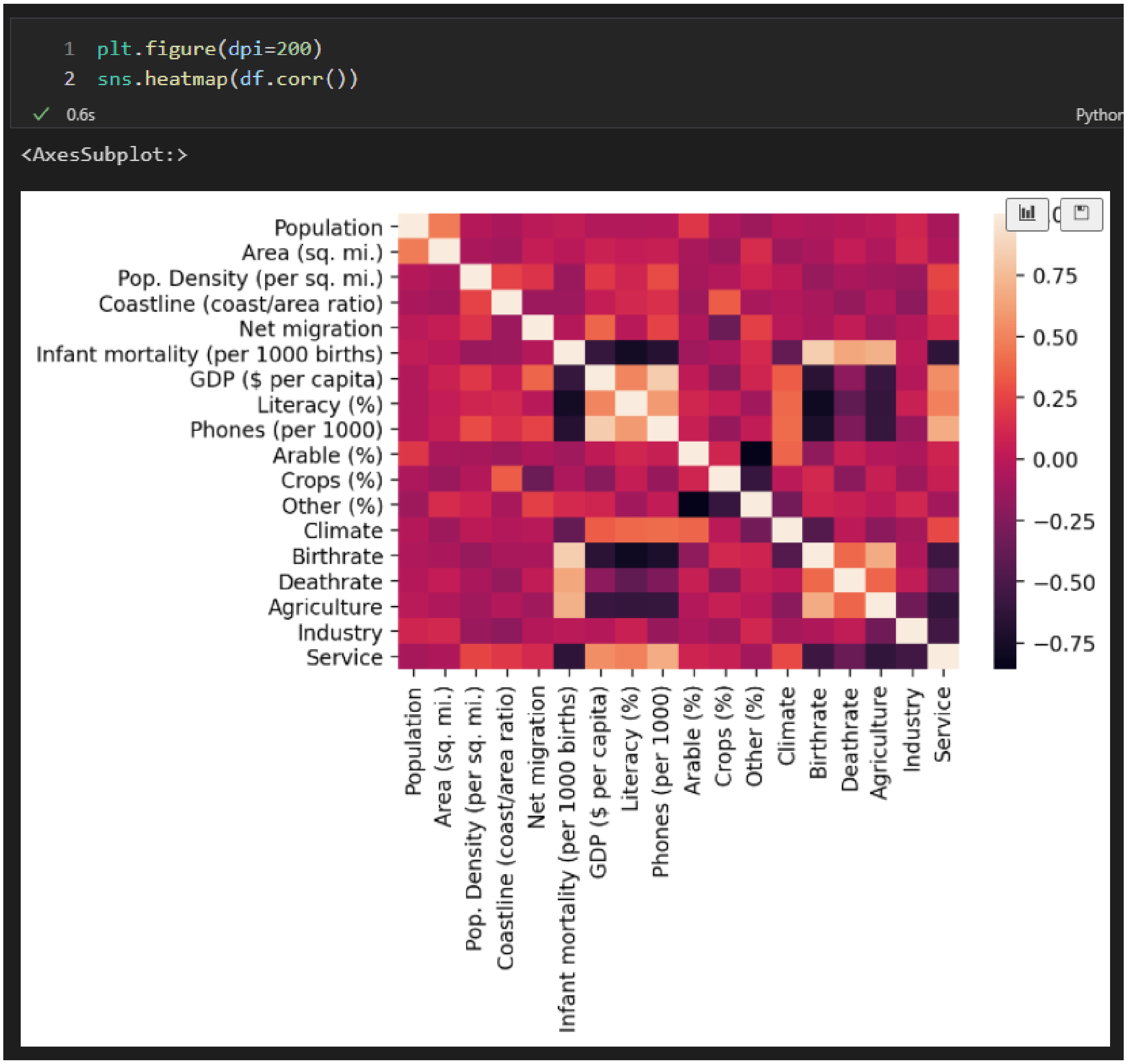
✓ 0.2s

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)
121	Luxembourg	WESTERN EUROPE	474413	2586	183.5	0.0	8.97	4.81	55100.0



```
plt.figure(figsize=(10,6),dpi=200)
sns.scatterplot(data=df, x='GDP ($ per capita)',y='Literacy (%)',hue='Region')
```

```
<AxesSubplot:xlabel='GDP ($ per capita)', ylabel='Literacy (%)'>
```

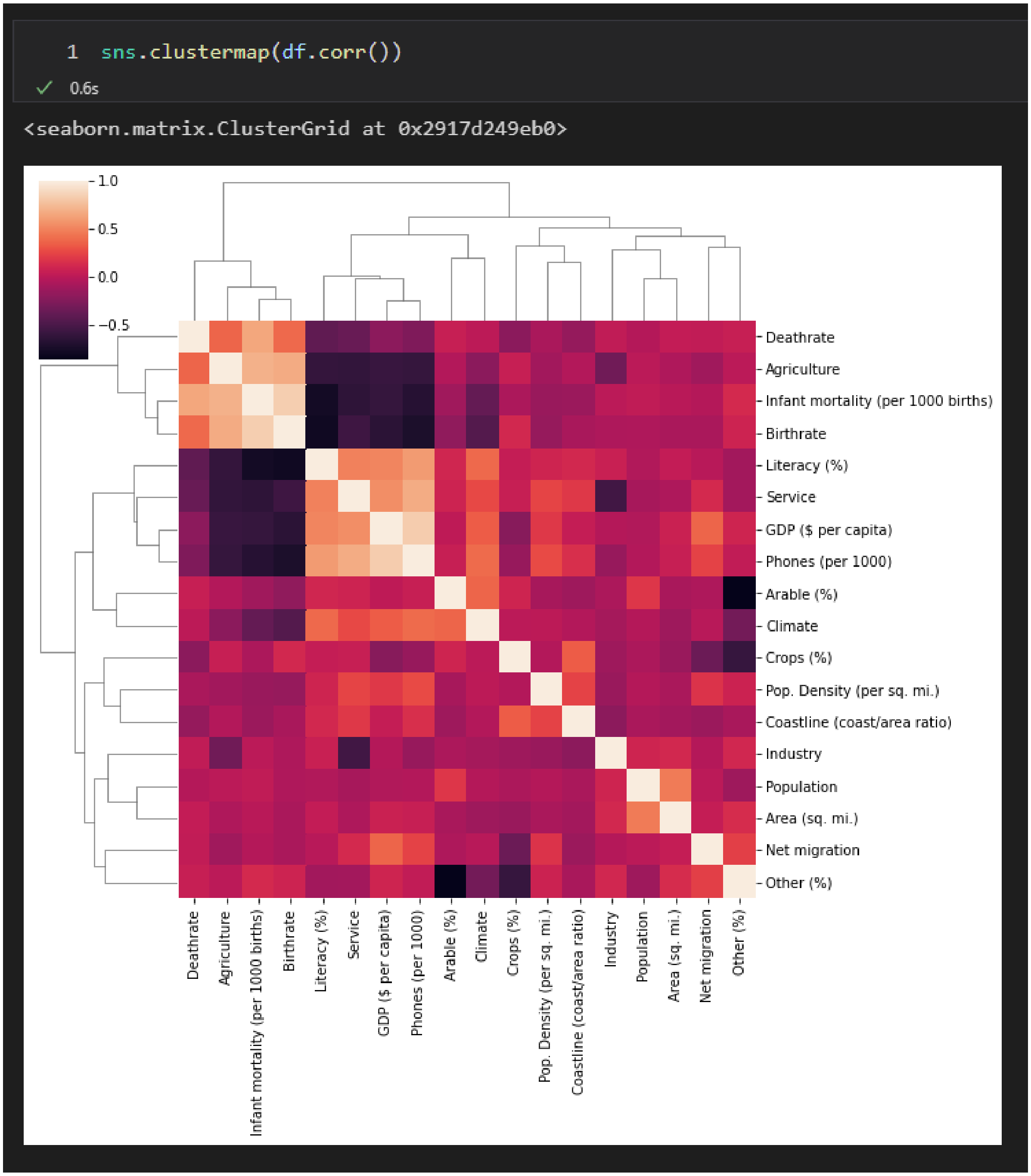


```
plt.figure(dpi=200)
sns.heatmap(df.corr())
```

✓ 0.6s

Pythor

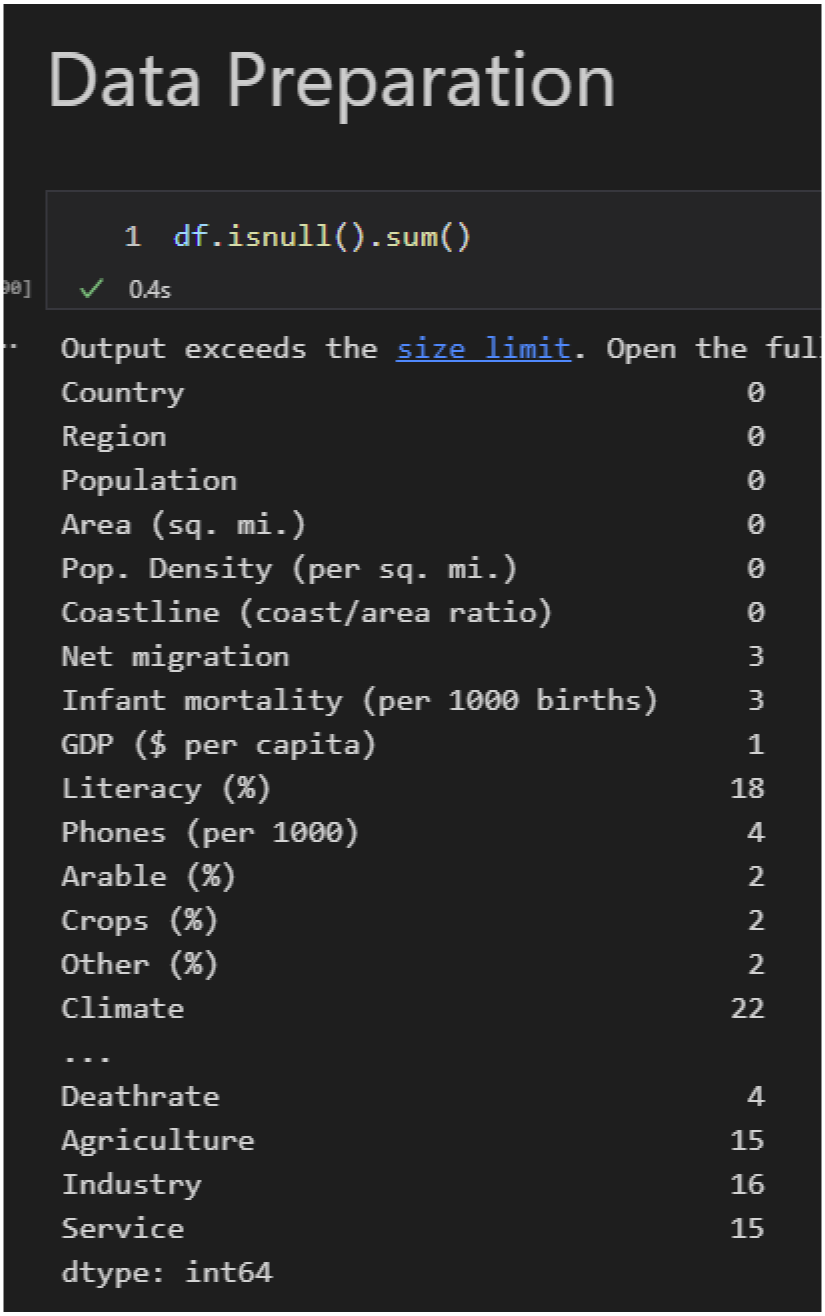
```
<AxesSubplot:>
```



```
sns.clustermap(df.corr())
```

✓ 0.6s

```
<seaborn.matrix.ClusterGrid at 0x2917d249eb0>
```

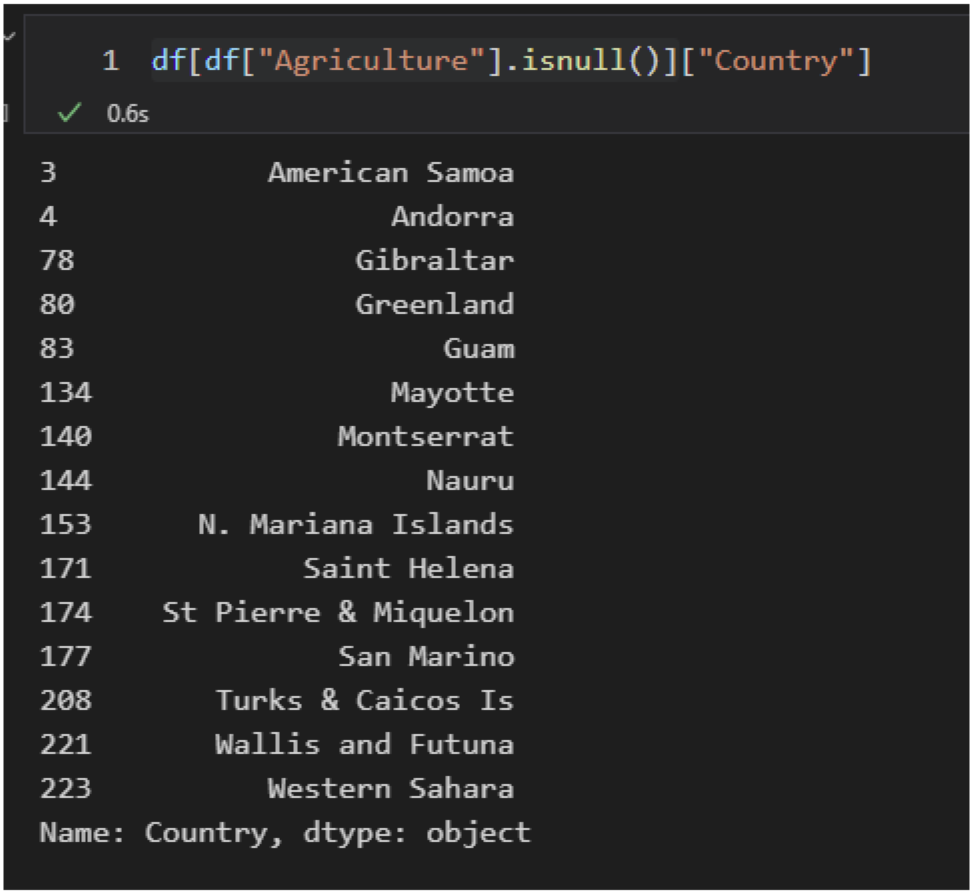


Data Preparation

```
df.isnull().sum()
```

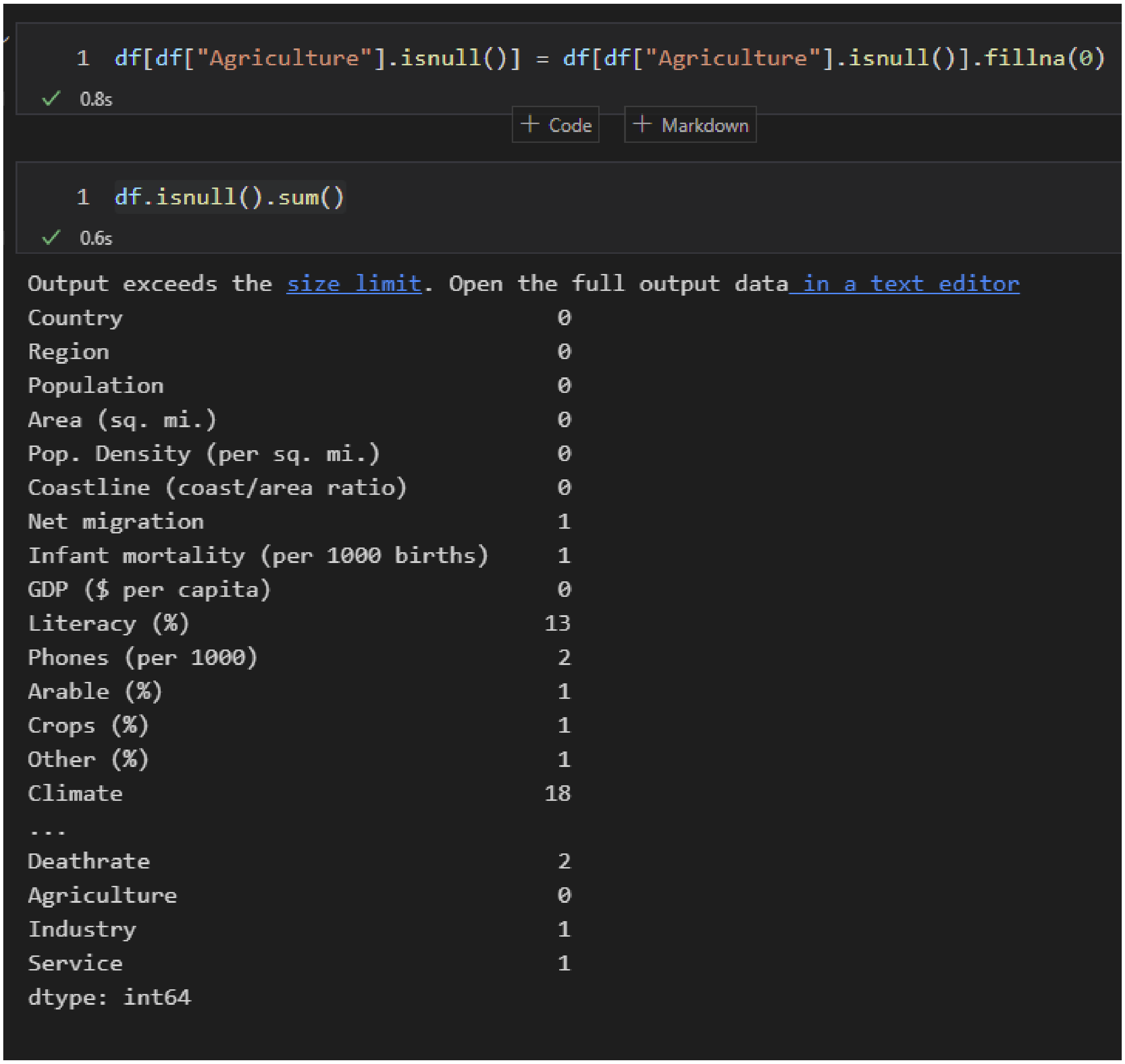
✓ 0.4s

```
Output exceeds the size limit. Open the ful
Country                                0
Region                                 0
Population                             0
Area (sq. mi.)                         0
Pop. Density (per sq. mi.)             0
Coastline (coast/area ratio)           0
Net migration                          3
Infant mortality (per 1000 births)     3
GDP ($ per capita)                     1
Literacy (%)                          18
Phones (per 1000)                      4
Arable (%)                             2
Crops (%)                              2
Other (%)                              2
Climate                               22
...
Deathrate                              4
Agriculture                           15
Industry                              16
Service                               15
dtype: int64
```



```python
df[df["Agriculture"].isnull()]["Country"]
```

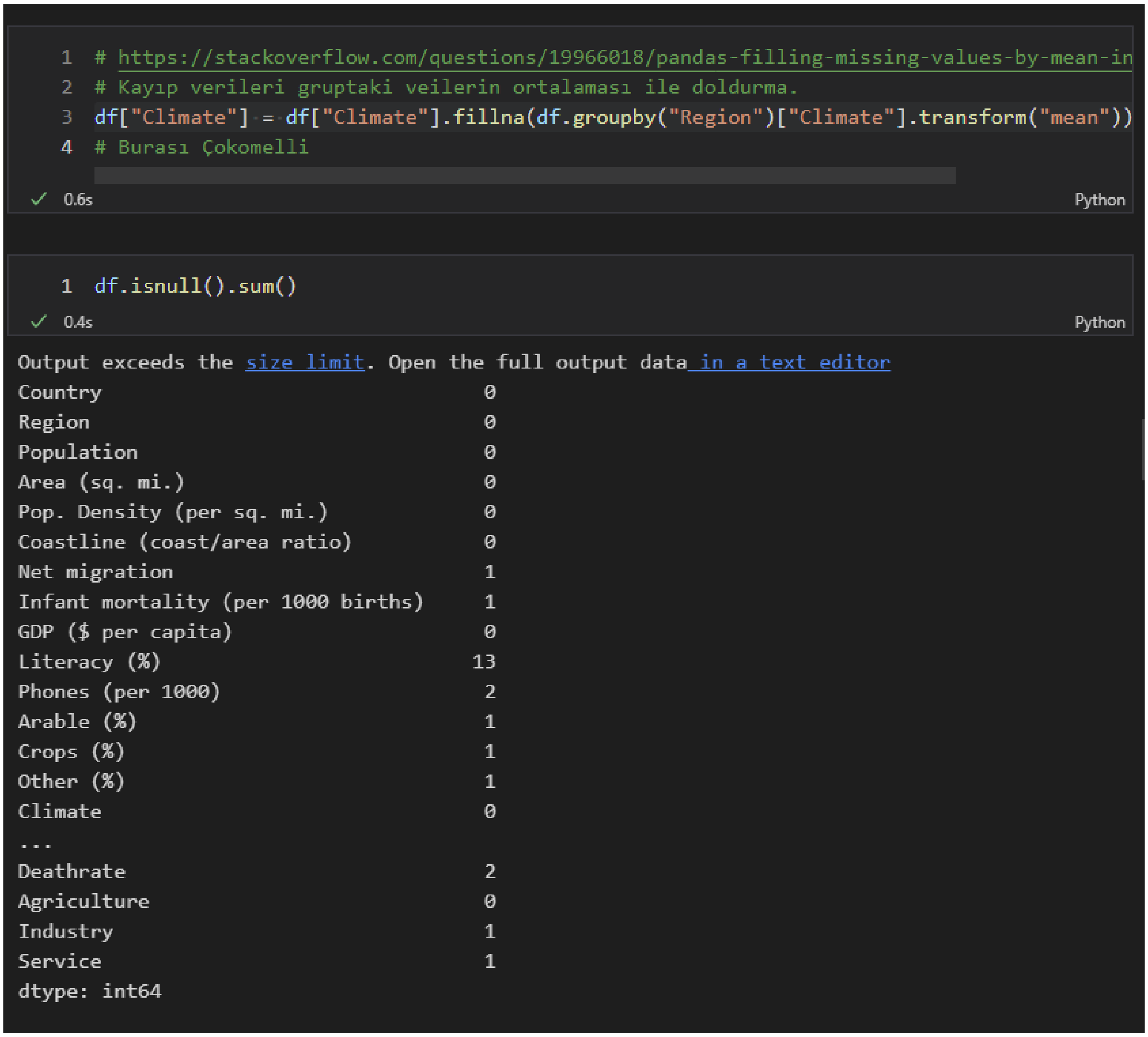
```
3            American Samoa
4                   Andorra
78                Gibraltar
80                Greenland
83                     Guam
134                 Mayotte
140              Montserrat
144                   Nauru
153      N. Mariana Islands
171            Saint Helena
174    St Pierre & Miquelon
177              San Marino
208       Turks & Caicos Is
221       Wallis and Futuna
223          Western Sahara
Name: Country, dtype: object
```



```
df[df["Agriculture"].isnull()] = df[df["Agriculture"].isnull()].fillna(0)
```

```
df.isnull().sum()
```

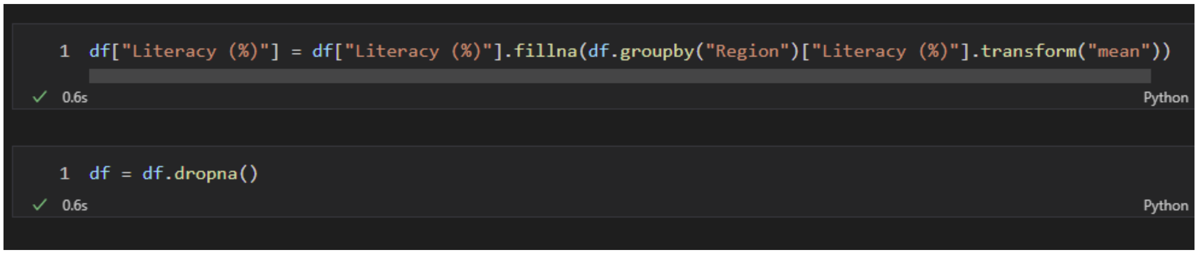
```
Output exceeds the size limit. Open the full output data in a text editor
Country                                0
Region                                 0
Population                             0
Area (sq. mi.)                         0
Pop. Density (per sq. mi.)             0
Coastline (coast/area ratio)           0
Net migration                          1
Infant mortality (per 1000 births)     1
GDP ($ per capita)                     0
Literacy (%)                          13
Phones (per 1000)                      2
Arable (%)                             1
Crops (%)                              1
Other (%)                              1
Climate                               18
...
Deathrate                              2
Agriculture                            0
Industry                               1
Service                                1
dtype: int64
```



```python
# https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in
# Kayıp verileri gruptaki veilerin ortalaması ile doldurma.
df["Climate"] = df["Climate"].fillna(df.groupby("Region")["Climate"].transform("mean"))
# Burası Çokomelli
```

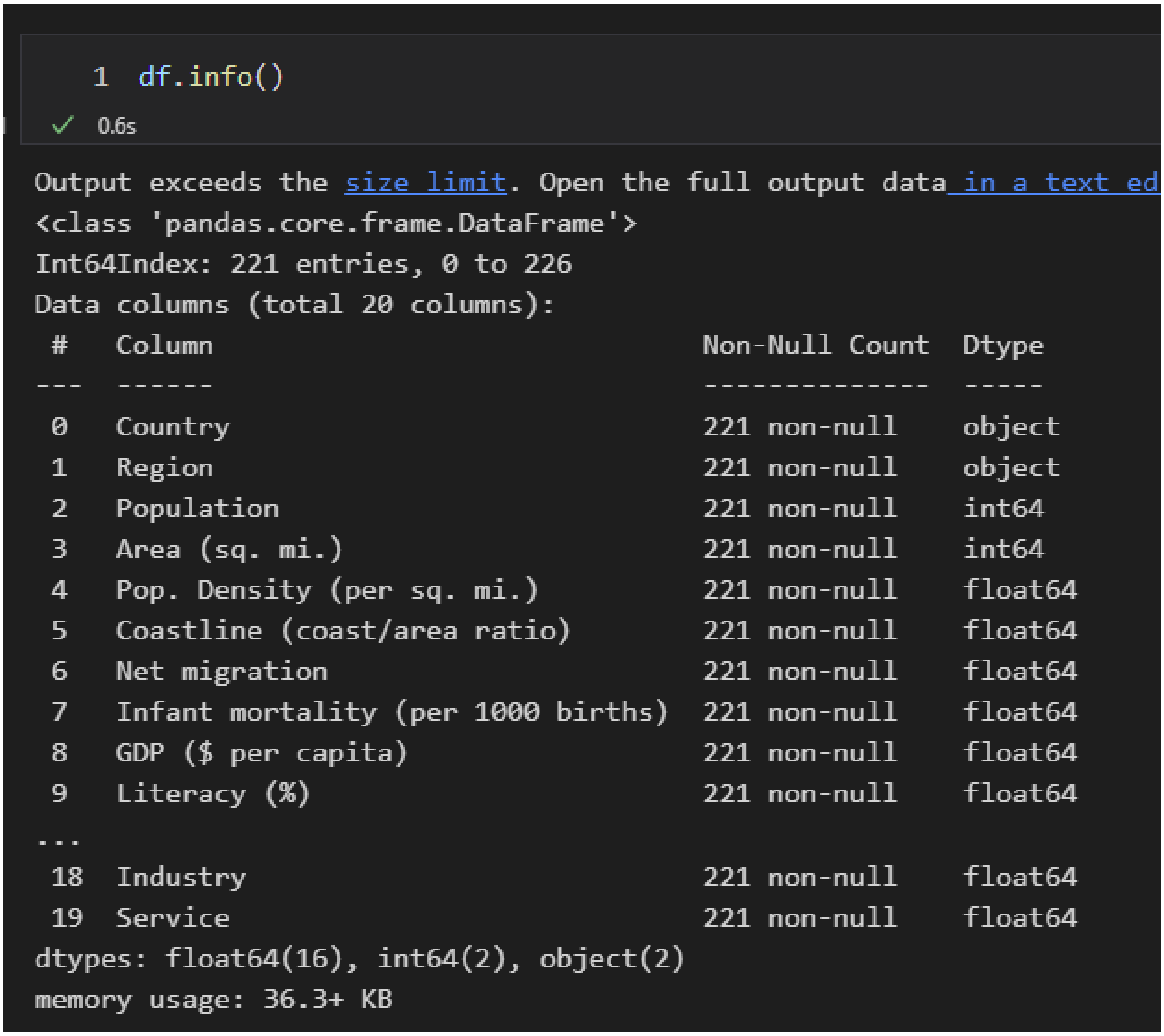
```python
df.isnull().sum()
```

```
Output exceeds the size limit. Open the full output data in a text editor
Country                                0
Region                                 0
Population                             0
Area (sq. mi.)                         0
Pop. Density (per sq. mi.)             0
Coastline (coast/area ratio)           0
Net migration                          1
Infant mortality (per 1000 births)     1
GDP ($ per capita)                     0
Literacy (%)                          13
Phones (per 1000)                      2
Arable (%)                             1
Crops (%)                              1
Other (%)                              1
Climate                                0
...
Deathrate                              2
Agriculture                            0
Industry                               1
Service                                1
dtype: int64
```



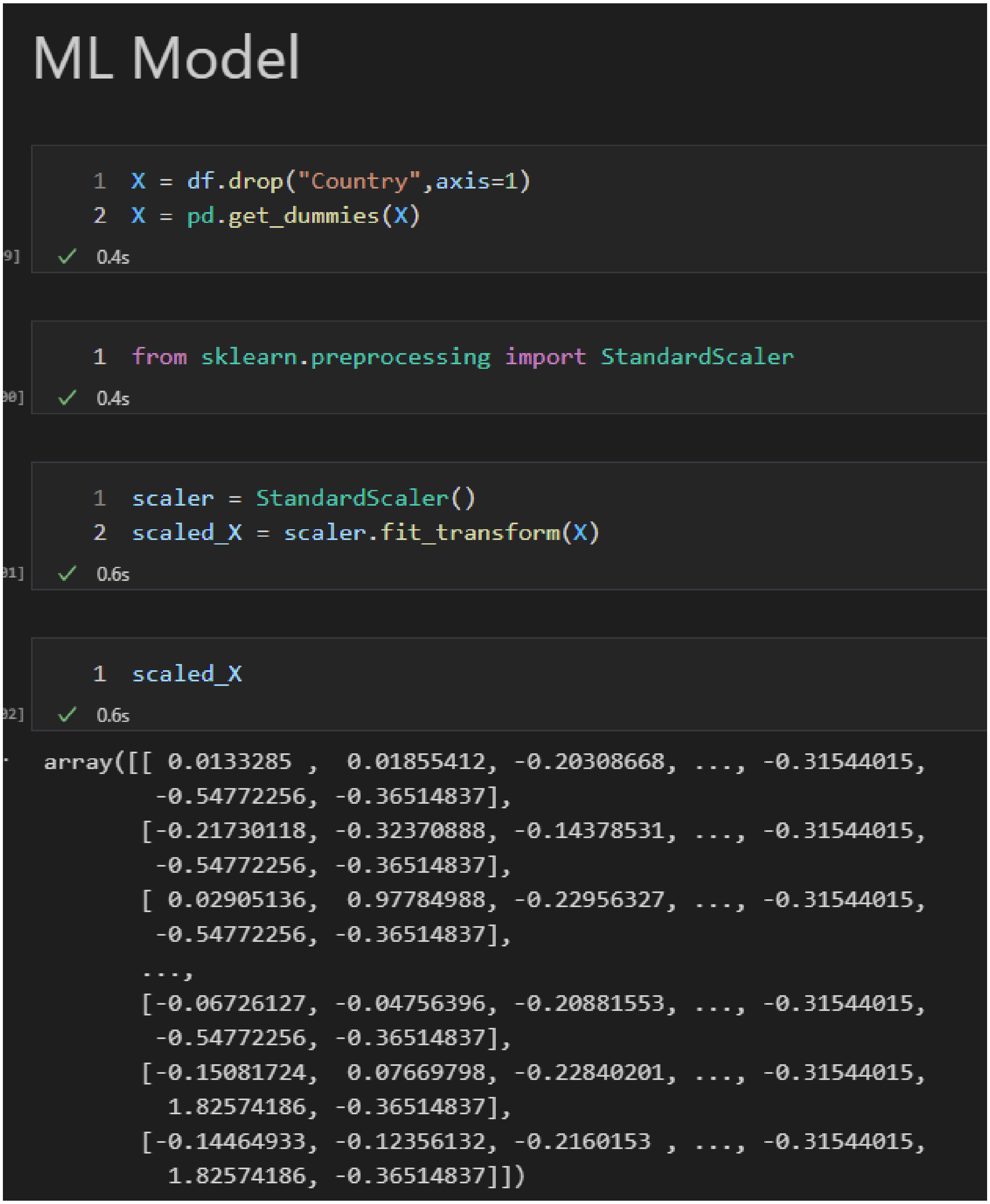
```python
df["Literacy (%)"] = df["Literacy (%)"].fillna(df.groupby("Region")["Literacy (%)"].transform("mean"))
```

```python
df = df.dropna()
```



```
df.info()
```

```
Output exceeds the size limit. Open the full output data in a text ed
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221 entries, 0 to 226
Data columns (total 20 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Country                            221 non-null    object 
 1   Region                             221 non-null    object 
 2   Population                         221 non-null    int64  
 3   Area (sq. mi.)                     221 non-null    int64  
 4   Pop. Density (per sq. mi.)         221 non-null    float64
 5   Coastline (coast/area ratio)       221 non-null    float64
 6   Net migration                      221 non-null    float64
 7   Infant mortality (per 1000 births) 221 non-null    float64
 8   GDP ($ per capita)                 221 non-null    float64
 9   Literacy (%)                       221 non-null    float64
...
 18  Industry                           221 non-null    float64
 19  Service                            221 non-null    float64
dtypes: float64(16), int64(2), object(2)
memory usage: 36.3+ KB
```



ML Model

```python
X = df.drop("Country",axis=1)
X = pd.get_dummies(X)
```

✓ 0.4s

```python
from sklearn.preprocessing import StandardScaler
```

✓ 0.4s

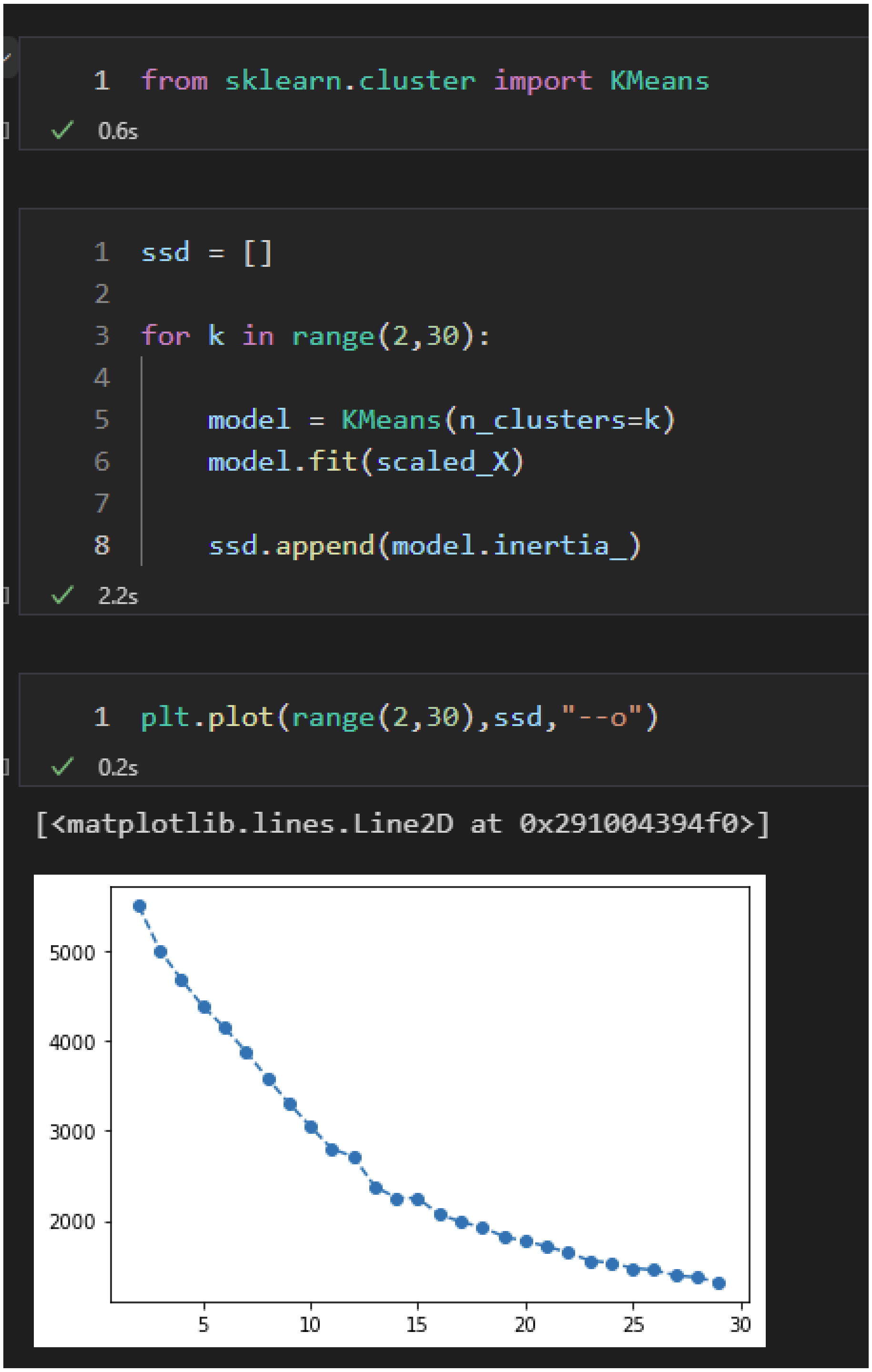
```python
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

✓ 0.6s

```python
scaled_X
```

✓ 0.6s

```
array([[ 0.0133285 ,  0.01855412, -0.20308668, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.21730118, -0.32370888, -0.14378531, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [ 0.02905136,  0.97784988, -0.22956327, ..., -0.31544015,
        -0.54772256, -0.36514837],
       ...,
       [-0.06726127, -0.04756396, -0.20881553, ..., -0.31544015,
        -0.54772256, -0.36514837],
       [-0.15081724,  0.07669798, -0.22840201, ..., -0.31544015,
         1.82574186, -0.36514837],
       [-0.14464933, -0.12356132, -0.2160153 , ..., -0.31544015,
         1.82574186, -0.36514837]])
```



```
from sklearn.cluster import KMeans
```

✓ 0.6s

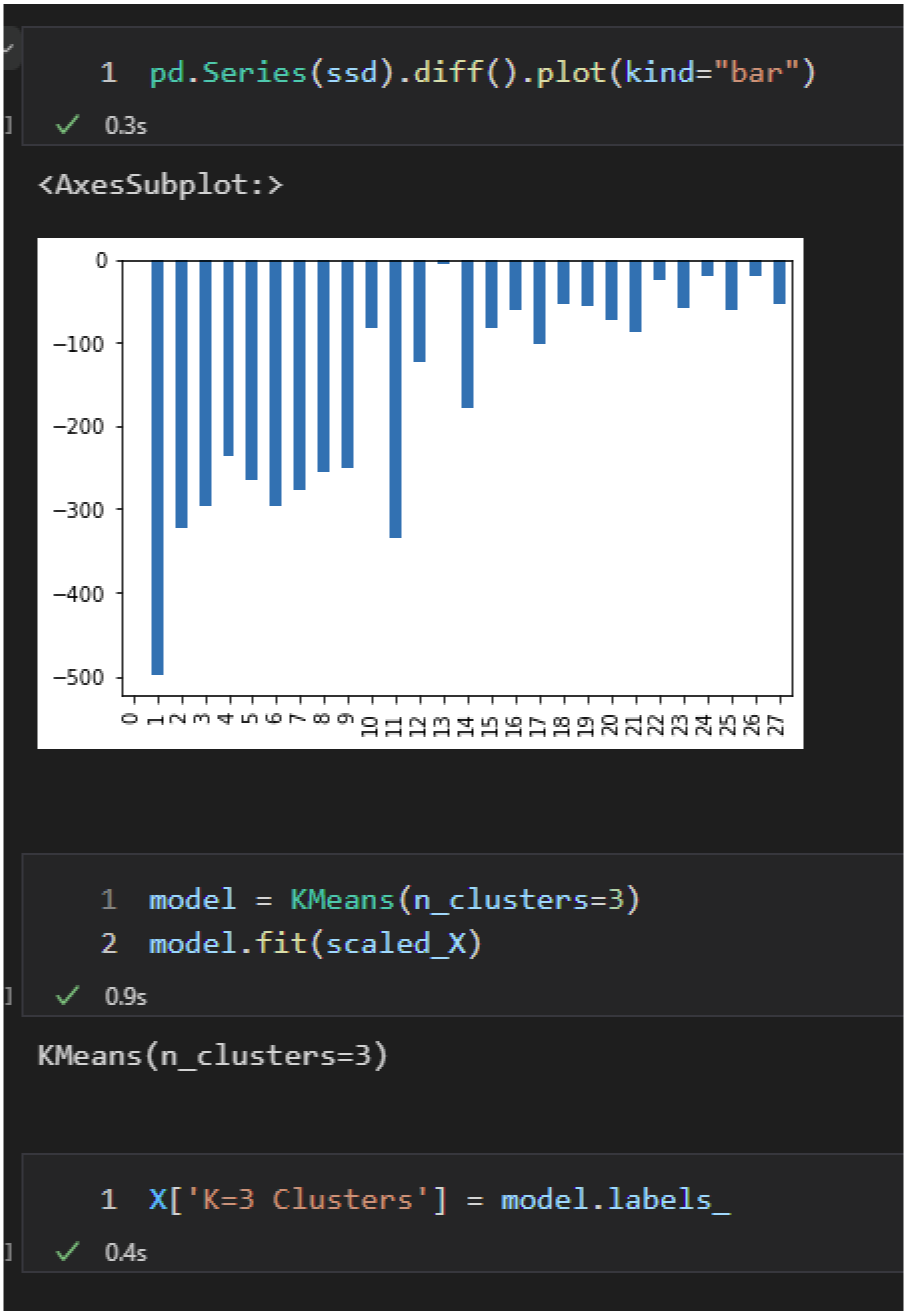
```
ssd = []

for k in range(2,30):

    model = KMeans(n_clusters=k)
    model.fit(scaled_X)

    ssd.append(model.inertia_)
```

✓ 2.2s

```
plt.plot(range(2,30),ssd,"--o")
```

✓ 0.2s

```
[<matplotlib.lines.Line2D at 0x291004394f0>]
```



```
pd.Series(ssd).diff().plot(kind="bar")
```

✓ 0.3s

```
<AxesSubplot:>
```

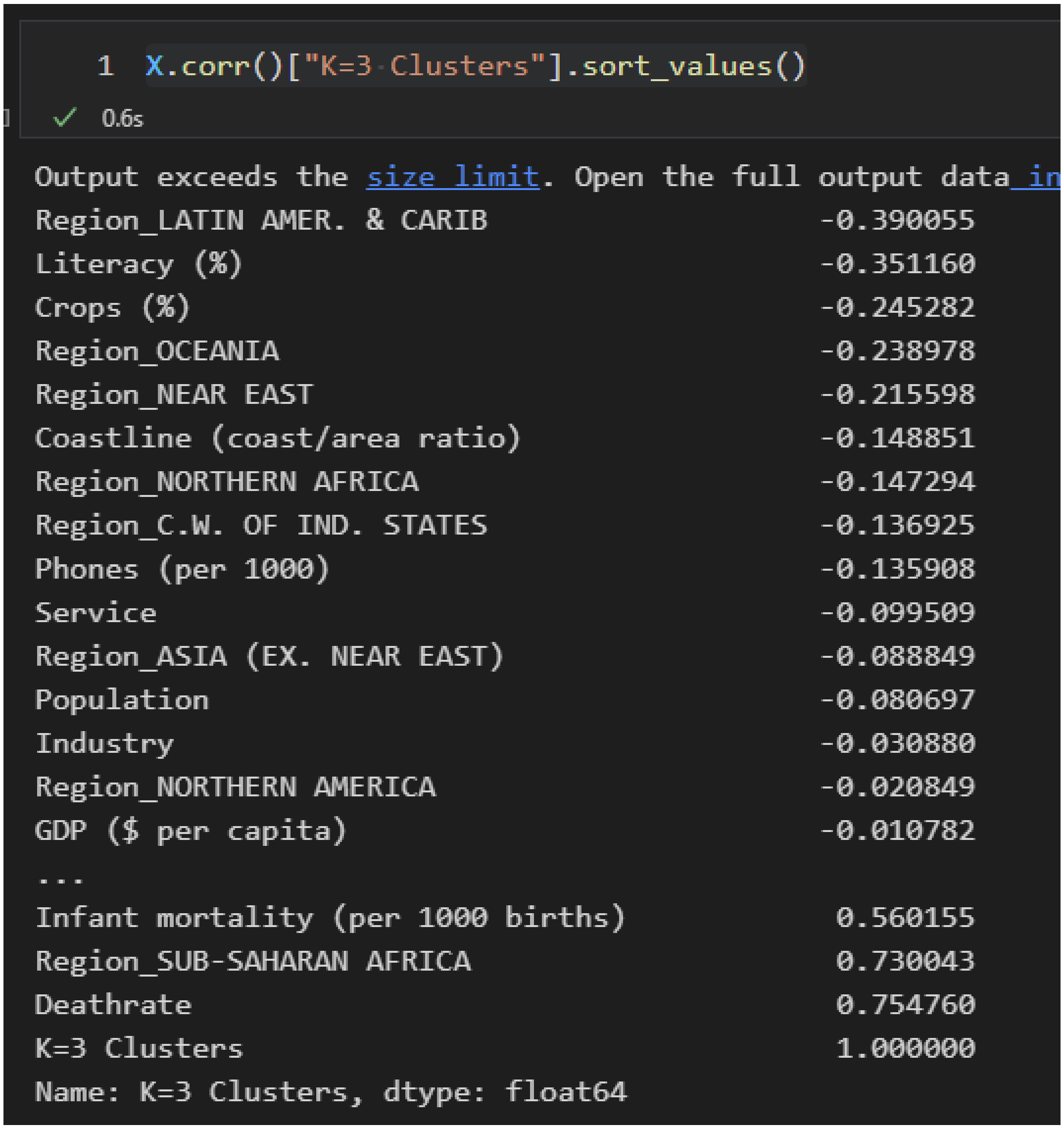
```
model = KMeans(n_clusters=3)
model.fit(scaled_X)
```

✓ 0.9s

```
KMeans(n_clusters=3)
```

```
X['K=3 Clusters'] = model.labels_
```

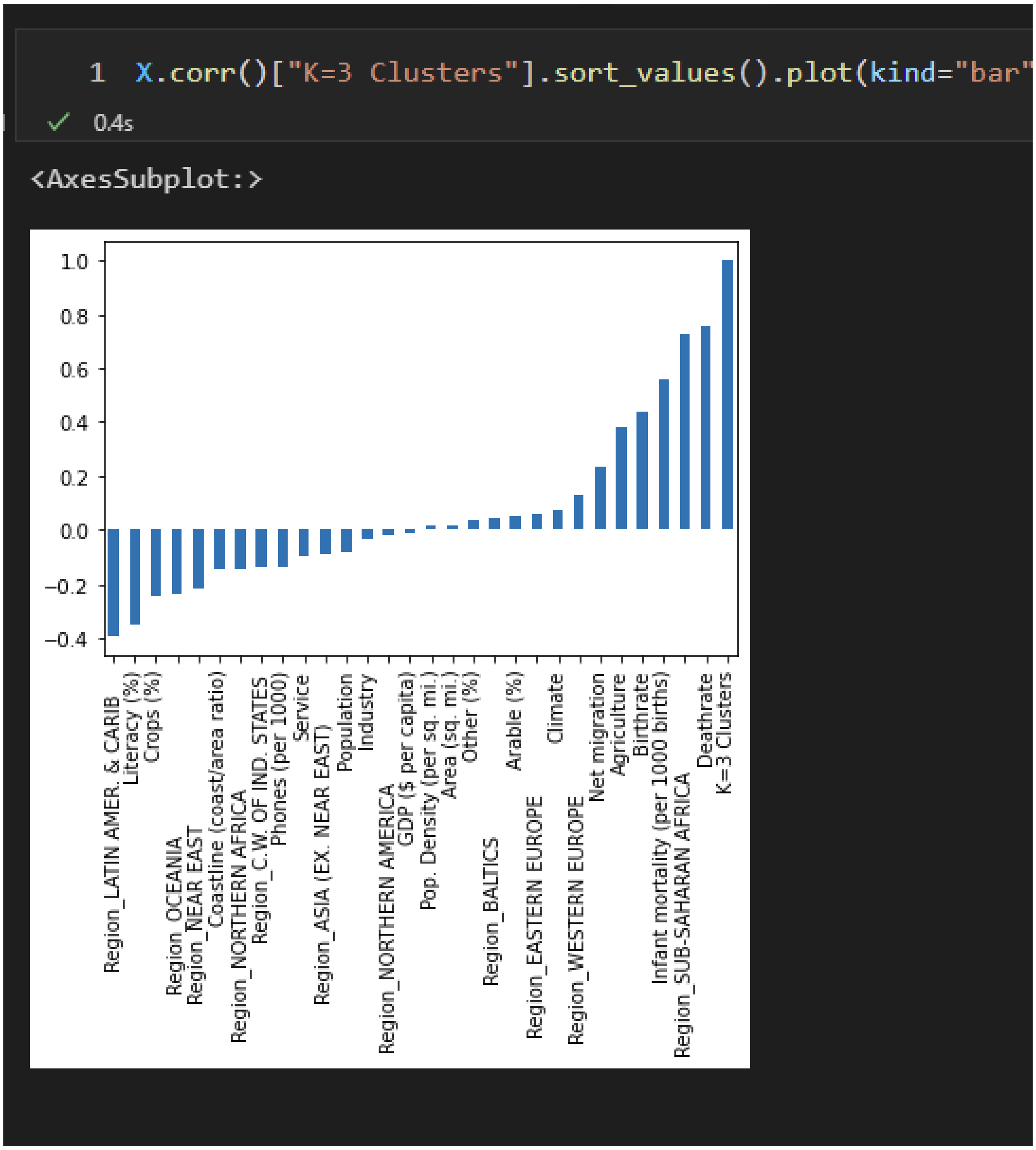
✓ 0.4s



```
X.corr()["K=3 Clusters"].sort_values()
```

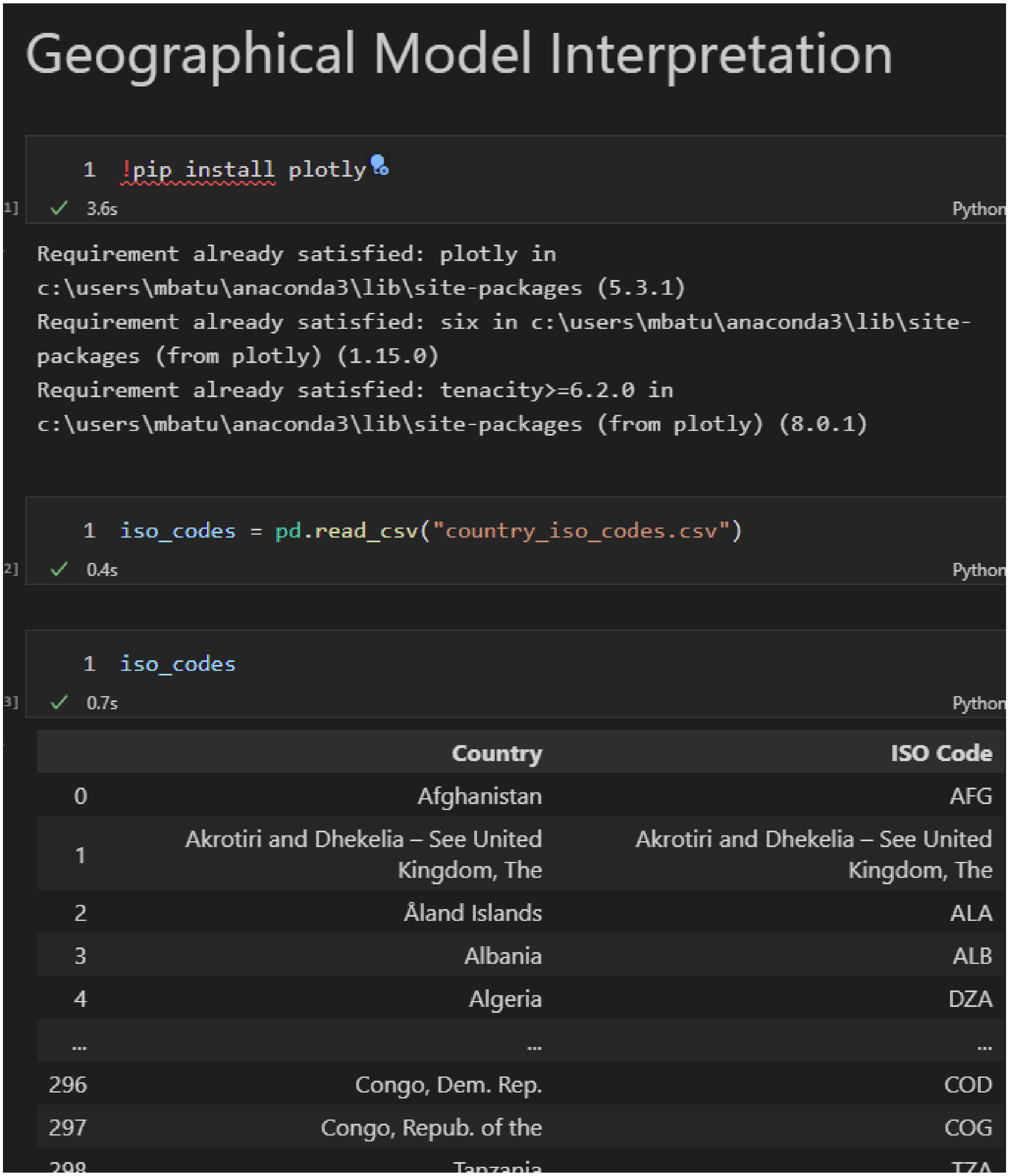
0.6s

```
Output exceeds the size limit. Open the full output data in
Region_LATIN AMER. & CARIB                  -0.390055
Literacy (%)                                -0.351160
Crops (%)                                   -0.245282
Region_OCEANIA                              -0.238978
Region_NEAR EAST                            -0.215598
Coastline (coast/area ratio)                -0.148851
Region_NORTHERN AFRICA                      -0.147294
Region_C.W. OF IND. STATES                  -0.136925
Phones (per 1000)                           -0.135908
Service                                     -0.099509
Region_ASIA (EX. NEAR EAST)                 -0.088849
Population                                  -0.080697
Industry                                    -0.030880
Region_NORTHERN AMERICA                     -0.020849
GDP ($ per capita)                          -0.010782
...
Infant mortality (per 1000 births)           0.560155
Region_SUB-SAHARAN AFRICA                    0.730043
Deathrate                                    0.754760
K=3 Clusters                                 1.000000
Name: K=3 Clusters, dtype: float64
```



```
X.corr()["K=3 Clusters"].sort_values().plot(kind="bar"
✓ 0.4s
```

```
<AxesSubplot:>
```



Geographical Model Interpretation

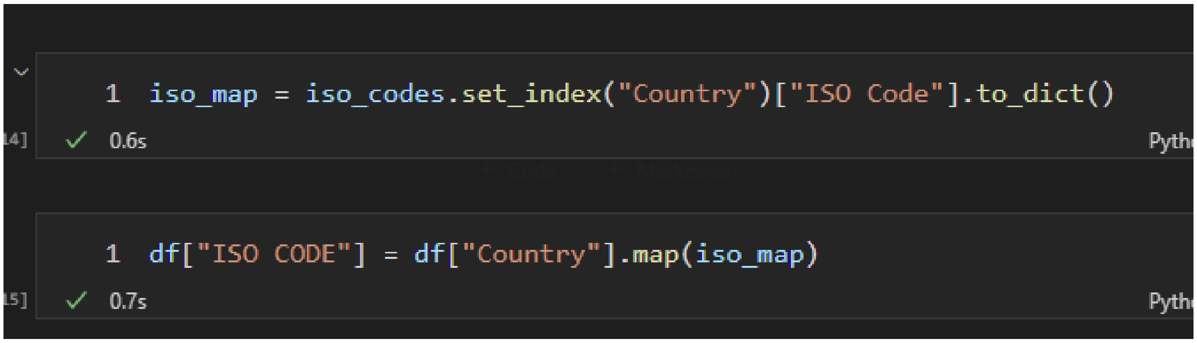
```
!pip install plotly
```

```
Requirement already satisfied: plotly in
c:\users\mbatu\anaconda3\lib\site-packages (5.3.1)
Requirement already satisfied: six in c:\users\mbatu\anaconda3\lib\site-
packages (from plotly) (1.15.0)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\mbatu\anaconda3\lib\site-packages (from plotly) (8.0.1)
```

```
iso_codes = pd.read_csv("country_iso_codes.csv")
```

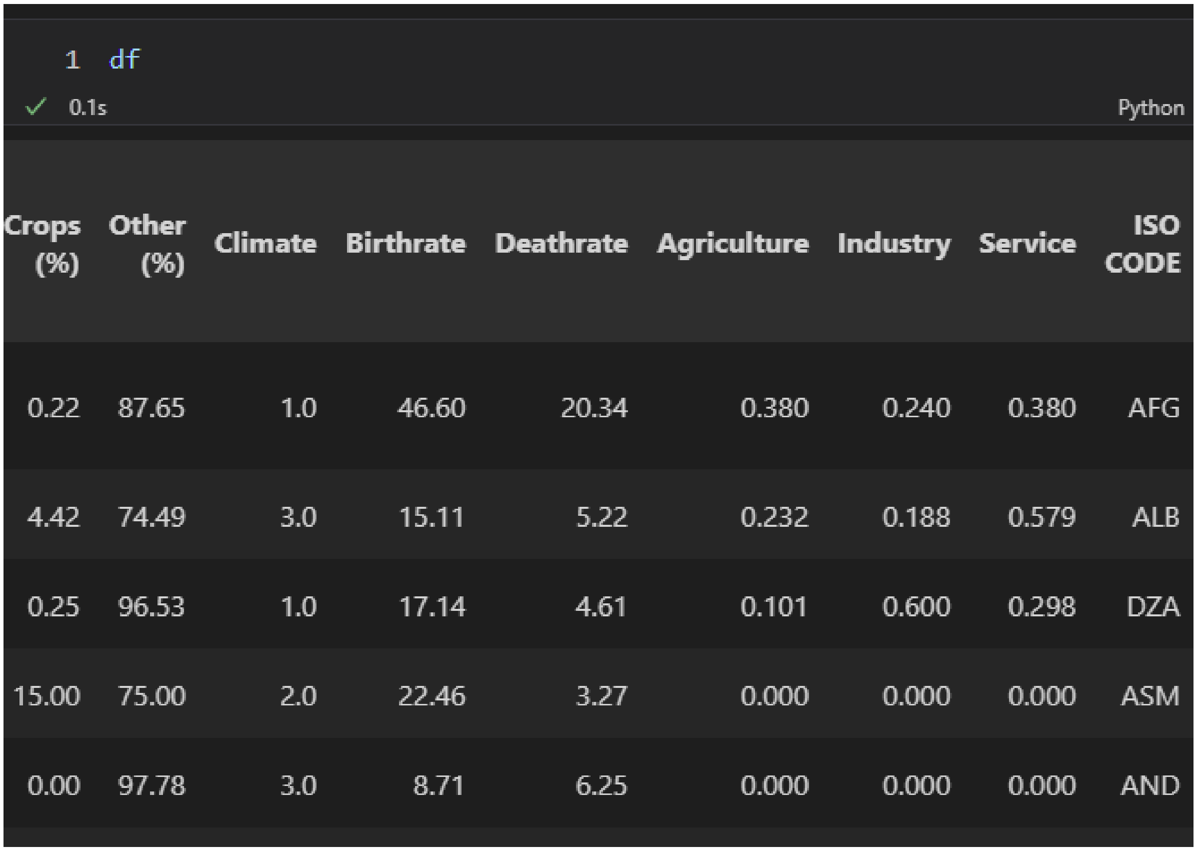
```
iso_codes
```

	Country	ISO Code
0	Afghanistan	AFG
1	Akrotiri and Dhekelia – See United Kingdom, The	Akrotiri and Dhekelia – See United Kingdom, The
2	Åland Islands	ALA
3	Albania	ALB
4	Algeria	DZA
...	...	...
296	Congo, Dem. Rep.	COD
297	Congo, Repub. of the	COG
298	Tanzania	TZA



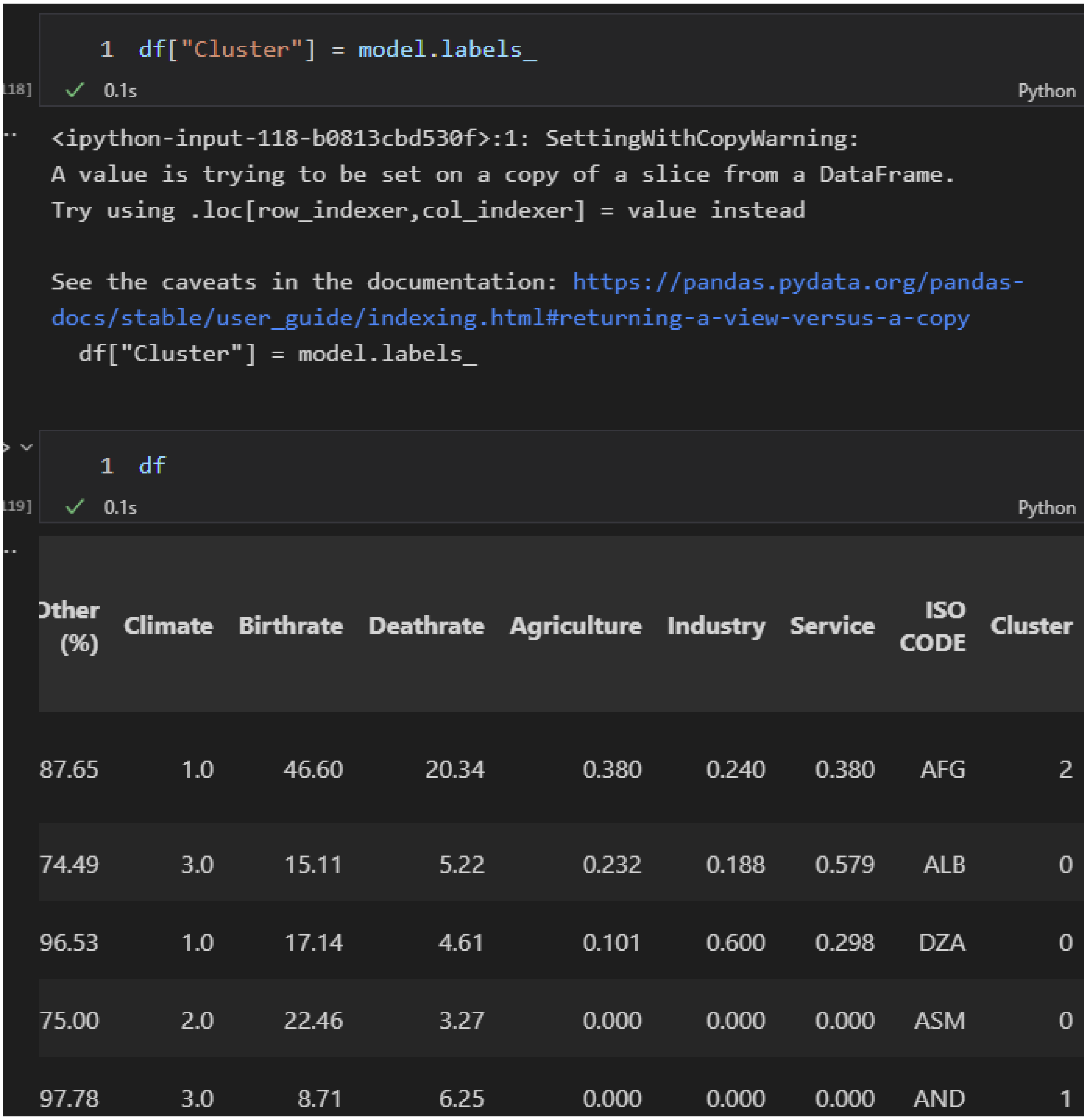
```python
iso_map = iso_codes.set_index("Country")["ISO Code"].to_dict()
```

```python
df["ISO CODE"] = df["Country"].map(iso_map)
```



```python
df
```

Crops (%)	Other (%)	Climate	Birthrate	Deathrate	Agriculture	Industry	Service	ISO CODE
0.22	87.65	1.0	46.60	20.34	0.380	0.240	0.380	AFG
4.42	74.49	3.0	15.11	5.22	0.232	0.188	0.579	ALB
0.25	96.53	1.0	17.14	4.61	0.101	0.600	0.298	DZA
15.00	75.00	2.0	22.46	3.27	0.000	0.000	0.000	ASM
0.00	97.78	3.0	8.71	6.25	0.000	0.000	0.000	AND

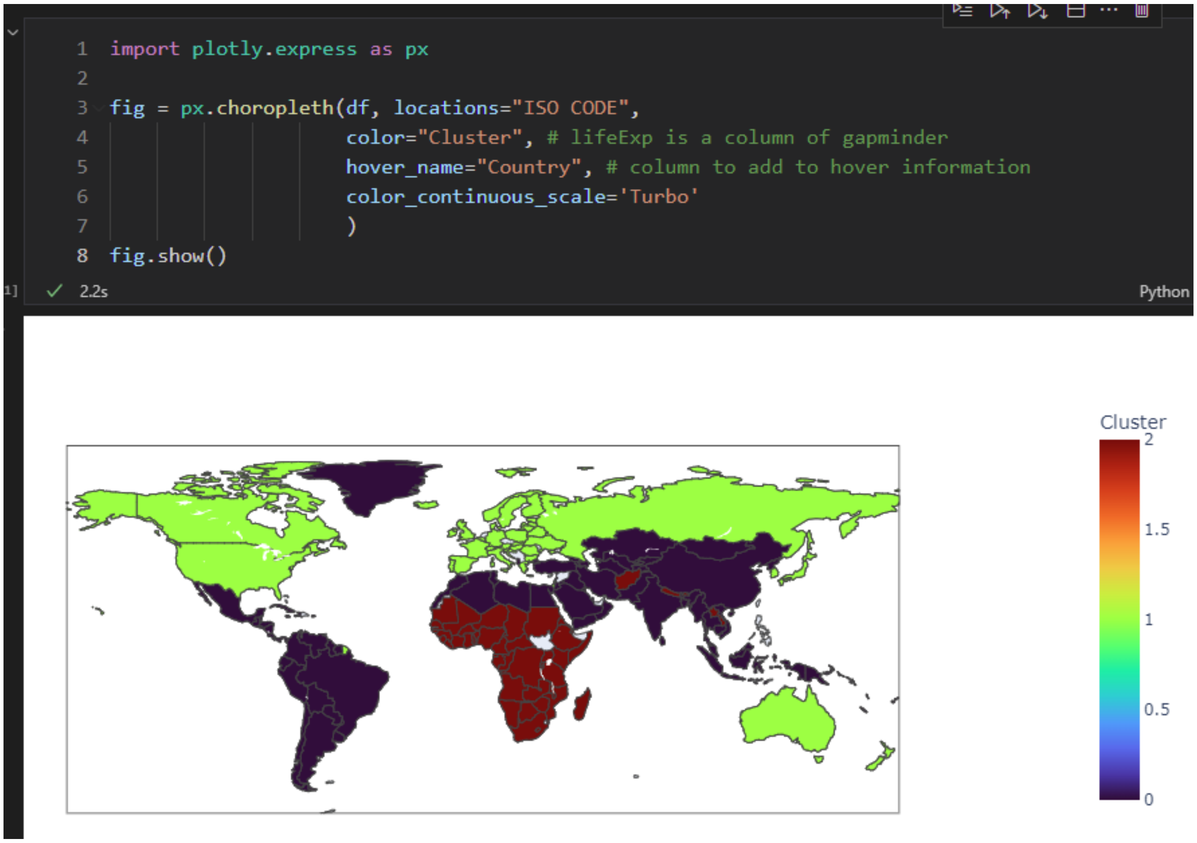


```python
df["Cluster"] = model.labels_
```

```
<ipython-input-118-b0813cbd530f>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["Cluster"] = model.labels_
```

```python
df
```

Other (%)	Climate	Birthrate	Deathrate	Agriculture	Industry	Service	ISO CODE	Cluster
87.65	1.0	46.60	20.34	0.380	0.240	0.380	AFG	2
74.49	3.0	15.11	5.22	0.232	0.188	0.579	ALB	0
96.53	1.0	17.14	4.61	0.101	0.600	0.298	DZA	0
75.00	2.0	22.46	3.27	0.000	0.000	0.000	ASM	0
97.78	3.0	8.71	6.25	0.000	0.000	0.000	AND	1



```python
import plotly.express as px

fig = px.choropleth(df, locations="ISO CODE",
                    color="Cluster", # lifeExp is a column of gapminder
                    hover_name="Country", # column to add to hover information
                    color_continuous_scale='Turbo'
                    )
fig.show()
```