

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	...	h
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	...	201.000000
mean	0.840796	122.00000	98.797015	0.837102	0.915126	0.899108	2555.666667	126.875622	3.330692	3.256874	...	101.000000
std	1.254802	31.99625	6.066366	0.059213	0.029187	0.040933	517.296727	41.546834	0.268072	0.316048	...	101.000000
min	-2.000000	65.00000	86.600000	0.678039	0.837500	0.799331	1488.000000	61.000000	2.540000	2.070000	...	101.000000
25%	0.000000	101.00000	94.500000	0.801538	0.890278	0.869565	2169.000000	98.000000	3.150000	3.110000	...	101.000000
50%	1.000000	122.00000	97.000000	0.832292	0.909722	0.904682	2414.000000	120.000000	3.310000	3.290000	...	101.000000
75%	2.000000	137.00000	102.400000	0.881788	0.925000	0.928094	2926.000000	141.000000	3.580000	3.410000	...	101.000000
max	3.000000	256.00000	120.900000	1.000000	1.000000	1.000000	4066.000000	326.000000	3.940000	4.170000	...	201.000000
8 rows × 21 columns												



### Збереження даних

Pandas дозволяє нам зберігати набір даних у форматі CSV. Використовуючи метод `dataframe.to_csv()`, ви можете додати шлях до файлу та назву разом із лапками в дужках.

Наприклад, якщо ви хочете зберегти фрейм даних "df" як "auto1.csv", можете використати наведений нижче синтаксис, де `index = False` означає, що назви рядків не будуть записані.

```
df.to_csv("auto1.csv", index=False)
```

### Read/Save інших форматів даних

Data Formate	Read	Save
csv	<code>pd.read_csv()</code>	<code>df.to_csv()</code>
json	<code>pd.read_json()</code>	<code>df.to_json()</code>
excel	<code>pd.read_excel()</code>	<code>df.to_excel()</code>
hdf	<code>pd.read_hdf()</code>	<code>df.to_hdf()</code>
sql	<code>pd.read_sql()</code>	<code>df.to_sql()</code>
...	...	...

Якщо використовуєте Google Colaboratory, але хочете зберегти файл з даними на локальний комп'ютер, скористайтесь наступною командою

```
from google.colab import files

files.download('auto1.csv')
```

В теоретичній частині роботи використано елементи курсу "Data Analysis with Python" від IBM Corporation, автор [Joseph Santarcangelo](#)



### Завдання, що оцінюються

1. Скачати дані із файлу ['Data2.csv'](#). Записати дані у dataframe. Дослідити структуру даних.
2. Виправити помилки в даних.
3. Заповнити пропуски.
4. Додати стовпчик із щільністю населення.
5. Побудувати діаграми розмаху та гістограми.



### Завдання #1:

Дослідити структуру даних



Зчитую дані з файлу у датафрейм

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
DATA_PATH = "/content/drive/My Drive/data/Data2.csv"
df = pd.read_csv(DATA_PATH, sep=';', encoding='cp1252')
```

► Натисніть тут, щоб побачити підказку

Досліджую структуру даних

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
#df.describe(include = "all")
#df.info()
#df.dtypes
df.head()
```

	Country Name	Region	GDP per capita	Populatiion	CO2 emission	Area	
0	Afghanistan	South Asia	561.778746	34656032.0	9809.225	652860.0	
1	Albania	Europe & Central Asia	4124.982390	2876101.0	5716.853	28750.0	
2	Algeria	Middle East & North Africa	3916.881571	40606052.0	145400.217	2381740.0	
3	American Samoa	East Asia & Pacific	11834.745230	55599.0	NaN	200.0	
4	Andorra	Europe & Central Asia	36988.622030	77281.0	462.042	470.0	

Next steps: [View recommended plots](#)

Бачу наступні проблеми в даних:

- 1. У деяких числах коми замість крапок
- 2. Неправильний тип даних у деяких стовпцях
- 3. Від'ємні значення в стовпцях
- 4. Є пропущені значення в ознаках таких-то



Завдання #2:

Виправити помилки в даних

Проблема 1.У деяких числах коми замість крапок

Для виправлення визначу в яких стовпцях таке відбувається та заміню всі коми на крапки

Двічі натисніть клітинку, щоб змінити її (або натисніть клавішу Enter)

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
numeric_columns = ['GDP per capita', 'CO2 emission', 'Area']
for column in numeric_columns:
    df[column] = df[column].str.replace(',', '.')

df.head()
```

Проблема 2. Неправильний тип даних у деяких стовпцях

Для виправлення заміню тим даних

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
df['Country Name'] = df['Country Name'].astype('string')
df['Region'] = df['Region'].astype('string')
df['Area'] = df['Area'].astype(int)
df['GDP per capita'] = df['GDP per capita'].astype(float)
df['CO2 emission'] = df['CO2 emission'].astype(float)
df['Populatiion'] = df['Populatiion'].astype(int)

df.dtypes
```

```
Country Name      string
Region            string
GDP per capita    float64
Populatiion       int64
CO2 emission      float64
Area              int64
dtype: object
```

Проблема 3. Від'ємні значення в стовпцях

Для виправлення заміню від'ємні значення в стовпцях Area та GDP per capita на середнє

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
mean_GDP = df['GDP per capita'].mean()
df.loc[df['GDP per capita'] < 0, 'GDP per capita'] = mean_GDP

mean_Area = df['Area'].mean()
df.loc[df['Area'] < 0, 'Area'] = mean_Area

df.describe(include = "all")
```

	Country Name	Region	GDP per capita	Populatiion	CO2 emission	Area
count	217	217	190.000000	2.160000e+02	2.050000e+02	2.170000e+02
unique	217	7	NaN	NaN	NaN	NaN
top	Afghanistan	Europe & Central Asia	NaN	NaN	NaN	NaN
freq	1	58	NaN	NaN	NaN	NaN
mean	NaN	NaN	13480.607151	3.432256e+07	1.651141e+05	6.185492e+05
std	NaN	NaN	18032.317511	1.347600e+08	8.335357e+05	1.827826e+06
min	NaN	NaN	285.727442	1.109700e+04	1.100100e+01	2.000000e+00
25%	NaN	NaN	2031.779671	7.900265e+05	1.334788e+03	1.088700e+04
50%	NaN	NaN	5235.308547	6.221590e+06	9.108828e+03	9.303000e+04
75%	NaN	NaN	16003.299818	2.350337e+07	5.986378e+04	4.474200e+05
max	NaN	NaN	100738.684200	1.378665e+09	1.029193e+07	1.709825e+07



Завдання #3:

Заповнити пропуски

Заповнювати пропуски для ознаки такої-то буду таким-то способом, тому що ...

- 1. GDP per capita: 27 відсутніх значень, заміню середнім значенням
- 2. Populatiion: 1 відсутнє значення, заміню середнім значенням
- 3. CO2 emission: 12 відсутніх значень, я їх видалю, бо це те що ми хочемо передбачити в залежності від кількості населення

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
import numpy as np

missing_data = df.isnull()
avg_GDP = df['GDP per capita'].astype('float').mean(axis=0)
df.replace({'GDP per capita':np.nan}, avg_GDP, inplace=True)

missing_data = df.isnull()
avg_Pop = df['Populatiion'].astype('int').mean(axis=0)
df.replace({'Populatiion':np.nan}, avg_Pop, inplace=True)

missing_data = df.isnull()
df.dropna(subset=['CO2 emission'], axis=0, inplace=True)
df.reset_index(drop=True, inplace=True)

df.head()
```

	Country Name	Region	GDP per capita	Populatiion	CO2 emission	Area	
0	Afghanistan	South Asia	561.778746	34656032	9809.225	652860	
1	Albania	Europe & Central Asia	4124.982390	2876101	5716.853	28750	
2	Algeria	Middle East & North Africa	3916.881571	40606052	145400.217	2381740	
3	Andorra	Europe & Central Asia	36988.622030	77281	462.042	470	
4	Angola	Sub-Saharan Africa	3308.700233	28813463	34763.160	1246700	



Next steps: [View recommended plots](#)

Досліджую структуру даних, чи всі пропуски заповнено

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
#df.info()
missing_data = df.isnull()
#missing_data.head(5)

for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")

Country Name
False    205
Name: Country Name, dtype: int64

Region
False    205
Name: Region, dtype: int64

GDP per capita
False    205
Name: GDP per capita, dtype: int64

Populatiion
False    205
Name: Populatiion, dtype: int64

CO2 emission
False    205
Name: CO2 emission, dtype: int64

Area
False    205
Name: Area, dtype: int64
```



### Завдання #4:

Додати стовпчик із щільністю населення

Щільність населення розрахую по формулі кількість населення поділена на площу. і додам у стовпчик Population Density.

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
df['Population Density'] = df['Populatiion'] / df['Area']
df.head()
```

	Country Name	Region	GDP per capita	Populatiion	CO2 emission	Area	Population Density
0	Afghanistan	South Asia	561.778746	34656032	9809.225	652860	53.083405
1	Albania	Europe & Central Asia	4124.982390	2876101	5716.853	28750	100.038296
2	Algeria	Middle East & North Africa	3916.881571	40606052	145400.217	2381740	17.048902
3	Andorra	Europe & Central Asia	36988.622030	77281	462.042	470	164.427660
4	Angola	Sub-Saharan Africa	3308.700233	28813463	34763.160	1246700	23.111786

Next steps:





### Завдання #5:

Побудувати діаграми розмаху та гістограми

Для побудови графіків скористайтесь бібліотекою Matplotlib. Спробуйте погратись з кольорами, розмірами та підписами.

```
# Напишіть ваш код нижче та натисніть Shift+Enter для виконання
import matplotlib.pyplot as plt

fig, axs = plt.subplots(2, 2, figsize=(12, 8))

fig.suptitle('Діаграми розмаху', fontsize=16)

axs[0, 0].set_title('GDP per capita')
axs[0, 0].boxplot(df['GDP per capita'])

axs[0, 1].set_title('Population')
axs[0, 1].boxplot(df['Populatiion'])

axs[1, 0].set_title('CO2 emission')
axs[1, 0].boxplot(df['CO2 emission'])

axs[1, 1].set_title('Area')
axs[1, 1].boxplot(df['Area'])

plt.show()

sorted_df = df.sort_values(by='Population Density', ascending=False)
top_10_density_countries = sorted_df.head(10)

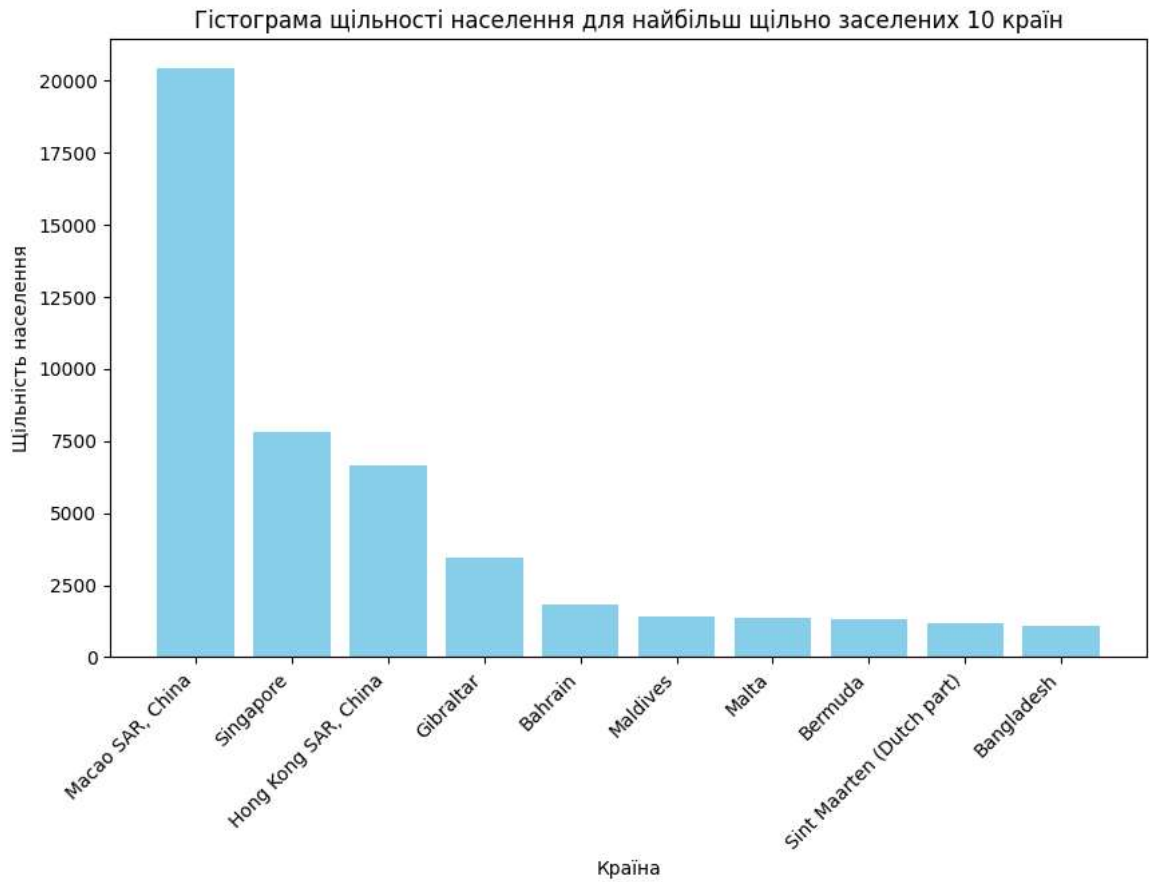
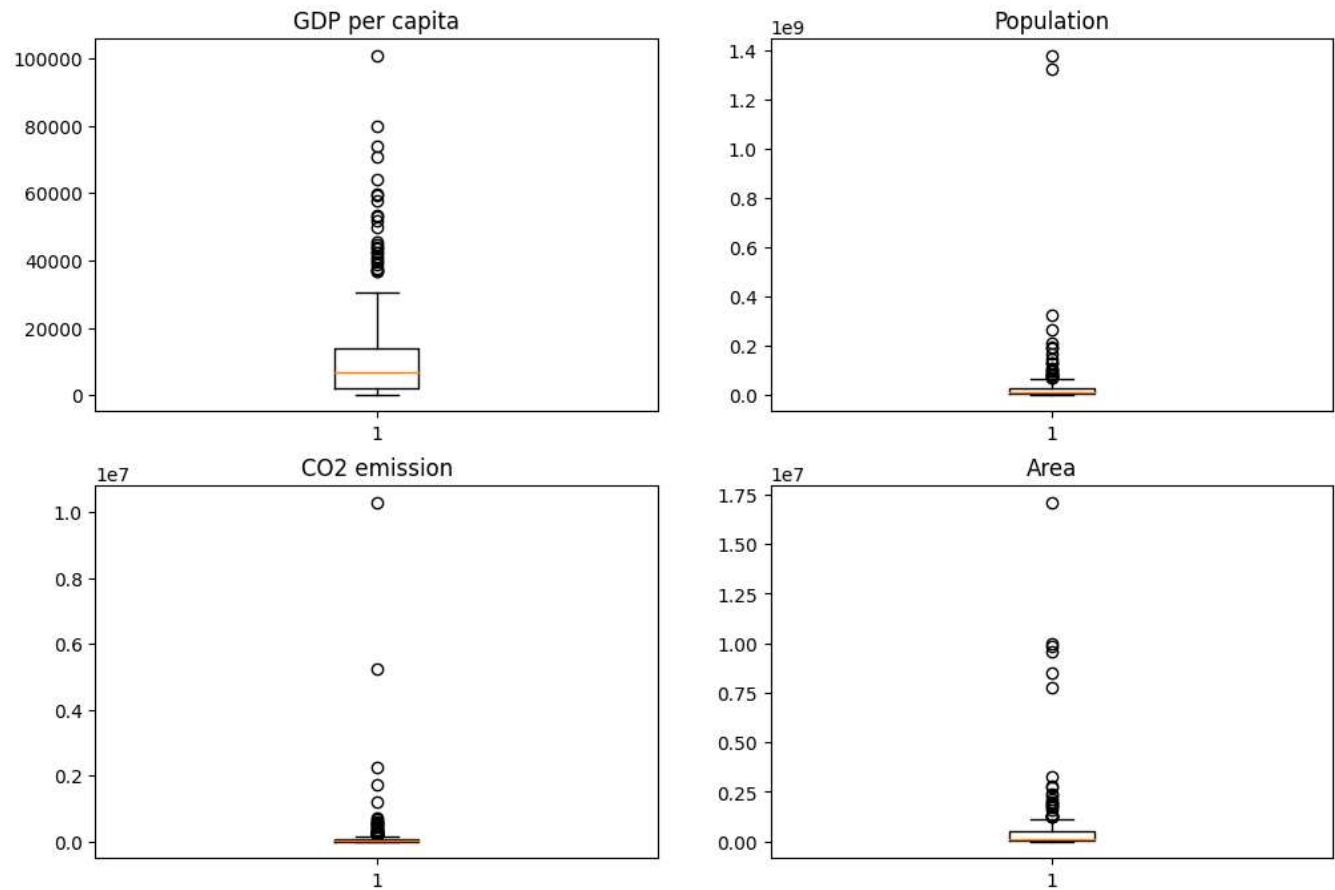
plt.figure(figsize=(10, 6))
plt.bar(top_10_density_countries['Country Name'], top_10_density_countries['Population Density'], color='skyblue')

plt.title('Гістограма щільності населення для найбільш щільно заселених 10 країн')
plt.xlabel('Країна')
plt.ylabel('Щільність населення')

plt.xticks(rotation=45, ha='right')

plt.show()
```

Діаграми розмаху



► Натисніть тут, щоб побачити підказку



## Додаткове завдання:

Дайте відповіді на питання

1. Яка країна має найбільший ВВП на людину (GDP per capita)?
2. Яка країна має найменшу площу?
3. Знайдіть країну з найбільшою щільністю населення у світі? У Європі та центральній Азії?
4. Покажіть топ 5 країн та 5 останніх країн по ВВП на людину.

▼ Натисніть тут, щоб побачити підказку

Скористайтесь методами `loc()` для повернення зрізу датафрейму, `idxmax()` для повернення номера рядка з найбільшим значенням якогось показника у стовпці та `idxmin()` для повернення номера рядка з найменшим значенням якогось показника у стовпці

```
df.loc[df['GDP per capita'].idxmax()]
```

Гарно оформити виведення інформації допоможе `print()`

```
print('\n' + df.loc[df['GDP per capita'].idxmax(), 'Country Name'] + ' має найбільший ВВП на людину')
```

# Напишіть ваш код нижче та натисніть Shift+Enter для виконання

#Яка країна має найбільший ВВП на людину (GDP per capita)?

```
print('\n' + df.loc[df['GDP per capita'].idxmax(), 'Country Name'] + ' має найбільший ВВП на людину')
```

#Яка країна має найменшу площу?