Haz doble clic (o ingresa) para editar

A1 Activities

```
import pandas as pd
```

A1.1. Load the iris.csv file in your computer and understand the dataset

```
iris_url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Cargar el dataset directamente desde la URL
df = pd.read_csv(iris_url, header=None)

# Definir los nombres de las columnas
attributes = ["sepal_length", "sepal_width", "petal_length", "petal_width", "species"]
df.columns = attributes
```

Check if the dataset was loaded correctly

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

A1.2. How many observations (rows) are in total?

```
row= df.shape[0]
print('Total of observations(rows): '+ str(row))
```

```
Total of observations(rows): 150
```

A1.3. How many variables (columns) are in total? What do they represent?

```
column= df.shape[1]
print('Total of variables(columns): '+ str(column))
```

```
Total of variables(columns): 5
```

```
print('Each column/variable represents:' )
for column in df.columns:
    print(column)
```

```
Each column/variable represents:
sepal_length
sepal_width
petal_length
petal_width
species
```

A1.4. How many observations are for each type of flower?

```
flower_counts = df['species'].value_counts()
print(flower_counts)
```

```
species
Iris-setosa        50
Iris-versicolor    50
```

```
Iris-virginica     50
Name: count, dtype: int64
```

A1.5. What is the type of data for each variable?

```
df.dtypes
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

A1.6. What are the units of each variable?

All of the measurements (length and width) are given in centimeters. The 'species' variable is a label and does not have a unit.

A2 Activities

1. Calculate the statistical summary for each quantitative variables. Explain the results
    - Identify the name of each column
    - Identify the type of each column
    - Minimum, maximum, mean, average, median, standar deviation

```
print("Name of each column ")
name = df.columns
print(name)
```

```
Name of each column
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
print("Type of each column ")
tipo = df.dtypes
print(tipo)
```

```
Type of each column
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

```
print("Maximum, minimum, mean, average, median, standar deviation ")
df.describe()
```

```
Maximum, minimum, mean, average, median, standar deviation
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

We can conclude several things from this description of the data. Firstly, the data is complete, it is most certainly that there's not a Null space; next is that the mean of sepal length is the one with the most longitude in the flowers. The petal width is the part of the flower that is small on the species.

```
df.species.describe()
```

```
count             150
unique              3
top       Iris-setosa
freq               50
Name: species, dtype: object
```

Theres the types of flowers, and the one with the most Frequency is the 'Iris-setosa'; or atleast thats what the command says, but in reality all the species have the same Frequency, so we should investigate further in the reason for this.

2. Are there missing data? If so, create a new dataset containing only the rows with the non-missing data

```
print("No data is missing")
df.isnull().sum()
```

```
No data is missing
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

3. Create a new dataset containing only the petal width and length and the type of Flower

```
keep = ['petal_length','petal_width','species']
df_petal = df[keep]
df_petal.head()
```

	petal_length	petal_width	species
0	1.4	0.2	Iris-setosa
1	1.4	0.2	Iris-setosa
2	1.3	0.2	Iris-setosa
3	1.5	0.2	Iris-setosa
4	1.4	0.2	Iris-setosa

4. Create a new dataset containing only the sepal width and length and the type of Flower

```
import pandas as pd
from sklearn.datasets import load_iris

# Cargar el conjunto de datos iris
iris = load_iris()

# Crear un DataFrame con los datos
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Agregar la columna de especies
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Seleccionar las columnas correctas
df_sepal = df[['sepal length (cm)', 'sepal width (cm)', 'species']]

# Mostrar los primeros resultados
df_sepal.head()
```

	sepal length (cm)	sepal width (cm)	species
0	5.1	3.5	setosa
1	4.9	3.0	setosa
2	4.7	3.2	setosa
3	4.6	3.1	setosa
4	5.0	3.6	setosa

Próximos pasos: Generar código con df_sepal | Ver gráficos recomendados | New interactive sheet

5. Create a new dataset containing the setal width and length and the type of Flower encoded as a categorical numerical column

```
import pandas as pd
from sklearn.datasets import load_iris

# Cargar el conjunto de datos iris
iris = load_iris()

# Crear un DataFrame con los datos
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Agregar la columna de especies
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Crear una copia del DataFrame
new_df = df[['sepal length (cm)', 'sepal width (cm)', 'species']].copy()

# Usar pd.factorize() para codificar 'species' en valores numéricos
new_df['species_encoded'] = pd.factorize(new_df['species'])[0]

# Mostrar los primeros resultados
new_df.head()
```

	sepal length (cm)	sepal width (cm)	species	species_encoded
0	5.1	3.5	setosa	0
1	4.9	3.0	setosa	0
2	4.7	3.2	setosa	0
3	4.6	3.1	setosa	0
4	5.0	3.6	setosa	0

Próximos pasos: Generar código con new_df | Ver gráficos recomendados | New interactive sheet