TELEDUC
Educación a Distancia

OPERACIONES BÁSICAS DE UN DATA FRAME

>>> Crear Data Frames

Operaciones de un Data Frame

Revisaremos sus características básicas y cómo trabajarlos

Es fundamental conocer un Data Frame antes de ver las herramientas avanzadas de *Pandas*

¿Qué es un Data Frame?

Es la estructura más usada en la librería *Pandas* y podemos imaginarlos como una matriz de datos, donde podemos agregar filas y columnas a nuestro antojo.

xxxxx	xxxxx	xxxxx	xxxxx
xxxxx	xxxxx	xxxxx	xxxxx
xxxxx	xxxxx	xxxxx	xxxxx
xxxxx	xxxxx	xxxxx	xxxxx
xxxxx	xxxxx	xxxxx	xxxxx

Columna: Representa datos para una variable específica. Las columnas de un Data Frame son series, y por eso es importante manejarlas.

Fila: Cada fila corresponde a medidas o valores de cada instancia y podrá tener valores de distintos tipo.

¿Cómo podríamos crear un Data Frame en Python?

Para resolver el ejercicio, es necesario ocupar Python IDLE

Nombre	Edad	Género	Calificación
Felipe	24	Masculino	4,5
Andrea	21	Femenino	7,0
Tomás	22	Masculino	6,1
Roberto	20	Masculino	5,5

Crear un Data Frame en Python

Recuerda revisar el archivo **m1-ej1.py** →

De acuerdo a la tabla anterior podemos hacerlo a partir de una lista de listas.

CÓDIGO

```
import pandas as pd

data = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],
        ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
df = pd.DataFrame(data)

print(df)
```

Es posible mediante la función `DataFrame` de la librería *Pandas* (la primera línea de este código está representada por la variable `pd`).

RESULTADO

```
        0   1          2    3
0   Felipe  24  Masculino  4.5
1   Andrea  21   Femenino  7.0
2    Tomás  22  Masculino  6.1
3  Roberto  20  Masculino  5.5
```

Una vez que creamos la lista de listas de nombre data, se debe crear el Data Frame.

Crear un Data Frame en Python

Recuerda revisar el archivo **m1-ej1.py** →

Hay números que acompañan a las filas y columnas, sirven para identificar cada una.

CÓDIGO

```
import pandas as pd

data = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],
        ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
df = pd.DataFrame(data)

print(df)
```

RESULTADO

```
        0   1          2    3
0  Felipe  24  Masculino  4.5
1  Andrea  21   Femenino  7.0
2   Tomás  22  Masculino  6.1
3  Roberto  20  Masculino  5.5
```

Para las columnas, tener un número no es muy útil, por lo tanto nos gustaría poner un nombre a cada columna.

Para las filas, llamaremos a estos números “**índices**”.

Crear un Data Frame en Python

Recuerda revisar el archivo **m1-ej2.py** →

En el segundo parámetro ocupamos el comando “columns”.

CÓDIGO

```
import pandas as pd

data = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],
        ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
df = pd.DataFrame(data,columns = ["Nombre","Edad","Género","Calificación"])

print(df)
```

RESULTADO

```
    Nombre  Edad     Género  Calificación
0   Felipe    24  Masculino           4.5
1   Andrea    21   Femenino           7.0
2    Tomás    22  Masculino           6.1
3  Roberto    20  Masculino           5.5
```

Usamos el comando “`columns`” e ingresamos una lista con los nombres de las columnas. El primer nombre de columna que anotamos en esta lista es “Nombre”, el segundo nombre es “Edad” y así sucesivamente.

Crear un Data Frame en Python

Recuerda revisar el archivo **m1-ej2.py** →

En el segundo parámetro ocupamos el comando “columns”.

CÓDIGO

```
import pandas as pd

data = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],
        ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
df = pd.DataFrame(data,columns = ["Nombre","Edad","Género","Calificación"])

print(df)
```

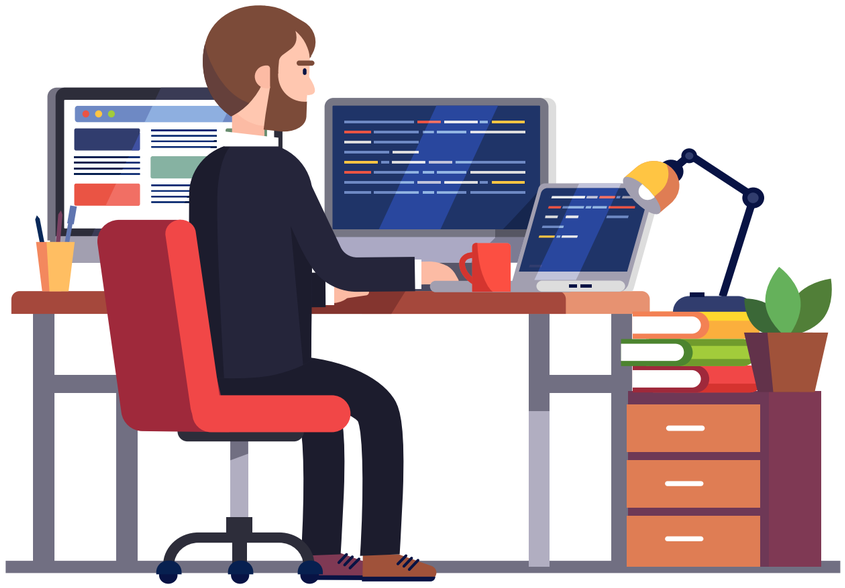
RESULTADO

```
    Nombre  Edad     Género  Calificación
0   Felipe    24  Masculino           4.5
1   Andrea    21   Femenino           7.0
2    Tomás    22  Masculino           6.1
3  Roberto    20  Masculino           5.5
```

Lo importante es que las columnas ya no están denominadas por un número, sino por el nombre que ingresamos.

>>> Data Frames mediante CSV

Data Frames mediante CSV



¿Por qué trabajar Data Frames mediante CSV?

La mayor ventaja de la librería *Pandas* es procesar grandes volúmenes de datos.

Definir Data Frames de forma manual nunca permitirá manejarlos con muchas líneas.

Por lo tanto, la mejor forma de cargar información es mediante archivos. Específicamente mediante archivos que estén en formato CSV.

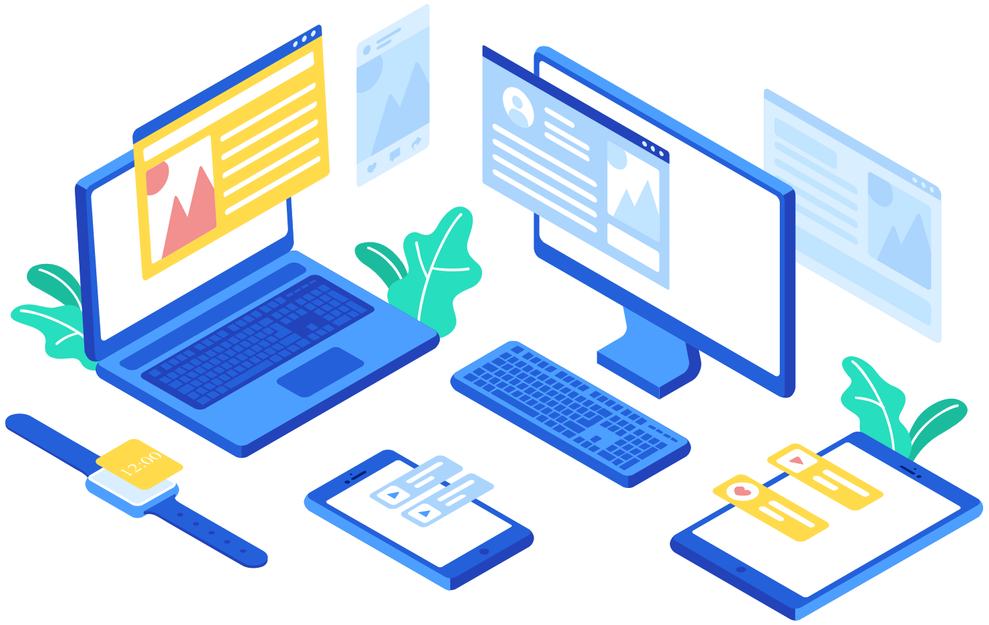
Practiquemos un Data Frames mediante CSV

1

Para desarrollar este ejercicio debes ir a los ejemplos.

2

Descargar un archivo CSV con cerca de 2.000 líneas de información de distintos clientes de un banco.



Con esta información trabajaremos y servirá como ejemplo para ir probando los conceptos que revisaremos.

Carguemos el archivo “clientes.csv” como un Data Frame

Recuerda revisar el archivo **m1-ej3.py** →

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df)
```

RESULTADO

```
          ID          RUT  ...    MONTO PUNTAJE_CREDITICIO
0          0  21.930.631-4  ...  5407949                1.17
1          1  11.269.366-8  ...  8153651                2.37
2          2   9.655.791-3  ...  9509104                9.91
3          3  16.644.711-4  ...  6065538                2.86
4          4  17.054.286-6  ...  8024077                0.56
```

¿Cómo hacerlo?

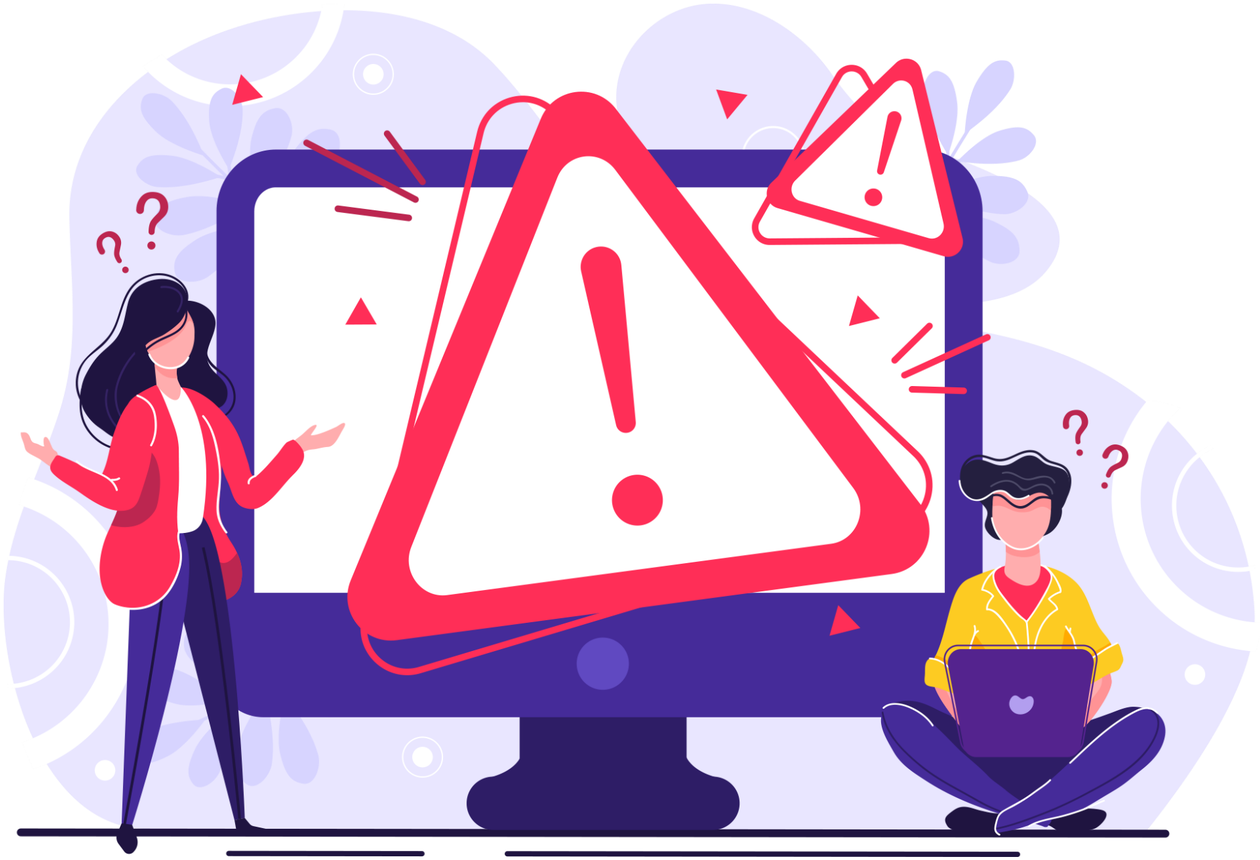
Lo más importante es que para cargar un archivo CSV, ocupamos la función `read_csv` de la librería *Pandas*.

El primer parámetro que ingresamos es el nombre del archivo.

Al imprimir en consola la variable `df` sabemos que el archivo de datos se cargó correctamente.

¿Qué errores podría encontrar al cargar un archivo CSV?

Es difícil anticiparnos a qué errores enfrentarán, así que la recomendación general es autonomía para investigar cuando aparezcan errores. Esto significa copiar el error que aparecerá en la consola, y buscar en Internet alguna solución. Esto además permitirá aprender más de esta grandiosa librería por tu cuenta.



Recuerda revisar el archivo **m1-ej3.py** →

Algunos errores comunes

El segundo parámetro usado al crear el Data Frame es:

```
encoding="latin-1"
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df)
```

Importante

De forma general, si tus datos tienen algún tipo de carácter que se use solo en el español (por ejemplo, tildes, ñ o ¡), debes ocupar este parámetro para evitar errores.

Recuerda revisar el archivo **m1-ej3.py** →

Algunos errores comunes

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df)
```

El tercer parámetro utilizado: `sep=";"`.

El separador de columnas dentro del archivo CSV “`;`” ó “`,`”. Asegura que la librería *Pandas* sepa cómo separar las columnas en cada fila.

Importante

Fíjate que el separador decimal para números decimales sea un punto.

Verifica que el archivo que estás cargando esté en la misma carpeta que el archivo tipo Python. Puedes ver un tutorial sobre lo anterior en el video "¿Cómo cargar un archivo a Pandas?".

>>> Tipos de datos

Tipos de datos: comando *dtypes*

Recuerda revisar el archivo **m1-ej4.py** →

Con esta operación logramos ver los tipos de datos para cada columna.

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.dtypes)
```

RESULTADO

```
ID                    int64
RUT                  object
NOMBRE               object
FECHA_NAC            object
TIPO_CLIENTE         object
MONTO                 int64
PUNTAJE_CREDITICIO  float64
dtype: object
```

La librería asignó correctamente los tipos de datos a cada columna, y de esa forma trabaja correctamente con ellas.

En *Pandas*, el tipo de datos `object` es equivalente a un `string`.

>>> Extraer columnas

Recuerda revisar el archivo **m1-ej5.py** →

Extraer columna

Permite mostrar la información de una sola columna. El comando general para `df` es:

```
df["nombre columna"]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df["RUT"])
```

RESULTADO

```
996      12.568.934-5
997      14.407.999-3
998      20.166.403-3
999      10.715.550-9
1000     16.396.396-6
Name: RUT, Length: 1001, dtype: object
```

Vemos el contenido de la columna “RUT”, además del tipo y la cantidad de datos de esa columna.

>>> Extraer fila

Extraer fila

Recuerda revisar el archivo **m1-ej6.py** →

Permite mostrar la información de una sola fila. El comando para `df` es:

```
df.loc[número fila]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.loc[658])
```

RESULTADO

```
ID                                658
RUT                        15.991.075-4
NOMBRE          María Daniela Saavedra Marín
FECHA_NAC                      1992/5/21
TIPO_CLIENTE                           C
MONTO                             874300
PUNTAJE_CREDITICIO                  1.66
Name: 658, dtype: object
```

Recuerda revisar el archivo **m1-ej7.py** →

Extraer filas

Sirve para mostrar la información de varias filas. El comando para `df` es:

```
df.loc[número fila inicial:número fila final]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.loc[100:105])
```

RESULTADO

```
      ID          RUT  ...  MONTO  PUNTAJE_CREDITICIO
100  100   8.489.240-5  ...  1361720                0.14
101  101  11.615.086-5  ...  5680734                8.39
102  102  12.061.292-0  ...  3130145                0.98
103  103  13.381.106-8  ...  3307564                0.62
104  104  12.866.411-7  ...  6225013                6.77
105  105  8.929.258-10  ...  1368881                7.42
```

En este caso vemos el contenido de las filas de la 100 a la 105.

>>> Extraer valor

Recuerda revisar el archivo **m1-ej8.py** →

Extraer valor

Se usa para exponer la información de una celda de la matriz representada por el Data Frame.

El comando para `df` es:

```
df.loc[número fila]["Nombre columna"]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.loc[658]["FECHA_NAC"])
```

RESULTADO

```
1992/5/21
```

El contenido de las fila 658 y la columna “FECHA_NAC” corresponde a la información de la persona.

Recuerda revisar el archivo **m1-ej9.py** →

Extraer valor

Es posible filtrar nuestros datos según los valores de ciertas columnas. El comando para `df` es:

```
df.loc[df['Nombre columna'](operación lógica)]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.loc[df['TIPO_CLIENTE'] == "A"])
```

RESULTADO

```
    ID           RUT  ...    MONTO PUNTAJE_CREDITICIO
 1   1  11.269.366-8  ...  8153651                2.37
12  12  13.674.785-2  ...   469341                2.81
19  19   7.699.998-8  ...  1836607                0.33
21  21   7.625.542-2  ...  5766978                7.70
22  22   7.371.571-0  ...   798432                2.90
```

Aquí filtramos solo a los clientes que son de tipo A.

Extraer valor

Recuerda revisar el archivo **m1-ej10.py** →

Es posible filtrar nuestros datos según los valores de ciertas columnas. El comando para `df` es:

```
df.loc[df['Nombre columna'](operación lógica)]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.loc[df['MONTO'] < 100000])
```

RESULTADO

```
      ID            RUT  ...  MONTO PUNTAJE_CREDITICIO
442  442   13.393.426-9  ...  54358                3.35
526  526    7.345.656-5  ...  42298                3.70
537  537   20.081.815-9  ...  72235                2.89
573  573   17.796.256-1  ...  27274                7.90
584  584    18.05.743-8  ...  97141                5.31
665  665   18.038.640-4  ...  36214                3.13
747  747  18.776.869-10  ...  48929                9.95
883  883   13.911.957-2  ...  10553                2.90
[8 rows x 7 columns]
```

Aquí filtramos solo a los clientes que tienen un monto menor a 100000.

Extraer valor

Recuerda revisar el archivo **m1-ej11.py** →

Podemos asignar la tabla filtrada a una variable para luego seguir trabajando con ella.

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

df_final = df.loc[df['PUNTAJE_CREDITICIO'] <= 8.0]

print(df_final)
```

RESULTADO

```
      ID            RUT  ...  MONTO PUNTAJE_CREDITICIO
442  442   13.393.426-9  ...  54358                3.35
526  526    7.345.656-5  ...  42298                3.70
537  537   20.081.815-9  ...  72235                2.89
573  573   17.796.256-1  ...  27274                7.90
584  584    18.05.743-8  ...  97141                5.31
```

La variable `df_final` contiene un Data Frame filtrado según el valor de una columna.

>>> Columnas

Recuerda revisar el archivo **m1-ej12.py →**

Crear columnas

Es posible agregar nuevas columnas a los Data Frames.

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

df["NACIONALIDAD"] = "CHILE"

print(df)
```

RESULTADO

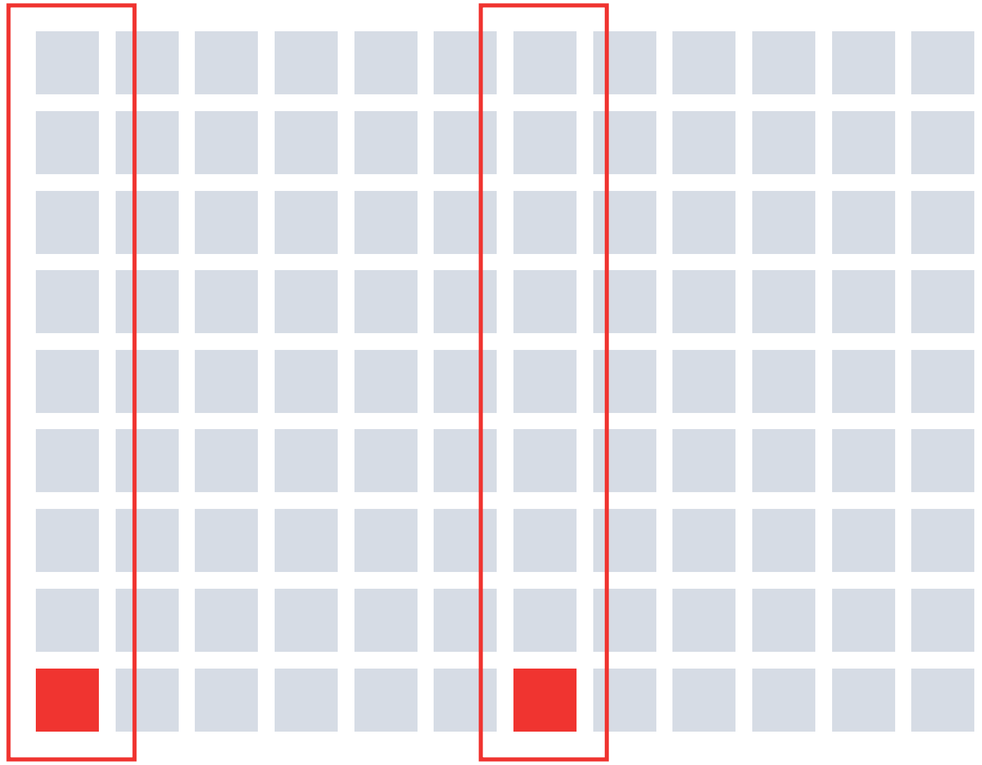
```
   ID           RUT  ... PUNTAJE_CREDITICIO NACIONALIDAD
0   0  21.930.631-4  ...               1.17        CHILE
1   1  11.269.366-8  ...               2.37        CHILE
2   2   9.655.791-3  ...               9.91        CHILE
3   3  16.644.711-4  ...               2.86        CHILE
4   4  17.054.286-6  ...               0.56        CHILE
```

Pensemos en la situación de agregar la nacionalidad de los clientes a nuestra base de datos.

En este caso son todos chilenos.

¿Es posible que una columna sea el resultado de una operación entre columnas?

Las columnas creadas no solo pueden ser el resultado de un cálculo entre una columna y un número, sino también el resultado de una operación entre columnas.



Columna como resultado de una operación entre columnas

Recuerda revisar el archivo **m1-ej13.py** →

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

df["BONO"] = df["MONTO"]/df["PUNTAJE_CREDITICIO"]

print(df)
```

En este caso, crearemos una nueva columna “BONO”, con el monto en la cuenta corriente de cada cliente (columna “MONTO”) dividida por el Puntaje crediticio (columna “PUNTAJE CREDITICIO”) de cada persona.

RESULTADO

ID	RUT	NOMBRE	MONTO	PUNTAJE_CREDITICIO	BONO
0	21.930.631-4	Isabel Blanca Marín Díaz	5407949	1.17	4.622179e+06
1	11.269.366-8	Cecilia Paula López Valenzuela	8153651	2.37	3.440359e+06
2	9.655.791-3	Vicente Felipe Robles Muñoz	9509104	9.91	9.595463e+05
3	16.644.711-4	Daniela María Robles Ruiz	6065538	2.86	2.120817e+06
4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	8024077	0.56	1.432871e+07

Recuerda revisar el archivo **m1-ej12.py →**

Eliminar columnas

Para eliminar columnas a los Data Frames, se ocupa el siguiente comando de manera general:

```
del nombre_data_frame["Nombre columna"]
```

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

df["NACIONALIDAD"] = "CHILE"

print(df)

del df["NACIONALIDAD"]

print(df)
```

RESULTADO

```
   ID           RUT  ... PUNTAJE_CREDITICIO
0   0  21.930.631-4  ...               1.17
1   1  11.269.366-8  ...               2.37
2   2   9.655.791-3  ...               9.91
3   3  16.644.711-4  ...               2.86
4   4  17.054.286-6  ...               0.56
```

Estadísticos descriptivos

Recuerda revisar el archivo **m1-ej14.py** →

Los estadísticos descriptivos son: cuenta, promedio, desviación estándar, mínimo, cuartiles, máximo. En nuestro ejemplo, estas son **Monto** y **Puntaje crediticio**.

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

print(df.describe())
```

RESULTADO

	ID	MONTO	PUNTAJE_CREDITICIO
count	1001.000000	1.001000e+03	1001.000000
mean	500.000000	4.863262e+06	5.073077
std	289.108111	2.864435e+06	2.872472
min	0.000000	1.055300e+04	0.000000
25%	250.000000	2.435418e+06	2.630000
50%	500.000000	4.781938e+06	5.130000
75%	750.000000	7.402329e+06	7.560000
max	1000.000000	9.998301e+06	9.990000

También calcula los estadísticos de la variable ID ya que es numérica.

Escribir archivo CSV

Recuerda revisar el archivo **m1-ej15.py** →

Finalmente, se guarda lo que hicimos escribiendo el Data Frame en un nuevo archivo CSV de la siguiente manera:

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

CÓDIGO

```
import pandas as pd

df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")

df["NACIONALIDAD"] = "CHILE"

df.to_csv("clientes_modificado.csv",sep=";",index=False)
```

En este ejemplo guardamos nuestro nuevo Data Frame con la columna “Nacionalidad” que no existía previamente.

Conclusiones

Hemos revisado las herramientas básicas de trabajo con la librería *Pandas*, que te permitirán:

- Cargar archivos CSV en un Data Frame.
- Hacer operaciones con sus columnas para trabajar con estos datos. Estas operaciones son la lectura, creación, edición, eliminación de columnas.
- Además, la creación de filtros básicos para extraer información de Data Frame, así como la posibilidad de guardar cualquier cambio realizado en el Data Frame en un archivo CSV.

>>> Cierre

Has finalizado la revisión de los contenidos de esta clase.

A continuación, te invitamos a realizar las actividades y a revisar los recursos del módulo que encontrarás en plataforma.