

Los volúmenes de datos con que las empresas trabajan son cada vez mayores y pocas tienen la capacidad de interpretarlos efectivamente. Esto se ha convertido en una necesidad transversal, ya que no cuentan las habilidades o herramientas para procesar y solucionar esta problemática. En consecuencia, este curso nace con el objetivo de ofrecer técnicas y herramientas avanzadas de procesamiento de datos mediante el uso de funciones de la librería pandas. Estas técnicas permiten obtener información muy valiosa y resolver problemas a partir de grandes volúmenes de datos.

Objetivo general

Aplicar herramientas avanzadas de la librería Pandas para el procesamiento de información a partir de Data Frames.

Instalación Python

Verificar PDF en el mismo directorio.

¿Cuántos datos se generan en internet?

Según estudio de DOMO Empresa mundial líder en tecnología digital y servicios web, **solo en 1 minuto**:

 4KK 4 millones de búsquedas en Google.

 4,5KK Se ven apróx. 4,5 millones de videos en Youtube.

 12KK Se mandan apróx. 12 millones de SMS.

 50K Suben casi 50 mil fotos a Instagram.



¿Por qué es relevante?

La cantidad de datos generados es enorme y aumenta.



¿Y en las empresas?

El desafío es darle valor a los datos disponibles.

¿Cómo dar valor a los datos de una empresa?



Big data



Algunas definiciones sobre Ciencia de Datos

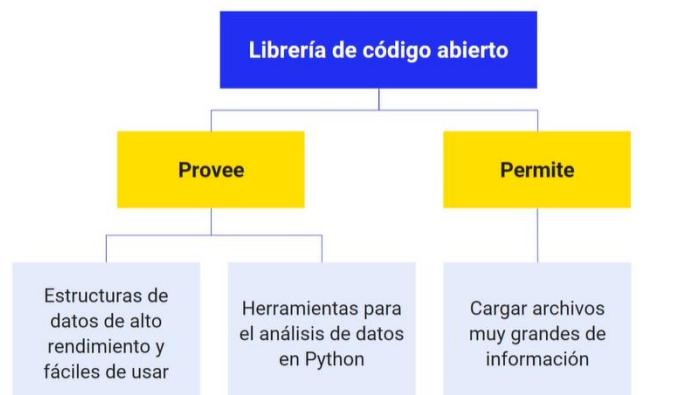
“ Es la disciplina que permite encontrar patrones predecibles en sets de datos estructurados y no estructurados.”

Vasant Dhar, 2013

Ciencia de Datos

Puede complementarse con otras ciencias que requieran un análisis de patrones en grandes cantidades de datos. Por ejemplo la **Astronomía**.

¿Qué entendemos por librería Pandas?



¿Qué archivos ocuparemos en Pandas?



Texto plano y replican una matriz.

Conclusiones



Big Data y Ciencia de Datos son herramientas que permiten extraer información de grandes volúmenes de datos.



La librería Pandas de Python es una de las más usadas para procesar grandes volúmenes de datos.



El formato de archivos CSV es ampliamente usado como una forma de representar y almacenar a un bajo costo datos como una tabla.

Las estructuras de datos básicas en Pandas son: series y Data Frames.

- **Serie:** Es una lista de datos con un largo fijo, aquí no se agregan o quitan elementos, sólo se modifican. Además, los datos de una serie son del mismo tipo.
- **Data Frame:** Los Data Frames son la estructura más usada en la librería Pandas. Se percibe como una matriz de datos. Cada columna, representa datos para una variable específica. Y cada fila corresponde a medidas o valores de cada instancia. Se observa que cada columna es de un tipo específico, y cada fila tiene valores de distintos tipos. En esta estructura de datos, se agregan filas y columnas según la preferencia del programador. Las columnas de un Data Frame son series. Es importante destacar que cada fila tendrá un índice, que permitirá acceder a esa fila en particular y su información específica.

Operaciones básicas de un Data Frame

¿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.

Crear un Data Frame en Python

m1-ej1.py

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

CÓDIGO
<pre>import pandas as pd\n\ndata = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],\n ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]\ndf = pd.DataFrame(data)\nprint(df)</pre>

RESULTADO
<pre> 0 1 2 3\n0 Felipe 24 Masculino 4.5\n1 Andrea 21 Femenino 7.0\n2 Tomás 22 Masculino 6.1\n3 Roberto 20 Masculino 5.5</pre>

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

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

Crear un Data Frame en Python

m1-ej1.py

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

CÓDIGO
<pre>import pandas as pd\n\ndata = [["Felipe",24,"Masculino",4.5],["Andrea",21,"Femenino",7.0],\n ["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]\ndf = pd.DataFrame(data)\nprint(df)</pre>

RESULTADO
<pre> 0 1 2 3\n0 Felipe 24 Masculino 4.5\n1 Andrea 21 Femenino 7.0\n2 Tomás 22 Masculino 6.1\n3 Roberto 20 Masculino 5.5</pre>

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

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.

```
import pandas as pd
```

```
data = [["Felipe", 24, "Masculino", 4.5], ["Andrea", 21, "Femenino", 7.0],
        ["Tomas", 22, "Masculino", 6.1], ["Roberto", 20, "Masculino", 5.5]]
df = pd.DataFrame(data, columns=["NOMBRE", "EDAD", "GENERO", "CALIFICACIONES"])
print(df)
```

	NOMBRE	EDAD	GENERO	CALIFICACIONES
0	Felipe	24	Masculino	4.5
1	Andrea	21	Femenino	7.0
2	Tomas	22	Masculino	6.1
3	Roberto	20	Masculino	5.5

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 qué trabajar Data Frames mediante CSV?



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

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

m1-ej3.py

CÓDIGO

```
import pandas as pd

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

print(df)
```

¿Cómo hacerlo?

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

RESULTADO

	ID	RUT	...	MONTO	PUNTAJE_CREDITICO
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

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.

Algunos errores comunes

m1-ej3.py -

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.

Algunos errores comunes

m1-ej3.py -

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

Fijate 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?".

```
import pandas as pd
df = pd.read_csv("clientes.csv", encoding="latin-1", sep=";")
print(df)
```

	ID	RUT	NOMBRE	FECHA_NAC	\
0	0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13	
1	1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28	
2	2	9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2	
3	3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7	
4	4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5	
...
996	996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6	

Tipos de datos: comando `dtypes`

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`.

```
print(df.dtypes)

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

Extraer columna

m1-ej5.py -

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

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

CÓDIGO	RESULTADO
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") print(df["RUT"])</pre>	<pre>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</pre>

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

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

0      21.930.631-4
1      11.269.366-8
2      9.655.791-3
3      16.644.711-4
4      17.054.286-6
...
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
```

Extraer fila

m1-ej6.py -

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

```
df.loc[número fila]
```

CÓDIGO	RESULTADO
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") print(df.loc[658])</pre>	<pre>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</pre>

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

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
```

Extraer filas

m1-ej7.py →

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

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

CÓDIGO	RESULTADO																												
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=",") print(df.loc[100:105])</pre>	<table border="1"><thead><tr><th>ID</th><th>RUT</th><th>...</th><th>MONTO_PUNTAJE_CREDITICIO</th></tr></thead><tbody><tr><td>100</td><td>100</td><td>8.489.240-5</td><td>...</td></tr><tr><td>101</td><td>101</td><td>11.615.086-5</td><td>5680734</td></tr><tr><td>102</td><td>102</td><td>12.061.292-0</td><td>3130145</td></tr><tr><td>103</td><td>103</td><td>13.381.106-8</td><td>3307564</td></tr><tr><td>104</td><td>104</td><td>12.866.411-7</td><td>6225013</td></tr><tr><td>105</td><td>105</td><td>8.929.258-10</td><td>1368881</td></tr></tbody></table>	ID	RUT	...	MONTO_PUNTAJE_CREDITICIO	100	100	8.489.240-5	...	101	101	11.615.086-5	5680734	102	102	12.061.292-0	3130145	103	103	13.381.106-8	3307564	104	104	12.866.411-7	6225013	105	105	8.929.258-10	1368881
ID	RUT	...	MONTO_PUNTAJE_CREDITICIO																										
100	100	8.489.240-5	...																										
101	101	11.615.086-5	5680734																										
102	102	12.061.292-0	3130145																										
103	103	13.381.106-8	3307564																										
104	104	12.866.411-7	6225013																										
105	105	8.929.258-10	1368881																										

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

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

```
ID          RUT           NOMBRE  FECHA_NAC \
100  100  8.489.240-5  Felipe Andrés Muñoz Rojas  1980/9/10
101  101  11.615.086-5  Ramón Gabriel Saavedra Rodriguez  1977/5/29
102  102  12.061.292-0  María Isidora Valenzuela Rodriguez  1980/3/5
103  103  13.381.106-8  Vicente Juan Marín Saavedra  1969/4/19
104  104  12.866.411-7  Valeria Pamela Saavedra Saavedra  1991/10/28
105  105  8.929.258-10  Andrés Ramón Rodríguez Valenzuela  1961/9/27

TIPO_CLIENTE    MONTO  PUNTAJE_CREDITICIO
100              E   1361720            0.14
101              B   5680734            8.39
102              D   3130145            0.98
103              C   3307564            0.62
104              E   6225013            6.77
105              E   1368881            7.42
```

Extraer valor

m1-ej8.py →

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 inicial]["Nombre columna"]
```

CÓDIGO	RESULTADO
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=",") print(df.loc[658]["FECHA_NAC"])</pre>	1992/5/21

El contenido de la fila 658 y la columna "FECHA_NAC" corresponde a la información de la persona.

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

1992/5/21

Extraer valor

m1-ej9.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	RESULTADO																														
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") print(df.loc[df['TIPO_CLIENTE'] == "A"])</pre>	<table><thead><tr><th>ID</th><th>RUT</th><th>...</th><th>MONTO</th><th>PUNTAJE_CREDITICIO</th></tr></thead><tbody><tr><td>1</td><td>11.269.366-8</td><td>...</td><td>8153651</td><td>2.37</td></tr><tr><td>12</td><td>13.674.785-2</td><td>...</td><td>469341</td><td>2.81</td></tr><tr><td>19</td><td>7.699.998-8</td><td>...</td><td>1836607</td><td>0.33</td></tr><tr><td>21</td><td>7.625.542-2</td><td>...</td><td>5766978</td><td>7.70</td></tr><tr><td>22</td><td>7.371.571-0</td><td>...</td><td>798432</td><td>2.90</td></tr></tbody></table>	ID	RUT	...	MONTO	PUNTAJE_CREDITICIO	1	11.269.366-8	...	8153651	2.37	12	13.674.785-2	...	469341	2.81	19	7.699.998-8	...	1836607	0.33	21	7.625.542-2	...	5766978	7.70	22	7.371.571-0	...	798432	2.90
ID	RUT	...	MONTO	PUNTAJE_CREDITICIO																											
1	11.269.366-8	...	8153651	2.37																											
12	13.674.785-2	...	469341	2.81																											
19	7.699.998-8	...	1836607	0.33																											
21	7.625.542-2	...	5766978	7.70																											
22	7.371.571-0	...	798432	2.90																											

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

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

      ID          RUT          NOMBRE FECHA_NAC \
2     2  9.655.791-3  Vicente Felipe Robles Muñoz  1972/2/2
8     8  20.749.832-5  Felipe Andrés Rodríguez Valenzuela  1973/2/9
13    13  17.452.036-6   Ramón Vicente Vergara Robles  1997/7/16
14    14  20.481.183-10  Andrés Francisco López López  1977/7/27
16    16  5.814.759-5  Andrés Felipe Quiroga Robles  1990/7/17
...   ...
989   989  15.116.681-1  Vicente Pedro Rodríguez Campos  1966/4/17
992   992  7.184.479-2  Isidora Paula Vergara Robles  1984/10/8
993   993  10.302.602-4  Javiera Cecilia Saavedra Valenzuela  1973/6/7
995   995  11.942.539-4  Isidora Javiera López Rodríguez  1977/1/5
998   998  20.166.403-3  Ignacio Bernardo López Valenzuela  1962/1/0

      TIPO_CLIENTE  MONTO  PUNTAJE_CREDITICIO
2                 E  9509104            9.91
8                 E  4441737            8.18
13                E  8845556            5.90
14                E  6285039            4.68
16                E  7790975            0.91
...   ...
989               ...  1472637            1.39
992               ...  3174100            1.92
993               ...  2068637            6.30
995               ...  2949763            1.17
998               ...  7822257            7.32
```

Extraer valor

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	RESULTADO																																													
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") print(df.loc[df['MONTO'] < 100000])</pre>	<table><thead><tr><th>ID</th><th>RUT</th><th>...</th><th>MONTO</th><th>PUNTAJE_CREDITICIO</th></tr></thead><tbody><tr><td>442</td><td>13.393.426-9</td><td>...</td><td>54358</td><td>3.35</td></tr><tr><td>526</td><td>7.345.656-5</td><td>...</td><td>42298</td><td>3.70</td></tr><tr><td>537</td><td>20.081.815-9</td><td>...</td><td>72235</td><td>2.89</td></tr><tr><td>573</td><td>17.796.256-1</td><td>...</td><td>27274</td><td>7.90</td></tr><tr><td>584</td><td>18.05.743-8</td><td>...</td><td>97141</td><td>5.31</td></tr><tr><td>665</td><td>18.038.649-4</td><td>...</td><td>36214</td><td>3.13</td></tr><tr><td>747</td><td>18.776.869-10</td><td>...</td><td>48929</td><td>9.95</td></tr><tr><td>883</td><td>13.911.957-2</td><td>...</td><td>10553</td><td>2.90</td></tr></tbody></table> <p>[8 rows x 7 columns]</p>	ID	RUT	...	MONTO	PUNTAJE_CREDITICIO	442	13.393.426-9	...	54358	3.35	526	7.345.656-5	...	42298	3.70	537	20.081.815-9	...	72235	2.89	573	17.796.256-1	...	27274	7.90	584	18.05.743-8	...	97141	5.31	665	18.038.649-4	...	36214	3.13	747	18.776.869-10	...	48929	9.95	883	13.911.957-2	...	10553	2.90
ID	RUT	...	MONTO	PUNTAJE_CREDITICIO																																										
442	13.393.426-9	...	54358	3.35																																										
526	7.345.656-5	...	42298	3.70																																										
537	20.081.815-9	...	72235	2.89																																										
573	17.796.256-1	...	27274	7.90																																										
584	18.05.743-8	...	97141	5.31																																										
665	18.038.649-4	...	36214	3.13																																										
747	18.776.869-10	...	48929	9.95																																										
883	13.911.957-2	...	10553	2.90																																										

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

```

print(df.loc[df["MONTO"] > 100000])

```

	ID	RUT	NOMBRE	FECHA_NAC
0	0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
1	1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
2	2	9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2
3	3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7
4	4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
...
996	996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6
997	997	14.407.999-3	Maria Valeria Marín Robles	1995/12/0
998	998	20.166.403-3	Ignacio Bernardo López Valenzuela	1962/1/0
999	999	10.715.550-9	Bernardo Pablo Saavedra Castro	1998/11/10
1000	1000	16.396.396-6	Javiera Ignacia Rojas Quiroga	1980/4/11

	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
0	C	5407949	1.17
1	A	8153651	2.37
2	E	9509104	9.91
3	B	6065538	2.86
4	C	8024077	0.56
...
996	B	5052388	6.01
997	B	765834	8.69
998	E	7822257	7.32
999	D	8506664	9.92
1000	C	7853434	0.10

Extraer valor

m1-ej11.py

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

CÓDIGO	RESULTADO																								
<pre> 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) </pre>	<table border="1"> <thead> <tr> <th>ID</th> <th>RUT</th> <th>MONTO</th> <th>PUNTAJE_CREDITICIO</th> </tr> </thead> <tbody> <tr><td>442</td><td>442</td><td>13.393.426-9</td><td>5.35</td></tr> <tr><td>526</td><td>526</td><td>7.345.656-5</td><td>3.70</td></tr> <tr><td>537</td><td>537</td><td>20.081.815-9</td><td>2.89</td></tr> <tr><td>573</td><td>573</td><td>17.796.256-1</td><td>7.90</td></tr> <tr><td>584</td><td>584</td><td>18.05.743-8</td><td>5.31</td></tr> </tbody> </table>	ID	RUT	MONTO	PUNTAJE_CREDITICIO	442	442	13.393.426-9	5.35	526	526	7.345.656-5	3.70	537	537	20.081.815-9	2.89	573	573	17.796.256-1	7.90	584	584	18.05.743-8	5.31
ID	RUT	MONTO	PUNTAJE_CREDITICIO																						
442	442	13.393.426-9	5.35																						
526	526	7.345.656-5	3.70																						
537	537	20.081.815-9	2.89																						
573	573	17.796.256-1	7.90																						
584	584	18.05.743-8	5.31																						

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

```

print(df.loc[df["PUNTAJE_CREDITICIO"] <= 8.0])

```

ID	RUT	NOMBRE	FECHA_NAC
0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7
4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
5	11.170.160-6	Vicente Vicente Marín Vergara	1971/7/11
...
994	21.307.599-10	Andrea Constanza Castro Robles	1992/7/10
995	11.942.539-4	Isidora Javiera López Rodríguez	1977/1/5
996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6
998	20.166.403-3	Ignacio Bernardo López Valenzuela	1962/1/0
1000	16.396.396-6	Javiera Ignacia Rojas Quiroga	1980/4/11

TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
C	5407949	1.17
A	8153651	2.37
B	6065538	2.86
C	8024077	0.56
C	4056141	5.98
...
D	9277469	2.31
E	2949763	1.17
B	5052388	6.01
E	7822257	7.32
C	7853434	0.10

Crear columnas

m1-ej12.py

Es posible agregar nuevas columnas a los Data Frames.

CÓDIGO	RESULTADO																								
<pre> import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") df["NACIONALIDAD"] = "CHILE" print(df) </pre>	<table border="1"> <thead> <tr> <th>ID</th> <th>RUT</th> <th>PUNTAJE_CREDITICIO</th> <th>NACIONALIDAD</th> </tr> </thead> <tbody> <tr><td>0</td><td>21.930.631-4</td><td>1.17</td><td>CHILE</td></tr> <tr><td>1</td><td>11.269.366-8</td><td>2.37</td><td>CHILE</td></tr> <tr><td>2</td><td>9.655.791-3</td><td>9.91</td><td>CHILE</td></tr> <tr><td>3</td><td>16.644.711-4</td><td>2.86</td><td>CHILE</td></tr> <tr><td>4</td><td>17.054.286-6</td><td>0.56</td><td>CHILE</td></tr> </tbody> </table>	ID	RUT	PUNTAJE_CREDITICIO	NACIONALIDAD	0	21.930.631-4	1.17	CHILE	1	11.269.366-8	2.37	CHILE	2	9.655.791-3	9.91	CHILE	3	16.644.711-4	2.86	CHILE	4	17.054.286-6	0.56	CHILE
ID	RUT	PUNTAJE_CREDITICIO	NACIONALIDAD																						
0	21.930.631-4	1.17	CHILE																						
1	11.269.366-8	2.37	CHILE																						
2	9.655.791-3	9.91	CHILE																						
3	16.644.711-4	2.86	CHILE																						
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.

```

df["NACIONALIDAD"] = "CHILE"
print(df)

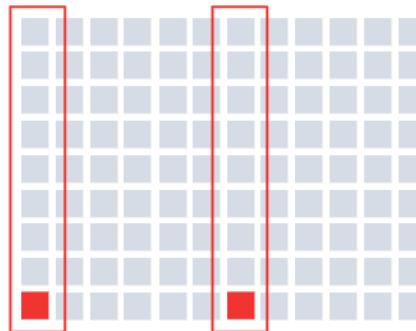
```

ID	RUT	NOMBRE	FECHA_NAC
0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
2	9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2
3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7
4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
...
996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6
997	20.166.403-3	María Valeria Marín Robles	1995/12/0
998	10.715.550-9	Ignacio Bernardo López Valenzuela	1962/1/0
999	16.396.396-6	Bernardo Pablo Saavedra Castro	1998/11/10
1000	16.396.396-6	Javiera Ignacia Rojas Quiroga	1980/4/11

TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO	NACIONALIDAD
C	5407949	1.17	CHILE
A	8153651	2.37	CHILE
E	9569104	9.91	CHILE
B	6065538	2.86	CHILE
C	8024077	0.56	CHILE
...
B	5052388	6.01	CHILE
B	765834	8.69	CHILE
E	7822257	7.32	CHILE
D	8506664	9.92	CHILE
C	7853434	0.10	CHILE

¿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

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

```
df["BONO"] = df["MONTO"] / df["PUNTAJE_CREDITICIO"]  
print(df)
```

ID	RUT	NOMBRE	FECHA_NAC
0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
2	9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2
3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7
4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
...
996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6
997	14.407.999-3	Maria Valeria Marín Robles	1995/12/0
998	20.166.403-3	Ignacio Bernardo López Valenzuela	1962/1/0
999	10.715.550-9	Bernardo Pablo Saavedra Castro	1998/11/10
1000	16.396.396-6	Javiera Ignacia Rojas Quiroga	1980/4/11

TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO	NACIONALIDAD	BONO
0	C	5407949	1.17	CHILE 4.622179e+06
1	A	8153651	2.37	CHILE 3.440359e+06
2	E	9509104	9.91	CHILE 9.595463e+05
3	B	6065538	2.86	CHILE 2.120817e+06
4	C	8024077	0.56	CHILE 1.432871e+07
...
996	B	5052388	6.01	CHILE 8.406636e+05
997	B	765834	8.69	CHILE 8.812819e+04
998	E	7822257	7.32	CHILE 1.068614e+06
999	D	8506664	9.92	CHILE 8.575266e+05
1000	C	7853434	0.10	CHILE 7.853434e+07

Eliminar columnas

m1-ej12.py

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

CÓDIGO	RESULTADO
<pre>del nombre_data_frame["Nombre columna"]</pre> <pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") df["NACIONALIDAD"] = "CHILE" print(df) del df["NACIONALIDAD"] print(df)</pre>	<pre>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</pre>

<pre>del df["NACIONALIDAD"] print(df)</pre>				
	ID	RUT	NOMBRE	FECHA_NAC \
0	0	21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
1	1	11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
2	2	9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2
3	3	16.644.711-4	Daniela María Robles Ruiz	1982/7/7
4	4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
...
996	996	12.568.934-5	Paula Daniela Muñoz Quiroga	1978/9/6
997	997	14.407.999-3	María Valeria Marín Robles	1995/12/0
998	998	20.166.403-3	Ignacio Bernardo López Valenzuela	1962/1/0
999	999	10.715.550-9	Bernardo Pablo Saavedra Castro	1998/11/10
1000	1000	16.396.396-6	Javiera Ignacia Rojas Quiroga	1980/4/11
	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO	BONO
0	C	5407949	1.17	4.622179e+06
1	A	8153651	2.37	3.440359e+06
2	F	9509104	9.91	9.595463e+05

Estadísticos descriptivos

m1-ej14.py

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

CÓDIGO	RESULTADO
<pre>import pandas as pd df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";") print(df.describe())</pre>	<pre>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</pre>

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

<pre>print(df.describe())</pre>				
	ID	MONTO	PUNTAJE_CREDITICIO	BONO
count	1001.000000	1.001000e+03	1001.000000	1.001000e+03
mean	500.000000	4.863262e+06	5.073077	inf
std	289.108111	2.864435e+06	2.872472	NaN
min	0.000000	1.055300e+04	0.000000	3.452405e+03
25%	250.000000	2.435418e+06	2.630000	4.546002e+05
50%	500.000000	4.781938e+06	5.130000	9.830709e+05
75%	750.000000	7.402329e+06	7.560000	1.866841e+06
max	1000.000000	9.998301e+06	9.990000	inf

Escribir archivo CSV

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.

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

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.

Conclusiones

Archivos .PY

```
1 # Importamos La libreria panda y la renombramos
2 import pandas as pd
3 # Creamos una lista de listas
4 data = [["Felipe",24,"Masculino",4.5],[["Andrea",21,"Femenino",7.0],
5 ---->["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
6 # Creamos un Data Frame a partir de la lista de listas
7 df = pd.DataFrame(data)
8 # Mostramos el Data Frame
9 print(df)
```

```
1 # Importamos la librería Pandas y la renombramos
2 import pandas as pd
3 # Creamos una lista de listas
4 data = [["Felipe",24,"Masculino",4.5],[["Andrea",21,"Femenino",7.0],
5 ---->["Tomás",22,"Masculino",6.1],["Roberto",20,"Masculino",5.5]]
6 # Creamos un Data Frame con la lista de listas, y agregamos las columnas
7 # para que no aparezcan números
8 df = pd.DataFrame(data,columns = ["Nombre","Edad","Género","Calificación"])
9 # Mostramos el Data Frame
10 print(df)
```

```
1 # Importamos la librería Pandas y la renombramos
2 import pandas as pd
3 # Leemos el archivo .CSV y generamos un Data Frame
4 # encoding para que reconozca Los caracteres Latino y
5 # el separador por punto y coma
6 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
7 # Mostramos el Data Frame
8 print(df)
```

```
1 # Importamos la librería Pandas y la renombramos
2 import pandas as pd
3 # Generamos el Data Frame a través del archivo .CSV
4 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
5 # Mostramos los tipos de datos de cada columna
6 print(df.dtypes)
```

```

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos una columna
5 print(df["RUT"])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos una Línea
5 print(df.loc[658])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos varias filas
5 print(df.loc[100:105])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos la fecha de nacimiento en esa linea
5 print(df.loc[658]["FECHA_NAC"])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Muestra todas las filas cuyo TIPO CLIENTE sea "A"
5 print(df.loc[df['TIPO_CLIENTE'] == "A"])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos todas las filas que tengan su columna MONTO
5 # menor a 100.000
6 print(df.loc[df['MONTO'] < 100000])

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos las filas cuya columna PUNTAJE_CREDITICIO sea
5 # menor o igual a 8.0
6 df_final = df.loc[df['PUNTAJE_CREDITICIO'] <= 8.0]
7
8 print(df_final)

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Creamos la columna NACIONALIDADN y la rellenamos con CHILE
5 df["NACIONALIDAD"] = "CHILE"
6
7 print(df)
8 # Eliminamos la columna NACIONALIDAD
9 del df["NACIONALIDAD"]
10
11 print(df)

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Creamos una columna a partir de la operación de otras
5 # dos columnas
6 df["BONO"] = df["MONTO"]/df["PUNTAJE_CREDITICIO"]
7
8 print(df)

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Mostramos las estadísticas descriptivas, que son:
5 # cuenta, promedio, desviación estándar, mínimo, cuartiles,
6 # máximo de las columnas numéricas.
7 print(df.describe())

```

```

1 import pandas as pd
2
3 df = pd.read_csv("clientes.csv",encoding="latin-1",sep=";")
4 # Agregamos la columna NACIONALIDAD y la completamos con CHILE
5 df["NACIONALIDAD"] = "CHILE"
6 # Guardamos Los cambios en un archivo nuevo .CSV
7 df.to_csv("clientes_modificado.csv",sep=";",index=False)

```

¿Caso práctico?

1

Debes descargar el archivo "datos_pacientes.csv" en la plataforma.

2

Cargar el archivo "datos_pacientes.csv" en Python. Para esto, crea un Data Frame de nombre `df_pacientes` con los datos de este archivo.

Este archivo contiene la información de los pacientes de la clínica UCO. En particular, cada fila contiene la siguiente información:

- a) RUT.
- b) Nombre: Con el formato "`primer_nombre segundo_nombre primer_apellido segundo_apellido`" de cada paciente.
- c) Fecha_Nacimiento: Fecha de nacimiento de cada paciente.
- d) Previsión: Tipo de previsión. FONASA o ISAPRE.
- e) Monto_Deuda: Monto de la deuda total en pesos chilenos de cada paciente.

```
# Mostramos la información de las columnas
print(df_pacientes)
```

	RUT	Nombre	Genero
0	16.302.272-9	Andrea Javiera Robles Díaz	FEMENINO
1	5.03.060-9	Daniela Victoria Ruiz López	FEMENINO
2	12.074.719-9	Francisco Felipe Castro Rojas	MASCULINO
3	19.121.201-0	Cecilia Valeria Díaz González	FEMENINO
4	20.348.342-8	Pablo Vicente Valenzuela Campos	MASCULINO
..
995	13.780.496-8	Ramón Francisco Valenzuela Valenzuela	MASCULINO
996	6.509.713-1	Isidora María Castro Díaz	FEMENINO
997	10.782.816-5	Pablo Bernardo Marín Rodríguez	MASCULINO
998	8.047.856-4	Isidora Ignacia Díaz Vergara	FEMENINO
999	20.312.199-4	Francisco Gabriel González González	MASCULINO
	Fecha_Nacimiento	Previsión	Monto_Deuda
0	1971/7/11	FONASA	1871093
1	1970/7/23	FONASA	793322
2	1966/2/22	FONASA	320538
3	1998/4/15	FONASA	469335
4	1978/1/15	ISAPRE	759052
..
995	1996/10/12	ISAPRE	1228834
996	1993/3/16	ISAPRE	805084
997	1962/3/19	FONASA	1273143
998	1992/11/13	ISAPRE	141725
999	1981/7/13	ISAPRE	1251681

3

El gerente de UCO te pidió cierta información, por lo tanto, imprime en consola lo siguiente:

- a) La información de la columna "Monto_Deuda".
- b) La fila con índice 651.
- c) Las filas desde el índice 100 al 200. Luego estas filas pero solo la columna "RUT" y "Nombre".
- d) Los pacientes que tienen previsión FONASA.
- e) Los pacientes que tienen un monto de deuda mayor a \$1.000.000.

4

Por orden del gerente, debes agregar el país de origen de los pacientes. Afortunadamente, todos los pacientes son de Chile, por lo que debes agregar una columna con esta información. Esta columna debe llamarse "Nacionalidad".

```
# Mostramos información de la columna "Monto_Deuda"
print(df_pacientes["Monto_Deuda"])
```

```
0    1871093
1    793322
2    320538
3    469335
4    759052
...
995   1228834
996   805084
997   1273143
998   141725
999   1251681
Name: Monto_Deuda, Length: 1000, dtype: int64
```

```
# Mostramos fila Nº651
print(df_pacientes.loc[651])
```

```
RUT           4.464.429-4
Nombre        Javiera Daniela Vergara Díaz
Genero        FEMENINO
Fecha_Nacimiento  1976/6/25
Previsión     ISAPRE
Monto_Deuda   191000
Name: 651, dtype: object
```

```
# Mostramos las filas desde el Nº100 al Nº200.
# Luego estas filas pero solo por columna "RUT" y "Nombre"
print(df_pacientes.loc[100:200])
print(df_pacientes.loc[100:200][["RUT"]])
print(df_pacientes.loc[100:200][["Nombre"]])
```

```
RUT          Nombre  Genero \
100  14.014.09-9  Gabriel Felipe Díaz Marín  MASCULINO
101  18.444.222-7  Francisco Bernardo Muñoz Castro  MASCULINO
102  13.716.907-0  Juan Vicente Rodríguez Quiroga  MASCULINO
103  17.734.624-0  Francisco Javier Marín Muñoz  MASCULINO
104  19.186.540-6  Rodrigo Rodrigo Rodríguez Campos  MASCULINO
```

```
# Los pacientes que tienen previsión FONASA
print(df_pacientes.loc[df_pacientes["Previsión"] == "FONASA"])
```

```
RUT          Nombre  Genero \
0   16.302.272-9  Andrea Javiera Robles Díaz  FEMENINO
1   5.03.060-9    Daniela Victoria Ruiz López  FEMENINO
2   12.074.719-9  Francisco Felipe Castro Rojas  MASCULINO
3   19.121.201-0  Cecilia Valeria Díaz González  FEMENINO
5   14.647.075-5  María Isabel López Campos    FEMENINO
...
991  16.652.447-7  Isidora Pamela Rodríguez Díaz  FEMENINO
```

```
# Pacientes que tienen un monto adeudado mayor a $1.000.000
df_pacientes.loc[df_pacientes["Monto_Deuda"] > 1000000]
```

	RUT	Nombre	Genero	Fecha_Nacimiento	Previsión	Monto_Deuda
0	16.302.272-9	Andrea Javiera Robles Díaz	FEMENINO	1971/7/11	FONASA	1871093
7	13.350.486-8	Pedro Bernardo Muñoz Vergara	MASCULINO	1990/5/12	FONASA	1130031
8	14.086.677-4	Pedro Juan Saavedra Quiroga	MASCULINO	1989/9/28	FONASA	1707784
9	6.111.964-8	Daniela Paula Valenzuela López	FEMENINO	2000/11/6	FONASA	1795495
11	13.880.857-6	Juan Gabriel Campos Vergara	MASCULINO	1966/6/30	FONASA	1717168

```
# Agregamos columna nacionalidad y la llenamos de "Chile"
df_pacientes["Nacionalidad"] = "Chile"
print(df_pacientes["Nacionalidad"])
```

```
0    Chile
1    Chile
2    Chile
3    Chile
4    Chile
...
995   Chile
996   Chile
997   Chile
998   Chile
999   Chile
Name: Nacionalidad, Length: 1000, dtype: object
```

5 Por un error del sistema, la deuda no había sido actualizada por IPC. Debes agregarle un 3% de reajuste.

6 La información sobre que todos los pacientes eran chilenos estaba equivocada, por lo que ahora debes eliminar la columna “**Nacionalidad**”.

7 En seguida, muestra los estadísticos descriptivos de **df_pacientes**.

8 Finalmente, guarda **df_pacientes** con el monto reajustado por **IPC** en un archivo CSV de nombre “**datos_pacientes_reajustado.csv**”.

```
# Actualizar la deuda en un 3%
df_pacientes["Monto_Deuda"] = df_pacientes["Monto_Deuda"] * 1.03
print(df_pacientes)
```

	RUT	Nombre	Genero
0	16.302.272-9	Andrea Javiera Robles Díaz	FEMENINO
1	5.03.060-9	Daniela Victoria Ruiz López	FEMENINO
2	12.074.719-9	Francisco Felipe Castro Rojas	MASCULINO
3	19.121.201-0	Cecilia Valeria Díaz González	FEMENINO
4	20.348.342-8	Pablo Vicente Valenzuela Campos	MASCULINO

```
# Eliminar columna Nacionalidad
del df_pacientes["Nacionalidad"]
print(df_pacientes)
```

	RUT	Nombre	Genero
0	16.302.272-9	Andrea Javiera Robles Díaz	FEMENINO
1	5.03.060-9	Daniela Victoria Ruiz López	FEMENINO
2	12.074.719-9	Francisco Felipe Castro Rojas	MASCULINO
3	19.121.201-0	Cecilia Valeria Diaz González	FEMENINO
4	20.348.342-8	Pablo Vicente Valenzuela Campos	MASCULINO

```
# Estadisticas
df_pacientes.describe()
```

	Monto_Deuda
count	1.000000e+03
mean	1.115393e+06
std	5.742349e+05
min	1.091779e+05
25%	6.131410e+05
50%	1.141549e+06
75%	1.623918e+06
max	2.059872e+06

```
# Guardamos los cambios
df_pacientes.to_csv("datos_pacientes_reajustado.csv", sep=";", index=False)
```

Apuntes:

- Recordemos que un Data Frame está compuesto por series, donde cada una de ellas representará una columna de la base de datos que simula el Data Frame.
- Recordemos que para crear un Data Frame a partir de una lista de listas, se puede hacer mediante el comando

```
Import pandas as pd
```

```
data = pd.DataFrame(datos)
```

Donde datos es la variable que almacena la lista de listas con la información.

- La forma de cargar un archivo CSV en un Data Frame es mediante el comando

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

Es importante incluir el nombre del archivo CSV, así como el encoding (para evitar la lectura incorrecta de ciertos caracteres como tildes), y definir el separador (para que el Data Frame se lea correctamente).

- Para acceder a una columna de un Data Frame, se hace bajo el siguiente formato:

```
variable_data_frame["nombre columna"]
```

- Para acceder a una fila de un Data Frame, se hace bajo el siguiente formato:

```
variable_data_frame.loc[número fila]
```

- Para ver las columnas y sus tipos se usa:

```
Variable_data_frame.dtypes()
```

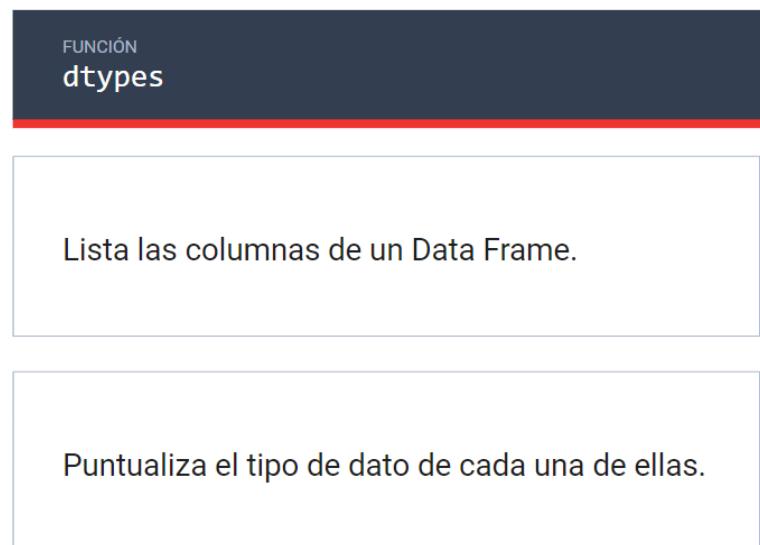
- A diferencia de un intervalo de strings, en este efectivamente se toma desde el valor inicial al final. Por ejemplo, si uno tiene un intervalo 0:4, entonces mostrará las filas con identificadores 0,1,2,3 y 4.

PROCESAMIENTO AVANZADO DE UN DATAFRAME

Introducción a la caracterización de un Data Frame:



Función dypes:



CÓDIGO	RESULTADO
<pre>print(df_clientes.dtypes)</pre>	<pre>RUT object NOMBRE object FECHA_NAC object TIPO_CLIENTE object MONTO int64 PUNTAJE_CREDITICIO float64 dtype: object</pre>

```
# Cargamos La Librería
import pandas as pd
```

```
# Cargamos el fichero
df_clientes = pd.read_csv("clientes.csv", encoding="latin-1", sep=";")
print(df_clientes.dtypes)
```

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

Función rename:

FUNCIÓN rename

Permite cambiar el nombre a una columna.

Recibe como parámetro el nombre de la columna antigua y el nombre al que se quiere cambiar.

Si tenemos un Data Frame de nombre `df`, podemos cambiar el nombre de las columnas:

```
df.rename(columns={"nombre_antiguo_columna": "nombre_nuevo_columna"})
```

CÓDIGO

```
df_clientes = df_clientes.rename(columns={"FECHA_NAC": "FECHA_NACIMIENTO"})
print(df_clientes.dtypes)
```

RESULTADO

RUT	object
NOMBRE	object
FECHA_NACIMIENTO	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	float64
dtype:	object

Es importante mantener los caracteres que aparecen en el parámetro `"columns"` de la función `rename`.

Siempre deben escribir el `{"nombre_antiguo_columna": "nombre_nuevo_columna"}`

CÓDIGO

```
df_clientes = df_clientes.rename(columns={"FECHA_NAC": "FECHA_NACIMIENTO"})
print(df_clientes.dtypes)
```

RESULTADO

RUT	object
NOMBRE	object
FECHA_NACIMIENTO	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	float64
dtype:	object

Al ejecutar la función `rename`, el resultado se asigna nuevamente a `df_clientes`, porque es necesario "guardarlo" en el Data Frame original o en alguna columna en particular.

```
df_clientes = df_clientes.rename(columns={"FECHA_NAC": "FECHA_NACIMIENTO"})
df_clientes.dtypes
```

ID	int64
RUT	object
NOMBRE	object
FECHA_NACIMIENTO	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	float64
dtype:	object

Columnas y tipo de datos:

En el caso `df_clientes` se observa que cada columna tiene un tipo de dato:

object	RUT, NOMBRE, FECHA_NAC y TIPO_CLIENTE	RESULTADO
int64	MONTO	RUT object NOMBRE object FECHA_NAC object TIPO_CLIENTE object MONTO int64 PUNTAJE_CREDITICIO float64 dtype: object
float64	PUNTAJE_CREDITICIO	

¿Qué tipos de datos puede tomar cada columna?

Tipo de dato	Resultado	Descripción
object	Es un string	Un texto
int64	Es un int	Un entero
float64	Es un float	Un decimal
bool	True o False	Valor binario, base de operaciones lógicas
datetime64	Valores fecha y tiempo	Día, hora, mes, etc.
timedelta[ns]	Diferencia entre valores de fecha y tiempo	En segundos
category	Lista finita	Valores de texto

Los tipos de datos más comúnmente usados son los 5 primeros.

Función astype – cambiar tipo a una columna:

En la siguiente situación:

Al cargar un archivo CSV, puede que los datos no se carguen correctamente. Por ejemplo, podría ocurrir que una columna que tiene decimales se cargue como texto. Es decir, la columna es de tipo `object`, y nos gustaría que pasará a ser de tipo `float64`.

Cargamos el Data Frame `df_clientes`, y la columna `PUNTAJE_CREDITICIO` se cargó como `object`, al aplicar la función `dtypes`.

CÓDIGO	RESULTADO
<pre>print(df_clientes.dtypes)</pre>	¿Pero cómo podríamos cambiar "PUNTAJE_CREDITICIO" a float64?

FUNCIÓN astype

Permite cambiar el tipo de dato de una columna.

No solo puede convertir el tipo de dato de una columna a *float64*, sino que a cualquier tipo de dato.

Si tenemos un Data Frame de nombre *df* creamos el comando general de la siguiente manera:

```
df= df[nombre columna].astype(tipo de dato al que se quiere cambiar el tipo de la columna)
```

CÓDIGO

```
print(df_clientes.dtypes)
df_clientes["PUNTAJE_CREDITICO"] = df_clientes["PUNTAJE_CREDITICO"].astype("float64")
print(df_clientes.dtypes)
```

Al aplicar la función *astype* en la columna "PUNTAJE CREDITICO", podemos cambiar el tipo de datos de "object" a "float64". No obstante, debemos volver asignarlo en la misma columna del Data Frame original. Si no lo hicieramos, y solo ejecutáramos lo que está a la derecha del "=" en la primera línea del código anterior, entonces esto no se "guarda" en ningún lado y se pierde.

RESULTADO

RUT	object
NOMBRE	object
FECHA_NAC	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	object
dtype:	object
RUT	object
NOMBRE	object
FECHA_NAC	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	float64
dtype:	object

```
df_clientes["PUNTAJE_CREDITICO"] = df_clientes["PUNTAJE_CREDITICO"].astype("float64")
print(df_clientes.dtypes)
```

ID	int64
RUT	object
NOMBRE	object
FECHA_NACIMIENTO	object
TIPO_CLIENTE	object
MONTO	int64
PUNTAJE_CREDITICO	float64
dtype:	object

Función shape:

FUNCIÓN shape

Esta función sirve para conocer la cantidad de filas y columnas de un Data Frame.

De forma general, para un Data Frame df:

df.shape	
CÓDIGO	RESULTADO
print(df_clientes.shape)	(999, 6)
La función shape indica que el Data Frame df_clientes tiene 999 filas y 6 columnas.	

```
print(df_clientes.shape)  
(1001, 7)
```

Función size:

FUNCIÓN size
Sirve para conocer la cantidad total de datos del Data Frame.

De forma general, para un Data Frame df:

df.size	
CÓDIGO	RESULTADO
print(df_clientes.size)	(5994)
La función size arroja que df_clientes tiene un total de 5994 datos, que es la multiplicación de filas y columnas.	

```
print(df_clientes.size)  
7007
```

Index de un Data Frame:

Cuando cargamos un Data Frame y lo imprimimos en consola, podemos observar lo siguiente:

RESULTADO						
RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO	
0 21.930.631-4	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17	
1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-62	A	8153651	2.37	
2 9.655.791-3	Vicente Felipe Robles Muñoz	02-02-72	E	9509104	9.91	
3 16.644.711-4	Daniela María Robles Ruiz	07-07-82	B	6665538	2.86	
4 17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024077	0.56	

Cada fila tiene un número asignado. Este número se denomina *index*, y es un identificador para cada fila.

FUNCIÓN set_index

El index es un identificador de cada fila. Para definir una columna del Data Frame como el *index*, podemos ocupar la función `set_index`.

De forma general, para setear una columna como el *index* de un Data Frame de nombre `df`:

CÓDIGO						
<pre>df = df.set_index(nombre columna)</pre>						
RESULTADO						
RUT	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17	
11.269.366-8	Cecilia Paula López Valenzuela	28-05-62	A	8153651	2.37	
9.655.791-3	Vicente Felipe Robles Muñoz	02-02-72	E	9509104	9.91	
16.644.711-4	Daniela María Robles Ruiz	07-07-82	B	6665538	2.86	
17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024077	0.56	

La columna con números enteros positivos desapareció, y en su lugar está la columna RUT.
Esta última también desapareció de la lista de columnas original.

Como aclaración, el cambio de la columna RUT al *index* del Data Frame es solo válido para este ejemplo.

ID		NOMBRE	FECHA_NACIMIENTO	\
RUT				
21.930.631-4	0	Isabel Blanca Marín Díaz	1997/4/13	
11.269.366-8	1	Cecilia Paula López Valenzuela	1962/5/28	
9.655.791-3	2	Vicente Felipe Robles Muñoz	1972/2/2	
16.644.711-4	3	Daniela María Robles Ruiz	1982/7/7	
17.054.286-6	4	Isabel Javiera Valenzuela Saavedra	1971/7/5	

12.568.934-5	996	Paula Daniela Muñoz Quiroga	1978/9/6	
14.407.999-3	997	Maria Valeria Marín Robles	1995/12/0	
20.166.403-3	998	Ignacio Bernardo López Valenzuela	1962/1/0	
10.715.550-9	999	Bernardo Pablo Saavedra Castro	1998/11/10	
16.396.396-6	1000	Javiera Ignacia Rojas Quiroga	1980/4/11	

- En esta clase aprendimos características básicas de un Data Frame. En particular, cómo caracterizar una columna o serie, y cómo editar estas características, es decir, cambiar su nombre y tipo.
- Además, aprendimos cómo caracterizar a un Data Frame, mediante el número de filas, columnas y la cantidad total de datos.

Conclusiones

- Finalmente, aprendimos que el *index* de un Data Frame es un identificador por fila, y cómo poder asignarlo de alguna de las columnas originales de un Data Frame.
- Todo esto será fundamental para el trabajo posterior. Especialmente, cuando veamos interacciones y operaciones entre Data Frames. Aquí, es muy importante saber reconocer características básicas de un Data Frame para poder hacer estas acciones.

Conclusiones

Extracción básica de datos de un Data frame:

Funciones head y tail:

FUNCIÓN head	FUNCIÓN tail
<p>Se ocupan para imprimir el contenido de un Data Frame, por sobre el uso del comando print directo en el Data Frame.</p>	

El comando general para un Data Frame de nombre **df** es:

CÓDIGO
<code>df.head()</code>

RESULTADO																																				
<table border="1"> <thead> <tr> <th>RUT</th> <th>NOMBRE</th> <th>FECHA_NAC</th> <th>TIPO_CLIENTE</th> <th>MONTO</th> <th>PUNTAJE_CREDITICIO</th> </tr> </thead> <tbody> <tr> <td>0 21.930.631-4</td> <td>Isabel Blanca Marín Díaz</td> <td>13-04-97</td> <td>C</td> <td>5407949</td> <td>1.17</td> </tr> <tr> <td>1 11.269.366-8</td> <td>Cecilia Paula López Valenzuela</td> <td>28-05-62</td> <td>A</td> <td>8153651</td> <td>2.37</td> </tr> <tr> <td>2 9.655.791-3</td> <td>Vicente Felipe Robles Muñoz</td> <td>02-02-72</td> <td>E</td> <td>9509104</td> <td>9.91</td> </tr> <tr> <td>3 16.644.711-4</td> <td>Daniela María Robles Ruiz</td> <td>07-07-82</td> <td>B</td> <td>6065538</td> <td>2.86</td> </tr> <tr> <td>4 17.054.280-6</td> <td>Isabel Javiera Valenzuela Saavedra</td> <td>05-07-71</td> <td>C</td> <td>8024877</td> <td>0.56</td> </tr> </tbody> </table>	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO	0 21.930.631-4	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17	1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-62	A	8153651	2.37	2 9.655.791-3	Vicente Felipe Robles Muñoz	02-02-72	E	9509104	9.91	3 16.644.711-4	Daniela María Robles Ruiz	07-07-82	B	6065538	2.86	4 17.054.280-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024877	0.56
RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO																															
0 21.930.631-4	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17																															
1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-62	A	8153651	2.37																															
2 9.655.791-3	Vicente Felipe Robles Muñoz	02-02-72	E	9509104	9.91																															
3 16.644.711-4	Daniela María Robles Ruiz	07-07-82	B	6065538	2.86																															
4 17.054.280-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024877	0.56																															

La función **head**, imprime por defecto las 5 primeras filas del Data Frame.

```
print(df_clientes.head())
      ID          NOMBRE FECHA_NACIMIENTO \
RUT
21.930.631-4   0      Isabel Blanca Marín Díaz  1997/4/13
11.269.366-8   1      Cecilia Paula López Valenzuela  1962/5/28
9.655.791-3   2      Vicente Felipe Robles Muñoz  1972/2/2
16.644.711-4   3      Daniela María Robles Ruiz  1982/7/7
17.054.286-6   4      Isabel Javiera Valenzuela Saavedra  1971/7/5

      TIPO_CLIENTE     MONTO PUNTAJE_CREDITICIO
RUT
21.930.631-4      C  5407949           1.17
11.269.366-8      A  8153651           2.37
9.655.791-3      E  9509104           9.91
16.644.711-4      B  6065538           2.86
17.054.286-6      C  8024077           0.56
```

El comando general para un Data Frame de nombre `df` es:

```
df.tail()

CÓDIGO
print(df_clientes.tail())

RESULTADO
```

RUT	ID	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
994	21.307.599-10	Andrea Constanza Castro Robles	10-07-92	D	9277469	2.31
995	11.942.539-4	Isidora Javiera López Rodríguez	05-01-77	E	2949763	1.17
996	12.568.934-5	Paula Daniela Muñoz Quiroga	06-09-78	B	5052388	6.01
997	14.407.999-3	Maria Valeria Marín Robles	1995-12-0	B	765834	8.69
998	20.166.403-3	Ignacio Bernardo López Valenzuela	1962-1-0	E	7822257	7.32

La función `tail` imprime por defecto las 5 últimas filas.

```
print(df_clientes.tail())
      ID          NOMBRE FECHA_NACIMIENTO \
RUT
12.568.934-5   996      Paula Daniela Muñoz Quiroga  1978/9/6
14.407.999-3   997      María Valeria Marín Robles  1995/12/0
20.166.403-3   998      Ignacio Bernardo López Valenzuela  1962/1/0
10.715.550-9   999      Bernardo Pablo Saavedra Castro  1998/11/10
16.396.396-6  1000      Javiera Ignacia Rojas Quiroga  1980/4/11

      TIPO_CLIENTE     MONTO PUNTAJE_CREDITICIO
RUT
12.568.934-5      B  5052388           6.01
14.407.999-3      B  765834            8.69
20.166.403-3      E  7822257           7.32
10.715.550-9      D  8506664           9.92
16.396.396-6      C  7853434           0.10
```

Función loc:

FUNCIÓN
loc

Sirve para extraer un grupo de filas en base a los *index* de un Data Frame. También puede ser en base a una operación lógica.

Los ejemplos que vimos de `loc` se basaban en un Data Frame que tenía como *index* números enteros positivos.

El comando general para un Data Frame de nombre `df` es:

```
df.loc[index, rango de index, o filtro]
```

RESULTADO

	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
0	21.930.631-4	Isabel Blanca María Diaz	13-04-97	C	5407949	1.17
1	11.269.366-8	Cecilia Paula López Valenzuela	28-05-62	A	8153651	2.37
2	9.600.301-3	Vicente Edilpino Morales	02-02-72	E	4056141	9.91
3	10.444.711-4	Daniela María Rojas Ruiz	07-07-92	B	60955318	2.86
4	17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	4056141	0.56
5	11.170.160-6	Vicente Vicente Marín Vergara	11-07-71	C	4056141	5.98
6	6.172.108-0	Pamela Isabel Castro Vergara	11-05-76	C	3061858	3.09
7	15.844.106-2	Daniela Pamela Saavedra Vergara	14-06-79	C	5197540	2.26
8	20.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18

Observamos que cada fila tiene asociado un `index` que es un número entero positivo.

CÓDIGO

```
print(df_clientes.loc[5])
```

RESULTADO

RUT	Vicente Vicente Marín Vergara	11.170.160-6
NOMBRE	Vicente Vicente Marín Vergara	11-07-71
FECHA_NAC		C
TIPO_CLIENTE		4056141
MONTO		5.98
PUNTAJE_CREDITICIO		Name: 5, dtype: object

En este caso imprimimos en consola la fila con `index` 5.

```
print(df_clientes.loc["20.166.403-3"])
```

ID	998
NOMBRE	Ignacio Bernardo López Valenzuela
FECHA_NACIMIENTO	1962/1/0
TIPO_CLIENTE	E
MONTO	7822257
PUNTAJE_CREDITICIO	7.32
Name: 20.166.403-3, dtype: object	

CÓDIGO

```
print(df_clientes.loc[5:9])
```

RESULTADO

	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
5	11.170.160-6	Vicente Vicente Marín Vergara	11-07-71	C	4056141	5.98
6	6.172.108-0	Pamela Isabel Castro Vergara	11-05-76	C	3061858	3.09
7	15.844.106-2	Daniela Pamela Saavedra Vergara	14-06-79	C	5197540	2.26
8	20.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18
9	15.910.648-9	Bernardo Ignacio Quiroga Muñoz	03-02-68	B	8613487	3.44

En este caso imprimimos en consola las filas con `index` del 5 al 9.

CÓDIGO

```
print(df_clientes.loc["11.170.160-6":"15.910.648-9"])
```

RESULTADO

	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
	11.170.160-6	Vicente Vicente Marín Vergara	11-07-71	C	4056141	5.98
	6.172.108-0	Pamela Isabel Castro Vergara	11-05-76	C	3061858	3.09
	15.844.106-2	Daniela Pamela Saavedra Vergara	14-06-79	C	5197540	2.26
	20.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18
	15.910.648-9	Bernardo Ignacio Quiroga Muñoz	03-02-68	B	8613487	3.44

Imprimimos en consola las filas con `index` del "11.170.160-6" al "15.910.648-9".

Es importante notar que los `index` no necesariamente son únicos.

Entonces si es que se aplica la función `loc` sobre `index` que están en más de una fila, Python arrojará error.

CÓDIGO

```
print(df_clientes.loc[df_clientes["TIPO_CLIENTE"]=="C"])
```

RESULTADO

	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICIO
	21.930.631-4	Isabel Blanca María Diaz	13-04-97	C	5407949	1.17
	17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	4056141	0.56
	11.170.160-6	Vicente Vicente Marín Vergara	11-07-71	C	4056141	5.98
	6.172.108-0	Pamela Isabel Castro Vergara	11-05-76	C	3061858	3.09
	15.844.106-2	Daniela Pamela Saavedra Vergara	14-06-79	C	5197540	2.26

En este ejemplo filtramos los clientes que tienen tipo de cliente "C".

```

print(df_clientes.loc[df_clientes["TIPO_CLIENTE"] == "C"])

      ID          NOMBRE FECHA_NACIMIENTO \
RUT
21.930.631-4    0   Isabel Blanca Marín Díaz  1997/4/13
17.054.286-6    4   Isabel Javiera Valenzuela Saavedra  1971/7/5
11.170.160-6    5   Vicente Vicente Marín Vergara  1971/7/11
6.172.108-0     6   Pamela Isabel Castro Vergara  1976/5/11
15.844.106-2    7   Daniela Pamela Saavedra Vergara  1979/6/14
...
11.528.775-7   968   Ignacia Ignacia Vergara Robles  1963/5/1
20.453.595-9   979   María Ignacia Rojas Rodríguez  1970/10/8
10.892.212-9   982   Isidora Valeria González Rodríguez  1996/7/13
10.667.321-1   983   Pedro Ramón Valenzuela Rodríguez  1966/2/11
16.396.396-6  1000   Javiera Ignacia Rojas Quiroga  1980/4/11

```

Función iloc:

¿Cómo podríamos filtrar en base al índice de cada fila (la posición del Data Frame), si es que cambiamos el *index* a un texto como “RUT” por ejemplo?.

Con esta función podemos filtrar por los índices de las filas, aunque el *index* no sea un número entero positivo.

CÓDIGO

```
print(df_clientes.iloc[5])
```

RESULTADO

RUT	11.170.160-6
NOMBRE	Vicente Vicente Marín Vergara
FECHA_NAC	11-07-71
TIPO_CLIENTE	C
MONTO	4056141
PUNTAJE_CREDITICO	5.98
Name: 5, dtype: object	

En este caso imprimimos en consola la fila de índice 5.

```
print(df_clientes.iloc[5])
```

ID	5
NOMBRE	Vicente Vicente Marín Vergara
FECHA_NACIMIENTO	1971/7/11
TIPO_CLIENTE	C
MONTO	4056141
PUNTAJE_CREDITICO	5.98
Name: 11.170.160-6, dtype: object	

El comando general para *df* es:

```
df.iloc[índice de fila o rango de indices de fila]
```

CÓDIGO

```
print(df_clientes.iloc[5:9])
```

RESULTADO

RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO
11.170.160-6	Vicente Vicente Marín Vergara	11-07-71	C	4056141	5.98
6.172.108-0	Pamela Isabel Castro Vergara	11-05-76	C	3061858	3.09
15.844.106-2	Daniela Pamela Saavedra Vergara	14-06-79	C	5197540	2.26
26.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18

Este caso es bastante interesante ya que a diferencia de *loc*, tenemos una fila menos. Esto se debe a que *iloc* sigue una lógica similar al extraer un conjunto de caracteres de un *string*. Es decir, si tenemos el rango *n:m*, en realidad extrae los valores de *n* a *m-1*.

```
print(df_clientes.iloc[5:9])
```

RUT	ID	NOMBRE	FECHA_NACIMIENTO
11.170.160-6	5	Vicente Vicente Marín Vergara	1971/7/11
6.172.108-0	6	Pamela Isabel Castro Vergara	1976/5/11
15.844.106-2	7	Daniela Pamela Saavedra Vergara	1979/6/14
20.749.832-5	8	Felipe Andrés Rodríguez Valenzuela	1973/2/9

Manejo de datos faltantes:

¿Cómo manejar los datos faltantes?

1 Muchas veces, la información que manejamos contiene datos faltantes.

2 Por ejemplo, si la información fue cargada por un ser humano en algún sistema de información, es altamente probable que algunos datos falten.

Es vital conocer las herramientas que posee Pandas para manejar estos casos.

Veamos un ejemplo alternativo con `df_clientes` donde hay muchos datos faltantes:

RESULTADO						
	RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO
0	20.137.631-2		NaN	1962/3/24	C 5463594,0	5.95
1	21.276.271-7	Ignacio Gabriel Quiroga Quiroga	NaN	D 5767794,0	0.31	
2	10.175.351-8	Francisco Alejandro Castro Valenzuela	1992/6/2	D 8360849,0	NaN	
3	11.107.732-2	Paula Javiera Campos Rojas	1999/8/5	D 3541238,0	7.35	
4	14.564.298-9	Andrés Bernardo Campos Quiroga	1961/9/15	E 1709146,0	7.99	
5	11.678.049-8	Pedro Javier Vergara Castro	NaN	A 6336266,0	8.50	
6	6.570.081-4	Gabriel Ramón Muñoz Robles	1984/3/2	D 1718269,0	5.21	
7	19.713.565-3	Ramón Juan Robles Vergara	1964/7/14	D 4552090,0	8.15	
8	18.222.859-3	Victoria Ignacia Muñoz Rodríguez	1971/4/24	D 8166157,0	9.04	
9	4.018.889-8	Felipe Alejandro Valenzuela Vergara	1992/12/13	A 6801606,0	0.65	
10	18.580.449-4	Ramón Andrés González González	1997/4/1	NaN 3133436,0	7.46	

Al cargar este archivo CSV en Pandas, automáticamente los datos faltantes se almacenan como `NaN`.

Función `fillna`:

FUNCIÓN `fillna`

Sirve para llenar los valores `NaN` podemos ocupar la función `fillna`.

Esta función reemplaza los `NaN` por el valor que se entregue como parámetro.

El comando general para `df` es:

df = df.fillna(valor con el que se quiere reemplazar los valores NaN)																																																							
CÓDIGO																																																							
<pre>df_clientes = df_clientes.fillna("SIN_INFO") print(df_clientes)</pre>																																																							
RESULTADO																																																							
<table border="1"> <thead> <tr> <th>RUT</th><th>NOMBRE</th><th>FECHA_NAC</th><th>TIPO_CLIENTE</th><th>MONTO</th><th>PUNTAJE_CREDITICO</th><th></th></tr> </thead> <tbody> <tr> <td>0 20.137.631-2</td><td>SIN_INFO</td><td>1962/5/24</td><td>C</td><td>5.4635e+06</td><td>5.95</td><td></td></tr> <tr> <td>1 21.276.271-7</td><td>Ignacio Gabriel Quiroga Quiroga</td><td>SIN INFO</td><td>D</td><td>5.76779e+06</td><td>6.31</td><td></td></tr> <tr> <td>2 10.175.351-8</td><td>Francisco Alejandro Castro Valenzuela</td><td>1992/6/2</td><td>D</td><td>8.36085e+06</td><td></td><td></td></tr> <tr> <td>3 11.107.732-2</td><td>Paula Javiera Campos Rojas</td><td>1999/8/5</td><td>D</td><td>3.54124e+06</td><td>7.35</td><td></td></tr> <tr> <td>4 14.564.298-9</td><td>Andrés Bernardo Campos Quiroga</td><td>1991/5/15</td><td>E</td><td>1.70915e+06</td><td>7.99</td><td></td></tr> <tr> <td>5 11.878.049-8</td><td>Pedro Javier Vergara Castro</td><td>SIN_INFO</td><td>A</td><td>6.33627e+06</td><td>8.5</td><td></td></tr> </tbody> </table>							RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO		0 20.137.631-2	SIN_INFO	1962/5/24	C	5.4635e+06	5.95		1 21.276.271-7	Ignacio Gabriel Quiroga Quiroga	SIN INFO	D	5.76779e+06	6.31		2 10.175.351-8	Francisco Alejandro Castro Valenzuela	1992/6/2	D	8.36085e+06			3 11.107.732-2	Paula Javiera Campos Rojas	1999/8/5	D	3.54124e+06	7.35		4 14.564.298-9	Andrés Bernardo Campos Quiroga	1991/5/15	E	1.70915e+06	7.99		5 11.878.049-8	Pedro Javier Vergara Castro	SIN_INFO	A	6.33627e+06	8.5	
RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO																																																		
0 20.137.631-2	SIN_INFO	1962/5/24	C	5.4635e+06	5.95																																																		
1 21.276.271-7	Ignacio Gabriel Quiroga Quiroga	SIN INFO	D	5.76779e+06	6.31																																																		
2 10.175.351-8	Francisco Alejandro Castro Valenzuela	1992/6/2	D	8.36085e+06																																																			
3 11.107.732-2	Paula Javiera Campos Rojas	1999/8/5	D	3.54124e+06	7.35																																																		
4 14.564.298-9	Andrés Bernardo Campos Quiroga	1991/5/15	E	1.70915e+06	7.99																																																		
5 11.878.049-8	Pedro Javier Vergara Castro	SIN_INFO	A	6.33627e+06	8.5																																																		

En este caso rellenamos los valores `NaN` por `SIN_INFO`.

`df_clientes = df_clientes.fillna("SIN_INFO")`

Función append:

Otro tipo de información faltante en un Data Frame pueden ser filas. Para agregar más filas a un Data Frame, lo podemos hacer mediante la función `append`.

Es necesario que al agregar estas filas tengan exactamente las mismas columnas que el Data Frame original.

Si tenemos el Data Frame `df` y el Data Frame `df2` con las mismas columnas, podemos agregar las filas de `df2` a `df` de la siguiente manera:

df = df.append(df2)																																																																																																																						
RESULTADO																																																																																																																						
<table border="1"> <tbody> <tr> <td>983 10.667.321-1</td><td>Pedro Ramón Valenzuela Rodríguez</td><td>11-02-66</td><td>C</td><td>3912129</td><td>2.53</td><td></td></tr> <tr> <td>984 12.395.690-6</td><td>Maria Isidora Robles Rodríguez</td><td>01-03-88</td><td>B</td><td>3672381</td><td>7.71</td><td></td></tr> <tr> <td>985 12.395.690-6</td><td>Maria Isidora Robles Rodríguez</td><td>01-03-88</td><td>D</td><td>3672381</td><td>6.34</td><td></td></tr> <tr> <td>986 13.073.200-2</td><td>Felipe Javier Rodriguez Rojas</td><td>22-05-61</td><td>D</td><td>6366008</td><td>3.24</td><td></td></tr> <tr> <td>987 14.129.210-7</td><td>Vicente Juan Castro Castro</td><td>10-10-72</td><td>D</td><td>7118017</td><td>5.85</td><td></td></tr> <tr> <td>988 16.986.836-10</td><td>Javiera María Vergara Valenzuela</td><td>12-01-89</td><td>A</td><td>3352147</td><td>8.92</td><td></td></tr> <tr> <td>989 16.986.836-10</td><td>Javiera María Vergara Valenzuela</td><td>12-01-89</td><td>E</td><td>1405207</td><td>1.59</td><td></td></tr> <tr> <td>990 15.269.026-6</td><td>Vicente Juan Castro Castro</td><td>10-10-72</td><td>D</td><td>4423556</td><td>8.33</td><td></td></tr> <tr> <td>991 16.148.625-10</td><td>Blanca Isabel Vergara González</td><td>20-07-73</td><td>A</td><td>3097448</td><td>9.98</td><td></td></tr> <tr> <td>992 7.700.000-4</td><td>Isidora Cecilia Saavedra Vergara</td><td>08-06-78</td><td>E</td><td>3112980</td><td>1.92</td><td></td></tr> <tr> <td>993 13.382.402-6</td><td>Javiera Cecilia Saavedra Vergara</td><td>08-06-73</td><td>E</td><td>2098637</td><td>0.38</td><td></td></tr> <tr> <td>994 21.387.399-10</td><td>Andrea Constanza Castro Robles</td><td>10-07-92</td><td>D</td><td>9277469</td><td>2.31</td><td></td></tr> <tr> <td>995 11.042.539-4</td><td>Isidora Cecilia Saavedra Vergara</td><td>09-01-77</td><td>E</td><td>2949763</td><td></td><td></td></tr> <tr> <td>996 21.387.399-10</td><td>Isidora Cecilia Saavedra Vergara</td><td>09-01-77</td><td>E</td><td>2949763</td><td></td><td></td></tr> <tr> <td>997 14.407.999-3</td><td>Maria Valeria Marin Robles</td><td>1995-12-0</td><td>B</td><td>765834</td><td>8.69</td><td></td></tr> <tr> <td>998 20.166.403-3</td><td>Ignacio Bernardo López Valenzuela</td><td>1962-1-0</td><td>E</td><td>7822257</td><td>7.32</td><td></td></tr> </tbody> </table>							983 10.667.321-1	Pedro Ramón Valenzuela Rodríguez	11-02-66	C	3912129	2.53		984 12.395.690-6	Maria Isidora Robles Rodríguez	01-03-88	B	3672381	7.71		985 12.395.690-6	Maria Isidora Robles Rodríguez	01-03-88	D	3672381	6.34		986 13.073.200-2	Felipe Javier Rodriguez Rojas	22-05-61	D	6366008	3.24		987 14.129.210-7	Vicente Juan Castro Castro	10-10-72	D	7118017	5.85		988 16.986.836-10	Javiera María Vergara Valenzuela	12-01-89	A	3352147	8.92		989 16.986.836-10	Javiera María Vergara Valenzuela	12-01-89	E	1405207	1.59		990 15.269.026-6	Vicente Juan Castro Castro	10-10-72	D	4423556	8.33		991 16.148.625-10	Blanca Isabel Vergara González	20-07-73	A	3097448	9.98		992 7.700.000-4	Isidora Cecilia Saavedra Vergara	08-06-78	E	3112980	1.92		993 13.382.402-6	Javiera Cecilia Saavedra Vergara	08-06-73	E	2098637	0.38		994 21.387.399-10	Andrea Constanza Castro Robles	10-07-92	D	9277469	2.31		995 11.042.539-4	Isidora Cecilia Saavedra Vergara	09-01-77	E	2949763			996 21.387.399-10	Isidora Cecilia Saavedra Vergara	09-01-77	E	2949763			997 14.407.999-3	Maria Valeria Marin Robles	1995-12-0	B	765834	8.69		998 20.166.403-3	Ignacio Bernardo López Valenzuela	1962-1-0	E	7822257	7.32	
983 10.667.321-1	Pedro Ramón Valenzuela Rodríguez	11-02-66	C	3912129	2.53																																																																																																																	
984 12.395.690-6	Maria Isidora Robles Rodríguez	01-03-88	B	3672381	7.71																																																																																																																	
985 12.395.690-6	Maria Isidora Robles Rodríguez	01-03-88	D	3672381	6.34																																																																																																																	
986 13.073.200-2	Felipe Javier Rodriguez Rojas	22-05-61	D	6366008	3.24																																																																																																																	
987 14.129.210-7	Vicente Juan Castro Castro	10-10-72	D	7118017	5.85																																																																																																																	
988 16.986.836-10	Javiera María Vergara Valenzuela	12-01-89	A	3352147	8.92																																																																																																																	
989 16.986.836-10	Javiera María Vergara Valenzuela	12-01-89	E	1405207	1.59																																																																																																																	
990 15.269.026-6	Vicente Juan Castro Castro	10-10-72	D	4423556	8.33																																																																																																																	
991 16.148.625-10	Blanca Isabel Vergara González	20-07-73	A	3097448	9.98																																																																																																																	
992 7.700.000-4	Isidora Cecilia Saavedra Vergara	08-06-78	E	3112980	1.92																																																																																																																	
993 13.382.402-6	Javiera Cecilia Saavedra Vergara	08-06-73	E	2098637	0.38																																																																																																																	
994 21.387.399-10	Andrea Constanza Castro Robles	10-07-92	D	9277469	2.31																																																																																																																	
995 11.042.539-4	Isidora Cecilia Saavedra Vergara	09-01-77	E	2949763																																																																																																																		
996 21.387.399-10	Isidora Cecilia Saavedra Vergara	09-01-77	E	2949763																																																																																																																		
997 14.407.999-3	Maria Valeria Marin Robles	1995-12-0	B	765834	8.69																																																																																																																	
998 20.166.403-3	Ignacio Bernardo López Valenzuela	1962-1-0	E	7822257	7.32																																																																																																																	

[999 rows x 6 columns]

En este caso, tenemos el archivo `clientes.csv` y el archivo `clientes2.csv`. Ambos tienen las mismas columnas (las mismas que `df_clientes`), por lo que podemos unirlos.

Inicialmente, el Data Frame que representa a `clientes.csv` se ve de esta manera:

Si tenemos el Data Frame `df` y el Data Frame `df2` con las mismas columnas, podemos agregar las filas de `df2` a `df` de la siguiente manera:

df = df.append(df2)																																																																																																																																											
RESULTADO																																																																																																																																											
<table border="1"> <tbody> <tr> <td>0 21.930.831-4</td><td>Isabel Blanca Marín Díaz</td><td>13-04-97</td><td>C</td><td>5407949</td><td>1.17</td><td></td></tr> <tr> <td>1 11.269.366-8</td><td>Cecilia Paula López Valenzuela</td><td>28-05-82</td><td>A</td><td>8159551</td><td>2.37</td><td></td></tr> <tr> <td>2 9.325.000-5</td><td>Isidora Cecilia Robles Rodríguez</td><td>20-09-72</td><td>E</td><td>2949763</td><td>9.91</td><td></td></tr> <tr> <td>3 16.644.711-4</td><td>Danielita María Robles Ruiz</td><td>07-07-82</td><td>B</td><td>6065538</td><td>2.48</td><td></td></tr> <tr> <td>4 17.054.286-6</td><td>Isabel Javiera Valenzuela Saavedra</td><td>05-07-71</td><td>C</td><td>8024077</td><td>0.56</td><td></td></tr> <tr> <td>5 11.170.160-0</td><td>Vicente Vicente Martín Vergara</td><td>11-07-71</td><td>C</td><td>4056141</td><td>5.98</td><td></td></tr> <tr> <td>6 11.170.160-0</td><td>Danielita Irene Vergara</td><td>08-06-78</td><td>C</td><td>3039001</td><td>3.89</td><td></td></tr> <tr> <td>7 15.844.106-2</td><td>Danielita Pamela Saavedra Vergara</td><td>14-06-79</td><td>C</td><td>5197548</td><td>2.26</td><td></td></tr> <tr> <td>8 20.749.832-5</td><td>Felipe Andrés Rodríguez Valenzuela</td><td>09-02-73</td><td>E</td><td>4441737</td><td>8.18</td><td></td></tr> <tr> <td>9 14.407.999-3</td><td>Bernardo López Valenzuela</td><td>1995-12-0</td><td>B</td><td>765834</td><td>8.44</td><td></td></tr> <tr> <td>10 9.487.844-9</td><td>Taviar Juan Castro Campos</td><td>1989/3/10</td><td>B</td><td>8030691</td><td>2.70</td><td></td></tr> <tr> <td>11 7.271.618-0</td><td>Isidora Andres Rodríguez Valenzuela</td><td>29-09-76</td><td>B</td><td>5832527</td><td>8.08</td><td></td></tr> <tr> <td>12 13.674.785-2</td><td>Javier Juan Robles Robles</td><td>23-07-73</td><td>A</td><td>469341</td><td>2.81</td><td></td></tr> <tr> <td>13 13.674.785-2</td><td>Ramón Juan Robles Robles</td><td>23-07-73</td><td>E</td><td>848530</td><td>5.98</td><td></td></tr> <tr> <td>14 20.481.183-10</td><td>Andrés Francisco López López</td><td>27-07-77</td><td>E</td><td>6289039</td><td>4.68</td><td></td></tr> <tr> <td>15 6.355.671-0</td><td>Rodrigo Andrés Valenzuela Valenzuela</td><td>01-10-70</td><td>B</td><td>6681186</td><td>9.84</td><td></td></tr> <tr> <td>16 15.452.563-10</td><td>Gabriel Vicente Castro Robles</td><td>11-06-85</td><td>E</td><td>7795079</td><td>0.21</td><td></td></tr> <tr> <td>17 15.452.563-10</td><td>Felipe Andrés Robles Rodríguez</td><td>06-05-70</td><td>E</td><td>3156647</td><td>7.26</td><td></td></tr> <tr> <td>18 8.471.223-1</td><td></td><td></td><td>E</td><td>7976926</td><td>9.06</td><td></td></tr> </tbody> </table>							0 21.930.831-4	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17		1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-82	A	8159551	2.37		2 9.325.000-5	Isidora Cecilia Robles Rodríguez	20-09-72	E	2949763	9.91		3 16.644.711-4	Danielita María Robles Ruiz	07-07-82	B	6065538	2.48		4 17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024077	0.56		5 11.170.160-0	Vicente Vicente Martín Vergara	11-07-71	C	4056141	5.98		6 11.170.160-0	Danielita Irene Vergara	08-06-78	C	3039001	3.89		7 15.844.106-2	Danielita Pamela Saavedra Vergara	14-06-79	C	5197548	2.26		8 20.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18		9 14.407.999-3	Bernardo López Valenzuela	1995-12-0	B	765834	8.44		10 9.487.844-9	Taviar Juan Castro Campos	1989/3/10	B	8030691	2.70		11 7.271.618-0	Isidora Andres Rodríguez Valenzuela	29-09-76	B	5832527	8.08		12 13.674.785-2	Javier Juan Robles Robles	23-07-73	A	469341	2.81		13 13.674.785-2	Ramón Juan Robles Robles	23-07-73	E	848530	5.98		14 20.481.183-10	Andrés Francisco López López	27-07-77	E	6289039	4.68		15 6.355.671-0	Rodrigo Andrés Valenzuela Valenzuela	01-10-70	B	6681186	9.84		16 15.452.563-10	Gabriel Vicente Castro Robles	11-06-85	E	7795079	0.21		17 15.452.563-10	Felipe Andrés Robles Rodríguez	06-05-70	E	3156647	7.26		18 8.471.223-1			E	7976926	9.06	
0 21.930.831-4	Isabel Blanca Marín Díaz	13-04-97	C	5407949	1.17																																																																																																																																						
1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-82	A	8159551	2.37																																																																																																																																						
2 9.325.000-5	Isidora Cecilia Robles Rodríguez	20-09-72	E	2949763	9.91																																																																																																																																						
3 16.644.711-4	Danielita María Robles Ruiz	07-07-82	B	6065538	2.48																																																																																																																																						
4 17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8024077	0.56																																																																																																																																						
5 11.170.160-0	Vicente Vicente Martín Vergara	11-07-71	C	4056141	5.98																																																																																																																																						
6 11.170.160-0	Danielita Irene Vergara	08-06-78	C	3039001	3.89																																																																																																																																						
7 15.844.106-2	Danielita Pamela Saavedra Vergara	14-06-79	C	5197548	2.26																																																																																																																																						
8 20.749.832-5	Felipe Andrés Rodríguez Valenzuela	09-02-73	E	4441737	8.18																																																																																																																																						
9 14.407.999-3	Bernardo López Valenzuela	1995-12-0	B	765834	8.44																																																																																																																																						
10 9.487.844-9	Taviar Juan Castro Campos	1989/3/10	B	8030691	2.70																																																																																																																																						
11 7.271.618-0	Isidora Andres Rodríguez Valenzuela	29-09-76	B	5832527	8.08																																																																																																																																						
12 13.674.785-2	Javier Juan Robles Robles	23-07-73	A	469341	2.81																																																																																																																																						
13 13.674.785-2	Ramón Juan Robles Robles	23-07-73	E	848530	5.98																																																																																																																																						
14 20.481.183-10	Andrés Francisco López López	27-07-77	E	6289039	4.68																																																																																																																																						
15 6.355.671-0	Rodrigo Andrés Valenzuela Valenzuela	01-10-70	B	6681186	9.84																																																																																																																																						
16 15.452.563-10	Gabriel Vicente Castro Robles	11-06-85	E	7795079	0.21																																																																																																																																						
17 15.452.563-10	Felipe Andrés Robles Rodríguez	06-05-70	E	3156647	7.26																																																																																																																																						
18 8.471.223-1			E	7976926	9.06																																																																																																																																						

Al cargar el archivo `clientes2.csv` en un Data Frame de nombre `df_clientes2`, se ve de esta manera (tiene solo 18 filas).

```
CÓDIGO
df_clientes = df_clientes.append(df_clientes2)
print(df_clientes)
```

RESULTADO

ID	NOMBRE	FECHA_NACIMIENTO
994	Andrés Constantino Castro Robles	18-07-02
995	Isidora Andrea Rodríguez Valenzuela	29-09-77
996	Paula Daniela Huizar Quiroga	06-09-78
997	Maria Valeria Marin Robles	1995-12-8
998	Ignacio Ignacio Quiroga Muñoz	17-02-92
9	Isabel Blanca Marin Diaz	13-04-97
1	Cecilia Paula López Valenzuela	28-05-02
2	Vicente Felipe Robles Muñoz	02-07-72
3	Daniela María Robles Ruiz	07-07-02
4	Isabel Javiera Valenzuela Saavedra	05-07-71
5	Vicente Felipe Robles Muñoz	02-07-72
6	Paula Isabel Castro Vergara	11-05-76
7	Gabriel Vicente Castro Robles	14-06-79
8	Felipe Andrés Robles Valenzuela	05-07-77
9	Bernardo Ignacio Quiroga Muñoz	03-02-98
10	Javier Juan Castro Campos	1985/3/8
11	Isidora Andrea Rodríguez Valenzuela	29-09-77
12	Javier Juan Robles Robles	23-07-73
13	Ramón Vicente Vergara Robles	16-07-97
14	Andrés Francisco López López	SIN_INFO
15	Rodrigo Andrés Valenzuela Valenzuela	SIN_INFO
16	Andrés Felipe Quiroga Robles	SIN_INFO
17	Gabriel Vicente Castro Robles	SIN_INFO
18	Felipe Andrés Robles Rodríguez	SIN_INFO

[1018 rows x 6 columns]

Notemos que si hacemos append podría ocurrir tener index repetidos.

```
df_clientes2 = pd.read_csv("clientes2.csv", encoding="latin-1", sep=";")
df_clientes = df_clientes.append(df_clientes2)
df_clientes = df_clientes.fillna("SIN_INFO")
print(df_clientes)
```

ID	NOMBRE	FECHA_NACIMIENTO
21.930.631-4	Isabel Blanca Marín Díaz	1997/4/13
11.269.366-8	Cecilia Paula López Valenzuela	1962/5/28
9.655.791-3	Vicente Felipe Robles Muñoz	1972/2/2
16.644.711-4	Daniela María Robles Ruiz	1982/7/7
17.054.286-6	Isabel Javiera Valenzuela Saavedra	1971/7/5
...
14	Andrés Francisco López López	SIN_INFO
15	Rodrigo Andrés Valenzuela Valenzuela	SIN_INFO
16	Andrés Felipe Quiroga Robles	SIN_INFO
17	Gabriel Vicente Castro Robles	SIN_INFO
18	Felipe Andrés Robles Rodríguez	SIN_INFO

Función drop: Permite eliminar filas.

De forma general, y para un Data Frame de nombre `df`:

```
df = df.drop(df.iloc[filas o rango de filas].index)
```

```
CÓDIGO
df_clientes = df_clientes.drop(df_clientes.iloc[0:10].index)
print(df_clientes)
```

RESULTADO

RUT	NOMBRE	FECHA_NAC	TIPO_CLIENTE	MONTO	PUNTAJE_CREDITICO
0 21.930.631-4	Isabel Blanca Marín Díaz	13-04-97	C	5487949	1.17
1 11.269.366-8	Cecilia Paula López Valenzuela	28-05-02	A	8153651	2.37
2 9.655.791-3	Vicente Felipe Robles Muñoz	02-07-72	E	5832527	9.05
3 16.644.711-4	Daniela María Robles Ruiz	07-07-02	B	6965538	2.86
4 17.054.286-6	Isabel Javiera Valenzuela Saavedra	05-07-71	C	8824077	0.56
...					
10 9.487.844-9	Javier Juan Castro Campos	1985-3-8	B	8830691	2.70
11 7.271.618-0	Isidora Andrea Rodríguez Valenzuela	29-09-76	B	5832527	8.08
12 13.674.785-2	Javier Juan Robles Robles	23-07-73	A	8845556	2.01
13 17.452.036-6	Ramón Vicente Vergara Robles	16-07-97	E	8845556	5.98
14 20.481.183-10	Andrés Francisco López López	27-07-77	E	6285839	4.68

Notemos que para eliminar filas, especificamos el intervalo de índices de las filas que queremos eliminar dentro del `iloc`. Todo lo otro se debe mantener igual.

```
df_clientes = df_clientes.drop(df_clientes.iloc[0:10].index)
```

```
print(df_clientes)
```

ID	NOMBRE
9.487.844-9	Javier Juan Castro Campos
7.271.618-0	Isidora Andrea Rodríguez Valenzuela
13.674.785-2	Javier Juan Robles Robles
17.452.036-6	Ramón Vicente Vergara Robles
20.481.183-10	Andrés Francisco López López

Caso práctico módulo 2:

Pasos a seguir

1

Carga el archivo “**consultas.csv**” en un Data Frame . Debes nombrarlo **df_consultas**.

Este archivo tiene las siguientes columnas:

- a) **Fecha_Hora_Consulta**: Fecha en que se solicitó la consulta.
- b) **RUT**: RUT del paciente.
- c) **Especialidad**: Especialidad de la consulta.
- d) **Diag**: Diagnóstico del paciente.
- e) **RUT2**: RUT del médico tratante.
- f) **Costo_consulta**: Cuánto costó la consulta (particular, sin bono).

```
import pandas as pd
```

```
df_consultas = pd.read_csv("consultas.csv", encoding="latin-1", sep=";")  
df_consultas.dtypes
```

```
Fecha_Hora_Consulta    object  
RUT                  object  
Especialidad          object  
Diag                 object  
RUT2                 object  
Costo_consulta        float64  
dtype: object
```

2

La columna **Fecha_Hora_Consulta** se cargó con tipo **object**, debes transformarla a formato **datetime**.

3

Mediante las funciones **shape** y **size** imprime en consola las filas, columnas y cantidad total de datos de **df_consultas**.

4

Cambia el **index** del Data Frame a la columna **RUT**.

5

Muestra las 10 primeras y últimas filas de **df_consultas** mediante las funciones **head** y **tail**.

```
# Paso 2  
df_consultas["Fecha_Hora_Consulta"] = df_consultas["Fecha_Hora_Consulta"].astype("datetime64")  
df_consultas.dtypes
```

```
Fecha_Hora_Consulta    datetime64[ns]  
RUT                  object  
Especialidad          object  
Diag                 object  
RUT2                 object  
Costo_consulta        float64  
dtype: object
```

```
# Paso 3  
print(df_consultas.shape)  
print(df_consultas.size)
```

```
(200, 6)  
1200
```

```
# Paso 4  
df_consultas = df_consultas.set_index("RUT")
```

```
# Paso 5  
print(df_consultas.head(10))
```

RUT	Fecha_Hora_Consulta	Especialidad	Diag
15.439.959-0	2018-11-15	CardiologÃa	Enfermedad 6
6.369.292-4	2018-02-27	Medicina General	Enfermedad 0
16.566.679-4	2018-03-02	CardiologÃa	Enfermedad 6
18.124.934-0	2018-09-18	CardiologÃa	Enfermedad 6
18.993.070-0	2018-08-27	CardiologÃa	Enfermedad 6
16.155.602-8	2018-07-09	DermatologÃa	Enfermedad 3
21.553.711-7	2018-07-26	CardiologÃa	Enfermedad 6
12.813.415-7	2018-04-21	DermatologÃa	Enfermedad 3
18.434.658-2	2018-06-08	DermatologÃa	Enfermedad 3
8.258.073-8	2018-01-22	CardiologÃa	Enfermedad 6

```
print(df_consultas.tail(10))
```

RUT	Fecha_Hora_Consulta	Especialidad	Diag
11.660.077-1	2018-05-01	Medicina General	Enfermedad 0
6.926.940-5	2018-10-31	DermatologÃa	Enfermedad 3
18.782.070-4	2018-11-30	DermatologÃa	Enfermedad 3
17.779.449-4	2018-12-07	DermatologÃa	Enfermedad 3
16.389.122-7	2018-04-29	DermatologÃa	Enfermedad 3
4.464.429-4	2018-12-03	CardiologÃa	Enfermedad 6
9.097.028-5	2018-11-09	DermatologÃa	Enfermedad 3
21.220.974-4	2018-10-06	Medicina General	Enfermedad 0
10.742.393-7	2018-09-13	Medicina General	Enfermedad 0
6.033.112-2	2018-01-10	DermatologÃa	Enfermedad 3

6

Muestra la siguiente información:

- a) La fila con **RUT** igual al **6.079.686-2**.
- b) La fila con **RUT** **4.367.282-7** al **19.500.328-3**.
- c) Las filas de **índice 23**.
- d) Las filas de **índice 76 al 89**.

7

Cambia el nombre a la columna “**Fecha_Hora_Consulta**” por “**Fecha_Consulta**”. También cambia el nombre de la columna “**RUT2**” por “**RUT_medico**”.

```
# Paso 6  
# a)  
print(df_consultas.loc["6.079.686-2"])  
# b)  
print(df_consultas.loc["4.367.282-7":"19.500.328-3"])  
# c)  
print(df_consultas.iloc[23])  
# d)  
print(df_consultas.iloc[76:90])
```

```

# Paso 7
df_consultas = df_consultas.rename(columns={"Fecha_Hora_Consulta":"Fecha_Consulta"})
df_consultas = df_consultas.rename(columns={"RUT2":"RUT_medico"})
print(df_consultas.dtypes)

Fecha_Consulta      datetime64[ns]
Especialidad        object
Diag                object
RUT_medico          object
Costo_consulta      float64
dtype: object

#?
df_consultas = df_consultas.rename(columns={"Fecha_Hora_Consulta":"Fecha_Consulta","RUT2":"RUT_medico"})
print(df_consultas.dtypes)

```

- 8** Lamentablemente, esta base de datos fue recopilada manualmente. Puedes notar que algunos datos de la columna “**Costo_consulta**” tienen un **NaN**. En este caso el gerente decidió guardar **\$10.000**, que es el costo mínimo de la consulta.
- 9** Además, el gerente te acaba de notificar que faltó la información de varias consultas. Están en el archivo “**consultas2.csv**”. Cárgala en Python y únela a **df_consultas**.
- 10** Finalmente, borra las filas de **índice 0 al 10**, ya que en esas fechas la clínica UCO estuvo cerrada entonces probablemente fue un error de digitación al crear estas bases de datos.

```

print(df_consultas)
df_consultas = df_consultas.fillna("10000")
print(df_consultas)

#9
df_consultas2 = pd.read_csv("consultas2.csv",sep=";")
print(df_consultas2)

df_consultas2["Fecha_Hora_Consulta"] = df_consultas2["Fecha_Hora_Consulta"].astype("datetime64")
df_consultas2 = df_consultas2.set_index("RUT")
df_consultas2 = df_consultas2.rename(columns={"Fecha_Hora_Consulta":"Fecha_Consulta","RUT2":"RUT_medico"})
df_consultas = df_consultas.append(df_consultas2)
print(df_consultas.shape)

#10
print(df_consultas.iloc[0:10])
print(df_consultas.shape)
df_consultas = df_consultas.drop(df_consultas.iloc[0:10].index)
print(df_consultas.iloc[0:10])
print(df_consultas.shape)

```

El 8 va entre comillas, para el punto 9 antes de unir hay que realizar los cambios que se le hizo al fichero anterior.

Nuevamente:

```

# 1
import pandas as pd
df_consultas = pd.read_csv("consultas.csv", encoding="utf-8", sep=";")
df_consultas.dtypes

# 2
df_consultas["Fecha_Hora_Consulta"] = df_consultas["Fecha_Hora_Consulta"].astype("datetime64")
df_consultas.dtypes

# 3
print(df_consultas.shape)
print(df_consultas.size)

# 4
df_consultas = df_consultas.set_index("RUT")

```

```

#5a
print(df_consultas.head())

# 5b
print(df_consultas.tail())

# 6a
print(df_consultas.loc["6.079.686-2"])

# 6b
print(df_consultas.loc["4.367.282-7":"19.500.328-3"])

# 6c
print(df_consultas.iloc[23])

# 6d
print(df_consultas.iloc[76:89])

# 7
df_consultas = df_consultas.rename(columns={"Fecha_Hora_Consulta":"Fecha_Consulta", "RUT2":"RUT_medico"})
df_consultas.dtypes

# 8
# Verificamos si existen filas NaN
print(df_consultas.loc[df_consultas.Costo_consulta.isnull()])

df_consultas = df_consultas.fillna("10000")
print(df_consultas.loc[df_consultas.Costo_consulta.isnull()])

# 9
# Cargamos
df_consultas2 = pd.read_csv("consultas2.csv", encoding="utf-8", sep=";")
# Normalizamos
df_consultas2["Fecha_Hora_Consulta"] = df_consultas2["Fecha_Hora_Consulta"].astype("datetime64")
df_consultas2 = df_consultas2.set_index("RUT")
df_consultas2 = df_consultas2.rename(columns={"Fecha_Hora_Consulta":"Fecha_Consulta", "RUT2":"RUT_medico"})
df_consultas2 = df_consultas2.fillna("10000")

# Unimos Los 2 dataframe
df_consultas = df_consultas.append(df_consultas2)

# 10
df_consultas = df_consultas.drop(df_consultas.iloc[0:10].index)

```

Data Frames y Strings

Una forma es recorrer todos estos valores mediante un `for`, y luego ir aplicando diversas funciones de `strings` sobre cada valor. No obstante, esta no es la forma más eficiente de hacerlo.

Existen varias funciones de `strings` que se pueden aplicar directamente sobre la columna.

Estas últimas además requieren de menor tiempo de procesamiento, lo que para grandes volúmenes de datos es muy relevante.

Es posible aplicar diversas funciones de `strings` sobre los valores de cada una de las filas

Iteración sobre Data Frames:

Si tenemos un Data Frame de nombre `df`, podemos iterar sobre sus filas de la siguiente manera:

<p>CÓDIGO</p> <pre>import pandas as pd df = pd.read_csv("ejemplo.csv", encoding="latin-1", sep=";") for index, row in df.iterrows(): print(index) print(row["Nombre"])</pre>	<p>La lista que recorremos en el <code>for</code> se obtiene al ejecutar la función <code>iterrows()</code>.</p>										
<p>RESULTADO</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tr><td style="text-align: center; width: 20px;">0</td><td>Javier Andrés López Castro</td></tr><tr><td style="text-align: center;">1</td><td>Pedro Andrés Vergara Campos</td></tr><tr><td style="text-align: center;">2</td><td>Paula Blanca Marín Campos</td></tr><tr><td style="text-align: center;">3</td><td>Isabel Ignacia González Muñoz</td></tr><tr><td style="text-align: center;">4</td><td>Bernardo Vicente Campos Rodríguez</td></tr></table>	0	Javier Andrés López Castro	1	Pedro Andrés Vergara Campos	2	Paula Blanca Marín Campos	3	Isabel Ignacia González Muñoz	4	Bernardo Vicente Campos Rodríguez	<p>En la declaración del <code>for</code> tenemos las variables <code>index</code> y <code>row</code>. Se deben declarar estas dos variables obligatoriamente. En caso contrario, el <code>for</code> no funcionará correctamente.</p>
0	Javier Andrés López Castro										
1	Pedro Andrés Vergara Campos										
2	Paula Blanca Marín Campos										
3	Isabel Ignacia González Muñoz										
4	Bernardo Vicente Campos Rodríguez										

```
# Cargamos la librería
import pandas as pd
# Creamos el data frame a través de un fichero csv
df = pd.read_csv("ejemplo.csv", encoding="Latin-1", sep=";")

for index, row in df.iterrows():
    print(index)
    print(row["Nombre"])
```

Si queremos extraer el primer nombre de cada fila, lo podríamos hacer de la siguiente manera:

CÓDIGO	<pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") for index, row in df.iterrows(): primer_nombre = row["Nombre"].split(" ")[0] print(primer_nombre)</pre>	Es posible aplicar funciones de strings sobre ellas. En este caso, hicimos un split con el carácter espacio, luego extraemos el primer elemento que corresponde al primer nombre de cada persona.
RESULTADO	<pre>Javier Pedro Paula Isabel Bernardo</pre>	A pesar de que podemos aplicar cualquier función de strings sobre el valor extraído, al recorrer la columna " Nombre ", no es recomendado hacer este tipo de operaciones de esta manera para editar datos en un Data Frame.

```
# Hacemos un split para mostrar solo el nombre
for index, row in df.iterrows():
    primer_nombre = row["Nombre"].split(" ")[0]
    print(primer_nombre)
```

Funciones de strings sobre columnas de un Data Frame:

Si tenemos un Data Frame de nombre **df** y una columna de tipo **object** en Pandas, podemos aplicar las funciones de **strings** que ya conoces.

El comando general se ve de la siguiente manera:

```
df["Nombre columna de tipo object"].str.(función de string)
```



Veamos algunos ejemplos, todos se basan el archivo de ejemplo "**ejemplo.csv**", para que así puedan ir verificando los outputs.

Función len():

Para saber la **cantidad de caracteres** de los elementos de una columna de tipo **object**.

Para un Data Frame de nombre **df**:

CÓDIGO	<pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") print(df["Nombre"].str.len())</pre>	RESULTADO												
		<table><tbody><tr><td>0</td><td>26</td></tr><tr><td>1</td><td>27</td></tr><tr><td>2</td><td>25</td></tr><tr><td>3</td><td>29</td></tr><tr><td>4</td><td>33</td></tr><tr><td>5</td><td>27</td></tr></tbody></table>	0	26	1	27	2	25	3	29	4	33	5	27
0	26													
1	27													
2	25													
3	29													
4	33													
5	27													
Aquí tenemos una columna " Nombre " con el nombre de distintas personas, y queremos saber la cantidad de caracteres de cada nombre.														

```
# Función len()
print(df["Nombre"].str.len())
```

Extraer caracteres de un Data Frame:

Podemos **extraer un carácter de los valores de una columna** de un Data Frame de tipo object, de igual manera a cómo lo haríamos en un **string normal**.

También se puede extraer una secuencia de **caracteres de una columna** de un Data Frame de tipo object, de igual manera a cómo lo haríamos en un **string normal**.

Es posible extraer un carácter, para un Data Frame de nombre **df**:

```
df["nombre columna"].str[posición carácter]
```

CÓDIGO	RESULTADO										
<pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") print(df["RUT"].str[0])</pre>	<table><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>8</td></tr><tr><td>4</td><td>1</td></tr></tbody></table>	0	1	1	1	2	1	3	8	4	1
0	1										
1	1										
2	1										
3	8										
4	1										

Aquí se puede ver, cómo extraer el primer carácter de la columna "RUT" (con el RUT de distintas personas) del Data Frame **df**. Se podría extraer cualquier carácter ingresando una posición determinada.

```
# Extraer carácter
print(df["RUT"].str[0])
```

Es posible extraer un conjunto de caracteres, para un Data Frame de nombre **df**:

```
df["nombre columna"].str[posición carácter inicial:posición carácter final]
```

CÓDIGO	RESULTADO										
<pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") print(df["RUT"].str[0:4])</pre>	<table><tbody><tr><td>0</td><td>13.6</td></tr><tr><td>1</td><td>17.1</td></tr><tr><td>2</td><td>15.3</td></tr><tr><td>3</td><td>8.29</td></tr><tr><td>4</td><td>15.7</td></tr></tbody></table>	0	13.6	1	17.1	2	15.3	3	8.29	4	15.7
0	13.6										
1	17.1										
2	15.3										
3	8.29										
4	15.7										

Sabemos que df tiene una columna "RUT" con el RUT de distintas personas, y queremos extraer los cuatro primeros dígitos.

Al igual que en los **strings**, el **slice** toma desde la posición inicial a la posición final-1.

```
# Extraer caracteres
print(df["RUT"].str[0:4])
```

Función lower():

Sirve para transformar los caracteres a **minúsculas** en los valores de una columna de un Data Frame de tipo **object**.

De forma general, para un Data Frame de nombre **df**:

```
df["nombre columna"].str.lower()
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
print(df["Nombre"].str.lower())
```

Tenemos un Data Frame de nombre **df**, que tiene una columna "**Nombre**" con el nombre de distintas personas, y queremos que el nombre quede solo en minúsculas.

RESULTADO

0	javier andrés lópez castro
1	pedro andrés vergara campos
2	paula blanca marin campos
3	isabel ignacia gonzález muñoz
4	bernardo vicente campos rodriguez

Si se desea "guardar" este cambio debe volver a asignarlo a la columna original. Esto es igual para las funciones que siguen.

```
# Función lower()
print(df["Nombre"].str.lower())
```

Función upper():

Sirve para transformar los caracteres a **mayúsculas** de los valores de una columna de un Data Frame de tipo **object**.

De forma general, para un Data Frame de nombre **df**:

```
df["nombre columna"].str.upper()
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
print(df["Nombre"].str.upper())
```

Tenemos un Data Frame de nombre **df**, que tiene una columna "**Nombre**" con el nombre de distintas personas, y queremos que el nombre quede solo en mayúsculas.

RESULTADO

0	JAVIER ANDRÉS LÓPEZ CASTRO
1	PEDRO ANDRÉS VERGARA CAMPOS
2	PAULA BLANCA MARÍN CAMPOS
3	ISABEL IGNACIA GONZÁLEZ MUÑOZ
4	BERNARDO VICENTE CAMPOS RODRÍGUEZ

Si uno desea "guardar" este cambio debe volver a asignarlo a la columna original.

```
# Función upper()
print(df["Nombre"].str.upper())
```

Función replace(x, y, regex=True|False): True (exp. Regular), False (cadena)

Se usa para **reemplazar** los **strings x** por **y** en los **valores** de una columna de un Data Frame de tipo **object** mediante la función **replace(x,y)**.

Para un Data Frame de nombre `df`, donde se desea reemplazar los `strings x` por los `y`.

```
df["nombre columna"].str.replace(x,y)
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
print(df["Fecha_Nac"].str.replace("/", "-"))
```

RESULTADO

0	1963-2-17
1	1993-10-24
2	1962-5-30
3	1972-3-20
4	1987-7-11

El Data Frame de nombre `df`, tiene una columna “`Fecha_Nac`” que corresponde a la fecha de nacimiento de distintas personas, en este caso se cambia el separador “`/`” por “`-`”.

```
# Función replace(x,y)
print(df["Fecha_Nac"].str.replace("/", "-"))
```

Función `containers(x)`:

Permite saber si los **valores** de una columna de un Data Frame de tipo `object` tienen un `string` específico.

De forma general para un Data Frame de nombre `df`, donde se desea buscar un `string x`.

```
df["nombre columna"].str.contains(x)
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
print(df["RUT"].str.contains("-"))
```

RESULTADO

0	True
1	True
2	True
3	True
4	True

En este caso `df` tiene una columna “`RUT`” con el RUT de distintas personas, y queremos saber si cada uno de los RUT tiene un guion (“`-`”).

```
# Función contains(x)
print(df["RUT"].str.contains("-"))
```

Función `find(x)`:

Podemos saber la **posición** de la primera aparición de un `string x` en una columna de un Data Frame.

De forma general para un Data Frame de nombre `df`:

```
df["nombre columna"].str.find(x)
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
print(df["Nombre"].str.find(" "))
```

RESULTADO

0	6
1	5
2	5
3	6
4	8

Aquí `df` tiene una columna “`Nombre`” con el nombre completo de distintas personas, y queremos saber en qué posición está el `string " "` (espacio).

```
# Función find(x)
print(df[ "Nombre"].str.find(" "))
```

Función Split(x):

Se utiliza para **separar**, según un **string x**, los **valores** de una columna de un Data Frame de tipo **object**.

Para un Data Frame de nombre **df**, donde se desea separar mediante el **string x**.

df["nombre columna"].str.split(x)	
CÓDIGO	RESULTADO
<pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") print(df["Nombre"].str.split(" "))</pre>	<pre>0 [Javier, Andrés, López, Castro] 1 [Pedro, Andrés, Vergara, Campos] 2 [Paula, Blanca, Marín, Campos] 3 [Isabel, Ignacia, González, Muñoz] 4 [Bernardo, Vicente, Campos, Rodríguez]</pre>

En este caso df tiene una columna **"Nombre"** con el nombre de distintas personas, y queremos obtener una lista con los nombres y apellidos.

```
# Función split(x)
print(df[ "Nombre"].str.split(" "))
```

Estructura y operaciones con Data Frames:

Función Split(x, expand=True):

FUNCIÓN
split(x, expand=True)

Podemos crear un Data Frame a partir de un **split** sobre una columna de tipo **object**, añadiendo como parámetro **"expand"** y dándole como valor **True**.

Pensemos en el caso de un Data Frame que tiene una columna **"Nombre"** con el nombre completo de distintas personas. ¿Cómo podríamos separar este nombre completo en primer nombre, segundo nombre, primer apellido y segundo apellido?

Para separar la columna nombre:

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df["Nombre"].str.split(" ",expand=True)
print(df2)
```

RESULTADO

	0	1	2	3
0	Javier	Andrés	López	Castro
1	Pedro	Andrés	Vergara	Campos
2	Paula	Blanca	Marín	Campos
3	Isabel	Ignacia	González	Muñoz
4	Bernardo	Vicente	Campos	Rodríguez
5	Isidora	Javiera	Muñoz	López

En este caso, `df2` es un Data Frame de 4 columnas distintas (primer nombre, segundo nombre, primer apellido y segundo apellido). Este se generó al agregar el parámetro `expand=True` al hacer el `split`.

```
# Función split(x, expand=True)
# Crea un dataframe de una columna
df2 = df[ "Nombre" ].str.split( " ", expand=True)
print(df2)
```

Cambiar nombre a las columnas:

El Data Frame creado anteriormente no tiene nombre en sus columnas. De forma general, para un Data Frame de nombre `df`.

```
df.columns = [nombres de las columnas]
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")

df2 = df["Nombre"].str.split(" ",expand=True)
df2.columns=[ "Primer Nombre", "Segundo Nombre", "Primer Apellido", "Segundo Apellido"]
print(df2)
```

RESULTADO

	Primer Nombre	Segundo Nombre	Primer Apellido	Segundo Apellido
0	Javier	Andrés	López	Castro
1	Pedro	Andrés	Vergara	Campos
2	Paula	Blanca	Marín	Campos
3	Isabel	Ignacia	González	Muñoz
4	Bernardo	Vicente	Campos	Rodríguez

Usaremos el atributo "columns" del Data Frame. Para acceder a un atributo, simplemente tomamos la variable que almacena el Data Frame, y con un punto accedemos a él.

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")

df2 = df["Nombre"].str.split(" ",expand=True)
df2.columns=[ "Primer Nombre", "Segundo Nombre", "Primer Apellido", "Segundo Apellido"]
print(df2)
```

RESULTADO

	Primer Nombre	Segundo Nombre	Primer Apellido	Segundo Apellido
0	Javier	Andrés	López	Castro
1	Pedro	Andrés	Vergara	Campos
2	Paula	Blanca	Marín	Campos
3	Isabel	Ignacia	González	Muñoz
4	Bernardo	Vicente	Campos	Rodríguez

Luego, asignamos la lista con los nombres de las columnas. Los atributos se definen como características del Data Frame que nos permitirán acceder a mayores funcionalidades con ellos.

```
df3 = df[ "Nombre" ].str.split( " ", expand=True)
df3.columns = [ "Primer Nombre", "Segundo Nombre", "Primer Apellido", "Segundo Apellido"]
print(df3)
```

Función join():

Esta función **agrega** las columnas de un Data Frame a otro.

De forma general, para un Data Frame `df` y otro `df2`.

```
df.join(df2)
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")

df2 = df["Nombre"].str.split(" ",expand=True)
df2.columns=["Primer Nombre","Segundo Nombre","Primer Apellido","Segundo Apellido"]
df = df.join(df2)
print(df.dtypes)
```

RESULTADO

RUT	object
Nombre	object
Sexo	object
Fecha_Nac	object
Monto	int64
Primer Nombre	object
Segundo Nombre	object
Primer Apellido	object
Segundo Apellido	object
dtype:	object

¿Qué pasaría si el Data Frame con el que intentamos hacer `join` (en este ejemplo `df2`) tuviera una o más columnas del mismo nombre que el Data Frame original? **En este caso Python arroja un error.**

```
# Función join()
df = df.join(df3)
print(df.dtypes)
```

Para evitar el error anterior, se ocupan los **parámetros `lsuffix` y `rsuffix`** que agregan un sufijo a las columnas que tienen el mismo nombre.

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")

df2 = df["Nombre"].str.split(" ",expand=True)
df2.columns=["Nombre","Segundo Nombre","Primer Apellido","Segundo Apellido"]
df = df.join(df2, rsuffix = "_df2", lsuffix=_df")
print(df.dtypes)
```

RESULTADO

RUT	object
Nombre_df	object
Sexo	object
Fecha_Nac	object
Monto	int64
Nombre_df2	object
Segundo Nombre	object
Primer Apellido	object
Segundo Apellido	object
dtype:	object

En el ejemplo, podemos ver qué pasa al ocupar estos parámetros y cómo evitamos el error que apareció anteriormente.

```
# Función join(), pero evitando errores de columnas repetidas
df = df.join(df3, rsuffix="_df3", lsuffix=_df3")
print(df.dtypes)
```

¿Qué pasa si queremos hacer `join` sobre dos Data Frames con distinto número de filas?

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")

df2 = pd.DataFrame([[1,2],[3,4]])
df = df.join(df2)
print(df)
```

Se observa que al unir ambos Data Frames, la mayoría de las filas quedó con datos "NaN", dado que tenían distintos números de filas.

RESULTADO

RUT	Nombre	Sexo	Fecha_Nac	Monto	0	1
13.626.365-6	Javier Andrés López Castro	MASCULINO	1963/2/17	9889868	1	2.0
17.135.958-1	Pedro Andrés Vergara Campos	MASCULINO	1993/10/24	4071184	3	4.0
15.391.058-0	Paula Blanca Marín Campos	FEMENINO	1962/5/30	9004634	NaN	NaN
8.296.689-7	Isabel Ignacia González Muñoz	FEMENINO	1972/3/20	1266523	NaN	NaN
15.755.894-8	Bernardo Vicente Campos Rodríguez	MASCULINO	1987/7/11	6515702	NaN	NaN

```
df4 = pd.DataFrame([[1, 2], [3, 4]])
df = df.join(df4)
print(df)
```

Unión de dos Data Frames:

1

Hacer un `join` solo para unir las columnas de dos Data Frames distintos, es una visión muy acotada de esta funcionalidad. Pero **¿si quisieramos unir dos Data Frame en base a valores en común entre ambos?**

Función merge():

La función `merge()` nos permite **unir** la información de dos Data Frames distintos basados en los valores de una columna, cuando haya un valor en común.

Digamos que tenemos la información de los clientes de dos bancos distintos. Pueden notar que las personas con RUT "13.626.365-6" y "17.135.958-1" tienen cuenta en ambos bancos.

Banco A		Banco B	
RUT	Monto	RUT	Monto
13.626.365-6	9889868	13.626.365-6	7565403
17.135.958-1	4071184	17.135.958-1	1237464
15.391.058-0	9004634	15.755.894-8	6515702
8.296.689-7	1260523	17.399.932-8	5507746

¿Cómo podríamos crear un Data Frame que solo tuviese la información de las personas que tienen cuenta en ambos bancos?

Según lo planteado anteriormente, nos gustaría crear el siguiente Data Frame:

RUT	Monto Banco A	Monto Banco B
13.626.365-6	9889868	7565403
17.135.958-1	4071184	1237464

Para unir dos Data Frames `df1` y `df2` distintos en base a una columna que comparta valores en común:

```
df1.merge(df2, on=columna_en_común)
```

Para el ejemplo del banco presentado anteriormente:

CÓDIGO	RESULTADO									
<pre>import pandas as pd df_banco_A = pd.read_csv("bancoA.csv", encoding="latin-1", sep=";") df_banco_B = pd.read_csv("bancoB.csv", encoding="latin-1", sep=";") df_banco_A = df_banco_A.merge(df_banco_B, on="RUT", suffixes=("_bancoA", "_bancoB")) print(df_banco_A)</pre>	<table border="1"><thead><tr><th>RUT</th><th>Monto_bancoA</th><th>Monto_bancoB</th></tr></thead><tbody><tr><td>13.626.365-6</td><td>9889868</td><td>7565403</td></tr><tr><td>17.135.958-1</td><td>4071184</td><td>1237464</td></tr></tbody></table>	RUT	Monto_bancoA	Monto_bancoB	13.626.365-6	9889868	7565403	17.135.958-1	4071184	1237464
RUT	Monto_bancoA	Monto_bancoB								
13.626.365-6	9889868	7565403								
17.135.958-1	4071184	1237464								

¿Qué pasaría si queremos hacer la misma operación, pero los dos Data Frames tienen columnas de RUT con distinto nombre? Podemos hacerlo sin necesidad de renombrar las columnas.

Digamos que los Data Frames de los bancos A y B respectivamente son los siguientes:

Banco A		Banco B	
ID	Monto	RUT	Monto
13.626.365-6	9889868	13.626.365-6	7565403
17.135.958-1	4071184	17.135.958-1	1237464
15.391.058-0	9004634	15.755.894-8	6515702
8.296.689-7	1260523	17.399.932-8	5507746

Podemos manejar este tipo de casos utilizando la función `merge` de la siguiente manera:

CÓDIGO

```
import pandas as pd
df_banco_A = pd.read_csv("bancoA.csv",encoding="latin-1",sep=";")
df_banco_B = pd.read_csv("bancoB.csv",encoding="latin-1",sep=";")

df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"))

print(df_banco_A)
```

RESULTADO

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868	13.626.365-6	7565403
1	17.135.958-1	4071184	17.135.958-1	1237464

Basta con que agreguemos los parámetros `left_on` y `right_on` para identificar en qué columnas vamos a buscar valores en común.

```
# Función merge()
df_banco_A = pd.read_csv("bancoA.csv", encoding="latin-1", sep=";")
df_banco_B = pd.read_csv("bancoB.csv", encoding="latin-1", sep=";")
df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"))
print(df_banco_A)
```

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868	13.626.365-6	7565403
1	17.135.958-1	4071184	17.135.958-1	1237464



¿Cuáles otros modos hay?
Vamos a ver dos más: `left` y `outer`



En realidad el “`merge`” que estamos haciendo tiene un modo, que se llama “`inner`”.



Para aquellos que están familiarizados con SQL, es el equivalente a un `Inner Join` entre dos tablas.



Aquí, el Data Frame resultante estará compuesto solo por aquellos valores que se encuentren en ambos.

Función `merge()` modo “left”:

CÓDIGO

```
import pandas as pd
df_banco_A = pd.read_csv("bancoA.csv",encoding="latin-1",sep=";")
df_banco_B = pd.read_csv("bancoB_2.csv",encoding="latin-1",sep=";")

df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID",right_on="RUT",suffixes=("_bancoA", "_bancoB"), how="left")

print(df_banco_A)
```

RESULTADO

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868	13.626.365-6	7565403.0
1	17.135.958-1	4071184	17.135.958-1	1237464.0
2	15.391.058-0	9004634		NaN
3	8.296.689-7	1260523		NaN

Importante

En el `merge` modo “`left`”, se mantienen todas las filas del Data Frame donde estamos haciendo el `merge` y solo se agregan las columnas del otro Data Frame que tengan valores en común según la columna que estemos analizando.

```
# Función merge() modo "left"
df_banco_B = pd.read_csv("bancoB_2.csv", encoding="latin-1", sep=";")
df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"), how="left")
print(df_banco_A)
```

	ID	Monto_bancoA	RUT_bancoA	Monto_bancoB	RUT_bancoB
0	13.626.365-6	9889868	13.626.365-6	7565403	13.626.365-6
1	17.135.958-1	4071184	17.135.958-1	1237464	17.135.958-1
			Monto		
0		7565403			
1		1237464			

CÓDIGO

```
import pandas as pd
df_banco_A = pd.read_csv("bancoA.csv", encoding="latin-1", sep=";")
df_banco_B = pd.read_csv("bancoB_2.csv", encoding="latin-1", sep=";")

df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"), how="left")
print(df_banco_A)
```

RESULTADO

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868	13.626.365-6	7565403.0
1	17.135.958-1	4071184	17.135.958-1	1237464.0
2	15.391.058-0	9004634	NaN	NaN
3	8.296.689-7	1260523	NaN	NaN

En el Data Frame resultante, todas las personas del Banco A quedaron. Además, Pandas buscó cuáles de las personas del Banco A estaban en el B y también rescató esos valores.

Función merge() modo “outer”:

Si tenemos los mismos Data Frames anteriores representando cuentas de dos bancos, veamos qué ocurre al aplicar esta función.

CÓDIGO

```
import pandas as pd
df_banco_A = pd.read_csv("bancoA.csv", encoding="latin-1", sep=";")
df_banco_B = pd.read_csv("bancoB_2.csv", encoding="latin-1", sep=";")

df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"), how="outer")
print(df_banco_A)
```

RESULTADO

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868.0	13.626.365-6	7565403.0
1	17.135.958-1	4071184.0	17.135.958-1	1237464.0
2	15.391.058-0	9004634.0	NaN	NaN
3	8.296.689-7	1260523.0	NaN	NaN
4	NaN	NaN	15.755.894-8	6515702.0

Importante

En el **merge** modo **“outer”**, se mantienen todas las filas del Data Frame donde estamos haciendo el **merge** y se agregan todas las filas y columnas del otro Data Frame que estemos agregando. En aquellas donde hay un valor en común en la columnas que se está analizando, se juntan las columnas. En aquellas que no, se llena con valores nulo para el Data Frame donde se está haciendo el **merge** e igualmente para el Data Frame que se está agregando.

```
# Función merge() modo "outer"
df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"), how="outer")
print(df_banco_A)
```

	ID	Monto_bancoA	RUT_bancoA	Monto_bancoB	RUT_bancoB
0	13.626.365-6	9889868.0	13.626.365-6	7565403.0	13.626.365-6
1	17.135.958-1	4071184.0	17.135.958-1	1237464.0	17.135.958-1
2	NaN	NaN	NaN	NaN	NaN
	Monto_bancoA	RUT	Monto_bancoB		
0	7565403.0	13.626.365-6	7565403		
1	1237464.0	17.135.958-1	1237464		
2	NaN	15.755.894-8	6515702		

Si tenemos los mismos Data Frames anteriores representando cuentas de dos bancos, veamos qué ocurre al aplicar esta función.

CÓDIGO

```
import pandas as pd
df_banco_A = pd.read_csv("bancoA.csv", encoding="latin-1", sep=";")
df_banco_B = pd.read_csv("bancoB_2.csv", encoding="latin-1", sep=";")

df_banco_A = df_banco_A.merge(df_banco_B, left_on="ID", right_on="RUT", suffixes=("_bancoA", "_bancoB"), how="outer")
print(df_banco_A)
```

RESULTADO

	ID	Monto_bancoA	RUT	Monto_bancoB
0	13.626.365-6	9889868.0	13.626.365-6	7565403.0
1	17.135.958-1	4071184.0	17.135.958-1	1237464.0
2	15.391.058-0	9004634.0	NaN	NaN
3	8.296.689-7	1260523.0	NaN	NaN
4	NaN	NaN	15.755.894-8	6515702.0

En el Data Frame resultante, todas las personas del Banco A y del Banco B quedaron. Es decir, se buscan aquellas filas que tengan valores en común en la columna que se está analizando y se juntan (como podemos ver en las dos primeras filas).

Importante

Al hacer `merge`, las columnas donde se buscarán valores en común deben ser del mismo tipo.

Esto se puede revisar con el comando `dtypes`.

Podemos transformar el tipo de una columna con la función `astype(tipo_de_dato)`. Específicamente, el parámetro que se ingresa es el tipo de dato al que se quiere convertir la columna.

Caso práctico:

- 1 En `df_pacientes`, transforma en dólares (el dólar está a \$700) la deuda de cada persona y guárdalo en una lista. Debes desarrollar el procedimiento mediante el comando `iterrows` (y un `for`). Luego, muestra los 10 primeros valores de esta lista.
- 2 La clínica está evaluando, si es posible almacenar el nombre de cada persona con solo los 10 primeros caracteres para abaratar costos, por lo tanto, debes mostrar cómo se vería en consola.
- 3 Debes cambiar el nombre de cada persona a mayúsculas.
- 4 Imprime en consola los nombres que tienen una letra ñ (ya que en general se visualizan mal en los sistemas de información de la clínica).
- 5 Hay más información de pacientes en el archivo original “`datos_pacientes2.csv`”. Carga este archivo en un Data Frame de nombre `df_pacientes2`.
- 6 Varios datos en las columnas RUT y Nombre vienen con información corrupta caracterizada por un “XXXX”. Debes eliminar estas filas de `df_pacientes`.
- 7 En este archivo venían nombres y apellidos separados con guiones en vez de espacio. Deja que todos los nombres y apellido estén separados solo con espacios, y guarda este cambio en `df_pacientes2`.
- 8 Limpia los “ ” que están en la información de la fecha de nacimiento de algunos pacientes, específicamente después del año.
- 9 Ahora que la información está limpia, debes unir `df_pacientes2` con `df_pacientes`.
- 10 Setea el `index` de `df_pacientes` por la columna RUT.
- 11 Los nombres tienen el formato “`primer_nombre segundo_nombre primer_apellido segundo_apellido`”. Debes separar los datos de la columna nombre en cuatro. Luego, agrega estas columnas al Data Frame original `df_pacientes`.
- 12 El gerente consolidó la ciudad de residencia y país de origen de cada paciente. Esta información se encuentra en el archivo “`nacionalidad.csv`”. Debes cargar esta información en un Data Frame y luego unirla a `df_pacientes`. La recomendación, es unir en base a la información RUT, aunque es posible que no esté completa, y no se tenga la información acerca de la nacionalidad de ciertos pacientes.

```
1 import pandas as pd
2
3 #1
4 df_pacientes = pd.read_csv(
5     "C:\\\\Users\\\\mlevi\\\\OneDrive - Bogado Ingenieros Consultores SpA\\\\MIRESPALDO\\\\Cursos\\\\Python\\\\Modulo 3\\\\datos_pacientes.csv",
6     encoding="utf-8",
7     sep=";")
8
9 deuda_dolar = []
10
11 for index, row in df_pacientes.iterrows():
12     dolar = row["Monto_Deuda"] * 700
13     deuda_dolar.append(dolar)
14
15 print(deuda_dolar[0:10])
16
17 #2
18 print(df_pacientes["Nombre"].str[0:10])
19
20 #3
21 #print(df_pacientes["Nombre"].str[0:10].str.upper())
22 #o
23 df_pacientes["Nombre"] = df_pacientes["Nombre"].str.upper()
24
25 #4
26 print(df_pacientes.loc[df_pacientes["Nombre"].str.contains("R")])
```

```
28 #5
29 df_pacientes2 = pd.read_csv(
30     "C:\\\\Users\\\\mlevi\\\\OneDrive - Bogado Ingenieros Consultores SpA\\\\MIRESPALDO\\\\Cursos\\\\Python\\\\Modulo 3\\\\datos_pacientes2.csv",
31     encoding="utf-8",
32     sep=";")
33
34 #6
35 df_pacientes2 = df_pacientes2.loc[~(df_pacientes2["RUT"].str.contains("XXXX")) & ~(df_pacientes2["Nombre"].str.contains("XXXX"))]
36 #o
37 #df_pacientes2 = df_pacientes2.drop(
38 #    df_pacientes2.loc[df_pacientes2["RUT"].str.contains("XXXX") | df_pacientes2["Nombre"].str.contains("XXXX")].index)
39
40 #7
41 df_pacientes2["Nombre"] = df_pacientes2["Nombre"].str.replace("-", " ")
42
43 #8
44 df_pacientes2["Fecha_Nacimiento"] = df_pacientes2["Fecha_Nacimiento"].str.replace("_/", "/")
45
46 #9
47 # Append porque tiene columnas iguales
48 df_pacientes2["Nombre"] = df_pacientes2["Nombre"].str.upper()
49 df_pacientes = df_pacientes.append(df_pacientes2)
```

```
#10
df_pacientes = df_pacientes.set_index("RUT")

#11
df = df_pacientes["Nombre"].str.split(" ", expand=True)
df.columns=["primer_nombre", "segundo_nombre", "primero_apellido", "segundo_apellido"]
df_pacientes = df_pacientes.join(df)

#12
df_nacionalidad = pd.read_csv([
    "C:\\\\Users\\\\mlevi\\\\OneDrive - Bogado Ingenieros Consultores SpA\\\\MIRESPALDO\\\\Cursos\\\\Python\\\\Modulo 3\\\\nacionalidad.csv",
    encoding="utf-8",
    sep=";"])
df_pacientes = df_pacientes.merge(df_nacionalidad, left_index=True, right_on="RUT", how="left")
df_pacientes = df_pacientes.set_index("RUT")
print(df_pacientes.head())
```

Procedimiento y Extracción de Información:

Parte 1. Herramientas avanzadas para la selección de datos:

```
import pandas as pd

df = pd.read_csv(
    "C:\\Users\\Usuario\\OneDrive - Bogado Ingenieros Consultores Sp
    encoding="latin-1", sep=";")

print(df.loc[df["Monto"] > 5000000])

print(df.loc[(df["Monto"] >= 5000000) & (df["Monto"] <= 7000000)])

print[(df.loc[(df["Monto"] < 1000000) | (df["Monto"] > 9000000)]]

print(df.loc[df["Nombre"].str.contains("González")])
```

```
# Buscar aquellas personas cuyo primer nombre es Daniela
print(df.loc[df["Nombre"].str.split(" ", expand=True)[0] == "Daniela"])
```

```
# Personas de 1er nombre Daniela, digito verificador 8 y un monto menor a
# $1000000
print(
    df.loc[(df["Nombre"].str.split(" ", expand=True)[0] == "Daniela") &
           (df["Monto"] < 1000000) & (df["RUT"].str[-1:] == "8")]
)
```

Ordenar valores en un Dataframe:

Función sort_values:

<pre>df = df.sort_values(by=(columna o lista de columnas), ascending = (True o False))</pre>	<p>CÓDIGO</p> <pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") df = df.sort_values(by=["Sexo","Monto"],ascending=False) print(df)</pre>	<p>Ordenamos en forma descendente (lo que se observa al darle valor False al parámetro "ascending").</p> <p>Además, ordenamos en base a dos columnas. Primero la columna Sexo y luego la columna Monto.</p>
--	---	---

<pre>df = df.sort_values(by=["Sexo", "Monto"], ascending=False) print(df)</pre>	<p>CÓDIGO</p> <pre>import pandas as pd df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";") df = df.sort_values(by=["Sexo", "Monto"], ascending=False) print(df)</pre>	<p>Cuando se ordena por más de una columna y hay valores iguales, se ocupa el segundo criterio. En el ejemplo, pueden observar que se ordenó la columna Sexo. Primero Masculino y después Femenino. Como en cada uno de ellos habían muchos valores iguales, se ocupa la segunda columna como criterio para ordenarlos.</p>
--	--	---

```
# Ordenar de forma descendente
df = df.sort_values(by=[ "Sexo", "Monto"], ascending=False)
print(df)
```

Extracción de datos:



Existen herramientas sumamente útiles para extraer datos de un Data Frame.



Veremos cómo reordenar un Data Frame, que es una maneras de extraer datos y llegar a lo más importante, es decir, transformarlos en información.

Función pivot:

Esta función nos permite reordenar los datos eligiendo qué valores queremos como filas y cuáles como columnas en una nueva tabla. En consecuencia, es posible darle una nueva interpretación a los datos dependiendo de cómo ocupamos esta función.

La siguiente base de datos muestra filas para cada cuenta que tenga un cliente en bancos:

RUT	NOMBRE	SEXO	FECHA_NAC	BANCO	MONTO
12.172.126-4	Ramón Javier Rodríguez Rodríguez	MASCULINO	22-03-60	BancoB	\$ 7,574,279
13.295.513-9	Rodrigo Francisco Robles González	MASCULINO	04-05-60	BancoA	\$ 1,494,955
11.975.455-5	Daniela Pamela Robles Robles	FEMENINO	30-01-61	BancoB	\$ 4,923,235
11.975.455-5	Daniela Pamela Robles Robles	FEMENINO	30-01-61	BancoA	\$ 9,922,167
17.462.609-3	Victoria Blanca Rodríguez Saavedra	FEMENINO	10-03-61	BancoB	\$ 7,942,728
18.950.490-3	Andrés Bernardo Vergara Valenzuela	MASCULINO	30-09-61	BancoA	\$ 5,462,427
18.950.490-3	Andrés Bernardo Vergara Valenzuela	MASCULINO	30-09-61	BancoB	\$ 5,806,832

Hay personas que aparecen dos veces en la fila. Esto es porque tienen cuenta en el Banco A y B. Esta tabla la llamaremos “ejemplo2.csv”.

Es posible crear un Data Frame consolidando los montos por banco para cada persona.

```
df.pivot(index=(columna que determinará las filas), columns=(columna que determinara las columnas), values=(columna a analizar))
```

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df.pivot(index="RUT",columns="BANCO",values="MONTO")
print(df2)
```

Lo importante para armar este nuevo Data Frame es saber identificar los tres elementos que la componen: **index**, **columns** y **values**.

RESULTADO

BANCO	BancoA	BancoB
RUT		
10.082.841-7	3217334.0	NaN
10.210.121-3	NaN	3031298.0
10.385.049-3	NaN	5319427.0
10.499.707-5	6277723.0	NaN

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df.pivot(index="RUT",columns="BANCO",values="MONTO")
print(df2)
```

RESULTADO

BANCO	BancoA	BancoB
RUT		
10.082.841-7	3217334.0	NaN
10.210.121-3	NaN	3031298.0
10.385.049-3	NaN	5319427.0
10.499.707-5	6277723.0	NaN

El parámetro `index` corresponde a las filas del Data Frame a formar. En este caso, como queremos saber la información de los bancos, las columnas serán el banco, y el valor para analizar es el monto.

```
df = pd.read_csv(
    "C:\\\\Users\\\\Usuario\\\\OneDrive - Bogado Ingenieros Consultores SpA\\\\MIRESPALDO\\\\Curs
encoding="latin-1", sep=";")

df2 = df.pivot(index="RUT", columns="BANCO", values="MONTO")
print(df2)
```

Función pivot_table:

Es un caso particular de la función pivot, y nos permite aplicar funciones matemáticas sobre la columna que especifiquemos en el parámetro `values`.

Podemos calcular el monto promedio de todas las cuentas por banco:

```
df.pivot_table(index=(columna que determinará las filas),columns=(columna que determinará las columnas),values=(columna a analizar),aggfunc=(funciones de la librería numpy))
```

CÓDIGO

```
import pandas as pd
import numpy as np

df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df.pivot_table(index="BANCO",values="MONTO", columns="SEXO", aggfunc=np.mean)
print(df2)
```

RESULTADO

SEXO	FEMENINO	MASCULINO
BANCO		
BancoA	4.282360e+06	5.615042e+06
BancoB	5.167930e+06	5.278892e+06

Lo primero es importar el paquete numpy, incorporando al principio del código `import numpy as np`.

En la función `pivot_table` nuestro `index` (o filas) será la columna BANCO, para analizar el Banco A y el Banco B.

CÓDIGO

```
import pandas as pd
import numpy as np

df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df.pivot_table(index="BANCO",values="MONTO", columns="SEXO", aggfunc=np.mean)
print(df2)
```

RESULTADO

SEXO	FEMENINO	MASCULINO
BANCO		
BancoA	4.282360e+06	5.615042e+06
BancoB	5.167930e+06	5.278892e+06

Luego, los valores para analizar serán aquellos que estén en la columna MONTO.

Finalmente, definimos la función que queremos ocupar para el análisis. En este caso es `np.mean`, que es la función "media" del paquete numpy.

```
df2 = df.pivot_table(index="BANCO", values="MONTO", columns="SEXO", aggfunc=np.mean)
print(df2)
```

La idea es potenciar las herramienta, en consecuencia no solo sacaremos la media sino también el mínimo y el máximo:

CÓDIGO

```
import pandas as pd
import numpy as np

df = pd.read_csv("ejemplo.csv",encoding="latin-1",sep=";")
df2 = df.pivot_table(index="BANCO",values="MONTO", columns="SEXO", aggfunc={np.mean,np.min,np.max})
print(df2)
```

RESULTADO

	amax	amin	mean			
SEXO	FEMENINO	MASCULINO	FEMENINO	MASCULINO		
BANCO	9979530.0	9911177.0	626641.0	197539.0	4.28236e+06	5.615942e+06
BancoA	9691754.0	9351786.0	154149.0	441915.0	5.167930e+06	5.278892e+06

A diferencia del código anterior, agregamos en el parámetro `columns` a la columna SEXO.

Luego, en el parámetro `aggfunc` agregamos los estadísticos descriptivos que queremos, en este caso, `mean, std, min, max` (dentro de `{}`).

```
df2 = df.pivot_table(index="BANCO", values="MONTO", columns="SEXO", aggfunc={np.mean, np.min, np.max})
print(df2)
```

Conclusiones:

- En esta clase aprendimos tres herramientas muy útiles para extraer información de un set de datos.
- Filtros avanzados, que permiten relacionar distintos datos y así extraer información útil, a partir de un archivo CSV o cualquier set de datos.
- Ordenar valores útiles para tener una panorámica general de los datos y saber casos de acuerdo a cierto criterio. Por ejemplo, fechas de inicio o término.
- Finalmente, estudiamos `pivot_table` que permite agrupar los datos en base a una operación matemática. Esta es la herramienta más poderosa para extraer información a partir de un set de datos, ya que permite realizar operaciones que pueden ser comunes en el mundo laboral pero de una forma eficiente y efectiva.

¿Cómo unir la información?

m4-ej13.py

Nos gustaría unir la información de "lista_productos.csv" con la lista de ventas, para tener más datos de cada producto (según su número).

CÓDIGO

```
df2 = pd.read_csv("lista_productos.csv",encoding="latin-1")
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)

df2["NUM"] = df2["NUM"].astype(str)

print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)

df = df.merge(df2,on="NUM")

print(df.head())
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
0	2018/9/22	23346	18	ABARROTES	Aceite	890
1	2018/10/7	23346	35	ABARROTES	Aceite	890
2	2018/7/1	23346	13	ABARROTES	Aceite	890
3	2018/8/3	23346	10	ABARROTES	Aceite	890
4	2018/7/13	23346	37	ABARROTES	Aceite	890

¿Cómo podríamos extraer información del Data Frame?

Don Juan intuye que hay productos que se venden muy poco. Por lo tanto, le gustaría buscar aquellos que tengan un valor inferior a \$1.000 y que se vendan menos de 5 veces por venta.

CÓDIGO

```
print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5)])
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
21	2018/9/26	23346	4	ABARROTES	Aceite	850
123	2018/1/20	71935	3	PANADERIA	Marrakaeta (kg)	890
126	2018/7/10	71935	1	PANADERIA	Marrakaeta (kg)	890
264	2018/12/7	72963	1	PANADERIA	Hallulla (kg)	890
272	2018/3/2	37847	1	LIMPIEZA	Esponja	390
275	2018/12/23	37847	1	LIMPIEZA	Esponja	390
374	2018/10/19	36736	3	LIMPIEZA	Limiapisos Z	990
377	2018/10/7	36736	3	LIMPIEZA	Limiapisos Z	990

¿Cómo lo podríamos saber?

En este caso, hacemos dos filtros: que el precio sea menor a 1000 y que la cantidad sea menor que 5. Ambos (y por eso se ocupa el “`&`” de and) se deben cumplir.

A partir del filtro anterior, a Don Juan le gustaría saber qué productos se vendieron menos de 5 veces, que además tengan un valor menor a \$1000 y se hayan vendido en los meses de abril y mayo.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/",expand=True)
df_aux.columns = ["ANHO","MES","DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo podríamos hacer este filtro?

Primer: separamos la columna `FECHA` en tres columnas con un `split`, y el Data Frame resultante lo agregamos a una variable.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/",expand=True)
df_aux.columns = ["ANHO","MES","DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo podríamos hacer este filtro?

Segundo: renombramos las columnas en este Data Frame como `ANHO` (se ocupa NH en vez de ñ para evitar problemas con los nombres), `MES` y `DIA`.

A partir del filtro anterior, a Don Juan le gustaría saber qué productos se vendieron menos de 5 veces, que además tengan un valor menor a \$1000 y se hayan vendido en los meses de abril y mayo.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/",expand=True)
df_aux.columns = ["ANHO","MES","DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo podríamos hacer este filtro?

Al filtro anterior, agregamos `or` (por eso se ocupa “`|`”). Así revisamos que el `MES` sea igual a 4 ó 5.

Para el resto de las condiciones usamos `and`, ya que se deben cumplir las 3.

¿Cómo calcular el promedio de ventas mensual?

Finalmente, a Don Juan le gustaría conocer el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO																																																																		
<pre>df["TOTAL"] = df["PRECIO"] * df["CANT"] print(df.head())</pre>																																																																		
RESULTADO																																																																		
<table border="1"><thead><tr><th></th><th>FECHA</th><th>NUM</th><th>CANT</th><th>CAT</th><th>NOMBRE</th><th>PRECIO</th><th>ANHO</th><th>MES</th><th>DIA</th><th>TOTAL</th></tr></thead><tbody><tr><td>0</td><td>2018/9/22</td><td>23346</td><td>18</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>9</td><td>22</td><td>16020</td></tr><tr><td>1</td><td>2018/10/7</td><td>23346</td><td>35</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>10</td><td>7</td><td>31150</td></tr><tr><td>2</td><td>2018/7/1</td><td>23346</td><td>13</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>7</td><td>1</td><td>11570</td></tr><tr><td>3</td><td>2018/8/3</td><td>23346</td><td>10</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>8</td><td>3</td><td>8900</td></tr><tr><td>4</td><td>2018/7/13</td><td>23346</td><td>37</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>7</td><td>13</td><td>32930</td></tr></tbody></table>		FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL	0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020	1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150	2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570	3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900	4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930
	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL																																																								
0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020																																																								
1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150																																																								
2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570																																																								
3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900																																																								
4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930																																																								

¿Cómo podríamos hacer este filtro?

Se podría hacer con un **pivot_table**. No obstante, antes es necesario sacar el total de ventas por día.

CÓDIGO																																																																		
<pre>df["TOTAL"] = df["PRECIO"] * df["CANT"] print(df.head())</pre>																																																																		
RESULTADO																																																																		
<table border="1"><thead><tr><th></th><th>FECHA</th><th>NUM</th><th>CANT</th><th>CAT</th><th>NOMBRE</th><th>PRECIO</th><th>ANHO</th><th>MES</th><th>DIA</th><th>TOTAL</th></tr></thead><tbody><tr><td>0</td><td>2018/9/22</td><td>23346</td><td>18</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>9</td><td>22</td><td>16020</td></tr><tr><td>1</td><td>2018/10/7</td><td>23346</td><td>35</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>10</td><td>7</td><td>31150</td></tr><tr><td>2</td><td>2018/7/1</td><td>23346</td><td>13</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>7</td><td>1</td><td>11570</td></tr><tr><td>3</td><td>2018/8/3</td><td>23346</td><td>10</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>8</td><td>3</td><td>8900</td></tr><tr><td>4</td><td>2018/7/13</td><td>23346</td><td>37</td><td>ABARROTES</td><td>Aceite</td><td>890</td><td>2018</td><td>7</td><td>13</td><td>32930</td></tr></tbody></table>		FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL	0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020	1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150	2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570	3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900	4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930
	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL																																																								
0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020																																																								
1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150																																																								
2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570																																																								
3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900																																																								
4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930																																																								

¿Cómo podríamos hacer este filtro?

Como tenemos la cantidad y el precio unitario, debemos comenzar calculando una nueva columna con esta multiplicación.

Ya tenemos el total de ventas por mes, ahora podemos sacar el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO																																																																								
<pre>import numpy as np df3 = df.pivot_table(index="MES",values="TOTAL", columns="CAT", aggfunc=[np.mean,np.std,np.min,np.max,]) print(df3.head())</pre>																																																																								
RESULTADO																																																																								
<table border="1"><thead><tr><th></th><th>CAT</th><th>ABARROTES</th><th>ASEO</th><th>PERSONAL</th><th>CARNES</th><th>FRUTAS</th><th>...</th><th>FRUTAS</th><th>LIMPIEZA</th><th>PANADERIA</th><th>VERDURAS</th></tr></thead><tbody><tr><td>1</td><td>27590.0</td><td>60420.0</td><td>234530.0</td><td>41420.0</td><td>...</td><td>15314.064451</td><td>15918.487767</td><td>30855.105034</td><td>16126.047873</td><td></td><td></td></tr><tr><td>10</td><td>31150.0</td><td>57360.0</td><td>255600.0</td><td>21830.0</td><td>...</td><td>6555.221583</td><td>18143.579213</td><td>153825.787103</td><td>13818.547922</td><td></td><td></td></tr><tr><td>11</td><td>27819.0</td><td>119500.0</td><td>166260.0</td><td>52320.0</td><td>...</td><td>13755.764347</td><td>9341.328867</td><td>164193.210226</td><td>13555.516712</td><td></td><td></td></tr><tr><td>12</td><td>43619.0</td><td>73140.0</td><td>233290.0</td><td>53410.0</td><td>...</td><td>22085.584121</td><td>22781.992889</td><td>146829.421605</td><td>10400.431241</td><td></td><td></td></tr><tr><td>2</td><td>32930.0</td><td>100380.0</td><td>230840.0</td><td>42510.0</td><td>...</td><td>11840.789318</td><td>15890.647238</td><td>161624.793673</td><td>13897.433016</td><td></td><td></td></tr></tbody></table>		CAT	ABARROTES	ASEO	PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS	1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873			10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922			11	27819.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	164193.210226	13555.516712			12	43619.0	73140.0	233290.0	53410.0	...	22085.584121	22781.992889	146829.421605	10400.431241			2	32930.0	100380.0	230840.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016		
	CAT	ABARROTES	ASEO	PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS																																																													
1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873																																																															
10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922																																																															
11	27819.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	164193.210226	13555.516712																																																															
12	43619.0	73140.0	233290.0	53410.0	...	22085.584121	22781.992889	146829.421605	10400.431241																																																															
2	32930.0	100380.0	230840.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016																																																															

Para saber el desglose de ventas mensuales por categoría, podríamos haber puesto **MES** o **CAT** en columnas o **index** (filas) indiferentemente.

CÓDIGO																																																																								
<pre>import numpy as np df3 = df.pivot_table(index="MES",values="TOTAL", columns="CAT", aggfunc=[np.mean,np.std,np.min,np.max,]) print(df3.head())</pre>																																																																								
RESULTADO																																																																								
<table border="1"><thead><tr><th></th><th>CAT</th><th>ABARROTES</th><th>ASEO</th><th>PERSONAL</th><th>CARNES</th><th>FRUTAS</th><th>...</th><th>FRUTAS</th><th>LIMPIEZA</th><th>PANADERIA</th><th>VERDURAS</th></tr></thead><tbody><tr><td>1</td><td>27590.0</td><td>60420.0</td><td>234530.0</td><td>41420.0</td><td>...</td><td>15314.064451</td><td>15918.487767</td><td>30855.105034</td><td>16126.047873</td><td></td><td></td></tr><tr><td>10</td><td>31150.0</td><td>57360.0</td><td>255600.0</td><td>21830.0</td><td>...</td><td>6555.221583</td><td>18143.579213</td><td>153825.787103</td><td>13818.547922</td><td></td><td></td></tr><tr><td>11</td><td>27819.0</td><td>119500.0</td><td>166260.0</td><td>52320.0</td><td>...</td><td>13755.764347</td><td>9341.328867</td><td>164193.210226</td><td>13555.516712</td><td></td><td></td></tr><tr><td>12</td><td>43619.0</td><td>73140.0</td><td>233290.0</td><td>53410.0</td><td>...</td><td>22085.584121</td><td>22781.992889</td><td>146829.421605</td><td>10400.431241</td><td></td><td></td></tr><tr><td>2</td><td>32930.0</td><td>100380.0</td><td>230840.0</td><td>42510.0</td><td>...</td><td>11840.789318</td><td>15890.647238</td><td>161624.793673</td><td>13897.433016</td><td></td><td></td></tr></tbody></table>		CAT	ABARROTES	ASEO	PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS	1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873			10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922			11	27819.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	164193.210226	13555.516712			12	43619.0	73140.0	233290.0	53410.0	...	22085.584121	22781.992889	146829.421605	10400.431241			2	32930.0	100380.0	230840.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016		
	CAT	ABARROTES	ASEO	PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS																																																													
1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873																																																															
10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922																																																															
11	27819.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	164193.210226	13555.516712																																																															
12	43619.0	73140.0	233290.0	53410.0	...	22085.584121	22781.992889	146829.421605	10400.431241																																																															
2	32930.0	100380.0	230840.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016																																																															

Lo importante es que en **values** esté el **TOTAL** (que fue la nueva columna que calculamos) y en **aggfunc** incorporar los estadísticos descriptivos del paquete **numpy** que se requieran calcular del **TOTAL**.

Además, guardaremos el reporte generado en **pivot_table**:

CÓDIGO
<pre>df3.to_csv("reporte_ventas.csv") df.to_csv("dataframe_reporte_ventas.csv")</pre>

- En este ejemplo práctico, conseguimos ver el proceso completo de procesamiento de información, mediante diversas técnicas y herramientas.
- Partimos de una lista de datos que no era muy valiosa (porque estaba sucia, y no estaba contextualizada con los datos de los productos), y terminamos con información muy valiosa que puede ser crítica para este negocio.
- Para lograr todos los procedimientos revisados en el ejemplo, usamos las herramientas y técnicas que sirven para **limpiar datos, consolidarlos y finalmente, extraer información valiosa a partir de ellos**.

Caso práctico:

```
# importamos la libreria
import pandas as pd

# creamos el dataframe
#df_pacientes = pd.read_csv("D:\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\Python\\DataFrames\\pacientes.csv")
#df_nacionalidad = pd.read_csv("D:\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\nacionalidad.csv")
df_pacientes = pd.read_csv("C:\\Users\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\pacientes.csv")
df_nacionalidad = pd.read_csv("C:\\Users\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\nacionalidad.csv")
print(df_pacientes)
print(df_nacionalidad)

# creamos dataframe con los nombre
df_nombres = df_pacientes["Nombre"].str.split(" ", expand=True)
df_nombres.columns = ["Primer Nombre", "Segundo Nombre", "Tercer Nombre", "Cuarto Nombre"]
print(df_nombres)

# unimos los dataframe (df_pacientes y df_nombres)
df_pacientes = df_pacientes.join(df_nombres)
print(df_pacientes)

# unimos los dataframe (df_pacientes y df_nacionalidad)
df_pacientes = df_pacientes.merge(df_nacionalidad, left_on="RUT", right_on="RUT", how="left")
print(df_pacientes)

#1 - En df_pacientes, filtra a los pacientes que no sean de Chile
print(df_pacientes.loc[df_pacientes["País_origen"] != "Chile"])

#2 - En df_pacientes , filtra a los pacientes que no sean de Chile y que tengan una deuda mayor a $1.000.000
print(df_pacientes.loc[(df_pacientes["País_origen"] != "Chile") & (df_pacientes["Monto_Deuda"] > 1000000)])

#3 - En df_pacientes , filtra a los pacientes que no sean de Chile y que tengan una deuda mayor a $1.000.000
# y que tengan previsión FONASA
print(df_pacientes.loc[(df_pacientes["País_origen"] != "Chile") & (df_pacientes["Monto_Deuda"] > 1000000) &
(df_pacientes["Previsión"] == "FONASA")])
```

```

#4 - En df_pacientes , filtra a los pacientes que sean de Chile pero solo de las ciudades de Santiago o Concepción,
# que tengan una deuda mayor a $1.000.000 y previsión FONASA.
print(df_pacientes.loc[(df_pacientes["País_origen"] == "Chile") &
((df_pacientes["Ciudad_Residencia"] == "Santiago") | (df_pacientes["Ciudad_Residencia"] == "Concepción")) &
(df_pacientes["Monto_Deuda"] > 1000000) & (df_pacientes["Previsión"] == "FONASA"))]

#5 - Llegó una nueva base de datos de pacientes. Carga esta información dn df_pacientes3
df_pacientes3 = pd.read_csv("C:\\Users\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\Python\\")
print(df_pacientes3.dtypes)

#6 - Hay pacientes con RUT terminado en J (se asume que son filas creadas por error). Elimina estas filas de df_pacientes3
#print(df_pacientes3.loc[df_pacientes3["RUT"].str.contains("J")])
#df_pacientes3 = df_pacientes3.drop(df_pacientes3.loc[df_pacientes3["RUT"].str.contains("J")].index)
df_pacientes3 = df_pacientes3.loc[df_pacientes3["RUT"].str[-1:] != "J"]
#print(df_pacientes3.loc[df_pacientes3["RUT"].str.contains("J")])
print(df_pacientes3)

#7 - Es necesario unir df_paciente3 a df_pacientes
print("--- datos_pacientes.csv ---")
print(df_pacientes.dtypes)
print("--- datos_pacientes3.csv ---")
print(df_pacientes3.dtypes)
df_pacientes = df_pacientes.set_index("RUT")
df_pacientes3 = df_pacientes3.set_index("RUT")
# append porque son columnas del mismo nombre
df_pacientes = df_pacientes.append(df_pacientes3)
print(df_pacientes)

#8 - Ordenar df_pacientes de forma ascendentes mediante la columna "Monto_Deuda" y "Fecha_Nacimiento" (en ese orden)
df_pacientes = df_pacientes.sort_values(by=["Monto_Deuda", "Fecha_Nacimiento"], ascending=True)
print(df_pacientes)

#9 - Extraer la media, mínimo y máximo de la deuda por tipo de previsión mediante una pivot table
import numpy as np
pt = df_pacientes.pivot_table(index="Previsión", values="Monto_Deuda", aggfunc=[np.mean, np.min, np.max])
print(pt)

#10 - Haz un pivot table, por RUT del médico, con la suma de cada consulta
df_consultas = pd.read_csv("C:\\Users\\Usuario\\OneDrive - Bogado Ingenieros Consultores SpA\\MIRESPALDO\\Cursos\\Python\\")
print(df_consultas.dtypes)
#df_consultas["Costo_consulta"] = df_consultas["Costo_consulta"].astype("int64")
pt2 = df_consultas.pivot_table(index="RUT", values="Costo_consulta", aggfunc=np.sum)
print(pt2)

#11 - Haz un merge de esta información, mediante el RUT de cada médico, con df_consultas
df_consultas = df_consultas.merge(pt2, left_on="RUT", right_on="RUT", how="left", suffixes=("_pacientes", "_medicos"))
print(df_consultas)

```

Glosario:

- **Archivo CSV:** Archivo de texto plano que replica una matriz de datos. Cada fila representa a medidas o valores de cada instancia. Cada columna representa distintos tipos de datos para cada variable específico. En general, los datos dentro de las columnas se separan por una "," o un ":". Además, se usa que la primera fila de este archivo lleve el nombre de las columnas.
- **Ciencia de datos:** "Es la disciplina que permite encontrar patrones predecibles en sets de datos estructurados, y no estructurados" (Dhar, 2013). También, "es un concepto que busca unificar estadísticas, análisis de datos, inteligencia artificial y cualquier otro método similar para poder entender y analizar fenómenos con los datos." (Hayashi, 1998). Un ejemplo de su uso es la astronomía, donde diariamente se genera un exabyte de datos (la cantidad de información que se genera en internet en un solo día).
- **Data Frame:** Los data frames son la estructura más usada en la librería Pandas. Se puede imaginar como una matriz de datos. Cada columna, representa datos para una variable específica. Y cada fila corresponde a medidas o valores de cada instancia. Cada columna será de un tipo específico, y cada fila podrá tener valores de distintos tipos. En esta estructura de datos, se agregan filas y columnas a preferencia. Las columnas de un data frame son series. Es importante destacar que cada fila tendrá un índice, que nos permitirá acceder a esa fila en particular y su información específica.
- **Encoding:** Es la codificación del archivo. Python y Pandas ocupan uno por default, que a veces no carga bien caracteres en español (como tildes, ¡ y ñ). Para evitarlo se emplea el encoding "latin-1" al cargar un data frame.
- **Estadísticos descriptivos:** Describen las estadísticas básicas de un conjunto de datos de forma cuantitativa. Específicamente, en Pandas se calcula:
 - **count:** Cantidad de ocurrencias en una columna.
 - **mean:** promedio simple entre los valores de una columna.
 - **std:** Desviación estándar. Distancia promedio de todos los valores respecto al promedio.
 - **min:** valor mínimo entre los valores de una columna.
 - **max:** valor máximo entre los valores de una columna.
 - **25%,50%,75%:** valores al separar en cuartiles los datos.
- **Librería:** Otra forma de llamar a los paquetes, que son códigos encapsulados en Python que permiten ejecutar operaciones específicas por medio de sus funciones. Se necesita cargarlos mediante el comando import.

- **Pandas:** Es una librería de código abierto, que provee estructuras de datos de alto rendimiento y fáciles de usar. Además, provee herramientas para el análisis de datos en Python. Tiene dos estructuras de datos básicas: series y data frames.
- **Paquete random:** permite generar un número aleatorio. Esto se puede hacer de la siguiente manera:

```
import random
num_aleatorio = random.randint(a,b)
```

Donde [a,b] definen el rango en el que podemos crear el número aleatorio. En este caso el número aleatorio generado se almacena en la variable num_aleatorio.

- **Serie:** Es una lista de datos con un largo fijo, aquí no podemos agregar o quitar elementos, sólo modificarlos. Además, los datos de una serie son del mismo tipo.
- **Columna:** Sinónimo de serie. Estructura básica de organización de datos en Pandas. Un Data Frame tiene varias columnas, cada una con un tipo de datos.
- **Fila:** Instancia única de valores para cada set de datos. Cada fila tiene valores distintos por columnas. Un Data Frame tiene múltiples filas.
- **Index:** Identificador por fila. Puede ser un número o texto. No necesariamente es único, aunque algunas funciones de Pandas requieren que lo sea.
- **Índice:** Posición de una fila dentro de un Data Frame. Se empiezan a contar desde 0 hasta la cantidad de filas-1. Puede ser distinto al index (por ejemplo, que el index sea el RUT y el índice una posición determinada).
- **NaN:** Abreviación de “Not a Number”. En general, Pandas rellena valores faltantes con NaN.
- **Tipos de datos en pandas:** La librería Pandas tiene sus propios tipos de datos. Son equivalentes a los de Python, pero tienen nombres distintos y existen nuevos que permiten manejar datos de manera más eficiente. Los más importantes son:
 - **object:** Equivalente al string en Python.
 - **int64:** Equivalente a int en Python.
 - **float64:** Equivalente a float en Python.
 - **bool:** Igual que Python.
 - **datetime:** Dato de tipo fecha (no tiene equivalente en Python).
- **Función:** Encapsulamiento de un código. Puede recibir parámetros (ciertos valores), ejecuta cierta operación, y puede devolver un valor (retorno).

- **Intervalo:** Si se tiene el string s = “Ejemplo”. Al ejecutar s[1:4], el string resultante es “jem”, ya que extrae los caracteres en las posiciones 1,2, y 3. El 1:4 es el intervalo, y al definir de 1:4 (en el contexto de strings, diferente a loc), entonces en realidad estamos definiendo los números 1,2 y 3. Es decir, crea una colección de números enteros desde el rango inicial al final-1.
- **Iterar:** Repetición. Se usa dentro de un contexto de un for (o ciclo) para denominar el acto de obtener cada uno de los valores de una lista en las repeticiones de este ciclo.
- **Slice:** De la definición anterior, s[1:4] es un slice. Consiste en extraer un string a partir de otro, en base a un intervalo específico.
- **SQL:** Nomenclatura de programación usada para extraer datos de una base de datos. No se enseña en el curso.
- **String:** Tipo de dato en Python para representar textos. Un string está compuesto por caracteres (letras, espacios, símbolos, etc.).
- **Filtro o Criterio:** Hace referencia a la aplicación de una operación lógica a una columna en la función loc. Por ejemplo:

```
df.loc[df["Monto"] > 100000]
```

- **Librería numpy:** Al igual que Pandas, es una librería que permite aplicar ciertas funciones. En este caso, son funciones matemáticas que se aplican sobre la pivot_table. Es muy importante recordar que es necesario importar esta librería antes de usarla.
- **Operación lógica:** Es el resultado de la aplicación de un operador lógico. En el contexto de Pandas, entre una columna y algún valor.
- **Operador lógico:** Operadores numéricos (<, >, <=, >=) de igualdad (==, !=) que al utilizarse entre distintos valores o variables, da como resultado un valor booleano.
- **Operador lógico binario:** Permite unir operaciones lógicas para obtener un valor booleano como resultado. Se verán dos: and y or. La sintaxis general es:

(operación lógica) and/or (operación lógica)

En el caso del and, se deben cumplir (valor booleano True) ambas operaciones lógicas para que el valor booleano como resultado sea True.

En el caso del or, se debe cumplir (valor booleano True) al menos una operación lógica para que el valor booleano como resultado sea True.

En Pandas, y en particular los filtros dentro de loc, el and se representa por el carácter “&” y el or por el carácter “|”.

- **Tabla dinámica de Excel:** Herramienta de Excel que de forma similar a un pivot_table permite reordenar los datos de un archivo Excel por ciertas filas, columnas y valores. De esta manera, se elige los valores de qué columna formarán las filas de esta tabla, los valores de qué columna formarán las columnas de esta tabla, y los valores de qué columna se analizarán dentro de esta tabla. A estos últimos se les puede aplicar alguna función matemática (como media, desviación estándar, mínimo, máximo, cuenta, etc.).
- **Valores booleanos:** Tipo de dato de una variable. Puede ser True o False. Es la base de la lógica booleana.