TELEDUC
Educación a Distancia

PROCESAMIENTO Y EXTRACCIÓN DE INFORMACIÓN

>>> Parte 1. Herramientas avanzadas para la selección de datos

>>> Filtros Avanzados de un Data Frame

Recordemos

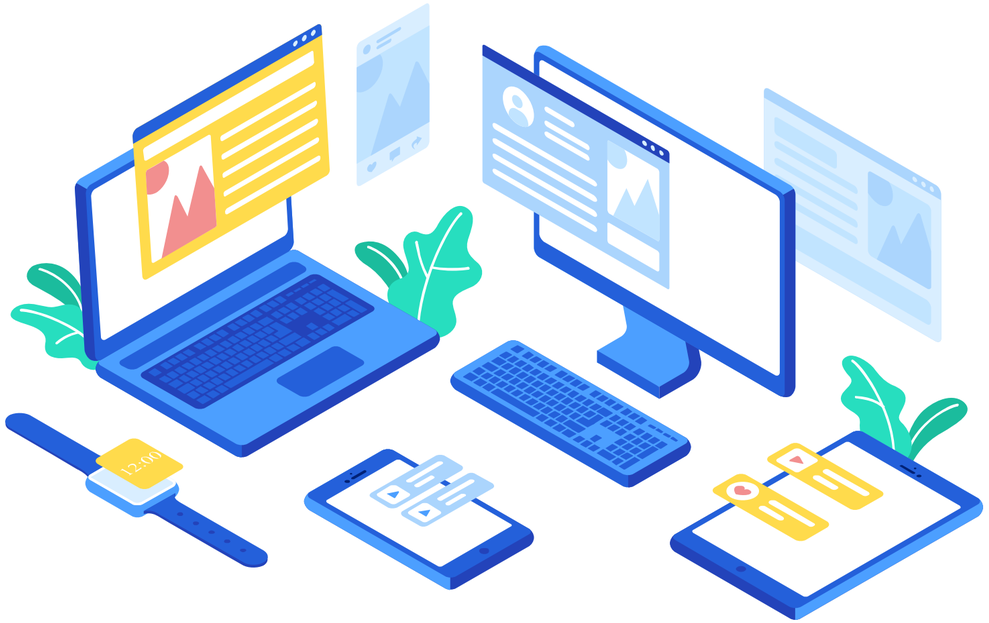
Vimos cómo “filtrar” de forma muy básica en un Data Frame

También vimos cómo limpiar datos para la extracción de información

Contexto

1

Ahora revisaremos filtros avanzados para un Data Frame y ocuparemos el mismo archivo CSV de ejemplo del módulo anterior, que simula los clientes de un banco y los montos de sus cuentas corrientes.



Filtro básico

Recuerda revisar el archivo **m4-ej1.py** →

Para filtrar a aquellos que tengan más de $5.000.000 en su cuenta. Recordemos, que lo hacíamos de la siguiente manera:

CÓDIGO

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

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

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
0	13.626.365-6	Javier Andrés López Castro	MASCULINO	1963/2/17	9889868
2	15.391.058-0	Paula Blanca Marín Campos	FEMENINO	1962/5/30	9004634
4	15.755.894-8	Bernardo Vicente Campos Rodríguez	MASCULINO	1987/7/11	6515702
5	17.399.932-8	Isidora Javiera Muñoz López	FEMENINO	1983/2/14	5507746
6	14.328.880-9	Rodrigo Rodrigo Castro González	MASCULINO	1975/6/25	9181732

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej2.py** →

¿Y si quisiéramos buscar todas las personas que tienen un monto **entre $5.000.000 y $7.000.000** en su cuenta?

CÓDIGO

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

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

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
4	15.755.894-8	Bernardo Vicente Campos Rodríguez	MASCULINO	1987/7/11	6515702
5	17.399.932-8	Isidora Javiera Muñoz López	FEMENINO	1983/2/14	5507746
8	7.043.426-3	Ignacia Constanza Marín Muñoz	FEMENINO	1986/2/27	6331299
10	5.843.498-4	Andrea Isabel Valenzuela Muñoz	FEMENINO	1995/7/8	5964375
11	17.609.01-8	Alejandro Juan Vergara Muñoz	MASCULINO	1974/5/20	6004107

Lo importante aquí es que combinamos operaciones lógicas mediante un and (específicamente con el carácter “&”). Por lo tanto, para ver los montos entre $5.000.000 y $7.000.000, se debe cumplir que sean mayores a $5.000.000 y menores a $7.000.000.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej3.py** →

¿Y si quisiéramos buscar todas las personas que tienen un monto **menor a $1.000.000** o **mayor a $9.000.000** en su cuenta?

CÓDIGO

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

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

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
0	13.626.365-6	Javier Andrés López Castro	MASCULINO	1963/2/17	9889868
2	15.391.058-0	Paula Blanca Marín Campos	FEMENINO	1962/5/30	9004634
6	14.328.880-9	Rodrigo Rodrigo Castro González	MASCULINO	1975/6/25	9181732
14	15.351.900-0	Alejandro Felipe González Castro	MASCULINO	1991/10/9	9380190
17	17.284.525-4	Alejandro Andrés Saavedra Muñoz	MASCULINO	1980/5/3	9195471

Combinamos dos operaciones lógicas mediante un `or` (específicamente con el carácter "`|`"). Para ver los montos menores a $1.000.000 o mayores a $9.000.000, podemos combinar ambas operaciones lógicas. No olviden incluir los paréntesis al combinar operaciones lógicas, de lo contrario Python arrojará error.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej4.py** →

Aún no hemos visto cómo hacer filtros con columnas que tengan `strings`. Por ejemplo, ¿cómo podríamos buscar todas las cuentas que tengan un apellido “González”?

CÓDIGO

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

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

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
3	8.296.689-7	Isabel Ignacia González Muñoz	FEMENINO	1972/3/20	1260523
6	14.328.880-9	Rodrigo Rodrigo Castro González	MASCULINO	1975/6/25	9181732
7	4.202.218-4	Rodrigo Vicente González Ruiz	MASCULINO	1995/2/2	3058987
14	15.351.900-0	Alejandro Felipe González Castro	MASCULINO	1991/10/9	9380190
15	16.306.988-6	Francisco Vicente Díaz González	MASCULINO	1983/8/27	8619031

Mediante de la función `contains(x)` de una columna tipo `object`, podemos saber si los elementos de la columna contienen el `string` que estamos buscando.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej5.py** →

¿Y si quisiéramos buscar aquellas personas cuyo primer nombre es Daniela?

CÓDIGO

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

print(df.loc[df["Nombre"].str.split(" ",expand=True)[0] == "Daniela"])
```

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
23	15.617.974-8	Daniela Isabel Rodríguez Castro	FEMENINO	1985/12/6	2142113
55	11.070.947-8	Daniela Valeria González Quiroga	FEMENINO	1983/10/16	255230
60	16.283.467-3	Daniela Isabel Vergara Robles	FEMENINO	2000/5/14	7805137

Primero hacemos un `split` en la columna Nombre, con espacio para separar los nombres, además de explicitar el parámetro `expand` en `True`.
Esto nos devuelve un Data Frame con cuatro columnas, que tienen los nombres de las personas separados.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej5.py** →

¿Y si quisiéramos buscar aquellas personas cuyo primer nombre es Daniela?

CÓDIGO

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

print(df.loc[df["Nombre"].str.split(" ",expand=True)[0] == "Daniela"])
```

El `[0]` nos permite obtener el primer nombre, y así podemos buscar cuáles de los primeros nombres son iguales a Daniela.

RESULTADO

```
         RUT                              Nombre      Sexo   Fecha_Nac    Monto
23  15.617.974-8   Daniela Isabel Rodríguez Castro  FEMENINO   1985/12/6  2142113
55  11.070.947-8  Daniela Valeria González Quiroga  FEMENINO  1983/10/16   255230
60  16.283.467-3     Daniela Isabel Vergara Robles  FEMENINO   2000/5/14  7805137
```

Recuerda revisar el archivo **m4-ej6.py** →

Filtros avanzados de un Data Frame

Por último, ¿cómo buscar a las personas de primer nombre Daniela, dígito verificador 8 y un monto menor a $1.000.000?

CÓDIGO

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

print(df.loc[(df["Nombre"].str.split(" ",expand=True)[0] == "Daniela") & (df["Monto"] < 1000000) & (df["RUT"].str[-1:] == "8")])
```

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
55	11.070.947-8	Daniela Valeria González Quiroga	FEMENINO	1983/10/16	255230

La primera operación lógica es la misma que hicimos anteriormente. Luego, vamos uniéndolas con paréntesis y separándolas con un &.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej6.py** →

Por último, ¿cómo buscar a las personas de primer nombre Daniela, dígito verificador 8 y un monto menor a $1.000.000?

CÓDIGO

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

print(df.loc[(df["Nombre"].str.split(" ",expand=True)[0] == "Daniela") & (df["Monto"] < 1000000) & (df["RUT"].str[-1:] == "8")])
```

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
55	11.070.947-8	Daniela Valeria González Quiroga	FEMENINO	1983/10/16	255230

La segunda operación lógica es para verificar que el monto sea menor a $1.000.000.

Filtros avanzados de un Data Frame

Recuerda revisar el archivo **m4-ej6.py** →

Por último, ¿cómo buscar a las personas de primer nombre Daniela, dígito verificador 8 y un monto menor a $1.000.000?

CÓDIGO

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

print(df.loc[(df["Nombre"].str.split(" ",expand=True)[0] == "Daniela") & (df["Monto"] < 1000000) & (df["RUT"].str[-1:] == "8")])
```

RESULTADO

```
                RUT                                Nombre      Sexo   Fecha_Nac   Monto
55  11.070.947-8  Daniela Valeria González Quiroga  FEMENINO  1983/10/16  255230
```

Finalmente, en la última operación lógica evaluamos que el dígito verificador sea igual a 8, a través de la función `str` para ocupar la columna “RUT” como un `string`. Luego, hacemos un `slice` que nos arroja el último carácter del `string`.

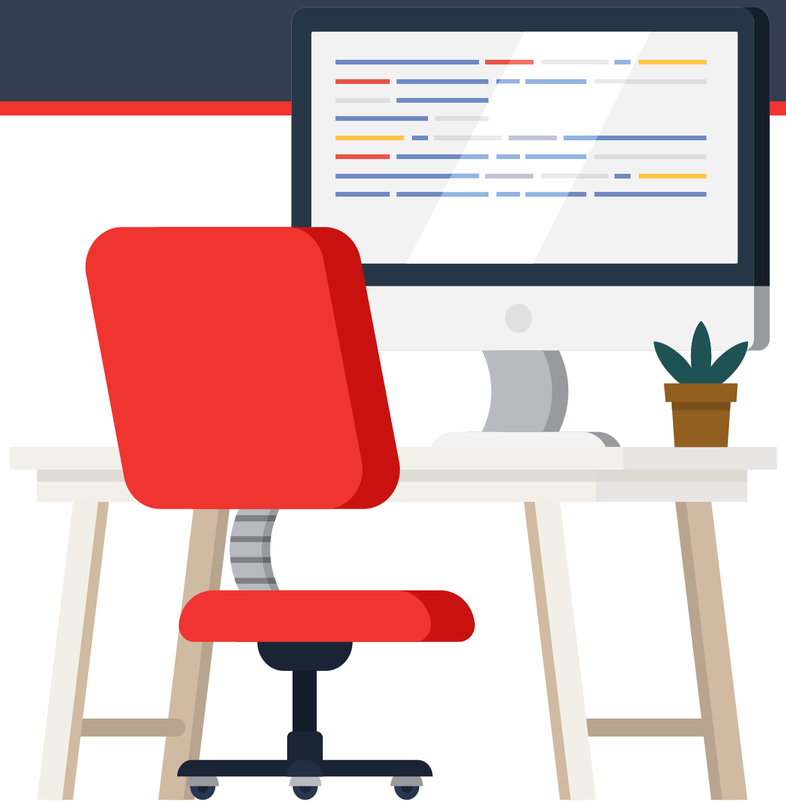
>>> Ordenar valores en un Data Frame

¿Cómo ordenar valores?

FUNCIÓN

`sort_values`

Esta función nos permite ordenar un Data Frame según una o más columnas, y puede ser en orden ascendente (menor a mayor) o descendente (mayor a menor).



Función sort_values

Recuerda revisar el archivo **m4-ej7.py** →

```
df = df.sort_values(by=(columna o lista de columnas), ascending = (True o False))
```

CÓDIGO

```
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)
```

RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
0	13.626.365-6	Javier Andrés López Castro	MASCULINO	1963/2/17	9889868
69	12.394.965-8	Juan Rodrigo Marín Rojas	MASCULINO	1973/6/22	9828562
94	5.126.125-1	Juan Andrés Marín López	MASCULINO	1995/1/10	9820196
19	9.721.797-1	Francisco Francisco Saavedra Campos	MASCULINO	1974/3/16	9697112
21	5.610.739-7	Pedro Ignacio Robles Campos	MASCULINO	1963/11/29	9641263

Ordenamos en forma descendente (lo que se observa al darle valor False al parámetro “ascending”).

Además, ordenamos en base a dos columnas. Primero la columna **Sexo** y luego la columna **Monto**.

Función sort_values

Recuerda revisar el archivo **m4-ej7.py** →

```
df = df.sort_values(by=(columna o lista de columnas), ascending = (True o False))
```

CÓDIGO

```
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)
```

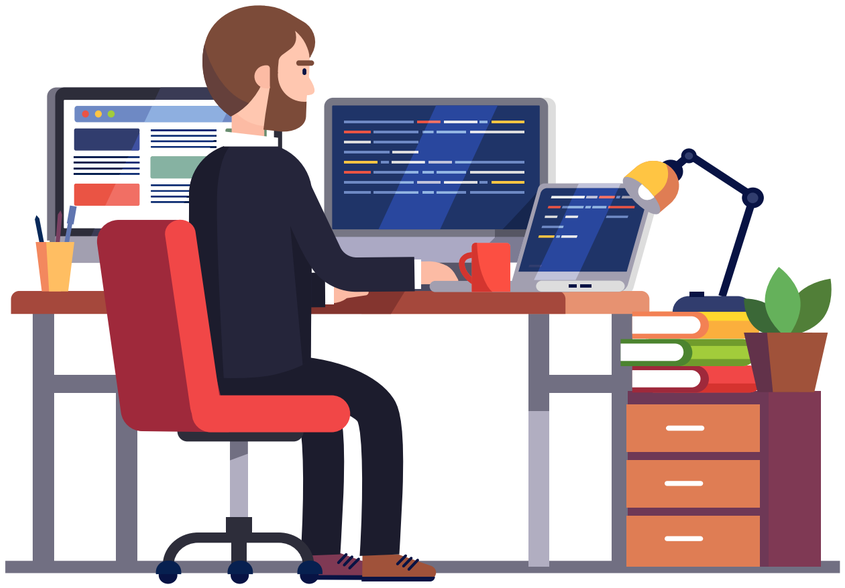
RESULTADO

	RUT	Nombre	Sexo	Fecha_Nac	Monto
0	13.626.365-6	Javier Andrés López Castro	MASCULINO	1963/2/17	9889868
69	12.394.965-8	Juan Rodrigo Marín Rojas	MASCULINO	1973/6/22	9828562
94	5.126.125-1	Juan Andrés Marín López	MASCULINO	1995/1/10	9820196
19	9.721.797-1	Francisco Francisco Saavedra Campos	MASCULINO	1974/3/16	9697112
21	5.610.739-7	Pedro Ignacio Robles Campos	MASCULINO	1963/11/29	9641263

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.

>>> Extracción de datos

¿Cómo extraer datos?



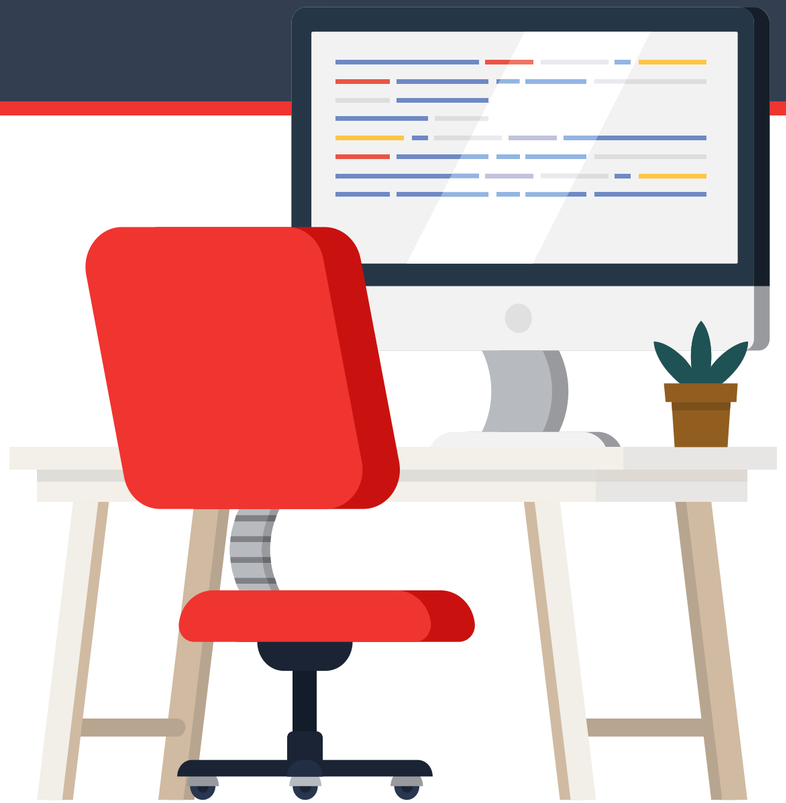
Existen herramientas sumamente útiles para extraer datos de un Data Frame.

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

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.



Función pivot

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**”.

Función pivot

Recuerda revisar el archivo **m4-ej8.py** →

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)
```

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
```

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

Recuerda revisar el archivo **m4-ej8.py** →

Función pivot

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)
```

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.

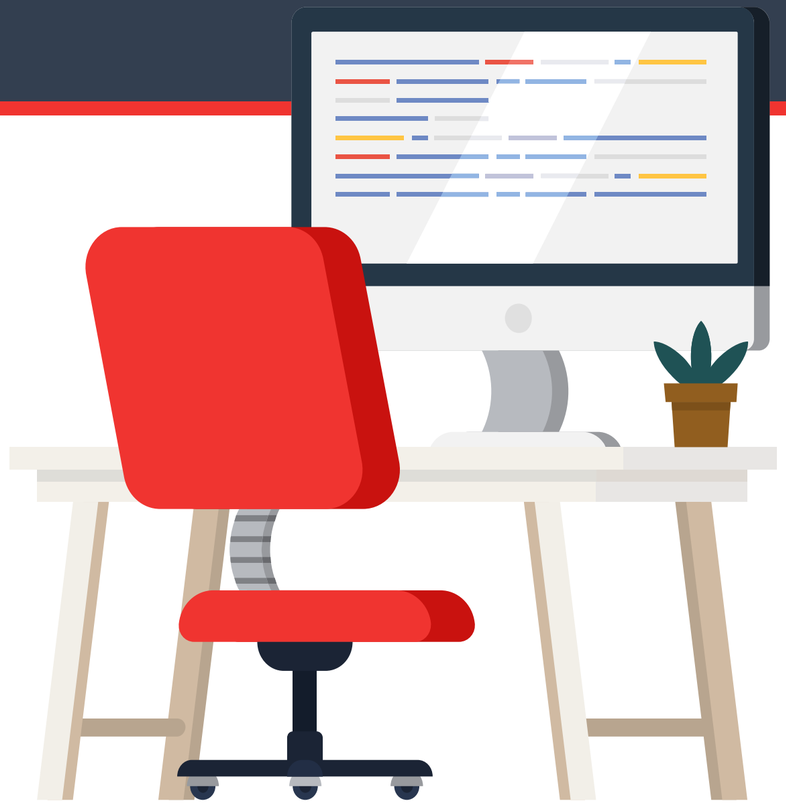
En este caso, como queremos saber la información de los bancos, las columnas serán el banco, y el valor para analizar es el monto.

Función pivot_table

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.



Función pivot_table

Recuerda revisar el archivo **m4-ej9.py** →

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

```
df.pivot_table(index=(columna que determinará las filas),columns=(columna que determinara 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.

Recuerda revisar el archivo **m4-ej9.py** →

Función pivot_table

Podemos hacer una tabla dinámica y calcular el monto promedio de todas las cuentas por banco:

```
df.pivot_table(index=(columna que determinará las filas),columns=(columna que determinara 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
```

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`.

Recuerda revisar el archivo **m4-ej10.py →**

Función pivot_table

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      FEMENINO     MASCULINO
BANCO
BancoA   9979530.0   9911177.0  626641.0  197530.0  4.282360e+06  5.615042e+06
BancoB   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 `{}`).

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.**

>>> Cierre

Has finalizado la revisión de los contenidos que corresponden a esta clase.

A continuación, te invitamos a estudiar la siguiente clase del módulo.