

## Tema: Pandas II

### Contenido

El propósito de esta practica es aprender los fundamentos de la librería Pandas, para ser aplicado posteriormente en la solución de problemas de aprendizaje automático y ciencias de datos.

### Objetivo Especifico

- a) Instalar Pandas y utilizar entornos de desarrollo IDE como Pycharm o utilizar un editor de texto como Geany.
- b) Aprender la sintaxis de Pandas.

### Material y Equipo

- a) Virtual Box
- b) Linux Mint 21.3

### Introduccion Teorica

Pandas es una librería de Python que proporciona herramientas para la manipulación y análisis exploratorio de conjuntos de datos. Trabaja principalmente con estructuras de datos bidimensionales e indexadas denominadas DataFrame, e implementa funciones avanzadas y eficientes que permiten llevar a cabo operaciones como transformación, selección, agrupación, entre otros.

En esta esta guía se proporciona una descripción de Pandas y se presentará, mediante ejemplos, la funcionalidad necesaria para abordar la mayoría casos en ciencia de datos.

## Procedimiento

### Instalación de Pip

Instalar Pip, en caso de no estar instalado.

```
$ sudo install python3-pip
```

```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
usuario@vostro-3400:~$ sudo apt install python3-pip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  libncurses5-dev linux-headers-5.15.0-70 linux-headers-5.15.0-70-generic
  linux-headers-5.15.0-73 linux-headers-5.15.0-73-generic
  linux-image-5.15.0-70-generic linux-image-5.15.0-73-generic
  linux-modules-5.15.0-70-generic linux-modules-5.15.0-73-generic
  linux-modules-extra-5.15.0-70-generic linux-modules-extra-5.15.0-73-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-setuptools
  python3-wheel python3.10-dev
Paquetes sugeridos:
  apache2 | lighttpd | httpd python3-setuptools-doc
Se instalarán los siguientes paquetes NUEVOS:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-pip python3-setuptools
  python3-wheel python3.10-dev
0 actualizados, 11 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 1,337 kB/7,563 kB de archivos.

```

```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
1 all.deb ...
Desempaquetando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-wheel previamente no seleccionado.
Preparando para desempaquetar .../09-python3-wheel_0.37.1-2ubuntu0.22.04.1_all.d
eb ...
Desempaquetando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-pip previamente no seleccionado.
Preparando para desempaquetar .../10-python3-pip_22.0.2+dfsg-1ubuntu0.4_all.deb
...
Desempaquetando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando javascript-common (11+nmu1) ...
Configurando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Configurando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Configurando libpython3.10-dev:amd64 (3.10.12-1~22.04.5) ...
Configurando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando python3.10-dev (3.10.12-1~22.04.5) ...
Configurando libjs-jquery (3.6.0+dfsg+~3.5.13-1) ...
Configurando libjs-underscore (1.13.2~dfsg-2) ...
Configurando libpython3-dev:amd64 (3.10.6-1~22.04) ...
Configurando libjs-sphinxdoc (4.3.2-1) ...
Configurando python3-dev (3.10.6-1~22.04) ...
Procesando disparadores para man-db (2.10.2-1) ...
Progreso: [ 98%] [#####..]

```

### 3 ASI104 - Guía 8

Verificar que Pip esta correctamente instalado

\$ pip3 --version

```
usuario@vostro-3400: ~  
Archivo  Editar  Ver  Buscar  Terminal  Pestañas  Ayuda  
usuario@vostro-3400: ~ x usuario@vostro-3400: ~ x  
usuario@vostro-3400:~$ pip3 --version  
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)  
usuario@vostro-3400:~$
```

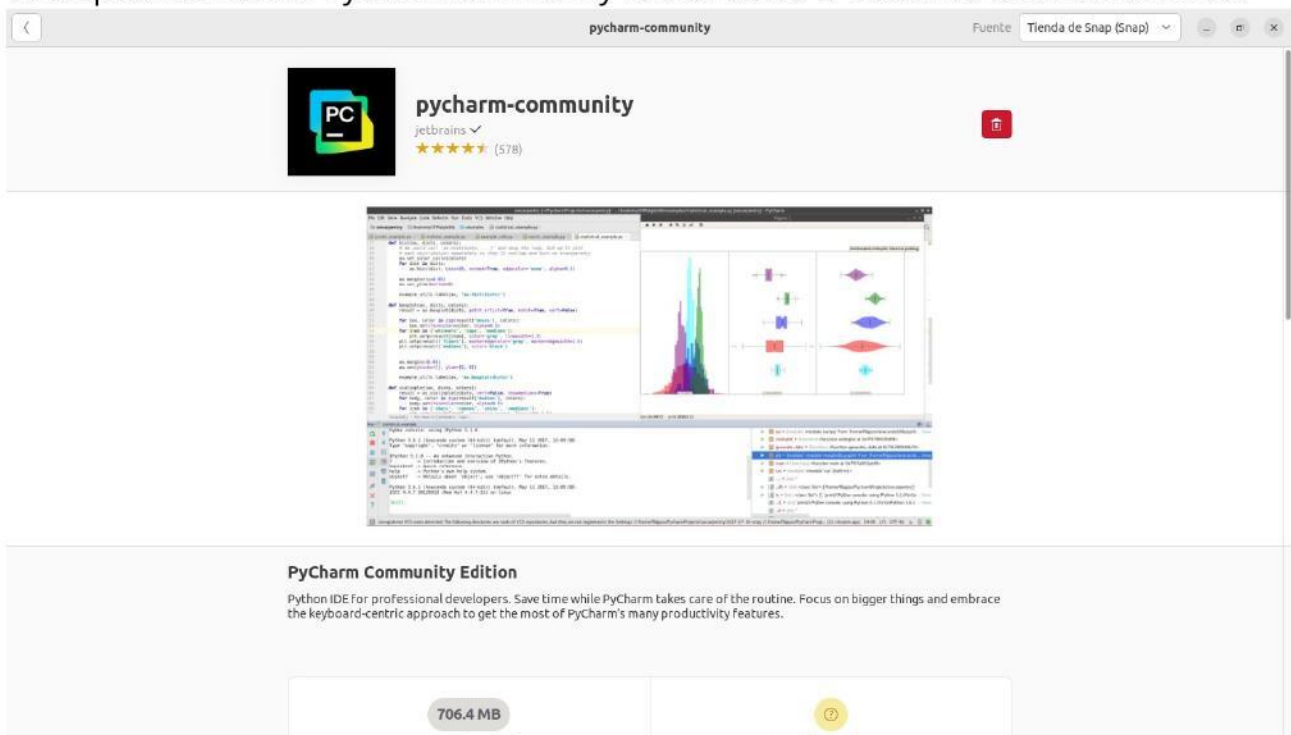
### Instalación de Pandas

Proceder a la instalación de Pandas

#sudo pip install pandas

### Instalación de Pycharm Community Edition o Geany

Otra opción es instalar Pycharm Community Edition desde el Gestor de Software de Linux:

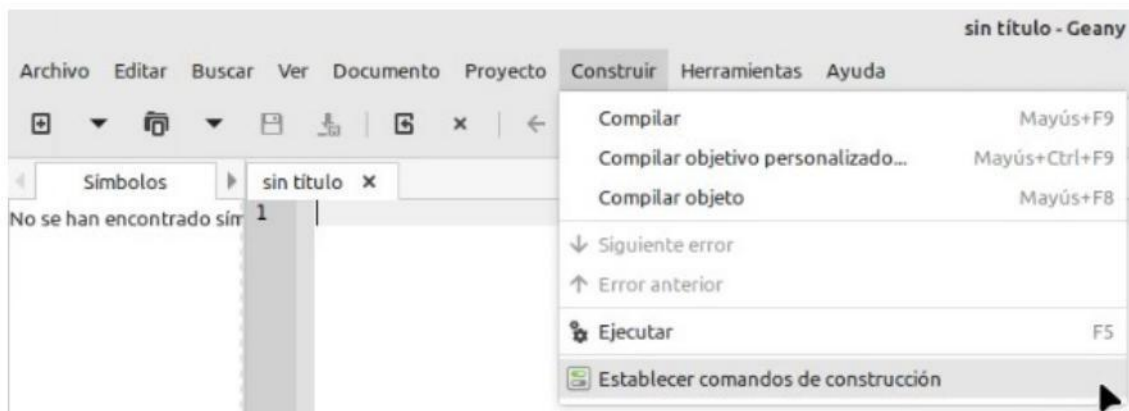


**Otra opción más ligera es instalar Geany como IDE:**

```
#sudo apt-get install geany
```

Instalado geany se debe configurar las opciones de compilación para que Geany pueda interpretar y ejecutar un programa en Python.

Desde el menú principal de Geany seleccionar la opción "Construir", después seleccionar "Establecer comandos de construcción"



## 5 ASI104 - Guía 8

En la ventana de “Establecer los comandos de construcción”, asegurarse que las opciones “Comandos de Python” y “Comandos de ejecución” estén configurados con la cadena que se indica en el campo “comando”.

Establecer los comandos de construcción

#	Etiqueta	Comando	Directorio de trabajo	Reiniciar
<b>Comandos de Python</b>				
1.	Compile	python3 -m py_compile "%f"		
2.				
3.	Lint	pep8 --max-line-length=80 '		
Expresión regular de error:		(.+:([0-9]+):([0-9]+)		
<b>Comandos independientes</b>				
1.	Compilar	make		
2.	Compilar objetivo personalizado...	make		
3.	Compilar objeto	make %e.o		
4.				
Expresión regular de error:				
<i>Nota: El elemento 2 abre un diálogo y añade la respuesta al comando.</i>				
<b>Comandos de ejecución</b>				
1.	Execute	python3 "%F"		
2.				
<i>%d, %e, %f, %p y %l se sustituirán en los campos de comandos y directorios, consulte el manual para más información.</i>				
			Cancelar	Aceptar



## Fundamentos de Pandas (Continuación)

### Modificar orden de las columnas de un Dataframe

El siguiente ejemplo muestra como modificar el orden de las columnas de un Dataframe

```
ejemplo01.py x
> Q 0 results
1 import pandas as pd
2
3 datos1 = {"Departamento": ["San Salvador", "San Salvador", "San Salvador", "La Libertad", "La Libertad", "La Libertad"],
4          "anio": [2000, 2001, 2002, 2001, 2002, 2003],
5          "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
6 frame = pd.DataFrame(datos1)
7 print(frame)
8 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento"])
9 print(datos2)
```

Salida:

```
/home/walter/PycharmProjects/ClaseTeoriaPython/
Departamento anio  valor
0 San Salvador 2000   1.5
1 San Salvador 2001   1.7
2 San Salvador 2002   3.6
3 La Libertad  2001   2.4
4 La Libertad  2002   2.9
5 La Libertad  2003   3.2
anio  valor  Departamento
0 2000   1.5  San Salvador
1 2001   1.7  San Salvador
2 2002   3.6  San Salvador
3 2001   2.4  La Libertad
4 2002   2.9  La Libertad
5 2003   3.2  La Libertad
```

## Modificar orden de las columnas de un Dataframe

El siguiente ejemplo muestra como modificar el orden de las columnas de un Dataframe, y que sucede si se agrega una columna adicional.

```
ejemplo01.py x
> Q- 0 results ↑ ↓ 🔍
1 import pandas as pd
2
3 datos1 = {"Departamento": ["San Salvador", "San Salvador", "San Salvador", "La Libertad", "La Libertad", "La Libertad"],
4           "anio": [2000, 2001, 2002, 2001, 2002, 2003],
5           "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
6
7 frame = pd.DataFrame(datos1)
8 print(frame)
9
10 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento", "Habitantes"])
11 print(datos2)
```

Salida:

```

↑ /home/walter/PycharmProjects/ClaseTeoriaPython/
↓
⌕
🔍
🗑

```

	Departamento	anio	valor
0	San Salvador	2000	1.5
1	San Salvador	2001	1.7
2	San Salvador	2002	3.6
3	La Libertad	2001	2.4
4	La Libertad	2002	2.9
5	La Libertad	2003	3.2

	anio	valor	Departamento	Habitantes
0	2000	1.5	San Salvador	NaN
1	2001	1.7	San Salvador	NaN
2	2002	3.6	San Salvador	NaN
3	2001	2.4	La Libertad	NaN
4	2002	2.9	La Libertad	NaN
5	2003	3.2	La Libertad	NaN

## Obtener el contenido de una columna de un Dataframe

El siguiente ejemplo muestra dos formas de obtener el contenido de una columna de un Dataframe.

```
ejemplo01.py x
> Q- 0 results
1 import pandas as pd
2
3 datos1 = {"Departamento": ["San Salvador", "San Salvador", "San Salvador", "La Libertad", "La Libertad", "La Libertad"],
4          "anio": [2000, 2001, 2002, 2001, 2002, 2003],
5          "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
6 frame = pd.DataFrame(datos1)
7 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento", "Habitantes"])
8 print(datos2["Departamento"])
9 print(datos2.Departamento)
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/
0 San Salvador
1 San Salvador
2 San Salvador
3 La Libertad
4 La Libertad
5 La Libertad
Name: Departamento, dtype: object
0 San Salvador
1 San Salvador
2 San Salvador
3 La Libertad
4 La Libertad
5 La Libertad
Name: Departamento, dtype: object
```



## Modificar contenido de columna de un Dataframe

El siguiente ejemplo muestra como modificar el contenido de columna de una Dataframe.

```
ejemplo01.py x
> Q- 0 results ↑ ↓ 🔍 ⋮
1 import pandas as pd
2 import numpy as np
3
4 datos1 = {"Departamento": ["San Salvador", "San Salvador", "San Salvador", "La Libertad", "La Libertad", "La Libertad"],
5          "anio": [2000, 2001, 2002, 2001, 2002, 2003],
6          "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
7 frame = pd.DataFrame(datos1)
8 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento", "Habitantes"])
9 datos2["valor"] = 9.5
10 print(datos2["valor"])
11 datos2["valor"] = np.arange(6.0)
12 print(datos2.valor)
```

Salida:

```

↑ ↓ ↶ ↷ ⌂ 🗑️
/home/walter/PycharmProjects/ClaseTeoriaPython/.
0    9.5
1    9.5
2    9.5
3    9.5
4    9.5
5    9.5
Name: valor, dtype: float64
0    0.0
1    1.0
2    2.0
3    3.0
4    4.0
5    5.0
Name: valor, dtype: float64
```

## Agregar columna en un Dataframe

El siguiente ejemplo muestra otro método para agregar columna a un DataFrame.

```
ejemplo01.py x
1 import pandas as pd
2 import numpy as np
3
4 datos1 = {"Departamento": ["Chalatenango", "San Salvador", "San Miguel", "Sonsonate", "San Salvador", "La Paz"],
5           "anio": [2000, 2001, 2002, 2001, 2002, 2003],
6           "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
7 frame = pd.DataFrame(datos1)
8 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento", "Habitantes"])
9 datos2["valor"] = 9.5
10 datos2["valor"] = np.arange(6.0)
11 print(datos2)
12 datos2["Capital"] = datos2["Departamento"] == "San Salvador"
13 print(datos2)
```

Salida:

```
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/
  anio  valor  Departamento  Habitantes
0  2000    0.0  Chalatenango         NaN
1  2001    1.0  San Salvador         NaN
2  2002    2.0   San Miguel         NaN
3  2001    3.0   Sonsonate         NaN
4  2002    4.0  San Salvador         NaN
5  2003    5.0     La Paz         NaN
  anio  valor  Departamento  Habitantes  Capital
0  2000    0.0  Chalatenango         NaN   False
1  2001    1.0  San Salvador         NaN    True
2  2002    2.0   San Miguel         NaN   False
3  2001    3.0   Sonsonate         NaN   False
4  2002    4.0  San Salvador         NaN    True
5  2003    5.0     La Paz         NaN   False
```

## Borrado de columna de un Dataframe

El siguiente ejemplo muestra el borrado de columna de un DataFrame.

```
ejemplo01.py x
1 import pandas as pd
2 import numpy as np
3
4 datos1 = {"Departamento": ["Chalatenango", "San Salvador", "San Miguel", "Sonsonate", "San Salvador", "La Paz"],
5          "anio": [2000, 2001, 2002, 2001, 2002, 2003],
6          "valor": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
7 frame = pd.DataFrame(datos1)
8 datos2 = pd.DataFrame(datos1, columns=["anio", "valor", "Departamento", "Habitantes"])
9 datos2["valor"] = 9.5
10 datos2["valor"] = np.arange(6.0)
11
12 datos2["Capital"] = datos2["Departamento"] == "San Salvador"
13 print(datos2)
14
15 del datos2["Capital"]
16 datos2.columns
17 print(datos2)
```

Salida:

```

/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/
  anio  valor  Departamento  Habitantes  Capital
0  2000    0.0  Chalatenango         NaN    False
1  2001    1.0  San Salvador         NaN     True
2  2002    2.0   San Miguel         NaN    False
3  2001    3.0   Sonsonate         NaN    False
4  2002    4.0  San Salvador         NaN     True
5  2003    5.0     La Paz          NaN    False
  anio  valor  Departamento  Habitantes
0  2000    0.0  Chalatenango         NaN
1  2001    1.0  San Salvador         NaN
2  2002    2.0   San Miguel         NaN
3  2001    3.0   Sonsonate         NaN
4  2002    4.0  San Salvador         NaN
5  2003    5.0     La Paz          NaN
```

## Diccionarios anidados con Dataframe

Si un diccionario anidado es pasado a un DataFrame, pandas interpretará las claves del diccionario externo como las columnas y las claves internas como los índices de fila.

```
ejemplo01.py x
1 import pandas as pd
2 import numpy as np
3
4 poblacion = {"La Paz": {2000: 1.5, 2001: 1.7, 2002: 3.6},
5              "San Salvador": {2001: 2.4, 2002: 2.9}}
6
7 datos1= pd.DataFrame(poblacion)
8 print(datos1)
9 print(datos1.T)
10
```

Salida:

```
/home/walter/PycharmProjects/ClaseTeoriaPython/
La Paz  San Salvador
2000    1.5         NaN
2001    1.7         2.4
2002    3.6         2.9
2000    NaN         2.4
2001    1.5         2.9
2002    3.6         NaN
La Paz    1.5    1.7    3.6
San Salvador NaN    2.4    2.9
```

### Atributos name para columnas e índices de un Dataframe

Si el índice y las columnas de un DataFrame tienen sus atributos name configurados, estos también serán mostrados.

```
ejemplo01.py x
1  import pandas as pd
2  import numpy as np
3
4  poblacion = {"La Paz": {2000: 1.5, 2001: 1.7, 2002: 3.6},
5               "San Salvador": {2001: 2.4, 2002: 2.9}}
6
7  datos1 = pd.DataFrame(poblacion)
8
9  datos1.index.name = "anio"
10 datos1.columns.name = "estado"
11 print(datos1)
12
```

Salida:

```
/home/walter/PycharmProjects/ClaseTeoriaPython/
estado  La Paz  San Salvador
anio
2000      1.5      NaN
2001      1.7      2.4
2002      3.6      2.9

Process finished with exit code 0
```



### Método reindex aplicado a Series

Al invocar al método `reindex()` se reorganizan los datos según el nuevo índice, introduciendo valores faltantes (NaN) si aún no había valores de índice presentes:

```
ejemplo01.py x
1 import pandas as pd
2 import numpy as np
3
4 datos1 = pd.Series( data: [4.5, 7.2, -5.3, 3.6], index=["d", "b", "a", "c"])
5 print(datos1)
6 datos2 = datos1.reindex(["a", "b", "c", "d", "e", "f"])
7 print(datos2)
8
```

Salida:

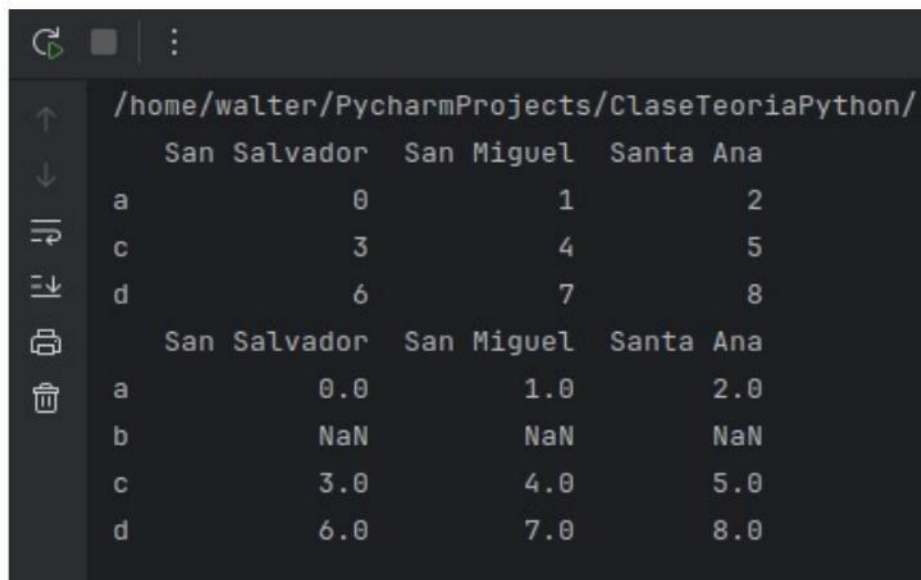
```
/home/walter/PycharmProjects/ClaseTeoriaPython/
d      4.5
b      7.2
a     -5.3
c      3.6
dtype: float64
a     -5.3
b      7.2
c      3.6
d      4.5
e      NaN
f      NaN
dtype: float64
```

### Método reindex aplicado a Dataframe

Al invocar al método `reindex()` se reorganizan los datos según el nuevo índice, introduciendo valores faltantes (NaN) si aún no había valores de índice presentes:

```
ejemplo01.py x
1 import pandas as pd
2 import numpy as np
3
4 datos1 = pd.DataFrame(np.arange(9).reshape((3, 3)),
5                       index=["a", "c", "d"],
6                       columns=["San Salvador", "San Miguel", "Santa Ana"])
7 print(datos1 )
8 datos2 = datos1 .reindex(index=["a", "b", "c", "d"])
9 print(datos2)
```

Salida:



	San Salvador	San Miguel	Santa Ana
a	0	1	2
c	3	4	5
d	6	7	8

	San Salvador	San Miguel	Santa Ana
a	0.0	1.0	2.0
b	NaN	NaN	NaN
c	3.0	4.0	5.0
d	6.0	7.0	8.0

## Lectura de archivo CSV con Pandas

El siguiente ejemplo muestra como leer un archivo csv desde Pandas.

```
ejemplo01.py x
1 import pandas as pd
2
3 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez/Pandas/clima_2018.csv')
4 print(clima.head())
5 datos_snow = clima.query('datatype == "SNOW" and value > 0')
6 print(datos_snow.head())
7
```

Salida:

```

/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python /home/walter
date datatype station attributes value
0 2018-01-01T00:00:00 PRCP GHCND:US1CTFR0039 ,,N, 0.0
1 2018-01-01T00:00:00 PRCP GHCND:US1NJBG0015 ,,N, 0.0
2 2018-01-01T00:00:00 SNOW GHCND:US1NJBG0015 ,,N, 0.0
3 2018-01-01T00:00:00 PRCP GHCND:US1NJBG0017 ,,N, 0.0
4 2018-01-01T00:00:00 SNOW GHCND:US1NJBG0017 ,,N, 0.0
date datatype station attributes value
114 2018-01-01T00:00:00 SNOW GHCND:US1NYWC0019 ,,N, 25.0
699 2018-01-04T00:00:00 SNOW GHCND:US1NJBG0015 ,,N, 229.0
702 2018-01-04T00:00:00 SNOW GHCND:US1NJBG0017 ,,N, 10.0
706 2018-01-04T00:00:00 SNOW GHCND:US1NJBG0018 ,,N, 46.0
713 2018-01-04T00:00:00 SNOW GHCND:US1NJES0018 ,,N, 10.0

Process finished with exit code 0

```

## Sentencia SQL equivalente

```
SELECT * FROM clima WHERE datatype == "SNOW" AND value > 0;
```

## Método query para consultas a Dataframe

El siguiente ejemplo muestra el uso del método query para realizar consultas a un DataFrame.

```
ejemplo01.py x
1 import numpy as np
2 import pandas as pd
3
4 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/clima_2018.csv')
5 print(clima.head())
6 datos_nieve = clima.query('datatype == "SNOW" and value > 0')
7 print(datos_nieve)
8 datos_nieve = clima.query('datatype == "PRCP" and value > 0')
9 print(datos_nieve)
10
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python /home/walter/...
date datatype station attributes value
0 2018-01-01T00:00:00 PRCP GHCND:US1CTFR0039 , , N, 0.0
1 2018-01-01T00:00:00 PRCP GHCND:US1NJB00015 , , N, 0.0
2 2018-01-01T00:00:00 SNOW GHCND:US1NJB00015 , , N, 0.0
3 2018-01-01T00:00:00 PRCP GHCND:US1NJB00017 , , N, 0.0
4 2018-01-01T00:00:00 SNOW GHCND:US1NJB00017 , , N, 0.0
...
114 2018-01-01T00:00:00 SNOW GHCND:US1NYWC0019 , , N, 25.0
699 2018-01-04T00:00:00 SNOW GHCND:US1NJB00015 , , N, 229.0
702 2018-01-04T00:00:00 SNOW GHCND:US1NJB00017 , , N, 10.0
706 2018-01-04T00:00:00 SNOW GHCND:US1NJB00018 , , N, 46.0
713 2018-01-04T00:00:00 SNOW GHCND:US1NJB00018 , , N, 10.0
...
77075 2018-12-24T00:00:00 SNOW GHCND:US1NJMS0097 , , N, 25.0
77085 2018-12-24T00:00:00 SNOW GHCND:US1NJPS0012 , , N, 3.0
77094 2018-12-24T00:00:00 SNOW GHCND:US1NJPS0025 , , N, 20.0
77120 2018-12-24T00:00:00 SNOW GHCND:US1NYWC0018 , , N, 18.0
78432 2018-12-30T00:00:00 SNOW GHCND:US1NYWC0018 , , N, 5.0

[639 rows x 5 columns]
date datatype station attributes value
698 2018-01-04T00:00:00 PRCP GHCND:US1NJB00015 , , N, 20.6
701 2018-01-04T00:00:00 PRCP GHCND:US1NJB00017 , , N, 2.0
705 2018-01-04T00:00:00 PRCP GHCND:US1NJB00018 , , N, 8.6
707 2018-01-04T00:00:00 PRCP GHCND:US1NJB00023 , , N, 1.5
712 2018-01-04T00:00:00 PRCP GHCND:US1NJB00018 , , N, 1.3
...
78736 2018-12-31T00:00:00 PRCP GHCND:USW00054787 , , W, 28.7
78743 2018-12-31T00:00:00 PRCP GHCND:USW00094728 , , W, 2400 25.9
78751 2018-12-31T00:00:00 PRCP GHCND:USW00094741 , , W, 29.2
```

## Sentencia SQL equivalente

SELECT \* FROM clima WHERE datatype == "SNOW" AND value > 0;

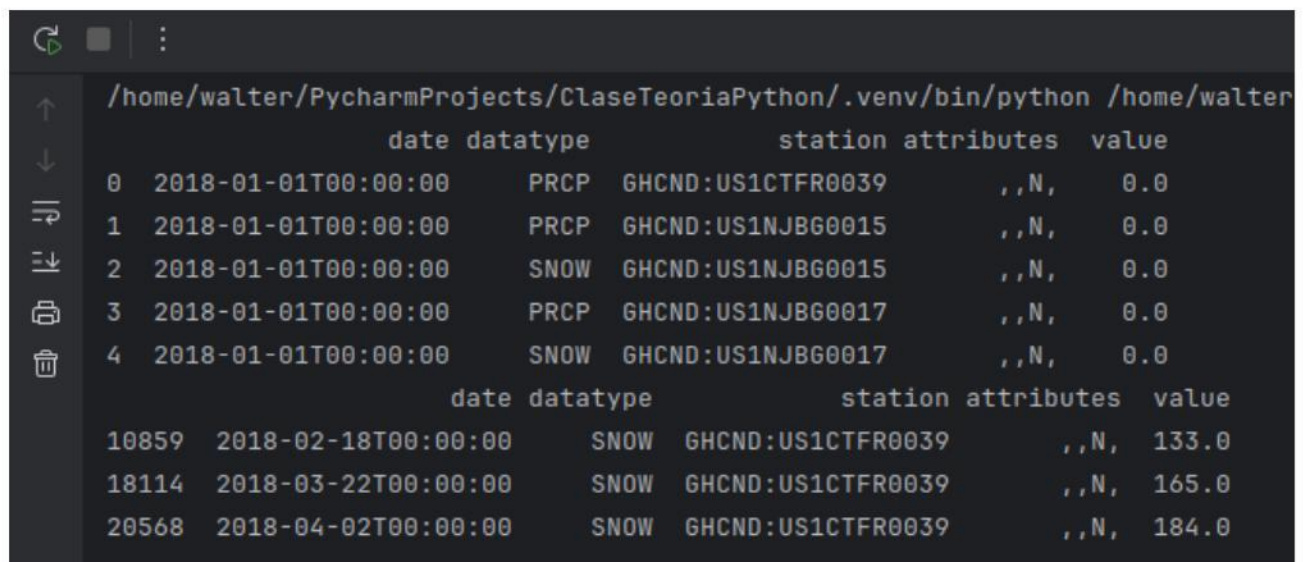
SELECT \* FROM clima WHERE datatype == "PRCP" AND value > 0;

## Método query para consultas a Dataframe

El siguiente ejemplo muestra el uso del método query para realizar consultas a un DataFrame.

```
ejemplo01.py x
1 import numpy as np
2 import pandas as pd
3
4 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/clima_2018.csv')
5 print(clima.head())
6
7 datos = clima.query('datatype == "SNOW" and value > 100 and station.str.contains("US1CTFR0039")')
8 print(datos)
9
```

Salida:



	date	datatype	station	attributes	value
0	2018-01-01T00:00:00	PRCP	GHCND:US1CTFR0039	,,N,	0.0
1	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0015	,,N,	0.0
2	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0015	,,N,	0.0
3	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0017	,,N,	0.0
4	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0017	,,N,	0.0
	date	datatype	station	attributes	value
10859	2018-02-18T00:00:00	SNOW	GHCND:US1CTFR0039	,,N,	133.0
18114	2018-03-22T00:00:00	SNOW	GHCND:US1CTFR0039	,,N,	165.0
20568	2018-04-02T00:00:00	SNOW	GHCND:US1CTFR0039	,,N,	184.0

## Sentencia SQL equivalente

SELECT \* FROM clima WHERE

datatype == "SNOW" AND value > 0 AND station LIKE "%US1CTFR0039%";



## Método describe aplicado a columnas de un Dataframe

El siguiente ejemplo muestra el uso del método describe() aplicado a columnas de un DataFrame.

```
ejemplo01.py x
1 import numpy as np
2 import pandas as pd
3
4 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/clima_2018.csv')
5 print(clima.head())
6
7 print("\n ***** clima.date.describe(): *****")
8 print(clima.date.describe())
9
10 print("\n ***** clima.value.describe(): *****")
11 print(clima.value.describe())
12
```

Salida:

```

/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python /home/walter/
↑
↓
≡
≡
≡
≡
≡

```

	date	datatype	station	attributes	value
0	2018-01-01T00:00:00	PRCP	GHCND:US1CTFR0039	,,N,	0.0
1	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0015	,,N,	0.0
2	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0015	,,N,	0.0
3	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0017	,,N,	0.0
4	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0017	,,N,	0.0

```

***** clima.date.describe(): *****
count          78780
unique           365
top    2018-03-22T00:00:00
freq           296
Name: date, dtype: object

***** clima.value.describe(): *****
count    78780.000000
mean       44.951456
std       192.835503
min       -72.800000
25%        0.000000
50%        1.000000
75%       12.800000
max      2359.000000
Name: value, dtype: float64

```

## Inner Join con Dataframes

El siguiente ejemplo muestra como realizar inner join con DataFrames.

inner-join.py ×

```
1 import pandas as pd
2
3 df1 = pd.DataFrame({
4     'ID': [1, 2, 3],
5     'Nombre': ['Ana', 'Luis', 'Carlos']
6 })
7 print(df1)
8
9
10 df2 = pd.DataFrame({
11     'ID': [2, 3, 4],
12     'Departamento': ['San Salvador', 'San Miguel', 'Sonsonate']
13 })
14 print(df2)
15
16 resultado = pd.merge(df1, df2, on='ID', how='inner')
17 print(resultado)
```

Salida:

```
/home/usuario/PyCharmMiscProject/
ID  Nombre
0   1    Ana
1   2    Luis
2   3   Carlos
ID  Departamento
0   2  San Salvador
1   3   San Miguel
2   4   Sonsonate
ID  Nombre  Departamento
0   2    Luis  San Salvador
1   3   Carlos  San Miguel
```

## Left Join con Dataframes

El siguiente ejemplo muestra como realizar left join con DataFrames.

```
control ▾ Current File ▾ ▶ ⚙️ ⋮

left-join.py ×

1  import pandas as pd
2
3  df1 = pd.DataFrame({
4      'ID': [1, 2, 3],
5      'Nombre': ['Ana', 'Luis', 'Carlos']
6  })
7  print(df1)
8
9
10 df2 = pd.DataFrame({
11     'ID': [2, 3, 4],
12     'Departamento': ['San Salvador', 'San Miguel', 'Sonsonate']
13 })
14 print(df2)
15
16 resultado = pd.merge(df1, df2, on='ID', how='left')
17 print(resultado)
18
```

Salida:

```
↺ ▢ ⋮
↑ /home/usuario/PyCharmMiscProject/.venv/
↓
ID Nombre
0 1 Ana
1 2 Luis
2 3 Carlos
ID Departamento
0 2 San Salvador
1 3 San Miguel
2 4 Sonsonate
ID Nombre Departamento
0 1 Ana NaN
1 2 Luis San Salvador
2 3 Carlos San Miguel
3 4 NaN Sonsonate
```

## Right Join con Dataframes

El siguiente ejemplo muestra como realizar right join con DataFrames.

```
1 import pandas as pd
2
3 df1 = pd.DataFrame({
4     'ID': [1, 2, 3],
5     'Nombre': ['Ana', 'Luis', 'Carlos']
6 })
7 print(df1)
8
9
10 df2 = pd.DataFrame({
11     'ID': [2, 3, 4],
12     'Departamento': ['San Salvador', 'San Miguel', 'Sonsonate']
13 })
14 print(df2)
15
16 resultado = pd.merge(df1, df2, on='ID', how='right')
17 print(resultado)
```

Salida:

```
/home/usuario/PyCharmMiscProject/.venv/bin/
ID  Nombre
0   1     Ana
1   2     Luis
2   3    Carlos
ID  Departamento
0   2  San Salvador
1   3   San Miguel
2   4   Sonsonate
ID  Nombre  Departamento
0   1     Ana           NaN
1   2     Luis  San Salvador
2   3    Carlos   San Miguel
3   4      NaN   Sonsonate
```

## Full Join con Dataframes

El siguiente ejemplo muestra como realizar full join con DataFrames.

```
full-join.py x
1 import pandas as pd
2
3 df1 = pd.DataFrame({
4     'ID': [1, 2, 3],
5     'Nombre': ['Ana', 'Luis', 'Carlos']
6 })
7 print(df1)
8
9
10 df2 = pd.DataFrame({
11     'ID': [2, 3, 4],
12     'Departamento': ['San Salvador', 'San Miguel', 'Sonsonate']
13 })
14 print(df2)
15
16 resultado = pd.merge(df1, df2, on='ID', how='outer')
17 print(resultado)
```

Salida:

```
/home/usuario/PyCharmMiscProject/.venv/
  ID  Nombre
0   1     Ana
1   2     Luis
2   3    Carlos
  ID  Departamento
0   2  San Salvador
1   3   San Miguel
2   4   Sonsonate
  ID  Nombre  Departamento
0   1     Ana           NaN
1   2     Luis  San Salvador
2   3    Carlos   San Miguel
3   4     NaN   Sonsonate
```



## Inner Join con Dataframes (segundo ejemplo)

El siguiente ejemplo muestra como realizar inner join con DataFrames.

```
ejemplo01.py x
1 import pandas as pd
2
3 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/Pandas/clima_2018.csv')
4 print(clima.head())
5 estaciones= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/Pandas/estaciones.csv')
6 print(estaciones)
7
8 print("clima.shape[0]",clima.shape[0])
9 print("estaciones.shape[0]",estaciones.shape[0])
10
11 union = clima.merge(estaciones, left_on='station', right_on='id')
12
13 print(union)
14 union.to_csv('/home/walter/ACD104-WalterSanchez-2024/Pandas/union.csv')
15
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python /home/
date datatype station attributes value
0 2018-01-01T00:00:00 PRCP GHCND:US1CTFR0039 , ,N, 0.0
1 2018-01-01T00:00:00 PRCP GHCND:US1NJB60015 , ,N, 0.0
2 2018-01-01T00:00:00 SNOW GHCND:US1NJB60015 , ,N, 0.0
3 2018-01-01T00:00:00 PRCP GHCND:US1NJB60017 , ,N, 0.0
4 2018-01-01T00:00:00 SNOW GHCND:US1NJB60017 , ,N, 0.0
id ... elevation
0 GHCND:US1CTFR0022 ... 36.6
1 GHCND:US1CTFR0039 ... 6.4
2 GHCND:US1NJB60001 ... 20.1
3 GHCND:US1NJB60002 ... 16.8
4 GHCND:US1NJB60003 ... 21.6
.. ... ..
274 GHCND:USW00054787 ... 24.7
275 GHCND:USW00094728 ... 42.7
276 GHCND:USW00094741 ... 2.7
277 GHCND:USW00094745 ... 111.9
278 GHCND:USW00094789 ... 3.4

[279 rows x 5 columns]
clima.shape[0] 78780
estaciones.shape[0] 279
date datatype ... longitude elevation
0 2018-01-01T00:00:00 PRCP ... -73.568176 6.4
1 2018-01-02T00:00:00 PRCP ... -73.568176 6.4
2 2018-01-03T00:00:00 PRCP ... -73.568176 6.4
3 2018-01-05T00:00:00 DAPR ... -73.568176 6.4
4 2018-01-05T00:00:00 MDPR ... -73.568176 6.4
... ... ..
78775 2018-12-27T00:00:00 PRCP ... -73.765371 24.7
```

## Right y Left Join con Dataframes (segundo ejemplo)

El siguiente ejemplo muestra como realizar left y right join con DataFrames.

```

ejemplo01.py x
1 import pandas as pd
2
3 clima= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/Pandas/clima_2018.csv')
4 print(clima.head())
5 estaciones= pd.read_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/estaciones.csv')
6 print(estaciones)
7
8 union_izquierda = estaciones.merge(clima, left_on='id', right_on='station', how='left')
9 union_derecha = clima.merge(estaciones, left_on='station', right_on='id', how='right')
10
11 print(union_izquierda)
12 union_izquierda.to_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/unionizquierda.csv')
13
14 print(union_derecha)
15 union_derecha.to_csv('/home/walter/ACD104-WalterSanchez-2024/demotracion01/Pandas/unionderecha.csv')
16

```

Salida:

```

Run ejemplo01 x
/...
id name ... attributes value
0 GHCND:US1CTFR0022 STAMFORD 2.6 SSW, CT US ... NaN NaN
1 GHCND:US1CTFR0039 STAMFORD 4.2 S, CT US ... ,N, 0.0
2 GHCND:US1CTFR0039 STAMFORD 4.2 S, CT US ... ,N, 0.0
3 GHCND:US1CTFR0039 STAMFORD 4.2 S, CT US ... ,N, 0.0
4 GHCND:US1CTFR0039 STAMFORD 4.2 S, CT US ... ,N, 2.0
...
78944 GHCND:USW00094789 JFK INTERNATIONAL AIRPORT, NY US ... ,W, 130.0
78945 GHCND:USW00094789 JFK INTERNATIONAL AIRPORT, NY US ... ,W, 9.8
78946 GHCND:USW00094789 JFK INTERNATIONAL AIRPORT, NY US ... ,W, 12.5
78947 GHCND:USW00094789 JFK INTERNATIONAL AIRPORT, NY US ... ,W, 1.0
78948 GHCND:USW00094789 JFK INTERNATIONAL AIRPORT, NY US ... ,W, 1.0

[78949 rows x 10 columns]
date datatype ... longitude elevation
0 NaN NaN ... -73.577000 36.6
1 2018-01-01T00:00:00 PRCP ... -73.568176 6.4
2 2018-01-02T00:00:00 PRCP ... -73.568176 6.4
3 2018-01-03T00:00:00 PRCP ... -73.568176 6.4
4 2018-01-05T00:00:00 DAPR ... -73.568176 6.4
...
78944 2018-12-31T00:00:00 WDF5 ... -73.764010 3.4
78945 2018-12-31T00:00:00 WSF2 ... -73.764010 3.4
78946 2018-12-31T00:00:00 WSF5 ... -73.764010 3.4
78947 2018-12-31T00:00:00 WT01 ... -73.764010 3.4
78948 2018-12-31T00:00:00 WT02 ... -73.764010 3.4

```

## Método groupby con Dataframes

El siguiente ejemplo muestra el uso del método groupby con DataFrames.

```
ejemplo01.py x
1 import pandas as pd
2 datos = {
3     'Habitantes': [950, 900, 990, 1004, 1005, 940, 990, 1040, 750, 1200 ],
4     'Departamento': ['Usulután', 'Usulután', 'Sonsonate', 'Usulután', 'San Salvador', 'Sonsonate',
5                     'Sonsonate', 'Usulután', 'Sonsonate', 'San Salvador']
6 }
7 df = pd.DataFrame(datos)
8
9 print(df)
10 print(df.groupby(["Departamento"]).mean())
11 print(df.groupby(["Departamento"]).last())
12 print(df.groupby(["Departamento"]).first())
13
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/
Habitantes  Departamento
0          950      Usulután
1          900      Usulután
2          990      Sonsonate
3         1004      Usulután
4         1005  San Salvador
5          940      Sonsonate
6          990      Sonsonate
7         1040      Usulután
8          750      Sonsonate
9         1200  San Salvador

Habitantes
Departamento
San Salvador      1102.5
Sonsonate         917.5
Usulután          973.5

Habitantes
Departamento
San Salvador      1200
Sonsonate         750
Usulután         1040

Habitantes
Departamento
San Salvador      1005
Sonsonate         990
Usulután          950
```

## Método groupby con Dataframes

El siguiente ejemplo muestra el uso del método groupby con DataFrames.

```
ejemplo01.py x
1 import pandas as pd
2 datos = {
3     'Habitantes': [950, 900, 990, 1004, 1005, 940, 990, 1040, 750, 1200 ],
4     'Departamento': ['Usulután', 'Usulután', 'Sonsonate', 'Usulután', 'San Salvador', 'Sonsonate',
5                     'Sonsonate', 'Usulután', 'Sonsonate', 'San Salvador']
6 }
7 df = pd.DataFrame(datos)
8
9 print(df)
10 print(df.groupby(["Departamento"]).var())
11 print(df.groupby(["Departamento"]).count())
12 print(df.groupby(["Departamento"]).std())
13
```

Salida:

```
Run ejemplo01 x
/ home/walter/PycharmProjects/ClaseTeoriaPython/
Habitantes  Departamento
0          950      Usulután
1          900      Usulután
2          990    Sonsonate
3         1004      Usulután
4         1005  San Salvador
5          940    Sonsonate
6          990    Sonsonate
7         1040      Usulután
8          750    Sonsonate
9         1200  San Salvador

Habitantes
Departamento
San Salvador    19012.5
Sonsonate       13025.0
Usulután        3769.0

Habitantes
Departamento
San Salvador         2
Sonsonate             4
Usulután             4

Habitantes
Departamento
San Salvador    137.885822
Sonsonate       114.127122
Usulután        61.392182
```



## Método groupby con Dataframes

El siguiente ejemplo muestra el uso del método groupby con DataFrames.

```
ejemplo01.py x
1 import pandas as pd
2 datos = {
3     'Habitantes': [950, 900, 990, 1004, 1005, 940, 990, 1040, 750, 1200 ],
4     'Departamento': ['Usulután', 'Usulután', 'Sonsonate', 'Usulután', 'San Salvador', 'Sonsonate',
5                     'Sonsonate', 'Usulután', 'Sonsonate', 'San Salvador']
6 }
7 df = pd.DataFrame(datos)
8
9 print(df)
10 print(df.groupby(["Departamento"]).quantile())
11 print(df.groupby(["Departamento"]).describe())
12
13
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python /home/walter/PycharmProjects/
Habitantes  Departamento
0          950      Usulután
1          900      Usulután
2          990      Sonsonate
3         1004      Usulután
4         1005  San Salvador
5          940      Sonsonate
6          990      Sonsonate
7         1040      Usulután
8          750      Sonsonate
9         1200  San Salvador
Habitantes
Departamento
San Salvador      1102.5
Sonsonate         965.0
Usulután          977.0
Habitantes
count  mean  std  ...  50%  75%  max
Departamento
San Salvador      2.0  1102.5  137.885822  ...  1102.5  1151.25  1200.0
Sonsonate         4.0   917.5  114.127122  ...   965.0   990.00   990.0
Usulután         4.0   973.5   61.392182  ...   977.0  1013.00  1040.0
[3 rows x 8 columns]
```



## Método aggregate con Dataframes

El siguiente ejemplo muestra el uso del método aggregate con DataFrames.

```

ejemplo01.py x
1  import pandas as pd
2
3  datos = pd.DataFrame( data: [[105.1, 501.3, 304.7],
4                               [405.6, 250.1, 460.3],
5                               [703.8, 860.7, 967.3],
6                               [430.5, 67.9, 150.2]],
7                               columns=['A', 'B', 'C'])
8
9  print(datos)
10
11 print("-----")
12 print(datos.aggregate(['sum']))
13 print(datos.aggregate(['sum']))
14 print("-----")
15 print(datos.aggregate(['std']))
16 print(datos.aggregate(['std']))
17 print("-----")
18 print(datos.agg(['mean']))
19 print(datos.aggregate(['mean']))
20 print("-----")
21 print(datos.aggregate(['var']))
22 print(datos.aggregate(['var']))

```

Salida:

```

Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python
      A      B      C
0  105.1  501.3  304.7
1  405.6  250.1  460.3
2  703.8  860.7  967.3
3  430.5   67.9  150.2
-----
datos.aggregate(['sum'])
      A      B      C
sum  1645.0  1680.0  1882.5
-----
datos.aggregate(['std'])
      A      B      C
std  244.755531  343.349482  354.493041
-----
datos.agg(['mean'])
      A      B      C
mean  411.25  420.0  470.625
-----
datos.aggregate(['var'])
      A      B      C
var  59905.27  117888.866667  125665.315833

```

## Método aggregate con Dataframes

El siguiente ejemplo muestra el uso del método aggregate con DataFrames.

```
ejemplo01.py x
1 import pandas as pd
2
3 datos = pd.DataFrame(data=[[105.1, 501.3, 304.7],
4                             [405.6, 250.1, 460.3],
5                             [703.8, 860.7, 967.3],
6                             [430.5, 67.9, 150.2]],
7                       columns=['A', 'B', 'C'])
8
9 print(datos)
10
11 print("-----")
12 print(datos.agg({'A': ['sum', 'min'], 'B': ['min', 'max']}))
13 print(datos.agg({'A': ['sum', 'min'], 'B': ['min', 'max']}))
14 print("-----")
15 print(datos.agg(x=('A', 'max'), y=('B', 'min'), z=('C', 'mean')))
16 print(datos.agg(a=('A', 'max'), b=('B', 'min'), c=('C', 'mean')))
17 print("-----")
18 print(datos.agg("mean", axis="columns"))
19 print(datos.agg(func="mean", axis="columns"))
```

Salida:

```
Run ejemplo01 x
/home/walter/PycharmProjects/ClaseTeoriaPython/.venv/bin/python
  A      B      C
0  105.1  501.3  304.7
1  405.6  250.1  460.3
2  703.8  860.7  967.3
3  430.5   67.9  150.2
-----
datos.agg({'A': ['sum', 'min'], 'B': ['min', 'max']})
  A      B
sum  1645.0  NaN
min   105.1   67.9
max     NaN  860.7
-----
datos.agg(x=('A', 'max'), y=('B', 'min'), z=('C', 'mean'))
  A      B      C
a  703.8  NaN   NaN
b   NaN  67.9   NaN
c   NaN  NaN  470.625
-----
datos.agg("mean", axis="columns")
0    303.700000
1    372.000000
2    843.933333
3    216.200000
dtype: float64
```