

# NIVEL 4

---

LIBRERÍAS - PANDAS



# PANDAS

---

- ¿Qué es Pandas?
  - ✓ Estructuras de datos de Pandas
  - ✓ Lectura de datos en Pandas
  - ✓ Tipos de datos de un DataFrame
  - ✓ Atributos de un DataFrame
  - ✓ Métodos de un DataFrame
  - ✓ Manipulación de DataFrames
  - ✓ Gráficos para explorar los datos de un DataFrame



# ¿QUÉ ES PANDAS?

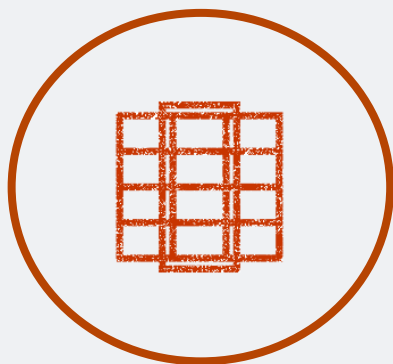
Pandas es una librería especializada para el análisis de datos que cuenta con las estructuras de datos que necesitamos para limpiar los datos en bruto y que sean aptos para el análisis (por ejemplo, tablas)



- ✓ Dado que pandas lleva a cabo tareas importantes, como alinear datos para su comparación, fusionar conjuntos de datos, gestionar datos perdidos, etc., **se ha convertido en una librería muy importante para procesar datos a alto nivel en Python** (es decir, hacer análisis estadístico)
- ✓ Pandas fue diseñada originalmente para gestionar datos financieros, y como alternativo al uso de hojas de cálculo (es decir, Microsoft Excel)

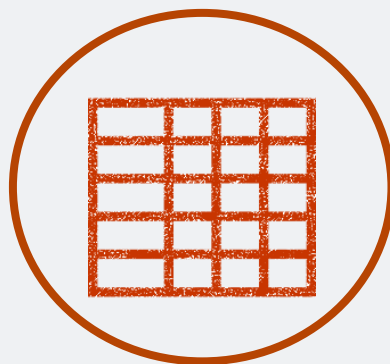
**Pandas nos proporciona las estructuras de datos y funciones necesarias para el análisis de datos**

# ESTRUCTURAS DE DATOS EN PANDAS



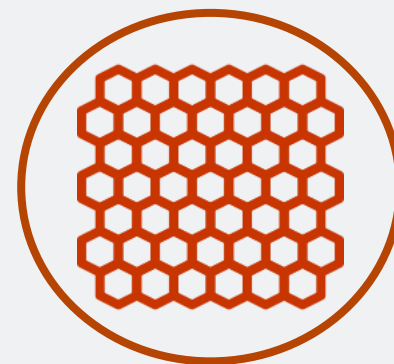
## **Series (1D)**

Similar a la columna de una tabla



## **DataFrame (2D)**

Similar a una tabla con columnas y filas



## **Panel (3D)**

Contenedor 3D.  
No es muy utilizado

# LECTURA DE DATOS EN PANDAS

Importamos la librería de pandas y la vamos a usar con el alias `pd`

Importamos la librería de gráficos de matplotlib y la vamos a usar con el alias `plt`

```
EjemploPeajes.py x
1 # Acá importamos la librería de pandas y la vamos a usar con el alias pd
2 import pandas as pd
3
4 # Acá importamos la librería de gráficos de matplotlib y la vamos a usar con el alias plt
5 import matplotlib.pyplot as plt
6
7
8 # Parte 1: cargar los datos
9
10 # Acá cargamos un DataFrame a partir de un archivo CSV
11 peajes = pd.read_csv('peajes.csv')
```

Cargamos un `DataFrame` a partir de un archivo `CSV`



# PEAJES – EJEMPLO DE DATAFRAME

```
Terminal 2/A x
In [14]: type(peajes)
Out[14]: pandas.core.frame.DataFrame

In [15]: peajes
Out[15]:
```

	NombreProyecto	...	FechaUltimoCambioDeTarifa
0	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
1	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
2	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
3	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
4	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
5	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
6	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
7	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLETA	...	2019-01-16T00:00:00.000
8	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLETA	...	2019-01-16T00:00:00.000
9	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
11	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
12	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
13	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
14	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
15	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
16	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
17	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
18	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
19	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
20	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
21	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
22	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
23	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
24	ÁREA METROPOLITANA DE CÚCUTA Y NORTE DE SANTANDER	...	Sin incremento
25	ÁREA METROPOLITANA DE CÚCUTA Y NORTE DE SANTANDER	...	Sin incremento
26	ÁREA METROPOLITANA DE CÚCUTA Y NORTE DE SANTANDER	...	2019-01-16T00:00:00.000

# TIPOS DE DATOS DE UN DATAFRAME

Tipo en Pandas	Tipo equivalente en Python	Descripción
object	string	Es el tipo más general. Será asignado a la columna si la columna tiene valores de varios tipos (números y strings)
int64	int	Valores numéricos
float64	float	Valores numéricos con decimales. Si una columna contiene números y NaNs, pandas lo pondrá por defecto en float64.
datetime64, timedelta[ns]	N/A	Valores destinados a contener datos de tiempo

# TIPOS DE DATOS DE «PEAJES»

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	NombreProyecto	NombreEstacionPeaje	Departamento	Municipio	TAR_PLENA_I	TAR_PLENA_II	TAR_PLENA_III	TAR_PLENA_IV	TAR_PLENA_V	TAR_PLENA_VI	TAR_PLENA_VII	TAR_PLENA_VIII	FechaUltimoCambioTarifa	
2	ARMENIA-PEREIRA CIRCASIA		QUINDÃO	FILANDIA	13800	17600	17600	17600	42900	52600	58600	0	2019-01-16T00:00:00.000	
3	ARMENIA-PEREIRA COROZAL		VALLE DEL CAUCA	SEVILLA	10400	12600	12600	12600	30700	38400	44600	0	2019-01-16T00:00:00.000	
4	ARMENIA-PEREIRA PAVAS		CALDAS	MANIZALES	10400	12600	12600	12600	30700	38400	44600	0	2019-01-16T00:00:00.000	

```

Terminal 2/A
In [16]: peajes.dtypes
Out[16]:
NombreProyecto           object
NombreEstacionPeaje      object
Departamento            object
Municipio                object
TAR_PLENA_I              int64
TAR_PLENA_II             int64
TAR_PLENA_III            int64
TAR_PLENA_IV             int64
TAR_PLENA_V              int64
TAR_PLENA_VI             int64
TAR_PLENA_VII            int64
TAR_PLENA_VIII           int64
FechaUltimoCambioTarifa  object
dtype: object

```



# ATRIBUTOS DE UN DATAFRAME

Los atributos son las características de un objeto. Los atributos de un `DataFrame` son:

df.atributo	Descripción
<code>dtypes</code>	Lista de los tipos de las columnas
<code>columns</code>	Lista de los nombres de las columnas
<code>axes</code>	Lista de las etiquetas de las filas y de los nombres de las columnas
<code>ndim</code>	Cantidad de dimensiones
<code>size</code>	Cantidad de elementos
<code>shape</code>	Retorna una tupla que representa la dimensionalidad
<code>values</code>	Representación “numpy” de los datos

# ALGUNOS ATRIBUTOS DE «PEAJES»

```

Terminal 2/A x
In [27]: peajes.columns
Out[27]:
Index(['NombreProyecto', 'NombreEstacionPeaje', 'Departamento', 'Municipio',
      'TAR_PLENA_I', 'TAR_PLENA_II', 'TAR_PLENA_III', 'TAR_PLENA_IV',
      'TAR_PLENA_V', 'TAR_PLENA_VI', 'TAR_PLENA_VII', 'TAR_PLENA_VIII',
      'FechaUltimoCambioTarifa'],
      dtype='object')

In [28]: peajes.size
Out[28]: 1300

In [29]: peajes.shape
Out[29]: (100, 13)

In [30]: peajes.values
Out[30]:
array([[ 'ARMENIA-PEREIRA-MANIZALES', 'CIRCASIA', 'QUINDÍO', ..., 58600,
        0, '2019-01-16T00:00:00.000'],
      [ 'ARMENIA-PEREIRA-MANIZALES', 'COROZAL', 'VALLE DEL CAUCA', ...,
        44600, 0, '2019-01-16T00:00:00.000'],
      [ 'ARMENIA-PEREIRA-MANIZALES', 'PAVAS', 'CALDAS', ..., 44600, 0,
        '2019-01-16T00:00:00.000'],
      ...,
      [ 'VILLAVICENCIO-YOPAL', 'PUENTE AMARILLO', 'META', ..., 26900, 0,
        '2019-01-16T00:00:00.000'],
      [ 'VILLAVICENCIO-YOPAL', 'SAN PEDRO', 'CASANARE', ..., 0, 0,
        '2019-01-16T00:00:00.000'],
      [ 'VILLAVICENCIO-YOPAL', 'VERACRUZ', 'META', ..., 29100, 0,
        '2019-01-16T00:00:00.000']], dtype=object)

```

# MÁS SOBRE LOS ATRIBUTOS DE «PEAJES»

Podemos recorrer iterativamente las columnas del `DataFrame`

También se puede hacer referencia a una columna particular para ver sus valores

```
Terminal 2/A x

In [32]: for c in peajes.columns:
...:     print("Columna:", c)
...:
Columna: NombreProyecto
Columna: NombreEstacionPeaje
Columna: Departamento
Columna: Municipio
Columna: TAR_PLENA_I
Columna: TAR_PLENA_II
Columna: TAR_PLENA_III
Columna: TAR_PLENA_IV
Columna: TAR_PLENA_V
Columna: TAR_PLENA_VI
Columna: TAR_PLENA_VII
Columna: TAR_PLENA_VIII
Columna: FechaUltimoCambioDeTarifa

In [33]: for d in peajes.Departamento:
...:     print("Departamento:", d)
...:
Departamento: QUINDÍO
Departamento: VALLE DEL CAUCA
Departamento: CALDAS
Departamento: CALDAS
Departamento: CALDAS
Departamento: CALDAS
Departamento: RISARALDA
Departamento: CUNDINAMARCA
Departamento: CUNDINAMARCA
Departamento: CUNDINAMARCA
Departamento: CUNDINAMARCA
Departamento: CUNDINAMARCA
```

# MÉTODOS DE UN DATAFRAME

- Los métodos son las **funciones** de un objeto. Los métodos de un **DataFrame** son:

df.método()	Descripción
head( [n]), tail( [n])	Primeras/últimas n filas
describe()	Genera estadísticas descriptivas (columnas numéricas )
max(), min()	Retorna los valores max/min de todas las columnas numéricas
mean(), median()	Retorna los valores media/mediana de todas las columnas numéricas
std()	Desviación estándar
sample([n])	Retorna un muestreo aleatorio del DataFrame
dropna()	Descarta todos los registros (filas) con valores faltantes

# EL MÉTODO HEAD DE «PEAJES»

`head()` sirve para calcular un `DataFrame` que tiene sólo las primeras 5 filas (si `head` no tiene parámetros, por defecto devuelve las 5 primeras filas)

```
Terminal 3/A x
In [7]: peajes.head()
Out[7]:
      NombreProyecto      ...      FechaUltimoCambioDeTarifa
0  ARMENIA-PEREIRA-MANIZALES  ...      2019-01-16T00:00:00.000
1  ARMENIA-PEREIRA-MANIZALES  ...      2019-01-16T00:00:00.000
2  ARMENIA-PEREIRA-MANIZALES  ...      2019-01-16T00:00:00.000
3  ARMENIA-PEREIRA-MANIZALES  ...      2019-01-16T00:00:00.000
4  ARMENIA-PEREIRA-MANIZALES  ...      2019-01-16T00:00:00.000

[5 rows x 13 columns]

In [8]: extracto = peajes.head()

In [9]: type(extracto)
Out[9]: pandas.core.frame.DataFrame

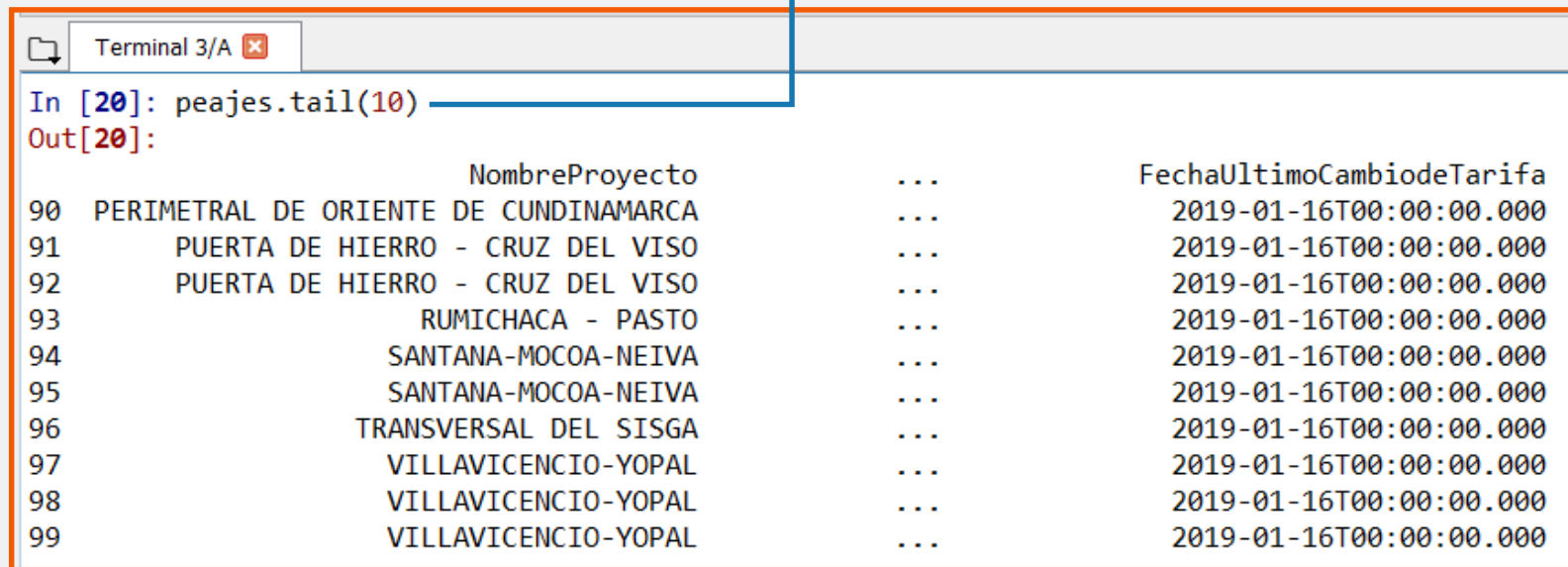
In [10]: len(extracto)
Out[10]: 5
```

Esto nos dice  
que el extracto  
es un `DataFrame`

Esto nos dice que el  
extracto tiene sólo 5 filas

# EL MÉTODO TAIL DE «PEAJES»

`tail( )` sirve para calcular un `DataFrame` que tiene sólo las últimas n filas (si `tail` no tiene parámetros, por defecto devuelve las 5 últimas filas)



The screenshot shows a terminal window titled "Terminal 3/A" with the following content:

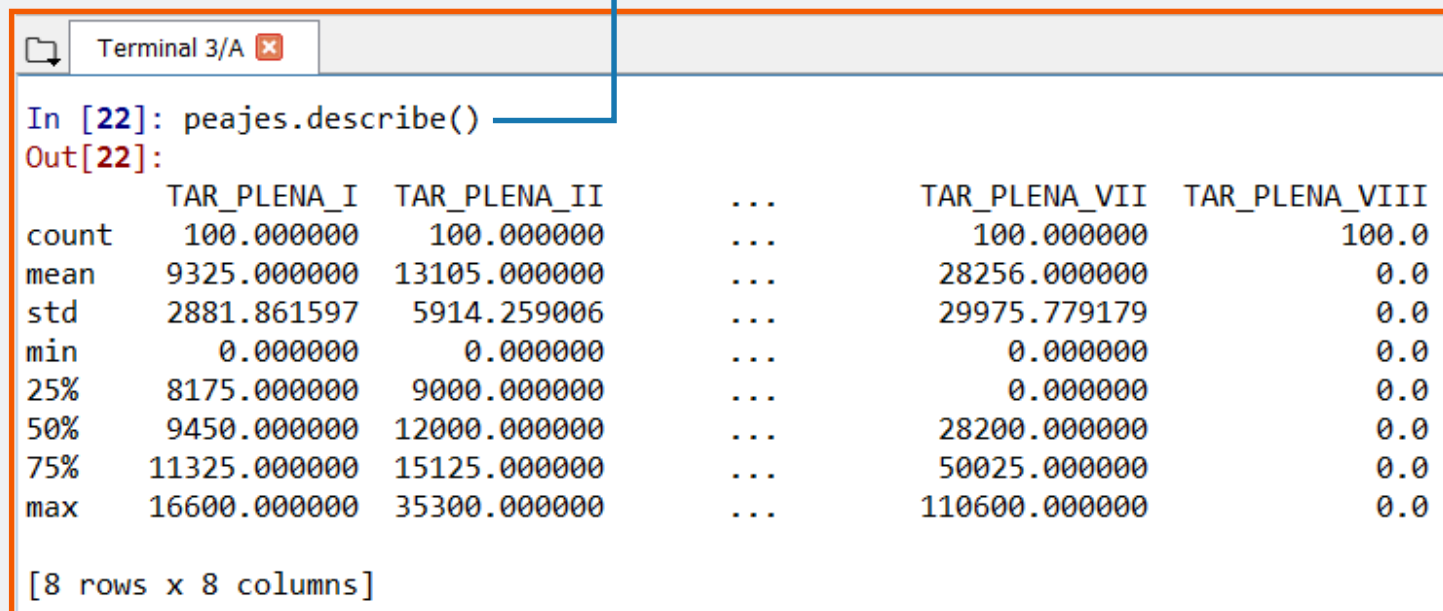
```
In [20]: peajes.tail(10)
```

The output is a DataFrame with 10 rows and 3 columns. The first column contains row indices from 90 to 99. The second column contains project names, and the third column contains timestamps. The output is as follows:

	NombreProyecto	FechaUltimoCambioDeTarifa
90	PERIMETRAL DE ORIENTE DE CUNDINAMARCA	2019-01-16T00:00:00.000
91	PUERTA DE HIERRO - CRUZ DEL VISO	2019-01-16T00:00:00.000
92	PUERTA DE HIERRO - CRUZ DEL VISO	2019-01-16T00:00:00.000
93	RUMICHACA - PASTO	2019-01-16T00:00:00.000
94	SANTANA-MOCHOA-NEIVA	2019-01-16T00:00:00.000
95	SANTANA-MOCHOA-NEIVA	2019-01-16T00:00:00.000
96	TRANSVERSAL DEL SISGA	2019-01-16T00:00:00.000
97	VILLAVICENCIO-YOPAL	2019-01-16T00:00:00.000
98	VILLAVICENCIO-YOPAL	2019-01-16T00:00:00.000
99	VILLAVICENCIO-YOPAL	2019-01-16T00:00:00.000

# EL MÉTODO DESCRIBE DE «PEAJES»

`describe()` calcula estadísticas solo sobre las columnas de tipo numérico. Técnicamente, `describe()` genera otro `DataFrame`, con otras columnas y otras filas completamente diferentes



```
Terminal 3/A [X]

In [22]: peajes.describe()
Out[22]:
```

	TAR_PLENA_I	TAR_PLENA_II	...	TAR_PLENA_VII	TAR_PLENA_VIII
count	100.000000	100.000000	...	100.000000	100.0
mean	9325.000000	13105.000000	...	28256.000000	0.0
std	2881.861597	5914.259006	...	29975.779179	0.0
min	0.000000	0.000000	...	0.000000	0.0
25%	8175.000000	9000.000000	...	0.000000	0.0
50%	9450.000000	12000.000000	...	28200.000000	0.0
75%	11325.000000	15125.000000	...	50025.000000	0.0
max	16600.000000	35300.000000	...	110600.000000	0.0

```
[8 rows x 8 columns]
```

# LOS MÉTODOS MEAN Y SUM DE «PEAJES»

`mean()`: Calcula el promedio por columna (sobre las numéricas)

`sum()`: Calcula el total de cada columna (incluye todos los tipos)

```
Terminal 3/A
In [26]: peajes.mean()
Out[26]:
TAR_PLENA_I      9325.0
TAR_PLENA_II     13105.0
TAR_PLENA_III    17176.0
TAR_PLENA_IV     23106.0
TAR_PLENA_V      33272.0
TAR_PLENA_VI     26661.0
TAR_PLENA_VII    28256.0
TAR_PLENA_VIII      0.0
dtype: float64

In [27]: peajes.sum()
Out[27]:
NombreProyecto      ARMENIA-PEREIRA-MANIZALESARMENIA-PEREIRA-MANIZ...
NombreEstacionPeaje  CIRCASIACOROZALPAVASSAN BERNARDOSANTÁGUEDATARA...
Departamento        QUINDÍOVALLE DEL CAUCACALDASCALDASCALDAS...
Municipio            FILANDIASVILLAMANIZALESMANIZALESMANIZALESCHIN...
TAR_PLENA_I          932500
TAR_PLENA_II         1310500
TAR_PLENA_III        1717600
TAR_PLENA_IV         2310600
TAR_PLENA_V          3327200
TAR_PLENA_VI         2666100
TAR_PLENA_VII        2825600
TAR_PLENA_VIII         0
FechaUltimoCambioDeTarifa  2019-01-16T00:00:00.0002019-01-16T00:00:00.000...
dtype: object
```



# LOS MÉTODOS MIN Y MAX DE «PEAJES»

`min()`: Calcula el mínimo de cada columna (incluye todos los tipos)

`max()`: Calcula el máximo de cada columna (incluye todos los tipos)

```
Terminal 3/A x
In [28]: peajes.min()
Out[28]:
NombreProyecto          ARMENIA-PEREIRA-MANIZALES
NombreEstacionPeaje      ACAPULCO
Departamento            ANTIOQUIA
Municipio                ACACÍAS
TAR_PLENA_I              0
TAR_PLENA_II             0
TAR_PLENA_III            0
TAR_PLENA_IV             0
TAR_PLENA_V              0
TAR_PLENA_VI             0
TAR_PLENA_VII            0
TAR_PLENA_VIII           0
FechaUltimoCambioTarifa  2018-11-16T00:00:00.000
dtype: object

In [29]: peajes.max()
Out[29]:
NombreProyecto          ÁREA METROPOLITANA DE CÚCUTA Y NORTE DE SANTANDER
NombreEstacionPeaje      YUCAO
Departamento            VALLE DEL CAUCA
Municipio                VILLAVICENCIO
TAR_PLENA_I              16600
TAR_PLENA_II             35300
TAR_PLENA_III            39900
TAR_PLENA_IV             53100
TAR_PLENA_V              74800
TAR_PLENA_VI             99700
TAR_PLENA_VII            110600
TAR_PLENA_VIII           0
FechaUltimoCambioTarifa  Sin incremento
dtype: object
```

# MANIPULACIÓN DE DATAFRAMES

## Slicing (selección de un subconjunto)

- Hay diferentes formas de extraer un subconjunto de datos de un DataFrame:
  - ✓ Una o más columnas
  - ✓ Una o más filas
  - ✓ Un subconjunto de filas y columnas
- Filas y columnas pueden ser seleccionadas por su posición o etiqueta



# SELECCIÓN DE UNA COLUMNA

```
Terminal 4/A x
In [2]: peajes.Departamento
Out[2]:
0          QUINDÍO
1    VALLE DEL CAUCA
2          CALDAS
3          CALDAS
4          CALDAS
5          CALDAS
6    RISARALDA
7    CUNDINAMARCA
8    CUNDINAMARCA
9    CUNDINAMARCA
10   CUNDINAMARCA
11   CUNDINAMARCA
12         META
13    BOLÍVAR
14   ATLÁNTICO
15   ATLÁNTICO
16   ANTIOQUIA
17   ANTIOQUIA
18   CUNDINAMARCA
19   CUNDINAMARCA
```

## Opción 1:

Utilizando el nombre de la columna como si fuera un atributo

## Opción 2:

Poniendo el nombre de la columna entre corchetes cuadrados y entre comillas

```
Terminal 4/A x
In [4]: peajes[["Departamento"]]
Out[4]:
      Departamento
0          QUINDÍO
1    VALLE DEL CAUCA
2          CALDAS
3          CALDAS
4          CALDAS
5          CALDAS
6    RISARALDA
7    CUNDINAMARCA
8    CUNDINAMARCA
9    CUNDINAMARCA
10   CUNDINAMARCA
11   CUNDINAMARCA
12         META
13    BOLÍVAR
14   ATLÁNTICO
15   ATLÁNTICO
16   ANTIOQUIA
17   ANTIOQUIA
18   CUNDINAMARCA
19   CUNDINAMARCA
```

# SELECCIÓN DE FILAS

**Opción 1:** Especificando el rango, utilizando ":". No es muy flexible. Por ejemplo, para extraer sólo la primera fila hay que escribir `peajes[0:1]`

In [6]: `peajes[10:20]`

Out[6]:

	NombreProyecto	...	FechaUltimoCambiodeTarifa
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
11	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
12	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
13	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
14	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
15	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
16	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
17	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
18	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
19	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000

[10 rows x 13 columns]

# SELECCIÓN DE FILAS

Opción 2: Usando el método `iloc`, el cual da más flexibilidad y nos permite trabajar con las posiciones de filas y columnas

Extrae la primera fila del `DataFrame`

```
Terminal 4/A ✕  
In [8]: peajes.iloc[0]  
Out[8]:  
NombreProyecto      ARMENIA-PEREIRA-MANIZALES  
NombreEstacionPeaje      CIRCASIA  
Departamento      QUINDÍO  
Municipio      FILANDIA  
TAR_PLENA_I      13800  
TAR_PLENA_II      17600  
TAR_PLENA_III      17600  
TAR_PLENA_IV      17600  
TAR_PLENA_V      42900  
TAR_PLENA_VI      52600  
TAR_PLENA_VII      58600  
TAR_PLENA_VIII      0  
FechaUltimoCambioDeTarifa      2019-01-16T00:00:00.000  
Name: 0, dtype: object
```

# SELECCIÓN DE FILAS CON EL MÉTODO ILOC

Extrae la última  
fila del  
DataFrame

```
Terminal 4/A x
In [17]: peajes.iloc[-1]
Out[17]:
NombreProyecto          VILLAVICENCIO-YOPAL
NombreEstacionPeaje      VERACRUZ
Departamento            META
Municipio                CUMARAL
TAR_PLENA_I              6700
TAR_PLENA_II             13500
TAR_PLENA_III            8700
TAR_PLENA_IV             13500
TAR_PLENA_V              19400
TAR_PLENA_VI             26000
TAR_PLENA_VII            29100
TAR_PLENA_VIII           0
FechaUltimoCambioDeTarifa 2019-01-16T00:00:00.000
Name: 99, dtype: object
```

# SELECCIÓN DE FILAS CON EL MÉTODO ILOC

```
Terminal 4/A
In [11]: peajes.iloc[:]
Out[11]:
```

	NombreProyecto	...	FechaUltimoCambioDeTarifa
0	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
1	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
2	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
3	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
4	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
5	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
6	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
7	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLET	...	2019-01-16T00:00:00.000
8	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLET	...	2019-01-16T00:00:00.000
9	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
11	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
12	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
13	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
14	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
15	CARTAGENA-BARRANQUILLA	...	2019-01-16T00:00:00.000
16	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
17	DESARROLLO VIAL DEL ORIENTE DE MEDELLÍN	...	2019-01-01T00:00:00.000
18	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
19	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
20	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
21	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000

Extrae todas las filas del DataFrame

# SELECCIÓN DE FILAS CON EL MÉTODO ILOC

```
Terminal 1/A x

In [47]: extracto = peajes.iloc[0:10:3]

In [48]: len(extracto)
Out[48]: 4

In [49]: extracto
Out[49]:
```

	NombreProyecto	...	FechaUltimoCambioDeTarifa
0	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
3	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
6	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
9	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000

```
[4 rows x 13 columns]
```

- ✓ Extrae las primeras 10 filas del **DataFrame**, avanzando de 3 en 3 (saca la 0, la 3, la 6 y la 9)
- ✓ Retorna un extrato (**DataFrame**) y el **DataFrame** completo



# ILOC TAMBIÉN PUEDE USARSE PARA FILTRAR LAS COLUMNAS DEL DATAFRAME

Extrae las primeras 5 filas y sólo la columna 3 (Municipio)

Extrae las primeras 5 filas y las columnas de la 2 a la 4 (la 5 queda excluida)

```
Terminal 4/A x
In [23]: peajes.iloc[0:5, 3]
Out[23]:
0    FILANDIA
1    SEVILLA
2    MANIZALES
3    MANIZALES
4    MANIZALES
Name: Municipio, dtype: object

In [24]: peajes.iloc[0:5, 2:5]
Out[24]:
      Departamento  Municipio  TAR_PLENA_I
0      QUINDÍO    FILANDIA      13800
1  VALLE DEL CAUCA    SEVILLA      10400
2        CALDAS  MANIZALES      10400
3        CALDAS  MANIZALES      10400
4        CALDAS  MANIZALES      10400
```

# ILOC TAMBIÉN PUEDE USARSE PARA FILTRAR LAS COLUMNAS DEL DATAFRAME

Extrae las primeras 5 filas y las columnas 2, 1, 3 y 5 que le pasamos en una lista de columnas (no necesariamente en el mismo orden)

Extrae todas las filas y sólo las columnas 4 a la 11 que son las columnas numéricas de este data frame

Terminal 4/A

```
In [26]: peajes.iloc[0:5, [2,1,3,5]]
Out[26]:
```

	Departamento	NombreEstacionPeaje	Municipio	TAR_PLENA_II
0	QUINDÍO	CIRCASIA	FILANDIA	17600
1	VALLE DEL CAUCA	COROZAL	SEVILLA	12600
2	CALDAS	PAVAS	MANIZALES	12600
3	CALDAS	SAN BERNARDO	MANIZALES	12600
4	CALDAS	SANTÁGUEDA	MANIZALES	12600

```
In [27]: peajes.iloc[:, 4:12]
Out[27]:
```

	TAR_PLENA_I	TAR_PLENA_II	...	TAR_PLENA_VII	TAR_PLENA_VIII
0	13800	17600	...	58600	0
1	10400	12600	...	44600	0
2	10400	12600	...	44600	0
3	10400	12600	...	44600	0
4	10400	12600	...	44600	0
5	11500	15200	...	56300	0
6	11500	15200	...	56300	0
7	9300	12100	...	0	0
8	10000	14300	...	41400	0
9	11900	35300	...	70500	0
10	11900	35300	...	70500	0
11	10200	26600	...	58800	0
12	16600	33000	...	85100	0

# MANIPULACIÓN DE DATAFRAMES

## Sorting (ordenamiento de datos)

- Podemos ordenar los datos de un **DataFrame** por columna
- Por defecto, el ordenamiento ocurrirá en orden ascendente y un nuevo **DataFrame** es retornado
- También es posible ordenar usando dos o más columnas



# ORDENAMIENTO DE «PEAJES»



Extraemos un subconjunto de los datos

```
Terminal 4/A x
In [37]: extracto = peajes[0:100:10]
In [38]: extracto
Out[38]:
```

	NombreProyecto	...	FechaUltimoCambiodeTarifa
0	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
20	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000
30	CÓRDOBA - SUCRE	...	2019-01-16T00:00:00.000
40	ruta CARIBE	...	2019-01-10T00:00:00.000
50	AUTOPISTA AL MAR 2	...	2019-01-16T00:00:00.000
60	CONCESIÓN AUTOPISTA AL MAR 1	...	2019-01-16T00:00:00.000
70	IP - ANTIOQUIA BOLIVAR	...	2019-01-16T00:00:00.000
80	IP - MALLA VIAL DEL META	...	2019-01-16T00:00:00.000
90	PERIMETRAL DE ORIENTE DE CUNDINAMARCA	...	2019-01-16T00:00:00.000

[10 rows x 13 columns]

# ORDENAMIENTO DE «PEAJES»



Ordenamos por el nombre del proyecto

Terminal 4/A

```
In [39]: ordenados = extracto.sort_values(by="NombreProyecto")
```

```
In [40]: ordenados
```

```
Out[40]:
```

	NombreProyecto	...	FechaUltimoCambiodeTarifa
0	ARMENIA-PEREIRA-MANIZALES	...	2019-01-16T00:00:00.000
50	AUTOPISTA AL MAR 2	...	2019-01-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
60	CONCESIÓN AUTOPISTA AL MAR 1	...	2019-01-16T00:00:00.000
30	CÓRDOBA - SUCRE	...	2019-01-16T00:00:00.000
70	IP - ANTIOQUIA BOLIVAR	...	2019-01-16T00:00:00.000
80	IP - MALLA VIAL DEL META	...	2019-01-16T00:00:00.000
90	PERIMETRAL DE ORIENTE DE CUNDINAMARCA	...	2019-01-16T00:00:00.000
40	ruta CARIBE	...	2019-01-10T00:00:00.000
20	SANTA MARTA-RIOHACHA-PARAGUACHÓN	...	2019-01-16T00:00:00.000

```
[10 rows x 13 columns]
```

# OTRO EJEMPLO DE ORDENAMIENTO DE «PEAJES»

Ordenamos todos los peajes por la tarifa `TAR_PLENA_I`

```
Terminal 4/A ✖  
In [41]: ordenados = peajes.sort_values(by="TAR_PLENA_I", ascending=False)  
In [42]: ordenados[0:5]  
...:  
Out[42]:  
      NombreProyecto      ...      FechaUltimoCambiodeTarifa  
12      BOGOTÁ - VILLAVICENCIO      ...      2018-11-16T00:00:00.000  
60  CONCESIÓN AUTOPISTA AL MAR 1      ...      2019-01-16T00:00:00.000  
61  CONCESIÓN AUTOPISTA AL MAR 1      ...      2019-01-16T00:00:00.000  
0      ARMENIA-PEREIRA-MANIZALES      ...      2019-01-16T00:00:00.000  
43      RUTA DEL SOL 1      ...      2019-01-20T00:00:00.000  
  
[5 rows x 13 columns]
```

Nos muestra los 5 peajes más costosos

# MANIPULACIÓN DE DATAFRAMES

## Filtering (filtrado de datos)

- Podemos extraer un subconjunto de datos, dadas unas condiciones booleanas (esto se llama indexamiento booleano). A esta extracción se le conoce como un filtro
- Es posible aplicar varios filtros sucesivamente



# FILTRADO DE «PEAJES»

Filtra las filas y se queda sólo con aquellas en que la expresión es verdadera (cuando el Departamento es CUNDINAMARCA)

```
Terminal 4/A
In [44]: peajes[peajes.Departamento == 'CUNDINAMARCA']
Out[44]:
```

	NombreProyecto	...	FechaUltimoCambioDeTarifa
7	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLET	...	2019-01-16T00:00:00.000
8	BOGOTÁ - SIBERIA - LA PUNTA - EL VINO - VILLET	...	2019-01-16T00:00:00.000
9	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
11	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
18	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
19	FONTIBÓN - FACATATIVÁ - LOS ALPES	...	2019-01-06T00:00:00.000
27	BRICEÑO - TUNJA - SOGAMOSO	...	2019-01-15T00:00:00.000
28	BRICEÑO - TUNJA - SOGAMOSO	...	2019-01-15T00:00:00.000
43	ruta del sol 1	...	2019-01-20T00:00:00.000
62	GIRARDOT - HONDA - PUERTO SALGAR	...	2019-01-16T00:00:00.000
63	GIRARDOT - HONDA - PUERTO SALGAR	...	2019-01-21T00:00:00.000
64	IP - ACCESOS NORTE	...	2019-01-16T00:00:00.000
65	IP - ACCESOS NORTE	...	2019-01-16T00:00:00.000
66	IP - ACCESOS NORTE	...	2019-01-16T00:00:00.000
85	IP - TERCER CARRIL	...	2019-01-16T00:00:00.000
86	IP - TERCER CARRIL	...	2019-01-16T00:00:00.000
89	PERIMETRAL DE ORIENTE DE CUNDINAMARCA	...	2019-01-16T00:00:00.000
90	PERIMETRAL DE ORIENTE DE CUNDINAMARCA	...	2019-01-16T00:00:00.000
96	TRANSVERSAL DEL SISGA	...	2019-01-16T00:00:00.000

[20 rows x 13 columns]



# FILTRADO DE «PEAJES»

Esta expresión nos va a mostrar sólo los peajes que valen **más de 10000** para vehículos **Y** están en **Cundinamarca**

```
Terminal 4/A [x]

In [47]: peajes[peajes.TAR_PLENA_I > 10000][peajes.Departamento == 'CUNDINAMARCA']
__main__:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
Out[47]:
```

	NombreProyecto	...	FechaUltimoCambioDeTarifa
9	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
10	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
11	BOGOTÁ - VILLAVICENCIO	...	2018-11-16T00:00:00.000
43	ruta del sol 1	...	2019-01-20T00:00:00.000
85	IP - TERCER CARRIL	...	2019-01-16T00:00:00.000
86	IP - TERCER CARRIL	...	2019-01-16T00:00:00.000

```
[6 rows x 13 columns]
```