Pandas – Ejercicios

Camacho Vázquez Vanessa Alejandra.

Biblioteca de Python para el análisis y manipulación de datos. Si quieres más información al respecto, revisa https://pandas.pydata.org/

In [1]:
```
import pandas as pd
```

¿Cómo importar datos?

Para leer los datos, la estructura básica es `pd.read_tipo-archivo`.

> Si estamos en Google Colab y queremos utilizar algunos datos que están en nuestro Google Drive, podemos utilizar el siguiente código para montar la carpeta raíz de Google Drive:
>
> ```
> from google.colab import drive
> drive.mount('/content/gdrive')
> ```

Para que el siguiente bloque de código funcione, tiene que estar instalado el paquete o módulo `openpyxl` (usando un bloque de código, por ejemplo en este cuaderno, al ejecutar: `!conda install openpyxl`, se instala el paquete)

In [2]:
```
ruta_carpeta = "../Datos/"
counties = pd.read_excel(ruta_carpeta + "counties.xlsx")
print(type(counties))
```

```
<class 'pandas.core.frame.DataFrame'>
```

Si quiero hacerme una idea de las variables y los datos que tengo en la variables `counties`:

In [3]:
```
display(counties)
```

	codestate	codecounty	county	population	area
0	1	1001	Auta#%&()uga	54571.0	594.436000
1	1	1003	Baldwin#%&() ?	182265.0	1589.784000
2	1	1005	Barbour	27457.0	884.876000
3	1	1007	Bi#%&()bb	22915.0	622.582000
4	1	1009	Blount ?	57322.0	644.776000

...

En lo anterior veíamos todas las variables con las 5 primeras y 5 últimas filas. Pero, podemos ver una cantidad determinada de registros (filas), tanto del inicio de la tabla como del final, utilizando `.head()` y `.tail()` respectivamente.

In [4]:
```
counties.head(3) # 5 es el valor predeterminado
```

Out[4]:

	codestate	codecounty	county	population	area
0	1	1001	Auta#%&()uga	54571.0	594.436
1	1	1003	Baldwin#%&() ?	182265.0	1589.784
2	1	1005	Barbour	27457.0	884.876

In [5]:
```
counties.tail(4) # 5 es el valor predeterminado
```

Out[5]:

	codestate	codecounty	county	population	area
3230	72	72153	Ya_uco	42043.0	68.192000
3231	78	78010	; St. Croix ?	50601.0	83.345868
3232	78	78020	St. John	4170.0	19.689867
3233	78	78030	St. Thomas	51634.0	31.313503

Si además de ver la tabla, queremos más información, por ejemplo sus dimensiones y tipo de variables, podemos utilizar lo siguiente:

In [6]:
```
counties.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3234 entries, 0 to 3233
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   codestate   3234 non-null   int64
 1   codecounty  3234 non-null   int64
 2   county      3234 non-null   object
 3   population  3232 non-null   float64
 4   area        3234 non-null   float64
dtypes: float64(2), int64(2), object(1)
memory usage: 126.5+ KB
```

In [7]:
```
print("El tamaño de la tabla de datos es:", counties.shape)
```

```
El tamaño de la tabla de datos es: (3234, 5)
```

In [8]:
```
print(counties.dtypes) #tipo de objeto en cada columna
```

```
codestate       int64
codecounty      int64
county         object
population    float64
area          float64
dtype: object
```

In [9]:
```
counties.describe()
```

Out[9]:

	codestate	codecounty	population	area
count	3234.000000	3234.000000	3.232000e+03	3234.000000
mean	31.441868	31544.737786	9.679656e+04	1093.361817
std	16.411236	16425.545223	3.088044e+05	3564.706999
min	1.000000	1001.000000	1.700000e+01	0.031696
25%	19.000000	19039.500000	1.129700e+04	416.360000
50%	30.000000	30038.000000	2.607550e+04	602.977500
75%	46.000000	46128.500000	6.566050e+04	913.884500
max	78.000000	78030.000000	9.818605e+06	145504.789000

In [10]:
```
counties.describe(include="all")
```

Out[10]:

	codestate	codecounty	county	population	area
count	3234.000000	3234.000000	3234	3.232000e+03	3234.000000
unique	NaN	NaN	2625	NaN	NaN
top	NaN	NaN	Lincoln	NaN	NaN
freq	NaN	NaN	14	NaN	NaN
mean	31.441868	31544.737786	NaN	9.679656e+04	1093.361817

Seleccionar columnas

Para ver el nombre de las columnas que existen, podemos utilizar `.columns`.

In [11]:
```
counties.columns
```

Out[11]:
```
Index(['codestate', 'codecounty', 'county', 'population', 'area'], dtype='object')
```

Escoger una columna específica o un conjunto de ellas:

In [12]:
```
condado = counties["county"]
print(type(condado))
condado
```

```
<class 'pandas.core.series.Series'>
```

Out[12]:
```
0               Auta#%&()uga
1        Baldwin#%&()  ?
2                   Barbour
3                 Bi#%&()bb
4              Blount  ?
               ...
3229                Yabucoa
3230                 Ya_uco
3231       ; St. Croix ?
3232                St. John
3233              St. Thomas
Name: county, Length: 3234, dtype: object
```

In [13]:
```
codi_condado = counties[["codecounty","county"]]
print(type(codi_condado))
codi_condado
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[13]:

	codecounty	county
0	1001	Auta#%&()uga
1	1003	Baldwin#%&() ?
2	1005	Barbour

Escoger filas

Podemos escoger ciertas filas a través de su posición, utilizando `.iloc`

In [14]:
```
seleccion = counties.iloc[0:1,]
print(type(seleccion))
seleccion
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[14]:

	codestate	codecounty	county	population	area
0	1	1001	Auta#%&()uga	54571.0	594.436

De hecho, podemos seleccionar simultaneamente filas y columnas:

In [15]:
```
counties.iloc[18:22,1:3]
```

Out[15]:

	codecounty	county
18	1037	Coo#%&()sa
19	1039	: Co#%&()vington
20	1041	Crenshaw
21	1043	Cullma_n

O también se puede seleccionar filas a partir de una condición dada.

In [16]:
```
idx_bool = counties["codestate"] > 72
print(type(idx_bool))
idx_bool
```

```
<class 'pandas.core.series.Series'>
```

Out[16]:
```
0       False
1       False
2       False
3       False
4       False
        ...
3229    False
3230    False
```

In [17]:
```
codsat72 = counties[counties["codestate"] > 72]
print(type(codsat72))
codsat72
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[17]:

	codestate	codecounty	county	population	area
3231	78	78010	; St. Croix ?	50601.0	83.345868
3232	78	78020	St. John	4170.0	19.689867
3233	78	78030	St. Thomas	51634.0	31.313503

Para seleccionar las filas que no tengan valores faltantes en determinada columna:

In [18]:
```
counties_sin_na = counties[counties["population"].notna()]
print(counties_sin_na.shape)
print(counties.shape)
```

```
(3232, 5)
(3234, 5)
```

In [19]:
```
counties[counties["population"].isna()]
```

Out[19]:

	codestate	codecounty	county	population	area
3145	60	60030	Rose Island	NaN	0.031696
3149	69	69085	Northern Islands	NaN	61.792916

Filas con valores especificos

Existen varias maneras de realizar estas búsquedas.

In [20]:
```
codcou1 = counties[counties["codecounty"].isin([78010, 72151])]
codcou1
```

Out[20]:

	codestate	codecounty	county	population	area
3229	72	72151	Yabucoa	37941.0	55.215000
3231	78	78010	; St. Croix ?	50601.0	83.345868

In [21]:
```
codcou2_bool = counties.loc[:, 'codecounty'] == 78010
print(codcou2_bool)
codcou2 = counties.loc[codcou2_bool]
codcou2
```

```
0       False
1       False
2       False
3       False
4       False
        ...
3229    False
3230    False
3231     True
3232    False
3233    False
Name: codecounty, Length: 3234, dtype: bool
```

Out[21]:

	codestate	codecounty	county	population	area
3231	78	78010	; St. Croix ?	50601.0	83.345868

In [22]:
```
counties[(counties["codecounty"] == 78010) | (counties["codecounty"] == 72151)]
```

Out[22]:

	codestate	codecounty	county	population	area
3229	72	72151	Yabucoa	37941.0	55.215000
3231	78	78010	; St. Croix ?	50601.0	83.345868

Filas con más de una característica específica.

In [23]:
```
counties[(counties["codestate"] == 72) & (counties["area"] >= 80)]
```

Out[23]:

	codestate	codecounty	county	population	area
3159	72	72013	Arec#%&()ibo	96440.0	125.947
3210	72	72113	Pon#%&()ce	166327.0	114.762

Y 3224

Eliminar filas o columnas en donde haya datos faltantes (na)

In [24]:
```
counties.dropna(subset=["population"], axis=0, inplace=False) # inplace: modifique directamente la BD
```

Out[24]:

	codestate	codecounty	county	population	area
0	1	1001	Auta#%&()uga	54571.0	594.436000
1	1	1003	Baldwin#%&() ?	182265.0	1589.784000
2	1	1005	Barbour	27457.0	884.876000
3	1	1007	Bi#%&()bb	22915.0	622.582000
4	1	1009	Blount ?	57322.0	644.776000
...	...	...	...	...	...
3229	72	72151	Yabucoa	37941.0	55.215000
3230	72	72153	Ya_uco	42043.0	68.192000
3231	78	78010	; St. Croix ?	50601.0	83.345868
3232	78	78020	St. John	4170.0	19.689867
3233	78	78030	St. Thomas	51634.0	31.313503

3232 rows × 5 columns

Manipulación datos textuales (Limpieza)

Para realizarle limpieza a columna **county** de la tabla de datos **counties**, podemos utilizar lo siguiente:

- **lower** Poner en minúscula todo el texto.
- **replace** Remplazar ciertos valores por otros.
- **strip** Eliminar los espacios al principio y al final de la cadena.
- **title** Poner primera letra de cada palabra en mayúscula.

In [25]:
```
counties["county"] = (counties["county"]
                      .str.lower()
                      .str.replace("[^a-zàéíóúüñ ]", "", regex=True)
                      .str.replace(" +", " ", regex=True)
                      .str.strip()
                      .str.title())
```

In [26]:
```
print(counties)
```

```
      codestate  codecounty     county  population         area
0             1        1001    Autauga     54571.0   594.436000
1             1        1003    Baldwin    182265.0  1589.784000
2             1        1005    Barbour     27457.0   884.876000
3             1        1007       Bibb     22915.0   622.582000
4             1        1009     Blount     57322.0   644.776000
...         ...         ...        ...         ...          ...
3229         72       72151    Yabucoa     37941.0    55.215000
3230         72       72153      Yauco     42043.0    68.192000
3231         78       78010   St Croix     50601.0    83.345868
3232         78       78020    St John      4170.0    19.689867
3233         78       78030  St Thomas     51634.0    31.313503

[3234 rows x 5 columns]
```

Pueden consultar acerca de expresiones regulares en el cuaderno de nuestro diplomado: https://github.com/AprendizajeProfundo/Diplomado/blob/master/Talleres/Cuadernos/Taller_Regex.ipynb (y pueden complementar con: https://www.regular-expressions.info/quickstart.html).

Creación de columnas a partir de otras

In [ ]:
```
counties["densidad"] = counties["population"] / counties["area"]
counties[["population","area","densidad"]]
```

Renombrar columnas

In [ ]:
```
print(counties.columns)
```

In [ ]:
```
counties.rename(columns = {"densidad":"densidad_pob"}, inplace=True)
print(counties.columns)
```

Modificar tipo de dato de una columna

In [ ]:
```
print(counties.info())
```

In [ ]:
```
counties["codestate"] = counties["codestate"].astype(float)
print(counties.info())
```

Eliminar filas y columnas

Eliminación filas, puede ser por posición o que cumpla una característica.

In [ ]:
```
print(counties.shape)
index = counties.iloc[0:3,].index
print(counties.iloc[0:3,].index)
counties = counties.drop(counties.iloc[0:3,].index)
print(counties.shape)
```

In [ ]:
```
print(counties.shape)
counties = counties.drop(counties[counties['codecounty']==70].index)
print(counties.shape)
```

In [ ]:
```
counties = counties.drop(columns=['densidad_pob'])
# counties  = counties.drop(['densidad_pob'], axis=1)
print(counties.shape)
```

Modificar tablas

Es posible hacer cambios de orden y estructura en las tablas de pandas.

In [ ]:
```
elections = pd.read_excel(ruta_carpeta + "elections.xlsx")
elections.head(5)
```

organicemos los datos descendentemente dependiendo de la cantidad de votos democratas

In [ ]:
```
elections.sort_values(["democrat"], ascending=False)
```

usando `groupby()` podemos juntar los datos por año y sumarlos para tener el total de votos por año

In [ ]:
```
elections.groupby('year')[['democrat', 'republic']].sum()
```

Podemos agrupar, sumar y luego ordenar en un mismo código

In [ ]:
```
elections_sort = (elections.groupby('year')[['democrat', 'republic']]
                  .sum()
                  .sort_values('democrat', ascending=False))
display(elections_sort)
```

Para reestructurar la tabla de datos usamos la función `pivot()`.

In [ ]:
```
pivot_elections = elections.pivot(index='codecounty',
                                  columns='year',
                                  values=['democrat','republic'])
pivot_elections.head()
```

múltiples índices

Se pueden elegir múltiples variables como índice del dataframe. Esto es util para facilitar la extracción de información en ciertos casos

In [ ]:
```
counties_multi = pd.read_excel(ruta_carpeta + "counties.xlsx", index_col = [0, 1])
display(counties_multi)
```

Podemos obtener la suma de población de cada estado

In [ ]:
```
counties_multi['population'].groupby(level='codestate').sum().head(5)
```

Concatenar y unir

Es posible unir varias tablas tanto vertical como horizontalmente.

En ambos casos, podemos usar `concat()`, y se juntarán las bases con bases a los nombres de columnas o los indices de las filas

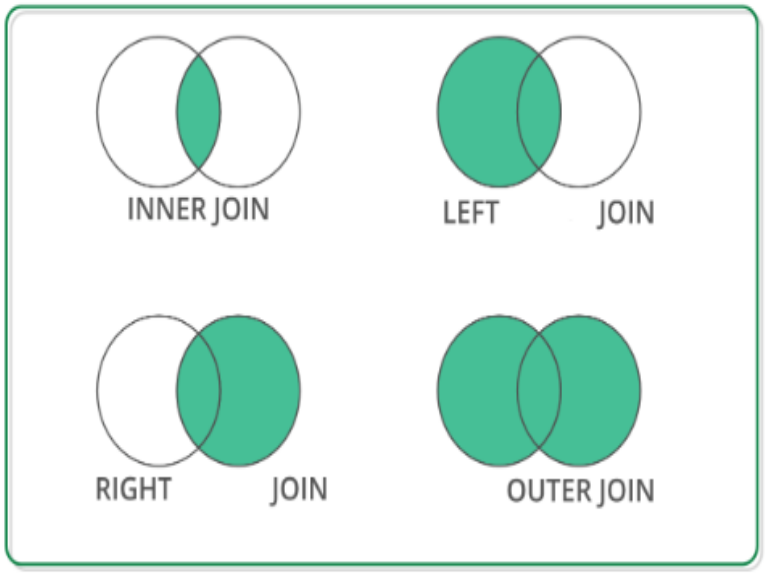
In [ ]:
```
elections_2000 = elections[elections['year']==2000]
display(elections_2000)
```

In [ ]:
```
elections_2004 = elections[elections['year'] == 2004]
elections_2004.loc[:, 'otra'] = 0
display(elections_2004)
```

Si se hace la concatenación sin más, se tomarán todas las columnas y se agregarán NaN

In [ ]:
```
elections_00_04 = pd.concat([elections_2000, elections_2004])
display(elections_00_04)
```

Ahora, usando `merge()`, podemos hacer uniones de las tablas de forma horizontal que compartan una columna/índice en común



Tipos de Joins

In [ ]:
```
display(counties)
```

In [ ]:
```
inner_joined=pd.merge(elections, counties)
inner_joined.head(10)
```

In [ ]:
```
left_joined=pd.merge(elections, counties, how='left')
left_joined.head(10)
```

In [ ]:
```
right_joined=pd.merge(elections, counties, how='right')
right_joined.head(10)
```

In [ ]:
```
outer_joined=pd.merge(elections, counties, how='outer')
outer_joined.head(10)
```