01_Exploracion

November 28, 2023

0.1 Operaciones Básicas: Exploración

En esta sesión vamos a tratar con operaciones básicas sobre `DataFrame` de forma explicita, pero recuerda que un `DataFrame` es una colección de objetos `Series`así que muchas veces trataremos con este tipo de objetos aunque no hagamos mención directa a ello.

0.1.1 Vistazo general

Manos a la obra, ejecuta la siguiente celda

[2]:
```
import numpy as np
import pandas as pd

df_aviones = pd.read_csv("./data/dataset_aviones.csv", index_col = "Id_vuelo")
```

Una vez tenemos un DataFrame, tenemos varias formas de explorarlo y ver su contenido. Veamos su aspecto general

[4]:
```
df_aviones.head()# con este metodo mostraremos su aspecto general viendo las 5␣
↪primeras filas
```

[4]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	avion
Air_PaGi_10737	Airnar	París	Ginebra	411	Boeing 737
Fly_BaRo_10737	FlyQ	Bali	Roma	12738	Boeing 737
Tab_GiLo_11380	TabarAir	Ginebra	Los Angeles	9103	Airbus A380
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370	Boeing 737
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480	Boeing 747

Id_vuelo	con_escala	consumo_kg	duracion	ingresos
Air_PaGi_10737	False	1028.691900	51	14232.65
Fly_BaRo_10737	True	33479.132544	1167	468527.19
Tab_GiLo_11380	False	109439.907200	626	584789.19
Mol_PaCi_10737	False	17027.010000	503	233342.51
Tab_CiRo_10747	False	86115.744000	518	438535.07

[5]:
```
df_aviones.head(15)# podemos ponerle las filas que queremos ver
```

[5]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	avion	\
Air_PaGi_10737	Airnar	París	Ginebra	411	Boeing 737	
Fly_BaRo_10737	FlyQ	Bali	Roma	12738	Boeing 737	
Tab_GiLo_11380	TabarAir	Ginebra	Los Angeles	9103	Airbus A380	
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370	Boeing 737	
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480	Boeing 747	
Mol_CaMe_10737	MoldaviAir	Cádiz	Melbourne	20029	Boeing 737	
Mol_PaLo_11320	MoldaviAir	París	Londres	344	Airbus A320	
Pam_PaMe_11380	PamPangea	París	Melbourne	16925	Airbus A380	
Pam_NuBa_10737	PamPangea	Nueva York	Bali	16589	Boeing 737	
Air_GiCa_11380	Airnar	Ginebra	Cádiz	1725	Airbus A380	
Tab_LoCi_10737	TabarAir	Los Angeles	Cincinnati	3073	Boeing 737	
Mol_LoBa_11380	MoldaviAir	Londres	Bali	12553	Airbus A380	
Tab_CiLo_10747	TabarAir	Cincinnati	Los Angeles	3073	Boeing 747	
Tab_GiLo_11380	TabarAir	Ginebra	Londres	739	Airbus A380	
Tab_CiRo_11320	TabarAir	Cincinnati	Roma	7480	Airbus A320	

Id_vuelo	con_escala	consumo_kg	duracion	ingresos
Air_PaGi_10737	False	1028.691900	51	14232.65
Fly_BaRo_10737	True	33479.132544	1167	468527.19
Tab_GiLo_11380	False	109439.907200	626	584789.19
Mol_PaCi_10737	False	17027.010000	503	233342.51
Tab_CiRo_10747	False	86115.744000	518	438535.07
Mol_CaMe_10737	True	53148.153240	1721	728045.68
Mol_PaLo_11320	False	915.246400	44	13805.52
Pam_PaMe_11380	True	217722.658400	1328	1056735.47
Pam_NuBa_10737	True	45277.618464	1459	600836.96
Air_GiCa_11380	False	20339.820000	135	110108.07
Tab_LoCi_10737	False	7915.433400	253	111056.67
Mol_LoBa_11380	False	156721.694400	856	764998.83
Tab_CiLo_10747	False	32758.180000	224	184079.01
Tab_GiLo_11380	False	8542.840000	69	46200.30
Tab_CiRo_11320	True	21087.855360	662	299451.12

[6]:
```
df_aviones.head(-5)#PREGUNTAR EN CLASE
```

[6]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	avion	con_escala	consumo_kg	duracion	ingresos
Air_PaGi_10737	Airnar	París	Ginebra	411	Boeing 737	False	1028.691900	51	14232.65
Fly_BaRo_10737	FlyQ	Bali	Roma	12738	Boeing 737	True	33479.132544	1167	468527.19
Tab_GiLo_11380	TabarAir	Ginebra	Los Angeles	9103	Airbus A380	False	109439.907200	626	584789.19
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370	Boeing 737	False	17027.010000	503	233342.51
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480	Boeing 747	False	86115.744000	518	438535.07
…	…	…	…	…	…	…	…	…	…
Tab_GiLo_10737	TabarAir	Ginebra	Los Angeles	9103	Boeing 737	False	22783.898700	711	314159.35
Pam_BaNu_10747	PamPangea	Bali	Nueva York	16589	Boeing 747	True	185751.412496	1305	962466.12
Pam_NuMe_11380	PamPangea	Nueva York	Melbourne	16082	Airbus A380	True	195277.679168	1272	1011040.68
Air_GiCa_11320	Airnar	Ginebra	Cádiz	1725	Airbus A320	False	4762.725000	145	64711.93
Fly_GiBa_11380	FlyQ	Ginebra	Bali	12383	Airbus A380	False	156030.753200	845	795002.13

[1195 rows x 9 columns]

[7]:
```
df_aviones.tail(10)# muestra las ultimas 10 filas
```

[7]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	avion
Tab_GiLo_10737	TabarAir	Ginebra	Los Angeles	9103	Boeing 737
Pam_BaNu_10747	PamPangea	Bali	Nueva York	16589	Boeing 747
Pam_NuMe_11380	PamPangea	Nueva York	Melbourne	16082	Airbus A380
Air_GiCa_11320	Airnar	Ginebra	Cádiz	1725	Airbus A320
Fly_GiBa_11380	FlyQ	Ginebra	Bali	12383	Airbus A380
Tab_LoLo_11320	TabarAir	Los Angeles	Londres	8785	Airbus A320
Mol_CiLo_10737	MoldaviAir	Cincinnati	Londres	6284	Boeing 737
Fly_RoCi_11320	FlyQ	Roma	Cincinnati	7480	Airbus A320
Tab_RoLo_10747	TabarAir	Roma	Londres	1433	Boeing 747
Air_PaLo_10737	Airnar	París	Los Angeles	9099	Boeing 737

Id_vuelo	con_escala	consumo_kg	duracion	ingresos
Tab_GiLo_10737	False	22783.898700	711	314159.35
Pam_BaNu_10747	True	185751.412496	1305	962466.12
Pam_NuMe_11380	True	195277.679168	1272	1011040.68
Air_GiCa_11320	False	4762.725000	145	64711.93
Fly_GiBa_11380	False	156030.753200	845	795002.13
Tab_LoLo_11320	True	24766.953120	756	340889.30
Mol_CiLo_10737	False	16491.729600	497	222424.54
Fly_RoCi_11320	True	19721.049920	662	285377.03
Tab_RoLo_10747	False	15734.053400	115	86373.94
Air_PaLo_10737	False	22331.675700	711	317996.77

[8]:
```
len(df_aviones)# nos dice el total de filas con informacion de viajes
```

[8]:
```
1200
```

0.1.2 Descripción inicial

Lo primero en general es qué columnas tiene:

[9]:
```
df_aviones.columns
```

[9]:
```
Index(['Aircompany', 'Origen', 'Destino', 'Distancia', 'avion', 'con_escala',
       'consumo_kg', 'duracion', 'ingresos'],
      dtype='object')
```

Una descripción general matemática de los valores numéricos:

[10]:
```
df_aviones.describe()# nos da columnas con valores numericos, y nos calcula el␣
↪nuemro total de viejes, la media, la desviacion standar, porcentajes␣
↪diversos el min y el maximo
```

[10]:

	Distancia	consumo_kg	duracion	ingresos
count	1200.000000	1200.000000	1200.000000	1.200000e+03
mean	8231.710000	68736.758895	651.153333	4.248612e+05
std	5567.286768	67605.206302	454.035184	3.184731e+05
min	344.000000	835.920000	42.000000	1.169364e+04
25%	3073.000000	15733.520400	224.000000	1.586815e+05
50%	6969.000000	38315.157192	572.000000	3.853903e+05
75%	12738.000000	120858.499500	1053.000000	6.899085e+05
max	20029.000000	264876.314560	1721.000000	1.295516e+06

Si quiero ver los tipos de cada columna

[11]:
```
df_aviones.dtypes
```

[11]:
```
Aircompany     object
Origen         object
```

```
Destino        object
Distancia       int64
avion          object
con_escala       bool
consumo_kg    float64
duracion        int64
ingresos      float64
dtype: object
```

Los tipos en Pandas se heredan parcialmente de numpy, por eso tienes int64, float64, pero además ves que los tipos string (y aquellas columnas que tengan tipos mezclados) se denominan object y que luego al tratar cada valor ya interpretará su tipo.

Ahora una descripción más completa, dentro de su generalidad, con el método info:

[13]:
```
df_aviones.info()#metodo:  el orden, el nombre, valores no nulos, y objetos de␣
↪cada tipo, que nos sirve para limpiar con los valores nulos (en otros spring)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1200 entries, Air_PaGi_10737 to Air_PaLo_10737
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Aircompany  1200 non-null   object
 1   Origen      1200 non-null   object
 2   Destino     1200 non-null   object
 3   Distancia   1200 non-null   int64
 4   avion       1200 non-null   object
 5   con_escala  1200 non-null   bool
 6   consumo_kg  1200 non-null   float64
 7   duracion    1200 non-null   int64
 8   ingresos    1200 non-null   float64
dtypes: bool(1), float64(2), int64(2), object(4)
memory usage: 85.5+ KB
```

[15]:
```
df_aviones.info# atributo # preguntar EN CLASE
```

[15]:
```
<bound method DataFrame.info of                Aircompany       Origen
Destino  Distancia        avion  \
Id_vuelo
Air_PaGi_10737      Airnar        París      Ginebra        411   Boeing 737
Fly_BaRo_10737        FlyQ         Bali         Roma      12738   Boeing 737
Tab_GiLo_11380    TabarAir      Ginebra  Los Angeles       9103  Airbus A380
Mol_PaCi_10737  MoldaviAir        París   Cincinnati       6370   Boeing 737
Tab_CiRo_10747    TabarAir   Cincinnati         Roma       7480   Boeing 747
…                    …          …          …        …          …
Tab_LoLo_11320    TabarAir  Los Angeles      Londres       8785  Airbus A320
Mol_CiLo_10737  MoldaviAir   Cincinnati      Londres       6284   Boeing 737
Fly_RoCi_11320        FlyQ         Roma   Cincinnati       7480  Airbus A320
```

```
Tab_RoLo_10747    TabarAir         Roma      Londres       1433   Boeing 747
Air_PaLo_10737      Airnar        París  Los Angeles       9099   Boeing 737

                con_escala     consumo_kg  duracion   ingresos
Id_vuelo
Air_PaGi_10737       False    1028.691900        51   14232.65
Fly_BaRo_10737        True   33479.132544      1167  468527.19
Tab_GiLo_11380       False  109439.907200       626  584789.19
Mol_PaCi_10737       False   17027.010000       503  233342.51
Tab_CiRo_10747       False   86115.744000       518  438535.07
…                        …            …       …         …
Tab_LoLo_11320        True   24766.953120       756  340889.30
Mol_CiLo_10737       False   16491.729600       497  222424.54
Fly_RoCi_11320        True   19721.049920       662  285377.03
Tab_RoLo_10747       False   15734.053400       115   86373.94
Air_PaLo_10737       False   22331.675700       711  317996.77

[1200 rows x 9 columns]>
```

0.1.3 Rascando los valores de las columnas

Pero si ahora quiero entrar en más detalle, ¿cómo puedo hacer una primera observación de los valores de una columna?

[16]:
```
df_aviones["Aircompany"].unique()# para saber el nombre de las compañias aereas␣
↪hay en el data set
```

[16]:
```
array(['Airnar', 'FlyQ', 'TabarAir', 'MoldaviAir', 'PamPangea'],
      dtype=object)
```

[18]:
```
df_aviones["avion"].unique()# tipos der aviones tiene el dataset
```

[18]:
```
array(['Boeing 737', 'Airbus A380', 'Boeing 747', 'Airbus A320'],
      dtype=object)
```

Pero igual quiero saber cómo están distribuidos

[19]:
```
df_aviones["Aircompany"].value_counts()# nuemros de vuelos que hay de cada␣
↪compañia
```

[19]:
```
Aircompany
TabarAir      271
MoldaviAir    264
PamPangea     231
Airnar        218
FlyQ          216
Name: count, dtype: int64
```

Fijate en que estos métodos, unique y value_counts son realmente métodos de series (porque al

escoger la columna primero estamos escogiendo una serie de pandas) y por tanto se pueden aplicar a cualquier serie

[20]:
```
serie= pd.Series(np.random.randint(1,5,50))
serie
```

[20]:
```
0     3
1     3
2     3
3     2
4     3
5     3
6     4
7     2
8     2
9     3
10    2
11    3
12    1
13    2
14    4
15    3
16    1
17    1
18    1
19    2
20    1
21    4
22    1
23    2
24    4
25    2
26    1
27    4
28    1
29    1
30    1
31    1
32    1
33    4
34    4
35    4
36    3
37    4
38    2
39    1
40    1
```

```
41    2
42    3
43    4
44    4
45    4
46    4
47    3
48    1
49    4
dtype: int32
```

[21]:
```
serie.unique()# nos dara el total de valores unicos divididos
```

[21]:
```
array([3, 2, 4, 1])
```

[22]:
```
serie.value_counts()# distribucion de la serie 15 nuemro 1, etc
```

[22]:
```
1    15
4    14
3    11
2    10
Name: count, dtype: int64
```

[25]:
```
serie.tail()# total de elementos que tiene cada columna
```

[25]:
```
45    4
46    4
47    3
48    1
49    4
dtype: int32
```

[26]:
```
serie.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 50 entries, 0 to 49
Series name: None
Non-Null Count  Dtype
--------------  -----
50 non-null     int32
dtypes: int32(1)
memory usage: 328.0 bytes
```

[28]:
```
serie.head()
```

[28]:
```
0    3
1    3
2    3
```

```
3    2
4    3
dtype: int32
```

[30]:
```
serie.describe()
```

[30]:
```
count    50.00000
mean      2.48000
std       1.19932
min       1.00000
25%       1.00000
50%       2.50000
75%       4.00000
max       4.00000
dtype: float64
```

[ ]:
```

```