02_Operaciones_Basicas

November 28, 2023

0.1 Operaciones Básicas (II)

En esta sesión volvemos a las operaciones básicas, pero ya no de exploración como en la sesión anterior. Ahora nos centraremos en qué posibilidades básicas hay para trabajar con las columnas, principalmente, de un `DataFrame`, comencemos creando columnas.

0.1.1 Columnas

Como en casi todos los notebooks de esta unidad, jugaremos con el dataset de aviones. Ejecuta la siguiente celda:

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

df_aviones = pd.read_csv("./data/dataset_inicial_aviones.csv", index_col =␣
 ↪"Id_vuelo")
```

Veamos sus columnas y digamos que nos gustaría añadir una columna nueva que por ahora tenga el mismo valor para todas. Por ejemplo, una columna booleana que nos diga si el viaje llego en hora.

[3]:
```
df_aviones.head()
```

[3]:

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

                   consumo_kg  duracion
Id_vuelo
Air_PaGi_10737    1028.691900        51
Fly_BaRo_10737   33479.132544      1167
Tab_GiLo_11380  109439.907200       626
Mol_PaCi_10737   17027.010000       503
Tab_CiRo_10747   86115.744000       518
```

[4]:
```
df_aviones.columns
```

[4]:
```
Index(['Aircompany', 'Origen', 'Destino', 'Distancia', 'avion', 'consumo_kg',
       'duracion'],
      dtype='object')
```

una columna booleana que nos diga si el viaje llego en hora

[5]:
```
df_aviones["En_hora"]= True# con el metodo de un doccionario
df_aviones# si no ponemos ningun metodo ni atributo al DF, nos dara las 5␣
 ↪primeras filas y las 5 ultimas(esto solo sire notebook no es asi script .py)
```

[5]:
```
                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

                   consumo_kg  duracion  En_hora
Id_vuelo
Air_PaGi_10737    1028.691900        51     True
Fly_BaRo_10737   33479.132544      1167     True
Tab_GiLo_11380  109439.907200       626     True
Mol_PaCi_10737   17027.010000       503     True
Tab_CiRo_10747   86115.744000       518     True
…                         …         …        …
Tab_LoLo_11320   24766.953120       756     True
Mol_CiLo_10737   16491.729600       497     True
Fly_RoCi_11320   19721.049920       662     True
```

```
Tab_RoLo_10747   15734.053400       115     True
Air_PaLo_10737   22331.675700       711     True

[1200 rows x 8 columns]
```

Y ahora algo más útil de forma directa, el consumo por kilometro:

[6]:
```
df_aviones["consumo_por_km"] = df_aviones["consumo_kg"] /␣
 ↪df_aviones["Distancia"]# divide elemento a elmento
df_aviones["consumo_por_km"] # si lo hago asi, me devuelve el resultado como si␣
 ↪fuera una serie panda
```

[6]:
```
Id_vuelo
Air_PaGi_10737     2.502900
Fly_BaRo_10737     2.628288
Tab_GiLo_11380    12.022400
Mol_PaCi_10737     2.673000
Tab_CiRo_10747    11.512800
                     …
Tab_LoLo_11320     2.819232
Mol_CiLo_10737     2.624400
Fly_RoCi_11320     2.636504
Tab_RoLo_10747    10.979800
Air_PaLo_10737     2.454300
Name: consumo_por_km, Length: 1200, dtype: float64
```

[7]:
```
#df_aviones# asi nos dara como dataframe
display(df_aviones)# tb nos dara la vision de dataframa, pero con print no dara␣
 ↪tb la vision como  serie panda
```

```
                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

                  consumo_kg  duracion  En_hora  consumo_por_km
Id_vuelo
Air_PaGi_10737   1028.691900        51     True        2.502900
Fly_BaRo_10737  33479.132544      1167     True        2.628288
```

```
Tab_GiLo_11380  109439.907200       626     True       12.022400
Mol_PaCi_10737   17027.010000       503     True        2.673000
Tab_CiRo_10747   86115.744000       518     True       11.512800
…                          …       …       …              …
Tab_LoLo_11320   24766.953120       756     True        2.819232
Mol_CiLo_10737   16491.729600       497     True        2.624400
Fly_RoCi_11320   19721.049920       662     True        2.636504
Tab_RoLo_10747   15734.053400       115     True       10.979800
Air_PaLo_10737   22331.675700       711     True        2.454300

[1200 rows x 9 columns]
```

Podríamos haber usado los atributos:

[8]:
```
df_aviones["consumo_por_Km"] = df_aviones.consumo_kg / df_aviones.Distancia
df_aviones
```

[8]:
```
                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

                   consumo_kg  duracion  En_hora  consumo_por_km  \
Id_vuelo
Air_PaGi_10737    1028.691900        51     True        2.502900
Fly_BaRo_10737   33479.132544      1167     True        2.628288
Tab_GiLo_11380  109439.907200       626     True       12.022400
Mol_PaCi_10737   17027.010000       503     True        2.673000
Tab_CiRo_10747   86115.744000       518     True       11.512800
…                         …       …       …              …
Tab_LoLo_11320   24766.953120       756     True        2.819232
Mol_CiLo_10737   16491.729600       497     True        2.624400
Fly_RoCi_11320   19721.049920       662     True        2.636504
Tab_RoLo_10747   15734.053400       115     True       10.979800
Air_PaLo_10737   22331.675700       711     True        2.454300

                consumo_por_Km
Id_vuelo
Air_PaGi_10737        2.502900
```

```
Fly_BaRo_10737        2.628288
Tab_GiLo_11380       12.022400
Mol_PaCi_10737        2.673000
Tab_CiRo_10747       11.512800
…                          …
Tab_LoLo_11320        2.819232
Mol_CiLo_10737        2.624400
Fly_RoCi_11320        2.636504
Tab_RoLo_10747       10.979800
Air_PaLo_10737        2.454300

[1200 rows x 10 columns]
```

[ ]:

Pero quizá nos ha quedado un nombre de columna largo o creemos que algún nombre podría estar mejor expresado de otra forma... ¿Cómo cambiamos los nombres de las columnas?

[9]:
```
df_aviones.rename(columns ={"consumo_por_km": "Consumo/km", "avion":"Avion",␣
↪"consumo_kg":"Consumo/kg"}) # esto no cambia la variable aviones solo lo␣
↪muestra , si qeuisieramos cambair la variable tendria que reasignar o usar␣
↪el atributo inplace = True
```

[9]:
```
                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

                  Consumo/kg  duracion  En_hora  Consumo/km  consumo_por_Km
Id_vuelo
Air_PaGi_10737    1028.691900        51     True    2.502900        2.502900
Fly_BaRo_10737   33479.132544      1167     True    2.628288        2.628288
Tab_GiLo_11380  109439.907200       626     True   12.022400       12.022400
Mol_PaCi_10737   17027.010000       503     True    2.673000        2.673000
Tab_CiRo_10747   86115.744000       518     True   11.512800       11.512800
…                         …         …        …           …               …
Tab_LoLo_11320   24766.953120       756     True    2.819232        2.819232
Mol_CiLo_10737   16491.729600       497     True    2.624400        2.624400
Fly_RoCi_11320   19721.049920       662     True    2.636504        2.636504
```

```
Tab_RoLo_10747   15734.053400       115     True   10.979800       10.979800
Air_PaLo_10737   22331.675700       711     True    2.454300        2.454300

[1200 rows x 10 columns]
```

[10]:
```
df_aviones
```

[10]:
```
                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

                   consumo_kg  duracion  En_hora  consumo_por_km  \
Id_vuelo
Air_PaGi_10737    1028.691900        51     True        2.502900
Fly_BaRo_10737   33479.132544      1167     True        2.628288
Tab_GiLo_11380  109439.907200       626     True       12.022400
Mol_PaCi_10737   17027.010000       503     True        2.673000
Tab_CiRo_10747   86115.744000       518     True       11.512800
…                         …       …      …                …
Tab_LoLo_11320   24766.953120       756     True        2.819232
Mol_CiLo_10737   16491.729600       497     True        2.624400
Fly_RoCi_11320   19721.049920       662     True        2.636504
Tab_RoLo_10747   15734.053400       115     True       10.979800
Air_PaLo_10737   22331.675700       711     True        2.454300

                consumo_por_Km
Id_vuelo
Air_PaGi_10737        2.502900
Fly_BaRo_10737        2.628288
Tab_GiLo_11380       12.022400
Mol_PaCi_10737        2.673000
Tab_CiRo_10747       11.512800
…                          …
Tab_LoLo_11320        2.819232
Mol_CiLo_10737        2.624400
Fly_RoCi_11320        2.636504
Tab_RoLo_10747       10.979800
```

```
Air_PaLo_10737        2.454300

[1200 rows x 10 columns]
```

O lo reasigno o bien utilizo el argumento `inplace` que es un argumento que existe en muchos métodos de los `DataFrame`.

[11]:
```
# Reasignado
df_aviones_2 = df_aviones.rename(columns ={"consumo_por_km": "Consumo/km",␣
 ↪"avion":"Avion", "consumo_kg":"Consumo/kg"})
df_aviones_2
```

[11]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	Avion	Consumo/kg	duracion	En_hora	Consumo/km	consumo_por_Km
Air_PaGi_10737	Airnar	París	Ginebra	411	Boeing 737	1028.691900	51	True	2.502900	2.502900
Fly_BaRo_10737	FlyQ	Bali	Roma	12738	Boeing 737	33479.132544	1167	True	2.628288	2.628288
Tab_GiLo_11380	TabarAir	Ginebra	Los Angeles	9103	Airbus A380	109439.907200	626	True	12.022400	12.022400
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370	Boeing 737	17027.010000	503	True	2.673000	2.673000
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480	Boeing 747	86115.744000	518	True	11.512800	11.512800
…	…	…	…	…	…	…	…	…	…	…
Tab_LoLo_11320	TabarAir	Los Angeles	Londres	8785	Airbus A320	24766.953120	756	True	2.819232	2.819232
Mol_CiLo_10737	MoldaviAir	Cincinnati	Londres	6284	Boeing 737	16491.729600	497	True	2.624400	2.624400
Fly_RoCi_11320	FlyQ	Roma	Cincinnati	7480	Airbus A320	19721.049920	662	True	2.636504	2.636504
Tab_RoLo_10747	TabarAir	Roma	Londres	1433	Boeing 747	15734.053400	115	True	10.979800	10.979800
Air_PaLo_10737	Airnar	París	Los Angeles	9099	Boeing 737	22331.675700	711	True	2.454300	2.454300

[1200 rows x 10 columns]

[15]:
```
# Inplace , cambia el archivo original
df_aviones = df_aviones.rename(columns ={"consumo_por_km": "Consumo/km",␣
 ↪"avion":"Avion", "consumo_kg":"Consumo/kg"},inplace= True)# columns porque␣
 ↪cambia nombre de columnas si fura fila fill
```

```
---------------------------------------------------------------------------
```

```
AttributeError                            Traceback (most recent call last)
e:
 ↪\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\ONLINE_DS_THEBRIDGE_V-1\SPRING␣
 ↪4\UNIT 2\02_Operaciones_Basicas.ipynb Celda 23 line 2

      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/ONLINE_DS_THEBRIDGE_V-1/SPRING%204/UNIT%202/
 ↪02_Operaciones_Basicas.ipynb#X31sZmlsZQ%3D%3D?line=0'>1</a> # Inplace , cambi␣
 ↪el archivo original
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/ONLINE_DS_THEBRIDGE_V-1/SPRING%204/UNIT%202/
 ↪02_Operaciones_Basicas.ipynb#X31sZmlsZQ%3D%3D?line=1'>2</a> df_aviones =␣
 ↪df_aviones.rename(columns ={"consumo_por_km": "Consumo/km", "avion":"Avion",␣
 ↪"consumo_kg":"Consumo/kg"},inplace= True)# columns porque cambia nombre de␣
 ↪columnas si fura fila fill


AttributeError: 'NoneType' object has no attribute 'rename'
```

0.1.2 Operaciones Sencillas

Veamos para terminar algunas operaciones sencillas de agregación que te sonarán porque se comparten casi en su totalidad con numpy. Para ello iremos contestando a una serie de preguntas [Que es otra forma de explorar los datos]

[14]:
```
# Cual es la mayor distancia recorrida
df_aviones["Distancia"].max()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
e:
 ↪\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\ONLINE_DS_THEBRIDGE_V-1\SPRING␣
 ↪4\UNIT 2\02_Operaciones_Basicas.ipynb Celda 26 line 2

      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/ONLINE_DS_THEBRIDGE_V-1/SPRING%204/UNIT%202/
 ↪02_Operaciones_Basicas.ipynb#X34sZmlsZQ%3D%3D?line=0'>1</a> # Cual es la mayo␣
 ↪distancia recorrida
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/ONLINE_DS_THEBRIDGE_V-1/SPRING%204/UNIT%202/
 ↪02_Operaciones_Basicas.ipynb#X34sZmlsZQ%3D%3D?line=1'>2</a>␣
 ↪df_aviones["Distancia"].max()


TypeError: 'NoneType' object is not subscriptable
```

[ ]:
```
# Cual es el menor consumo
df_aviones["Consumo/Kg"].min()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 27 line 2

     <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X35sZmlsZQ%3D%3D?line=0'>1</a> # Cual es el menor consumo
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X35sZmlsZQ%3D%3D?line=1'>2</a> df_aviones["Consumo/Kg"].min()

TypeError: 'NoneType' object is not subscriptable
```

[ ]:
```
# Cuanta distancia se han recorrido en los 1200 vuelos
df_aviones["Distancia"].sum()
```

[ ]:
```
9878052
```

[ ]:
```
# Cual es la media recorrida por estos viajes
df_aviones["Distancia"].mean()
```

[ ]:
```
8231.71
```

[ ]:
```
# Y el consumo medio
df_aviones["Consumo/Kg"].mean()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexes\
 ↪py:3790, in Index.get_loc(self, key)
   3789 try:
-> 3790     return self._engine.get_loc(casted_key)
   3791 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: 'Consumo/Kg'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
```

```
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 30 line 2

     <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X41sZmlsZQ%3D%3D?line=0'>1</a> # Y el consumo medio
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X41sZmlsZQ%3D%3D?line=1'>2</a> df_aviones["Consumo/Kg"].mean()

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\frame.
 ↪py:3893, in DataFrame.__getitem__(self, key)
  3891 if self.columns.nlevels > 1:
  3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
  3894 if is_integer(indexer):
  3895     indexer = [indexer]

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexes\
 ↪py:3797, in Index.get_loc(self, key)
  3792     if isinstance(casted_key, slice) or (
  3793         isinstance(casted_key, abc.Iterable)
  3794         and any(isinstance(x, slice) for x in casted_key)
  3795     ):
  3796         raise InvalidIndexError(key)
-> 3797     raise KeyError(key) from err
  3798 except TypeError:
  3799     # If we have a listlike key, _check_indexing_error will raise
  3800     #  InvalidIndexError. Otherwise we fall through and re-raise
  3801     #  the TypeError.
  3802     self._check_indexing_error(key)

KeyError: 'Consumo/Kg'
```

Bueno, como medidas agregadas están bien, pero si quiero algo más de detalle y sin entrar en como quedarnos con solo las filas que cumplan una condición también podemos hacer lo siguiente

[ ]:
```
# Cual es el viaje con menor consumo
viaje = df_aviones["Consumo_kg"].idxmin()# el id de vuelo con menor consumo
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 32 line 2

     <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X43sZmlsZQ%3D%3D?line=0'>1</a> # Cual es el viaje con menor consumo
```

```
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X43sZmlsZQ%3D%3D?line=1'>2</a> viaje = df_aviones["Consumo_kg"].
 ↪idxmin()# el id de vuelo con menor consumo

NameError: name 'df_aviones' is not defined
```

[ ]:
```
df_aviones.loc(viaje)# aqui nos dara informacion de toda la fila del avion con␣
 ↪menor consumo (Mol PaLo_10737)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 33 line 1
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X44sZmlsZQ%3D%3D?line=0'>1</a> df_aviones.loc(viaje)

NameError: name 'viaje' is not defined
```

[ ]:
```
# Cual es el avion con el mayor consumo medio
viaje = df_aviones["Consumo/km"].idxmax()# nos dara el veulo con mayor consumo␣
 ↪( MoL_MeCa_11380)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexes\
 ↪py:3790, in Index.get_loc(self, key)
   3789 try:
-> 3790     return self._engine.get_loc(casted_key)
   3791 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: 'Consumo/km'

The above exception was the direct cause of the following exception:
```

```
KeyError                                  Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 34 line 2

      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X45sZmlsZQ%3D%3D?line=0'>1</a> # Cual es el avion con el mayor consumo␣
 ↪medio
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X45sZmlsZQ%3D%3D?line=1'>2</a> viaje = df_aviones["Consumo/km"].idxmax(

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\frame.
 ↪py:3893, in DataFrame.__getitem__(self, key)
   3891 if self.columns.nlevels > 1:
   3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
   3894 if is_integer(indexer):
   3895     indexer = [indexer]

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexes\
 ↪py:3797, in Index.get_loc(self, key)
   3792     if isinstance(casted_key, slice) or (
   3793         isinstance(casted_key, abc.Iterable)
   3794         and any(isinstance(x, slice) for x in casted_key)
   3795     ):
   3796         raise InvalidIndexError(key)
-> 3797     raise KeyError(key) from err
   3798 except TypeError:
   3799     # If we have a listlike key, _check_indexing_error will raise
   3800     #  InvalidIndexError. Otherwise we fall through and re-raise
   3801     #  the TypeError.
   3802     self._check_indexing_error(key)

KeyError: 'Consumo/km'
```

[ ]:
```
# si qyeremos mas informacion de la fila del vuelo:
df_aviones.loc[viaje]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 35 line 2

      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X46sZmlsZQ%3D%3D?line=0'>1</a> # si qyeremos mas informacion de la fila␣
 ↪del vuelo:
```

```
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X46sZmlsZQ%3D%3D?line=1'>2</a> df_aviones.loc[viaje]

NameError: name 'viaje' is not defined
```

[ ]:
```
# Pero si solo queremos ver el uno...
df_aviones.loc[viaje,"Avion"].unique()# nos solo el avion que gasto mayor␣
 ↪consumo. esto nos dara un array,
#si quisieramos que nos lo imprima normal un print:
valor_avion= df_aviones.loc[viaje,"Avion"].unique()[0]
print(valor_avion)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪2\02_Operaciones_Basicas.ipynb Celda 36 line 2

      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X50sZmlsZQ%3D%3D?line=0'>1</a> # Pero si solo queremos ver el uno…
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X50sZmlsZQ%3D%3D?line=1'>2</a> df_aviones.loc[viaje,"Avion"].unique()#␣
 ↪nos solo el avion que gasto mayor consumo. esto nos dara un array,
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X50sZmlsZQ%3D%3D?line=2'>3</a> #si quisieramos que nos lo imprima norma␣
 ↪un print:
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%202/02_Operaciones_Basicas.
 ↪ipynb#X50sZmlsZQ%3D%3D?line=3'>4</a> valor_avion= df_aviones.
 ↪loc[viaje,"Avion"].unique()[0]

NameError: name 'viaje' is not defined
```

Por si quieres practicas, las siguientes agregaciones vienen con el paquete de Pandas:

Agregación	Descripción
`count()`	Número total de elementos
`first()`, `last()`	Primer y último elemento
`mean()`, `median()`	Media y mediana
`min()`, `max()`	Mínimo y máximo
`std()`, `var()`	Desviación estándar y varianza
`mad()`	Desviación media absoluta
`prod()`	Producto de todos los elementos
`sum()`	Suma de todos los elementos

Todos están presentes como objetos de `Dataframe` y `Series`.

[ ]:

[ ]:

[ ]: