04_Operaciones_Sobre_Filtrado

November 28, 2023

0.1 Operaciones sobre subconjuntos

En esta sesión vamos a aprovechar lo aprendido sobre filtrado en la anterior para trabajar solo con parte del dataframe, de forma que podamos hacer cálculos más precios, obtener respuestas más concretas y manipular datos de una manera selectiva.

Y para empezar, como en las últimas sesiones, recuperemos nuestra `DataFrame` de referencia:

[1]:
```
import numpy as  np
import pandas as pd

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

[27]:
```
df_aviones
```

[27]:

Id_vuelo	Aircompany	Origen	Destino	Distancia
Air_PaGi_10737	Airnar	París	Ginebra	411
Fly_BaRo_10737	FlyQ	Bali	Roma	12738
Tab_GiLo_11380	TabarAir	Ginebra	San Francisco	9103
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480
…	…	…	…	…
Tab_LoLo_11320	TabarAir	Los Angeles	Londres	8785
Mol_CiLo_10737	MoldaviAir	Cincinnati	Londres	6284
Fly_RoCi_11320	FlyQ	Roma	Cincinnati	7480

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

                       avion     consumo_kg  duracion  supertrayectos
Id_vuelo
Air_PaGi_10737   Boeing 737    1028.691900        51           False
Fly_BaRo_10737   Boeing 737   33479.132544      1167            True
Tab_GiLo_11380  Airbus A380  109439.907200       626           False
Mol_PaCi_10737   Boeing 737   17027.010000       503           False
Tab_CiRo_10747   Boeing 747   86115.744000       518           False
…                      …           …       …             …
Tab_LoLo_11320  Airbus A320   24766.953120       756           False
Mol_CiLo_10737   Boeing 737   16491.729600       497           False
Fly_RoCi_11320  Airbus A320   19721.049920       662           False
Tab_RoLo_10747   Boeing 747   15734.053400       115           False
Air_PaLo_10737   Boeing 737   22331.675700       711           False

[1200 rows x 8 columns]
```

0.1.1 Operaciones y Consultas Selectivas

Para empezar vamos a realizar algunas operaciones de las que hicimos hace dos sesiones pero ahora ya podemos ser más selectivos

NOTA PERSONAL :El metodo **groupby** divide el DataFrame en grupos de filas y columnas y cada grupo contendrá filas con valores idénticos en las columnas de agrupación, obteniendo un objeto de grupo con referencias a cada uno de los grupos creados, Despues puedes hacer operaciones diversas en esos grupos como la media, la suma, el máximo, el mínimo o cualquier otra operación en función de tus necesidades.

[40]:
```
#METODO FACIL PARA MAS ADEALNTE#
""" mayor distancia por compañias
Distancia_company = df_aviones.groupby("Aircompany")["Distancia"].max()#␣
 ↪agrupame en el dataframe las compañias con las mayores distancias recorridas␣
 ↪en sus viajes
display(Distancia_company)"""
```

[40]:
```
' mayor distancia por compañias\nDistancia_company =
df_aviones.groupby("Aircompany")["Distancia"].max()# agrupame en el dataframe
las compañias con las mayores distancias recorridas en sus
viajes\ndisplay(Distancia_company)'
```

[2]:
```
### mayor distancia por compañias:


"""#creo un diccionario para guarda de la mayor distancia recorrida por compañía
dict_mayor_dist_c = {}
```

```
# Obtener las compañías únicas
company_unicas = df_aviones['Aircompany'].unique()
#print(company_unicas) # lista de las 5 compañias

# Iterar a través de las compañías únicas, filtrando el DF por compañia
for company in company_unicas:
    listado_company = df_aviones[df_aviones['Aircompany'] == company]
    #print(listado_company) # extrayendo un listado de 231 elementos con␣
 ↪infromacion del DF

    # calculo con un array la distancia máxima para cada compañía
    distancia_maxima = np.max(listado_company['Distancia'])
    #print(distancia_maxima)# 5 distancias maximas

    # Guardo la distancia máxima en el diccionario (diccionario - clave - valor)
    dict_mayor_dist_c [company] = distancia_maxima
    #print(company) # misma informacion que company_unicas pero al iterar esta␣
 ↪en una columna

# Imprimir la mayor distancia por compañía, iterando por el dict en clave y␣
 ↪valor
for company, distancia in dict_mayor_dist_c.items():
    print(f"La Compañía {company}, ha recorrido un total de {distancia}␣
 ↪kilometros, siendo su mayor distancia recorrida")"""
```

[2]:
```
'#creo un diccionario para guarda de la mayor distancia recorrida por
compañía\ndict_mayor_dist_c = {}\n\n# Obtener las compañías
únicas\ncompany_unicas =
df_aviones[\'Aircompany\'].unique()\n#print(company_unicas) # lista de las 5
compañias\n\n# Iterar a través de las compañías únicas, filtrando el DF por
compañia\nfor company in company_unicas:\n    listado_company =
df_aviones[df_aviones[\'Aircompany\'] == company]\n    #print(listado_company) #
extrayendo un listado de 231 elementos con infromacion del DF\n    \n    #
calculo con un array la distancia máxima para cada compañía \n
distancia_maxima = np.max(listado_company[\'Distancia\'])\n
#print(distancia_maxima)# 5 distancias maximas\n\n    # Guardo la distancia
máxima en el diccionario (diccionario - clave - valor)\n    dict_mayor_dist_c
[company] = distancia_maxima\n    #print(company) # misma informacion que
company_unicas pero al iterar esta en una columna\n\n# Imprimir la mayor
distancia por compañía, iterando por el dict en clave y valor\nfor company,
distancia in dict_mayor_dist_c.items():\n    print(f"La Compañía {company}, ha
recorrido un total de {distancia} kilometros, siendo su mayor distancia
recorrida")'
```

[3]:
```
### mayor distancia por compañias:## ESTA ES LA DEL PROFE
```

```python
# itero sobre el las compañias del DF en valores unicos no todos los valores de␣
 ↪las 1200 columnas
for company in df_aviones["Aircompany"].unique():
    distancia_max = df_aviones.loc[df_aviones["Aircompany"] == company,␣
 ↪"Distancia"]# hago un filtrado con una serie panda con la mascara␣
 ↪(DF[compañia ])de las compañias del DF como filtro == me quedo con las␣
 ↪compañias y distancia
    # cumplen la condicion
    print(f"La mayor distancia cubierta por la compañia {company} es de␣
 ↪{distancia_max} Km.")
```

```
La mayor distancia cubierta por la compañia Airnar es de Id_vuelo
Air_PaGi_10737      411
Air_GiCa_11380     1725
Air_GiLo_10747     9103
Air_BaGi_11380    12383
Air_BaCa_10737    12798
                  …
Air_PaCa_11320     1447
Air_GiCa_10737     1725
Air_GiCa_10747     1725
Air_GiCa_11320     1725
Air_PaLo_10737     9099
Name: Distancia, Length: 218, dtype: int64 Km.
La mayor distancia cubierta por la compañia FlyQ es de Id_vuelo
Fly_BaRo_10737    12738
Fly_RoNu_11320     6877
Fly_GiCi_10737     6969
Fly_GiRo_10737      698
Fly_BaGi_10737    12383
                  …
Fly_NuBa_11380     6170
Fly_NuRo_11320     6877
Fly_BaRo_10747    12738
Fly_GiBa_11380    12383
Fly_RoCi_11320     7480
Name: Distancia, Length: 216, dtype: int64 Km.
La mayor distancia cubierta por la compañia TabarAir es de Id_vuelo
Tab_GiLo_11380    9103
Tab_CiRo_10747    7480
Tab_LoCi_10737    3073
Tab_CiLo_10747    3073
Tab_GiLo_11380     739
                  …
Tab_NuGi_10747    6206
Tab_GiCi_11320    6969
```

```
Tab_GiLo_10737    9103
Tab_LoLo_11320    8785
Tab_RoLo_10747    1433
Name: Distancia, Length: 271, dtype: int64 Km.
La mayor distancia cubierta por la compañia MoldaviAir es de Id_vuelo
Mol_PaCi_10737     6370
Mol_CaMe_10737    20029
Mol_PaLo_11320      344
Mol_LoBa_11380    12553
Mol_MeCa_10737    20029
                  …
Mol_PaLo_11320      344
Mol_PaMe_11380    16925
Mol_LoCi_10737     6284
Mol_BaLo_10737    12553
Mol_CiLo_10737     6284
Name: Distancia, Length: 264, dtype: int64 Km.
La mayor distancia cubierta por la compañia PamPangea es de Id_vuelo
Pam_PaMe_11380    16925
Pam_NuBa_10737    16589
Pam_MePa_10737    16925
Pam_GiNu_11320     6206
Pam_MeBa_10737     2779
                  …
Pam_GiMe_10747    16674
Pam_BaMe_11320     2779
Pam_GiMe_10737    16674
Pam_BaNu_10747    16589
Pam_NuMe_11380    16082
Name: Distancia, Length: 231, dtype: int64 Km.
```

[17]:
```
#METODO FACIL PARA MAS ADEALNTE#
"""viaje_menor = df_aviones.groupby("avion")["consumo_kg"].min()# agrupame en␣
 ↪el dataframe las aviones con los menores consumo por viaje
print(viaje_menor)"""
```

```
avion
Airbus A320     863.4400
Airbus A380    3976.6400
Boeing 737      835.9200
Boeing 747     3740.3808
Name: consumo_kg, dtype: float64
```

[3]:
```
"""#### Cual es el viaje con menor consumo por tipo de avion

# creo un diccionario para los viajes con menor consumo por tipo de avión
dict_menor_consumo_avion = {}
```

```
# Encuentra los tipos únicos de avión
tipos = df_aviones["avion"].unique()
#print(tipos)# una lista con 4 tipos de aviones

# Itera a través de los tipos de avión, filtrando el DF para cada tipo de avion
for tipo_avion in tipos:
    avionacos = df_aviones[df_aviones["avion"] == tipo_avion]
    #print(avionacos) #extrayendo un listado de 315 elementos con informaion de␣
 ↪todas las columnas de DF

    # claculo el índice de la fila donde se encuntre el valor minimo del viaje␣
 ↪con el menor consumo en ese tipo de avión
    indice_ecologista = avionacos["consumo_kg"].idxmin()
    #print(indice_ecologista) # informacion con id de  4 vuelos que seran los␣
 ↪de menor consumo

    # mediante una serie panda obtengo la fila correspondiente al viaje con el␣
 ↪menor consumo
    viaje_ecologista = avionacos.loc[indice_ecologista]

    # guardo el resultado en el diccionario (dict - clave - valor)
    dict_menor_consumo_avion[tipo_avion] = viaje_ecologista

# Imprimo los viajes con menor consumo por tipo de avión, iterando por el dict␣
 ↪en clave y calor
for tipo_avion, viaje in dict_menor_consumo_avion.items():
    print(f"El Tipo de Avión {tipo_avion} ha realizado los siguientes viajes␣
 ↪con menor consumo:\n {viaje}\n")"""
```

[3]:
```
'#### Cual es el viaje con menor consumo por tipo de avion \n\n# creo un
diccionario para los viajes con menor consumo por tipo de
avión\ndict_menor_consumo_avion = {}\n\n# Encuentra los tipos únicos de
avión\ntipos = df_aviones["avion"].unique()\n#print(tipos)# una lista con 4
tipos de aviones\n\n# Itera a través de los tipos de avión, filtrando el DF para
cada tipo de avion\nfor tipo_avion in tipos:\n    avionacos =
df_aviones[df_aviones["avion"] == tipo_avion]\n    #print(avionacos) #extrayendo
un listado de 315 elementos con informaion de todas las columnas de DF\n
\n    # claculo el índice de la fila donde se encuntre el valor minimo del viaje
con el menor consumo en ese tipo de avión\n    indice_ecologista =
avionacos["consumo_kg"].idxmin()\n    #print(indice_ecologista) # informacion
con id de  4 vuelos que seran los de menor consumo\n    \n    # mediante una
serie panda obtengo la fila correspondiente al viaje con el menor consumo\n
viaje_ecologista = avionacos.loc[indice_ecologista]\n    \n    # guardo el
resultado en el diccionario (dict - clave - valor)\n
dict_menor_consumo_avion[tipo_avion] = viaje_ecologista\n\n# Imprimo los viajes
con menor consumo por tipo de avión, iterando por el dict en clave y calor\nfor
```

```
tipo_avion, viaje in dict_menor_consumo_avion.items():\n    print(f"El Tipo de
Avión {tipo_avion} ha realizado los siguientes viajes con menor consumo:\n
{viaje}\n")'
```

[30]:
```
#METODO FACIL PARA MAS ADEALNTE#
"""aviones_company = df_aviones.groupby("avion")["Aircompany"].unique()#␣
 ↪agrupame el el DF los tipos de avion que esa las compañias en una lista de␣
 ↪valores unicos
print("Las compañias que usan cada tipo de avion son\n",aviones_company)
print("\n")
numero_aviones_company = df_aviones.groupby("avion")["Aircompany"].nunique()#␣
 ↪agrupa el numero de compañias que usas los diferentes tipos de aviones a␣
 ↪partir de valores unicos en la columna "avion" de cada compañia
display("El numero de compañias que usan los distintos tipos de aviones␣
 ↪son\n",numero_aviones_company)"""
```

```
Las compañias que usan cada tipo de avion son
 avion
Airbus A320     [MoldaviAir, TabarAir, PamPangea, FlyQ, Airnar]
Airbus A380     [TabarAir, PamPangea, Airnar, MoldaviAir, FlyQ]
Boeing 737      [Airnar, FlyQ, MoldaviAir, PamPangea, TabarAir]
Boeing 747      [TabarAir, Airnar, MoldaviAir, PamPangea, FlyQ]
Name: Aircompany, dtype: object
```

```
'El numero de compañias que usan los distintos tipos de aviones son\n'
```

```
avion
Airbus A320    5
Airbus A380    5
Boeing 737     5
Boeing 747     5
Name: Aircompany, dtype: int64
```

[78]:
```
## Cual es el viaje con menor consumo por tipo de avion

viaje = df_aviones["consumo_kg"].idxmin()


for tipo_avion in df_aviones["avion"].unique():
    res = df_aviones.loc[df_aviones["avion"] ==  viaje]
    #print(avioneta)



#print(condicion)
print(f" EL viaje con el consumo minimo {viaje} por tipo de avion: {res}")
```

```
 EL viaje con el consumo minimo Mol_PaLo_10737 por tipo de avion: Empty
DataFrame
Columns: [Aircompany, Origen, Destino, Distancia, avion, consumo_kg, duracion,
supertrayectos]
Index: []
```

[ ]:

[5]:
```
"""### Cuántos aviones usan de cada tipo en cada compañía

# creo un diccionario para guardar los resultados
dict_aviones_company = {}

# Encuentra los tipos únicos de avión
tipos = df_aviones["avion"].unique()
#print(tipos)# una lista con 4 tipos de aviones

# itero por los tipops de aviones , filtrando en el dataframe toda la␣
 ↪infomacion
for tipo in tipos:
    avioncillos = df_aviones[df_aviones["avion"] == tipo]
    #print(avioncillos) # salen 198 elementos con informacion de todas las␣
 ↪columnas del DF

# Itera a través de las compañías únicas, Filtrando el DataFrame por compañía
for company in avioncillos["Aircompany"].unique():
    filas_tipo_av = avioncillos[avioncillos["Aircompany"] == company]

# Ccantidad de aviones de ecada tipo en cada compañía
num_aviones = len(filas_tipo_av)

#almacena el resultado en el diccionario
dict_aviones_company[(company, tipo)] = num_aviones

# Imprimo los diferentes tipos de avión que usan cada compañia, iterando por el␣
 ↪dict en clave y calor
for (company, tipo), numero in dict_aviones_company.items():
    print(f" La compañía {company} usa {numero} aviones , siendo sus tipos:␣
 ↪{tipo}")"""
```

[5]:
```
'### Cuántos aviones usan de cada tipo en cada compañía\n\n# creo un diccionario
para guardar los resultados\ndict_aviones_company = {}\n\n# Encuentra los tipos
únicos de avión\ntipos = df_aviones["avion"].unique()\n#print(tipos)# una lista
con 4 tipos de aviones\n\n# itero por los tipops de aviones , filtrando en el
dataframe toda la infomacion \nfor tipo in tipos:\n    avioncillos =
df_aviones[df_aviones["avion"] == tipo]\n    #print(avioncillos) # salen 198
```

```
elementos con informacion de todas las columnas del DF\n\n# Itera a través de
las compañías únicas, Filtrando el DataFrame por compañía\nfor company in
avioncillos["Aircompany"].unique():\n    filas_tipo_av =
avioncillos[avioncillos["Aircompany"] == company]\n    \n# Ccantidad de aviones
de ecada tipo en cada compañía \nnum_aviones = len(filas_tipo_av)\n\n#almacena
el resultado en el diccionario\ndict_aviones_company[(company, tipo)] =
num_aviones\n\n# Imprimo los diferentes tipos de avión que usan cada compañia,
iterando por el dict en clave y calor\nfor (company, tipo), numero in
dict_aviones_company.items():\n    print(f" La compañía {company} usa {numero}
aviones , siendo sus tipos: {tipo}")'
```

[34]:
```
#METODO FACIL PARA MAS ADEALNTE#
"""a_Ginebra=(df_aviones['Destino'] == "Ginebra").sum()# buscame en el DF los␣
 ↪vuelos con destinos, sumados previante ,a Ginebra
print(f"Los vuelos con destino a Ginebra son un total de {a_Ginebra}")"""
```

```
Los vuelos con destino a Ginebra son un total de 163
```

[8]:
```
######## Cuántos aviones usan de cada tipo en cada compañía ##### ESTA ES LA␣
 ↪DEL PROFE

# itero sobre el las compañias del DF en valores unicos no todos los valores de␣
 ↪todas las columnas
for company in df_aviones["Aircompany"].unique():
      # da el mismo valor pero la diferencia entre poner df_aviones.Aircompany␣
 ↪( atributo ) y df_aviones ["Aircompany"], es que si la columna Aircompany ,␣
 ↪fuera Air company(separado) con el atributo daria error
    print(f" la distribucion de aviones de uso por tipo de avion para la␣
 ↪compañia {company} es:")
    print(df_aviones.loc[df_aviones.Aircompany == company, "avion"].
 ↪value_counts())# filtramos por sere panda en el DF por␣
 ↪compañias(DF[compañia])  para que nos de las columnas compañia y avion, pero␣
 ↪como quiero
    # saber la distrubucion de uso( realemnte el nuemro de ves que usa cada␣
 ↪avion cada compañia) , uso al final el metodo .value_counts()
```

```
 la distribucion de aviones de uso por tipo de avion para la compañia Airnar es:
avion
Boeing 747     68
Airbus A380    64
Boeing 737     49
Airbus A320    37
Name: count, dtype: int64
 la distribucion de aviones de uso por tipo de avion para la compañia FlyQ es:
avion
Boeing 747     67
Airbus A380    61
```

```
Boeing 737     54
Airbus A320    34
Name: count, dtype: int64
 la distribucion de aviones de uso por tipo de avion para la compañia TabarAir
es:
avion
Boeing 747     77
Airbus A380    71
Boeing 737     62
Airbus A320    61
Name: count, dtype: int64
 la distribucion de aviones de uso por tipo de avion para la compañia MoldaviAir
es:
avion
Boeing 737     84
Airbus A380    74
Boeing 747     71
Airbus A320    35
Name: count, dtype: int64
 la distribucion de aviones de uso por tipo de avion para la compañia PamPangea
es:
avion
Airbus A380    73
Boeing 737     66
Boeing 747     61
Airbus A320    31
Name: count, dtype: int64
```

[10]:
```python
# Cuántos vuelos hay a Ginebra## ESTA ES LA DEL PROFE

condicion = df_aviones ["Destino"] == "Ginebra"
print(len(df_aviones.loc[condicion])) # contamos (usamos la funcion len()) los␣
 ↪aviones con destino a Ginebra(condicion con una serie panda, y no tengo que␣
 ↪poner nada mas, porque solo quiero el numero de filas que
#tiene destino a ginebra




# Cómo se distribuyen los origenes de los vuelos anteriores
# usaremio value_counts (distrobucion)
df_aviones.loc[condicion, "Origen"].value_counts()# con uan serie panda␣
 ↪comprobamos la misma condicion anterior pero ahora, le pedimos tb la columna␣
 ↪Origen, y con el metodo vaule counts tenemos la distribucion
# del numero de vuelos
```

```
163
```

[10]:
```
Origen
Bali           32
Nueva York     29
Londres        21
Cincinnati     17
Los Angeles    17
París          15
Melbourne      10
Barcelona       8
Roma            7
Cádiz           7
Name: count, dtype: int64
```

Muchas de estas consultas tienen un forma alternativa y, a veces, más eficiente de realizarse. Es el caso de usar agrupaciones (groupby) y que veremos al final de la unidad

0.1.2 Manipulación de Valores

Para terminar la sesión, veamos como usar los filtros para cambiar los valores del dataframe de una forma selectiva, lo que nos vendrá muy bien en el futuro para hacer "limpieza".

[19]:
```
es_TabarAir = df_aviones.Aircompany == "TabarAir" # condicion 1
a_los_Angeles = df_aviones .Destino =="Los Angeles" # condicion 2
df_aviones.loc[es_TabarAir & a_los_Angeles, "Destino"] = "San Francisco"# con␣
 ↪uan sere panda comprubo las filas del DF las que cumplen las 2 condciones y␣
 ↪despues de la coma,
#ponermos la columna destino ques es lo que quiero cambiar
# como comprobamos si lo ha cambiado pues preguntado la condicion, si dice que␣
 ↪no hay vuelos se ha cambiado:
df_aviones.loc[es_TabarAir & a_los_Angeles]
# ahora miramos en las 20 filas si se ha cambiado ya que no muetra ninguno al␣
 ↪imprimr la condicion
df_aviones.head(20)
# lo que no hemos cambiado su identificador, podriamos dicho cambiamos en la␣
 ↪misma condicionm cambiar el indice y despues el destino (lo veremos mas␣
 ↪adelante)
```

[19]:

Id_vuelo	Aircompany	Origen	Destino	Distancia
Air_PaGi_10737	Airnar	París	Ginebra	411
Fly_BaRo_10737	FlyQ	Bali	Roma	12738
Tab_GiLo_11380	TabarAir	Ginebra	San Francisco	9103
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480
Mol_CaMe_10737	MoldaviAir	Cádiz	Melbourne	20029
Mol_PaLo_11320	MoldaviAir	París	Londres	344
Pam_PaMe_11380	PamPangea	París	Melbourne	16925
Pam_NuBa_10737	PamPangea	Nueva York	Bali	16589

```
Air_GiCa_11380      Airnar      Ginebra          Cádiz       1725
Tab_LoCi_10737    TabarAir  Los Angeles     Cincinnati       3073
Mol_LoBa_11380  MoldaviAir      Londres           Bali      12553
Tab_CiLo_10747    TabarAir   Cincinnati  San Francisco       3073
Tab_GiLo_11380    TabarAir      Ginebra        Londres        739
Tab_CiRo_11320    TabarAir   Cincinnati           Roma       7480
Tab_RoLo_10747    TabarAir         Roma  San Francisco      10077
Mol_MeCa_10737  MoldaviAir    Melbourne          Cádiz      20029
Air_GiLo_10747      Airnar      Ginebra    Los Angeles       9103
Pam_MePa_10737   PamPangea    Melbourne          París      16925
Pam_GiNu_11320   PamPangea      Ginebra     Nueva York       6206

                     avion     consumo_kg  duracion
Id_vuelo
Air_PaGi_10737   Boeing 737    1028.691900        51
Fly_BaRo_10737   Boeing 737   33479.132544      1167
Tab_GiLo_11380  Airbus A380  109439.907200       626
Mol_PaCi_10737   Boeing 737   17027.010000       503
Tab_CiRo_10747   Boeing 747   86115.744000       518
Mol_CaMe_10737   Boeing 737   53148.153240      1721
Mol_PaLo_11320  Airbus A320     915.246400        44
Pam_PaMe_11380  Airbus A380  217722.658400      1328
Pam_NuBa_10737   Boeing 737   45277.618464      1459
Air_GiCa_11380  Airbus A380   20339.820000       135
Tab_LoCi_10737   Boeing 737    7915.433400       253
Mol_LoBa_11380  Airbus A380  156721.694400       856
Tab_CiLo_10747   Boeing 747   32758.180000       224
Tab_GiLo_11380  Airbus A380    8542.840000        69
Tab_CiRo_11320  Airbus A320   21087.855360       662
Tab_RoLo_10747   Boeing 747  109569.236400       691
Mol_MeCa_10737   Boeing 737   51629.634576      1721
Air_GiLo_10747   Boeing 747  104801.018400       626
Pam_MePa_10737   Boeing 737   46622.417400      1485
Pam_GiNu_11320  Airbus A320   16200.142400       569
```

Supongamos que nos avisan de que los datos de TabarAir son erróneos porque para los vuelos a Los Angeles en realidad son a San Francisco. ¿Cómo podríamos cambiar ese valor?

[ ]:

Y ahora nos dicen que creemos una columna nueva "Supertrayectos" que debe ser True cuando el trayecto es mayor de 12000Km y False en caso contrario

[20]:
```
# Creandolo con un filtro/Selección
# cuanno nos pidan cambiar una columna y despues cambair los valores, nos␣
 ↪crearemos la columna con un valor por defecto , y asi no te de problemas de␣
 ↪referencia  e igularlos a False
```

```
df_aviones["supertrayectos"] = False # condicion que decimos que todos los␣
 ↪valores valen False y despues:
df_aviones.loc[df_aviones.Distancia > 12000, "supertrayectos"] = True # despues␣
 ↪con una serie panda extraemos del DF las filas con distancia >120000, y␣
 ↪despues la columna que quiero cambiar  y el valor lo cambiamos a true
# asi todos los valores seran fdlse menos los que sea mayores a 12000
df_aviones
```

[20]:

Id_vuelo	Aircompany	Origen	Destino	Distancia	avion	consumo_kg	duracion	supertrayectos
Air_PaGi_10737	Airnar	París	Ginebra	411	Boeing 737	1028.691900	51	False
Fly_BaRo_10737	FlyQ	Bali	Roma	12738	Boeing 737	33479.132544	1167	True
Tab_GiLo_11380	TabarAir	Ginebra	San Francisco	9103	Airbus A380	109439.907200	626	False
Mol_PaCi_10737	MoldaviAir	París	Cincinnati	6370	Boeing 737	17027.010000	503	False
Tab_CiRo_10747	TabarAir	Cincinnati	Roma	7480	Boeing 747	86115.744000	518	False
…	…	…	…	…	…	…	…	…
Tab_LoLo_11320	TabarAir	Los Angeles	Londres	8785	Airbus A320	24766.953120	756	False
Mol_CiLo_10737	MoldaviAir	Cincinnati	Londres	6284	Boeing 737	16491.729600	497	False
Fly_RoCi_11320	FlyQ	Roma	Cincinnati	7480	Airbus A320	19721.049920	662	False
Tab_RoLo_10747	TabarAir	Roma	Londres	1433	Boeing 747	15734.053400	115	False
Air_PaLo_10737	Airnar	París	Los Angeles	9099	Boeing 737	22331.675700	711	False

[1200 rows x 8 columns]

[25]:
```
# pero despue spodemos hacer
df_aviones.loc[df_aviones.supertrayectos, "Destino"].value_counts()#␣
 ↪comprobamos en el DF con serie panda, la fila nueva creada, y recupero␣
 ↪"Destino" , pq quier saber la distrbucion de destinos
# me da el numero de destinos que han recorrido mas de 12000 km
```

[25]:
```
Destino
Bali           114
Melbourne       68
```

```
Londres         47
Ginebra         42
Cincinnati      36
Cádiz           22
París           19
Nueva York      18
Roma            16
Barcelona        7
Los Angeles      3
Name: count, dtype: int64
```

[26]:
```
# Versión alternativa(la de arriba mejor)

df_aviones["supertrayectos"] = df_aviones["Distancia"] >12000 # directamente␣
 ↪creo la columna de supertrayectos a partir de la condicion (> 120000km)
df_aviones
```

[26]:
```
                Aircompany       Origen        Destino  Distancia  \
Id_vuelo
Air_PaGi_10737      Airnar        París        Ginebra        411
Fly_BaRo_10737        FlyQ         Bali           Roma      12738
Tab_GiLo_11380    TabarAir      Ginebra  San Francisco       9103
Mol_PaCi_10737  MoldaviAir        París     Cincinnati       6370
Tab_CiRo_10747    TabarAir   Cincinnati           Roma       7480
…                     …          …              …        …
Tab_LoLo_11320    TabarAir  Los Angeles        Londres       8785
Mol_CiLo_10737  MoldaviAir   Cincinnati        Londres       6284
Fly_RoCi_11320        FlyQ         Roma     Cincinnati       7480
Tab_RoLo_10747    TabarAir         Roma        Londres       1433
Air_PaLo_10737      Airnar        París    Los Angeles       9099

                      avion     consumo_kg  duracion  supertrayectos
Id_vuelo
Air_PaGi_10737   Boeing 737    1028.691900        51           False
Fly_BaRo_10737   Boeing 737   33479.132544      1167            True
Tab_GiLo_11380  Airbus A380  109439.907200       626           False
Mol_PaCi_10737   Boeing 737   17027.010000       503           False
Tab_CiRo_10747   Boeing 747   86115.744000       518           False
…                      …           …       …              …
Tab_LoLo_11320  Airbus A320   24766.953120       756           False
Mol_CiLo_10737   Boeing 737   16491.729600       497           False
Fly_RoCi_11320  Airbus A320   19721.049920       662           False
Tab_RoLo_10747   Boeing 747   15734.053400       115           False
Air_PaLo_10737   Boeing 737   22331.675700       711           False

[1200 rows x 8 columns]
```

En sesiones posteriores veremos la forma de manipular valores utilizando funciones definidas por el

usuario, con lo que terminarás de ver ya casi todo el potencial de manipulación de datos a partir de un `DataFrame`.