Pandas

diciembre 2022

Pandas es una librería de Python especializada en el **manejo y análisis de estructura de datos**, el nombre pandas viene de "**Panel data**". Se caracteriza por su:

- Velocidad
- Poco código
- Múltiples formatos de archivos
- Alineación inteligente

La forma común de importarla es

```
import pandas as pd
```

Series y DataFrames

Series

Son arreglos **unidimensionales** indexados

```
psg_players = pd.Series(['Navas','Mbappe','Neymar','Messi'], index=[1,7,10,30])
psg_players
```

Series con index personalizados

```
## 1      Navas
## 7     Mbappe
## 10    Neymar
## 30     Messi
## dtype: object
```

```
psg_players = pd.Series(['Navas','Mbappe','Neymar','Messi'])
psg_players
```

Series con index default

```
## 0     Navas
## 1    Mbappe
## 2    Neymar
## 3     Messi
## dtype: object
```

```
dictionary = {1: 'Navas', 7:'Mbappe', 10:'Neymar', 30:'Messi'}
pd.Series(dictionary)
```

Series con diccionario JSON

```
## 1      Navas
## 7     Mbappe
## 10    Neymar
## 30     Messi
## dtype: object
```

Slicing

Podemos acceder a los datos de la misma forma que lo hacemos con NumPy

```
psg_players[0:3]
```

```
## 0     Navas
## 1    Mbappe
## 2    Neymar
## dtype: object
```

DataFrame matricial

Podemos trabajar con estructuras matriciales

```
dictionary = {'jugador':['Navas','Mbappe','Neymar','Messi'],
'altura':[183.0, 170.0, 170.0, 165.0],
'goles':[2, 200, 200, 200]
}
dictionary
```

```
## {'jugador': ['Navas', 'Mbappe', 'Neymar', 'Messi'], 'altura': [183.0, 170.0, 170.0, 165.0], 'goles':
```

```
pd.DataFrame(dictionary, index=[1,7,10,30])
```

DataFrame con index personalizado

```
##    jugador  altura  goles
## 1    Navas   183.0      2
## 7   Mbappe   170.0    200
## 10  Neymar   170.0    200
## 30   Messi   165.0    200
```

```
pd.DataFrame(dictionary)
```

DataFrame con index default

```
##   jugador  altura  goles
## 0   Navas   183.0      2
## 1  Mbappe   170.0    200
## 2  Neymar   170.0    200
## 3   Messi   165.0    200
```

Visualizar ejes en DataFrame

```
df_Players = pd.DataFrame(dictionary)
df_Players
```

```
##   jugador  altura  goles
```

```
## 0   Navas   183.0      2
## 1  Mbappe   170.0    200
## 2  Neymar   170.0    200
## 3   Messi   165.0    200
```

```
df_Players.columns
```

Visualizar Data por columnas

```
## Index(['jugador', 'altura', 'goles'], dtype='object')
```

Visualizar Data por indices

```
df_Players.index
```

Indice default

```
## RangeIndex(start=0, stop=4, step=1)
```

```
df_Players = pd.DataFrame(dictionary, index=[1,7,10,30])
df_Players.index
```

Indice personalizado

```
## Int64Index([1, 7, 10, 30], dtype='int64')
```

Leer archivos CSV y JSON

Podemos cargar archivos a través de URL, JSON, HTML, etc.

CSV

```
csv_per_default = pd.read_csv('./Data/bestsellers-with-categories.csv')
csv_per_default
```

```
##                                                  Name  ...         Genre
## 0                       10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                   11/22/63: A Novel  ...       Fiction
## 2               12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                                1984 (Signet Classics)  ...       Fiction
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## ..                                                 ...  ...          ...
## 545       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...       Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

Cambiar el encabezado Lo podemos hacer con "**Header**", este pondrá de encabezado los valores que tenga en esa posición.

```
pd.read_csv('./Data/bestsellers-with-categories.csv', sep=",", header= 2)
```

```
##                                      11/22/63: A Novel  ...      Fiction
## 0              12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 1                                1984 (Signet Classics)  ...      Fiction
## 2    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## 3        A Dance with Dragons (A Song of Ice and Fire)  ...      Fiction
## 4    A Game of Thrones / A Clash of Kings / A Storm...  ...      Fiction
## ..                                                 ...  ...          ...
## 543       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...      Fiction
## 544  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 545  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [548 rows x 7 columns]
```

Cambiar el nombre de las columnas Lo podemos hacer con "**names**"

```
df_books = pd.read_csv(
    './Data/bestsellers-with-categories.csv',
    sep=",",
    header=0
)
df_books.columns
```

```
## Index(['Name', 'Author', 'User Rating', 'Reviews', 'Price', 'Year', 'Genre'], dtype='object')
```

```
df_books = pd.read_csv(
    './Data/bestsellers-with-categories.csv',
    sep=",",
    header=0,
    names=['Nombre', 'Autor', 'Valoracion', 'Reseñas', 'Precio', 'Año', 'Genero']
)
df_books
```

```
##                                               Nombre  ...       Genero
## 0                       10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                  11/22/63: A Novel  ...      Fiction
## 2              12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                                1984 (Signet Classics)  ...      Fiction
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## ..                                                 ...  ...          ...
## 545       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...      Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

JSON

Podemos cargar el archivo JSON como un DataFrame por default, u otra forma de llevar la estructura JSON a una estructura de Pandas como una Series

```
json_per_default = pd.read_json('./Data/hpcharactersdataraw.json')
json_per_default
```

```
##                       Name  ...                                      Profession
## 0              Mrs. Abbott  ...                                         Unknown
## 1            Hannah Abbott  ...                 Landlady of the Leaky Cauldron
## 2            Abel Treetops  ...                                         Unknown
## 3         Euan Abercrombie  ...                                         Unknown
## 4     Aberforth Dumbledore  ...                                          Barman
## ...                    ...  ...                                             ...
## 1935        Georgi Zdravko  ...                      Quidditch player (Seeker)
## 1936                Zograf  ...                      Quidditch player (Keeper)
## 1937                 Zonko  ...                                         Unknown
## 1938     Valentina Vázquez  ...  President of the Argentinian Council of Magic
## 1939         Zygmunt Budge  ...    Potions Master, Potioneer, Inventor, Author
##
## [1940 rows x 8 columns]
```

podemos indexar una lista en raw del formato llave valor del JSON

```
pd.read_json('./Data/hpcharactersdataraw.json',typ='Series')
```

```
## 0       {'Name': 'Mrs. Abbott', 'Link': 'https://www.h...
## 1       {'Name': 'Hannah Abbott', 'Link': 'https://www...
## 2       {'Name': 'Abel Treetops', 'Link': 'https://www...
## 3       {'Name': 'Euan Abercrombie', 'Link': 'https://...
## 4       {'Name': 'Aberforth Dumbledore', 'Link': 'http...
##                                ...
## 1935    {'Name': 'Georgi Zdravko', 'Link': 'https://ww...
## 1936    {'Name': 'Zograf', 'Link': 'https://www.hp-lex...
## 1937    {'Name': 'Zonko', 'Link': 'https://www.hp-lexi...
## 1938    {'Name': 'Valentina Vázquez', 'Link': 'https:/...
## 1939    {'Name': 'Zygmunt Budge', 'Link': 'https://www...
## Length: 1940, dtype: object
```

Filtrado con loc y iloc

Cuando queremos navegar por un dataFrame estas funciones permiten filtrar datos de manera más específica

```
df_books = pd.read_csv('./data/bestsellers-with-categories.csv')
df_books
```

```
##                                                  Name  ...        Genre
## 0                      10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                 11/22/63: A Novel  ...      Fiction
## 2           12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                             1984 (Signet Classics)  ...      Fiction
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## ..                                                ...  ...          ...
## 545      Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...      Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

```
df_books[0:4]
```

```
##                                        Name  ...        Genre
## 0            10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                       11/22/63: A Novel  ...      Fiction
## 2  12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                   1984 (Signet Classics)  ...      Fiction
##
## [4 rows x 7 columns]
```

Filtrado por columnas

Filtrar una columna

```
df_books['Name']
```

```
## 0                          10-Day Green Smoothie Cleanse
## 1                                    11/22/63: A Novel
## 2                12 Rules for Life: An Antidote to Chaos
## 3                                  1984 (Signet Classics)
## 4      5,000 Awesome Facts (About Everything!) (Natio...
##                               ...
## 545          Wrecking Ball (Diary of a Wimpy Kid Book 14)
## 546    You Are a Badass: How to Stop Doubting Your Gr...
## 547    You Are a Badass: How to Stop Doubting Your Gr...
## 548    You Are a Badass: How to Stop Doubting Your Gr...
## 549    You Are a Badass: How to Stop Doubting Your Gr...
## Name: Name, Length: 550, dtype: object
```

Filtrar dos o mas columnas

```
df_books[['Name','Author','Year']]
```

```
##                                                  Name  ...  Year
## 0                       10-Day Green Smoothie Cleanse  ...  2016
## 1                                  11/22/63: A Novel  ...  2011
## 2              12 Rules for Life: An Antidote to Chaos  ...  2018
## 3                                1984 (Signet Classics)  ...  2017
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  2019
## ..                                                   ...  ...   ...
## 545       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...  2019
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  2016
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  2017
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  2018
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  2019
##
## [550 rows x 3 columns]
```

Filtrado con loc

Acceda a un grupo de filas y columnas por etiqueta(s) o una matriz booleana.

A diferencia del el slicing normal, que corta por la posición del index con dataframe. loc tienes la posición de start y stop para filtrar.

```
df_books.loc[0:4]
```

Filtrar con slicing

```
##                                                 Name  ...        Genre
## 0                       10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                    11/22/63: A Novel  ...      Fiction
## 2              12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                                1984 (Signet Classics)  ...      Fiction
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
##
## [5 rows x 7 columns]
```

```
df_books.loc[0:4, ['Name','User Rating','Reviews']]
```

Filtrar varios tags

```
##                                                 Name  User Rating  Reviews
## 0                       10-Day Green Smoothie Cleanse          4.7    17350
## 1                                    11/22/63: A Novel          4.6     2052
## 2              12 Rules for Life: An Antidote to Chaos          4.7    18979
## 3                                1984 (Signet Classics)          4.7    21424
## 4  5,000 Awesome Facts (About Everything!) (Natio...          4.8     7665
```

```
df_books.loc[0:4, ['User Rating']] * 20
```

Filtrar y modificar

```
##    User Rating
## 0         94.0
## 1         92.0
## 2         94.0
## 3         94.0
## 4         96.0
```

Filtrar por condición

Devuelve una lista booleana de quien cumple la condición

```
df_books.loc[:, ['Author']] == 'Jen Sincero '
```

```
##      Author
## 0     False
## 1     False
## 2     False
## 3     False
## 4     False
## ..      ...
## 545   False
## 546   False
## 547   False
## 548   False
## 549   False
##
## [550 rows x 1 columns]
```

Filtrado con iloc

Indexación puramente basada en la ubicación de enteros para la selección por posición.

```
df_books.iloc[:]
```

Filtrar con Slicing

```
##                                                  Name  ...         Genre
## 0                       10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                   11/22/63: A Novel  ...       Fiction
## 2             12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                               1984 (Signet Classics)  ...       Fiction
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## ..                                                 ...  ...           ...
## 545      Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...       Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

```
df_books.iloc[:,0:3]
```

Filtrar con Indice

```
##                                                  Name  ... User Rating
## 0                       10-Day Green Smoothie Cleanse  ...         4.7
## 1                                   11/22/63: A Novel  ...         4.6
## 2             12 Rules for Life: An Antidote to Chaos  ...         4.7
## 3                               1984 (Signet Classics)  ...         4.7
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...         4.8
## ..                                                 ...  ...         ...
## 545      Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...         4.9
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...         4.7
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...         4.7
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...         4.7
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...         4.7
##
## [550 rows x 3 columns]
```

```
df_books.iloc[1,3]
```

Buscar un dato especifico

```
## 2052
```

Agregar o eliminar datos

Partimos con nuestro dataFrame

```
df_books = pd.read_csv('./data/bestsellers-with-categories.csv')
df_books.head(2)
```

```
##                             Name        Author  ...  Year        Genre
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...  2016  Non Fiction
## 1              11/22/63: A Novel  Stephen King  ...  2011      Fiction
##
## [2 rows x 7 columns]
```

Eliminar columnas

```
df_books.drop('Genre',axis=1).head(2)
```

Sin modificar el DataFrame

```
##                             Name        Author  ...  Price  Year
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...      8  2016
## 1              11/22/63: A Novel  Stephen King  ...     22  2011
##
## [2 rows x 6 columns]
```

```
df_books.head(2)
```

```
##                             Name        Author  ...  Year        Genre
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...  2016  Non Fiction
## 1              11/22/63: A Novel  Stephen King  ...  2011      Fiction
##
## [2 rows x 7 columns]
```

Remplazando el DataFame original

```
df_books.drop('Genre', axis=1, inplace=True)
df_books.head(2)
```

utilizando `inplace`

```
##                             Name        Author  ...  Price  Year
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...      8  2016
## 1              11/22/63: A Novel  Stephen King  ...     22  2011
##
## [2 rows x 6 columns]
```

```
df_books = df_books.drop('Year',axis=1)
df_books.head(2)
```

Cambiando el valor de referencia en memoria

```
##                             Name        Author  User Rating  Reviews  Price
## 0  10-Day Green Smoothie Cleanse      JJ Smith          4.7    17350      8
## 1              11/22/63: A Novel  Stephen King          4.6     2052     22
```

```
del df_books['Price']
df_books.head(2)
```

Usando Python Vainilla

```
##                              Name        Author  User Rating  Reviews
## 0  10-Day Green Smoothie Cleanse      JJ Smith          4.7    17350
## 1              11/22/63: A Novel  Stephen King          4.6     2052
```

Eliminar filas

Podemos eliminar filas de la misma manera que con las columnas simplemente cambiando el valor del axis

```
df_books.drop(0,axis=0).head(2)
```

```
##                                     Name  ... Reviews
## 1                        11/22/63: A Novel  ...    2052
## 2  12 Rules for Life: An Antidote to Chaos  ...   18979
##
## [2 rows x 4 columns]
```

Eliminar una lista de filas

```
df_books.drop([0,1,2],axis=0).head(2)
```

Utilizando un array

```
##                                                 Name  ... Reviews
## 3                            1984 (Signet Classics)  ...   21424
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...    7665
##
## [2 rows x 4 columns]
```

```
df_books.drop(range(0,10), axis=0).head(2)
```

Utilizando `range()`

```
##                                                  Name  ... Reviews
## 10                         A Man Called Ove: A Novel  ...   23848
## 11  A Patriot's History of the United States: From...  ...     460
##
## [2 rows x 4 columns]
```

Agregar columnas

Mostremos primero nuestro DataFrame e importamos NumPy para asignar el valor nulo

```
import numpy as np

df_books.head(2)
```

```
##                              Name        Author  User Rating  Reviews
## 0  10-Day Green Smoothie Cleanse      JJ Smith          4.7    17350
## 1              11/22/63: A Novel  Stephen King          4.6     2052
```

```
df_books['Nueva Columna'] = np.nan
df_books.head(2)
```

Agregar una columna con valores nulos

```
##                              Name        Author  ...  Reviews  Nueva Columna
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...    17350            NaN
## 1              11/22/63: A Novel  Stephen King  ...     2052            NaN
##
## [2 rows x 5 columns]
```

Asignando datos a una columna Creamos un array de datos con el numero de elementos de nuestro DataFrame

```
data = np.arange(0, df_books.shape[0])
#data
```

Asignamos estos valores a una nueva columna

> *Una regla es que todo rango que vaya a asignar al DataFrame debe tener la misma longitud en columnas y filas para que no haya error*

```
df_books['Rango'] = data
df_books.head(2)
```

```
##                              Name        Author  ...  Nueva Columna  Rango
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...            NaN      0
## 1              11/22/63: A Novel  Stephen King  ...            NaN      1
##
## [2 rows x 6 columns]
```

Agregar filas

Para añadir filas se utiliza la función `append` de Python añadiendo como parámetro una lista, diccionario o añadiendo los valores manualmente.

Podemos por ejemplo duplicar los datos de nuestro DataFrame

```
df_books.append(df_books)
```

```
##                                                  Name  ... Rango
## 0                       10-Day Green Smoothie Cleanse  ...     0
## 1                                   11/22/63: A Novel  ...     1
## 2             12 Rules for Life: An Antidote to Chaos  ...     2
## 3                               1984 (Signet Classics)  ...     3
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...     4
## ..                                                 ...  ...   ...
## 545      Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...   545
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...   546
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...   547
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...   548
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...   549
##
## [1100 rows x 6 columns]
##
## <string>:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a
```

Manejo de datos nulos

En muchas ocasiones nos encotraremos con datos nulos, podemos entonces trabajar con ellos

Partiendo de el suguiente ejemplo de DataFrame

```
dict = {'Col1':[1,2,3,np.nan],
'Col2':[4, np.nan,6,7],
'Col3':['a','b','c', None]}

df = pd.DataFrame(dict)
df
```

```
##    Col1  Col2  Col3
## 0   1.0   4.0     a
## 1   2.0   NaN     b
## 2   3.0   6.0     c
## 3   NaN   7.0  None
```

Identificar valores nulos en un DataFrame

```
df.isnull()
```

forma booleana

```
##     Col1   Col2   Col3
## 0  False  False  False
## 1  False   True  False
## 2  False  False  False
## 3   True  False   True
```

```
df.isnull() * 1
```

Forma numerica

```
##    Col1  Col2  Col3
## 0     0     0     0
## 1     0     1     0
## 2     0     0     0
## 3     1     0     1
```

Sustituir los valores nulos

```
df.fillna('Missing')
```

Por una cadena

```
##       Col1     Col2     Col3
## 0      1.0      4.0        a
## 1      2.0  Missing        b
## 2      3.0      6.0        c
## 3  Missing      7.0  Missing
```

```
df.fillna(df.mean())
```

Por una medida estadística realizada con los valores de las columnas

```
##    Col1      Col2  Col3
## 0   1.0  4.000000     a
## 1   2.0  5.666667     b
```

```
## 2   3.0  6.000000     c
## 3   2.0  7.000000  None
##
## <string>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a fu
```

```
df.interpolate()
```

por valores de interpolación

```
##    Col1  Col2  Col3
## 0   1.0   4.0     a
## 1   2.0   5.0     b
## 2   3.0   6.0     c
## 3   3.0   7.0  None
```

Eliminar datos nulos

```
df.dropna()
```

```
##    Col1  Col2 Col3
## 0   1.0   4.0    a
## 2   3.0   6.0    c
```

Filtrado por conciciones

Dado nuestro DataFrame

```
df_books = pd.read_csv('./data/bestsellers-with-categories.csv')
df_books.head(2)
```

```
##                             Name        Author  ...  Year        Genre
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...  2016  Non Fiction
## 1              11/22/63: A Novel  Stephen King  ...  2011      Fiction
##
## [2 rows x 7 columns]
```

Mostrar datos mayores a cierto valor

mostramos un DataFrame booleano con los datos que cumplen la condicion

```
mayor_a_2016 = df_books['Year'] > 2016
df_books[mayor_a_2016].head(4)
```

```
##                                                Name  ...        Genre
## 2            12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                              1984 (Signet Classics)  ...      Fiction
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## 7                      A Gentleman in Moscow: A Novel  ...      Fiction
##
## [4 rows x 7 columns]
```

Podemos directamente colocar la condición como parámetro

```
df_books[df_books['Year'] > 2016].head(4)
```

```
##                                                Name  ...        Genre
## 2            12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
```

```
## 3                              1984 (Signet Classics)  ...      Fiction
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## 7                     A Gentleman in Moscow: A Novel  ...      Fiction
##
## [4 rows x 7 columns]
```

Mostrar datos que coincidan con cierto valor

```
genre_fiction = df_books['Genre'] == "Fiction"
df_books[genre_fiction].head(4)
```

```
##                                                 Name  ...    Genre
## 1                                 11/22/63: A Novel  ...  Fiction
## 3                              1984 (Signet Classics)  ...  Fiction
## 5      A Dance with Dragons (A Song of Ice and Fire)  ...  Fiction
## 6  A Game of Thrones / A Clash of Kings / A Storm...  ...  Fiction
##
## [4 rows x 7 columns]
```

Multiples condiciones

```
df_books[genre_fiction & mayor_a_2016]
```

```
##                                                 Name  ...    Genre
## 3                              1984 (Signet Classics)  ...  Fiction
## 7                     A Gentleman in Moscow: A Novel  ...  Fiction
## 10                         A Man Called Ove: A Novel  ...  Fiction
## 13                  A Wrinkle in Time (Time Quintet)  ...  Fiction
## 40           Brown Bear, Brown Bear, What Do You See?  ...  Fiction
## ..                                               ...  ...      ...
## 509                          To Kill a Mockingbird  ...  Fiction
## 529  What Should Danny Do? (The Power to Choose Ser...  ...  Fiction
## 534                          Where the Crawdads Sing  ...  Fiction
## 544                                           Wonder  ...  Fiction
## 545       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...  Fiction
##
## [65 rows x 7 columns]
```

Filtrado con negacion

```
df_books[~mayor_a_2016]
```

```
##                                                 Name  ...        Genre
## 0                       10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                 11/22/63: A Novel  ...      Fiction
## 5      A Dance with Dragons (A Song of Ice and Fire)  ...      Fiction
## 6  A Game of Thrones / A Clash of Kings / A Storm...  ...      Fiction
## 9                          A Man Called Ove: A Novel  ...      Fiction
## ..                                               ...  ...          ...
## 540                                           Wonder  ...      Fiction
## 541                                           Wonder  ...      Fiction
## 542                                           Wonder  ...      Fiction
## 543                                           Wonder  ...      Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
```

```
##
## [400 rows x 7 columns]
```

Filtrado contenga una palabra especifica en una cadena de texto

```
smith_surname = df_books['Author'].str.contains('Mel')
df_books[smith_surname]
```

```
##                                                  Name  ...        Genre
## 483  The Whole30: The 30-Day Guide to Total Health ...  ...  Non Fiction
## 484  The Whole30: The 30-Day Guide to Total Health ...  ...  Non Fiction
## 485  The Whole30: The 30-Day Guide to Total Health ...  ...  Non Fiction
##
## [3 rows x 7 columns]
```

Funciones principales

`.info()`

Mostrar información acerca de los datos que contiene el DataFrame

```
df_books.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 550 entries, 0 to 549
## Data columns (total 7 columns):
##  #   Column       Non-Null Count  Dtype
## ---  ------       --------------  -----
##  0   Name         550 non-null    object
##  1   Author       550 non-null    object
##  2   User Rating  550 non-null    float64
##  3   Reviews      550 non-null    int64
##  4   Price        550 non-null    int64
##  5   Year         550 non-null    int64
##  6   Genre        550 non-null    object
## dtypes: float64(1), int64(3), object(3)
## memory usage: 30.2+ KB
```

`.describe()`

Obtener diferentes datos estadísticos de las columnas numéricas

```
df_books.describe()
```

```
##        User Rating       Reviews       Price         Year
## count   550.000000    550.000000  550.000000   550.000000
## mean      4.618364  11953.281818   13.100000  2014.000000
## std       0.226980  11731.132017   10.842262     3.165156
## min       3.300000     37.000000    0.000000  2009.000000
## 25%       4.500000   4058.000000    7.000000  2011.000000
## 50%       4.700000   8580.000000   11.000000  2014.000000
## 75%       4.800000  17253.250000   16.000000  2017.000000
## max       4.900000  87841.000000  105.000000  2019.000000
```

`.tail()`

Mostrar los últimos 5 registros del DataFrame

```
df_books.tail()
```

```
##                                                  Name  ...        Genre
## 545       Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...      Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [5 rows x 7 columns]
```

.memory_usage()

Obtener el uso de la memoria de cada columna

```
df_books.memory_usage(deep=True)
```

```
## Index           128
## Name          59737
## Author        39078
## User Rating    4400
## Reviews        4400
## Price          4400
## Year           4400
## Genre         36440
## dtype: int64
```

.value_counts()

Obtener *cuantos datos* tenemos de algo en específico

```
df_books['Author'].value_counts()
```

```
## Jeff Kinney                          12
## Gary Chapman                         11
## Rick Riordan                         11
## Suzanne Collins                      11
## American Psychological Association   10
##                                       ..
## Keith Richards                        1
## Chris Cleave                          1
## Alice Schertle                        1
## Celeste Ng                            1
## Adam Gasiewski                        1
## Name: Author, Length: 248, dtype: int64
```

.drop_duplicates()

Eliminar registros duplicados

```
df_books.drop_duplicates()
```

```
##                                                  Name  ...        Genre
## 0                      10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                 11/22/63: A Novel  ...      Fiction
## 2           12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                           1984 (Signet Classics)  ...      Fiction
```

```
## 4    5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
## ..                                                   ...  ...          ...
## 545      Wrecking Ball (Diary of a Wimpy Kid Book 14)  ...      Fiction
## 546  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 547  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 548  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

`.sort_values()`

Ordenar los registros según valores de la columna

```
df_books.sort_values('Year')
```

Orden ascendente

```
##                                                  Name  ...        Genre
## 177                                     I, Alex Cross  ...      Fiction
## 131  Glenn Beck's Common Sense: The Case Against an...  ...  Non Fiction
## 417                                 The Last Lecture  ...  Non Fiction
## 241                      New Moon (The Twilight Saga)  ...      Fiction
## 72       Diary of a Wimpy Kid: The Last Straw (Book 3)  ...      Fiction
## ..                                                   ...  ...          ...
## 150                                             Guts  ...  Non Fiction
## 466  The Subtle Art of Not Giving a F*ck: A Counter...  ...  Non Fiction
## 462                                The Silent Patient  ...      Fiction
## 130  Girl, Wash Your Face: Stop Believing the Lies ...  ...  Non Fiction
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
##
## [550 rows x 7 columns]
```

```
df_books.sort_values('Year', ascending=False)
```

Orden descendente

```
##                                                  Name  ...        Genre
## 549  You Are a Badass: How to Stop Doubting Your Gr...  ...  Non Fiction
## 294  School Zone - Big Preschool Workbook - Ages 4 ...  ...  Non Fiction
## 489                  The Wonderful Things You Will Be  ...      Fiction
## 263    P is for Potty! (Sesame Street) (Lift-the-Flap)  ...  Non Fiction
## 130  Girl, Wash Your Face: Stop Believing the Lies ...  ...  Non Fiction
## ..                                                   ...  ...          ...
## 418  The Last Olympian (Percy Jackson and the Olymp...  ...      Fiction
## 38          Breaking Dawn (The Twilight Saga, Book 4)  ...      Fiction
## 92   Eat This, Not That! Thousands of Simple Food S...  ...  Non Fiction
## 139  Good to Great: Why Some Companies Make the Lea...  ...  Non Fiction
## 299                                Sookie Stackhouse  ...      Fiction
##
## [550 rows x 7 columns]
```

Groupby

Permite agrupar datos en función de los demás. Es decir, hacer el análisis del DataFrame en función de una de las columnas.

`.count()`

Agrupar por **Author** y mostrar el *conteo* de los datos de las demás columnas

```
df_books.groupby('Author').count()
```

```
##                            Name  User Rating  Reviews  Price  Year  Genre
## Author
## Abraham Verghese              2            2        2      2     2      2
## Adam Gasiewski                1            1        1      1     1      1
## Adam Mansbach                 1            1        1      1     1      1
## Adir Levy                     1            1        1      1     1      1
## Admiral William H. McRaven    1            1        1      1     1      1
## ...                         ...          ...      ...    ...   ...    ...
## Walter Isaacson               3            3        3      3     3      3
## William Davis                 2            2        2      2     2      2
## William P. Young              2            2        2      2     2      2
## Wizards RPG Team              3            3        3      3     3      3
## Zhi Gang Sha                  2            2        2      2     2      2
##
## [248 rows x 6 columns]
```

Agrupar por **'Author - Year'** y contar los valores de las demás columnas

```
df_books.groupby(['Author','Year']).count()
```

```
##                        Name  User Rating  Reviews  Price  Genre
## Author           Year
## Abraham Verghese 2010     1            1        1      1      1
##                  2011     1            1        1      1      1
## Adam Gasiewski   2017     1            1        1      1      1
## Adam Mansbach    2011     1            1        1      1      1
## Adir Levy        2019     1            1        1      1      1
## ...                       ...          ...      ...    ...    ...
## Wizards RPG Team 2017     1            1        1      1      1
##                  2018     1            1        1      1      1
##                  2019     1            1        1      1      1
## Zhi Gang Sha     2009     1            1        1      1      1
##                  2013     1            1        1      1      1
##
## [493 rows x 5 columns]
```

`.median()`

Agrupar por **Author** y mostrar la *media* de los datos de las demás columnas

```
df_books.groupby('Author').median()
```

```
##                            User Rating  Reviews  Price    Year
## Author
## Abraham Verghese                   4.6   4866.0   11.0  2010.5
## Adam Gasiewski                     4.4   3113.0    6.0  2017.0
## Adam Mansbach                      4.8   9568.0    9.0  2011.0
```

```
## Adir Levy                          4.8   8170.0   13.0  2019.0
## Admiral William H. McRaven         4.7  10199.0   11.0  2017.0
## ...                                ...      ...    ...     ...
## Walter Isaacson                    4.6   7827.0   20.0  2012.0
## William Davis                      4.4   7497.0    6.0  2012.5
## William P. Young                   4.6  19720.0    8.0  2013.0
## Wizards RPG Team                   4.8  16990.0   27.0  2018.0
## Zhi Gang Sha                       4.6    128.5   11.5  2011.0
##
## [248 rows x 4 columns]
##
## <string>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.median is deprecate
```

La columna **Author**, en los casos anteriores, pasa a ser el índice.

`.loc()`

Podemos usar `loc()` y acceder a un dato específico del DataFrame. Agrupar por **Author** y mostrar la **suma** de los valores de las demás columnas para *William Davis*

```
df_books.groupby('Author').sum().loc['William Davis']
```

```
## User Rating        8.8
## Reviews        14994.0
## Price             12.0
## Year            4025.0
## Name: William Davis, dtype: float64
##
## <string>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
```

`.reset_index()`

Agrupar por **Author** y mostrar la **suma** de los valores de las demás columnas. Colocar los *índices* que el DataFrame trae por defecto

```
df_books.groupby('Author').sum().reset_index()
```

```
##                          Author  User Rating  Reviews  Price  Year
## 0              Abraham Verghese          9.2     9732     22  4021
## 1                Adam Gasiewski          4.4     3113      6  2017
## 2                 Adam Mansbach          4.8     9568      9  2011
## 3                     Adir Levy          4.8     8170     13  2019
## 4    Admiral William H. McRaven          4.7    10199     11  2017
## ..                           ...          ...      ...    ...   ...
## 243             Walter Isaacson         13.7    18668     61  6040
## 244               William Davis          8.8    14994     12  4025
## 245            William P. Young          9.2    39440     16  4026
## 246            Wizards RPG Team         14.4    50970     81  6054
## 247                Zhi Gang Sha          9.2      257     23  4022
##
## [248 rows x 5 columns]
##
## <string>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
```

`.agg()`

Permite aplicar varias funciones al DataFrame una vez agrupado según una columna específica.

Agrupar por **Author** y mostrar el *mínimo* y *máximo* de las demás columnas

```
df_books.groupby('Author').agg(['min','max'])
```

```
##                                                                 Name  ...        Genre
##                                                                  min  ...          max
## Author                                                                ...
## Abraham Verghese                                    Cutting for Stone  ...      Fiction
## Adam Gasiewski              Milk and Vine: Inspirational Quotes From Class...  ...  Non Fiction
## Adam Mansbach                                     Go the F**k to Sleep  ...      Fiction
## Adir Levy                   What Should Danny Do? (The Power to Choose Ser...  ...      Fiction
## Admiral William H. McRaven  Make Your Bed: Little Things That Can Change Y...  ...  Non Fiction
## ...                                                               ...  ...          ...
## Walter Isaacson                                     Leonardo da Vinci  ...  Non Fiction
## William Davis               Wheat Belly: Lose the Wheat, Lose the Weight, ...  ...  Non Fiction
## William P. Young                  The Shack: Where Tragedy Confronts Eternity  ...      Fiction
## Wizards RPG Team                       Player's Handbook (Dungeons & Dragons)  ...      Fiction
## Zhi Gang Sha                Divine Soul Mind Body Healing and Transmission...  ...  Non Fiction
##
## [248 rows x 12 columns]
```

Agrupar por **Author**, obtener el *mínimo* y *máximo* de la columna **'Reviews'** y sumar los valores de la columna **'User Rating'**

```
df_books.groupby('Author').agg({'Reviews':['min','max'], 'User Rating':'sum'})
```

```
##                            Reviews        User Rating
##                                min    max         sum
## Author
## Abraham Verghese              4866   4866         9.2
## Adam Gasiewski                3113   3113         4.4
## Adam Mansbach                 9568   9568         4.8
## Adir Levy                     8170   8170         4.8
## Admiral William H. McRaven   10199  10199         4.7
## ...                            ...    ...         ...
## Walter Isaacson               3014   7827        13.7
## William Davis                 7497   7497         8.8
## William P. Young             19720  19720         9.2
## Wizards RPG Team             16990  16990        14.4
## Zhi Gang Sha                    37    220         9.2
##
## [248 rows x 3 columns]
```

Combinando DataFrames

Existen diferentes formas de fusionar dos DataFrames. Esto se hace a través de la lógica de combinación

Left join

Da prioridad al DataFrame de la izquierda. Trae siempre los datos de la izquierda y las filas en común con el DataFrame de la derecha.

Right join

Da prioridad al DataFrame de la derecha. Trae siempre los datos de la derecha y las filas en común con el DataFrame de la izquierda.

Inner join

Trae solamente aquellos datos que son común en ambos DataFrame

Outer join

Trae los datos tanto del DataFrame de la izquierda como el de la derecha, incluyendo los datos que comparten ambos.

Merge y concat

concat

Permite combinar dos DataFrames a nivel de filas. Crecimiento vertical

```
df1 = pd.DataFrame({'A':['A0', 'A1', 'A2','A3'],
        'B':['B0', 'B1', 'B2','B3'],
    'C':['C0', 'C1', 'C2','C3'],
    'D':['D0', 'D1', 'D2','D3']})


df2 = pd.DataFrame({'A':['A4', 'A5', 'A6','A7'],
    'B':['B4', 'B5', 'B6','B7'],
    'C':['C4', 'C5', 'C6','C7'],
    'D':['D4', 'D5', 'D6','D7']})
```

```
print(f'DataFrame 1: \n {df1} \n DataFrame 2: \n {df2}')
```

Crear un DataFrame nuevo

```
## DataFrame 1:
##      A   B   C   D
## 0  A0  B0  C0  D0
## 1  A1  B1  C1  D1
## 2  A2  B2  C2  D2
## 3  A3  B3  C3  D3
##  DataFrame 2:
##      A   B   C   D
## 0  A4  B4  C4  D4
## 1  A5  B5  C5  D5
## 2  A6  B6  C6  D6
## 3  A7  B7  C7  D7
```

```
pd.concat([df1,df2])
```

Concatenar los DataFrames

```
##     A   B   C   D
## 0  A0  B0  C0  D0
## 1  A1  B1  C1  D1
## 2  A2  B2  C2  D2
## 3  A3  B3  C3  D3
## 0  A4  B4  C4  D4
## 1  A5  B5  C5  D5
```

```
## 2  A6  B6  C6  D6
## 3  A7  B7  C7  D7
```

```
pd.concat([df1,df2], ignore_index= True)
```

Corregir los indices

```
##     A   B   C   D
## 0  A0  B0  C0  D0
## 1  A1  B1  C1  D1
## 2  A2  B2  C2  D2
## 3  A3  B3  C3  D3
## 4  A4  B4  C4  D4
## 5  A5  B5  C5  D5
## 6  A6  B6  C6  D6
## 7  A7  B7  C7  D7
```

```
pd.concat([df1,df2], axis = 1)
```

Por *axis 1*

```
##     A   B   C   D   A   B   C   D
## 0  A0  B0  C0  D0  A4  B4  C4  D4
## 1  A1  B1  C1  D1  A5  B5  C5  D5
## 2  A2  B2  C2  D2  A6  B6  C6  D6
## 3  A3  B3  C3  D3  A7  B7  C7  D7
```

Merge

Permite combinar dos DataFrames a nivel de columnas. La organización por columnas no va a ser la misma para ambos DataFrames, por tanto, se crearan valores `NaN` para rellenar los espacios vacíos. Crecimiento horizontal

```
izq = pd.DataFrame({'key' : ['k0', 'k1', 'k2','k3'],
 'A' : ['A0', 'A1', 'A2','A3'],
'B': ['B0', 'B1', 'B2','B3']})

der = pd.DataFrame({'key' : ['k0', 'k1', 'k2','k3'],
 'C' : ['C0', 'C1', 'C2','C3'],
'D': ['D0', 'D1', 'D2','D3']})
```

Key en común Unir el DataFrame **Der** a **Izq**

```
izq.merge(der)
```

```
##   key   A   B   C   D
## 0  k0  A0  B0  C0  D0
## 1  k1  A1  B1  C1  D1
## 2  k2  A2  B2  C2  D2
## 3  k3  A3  B3  C3  D3
```

```
izq = pd.DataFrame({'key' : ['k0', 'k1', 'k2','k3'],
 'A' : ['A0', 'A1', 'A2','A3'],
'B': ['B0', 'B1', 'B2','B3']})

der = pd.DataFrame({'key_2' : ['k0', 'k1', 'k2','k3'],
 'C' : ['C0', 'C1', 'C2','C3'],
'D': ['D0', 'D1', 'D2','D3']})
```

Sin columnas en común Debemos especificar las llaves

```
izq.merge(der, left_on = 'key', right_on='key_2')
```

```
##   key   A   B key_2   C   D
## 0  k0  A0  B0    k0  C0  D0
## 1  k1  A1  B1    k1  C1  D1
## 2  k2  A2  B2    k2  C2  D2
## 3  k3  A3  B3    k3  C3  D3
```

```
izq = pd.DataFrame({'key' : ['k0', 'k1', 'k2','k3'],
 'A' : ['A0', 'A1', 'A2','A3'],
'B': ['B0', 'B1', 'B2','B3']})

der = pd.DataFrame({'key_2' : ['k0', 'k1', 'k2',np.nan],
 'C' : ['C0', 'C1', 'C2','C3'],
'D': ['D0', 'D1', 'D2','D3']})
```

```
izq.merge(der, left_on = 'key', right_on='key_2', how='left')
```

Cuando hay algún `NaN` en nuestro DataFrame

```
##   key   A   B key_2    C    D
## 0  k0  A0  B0    k0   C0   D0
## 1  k1  A1  B1    k1   C1   D1
## 2  k2  A2  B2    k2   C2   D2
## 3  k3  A3  B3   NaN  NaN  NaN
```

JOIN

Es otra herramienta para hacer exactamente lo mismo, una combinación. La diferencia es que `join` va a ir a los índices y no a columnas específicas.

```
izq = pd.DataFrame({'A': ['A0','A1','A2'],
  'B':['B0','B1','B2']},
  index=['k0','k1','k2'])

der = pd.DataFrame({'C': ['C0','C1','C2'],
  'D':['D0','D1','D2']},
  index=['k0','k2','k3'])
```

Combinar izq con der

```
izq.join(der)
```

```
##      A   B    C    D
```

```
## k0  A0  B0   C0   D0
## k1  A1  B1  NaN  NaN
## k2  A2  B2   C1   D1
```

Traer todos los datos aunque no hagan match

```
izq.join(der, how = 'outer')
```

```
##       A    B    C    D
## k0   A0   B0   C0   D0
## k1   A1   B1  NaN  NaN
## k2   A2   B2   C1   D1
## k3  NaN  NaN   C2   D2
```

`join` puede considerase como un caso específico de merge, pero más eficiente

Pivot y Melt

Son funciones que sirven para cambiar la estructura de nuestro DataFrame de acuerdo a nuestras necesidades.

Pivot

Pivot, básicamente, transforma los valores de determinadas columnas o filas en los índices de un nuevo DataFrame, y la intersección de estos es el valor resultante.

Partamos del siguiente DataFrame

```
df_books.head()
```

```
##                                                Name  ...        Genre
## 0                      10-Day Green Smoothie Cleanse  ...  Non Fiction
## 1                                11/22/63: A Novel  ...      Fiction
## 2            12 Rules for Life: An Antidote to Chaos  ...  Non Fiction
## 3                             1984 (Signet Classics)  ...      Fiction
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...  Non Fiction
##
## [5 rows x 7 columns]
```

Aplicamos `.pivot_table()`

```
df_books.pivot_table(index='Author',columns='Genre',values='User Rating')
```

```
## Genre                        Fiction  Non Fiction
## Author
## Abraham Verghese                 4.6          NaN
## Adam Gasiewski                   NaN     4.400000
## Adam Mansbach                    4.8          NaN
## Adir Levy                        4.8          NaN
## Admiral William H. McRaven       NaN     4.700000
## ...                              ...          ...
## Walter Isaacson                  NaN     4.566667
## William Davis                    NaN     4.400000
## William P. Young                 4.6          NaN
## Wizards RPG Team                 4.8          NaN
## Zhi Gang Sha                     NaN     4.600000
##
## [248 rows x 2 columns]
```

Como resultado, los valores de Author pasan a formar el índice por fila y los valores de Genre pasan a formar parte de los índices por columna, y el User Rating se mantiene como valor.

```
df_books.pivot_table(index='Genre',columns='Year', values='User Rating',aggfunc='sum')
```

aggfunc

```
## Year          2009   2010   2011   2012  ...   2016   2017   2018   2019
## Genre                                    ...
## Fiction      110.2   92.3   97.0   94.4  ...   89.6  113.7   99.5   96.4
## Non Fiction  119.0  135.6  130.9  132.2  ...  144.3  119.3  133.9  140.6
##
## [2 rows x 11 columns]
```

En este caso tenemos por cada género, la suma a lo largo de los años.

También podemos obtener la media, la desviación estándar, el conteo, la varianza, etc. Únicamente con cambiar el parámetro `aggfunc` que traduce la función de agrupamiento.

```
df_books.pivot_table(index='Genre', columns='Year', values='User Rating'
                                                ,aggfunc= [min, max, np.mean])
```

```
##              min                  ...      mean
## Year        2009 2010 2011 2012  ...      2016      2017      2018      2019
## Genre                            ...
## Fiction      4.0  4.1  4.2  3.3  ...  4.715789  4.737500  4.738095  4.820000
## Non Fiction  4.0  4.0  4.0  4.0  ...  4.654839  4.588462  4.617241  4.686667
##
## [2 rows x 33 columns]
```

Melt

El método melt toma las columnas del DataFrame y las pasa a filas, con dos nuevas columnas para especificar la antigua columna y el valor que traía.

```
df_books[['Name','Genre']].head(5)
```

```
##                                                 Name        Genre
## 0                      10-Day Green Smoothie Cleanse  Non Fiction
## 1                                  11/22/63: A Novel      Fiction
## 2             12 Rules for Life: An Antidote to Chaos  Non Fiction
## 3                               1984 (Signet Classics)      Fiction
## 4  5,000 Awesome Facts (About Everything!) (Natio...  Non Fiction
```

Aplicando `melt()`

```
df_books[['Name','Genre']].head(5).melt()
```

```
##   variable                                              value
## 0     Name                       10-Day Green Smoothie Cleanse
## 1     Name                                   11/22/63: A Novel
## 2     Name              12 Rules for Life: An Antidote to Chaos
## 3     Name                                1984 (Signet Classics)
## 4     Name  5,000 Awesome Facts (About Everything!) (Natio...
## 5    Genre                                          Non Fiction
## 6    Genre                                              Fiction
## 7    Genre                                          Non Fiction
## 8    Genre                                              Fiction
```

```
## 9    Genre                                          Non Fiction
```

Ahora cada resultado de las dos columnas pasa a una fila de este modo a tipo **llave**:**valor**.

Podemos utilizar melt también de la siguiente forma

```
df_books.melt(id_vars='Year',value_vars='Genre')
```

```
##      Year variable        value
## 0    2016    Genre  Non Fiction
## 1    2011    Genre      Fiction
## 2    2018    Genre  Non Fiction
## 3    2017    Genre      Fiction
## 4    2019    Genre  Non Fiction
## ..    ...      ...          ...
## 545  2019    Genre      Fiction
## 546  2016    Genre  Non Fiction
## 547  2017    Genre  Non Fiction
## 548  2018    Genre  Non Fiction
## 549  2019    Genre  Non Fiction
##
## [550 rows x 3 columns]
```

Simplemente, podemos seleccionar las columnas que no quiero hacer `melt` usando el parámetro `id_vars`. Para este caso **Year** y también la única columna que quiero aplicar el `melt`, para este caso **Genre** con la propiedad `value_vars`.

Apply

Es un comando muy poderoso que nos deja aplicar funciones a nuestro DataFrame

```
df_books.head(2)
```

```
##                             Name        Author  ...  Year        Genre
## 0  10-Day Green Smoothie Cleanse      JJ Smith  ...  2016  Non Fiction
## 1              11/22/63: A Novel  Stephen King  ...  2011      Fiction
##
## [2 rows x 7 columns]
```

Creamos nuestra función

```
def two_times(value):
        return value * 2
```

Lo aplicamos a la columna de **User Rating**

```
df_books['User Rating'].apply(two_times)
```

```
## 0      9.4
## 1      9.2
## 2      9.4
## 3      9.4
## 4      9.6
##       ...
## 545    9.8
## 546    9.4
## 547    9.4
## 548    9.4
## 549    9.4
```

```
## Name: User Rating, Length: 550, dtype: float64
```

Podemos guardar los resultados en una nueva columna

```
df_books['User Rating2'] = df_books['User Rating'].apply(two_times)
df_books.head(5)
```

```
##                                                 Name  ... User Rating2
## 0                       10-Day Green Smoothie Cleanse  ...          9.4
## 1                                    11/22/63: A Novel  ...          9.2
## 2              12 Rules for Life: An Antidote to Chaos  ...          9.4
## 3                                1984 (Signet Classics)  ...          9.4
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...          9.6
##
## [5 rows x 8 columns]
```

Utilizando lambda functions

```
df_books['User Rating2'] =df_books['User Rating'].apply(lambda x: x* 3)
df_books.head()
```

```
##                                                 Name  ... User Rating2
## 0                       10-Day Green Smoothie Cleanse  ...         14.1
## 1                                    11/22/63: A Novel  ...         13.8
## 2              12 Rules for Life: An Antidote to Chaos  ...         14.1
## 3                                1984 (Signet Classics)  ...         14.1
## 4  5,000 Awesome Facts (About Everything!) (Natio...  ...         14.4
##
## [5 rows x 8 columns]
```

Apply en varias columnas con condiciones, hay que especificar a que los vamos a aplicar (filas o columnas)

```
df_books.apply(lambda x: x['User Rating'] * 2 if x['Genre'] == 'Fiction' else x['User Rating'], axis =
```

```
## 0      4.7
## 1      9.2
## 2      4.7
## 3      9.4
## 4      4.8
##       ...
## 545    9.8
## 546    4.7
## 547    4.7
## 548    4.7
## 549    4.7
## Length: 550, dtype: float64
```

Puedes usar funciones que reciban más argumentos especificando los demás argumentos en `args`. Por ejemplo una función que convierta el **rating** desde cualquier base a cualquier base sería

```
# Una función con parametros
def convert_rating_base(rating, current_base, new_base):
    # Convertimos el Rating de la base actual a la nueva
        return rating * new_base / current_base

# Apply y se pasan todos los demás argumentos después del primero en args
df_books["User Rating"].apply(convert_rating_base, args=(5, 7))
```

```
## 0      6.58
## 1      6.44
## 2      6.58
## 3      6.58
## 4      6.72
##        ...
## 545    6.86
## 546    6.58
## 547    6.58
## 548    6.58
## 549    6.58
## Name: User Rating, Length: 550, dtype: float64
```

Recordemos que podemos seleccionar una columna también de esta forma `df.Name` (Siempre y cuando su nombre no contenga espacios)