¿Qué es lo primero que se hace?

- Importar las bibliotecas
- Leer el archivo csv
- Filtrar por las columnas que queremos usar
- Eliminamos las filas que tengan al menos una columna con un NaN

Como obtener información sobre un DataFrame:

- *df.info()* : Detalles de la estructura
- *df.describe()* : Estadísticas sobre datos numéricos de las columnas
- *df.values* : Devuelve el dataframe en forma de vector
- *df.dtypes* : Tipos de datos asociados a las columnas
- *df.columns* : Devuelve las columnas
- *df.index* : Devuelve las filas
- *df.value_counts()* : Cuantas ocurrencias de cada valor
- *df.size* : Cantidad de elementos totales (filas x columnas)
- *df.shape* : (Filas, Columnas)
- *df.head(n)* : Devuelve las primeras n filas del dataframe
- *df.tail(n)* : Devuelve las últimas n filas del dataframe
- *len(df)* : Cantidad de filas que tiene el dataframe
- *df.count()* : Cantidad de elementos válidos por columna del dataframe

Como obtener información sobre una Serie:

- *serie.hasnans()* : Devuelve si true si hay al menos un nan
- *serie.isnull()* : Devuelve una serie booleana, con True si en la serie hay nan, sino false
- *serie.count()* : Cantidad de elementos válidos de la serie
- *type(df['Nombre_columna'])* : Tipo de dato de la columna
- df.nlargest/nsmallest (n, 'columna') → devuelve n elementos más grandes/chicos de la columna pedida
- df.min()
- df.max()
- df.mean()
- df[*'Nombre_columna']*.replace (*to_replace*='algo_a_reemplazar', *value*='nuevo_valor', inplace = True)

Como cambiar el tipo de dato de una columna:

df['col a'] = pd.to_datetime(df['col a'], *errors*='coerce') : Cambia el tipo de dato a datetime
obtener el año: pandas.Series.dt.year

obtener el mes: pandas.Series.dt.month
obtener el dia: pandas.Series.dt.day
obtener la hora: pandas.Series.dt.hour
obtener el minuto: pandas.Series.dt.minute

df[‘col a’].to_frame()

Renombrar filas y columnas:

idx_rename = { 'Nombre index original' : 'Nombre index nuevo'}
col_rename = {'Nombre columna original' : 'Nombre columna nueva'}
df_nuevo = df_viejo(index = idx_remove, columns = col_rename)

Creación y Borrado de columnas:

- *df.drop(columns='columna')* # borra columna del df
- *df.columns.get_loc('columna')+1* # devuelve la posición de la columna indicada
- *df.insert(loc=posición de la columna, column 'nueva columna', value=(condición))* #inserta la nueva columna en una posición específica
- *del df['columna']* # la borra del df

Cómo acceder a la información del DataFrame:

Columnas

- *df[‘nombre col’]* : Una sola
- *df[['columna1','columna2','columna3']]* : Selecciona lista de columnas

Subsets de data:

Puede ser con **loc** (usa strings) y **iloc** (usa enteros)

- *df.iloc[n]* : Accedes a todas las columnas de la fila n, n es un número entero
- *df.iloc[x:y:z]* de x a y, de z en z. x e y son filas.
- *df.iloc[: , [A,B]]* : Todas las filas de las columnas de A a B
- *df.loc[(‘A’)]* : Accedes a todas las columnas de la fila ‘A’
- *df.loc[df.col_a.str.contains(‘value’), :]* : Trae todas las filas que tengan ‘value’ en la ‘col_a’ junto a todas las demás columnas de estas filas.

GroupBy:

- *df.groupby('columna')* : Agrupa por los diferentes valores de la columna
- *df.groupby(['columna1'],['columna2'])*
- *agg*: Realiza una función para una columna agrupada
- *df.groupby('columna1').agg({'columna2' : ['mean','max'], 'columna' : ['mean','max'] })* varias funciones, varias columnas

Idx	Nombre	Mean
0	Jorge	8
1	Leticia	5

- transform: misma cantidad de filas (esta falopa, ojo)
  Agrupar y usar transform al mismo tiempo sobre una serie específica. NASHE
  `df.groupby('order')["ext price"].transform('sum')`

- *df.groupby('columna1'). transform('mean')* quedaria algo asi:

IDX	Nombre	Nota	Mean
0	Jorge	9	8
1	Leticia	5	5
2	Jorge	7	8

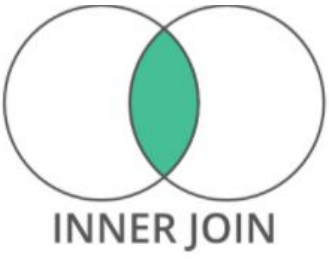
- Podemos definir nuestras propias funciones a aplicar:
- *def add_one(x):*
  return x +1
- *df.groupby('columna').agg({'otra_columna': add_one})*

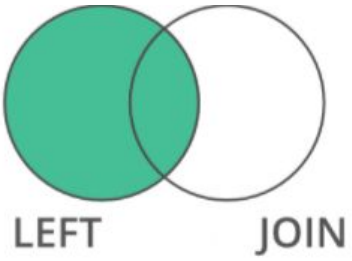
Cuando tenes lvl 1 y lvl 0 pa ponerlo juntito bro

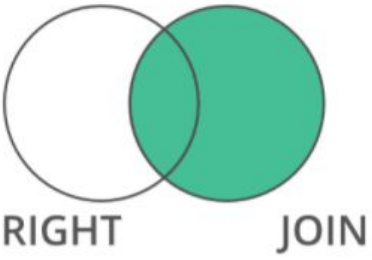
```
level0 = interno.columns.get_level_values(0)
level1 = interno.columns.get_level_values(1)
interno.columns = level0 + '_' + level1
```

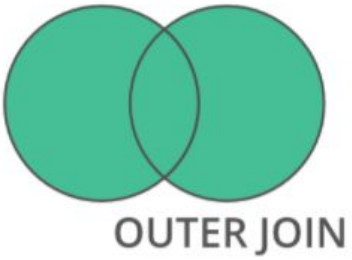
Merge y mas cosas:

1. Append:
    a. *df_pendiado = df1.append(df2)* # si los dos df se parecen mucho pero tienen una columna distinta el append va unirlos y en la misma columna pone los datos de ambos df.
2. Concat:
    a. *df_contac = pd.concat([df1, df2]* # hace lo mismo que append pero con otra notación.
3. Merge:
    a. *df_merge = df1.merge(df2, left_on= 'columna' , right_on='columna', how = modo)* # los modos pueden ser outer, inner(viene por defecto), left y right.



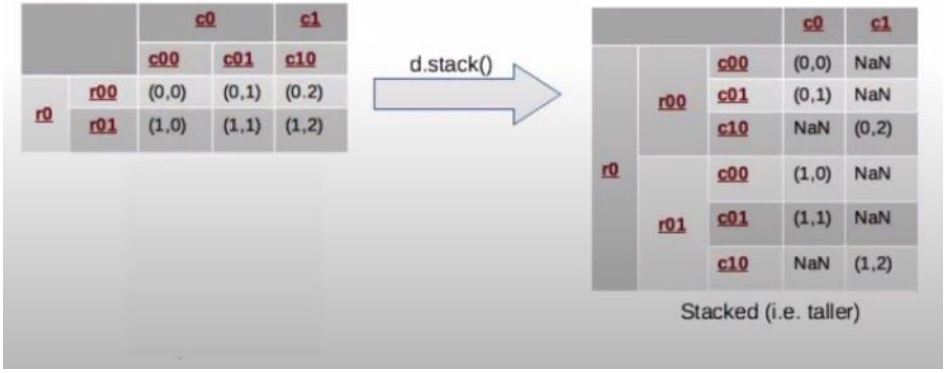




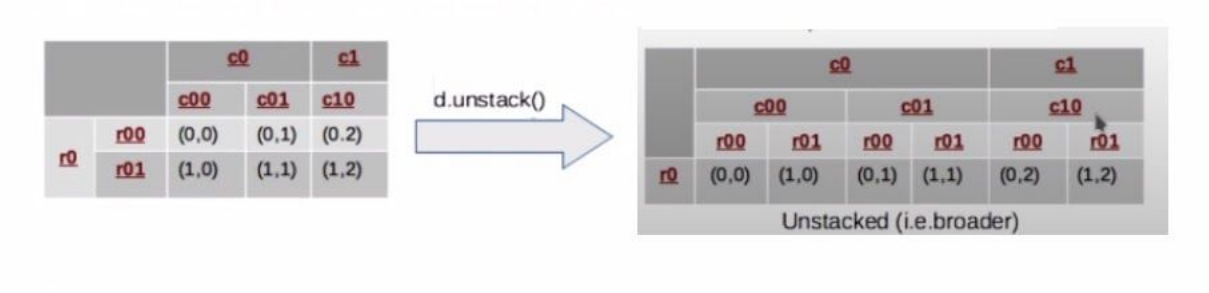


Stack y unstack

Realizar un **stacking** de un DF significa mover el indice por columna más interna del df para que se convierta en el indice por fila más interno del df resultante



Realizar **unstacking** es mover el indice por fila más interno del DF para que se convierta en el indice por columna más interna del df resultante



Como usar el **Pivot Table:**

Sirve para crear una tabla a partir de un dataframe existente. Consta principalmente de 4 parámetros: Index, Columns, Values, Aggfunc. También, se le puede agregar parámetros del estilo: **dropna** (no incluye columnas con todas celdas nan)

df.pivot_table(index='index', columns='columns', value='values')

Index: Es una columna del dataframe original que se va a usar como índice de la tabla

Columns: Los valores como esta serán las nuevas columnas.

Value: Columna que se utiliza para llenar las celdas formadas.

Aggfunc: Función/es que se le aplican a la columna 'values' (pivot no tiene este parámetro y si hay values duplicados tira ValueError).