

### Step 1. Import the necessary libraries

```
MI >>>
import pandas as pd
import numpy as np
```

### Step 2. Import the dataset from this [address](https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv).

```
MI >>>
address = "https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv"
```

### Step 3. Assign it to a variable called chipo.

```
MI >>>
url= address
chipo = pd.read_csv(url,sep="\t") #transforma en formato dataframe y el argumento sep= es para que sepa
chipo                             #diferenciar unas columnas de otras.
```

### Step 4. See the first 10 entries

```
MI >>>
chipo.loc[0:9] #loc toma el valor explicito del indice, me saca las 10 primeras lineas
```

### Step 5. What is the number of observations in the dataset?

```
MI >>>
len(chipo.index) #Me da las posiciones totales del indice
```

4622

```
MI >>>
chipo.info() #Me da información general de la tabla
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   order_id        4622 non-null   int64
1   quantity        4622 non-null   int64
```

```
MI >>>
chipo.shape[1] #Me da shape las filas y las colum. en una tupla (4622, 5), accedo a su pos 1
```

#### Step 6. What is the number of columns in the dataset?

▶ MI 

```
chipo.columns  
len(chipo.columns) #Me da las columnas pero al poner len me da el número total
```

5

#### Step 7. Print the name of all the columns.

1] ▶ MI 

```
chipo.columns #Me imprime el nombre de todas las columnas.
```

```
Index(['order_id', 'quantity', 'item_name', 'choice_description',  
      'item_price'],  
      dtype='object')
```

#### Step 8. How is the dataset indexed?

▶ MI 

```
print(chipo.index) #Me da como está indexado el dataset Comienza 0, Ultimo 4622, Con paso 1
```

```
RangeIndex(start=0, stop=4622, step=1)
```

#### Step 9. Which was the most-ordered item?

1] ▶ MI 

```
chipo.mode() #Me da el mas repetido de dataframe
```

	order_id	quantity	item_name	choice_description	item_price
0	926	1	Chicken Bowl	[Diet Coke]	\$8.75

5] ▶ MI 

```
chipo.groupby(["item_name"]).sum().max()  
#selecciono por colum item_name con groupby funciona agrupando sin repetir los elementos. Luego le pido que me de  
la suma de los valores de toda la tabla de cada item (puesto que tienen varios cada uno) y luego le digo que me  
busque el valor máximo de todos.
```


```
order_id    713926  
quantity      761  
dtype: int64
```

▶ MI 


```
c=chipo.groupby(["item_name"]).sum()["quantity"]  
c  
#otra manera de hacer el ejercicio es agrupar por columna item_name y que sume valores de quantity
```

```
item_name  
6 Pack Soft Drink          55  
Barbacoa Bowl              66  
Barbacoa Burrito           91  
Barbacoa Crispy Tacos      12  
Barbacoa Salad Bowl        12
```


**Step 10. For the most-ordered item, how many items were ordered?**


```
▶ MI +B  
chipo.groupby(["item_name"]).sum().max() #para el más pedido cuantos se pidieron en total|
```

```
order_id    713926  
quantity      761  
dtype: int64
```

```
] ▶ MI +B  
chipo.groupby(["choice_description"]).sum().max() #artículo mas pedido en columna choice_description|
```


```
order_id    123455  
quantity      159  
dtype: int64
```

```
3] ▶ MI +B  
c=chipo.groupby(["choice_description"]).sum()  
c=c.sort_values(["quantity"], ascending=False) #Enmascaramiento|  
  
{}
```


```
| ▶ MI +B  
chipo.quantity.sum() #cuantos items se pidieron en total|
```


4972

**Step 13. Turn the item price into a float**

```
▶ MI +B  
chipo["item_price"].dtype #Me dice que el dato es tipo objeto
```


dtype('O')

```
i] ▶ MI +B  
chipo["item_price"]  
  
0      $2.39  
1      $3.39  
2      $3.39  
3      $2.39  
4      $16.98  
...  
4617   $11.75  
4618   $11.75  
4619   $11.25  
4620    $8.75  
4621    $8.75  
Name: item_price, Length: 4622, dtype: object
```

```
17] ▶ MI 
#def quitar(palabra):
#    return palabra[1:]
#chipo aplica una funcion lambda que le dice que palabra tome el valor [1:] y le aplique float. en un elemento solo
#apply es funcion de pandas que hace que lo que la funcion hace para un elem, el lo hace en todos los elementos.

chipo["item_price"]=chipo["item_price"].apply(lambda palabra:float(palabra[1:]))
```

### Step 13.c. Check the item price type


```
1] ▶ MI 
chipo["item_price"]

0      2.39
1      3.39
2      3.39
3      2.39
4     16.98
...
4617   11.75
4618   11.75
4619   11.25
4620    8.75
4621    8.75
Name: item_price, Length: 4622, dtype: float64
```

```
1] ▶ MI
dtype('float64')
```

### Step 14. How much was the revenue for the period in the dataset?

```
2] ▶ MI
Revenue was: $39237.02
```

```
3] ▶ MI 
facturacion_platos = chipo["item_price"] * chipo["quantity"]
print(facturacion_platos) #Multiplico cada plato por el numero de veces que se ha pedido

0      2.39
1      3.39
2      3.39
3      2.39
4     33.96
...
4617   11.75
4618   11.75
4619   11.25
4620    8.75
4621    8.75
Length: 4622, dtype: float64
```

```

) ] ▶ MI 8+8

fact_platos=facturacion_platos.sum()
print(fact_platos) #sumo los resultados del apartado anterior.

39237.02

```

```

[71] ▶ MI 8+8

id_orders=chipo["order_id"].value_counts()
print(id_orders) #cuantos pedidos se han hecho en el periodo

926      23
1483     14
205      12
691      11
1786     11
..
105       1
702       1
718       1
568       1
800       1
Name: order_id, Length: 1834, dtype: int64

```

```

[72] ▶ MI 8+8

number_of_orders=chipo["order_id"].max()
print(number_of_orders)

1834

```

```

] ▶ MI 8+8

media=fact_platos/number_of_orders #media de facturacion
print(media)

21.39423118865867

```

```

] ▶ MI 8+8

platos=chipo["item_name"].unique().tolist()
platos #Me muestra los elementos únicos diferentes totales

['Chips and Fresh Tomato Salsa',
 'Izze',
 'Nantucket Nectar',
 'Chips and Tomatillo-Green Chili Salsa',
 'Chicken Bowl',
 'Side of Chips',
 'Steak Burrito',
 'Steak Soft Tacos',
 'Chips and Guacamole',
 'Chicken Crispy Tacos',
 'Chicken Soft Tacos',
 'Chicken Burrito',
 'Canned Soda',
 'Barbacoa Burrito',
 'Carnitas Burrito',
 'Carnitas Bowl',
 'Bottled Water',

```

```

] ▶ MI 8+8

print(len(platos))

50

```

```

▶ MI 8→8
platos=chipo.groupby(["item_name"])
platos

```

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0BA70D18>

```

▶ MI 8→8
print(len(platos))

```

50

```

] ▶ MI 8→8
media_edad=b/a
print(round(media_edad)) #Round redondea los decimales

```

34

```

] ▶ MI 8→8
media_edad=b/a
print(media_edad)

```

34.05196182396607

.....

```

] ▶ MI 8→8
a=usuarios["age"].mean() #saca la media de los valores de una columna
print(a)

```

34.05196182396607

#### Step 4. See the first 25 entries

```

] ▶ MI 8→8
usuarios.loc[:24] #Me saca las 25 primeras por posicion

```

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101

24 21 F artist 94533

[5]

▶ MI  → B

```
usuarios.head(25) #Por defecto saca las 5 primeras pero con parámetro saca las que le indique
```

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344

**Step 5. See the last 10 entries**

[6]

▶ MI  → B

```
usuarios.tail(10) #Lo contrario de head, me saca por defecto las 5 últimas
```

	age	gender	occupation	zip_code
user_id				
934	61	M	engineer	22902

**Step 6. What is the number of observations in the dataset?**

▶ MI  → B

```
usuarios.shape[0] # da el número de filas
```

943

**Step 7. What is the number of columns in the dataset?**

▶ MI  → B

```
usuarios.shape[1] #Me da el número de columnas
```

4

**Step 8. Print the name of all the columns.**

▶ MI  → B

```
usuarios.columns #Me da el nombre de las columnas
```

```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

### Step 9. How is the dataset indexed?

▶ MI 

```
# "the index" (aka "the labels")
```

▶ MI 

```
usuarios.index #Me saca el índice de la data
```

```
Int64Index([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
            ...
            934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
           dtype='int64', name='user_id', length=943)
```

### Step 10. What is the data type of each column?

!▶ MI 

```
usuarios.dtypes #Me saca el tipo de dato de las columnas
```

```
age          int64
gender       object
occupation   object
zip_code     object
dtype: object
```

### Step 11. Print only the occupation column

.3] ▶ MI 

```
usuarios["occupation"] #Me imprime una columna
```

.4] ▶ MI 

```
usuarios.iloc[:,2] #Me imprime la columna por localizacion
```

### Step 12. How many different occupations are in this dataset?

!▶ MI 

```
len(usuarios.groupby(["occupation"]).sum()) #selecciono por columna occupation con groupby que funciona agrupando sin repetir los elementos.
Luego le pido que me de la suma de los valores y me sacaría los valores escritos uno a uno pero con el len me da la suma total.
```



### Step 13. What is the most frequent occupation?

```
▶ MI 8→8  
usuarios.mode()#Me da el mas repetido de dataframe
```

### Step 14. Summarize the DataFrame.

```
] ▶ MI 8→8  
usuarios.describe() #describe hace un resumen de datos estadísticos de toda la tabla
```

age

sumarize

```
▶ MI 8→8  
usuarios.columns
```

Index(['age', 'gender', 'occupation', 'zip\_code'], dtype='object')

```
▶ MI 8→8  
for column in usuarios.columns:  
    print(usuarios[column].describe()) #para las columnas en usuarios, saca de usuarios las columnas y haz resumen estadístico.
```

```
▶ MI 8→8  
usuarios.occupation.describe() #sumariza una columna en este caso occupation
```

```
▶ MI 8→8  
a=usuarios["age"].mean() #saca la media de los valores de una columna  
print(a)
```

34.05196182396607

```
▶ MI 8→8  
usuarios.age.value_counts().tail() # Tail Saca los 5 valores mas bajos de la columna age, al contrario de head
```

```
11    1  
10    1  
73    1  
66    1  
7     1  
Name: age, dtype: int64
```

```
▶ MI 8→8  
usuarios.age.value_counts().min() #saco el valor minimo de una columna, en este caso columna age
```

## Borrar una columna

▶ MI 

```
# del(food['creator']) me borra la columna creator, y esa fórmula me borra poniendo el nombre la columna que quiera.
```

Step 8. What is the name of 105th column?

1] ▶ MI 

```
food.columns[105] #imprime una columna determinada
```

'-fructose\_100g'

## Tipo de datos de una columna

1] ▶ MI 

```
food.dtypes[104] # me dice el tipo de los datos de la columna 104
```

dtype('float64')

Step 11. What is the product name of the 19th observation?

2] ▶ MI 

```
food["product_name"].values[19] #values me da la lista completa de una serie de valores y en este caso le delimito a una posición
```

'Organic Oat Groats'

## Otra manera de hacer lo anterior

0] ▶ MI 

```
food.at[18, "product_name"] # at te devuelve el valor de la columna que esta a la derecha de la coma en la posición 18
```

'Lotus Organic Brown Jasmine Rice'