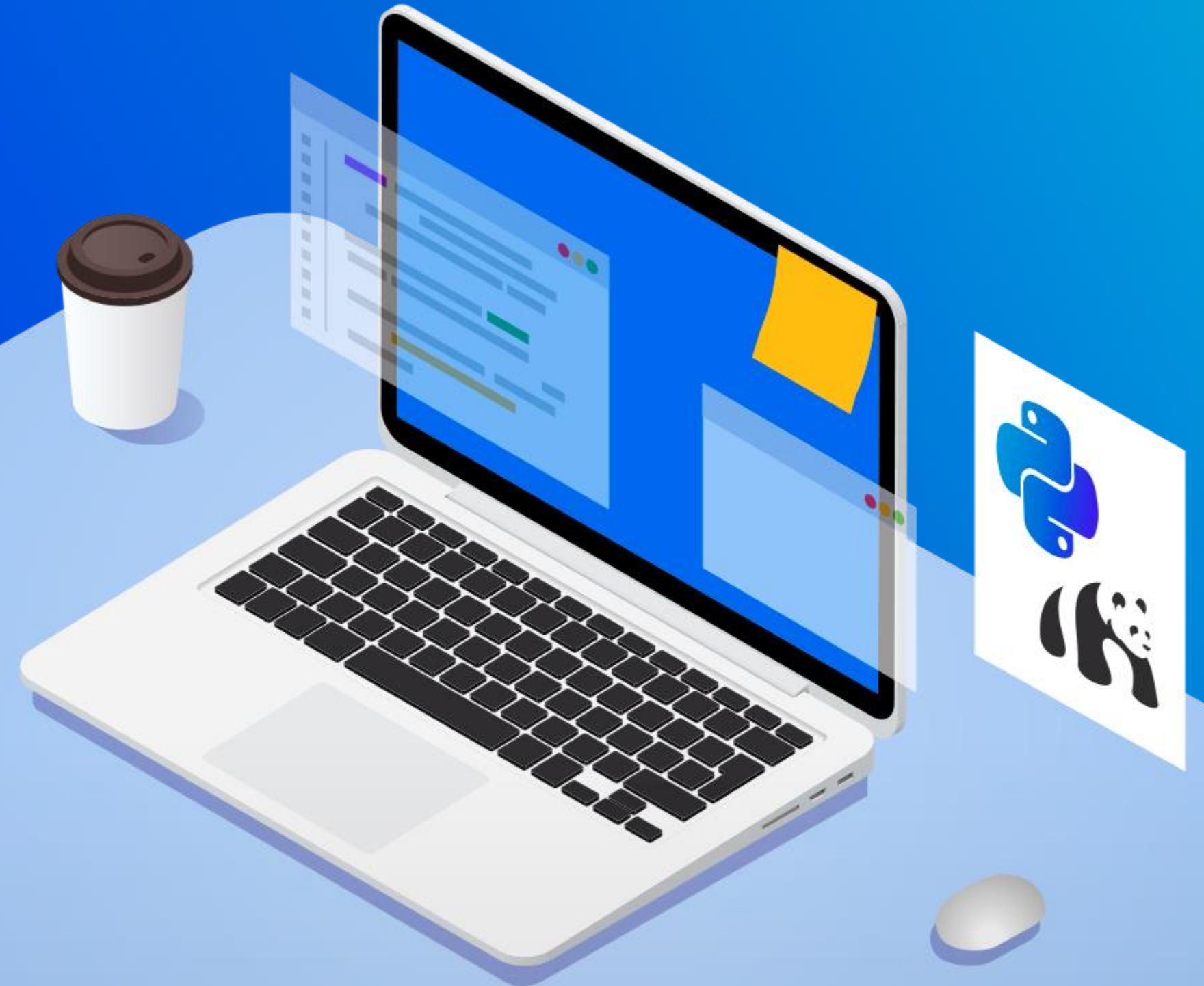


PROCESAMIENTO Y EXTRACCIÓN DE INFORMACIÓN

>>> Parte 2. Ejemplos prácticos



Ejemplo Práctico: Ventas Supermercado

En un pequeño supermercado:
Su dueño, Don Juan, anota todas las ventas en su cuaderno del año anterior. Le recomendaron numerar sus productos para tener un mejor control, así que hizo un inventario asignándole a cada producto un número de 5 dígitos.




Ejemplo Práctico: Ventas Supermercado

Don Juan registra a mano la fecha de la venta, el número del producto que se vendió y la cantidad que se vendió. De ahí que, surge la necesidad de digitalizar la información para lograr hacer reportes de ventas, y proyectar qué productos debe comprar para tenerlos disponibles para sus clientes.



Traspaso de los datos a un archivo CSV

Una sobrina de Don Juan, trabajó en los datos y los traspasó a un archivo CSV de nombre “**ventas.csv**”. A continuación una muestra de cómo se ven sus datos:



FECHA	NUM	CANT
2018=9=22	23346	18
2018=2=8	60128	49
2018=6=14	52.605,	48
2018=8=5	74.437	44
2018=5=21	52605,	36
2018=8=21	71.935,	14
2018=6=11	61814	35

Traspaso de los datos a un archivo CSV

Al observar los datos, notamos que hay varias dificultades:

FECHA	NUM	CANT
2018=9=22	23346	18
2018=2=8	60128	49
2018=6=14	52.605,	48
2018=8=5	74.437	44
2018=5=21	52605,	36
2018=8=21	71.935,	14
2018=6=11	61814	35

En la columna FECHA: el año, mes y día están separados por un signo "=", cuando se debería ocupar "/".

En la columna NUM: el número de cada producto a veces tiene espacios, puntos y comas.

¿Cómo podríamos limpiar la información?

Una forma de limpiar la columna **FECHA**:

CÓDIGO

```
import pandas as pd
df = pd.read_csv("ventas.csv",encoding="latin-1",sep=";")

print(df.head())
df["FECHA"] = df["FECHA"].str.replace("=", "/")
print(df.head())
```

RESULTADO

	FECHA	NUM	CANT
0	2018=9=22	23346	18
1	2018=2=8	60128	49
2	2018=6=14	52.605,	48
3	2018=8=5	74.437	44
4	2018=5=21	52605,	36

	FECHA	NUM	CANT
0	2018/9/22	23346	18
1	2018/2/8	60128	49
2	2018/6/14	52.605,	48
3	2018/8/5	74.437	44
4	2018/5/21	52605,	36

Para limpiar la columna fecha, hacemos un **replace** de "=" por "/". Específicamente, en la columna FECHA y volvemos asignar este valor a la misma columna, para que adquiera los valores editados.

¿Cómo podríamos limpiar la información?

Una forma de limpiar la columna **NUM**:

CÓDIGO

```
print(df.head())
```

```
df["NUM"] = df["NUM"].str.replace(" ", "")  
df["NUM"] = df["NUM"].str.replace(".", "")  
df["NUM"] = df["NUM"].str.replace(",", "")
```

```
print(df.head())
```

RESULTADO

	FECHA	NUM	CANT
0	2018/9/22	23346	18
1	2018/2/8	60128	49
2	2018/6/14	52.605,	48
3	2018/8/5	74.437	44
4	2018/5/21	52605,	36

	FECHA	NUM	CANT
0	2018/9/22	23346	18
1	2018/2/8	60128	49
2	2018/6/14	52605	48
3	2018/8/5	74437	44
4	2018/5/21	52605	36

Para eliminar los caracteres espacios, puntos y comas, ocupamos la función **replace** en la columna **NUM** y la volvemos asignar en la misma columna.

¿Cómo podríamos limpiar la información?

Una forma de limpiar la columna **NUM**:

CÓDIGO

```
print(df.head())

df["NUM"] = df["NUM"].str.replace(" ", "")
df["NUM"] = df["NUM"].str.replace(".", "")
df["NUM"] = df["NUM"].str.replace(",", "")

print(df.head())
```

RESULTADO

	FECHA	NUM	CANT
0	2018/9/22	23346	18
1	2018/2/8	60128	49
2	2018/6/14	52.605,	48
3	2018/8/5	74.437	44
4	2018/5/21	52605,	36

	FECHA	NUM	CANT
0	2018/9/22	23346	18
1	2018/2/8	60128	49
2	2018/6/14	52605	48
3	2018/8/5	74437	44
4	2018/5/21	52605	36

! Importante:

De ahora en adelante, el avance será progresivo por eso no se cargó el Data Frame nuevamente.

Datos limpios ¿ahora qué sigue?

Los datos no nos dicen mucho, por lo tanto Don Juan facilitó la lista de productos.

Esta lista ya está limpia y en un archivo CSV de nombre “**lista_productos.csv**”.

NUM	CAT	NOMBRE	PRECIO
10878	ASEO PERSONAL	Desodorante	\$ 1,590
14462	ASEO PERSONAL	Shampoo	\$ 2,390
23295	ABARROTES	Arroz	\$ 659
23346	ABARROTES	Aceite	\$ 890
30400	LIMPIEZA	Limpiapisos X	\$ 1,090
34424	LIMPIEZA	Limpiapisos Y	\$ 1,190

¿Cómo unir la información?

Nos gustaría unir la información de “**lista_productos.csv**” con la lista de ventas, para tener más datos de cada producto (según su número).

CÓDIGO

```
df2 = pd.read_csv("lista_productos.csv",encoding="latin-1")
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)

df2["NUM"] = df2["NUM"].astype(str)

print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)

df = df.merge(df2,on="NUM")

print(df.head())
```

RESULTADO

```
DF: (ventas.csv)
FECHA    object
NUM      object
CANT     int64
dtype: object
DF: (lista_productos.csv)
NUM      int64
CAT      object
NOMBRE   object
PRECIO   int64
dtype: object
DF: (lista_productos.csv)
NUM      object
CAT      object
NOMBRE   object
PRECIO   int64
dtype: object
```

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
0	2018/9/22	23346	18	ABARROTES	Aceite	890
1	2018/10/7	23346	35	ABARROTES	Aceite	890
2	2018/7/1	23346	13	ABARROTES	Aceite	890
3	2018/8/3	23346	10	ABARROTES	Aceite	890
4	2018/7/13	23346	37	ABARROTES	Aceite	890

¿Cómo unir la información?

Nos gustaría unir la información de “**lista_productos.csv**” con la lista de ventas, para tener más datos de cada producto (según su número).

CÓDIGO

```
df2 = pd.read_csv("lista_productos.csv",encoding="latin-1")
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)
```

```
df2["NUM"] = df2["NUM"].astype(str)
```

```
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)
```

```
df = df.merge(df2,on="NUM")
```

```
print(df.head())
```

RESULTADO

```
DF: (ventas.csv)
FECHA    object
```

Aquí cargamos el CSV “**lista_productos.csv**” en Pandas. Luego, analizamos los tipos de datos para las columnas **NUM** en el Data Frame de **ventas.csv** (**df**) y el Data Frame de **lista_productos.csv** (**df2**) para unir ambos.

```
dtype: object
DF: (lista_productos.csv)
NUM      object
CAT      object
NOMBRE   object
PRECIO   int64
dtype: object
```

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
0	2018/9/22	23346	18	ABARROTES	Aceite	890
1	2018/10/7	23346	35	ABARROTES	Aceite	890
2	2018/7/1	23346	13	ABARROTES	Aceite	890
3	2018/8/3	23346	10	ABARROTES	Aceite	890
4	2018/7/13	23346	37	ABARROTES	Aceite	890

¿Cómo unir la información?

Nos gustaría unir la información de “**lista_productos.csv**” con la lista de ventas, para tener más datos de cada producto (según su número).

CÓDIGO

```
df2 = pd.read_csv("lista_productos.csv",encoding="latin-1")
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)
```

```
df2["NUM"] = df2["NUM"].astype(str)

print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)

df = df.merge(df2,on="NUM")

print(df.head())
```

RESULTADO

```
DF: (ventas.csv)
FECHA    object
NUM      object
CANT     int64
dtype: object
DF: (lista_productos.csv)
NUM      int64
CAT      object
NOMBRE   object
PRECIO   int64
```

En **df** la columna NUM es de tipo **object** (**string**) y en **df2** es de tipo **int64** (**int**). Por lo tanto, transformaremos esta columna a **string** usando la función **astype**. Luego, como ambas columnas tienen el mismo tipo, hacemos **merge** en la columna **NUM**. Asignamos el resultado de este **merge** a **df**.

2	2018/7/1	23346	13	ABARROTES	Aceite	890
3	2018/8/3	23346	10	ABARROTES	Aceite	890
4	2018/7/13	23346	37	ABARROTES	Aceite	890

¿Cómo unir la información?

Nos gustaría unir la información de “**lista_productos.csv**” con la lista de ventas, para tener más datos de cada producto (según su número).

CÓDIGO

```
df2 = pd.read_csv("lista_productos.csv",encoding="latin-1")
print("DF: (ventas.csv)")
print(df.dtypes)
print("DF: (lista_productos.csv)")
print(df2.dtypes)
```

```
df2
```

Finalmente, al imprimir en consola podemos ver cómo se agregaron las columnas de **lista_productos.csv**.

```
print(df.dtypes)
print(df2.dtypes)
```

```
df = df.merge(df2,on="NUM")
```

```
print(df.head())
```

RESULTADO

```
DF: (ventas.csv)
FECHA    object
NUM      object
CANT     int64
dtype: object
```

```
DF: (lista_productos.csv)
NUM      int64
CAT      object
NOMBRE   object
PRECIO   int64
dtype: object
```

```
DF: (lista_productos.csv)
NUM      object
CAT      object
NOMBRE   object
PRECIO   int64
dtype: object
```

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
0	2018/9/22	23346	18	ABARROTES	Aceite	890
1	2018/10/7	23346	35	ABARROTES	Aceite	890
2	2018/7/1	23346	13	ABARROTES	Aceite	890
3	2018/8/3	23346	10	ABARROTES	Aceite	890
4	2018/7/13	23346	37	ABARROTES	Aceite	890

¿Cómo podríamos extraer información del Data Frame?

Don Juan intuye que hay productos que se venden muy poco. Por lo tanto, le gustaría buscar aquellos que tengan un valor inferior a \$1.000 y que se vendan menos de 5 veces por venta.

CÓDIGO

```
print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5)])
```

¿Cómo lo podríamos saber?

En este caso, hacemos dos filtros: que el precio sea menor a 1000 y que la cantidad sea menor que 5. Ambos (y por eso se ocupa el “&” de and) se deben cumplir.

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO
21	2018/9/26	23346	4	ABARROTES	Aceite	890
123	2018/1/20	71935	3	PANADERIA	Marraqueta (kg)	890
126	2018/7/10	71935	1	PANADERIA	Marraqueta (kg)	890
264	2018/12/7	72963	1	PANADERIA	Hallulla (kg)	890
272	2018/3/2	37847	1	LIMPIEZA	Esponja	390
275	2018/12/23	37847	1	LIMPIEZA	Esponja	390
374	2018/10/19	36736	3	LIMPIEZA	Limpiapisos Z	990
377	2018/10/7	36736	3	LIMPIEZA	Limpiapisos Z	990

¿Cómo trabajar filtros para el Data Frame?

A partir del filtro anterior, a Don Juan le gustaría saber qué productos se vendieron menos de 5 veces, que además tengan un valor menor a \$1000 y se hayan vendido en los meses de abril y mayo.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/", expand=True)
df_aux.columns = ["ANHO", "MES", "DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

¿Cómo podríamos hacer este filtro?

Primero: separamos la columna **FECHA** en tres columnas con un **split**, y el Data Frame resultante lo agregamos a una variable.

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo trabajar filtros para el Data Frame?

A partir del filtro anterior, a Don Juan le gustaría saber qué productos se vendieron menos de 5 veces, que además tengan un valor menor a \$1000 y se hayan vendido en los meses de abril y mayo.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/", expand=True)
df_aux.columns = ["ANHO", "MES", "DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

¿Cómo podríamos hacer este filtro?

Segundo: renombramos las columnas en este Data Frame como **ANHO** (se ocupa NH en vez de ñ para evitar problemas con los nombres), **MES** y **DÍA**.

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo trabajar filtros para el Data Frame?

A partir del filtro anterior, a Don Juan le gustaría saber qué productos se vendieron menos de 5 veces, que además tengan un valor menor a \$1000 y se hayan vendido en los meses de abril y mayo.

CÓDIGO

```
df_aux = df["FECHA"].str.split("/", expand=True)
df_aux.columns = ["ANHO", "MES", "DIA"]

df = df.join(df_aux)
print(df.head())

print(df.loc[(df["PRECIO"] < 1000) & (df["CANT"] < 5) & ((df["MES"] == "4") | (df["MES"] == "5"))])
```

¿Cómo podríamos hacer este filtro?

Al filtro anterior, agregamos **or** (por eso se ocupa "|"), Así revisamos que el **MES** sea igual a 4 ó 5.

Para el resto de las condiciones usamos **and**, ya que se deben cumplir las 3.

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA
508	2018/5/27	53447	2	VERDURAS	Espinaca	890	2018	5	27
534	2018/4/24	40359	1	FRUTAS	Manzana (kg)	590	2018	4	24

¿Cómo calcular el promedio de ventas mensual?

Finalmente, a Don Juan le gustaría conocer el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO

```
df["TOTAL"] = df["PRECIO"] * df["CANT"]  
print(df.head())
```

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL
0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020
1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150
2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570
3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900
4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930

¿Cómo podríamos hacer este filtro?

Se podría hacer con un **pivot_table**. No obstante, antes es necesario sacar el total de ventas por día.

¿Cómo calcular el promedio de ventas mensual?

Finalmente, a Don Juan le gustaría conocer el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO

```
df["TOTAL"] = df["PRECIO"] * df["CANT"]  
print(df.head())
```

¿Cómo podríamos hacer este filtro?

Como tenemos la cantidad y el precio unitario, debemos comenzar calculando una nueva columna con esta multiplicación.

RESULTADO

	FECHA	NUM	CANT	CAT	NOMBRE	PRECIO	ANHO	MES	DIA	TOTAL
0	2018/9/22	23346	18	ABARROTES	Aceite	890	2018	9	22	16020
1	2018/10/7	23346	35	ABARROTES	Aceite	890	2018	10	7	31150
2	2018/7/1	23346	13	ABARROTES	Aceite	890	2018	7	1	11570
3	2018/8/3	23346	10	ABARROTES	Aceite	890	2018	8	3	8900
4	2018/7/13	23346	37	ABARROTES	Aceite	890	2018	7	13	32930

¿Cómo calcular el promedio de ventas mensual?

Ya tenemos el total de ventas por mes, ahora podemos sacar el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO

```
import numpy as np

df3 = df.pivot_table(index="MES",values="TOTAL", columns="CAT", aggfunc={np.mean,np.std,np.min,np.max,})
print(df3.head())
```

Para saber el desglose de ventas mensuales por categoría, podríamos haber puesto **MES** o **CAT** en columnas o **index** (filas) indiferentemente.

RESULTADO

	amax					...	std			
CAT	ABARROTES	ASEO PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS	
MES					...					
1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873	
10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922	
11	27019.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	104193.210226	13555.516712	
12	43610.0	73140.0	233290.0	53410.0	...	22055.584121	22781.992889	146829.421605	10400.431241	
2	32930.0	100380.0	230040.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016	

¿Cómo calcular el promedio de ventas mensual?

Ya tenemos el total de ventas por mes, ahora podemos sacar el promedio, mínimo, máximo y desviación estándar del promedio mensual de ventas diarias, desglosado por categorías.

CÓDIGO

```
import numpy as np

df3 = df.pivot_table(index="MES",values="TOTAL", columns="CAT", aggfunc={np.mean,np.std,np.min,np.max,})
print(df3.head())
```

Lo importante es que en **values** esté el **TOTAL** (que fue la nueva columna que calculamos) y en **aggfunc** incorporar los estadísticos descriptivos del paquete **numpy** que se requieran calcular del **TOTAL**.

RESULTADO

CAT	amax					std				
	ABARROTES	ASEO PERSONAL	CARNES	FRUTAS	...	FRUTAS	LIMPIEZA	PANADERIA	VERDURAS	
MES					...					
1	27590.0	60420.0	234530.0	41420.0	...	15314.064451	15918.487767	30855.105034	16126.047873	
10	31150.0	57360.0	255600.0	21830.0	...	6555.221583	18143.579213	153825.787103	13818.547922	
11	27019.0	119500.0	166260.0	52320.0	...	13755.764347	9341.328867	104193.210226	13555.516712	
12	43610.0	73140.0	233290.0	53410.0	...	22055.584121	22781.992889	146829.421605	10400.431241	
2	32930.0	100380.0	230040.0	42510.0	...	11840.789318	15890.647238	161624.793673	13897.433016	

Ejemplo Práctico: Ventas Supermercado

Finalmente, para que el Data Frame con las columnas que creamos y editamos no se borren, debemos guardarlo en un archivo CSV. Además, guardaremos el reporte generado en `pivot_table`:

CÓDIGO

```
df3.to_csv("reporte_ventas.csv")  
df.to_csv("dataframe_reporte_ventas.csv")
```



Conclusiones

- En este ejemplo práctico, conseguimos ver el proceso completo de procesamiento de información, mediante diversas técnicas y herramientas.
- Partimos de una lista de datos que no era muy valiosa (porque estaba sucia, y no estaba contextualizada con los datos de los productos), y terminamos con información muy valiosa que puede ser crítica para este negocio.
- Para lograr todos los procedimientos revisados en el ejemplo, usamos las herramientas y técnicas que sirven para **limpiar datos, consolidarlos y finalmente, extraer información valiosa a partir de ellos.**

>>> Cierre

Has finalizado la revisión de los contenidos de esta clase.

A continuación, te invitamos a realizar las actividades y a revisar los recursos del módulo que encontrarás en plataforma.