

---

# PowerBI

---

BDA

22/05/25 - IES FERNANDO WIRTZ

Santiago Fernández Seoane

Fecha	Motivo del cambio
	Versión inicial

## Índice

Orígenes de datos.....	2
Relaciones entre los datos.....	3
Elementos visuales - Informe 1.....	4
Elementos visuales - Informe 2.....	5
Spark-HDFS.....	6

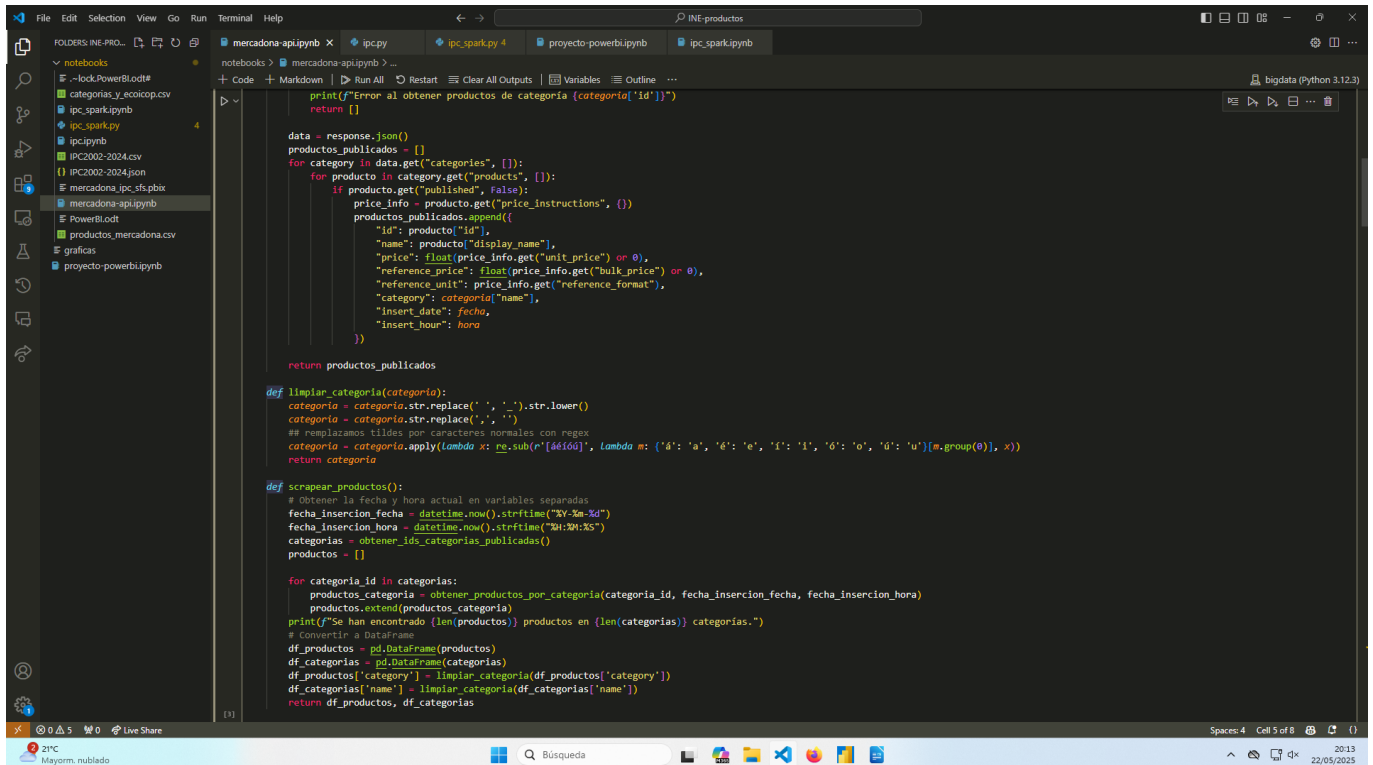
## Orígenes de datos

Como hemos conseguido los siguientes orígenes de datos y como los hemos importado en el PowerBI.

- CSV Datos precios de Mercadona (2020) descargados de kaggle :
  - <https://www.kaggle.com/datasets/thegurusteam/mercadona-es-product-pricing>
- Scrapping (Api Mercadona):
  - Scrapeados los datos de categorías + parsear categorías en los GRUPOS ECOICOP (Guardados en CSV e importados como formato Archivo CSV)
  - Scrapeados los datos de productos. (Guardado en CSV e importado desde Script de Python)
    - <https://tienda.mercadona.es/api/categories/>
- CSV: Datos descargados del INE - IPC desde 2020 hasta 2024:
  - <https://www.ine.es/jaxiT3/Tabla.htm?t=50918&L=0>
- Edición a la anterior tabla de donde se extraen las provincias y se parsean con su coordenadas geográficas para poder graficar en un mapa, se concatenan al archivo y se guarda como JSON para su posterior inserción como archivo json.
- usamos Geopy para la conversión de nombre de provincia a coordenadas.

# Scrapping API Mercadona

A la hora de scrappear tenía scrppeada la web de compra online de mercadona, pero encontré que tienen una api, y tanto para optimización de tiempo como de recursos opté por dejar el scrapping de la api, en ella cogemos tanto los datos de las categorías como sus respectivas id de categoría como los datos de productos, está todo el código en funciones separadas que scrappa.



```
print(f"Error al obtener productos de categoria {categoria['id']}")
return []

data = response.json()
productos_publicados = []
for category in data.get("categories", []):
    for producto in category.get("products", []):
        if producto.get("published", False):
            price_info = producto.get("price_instructions", {})
            productos_publicados.append({
                "id": producto["id"],
                "name": producto["display_name"],
                "price": float(price_info.get("unit_price") or 0),
                "reference_price": float(price_info.get("bulk_price") or 0),
                "reference_unit": price_info.get("reference_format"),
                "category": categoria["name"],
                "insert_date": fecha,
                "insert_hour": hora
            })

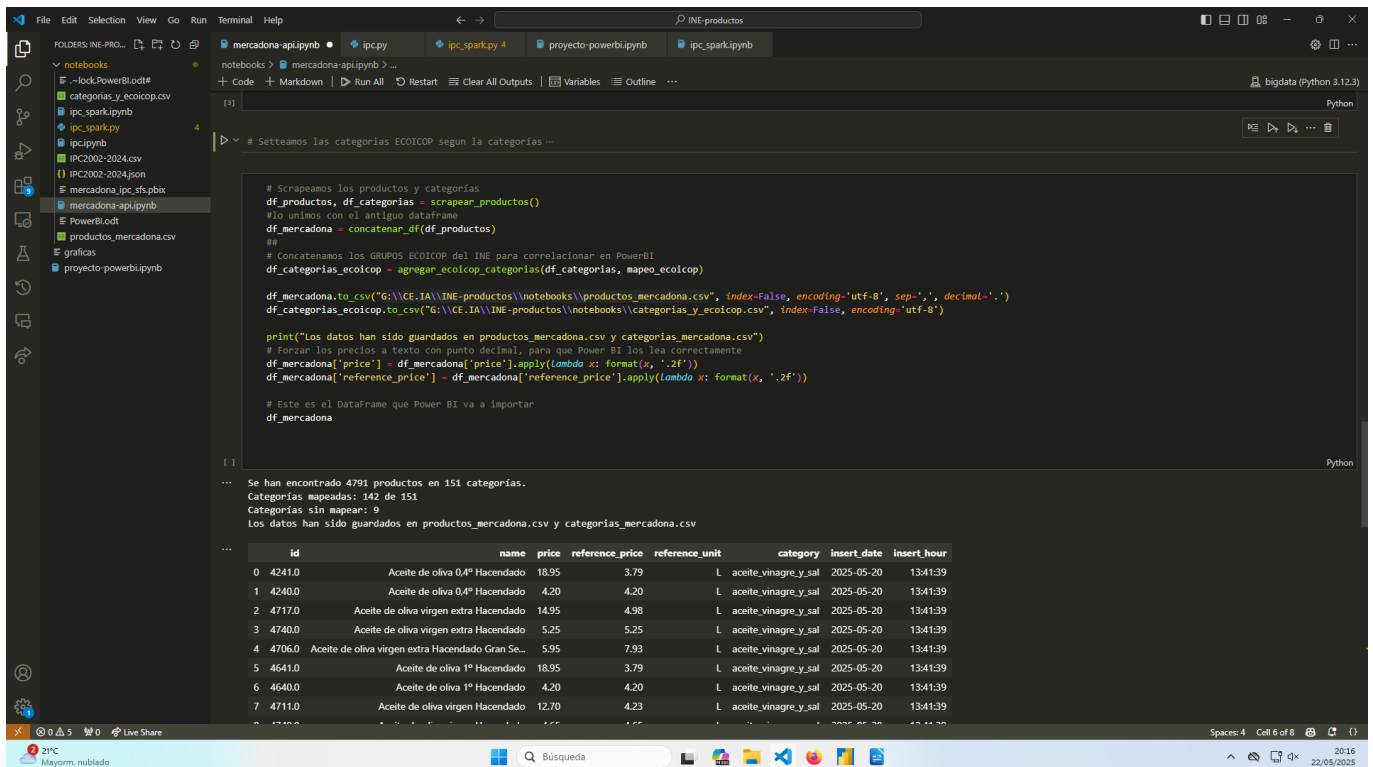
return productos_publicados

def limpiar_categoria(categoria):
    categoria = categoria.str.replace(' ', '_').str.lower()
    categoria = categoria.str.replace('.', '')
    ## reemplazamos tildes por caracteres normales con regex
    categoria = categoria.apply(lambda x: re.sub(r'[áéíó]', lambda m: ('á': 'a', 'é': 'e', 'í': 'i', 'ó': 'o', 'ú': 'u')[m.group(0)], x))
    return categoria

def scrappear_productos():
    # Obtener la fecha y hora actual en variables separadas
    fecha_insercion_fecha = datetime.now().strftime("%Y-%m-%d")
    fecha_insercion_hora = datetime.now().strftime("%H:%M:%S")
    categorias = obtener_ids_categorias_publicadas()
    productos = []

    for categoria_id in categorias:
        productos_categoria = obtener_productos_por_categoria(categoria_id, fecha_insercion_fecha, fecha_insercion_hora)
        productos.extend(productos_categoria)
    print(f"Se han encontrado {len(productos)} productos en {len(categorias)} categorias.")
    # Convertir a DataFrame
    df_productos = pd.DataFrame(productos)
    df_categorias = pd.DataFrame(categorias)
    df_productos['category'] = limpiar_categoria(df_productos['category'])
    df_categorias['name'] = limpiar_categoria(df_categorias['name'])
    return df_productos, df_categorias
```

Cuando leemos las funciones, podemos ejecutar el código que concatena el df sacado de kaggle (que contiene datos scrappados anteriormente) y los vuelve a concatenar y añadirlo directamente a powerbi



```
notebooks > mercadona-api.ipynb > ...
+ Code + Markdown + Run All + Restart + Clear All Outputs + Variables + Outline ...

# Setteamos las categorías ECOICOP según la categorías ...

# Scraperamos los productos y categorías
df_productos, df_categorias = scrapear_productos()
# lo unimos con el antiguo dataframe
df_mercadona = concatenar_df(df_productos)
# Concatenamos los GRUPOS ECOICOP del INE para correlacionar en PowerBI
df_categorias_ecoicop = agregar_ecoicop_categorias(df_categorias, mapeo_ecoicop)

df_mercadona.to_csv("G:\\CE-IA\\INE-productos\\notebooks\\productos_mercadona.csv", index=False, encoding='utf-8', sep=',', decimal='.')
df_categorias_ecoicop.to_csv("G:\\CE-IA\\INE-productos\\notebooks\\categorias_y_ecoicop.csv", index=False, encoding='utf-8')

print("Los datos han sido guardados en productos_mercadona.csv y categorias_mercadona.csv")
# Forzar los precios a texto con punto decimal, para que Power BI los lea correctamente
df_mercadona['price'] = df_mercadona['price'].apply(lambda x: format(x, '.2f'))
df_mercadona['reference_price'] = df_mercadona['reference_price'].apply(lambda x: format(x, '.2f'))

# Este es el DataFrame que Power BI va a importar
df_mercadona

Se han encontrado 4791 productos en 151 categorías.
Categorías mapeadas: 142 de 151
Categorías sin mapear: 9
Los datos han sido guardados en productos_mercadona.csv y categorias_mercadona.csv

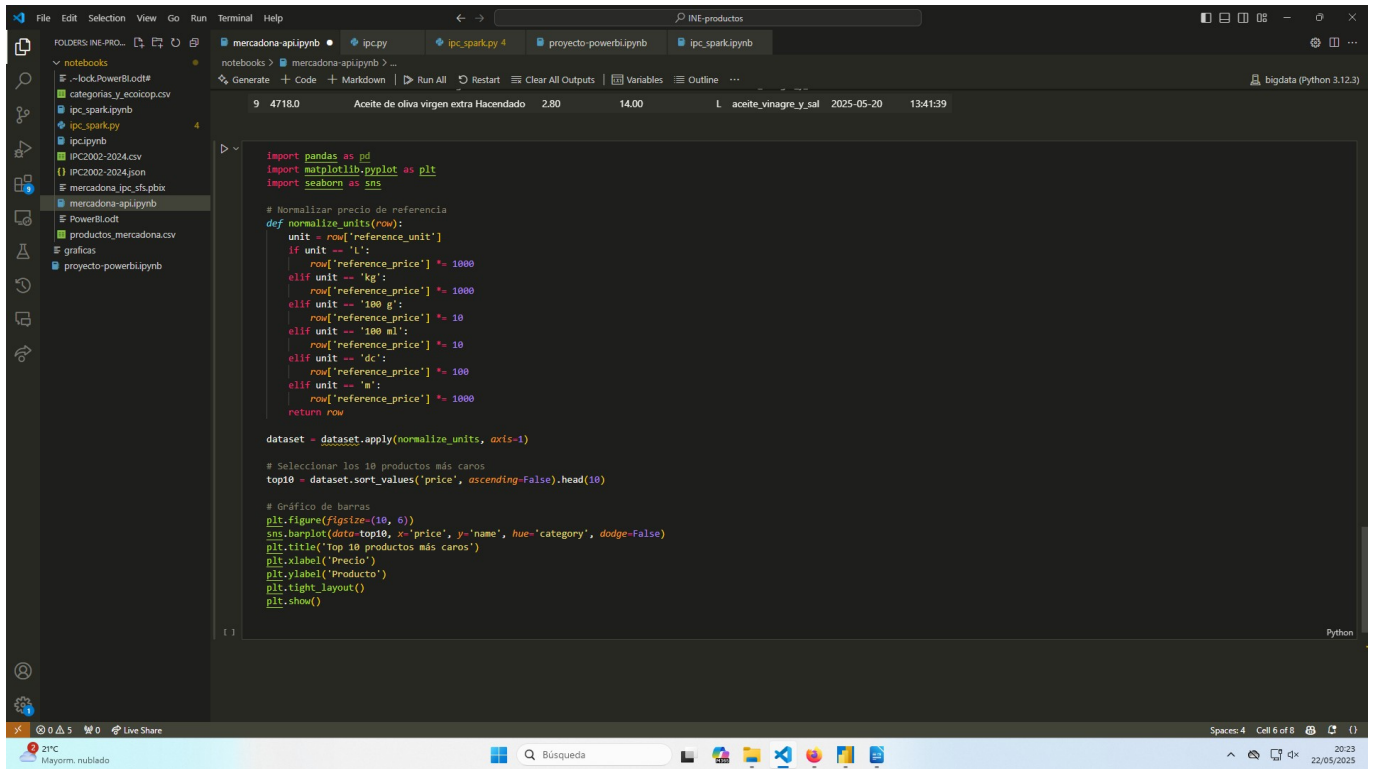

```

	id	name	price	reference_price	reference_unit	category	insert_date	insert_hour
0	4241.0	Aceite de oliva 0,4º Hacendado	18.95	3.79	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
1	4240.0	Aceite de oliva 0,4º Hacendado	4.20	4.20	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
2	4717.0	Aceite de oliva virgen extra Hacendado	14.95	4.98	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
3	4740.0	Aceite de oliva virgen extra Hacendado	5.25	5.25	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
4	4706.0	Aceite de oliva virgen extra Hacendado Gran Se...	5.95	7.93	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
5	4641.0	Aceite de oliva 1º Hacendado	18.95	3.79	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
6	4640.0	Aceite de oliva 1º Hacendado	4.20	4.20	L	aceite_vinagre_y_sal	2025-05-20	13:41:39
7	4711.0	Aceite de oliva virgen Hacendado	12.70	4.23	L	aceite_vinagre_y_sal	2025-05-20	13:41:39

Las ultimas lineas settean las columnas price y reference\_price en formato float (string) con formato estadounidense, una vez leído desde powerbi tendremos que transformar los datos de las columnas para ponerlos en modo float para que asi si que detecte bien los decimales

# Script Visual De Python

El script que he añadido directamente desde PowerBI para visualizar datos es el siguiente, en el se normalizan las unidades



The screenshot shows a Jupyter Notebook interface within the PowerBI environment. The notebook is titled 'mercadona-api.py' and is part of a project named 'proyecto-powerbi'. The code is written in Python and is designed to process and visualize data from a dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Normalizar precio de referencia
def normalize_units(row):
    unit = row['reference_unit']
    if unit == 'l':
        row['reference_price'] *= 1000
    elif unit == 'kg':
        row['reference_price'] *= 1000
    elif unit == '100 g':
        row['reference_price'] *= 10
    elif unit == '100 ml':
        row['reference_price'] *= 10
    elif unit == 'dc':
        row['reference_price'] *= 100
    elif unit == 'm':
        row['reference_price'] *= 1000
    return row

dataset = dataset.apply(normalize_units, axis=1)

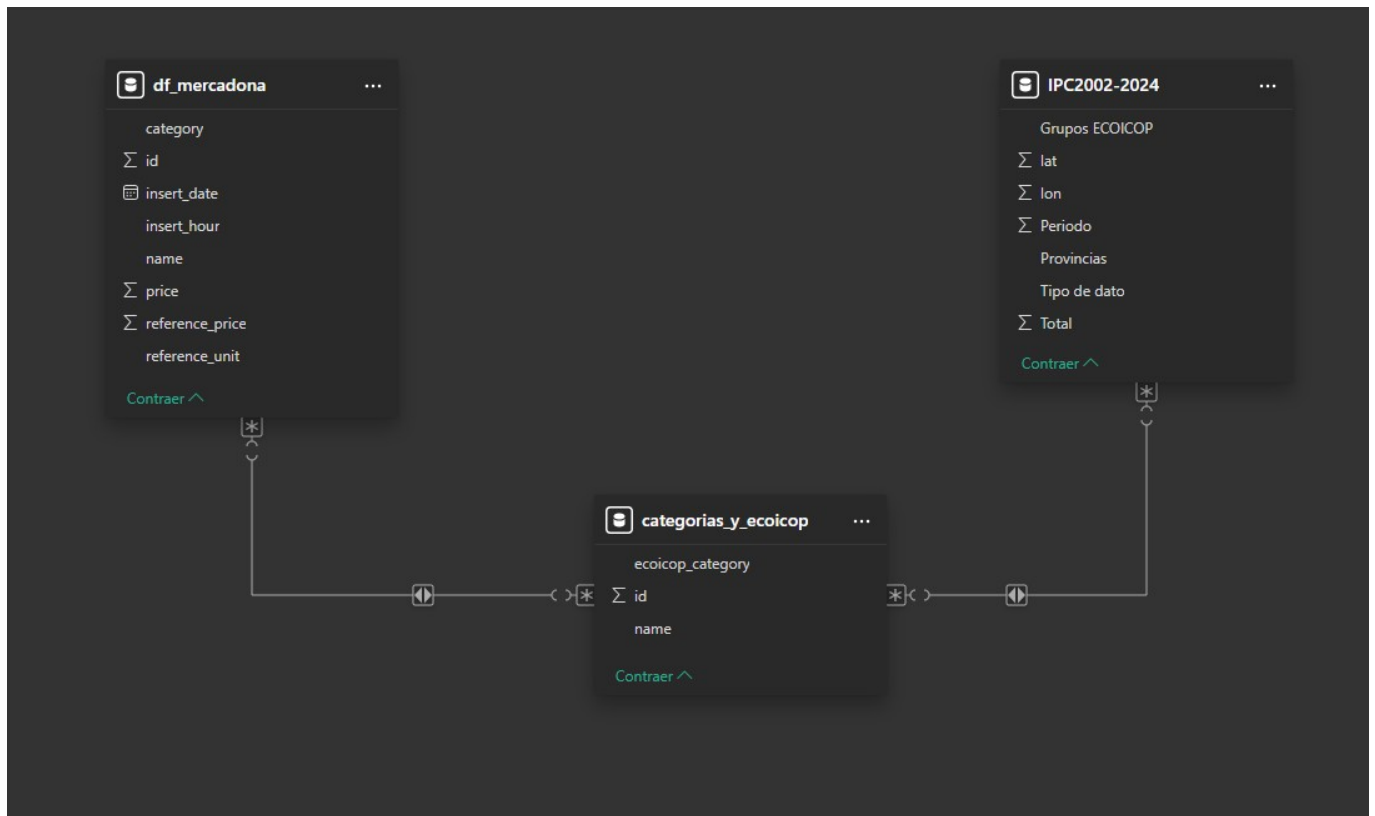
# Seleccionar los 10 productos más caros
top10 = dataset.sort_values('price', ascending=False).head(10)

# Gráfico de barras
plt.figure(figsize=(10, 6))
sns.barplot(data=top10, x='price', y='name', hue='category', dodge=False)
plt.title('Top 10 productos más caros')
plt.xlabel('Precio')
plt.ylabel('Producto')
plt.tight_layout()
plt.show()
```

The code defines a function `normalize_units` that takes a row of data and normalizes the 'reference\_price' based on the 'reference\_unit'. The units and their corresponding multipliers are: 'l' (1000), 'kg' (1000), '100 g' (10), '100 ml' (10), 'dc' (100), and 'm' (1000). The function is applied to the entire dataset using `dataset.apply(normalize_units, axis=1)`. Then, the top 10 most expensive products are selected using `dataset.sort_values('price', ascending=False).head(10)`. Finally, a bar chart is created using `sns.barplot` with the top 10 products, showing their price, name, and category. The chart is titled 'Top 10 productos más caros' and has 'Precio' on the x-axis and 'Producto' on the y-axis.

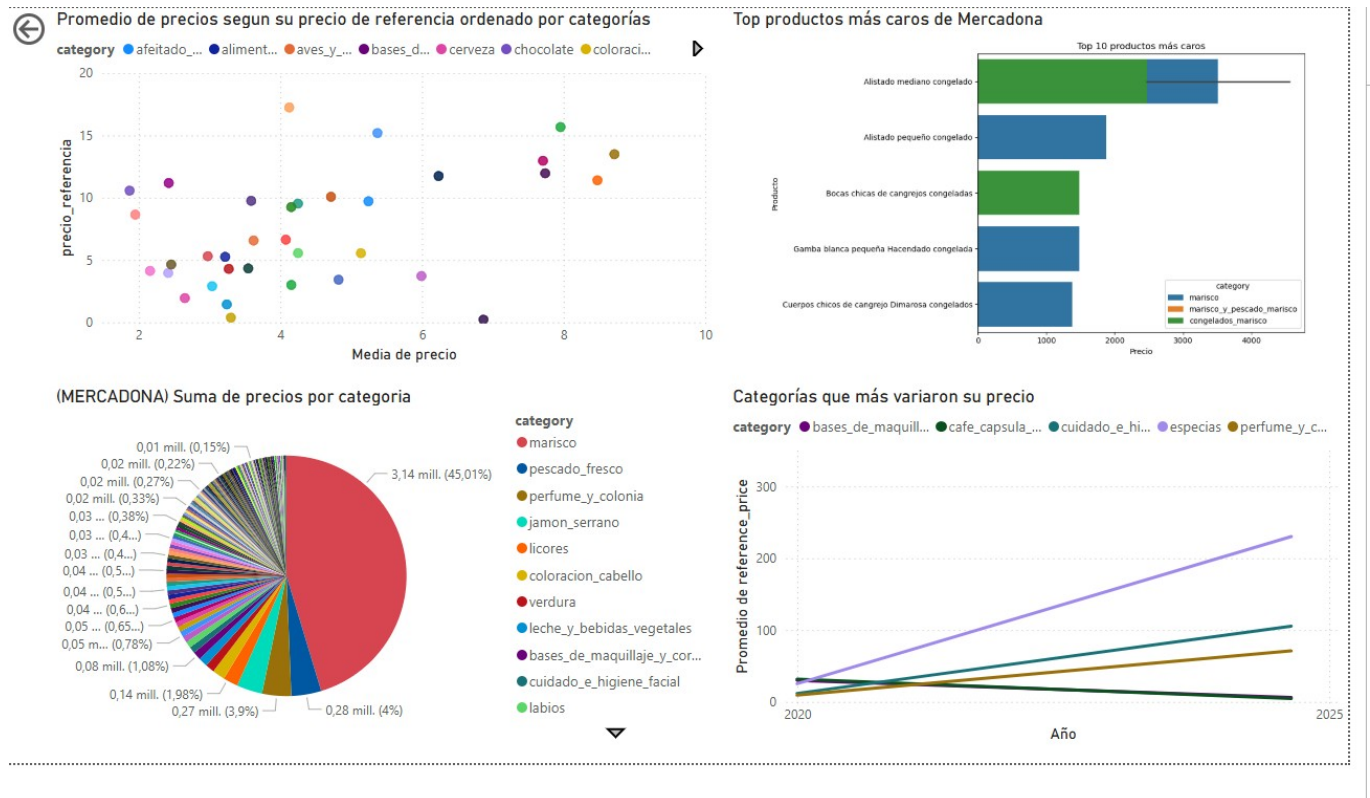
## Relaciones entre los datos

Hemos unido la tabla 'category' que referencia la categoría del producto junto con la columna name de la tabla 'categorias\_y\_ecoicop', que contiene las categorías junto con su id único. Además en esa misma tabla se une 'ecoicop\_category' con la columna de IPC2002-2024 llamada 'Grupos ECOICOP', la cual engloba las categorías del mercadona.



## Elementos visuales – Informe 1

En el primer informe nos hemos centrado sobre todo en los datos del DF de mercadona, del cual hemos graficado las siguientes relaciones de la siguiente manera y con los siguientes elementos visuales:

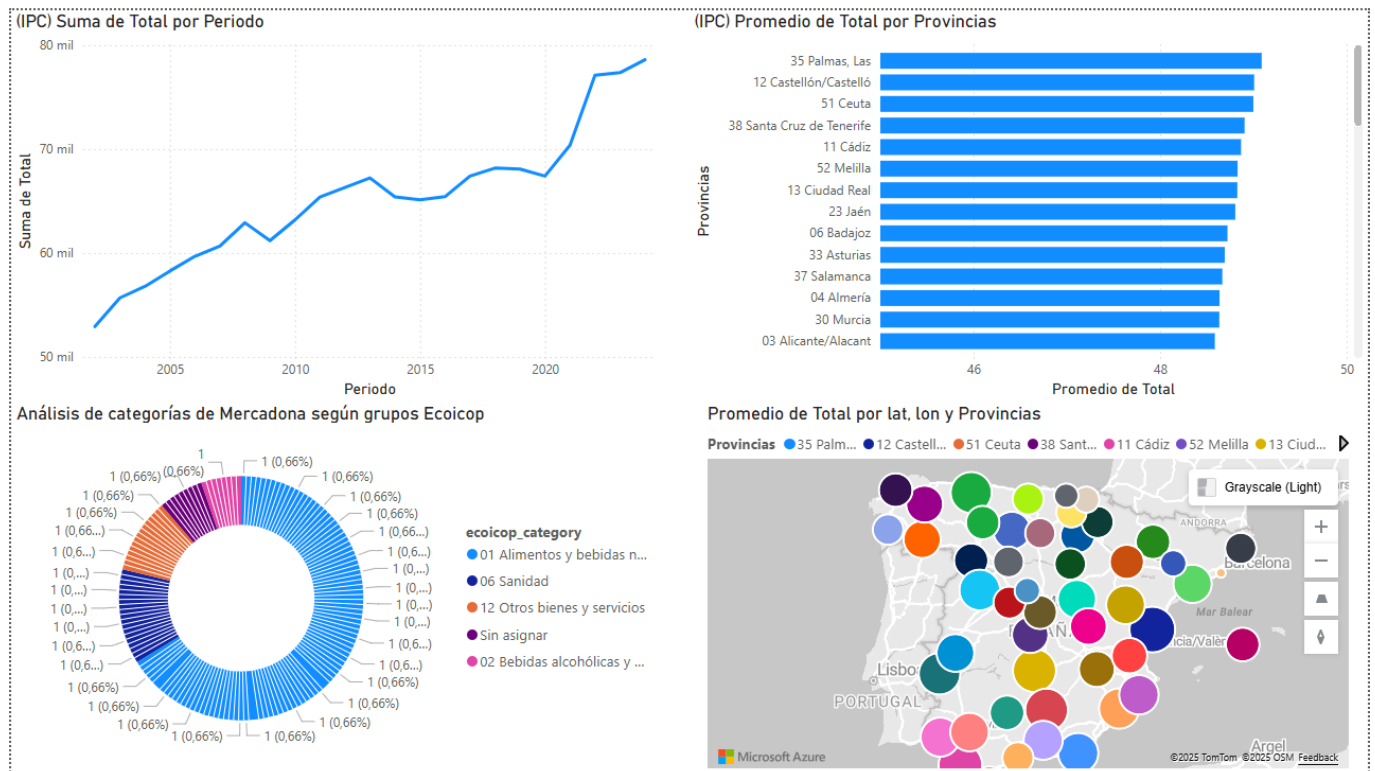


1. Scatterplot de PowerBI con promedio de precios según su media de precio y su precio de referencia, con lo que podemos ver visualmente las categorías que varían mas en cuanto a estos dos campos.
2. Script visual de Python que gráfica el Top productos más caros de mercadona.
3. Grafico circular de PowerBI que gráfica porcentualmente cuanto suma el total de precios de los productos respecto su categoría, donde observamos una predominancia absoluta en los mariscos sobre todo, siguiéndolo el pescado y el perfume y colonia
4. Esta ultima gráfica es una gráfica de líneas de PowerBI donde podemos ver los productos que mas han variado su precio de referencia según el año, donde resalta sobre todo el precio de las especias.



## Elementos visuales – Informe 2

En el segundo informe hemos graficado sobre todo en los datos del IPC nacional desde 2002 hasta 2024, aunque, graficaremos también con la correlación de la categoría ecoicop y la categoría del df de mercadona. Teniendo de resultado estos elementos visuales:



1. Gráfica de líneas de PowerBI, graficamos como han subido la suma del total del IPC según el período (año), podemos destacar una subida muy considerable, lo que induce a inflación.
2. Gráfico de barras apiladas de PowerBI, en el comparamos del promedio de Total en todas las categorías ecoicop agrupado por provincia, donde distinguimos a Las palmas como el Top 1.
3. Gráfico de aro de PowerBI, podemos ver que casi el 65% de todos sus productos corresponden a la categoría de Alimentos y bebidas y sobre un 10% a Sanidad.
4. Mapa de Azure, en el presentamos el total de la provincia pero visualmente sobre el mapa, donde dependiendo del total, crea un círculo más grande o más pequeño.

# Spark-HDFS

Hemos Ejecutado unas graficas con el dataset JSON de IPC dentro del cluster de hadoop del cesga, el código lo he hecho desde el propio jupyter lab instalado en el clúster:

```

In [ ]: if __name__ == '__main__':
        spark = SparkSession \
            .builder \
            .appName('IPC_2002-2024_085') \
            .getOrCreate()
        sc = spark.sparkContext

In [10]: ruta_hdfs = "hdfs://user/xuedua885/dataset/IPC2002-2024.json"
        df = spark.read.option("mode", "FAILFAST").json(ruta_hdfs)

In [11]: print("## Esquema del DataFrame:")
        df.printSchema()

        print("\n## Algunos datos de ejemplo (primeras 5 filas):")
        df.show(5, truncate=False)

## Esquema del DataFrame:
root
 |-- Grupos ECOICOP: string (nullable = true)
 |-- Periodo: long (nullable = true)
 |-- Provincias: string (nullable = true)
 |-- Tipo de dato: string (nullable = true)
 |-- Total: double (nullable = true)
 |-- lat: double (nullable = true)
 |-- lon: double (nullable = true)

## Algunos datos de ejemplo (primeras 5 filas):
+-----+-----+-----+-----+-----+-----+
|Grupos ECOICOP|Periodo|Provincias|Tipo de dato|Total|lat|lon|
+-----+-----+-----+-----+-----+-----+
|Indice general|2024|01 Araba/Alava|Media anual|114.19|42.8448722|-2.6828829|
|Indice general|2023|01 Araba/Alava|Media anual|111.378|42.8448722|-2.6828829|
|Indice general|2022|01 Araba/Alava|Media anual|108.092|42.8448722|-2.6828829|
|Indice general|2021|01 Araba/Alava|Media anual|100.0|42.8448722|-2.6828829|
|Indice general|2020|01 Araba/Alava|Media anual|96.923|42.8448722|-2.6828829|
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

--- Consultas Interesantes ---

In [14]: # 1. Evolución del IPC 'Índice general' (Media anual) para '15 Coruña, A' - 5 periodos más recientes.
        print("\n## 1. Evolución del IPC 'Índice general' (Media anual) para '15 Coruña, A' (5 más recientes):")
        evolucion_ipc_coru = df.filter(
            (F.col("Provincias") == "15 Coruña, A")

```

Al ejecutarlo con el comando de spark-submit vemos los logs directamente y los resultados esperados:

```

[xuedua885@cdh61-login3 ~]$ spark-submit --driver-memory 4g --executor-memory 2g --num-executors 4 datasets/ipc.py

## Esquema del DataFrame:
root
 |-- Grupos ECOICOP: string (nullable = true)
 |-- Periodo: long (nullable = true)
 |-- Provincias: string (nullable = true)
 |-- Tipo de dato: string (nullable = true)
 |-- Total: double (nullable = true)
 |-- lat: double (nullable = true)
 |-- lon: double (nullable = true)

## Algunos datos de ejemplo (primeras 5 filas):
+-----+-----+-----+-----+-----+-----+
|Grupos ECOICOP|Periodo|Provincias|Tipo de dato|Total|lat|lon|
+-----+-----+-----+-----+-----+-----+
|Indice general|2024|01 Araba/Alava|Media anual|114.19|42.8448722|-2.6828829|
|Indice general|2023|01 Araba/Alava|Media anual|111.378|42.8448722|-2.6828829|
|Indice general|2022|01 Araba/Alava|Media anual|108.092|42.8448722|-2.6828829|
|Indice general|2021|01 Araba/Alava|Media anual|100.0|42.8448722|-2.6828829|
|Indice general|2020|01 Araba/Alava|Media anual|96.923|42.8448722|-2.6828829|
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

## 1. Evolución del IPC 'Índice general' (Media anual) para '15 Coruña, A' (5 más recientes):
+-----+-----+
|Periodo|Total|
+-----+-----+
|2024|115.693|
|2023|112.362|
|2022|108.481|
|2021|100.0|
|2020|96.755|
+-----+-----+
only showing top 5 rows

## 2. Top 5 'Grupos ECOICOP' por IPC (Media anual) en 2023 para '15 Coruña, A' (excl. Índice general):
+-----+-----+
|Grupos ECOICOP|Total|
+-----+-----+
|01 Alimentos y bebidas no alcohólicas|123.1|
|07 Transporte|113.195|
|11 Restaurantes y hoteles|112.509|
|02 Bebidas alcohólicas y tabaco|112.583|
|05 Muebles, artículos del hogar y artículos para el mantenimiento corriente del hogar|112.522|
+-----+-----+
only showing top 5 rows

```

```
## 2. Top 5 'Grupos ECOICOP' por IPC (Media anual) en 2023 para '15 Coruña, A' (excl. Índice general):
+-----+-----+
| Grupos ECOICOP | Total |
+-----+-----+
| 01 Alimentos y bebidas no alcohólicas | 123.1 |
| 07 Transporte | 113.195 |
| 11 Restaurantes y hoteles | 112.589 |
| 02 Bebidas alcohólicas y tabaco | 112.583 |
| 05 Muebles, artículos del hogar y artículos para el mantenimiento corriente del hogar | 112.522 |
+-----+-----+
only showing top 5 rows

## 3. Años con mayor y menor 'Variación de las medias anuales' del 'Índice general' en '15 Coruña, A':
### Dato del Año con MAYOR variación:
(' Período:', 2022, ', Variaci\xc3\xbb3n (Total):', 8.5)

### Dato del Año con MENOR variación:
(' Período:', 2015, ', Variaci\xc3\xbb3n (Total):', -0.6)

## 5. Comparación IPC 'Índice general' (Media anual) en '15 Coruña, A' entre 2002 y 2024:
(' IPC en 2002: ', 70.412)
(' IPC en 2024: ', 115.693)
Incremento porcentual (2002-2024): 64.31%
SparkSession detenida.
[xuedua085@cdh61-login3 ~]$
```

Para hacer la ejecución correctamente hemos tenido que añadirle estas líneas para que tome correctamente el texto y así pueda entender la codificación UTF-8 (los acentos) con el carácter de acento, ya que sino nos lo toma como ascii y rompe el programa.

Estas líneas se añaden justo después de el bloque de los imports en el se settea como codificación por defecto.

```
reload(sys)
sys.setdefaultencoding('utf-8')
locale.setlocale(locale.LC_ALL, 'es_ES.UTF-8')
```