Notebook Obligatorio 315622 - 208166 - 338835 - V2

July 1, 2024

1 Obligatorio TAD

- 315622: Juan Assandri
- 338835: Oliver Kaminski
- 208166: Felipe Burgos

1.1 Introduccion

En el mundo de los negocios las empresas compiten diariamente para brindar la oferta de sus servicios mejor de lo que su competidor hace, buscando aumentar sus ingresos y disminuir sus costos, tratando de mantener una oferta vigente según las leyes de mercado actual e incluso con miras y objetivos claros para insertarse en nichos de mercado no explotados o simplemente expandiéndose o desarrollándose en los actuales.

PedidosYa no es la excepción a lo que cualquier otra empresa busca, maximizar sus ganancias, se encuentra en un proceso de expansión con un objetivo claro en su estrategia de crecimiento, desarrollando su verticalidad en diferentes sectores. Su modelo de ingresos se basa en comisiones, por lo que, a mayor cantidad de órdenes, mayor será su ingreso. Actualmente, el comportamiento de negocio en la rama de restaurantes muestra dos franjas horarias con alta demanda, uno al mediodía y otro por la noche. Para suavizar estos picos y aumentar la demanda en horarios de menor volumen de pedidos, se han incorporado supermercados y otros sectores. Uno de los principales desafíos es mantener los altos costos de la plataforma, para esto la empresa busca optimizar la logística y así mejorar la rentabilidad, con un enfoque en el desarrollo de la vertical de supermercados. Esta estrategia no solo busca distribuir mejor los costos, sino también aumentar la cantidad de compras de cada usuario y mejorar su experiencia general en la plataforma.

Entender y clasificar los diferentes tipos de órdenes es fundamental para cualquier empresa, ya que permite diversificar su oferta de productos y servicios de manera efectiva. Como resultado, la empresa puede incrementar su base de usuarios y generar más oportunidades de ingresos.

Además, con una buena clasificación de los tipos de órdenes, se logra una segmentación de mercado mucho más efectiva y una comprensión mayor del comportamiento del cliente. Con esta información, las empresas pueden diseñar campañas de marketing y promociones personalizadas que resuenen mejor con cada consumidor del servicio. Ofrecer experiencias de compra adaptadas a las necesidades específicas de cada tipo de orden aumenta considerablemente la satisfacción y la fidelidad de los clientes. Para PedidosYa, identificar y analizar diferentes tipos de órdenes en la vertical

de supermercados es crucial para potenciar esta línea de negocio. Entender estos patrones de compra permite a PedidosYa ajustar su oferta de productos, asegurando que siempre haya un surtido relevante y conveniente disponible para sus usuarios. Además, al comprender los diferentes tipos de órdenes, PedidosYa puede identificar áreas con alta demanda que no están siendo adecuadamente satisfechas.

En conclusión, el desafío de identificar diferentes tipos de órdenes no sólo es fundamental para mejorar la oferta actual de PedidosYa, sino que también es una estrategia clave para descubrir nuevas oportunidades de crecimiento y desarrollo en el competitivo sector de supermercados. Esto permitirá a PedidosYa seguir consolidando su posición en el mercado y ofrecer un servicio de mayor valor a sus usuarios.

1.2 Objetivo

Identificar diferentes tipos de órdenes para detectar oportunidades de desarrollo del negocio de supermercados

1.3 Links de referencia

1. **Pandas**:
   - Pandas Documentation
2. **Seaborn**:
   - Seaborn Documentation
3. **Matplotlib**:
   - Matplotlib Documentation
4. **Scikit-learn**:
   - Scikit-learn Documentation
   - KMeans
   - MinMaxScaler
   - Silhouette Score
5. **Gower**:
   - Gower's Distance
6. **KMedoids**:
   - KMedoids Clustering
7. **Yellowbrick**:
   - Yellowbrick Documentation
   - KElbowVisualizer
   - SilhouetteVisualizer

1.4 Instalacion e importacion de las librerias

[1]:
```
#pip install --upgrade pip
```

[2]:
```
#pip cache purge
```

[3]:
```
!pip install scikit-learn-extra
!pip install gower
!pip install kmedoids
!pip install yellowbrick
!pip install matplotlib==3.4.3
```

```
Requirement already satisfied: scikit-learn-extra in
/opt/conda/lib/python3.7/site-packages (0.3.0)
Requirement already satisfied: scipy>=0.19.1 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn-extra) (1.7.3)
Requirement already satisfied: scikit-learn>=0.23.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn-extra) (1.0.2)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn-extra) (1.21.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.23.0->scikit-learn-
extra) (3.0.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn>=0.23.0->scikit-learn-extra) (1.1.0)
Requirement already satisfied: gower in /opt/conda/lib/python3.7/site-packages
(0.1.2)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages
(from gower) (1.21.6)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages
(from gower) (1.7.3)
Requirement already satisfied: kmedoids in /opt/conda/lib/python3.7/site-
packages (0.3.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages
(from kmedoids) (1.21.6)
Requirement already satisfied: yellowbrick in /opt/conda/lib/python3.7/site-
packages (1.5)
Requirement already satisfied: cycler>=0.10.0 in /opt/conda/lib/python3.7/site-
packages (from yellowbrick) (0.11.0)
Requirement already satisfied: scikit-learn>=1.0.0 in
/opt/conda/lib/python3.7/site-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.7/site-
packages (from yellowbrick) (1.7.3)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/opt/conda/lib/python3.7/site-packages (from yellowbrick) (3.4.3)
Requirement already satisfied: numpy>=1.16.0 in /opt/conda/lib/python3.7/site-
packages (from yellowbrick) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.7/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.7/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.6)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.7/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.3.2)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-
packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (8.4.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn>=1.0.0->yellowbrick) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn>=1.0.0->yellowbrick)
(3.0.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-
packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(1.16.0)
Requirement already satisfied: matplotlib==3.4.3 in
/opt/conda/lib/python3.7/site-packages (3.4.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-
packages (from matplotlib==3.4.3) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib==3.4.3) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.7/site-packages (from matplotlib==3.4.3) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-
packages (from matplotlib==3.4.3) (8.4.0)
Requirement already satisfied: numpy>=1.16 in /opt/conda/lib/python3.7/site-
packages (from matplotlib==3.4.3) (1.21.6)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib==3.4.3) (3.0.6)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-
packages (from python-dateutil>=2.7->matplotlib==3.4.3) (1.16.0)
```

[4]:
```
import pandas as pd
import os # Obtener directorios de trabajo
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.preprocessing import scale

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MinMaxScaler

import gower
```

```
from gower import gower_matrix

import kmedoids
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer
from sklearn_extra.cluster import KMedoids
```

1.5 Seteo de directorio de trabajo

[5]:
```
# obtengo el directorio de trabajo actual
os.getcwd()
```

[5]:
```
'/home/jovyan/work/src'
```

[6]:
```
# seteo y verifico el directorio de trabajo en la raíz del proyecto
os.chdir(r'/home/jovyan/work') # se incluye 'r' antes de la dirección para␣
 ↪atender adecuadamente a los caracteres especiales

print('El nuevo -actual- directorio de trabajo es: ', os.getcwd())
```

```
El nuevo -actual- directorio de trabajo es:  /home/jovyan/work
```

2 Analisis Exploratyorio de los datos

2.1 Importacion de datasets

[7]:
```
# Cargar los archivos CSV en dataframes
df1 = pd.read_csv(r'data/dataset1.csv')
df2 = pd.read_csv(r'data/dataset2.csv')
```

3 Analisis Exploratorio de los datos

3.1 Dataframe 1

[8]:
```
df1.head()
```

[8]:

	order_id	weekday	hour	\
0	ZbyJUv9idjhjMv9qc9PAKIe8d9QqOtR0ZB4yajPGAt4=	Tuesday	23	
1	FHWMJpivzLpIv2hzISgOqX87SmPPWMVD3Z1xWAqYbxU=	Monday	8	
2	PgZPV2+ewV78vSWDOUBpukyZT3JoZPKEhzlNu5qWLTA=	Saturday	23	
3	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	
4	TMelBjTKbPOk5KQKhFQ23fmUCL++Emdo792rQCWwLns=	Friday	21	

```
  business_type_name                                partner_id  \
0             Kiosks  dtZsgbcbOVA2FQQRxnFBihy3O697FEFX3OsxmeIs7dA=
1             Market  nnnnu1XWZ8/rhXlbuClEzWdIyB6WxqzVU4R1O5WTSvU=
2             Kiosks  rmLHOHBXmepAsF/EOhKCWlsHEphxcTeJuOtVsWgzc+E=
3             Kiosks  vXtLI7qWnGDd4ZTQOiZJZ/wQXXY1GLgkEE+pErgYcCc=
4             Market  CX5eIMrz9/Vk6rPyka9r0Yhnf6MtJyS02BOU2JlXl3M=

                                      user_id  qty_total_products  \
0  SvyuKJwQSZ3FLE1kp4hd6w6r0G2Y2rOoVyaPbGzO2ls=                   1
1  Sw2a18waMqetNdc7gKPPEVfLZICJGkwz69Uxz+gdUb8=                   1
2  Sw5iS5ffSt/y3+kDtdOQbrc7Gan0c8NSEUkIbgZOuPE=                   1
3  SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=                   1
4  SwFxPMx2bMVcPOYsQhlAAhqG4//pL5dDTdWyfVJ7KjY=                   1

   total_amount  has_discount
0      8.404327         False
1     27.370810         False
2     13.432194         False
3      8.049295         False
4     21.998182         False
```

Estructura y Tamaño del DataFrame

El comando print("Shape of data orders:", df1.shape) muestra el tamaño del DataFrame, mientras que df1.info() proporciona detalles sobre el tipo de datos y la cantidad de valores no nulos en cada columna.

[9]:
```
print("Shape of data orders:",df1.shape)
df1.info()
```

```
Shape of data orders: (1140899, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1140899 entries, 0 to 1140898
Data columns (total 9 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   order_id            1140899 non-null  object
 1   weekday             1140899 non-null  object
 2   hour                1140899 non-null  int64
 3   business_type_name  1140899 non-null  object
 4   partner_id          1140899 non-null  object
 5   user_id             1140899 non-null  object
 6   qty_total_products  1140899 non-null  int64
 7   total_amount        1140899 non-null  float64
 8   has_discount        1140899 non-null  bool
dtypes: bool(1), float64(1), int64(2), object(5)
memory usage: 70.7+ MB
```

Análisis Complementario de la Distribución de Datos mediante Boxplots y Resumen

Estadístico

Los boxplots, en conjunto con el resumen estadístico de las variables numéricas, proporcionan una visión integral de la distribución de los datos. Los boxplots ilustran la mediana, los cuartiles y los valores atípicos de las distribuciones de qty_total_products y total_amount. Estos gráficos ayudan a identificar rápidamente la dispersión y la presencia de valores extremos, complementando las estadísticas descriptivas que detallan la media, desviación estándar y percentiles. En el caso de los datos presentados, ambos métodos evidencian que existen valores atípicos significativos, lo cual es crucial para decisiones sobre el tratamiento de estos valores en análisis posteriores.

Aqui observamos que existen valores atipicos quer podriamos tratar antes de nustro analisis final

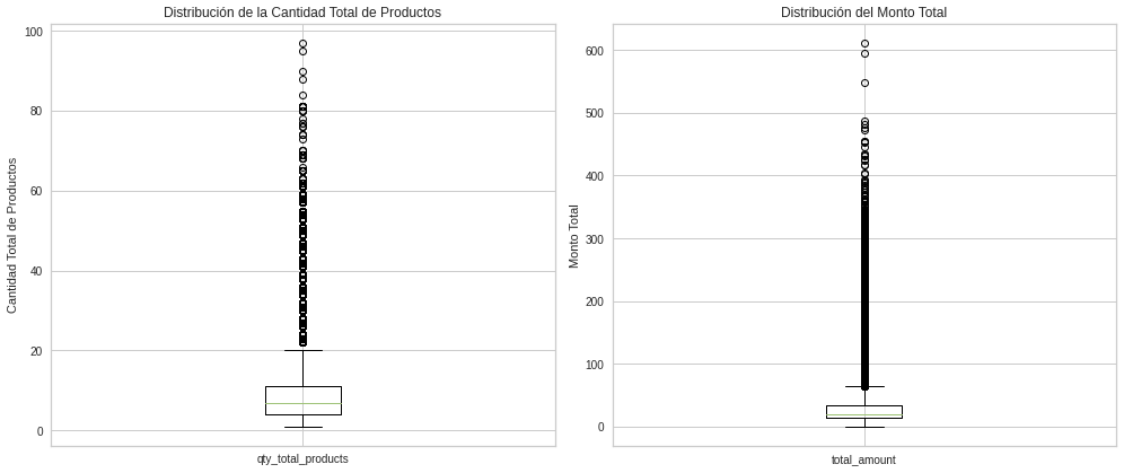
[10]:
```
# Para forzar que no se muestre en notación cientifica
pd.options.display.float_format = '{:.2f}'.format

df1.describe(percentiles=[.25, .5, .75, 0.99])
```

[10]:

	hour	qty_total_products	total_amount
count	1140899.00	1140899.00	1140899.00
mean	16.18	8.42	28.01
std	4.30	7.36	25.09
min	0.00	1.00	0.00
25%	13.00	4.00	13.18
50%	17.00	7.00	19.66
75%	20.00	11.00	33.66
99%	23.00	38.00	128.01
max	23.00	97.00	611.78

[11]:
```
# Crear una figura con dos subplots
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Crear box plot para 'qty_total_products'
axs[0].boxplot(df1['qty_total_products'])
axs[0].set_title('Distribución de la Cantidad Total de Productos')
axs[0].set_ylabel('Cantidad Total de Productos')
axs[0].set_xticks([1])
axs[0].set_xticklabels(['qty_total_products'])

# Crear box plot para 'total_amount'
axs[1].boxplot(df1['total_amount'])
axs[1].set_title('Distribución del Monto Total')
axs[1].set_ylabel('Monto Total')
axs[1].set_xticks([1])
axs[1].set_xticklabels(['total_amount'])

# Ajustar el layout
plt.tight_layout()
```

```
# Mostrar la gráfica
plt.show()
```



Análisis Descriptivo de Variables Categóricas y Booleanas

resumen estadístico para las columnas categóricas y booleanas del DataFrame, mostrando el conteo, número de valores únicos, valor más frecuente (moda) y su frecuencia. Este análisis ayuda a identificar la distribución de categorías y la prevalencia de ciertos valores, como se observa en las columnas order_id, weekday, business_type_name, partner_id, user_id y has_discount.

Descripción de los Resultados:

- order_id: Cada pedido tiene un identificador único (unique=1140899).
- weekday: Hay 7 días de la semana, con el sábado siendo el más común (freq=173061).
- business_type_name: Hay 3 tipos de negocios, siendo el mercado (Market) el más frecuente (freq=896193).
- partner_id: Hay 3367 socios, con el socio más frecuente teniendo 19045 órdenes.
- user_id: Existen 199999 usuarios únicos, con el más frecuente teniendo 302 órdenes.
- has_discount: Es una variable booleana con 2 valores (True y False), siendo True el valor más común (freq=731184).

[12]:
```
df1.describe(include=['object', 'bool'])
```

[12]:
```
                                        order_id   weekday  \
count                                    1140899   1140899
unique                                   1140899         7
top     ZbyJUv9idjhjMv9qc9PAKIe8d9QqOtROZB4yajPGAt4=  Saturday
freq                                           1    173061

       business_type_name                                    partner_id  \
count             1140899                                       1140899
unique                  3                                          3367
top                Market  CSq+DtmiVKcNLprPiiWNTgH5sg97DimLHyswaz3tA5M=
```

```
freq                896193                                              19045

                                                    user_id has_discount
count                                               1140899      1140899
unique                                               199999            2
top     Hm0t3URT7FulDsq9V00Rw7lpy/PLY2K2BFnTTjaiLbY=          True
freq                                                    302       731184
```

Convertir la columna 'weekday'y 'business_type_name' a tipo categórico

[13]:
```
# Convertir la columna 'weekday'y 'business_type_name' a tipo categórico
df1['weekday'] = df1['weekday'].astype('category')
df1['business_type_name'] = df1['business_type_name'].astype('category')
```

[14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1140899 entries, 0 to 1140898
Data columns (total 9 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   order_id            1140899 non-null  object
 1   weekday             1140899 non-null  category
 2   hour                1140899 non-null  int64
 3   business_type_name  1140899 non-null  category
 4   partner_id          1140899 non-null  object
 5   user_id             1140899 non-null  object
 6   qty_total_products  1140899 non-null  int64
 7   total_amount        1140899 non-null  float64
 8   has_discount        1140899 non-null  bool
dtypes: bool(1), category(2), float64(1), int64(2), object(3)
memory usage: 55.5+ MB
```

[15]:
```
df1['weekday'].cat.categories
```

[15]:
```
Index(['Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday',
       'Wednesday'],
      dtype='object')
```

[16]:
```
df1['business_type_name'].cat.categories
```

[16]:
```
Index(['Kiosks', 'Market', 'Shop'], dtype='object')
```

Distribución de Órdenes por Tipo de Negocio

El agrupamiento por business_type_name y el conteo de órdenes en cada grupo proporcionan una visión clara de la cantidad de órdenes asociadas a cada tipo de negocio.

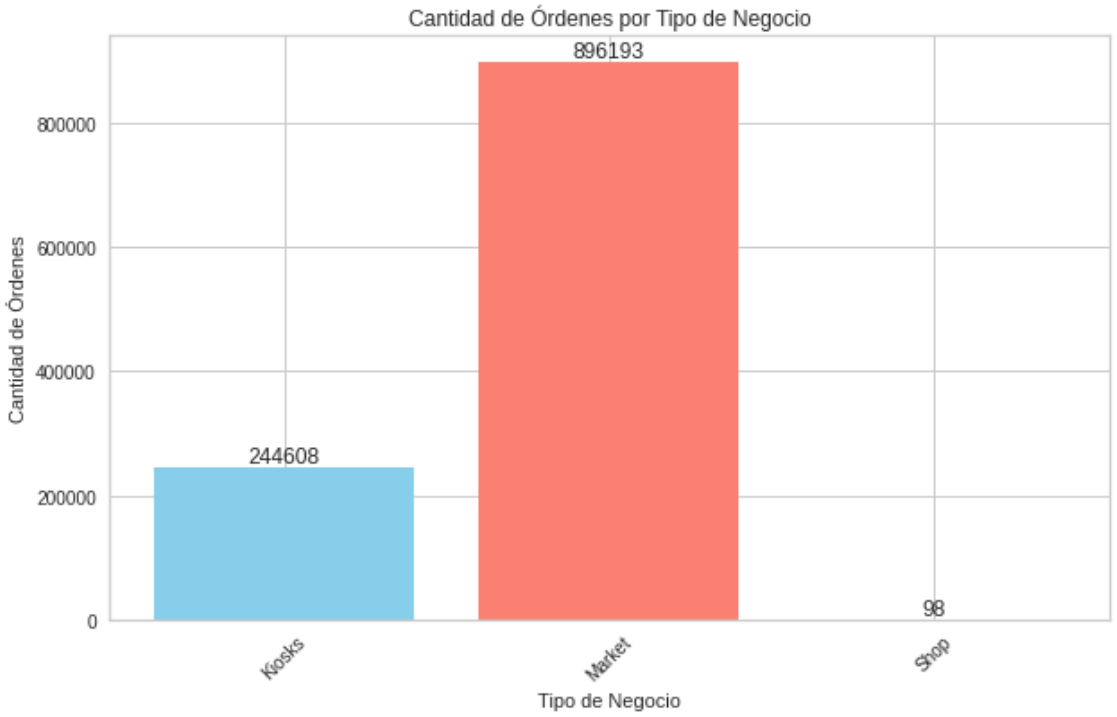
Los resultados indican que la mayoría de las órdenes provienen de Markets (896,193), seguidas de

Kiosks (244,608) y Shops (98). Esta variación significativa en la cantidad de órdenes sugiere que hay patrones de consumo distintos entre los tipos de negocio, lo cual es favorable para la clusterización.

[17]:
```
# Agrupar por 'business_type_name' y contar la cantidad de órdenes en cada grupo
resumen_tipo_negocio = df1.groupby('business_type_name').agg(
    cantidad=('business_type_name', 'count')
).reset_index()

# Mostrar los primeros 100 registros del resultado
print(resumen_tipo_negocio.head(100))
```

```
  business_type_name  cantidad
0              Kiosks    244608
1              Market    896193
2                Shop        98
```

[18]:
```
# Crear una gráfica de barras
plt.figure(figsize=(10, 6))
plt.bar(resumen_tipo_negocio['business_type_name'],␣
 ↪resumen_tipo_negocio['cantidad'], color=['skyblue', 'salmon', 'lightgreen'])
plt.xlabel('Tipo de Negocio')
plt.ylabel('Cantidad de Órdenes')
plt.title('Cantidad de Órdenes por Tipo de Negocio')
plt.xticks(rotation=45)

# Mostrar los valores en las barras
for index, value in enumerate(resumen_tipo_negocio['cantidad']):
    plt.text(index, value, str(value), ha='center', va='bottom')

# Mostrar la gráfica
plt.show()
```



Análisis de Órdenes con y sin Descuento

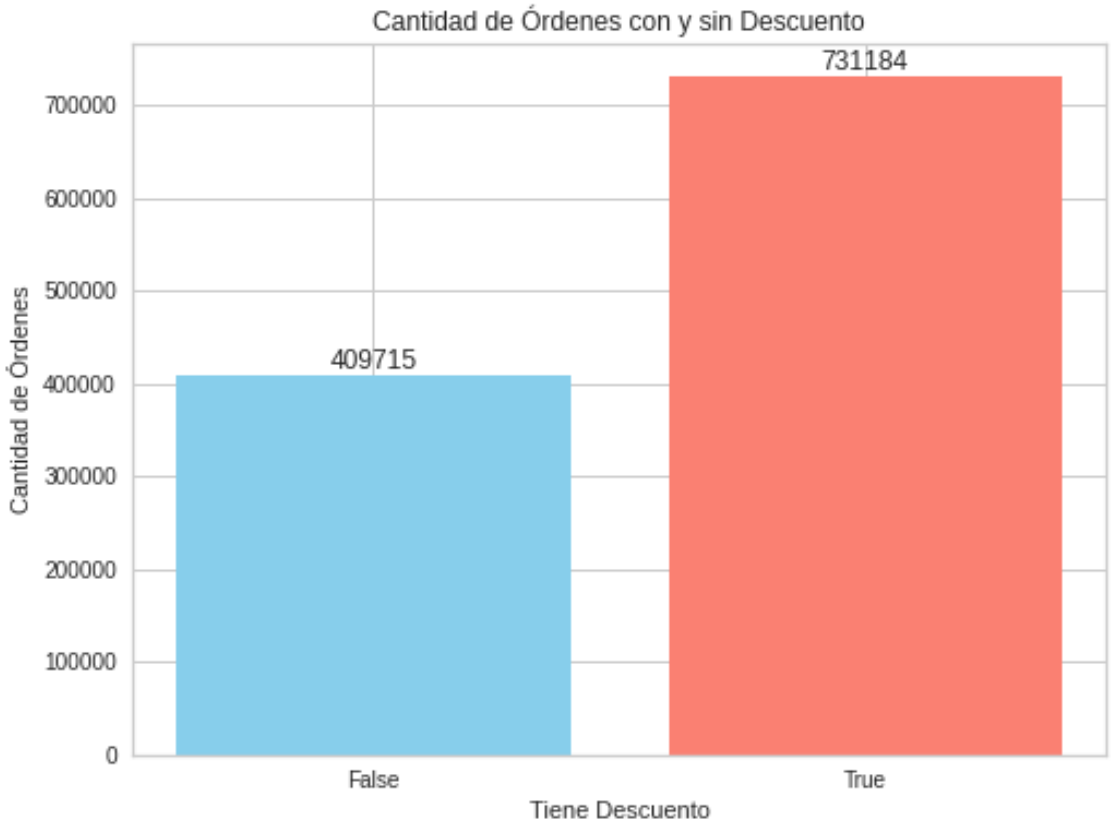
El análisis de la distribución de órdenes según el uso de descuentos proporciona una visión clara de cómo los descuentos afectan el comportamiento de compra de los clientes.Estos resultados son favorables para la clusterización porque permiten segmentar a los clientes en grupos basados en su sensibilidad a los descuentos. Esta información puede ser utilizada para crear clusters que diferencien entre compradores regulares y aquellos que compran principalmente cuando hay descuentos, optimizando así las campañas de marketing y promociones personalizadas.

Este análisis muestra que una proporción significativa de las órdenes (731,184) utilizan descuentos, lo cual es más del doble de las órdenes sin descuento. Esto indica que los descuentos son un factor importante en la decisión de compra de los clientes

[19]:
```
# Agrupar por 'has_discount' y contar la cantidad de órdenes en cada grupo
resumen_descuento = df1.groupby('has_discount').agg(
    cantidad=('has_discount', 'count')
).reset_index()

# Mostrar los primeros 100 registros del resultado (aunque en este caso, probablemente solo haya 2)
print(resumen_descuento.head(100))
```

```
   has_discount  cantidad
0         False    409715
1          True    731184
```

[20]:
```
# Crear una gráfica de barras
plt.figure(figsize=(8, 6))
plt.bar(resumen_descuento['has_discount'].astype(str),␣
 ↪resumen_descuento['cantidad'], color=['skyblue', 'salmon'])
plt.xlabel('Tiene Descuento')
plt.ylabel('Cantidad de Órdenes')
plt.title('Cantidad de Órdenes con y sin Descuento')
plt.xticks(rotation=0)

# Mostrar los valores en las barras
for index, value in enumerate(resumen_descuento['cantidad']):
    plt.text(index, value, str(value), ha='center', va='bottom')

# Mostrar la gráfica
plt.show()
```



Distribución de Órdenes por Día de la Semana

El análisis de la distribución de órdenes por día de la semana revela el patrón de actividad de los clientes y ayuda a identificar los días de mayor y menor demanda.

Los resultados indican que, aunque hay ligeras variaciones, la cantidad de órdenes es relativamente

similar cada día de la semana. El día con mayor cantidad de órdenes es el sábado (173,061) y el día con menor cantidad es el jueves (155,546), pero la diferencia no es muy significativa.

La consistencia en el volumen de órdenes por día de la semana sugiere que los patrones de compra no varían drásticamente a lo largo de la semana. Sin embargo, esta información puede todavía ser útil para la clusterización, ya que permite segmentar a los clientes según hábitos de compra más sutiles y consistentes, optimizando estrategias de servicio y disponibilidad de productos uniformemente a lo largo de la semana.

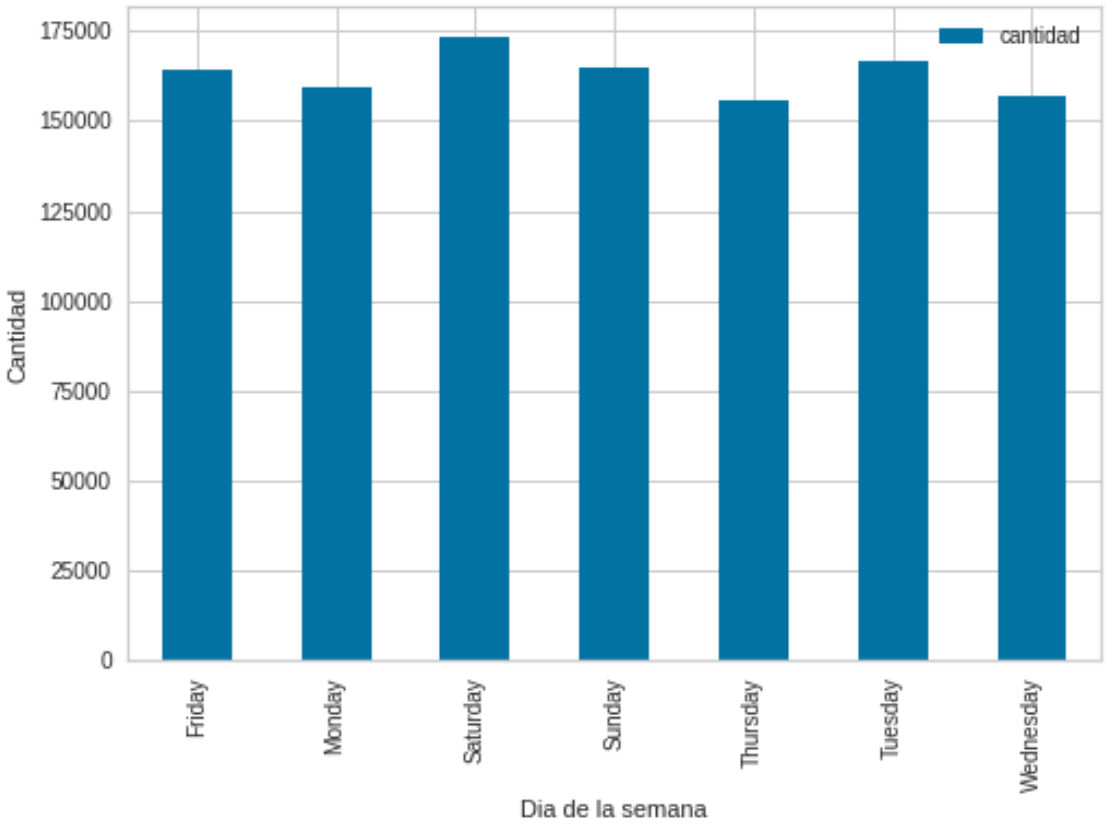
[21]:
```
df1.groupby('weekday').agg(
    cantidad=('weekday', 'count')
).reset_index().head(100)
```

[21]:

	weekday	cantidad
0	Friday	164208
1	Monday	159450
2	Saturday	173061
3	Sunday	164929
4	Thursday	155546
5	Tuesday	166655
6	Wednesday	157050

[22]:
```
df1.groupby('weekday').agg(
    cantidad=('weekday', 'count')
).plot(kind='bar', xlabel='Dia de la semana', ylabel='Cantidad')
```

[22]:
```
<AxesSubplot:xlabel='Dia de la semana', ylabel='Cantidad'>
```



3.2 Dataframe 2

Estructura y Tamaño del DataFrame El comando print("Shape of data orders:", df1.shape) muestra el tamaño del DataFrame, mientras que df1.info() proporciona detalles sobre el tipo de datos y la cantidad de valores no nulos en cada columna.

[23]:
```
df2.head()
```

[23]:

	order_id	product_id	level_one
0	ACJR23rpM9HpqSc3pGJ+mq5X9zNFmhSzfSoFVKpteP8=	SArns9ZSXPvbm8OUEhJ2mM3jpYHqzSm2igKfMPEENKs=	Beverages
1	ACJR23rpM9HpqSc3pGJ+mq5X9zNFmhSzfSoFVKpteP8=	H+Bl0ytpbrTPXc1BrleXXvqqsGWfkQMCqVWuBgl1fsU=	Snacks
2	ACJR23rpM9HpqSc3pGJ+mq5X9zNFmhSzfSoFVKpteP8=	5GkVgmMP+Xme/VSAjXpfwlgZanZkMaLQ1IZrb+HHbck=	Beverages
3	ACJR23rpM9HpqSc3pGJ+mq5X9zNFmhSzfSoFVKpteP8=	rKStVkhTq3jHNteYX+WgrykUVAFDwsnDWBBB5Ccrm44=	Bread / Bakery
4	ACJR23rpM9HpqSc3pGJ+mq5X9zNFmhSzfSoFVKpteP8=		

```
4  I4c44vZHw+EGk+OE5NjPDnmWoS+KbBwLy5+VST51eks=  Dairy / Chilled / Eggs

              level_two
0                 Water
1           Confectionary
2  Soft Drinks / Mixers
3                 Bread
4           Dairy / Eggs
```

[24]:
```
print("Shape of data orders:",df2.shape)
df2.info()
```

```
Shape of data orders: (5036344, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5036344 entries, 0 to 5036343
Data columns (total 4 columns):
 #   Column      Dtype
---  ------      -----
 0   order_id    object
 1   product_id  object
 2   level_one   object
 3   level_two   object
dtypes: object(4)
memory usage: 153.7+ MB
```

Análisis Descriptivo de Variables Categóricas y Booleanas

resumen estadístico para las columnas categóricas y booleanas del DataFrame, mostrando el conteo, número de valores únicos, valor más frecuente (moda) y su frecuencia. Este análisis ayuda a identificar la distribución de categorías y la prevalencia de ciertos valores.

Resultados:

- order_id: 1,129,555 órdenes únicas
- product_id: 898,075 productos únicos
- level_one: 14 categorías únicas
- level_two: 53 subcategorías únicas
- order_id más frecuente: 53 ocurrencias
- product_id más frecuente: 1051 ocurrencias
- level_one más frecuente: Snacks (869,319 ocurrencias)
- level_two más frecuente: Confectionary (682,787 ocurrencias)

Los resultados indican una gran diversidad de productos y categorías, con Snacks y Confectionary siendo las más comunes en sus respectivos niveles.Estos resultados son favorables para la clusterización ya que permiten segmentar los datos según la popularidad y diversidad de los productos.

[25]:
```
# Para forzar que no se muestre en notación cientifica
pd.options.display.float_format = '{:.2f}'.format

df2.describe()
```

[25]:

	order_id	product_id	level_one	level_two
count	5036344	5036344	5036344	5036344
unique	1129555	898075	14	53
top	eRnrYFqWzICTE8AQuqz8BDMzOTVtMNLUUg02R1d2oM8=	S2Psa75x16XpYY6AIIprddLBHV0F5TsPMIv3odT7AQ0=	Snacks	Confectionary
freq	53	1051	869319	682787

[26]:
```
# Convertir la columna 'level_one'y 'level_two' a tipo categórico
df2['level_one'] = df2['level_one'].astype('category')
df2['level_two'] = df2['level_two'].astype('category')
```

[27]:
```
df2['level_one'].cat.categories
```

[27]:
```
Index(['BWS', 'Beverages', 'Bread / Bakery', 'Dairy / Chilled / Eggs',
       'Frozen', 'General Merchandise', 'Home / Pet', 'Meat / Seafood',
       'Packaged Foods', 'Personal Care / Baby / Health', 'Produce',
       'Ready To Consume', 'Smoking / Tobacco', 'Snacks'],
      dtype='object')
```

[28]:
```
df2['level_two'].cat.categories
```

[28]:
```
Index(['Apparel / Footwear / Sports equipment', 'Baby', 'Beer / Cider',
       'Beverages', 'Books / Magazines', 'Bread', 'Breakfast / Spreads',
       'Canned / Jarred / Instant Meals', 'Cleaning / Laundry',
       'Confectionary', 'Cooking / Condiments / Baking / Herbs / Spices',
       'Dairy / Eggs', 'Deli / Snacking', 'Desserts', 'Disposables',
       'E-Cigarettes / Accessories', 'Electronics', 'Fish / Seafood', 'Food',
       'Frozen Convenience / Bakery', 'Frozen Fruit / Vegetables / Potato',
       'Frozen Meat / Seafood', 'Fruit', 'Hardware / Misc', 'Health',
       'Household', 'Ice', 'Ice Cream / Desserts',
       'Juice / Ice Tea / Sports / Energy', 'Meat', 'Milk', 'Other Snacks',
       'Pasta / Rice / Grains', 'Personal Care / Beauty', 'Pet', 'Poultry',
       'Pre-Mixed', 'Prepared F&V / Fresh Herbs', 'Ready Meals',
       'Salty Snacks', 'Seasonal / Occasion', 'Smoking Accessories',
       'Soft Drinks / Mixers', 'Special Diet', 'Specialty', 'Spirits',
       'Sweet Bakery', 'Tea / Coffee', 'Tobacco', 'Toys', 'Vegetables',
       'Water', 'Wine / Sparkling Wine'],
```

```
      dtype='object')
```

3.3 Seteo de semilla, k_cluster y iteraciones

[29]:
```
semilla = 123
k_cluster = 10
iteraciones = 10
tope_range = k_cluster + 1
```

3.4 Combinacion Left Join Basado en la Columna "order_id" y Eliminar Filas con Valores Nulos

Realizamos un left join entre los DataFrames df1 y df2 utilizando la columna order_id para combinar la información de ambos conjuntos de datos, asegurando que todas las filas de df1 se mantengan. Posteriormente, eliminamos las filas con valores nulos utilizando dropna() para garantizar la integridad y completitud de los datos. Este proceso es crucial para preparar un conjunto de datos limpio y cohesivo, que sea adecuado para análisis posteriores. Eliminar valores nulos ayuda a evitar problemas en análisis y modelos predictivos, asegurando que los resultados sean precisos y significativos.

[30]:
```
# Realizar el left join basado en la columna "order_id"
df = pd.merge(df1, df2, on='order_id', how='left')
df = df.dropna()
```

Conteo de Registros Distintos de order_id y user_id

El análisis del número de registros distintos para order_id y user_id proporciona una visión clara de la unicidad de los datos en el DataFrame combinado. Los resultados indican que hay 197,011 órdenes únicas en el conjunto de datos, distribuidas entre 35,078 usuarios distintos. Esto sugiere que, en promedio, cada usuario ha realizado varias órdenes.

[31]:
```
df['level_one'] = df['level_one'].astype('object')
```

[32]:
```
# Contar los registros de order_id distintos
num_distinct_order_ids = df['order_id'].nunique()
# Contar los registros de user_id distintos
num_distinct_user_ids = df['user_id'].nunique()
print("Número de registros de order_id distintos:", num_distinct_order_ids)
print("Número de registros de user_id distintos:", num_distinct_user_ids)
```

```
Número de registros de order_id distintos: 197011
Número de registros de user_id distintos: 35078
```

Estructura y Tamaño del DataFrame

El comando print("Shape of data orders:", df1.shape) muestra el tamaño del DataFrame, mientras que df1.info() proporciona detalles sobre el tipo de datos y la cantidad de valores no nulos en cada

columna.

[33]:
```
# Mostrar el resultado
df.head()
```

[33]:

	order_id	weekday	hour	business_type_name	partner_id	user_id	qty_total_products	total_amount	has_discount	product_id	level_one	level_two
3	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	Kiosks	vXtLI7qWnGDd4ZTQOiZJZ/wQXXY1GLgkEE+pErgYcCc=	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	1	8.05	False	CNOSao/6N6ofzazjxaKKpNX/uJJmYmRyEwsj0DC8FYM=	Smoking / Tobacco	Tobacco
14	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	Kiosks	jm4h8a5ugX13K8EyOwmt3XmFG+tC+GcN9Od0EX8TeLQ=	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	1	7.91	False	KnXqi39yRIczh1FaJo1LSzCJXK7K0ypgFu83U2qq8JY=	Beverages	Soft Drinks / Mixers
15	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	Kiosks	1JhXLeaP56TZ3S0s3L4ze2uF1z6TzLNNG4l+ZjrGGpc=	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	7.85	False	wm+wxfQbe8WeSIOD/2fTWWQ4PYtqLel1kNOx7Ltk/zY=	Beverages	Soft Drinks / Mixers
16	GdvT/R8JqkIOzSzfwVrVag6okmBzO0ms88KxgAseht8=	Saturday	23	Kiosks	bmaUtn62nJYTQWQDGzL/E8Ka06Mx4EhRVVfk5OQwLZQ=	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	8.60	False	6ZvoEhv/vRYiuRQhx477vpGbwLxJYbWAeXyK/yy+kwM=	Smoking / Tobacco	Tobacco
17	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	Kiosks	MLb861rH4PvXNX5DI2ODCQfEX82/0OnwHbRTaCBXD0U=	Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=	1	8.38	False	N6x+F9p/qU2YCJsTOvSq4JGVrtXKVGstCBjqBIl3Pb4=	Packaged Foods	Cooking / Condiments / Baking / Herbs / Spices

[34]:
```
print("Shape of data orders:",df1.shape)
df.info()
```

```
Shape of data orders: (1140899, 9)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 873635 entries, 3 to 1817505
Data columns (total 12 columns):
 #   Column              Non-Null Count   Dtype
```

```
---  ------                --------------   -----
 0   order_id              873635 non-null  object
 1   weekday               873635 non-null  category
 2   hour                  873635 non-null  int64
 3   business_type_name    873635 non-null  category
 4   partner_id            873635 non-null  object
 5   user_id               873635 non-null  object
 6   qty_total_products    873635 non-null  int64
 7   total_amount          873635 non-null  float64
 8   has_discount          873635 non-null  bool
 9   product_id            873635 non-null  object
 10  level_one             873635 non-null  object
 11  level_two             873635 non-null  category
dtypes: bool(1), category(3), float64(1), int64(2), object(5)
memory usage: 63.3+ MB
```

[35]:
```
df.columns
```

[35]:
```
Index(['order_id', 'weekday', 'hour', 'business_type_name', 'partner_id',
       'user_id', 'qty_total_products', 'total_amount', 'has_discount',
       'product_id', 'level_one', 'level_two'],
      dtype='object')
```

[36]:
```
ordenes_por_categoria_df_l1 = df.groupby('level_one').size().
 ↪reset_index(name='cantidad')
```

3.4.1 Análisis de Órdenes por Categoría de Producto en Level One

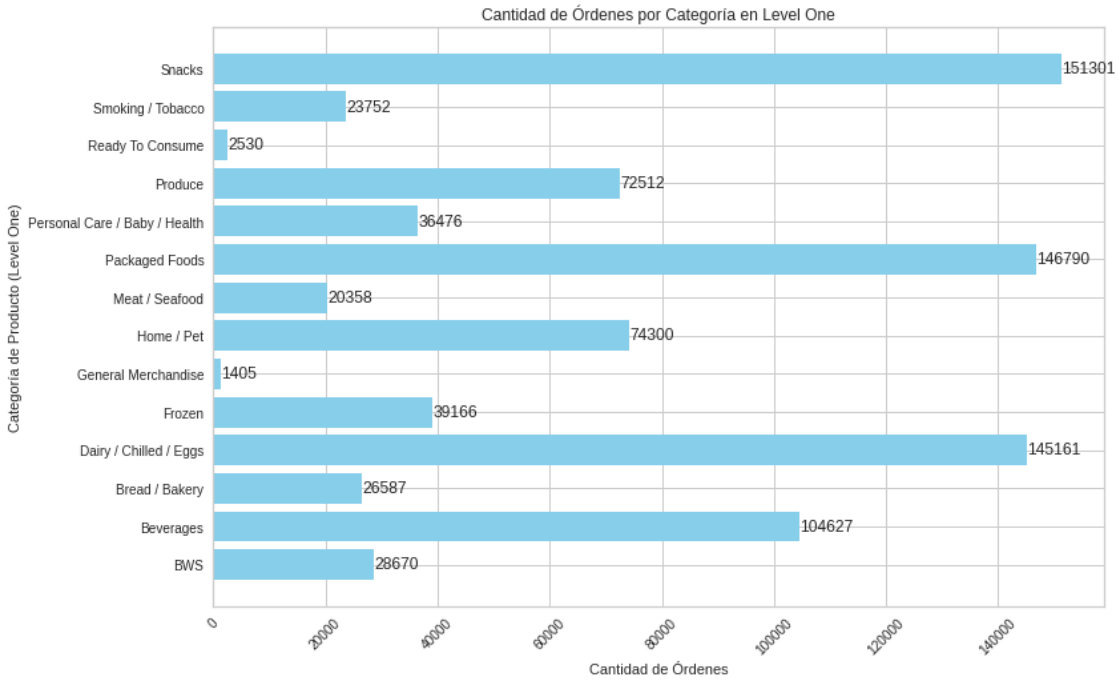
El gráfico muestra la cantidad de órdenes distribuidas por categoría de producto en level_one, proporcionando una visión clara de las preferencias de los clientes en diferentes categorías.

Los resultados indican que las categorías Snacks, Packaged Foods y Beverages son las más populares entre los clientes. Esto proporciona una comprensión clara de qué productos son más demandados, permitiendo una mejor gestión del inventario y estrategias de marketing específicas.

Estos resultados son favorables para la clusterización ya que permiten segmentar a los clientes según sus preferencias de productos. Identificar categorías más demandadas facilita la creación de clusters basados en hábitos de consumo específicos, optimizando así las ofertas y promociones personalizadas.

[37]:
```
# Crear una gráfica de barras
plt.figure(figsize=(12, 8))
plt.barh(ordenes_por_categoria_df_l1['level_one'],␣
 ↪ordenes_por_categoria_df_l1['cantidad'], color='skyblue')
plt.xlabel('Cantidad de Órdenes')
plt.ylabel('Categoría de Producto (Level One)')
plt.title('Cantidad de Órdenes por Categoría en Level One')
```

```
plt.xticks(rotation=45)

# Mostrar los valores en las barras
for index, value in enumerate(ordenes_por_categoria_df_l1['cantidad']):
    plt.text(value, index, str(value), va='center')
```



[38]:
```
# Agrupar por 'level_two' y contar la cantidad de órdenes en cada categoría
ordenes_por_categoria_df_l2 = df.groupby('level_two').size().
 ↪reset_index(name='cantidad')
```

3.4.2 Análisis de Órdenes por Categoría de Producto en Level Two

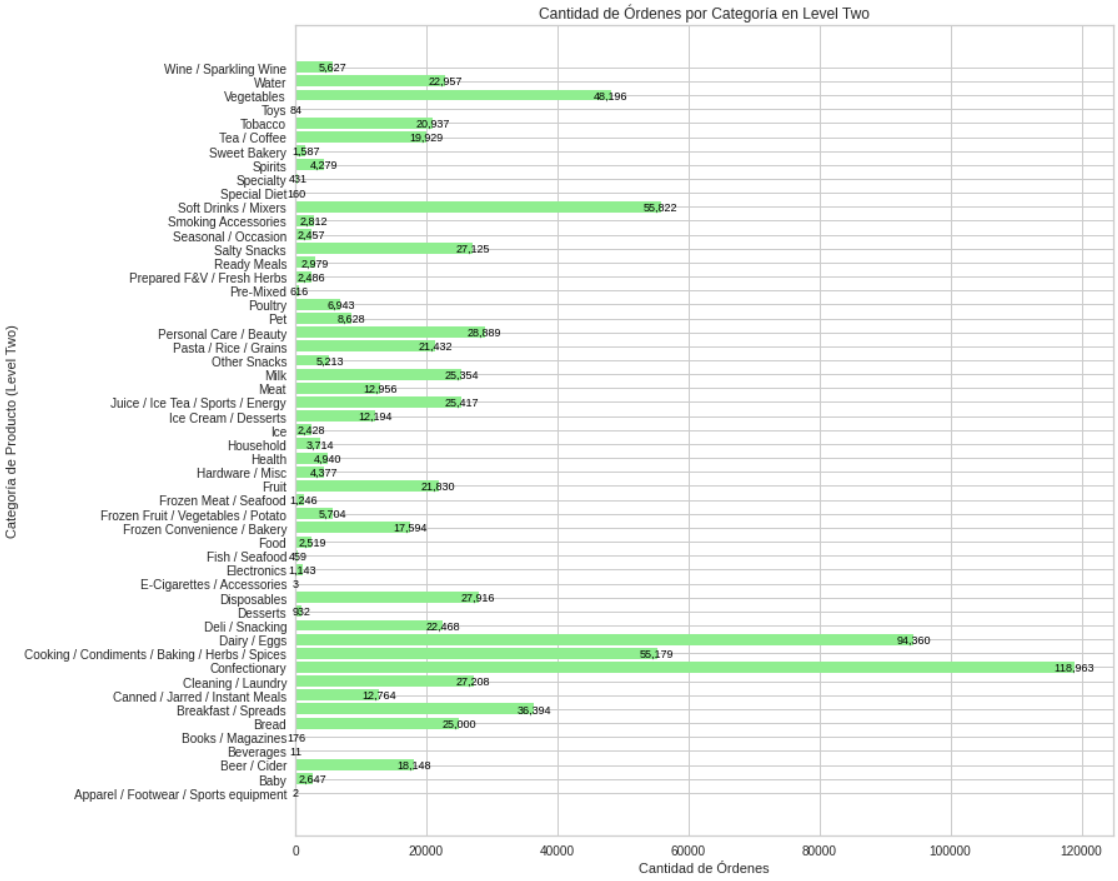
El gráfico muestra la cantidad de órdenes distribuidas por categoría de producto en level_two, proporcionando una visión detallada de las preferencias de los clientes en diferentes subcategorías.

Los resultados indican que las subcategorías Confectionary, Soft Drinks / Mixers, y Water son las más populares entre los clientes. Esto proporciona una comprensión clara de qué productos específicos son más demandados.

Estos resultados son favorables para la clusterización ya que permiten segmentar a los clientes según sus preferencias de productos a un nivel más granular. Identificar subcategorías más demandadas facilita la creación de clusters basados en hábitos de consumo específicos

[39]:
```
# Crear una gráfica de barras
plt.figure(figsize=(14, 10))
```

```
# Crear el gráfico de barras horizontal
bars = plt.barh(ordenes_por_categoria_df_l2['level_two'],␣
 ↪ordenes_por_categoria_df_l2['cantidad'], color='lightgreen')

# Configurar las etiquetas del eje x y del eje y
plt.xlabel('Cantidad de Órdenes')
plt.ylabel('Categoría de Producto (Level Two)')

# Configurar el título del gráfico
plt.title('Cantidad de Órdenes por Categoría en Level Two')

# Mostrar los valores en las barras
for bar in bars:
    width = bar.get_width()
    plt.text(width, bar.get_y() + bar.get_height() / 2, f'{int(width):,}',␣
 ↪ha='center', va='center', fontsize=9, color='black')

# Ajustar los márgenes
plt.subplots_adjust(left=0.3, right=0.95, top=0.95, bottom=0.05)

# Mostrar la gráfica
plt.show()
```



3.4.3 Análisis de la Matriz de Correlación de Pearson

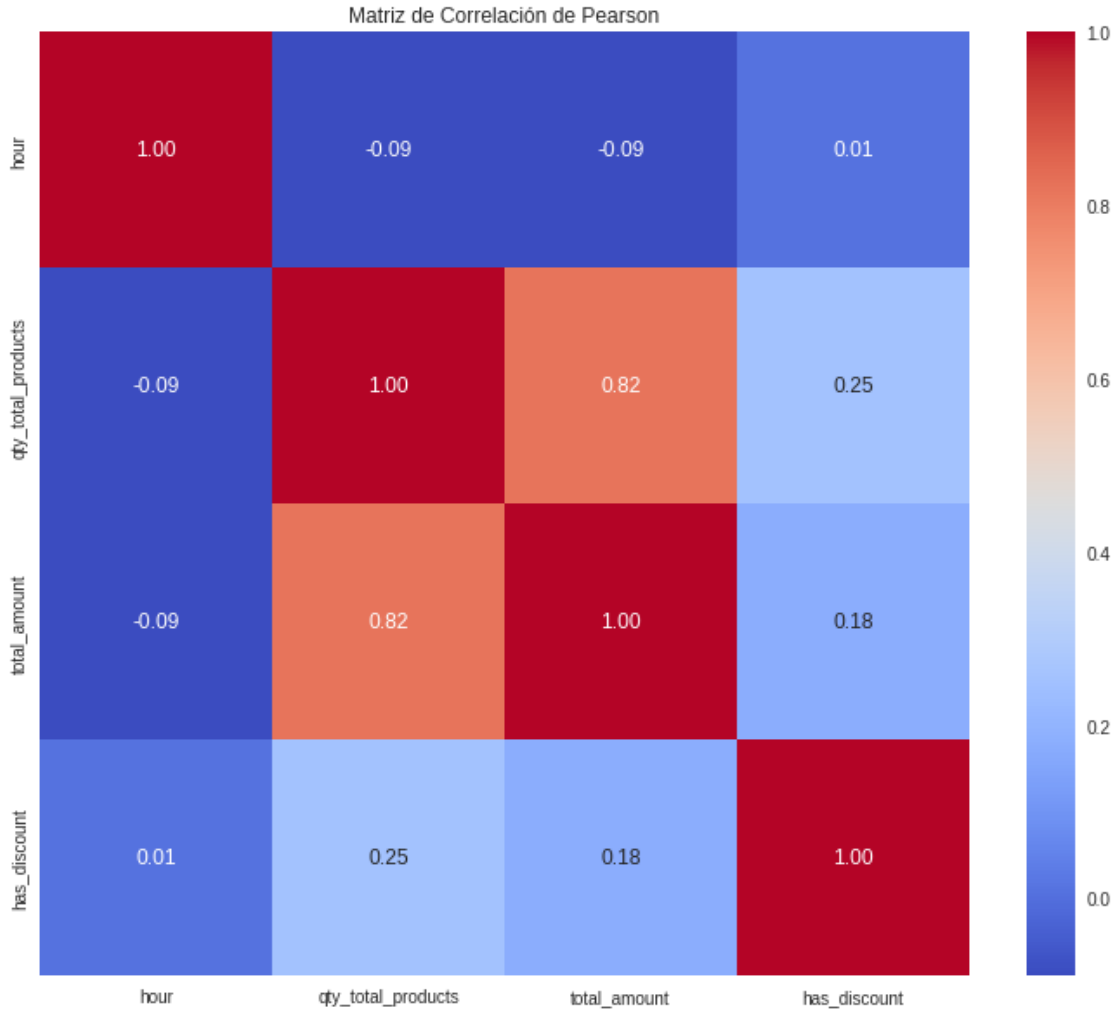
La matriz de correlación de Pearson proporciona información sobre la relación lineal entre las variables numéricas del DataFrame. Los valores de la correlación oscilan entre -1 y 1, donde 1 indica una correlación positiva perfecta, -1 indica una correlación negativa perfecta, y 0 indica que no hay correlación lineal.

La matriz de correlación de Pearson revela que la cantidad total de productos (qty_total_products) tiene una fuerte correlación positiva con el monto total (total_amount) de las órdenes (0.82), indicando que a mayor cantidad de productos, mayor es el monto total. Además, se observa una correlación moderada positiva de has_discount con qty_total_products (0.25) y total_amount (0.18), sugiriendo que las órdenes con descuentos tienden a tener más productos y un monto total mayor. En contraste, la variable hour no muestra una correlación significativa con ninguna otra variable, indicando que la hora del día no afecta notablemente la cantidad de productos ni el monto total de las órdenes.

Este análisis de la matriz de correlación ayuda a identificar las relaciones entre las variables, proporcionando una base para futuras decisiones en análisis predictivo y estrategias de negocio.

[40]:
```
# Calcular la matriz de correlación de Pearson
correlation_matrix = df.corr(method='pearson')

# Mostrar la matriz de correlación
print(correlation_matrix)

# Crear un mapa de calor para visualizar la matriz de correlación
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación de Pearson')
plt.show()
```

```
                    hour  qty_total_products  total_amount  has_discount
hour                1.00               -0.09         -0.09          0.01
qty_total_products -0.09                1.00          0.82          0.25
total_amount       -0.09                0.82          1.00          0.18
has_discount        0.01                0.25          0.18          1.00
```



3.4.4 Análisis del Pairplot

El pairplot proporciona una visualización detallada de las relaciones entre las variables numéricas hour, qty_total_products, y total_amount. Aquí están algunas observaciones clave del gráfico:

Distribución de qty_total_products:

La mayoría de las órdenes contienen menos de 40 productos, con una concentración significativa alrededor de los 10 a 20 productos. Existen algunos valores atípicos con cantidades mayores, pero estos son menos frecuentes.

Distribución de total_amount:

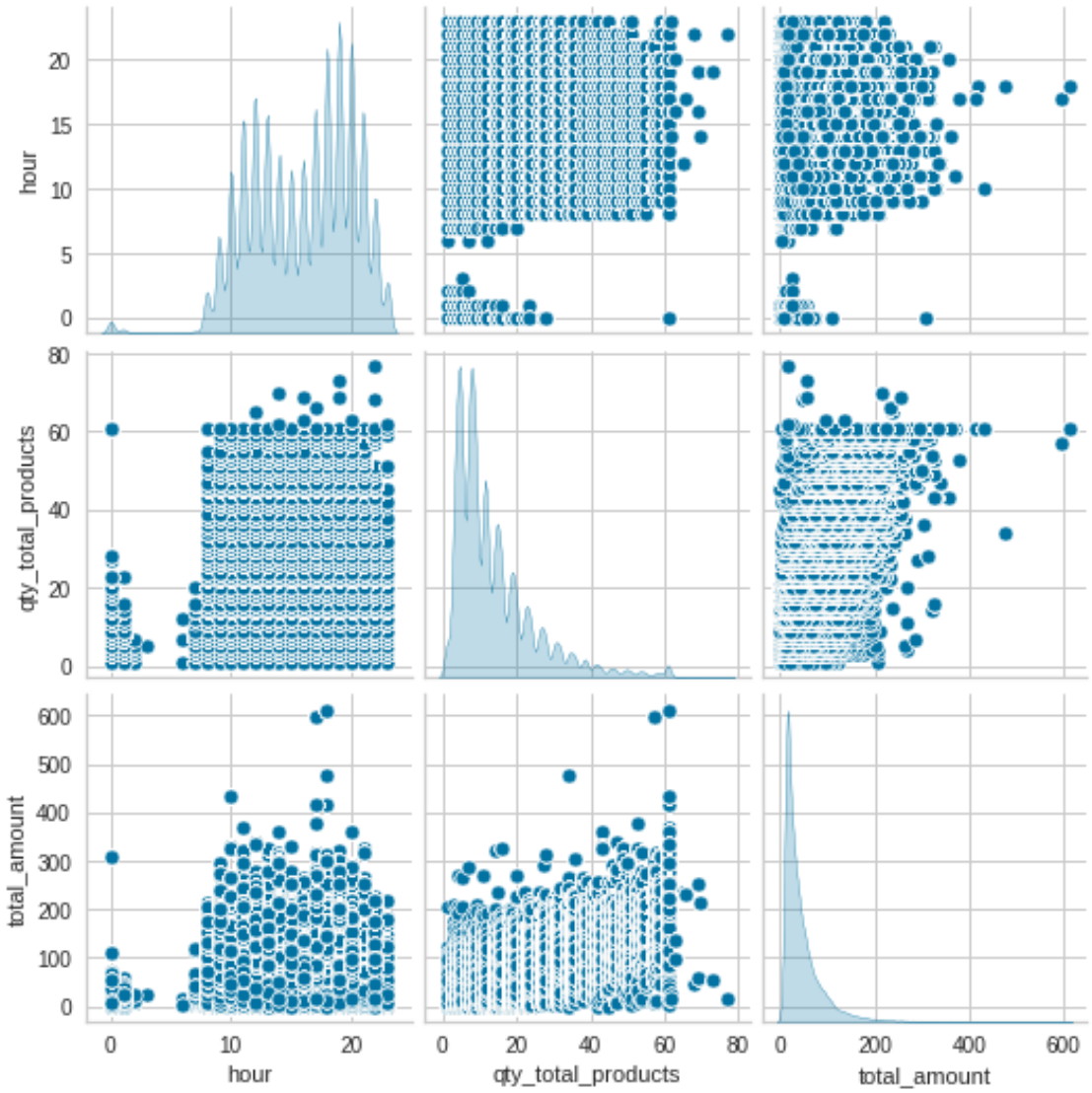
La mayoría de las órdenes tienen un monto total inferior a 100, con una concentración alrededor de los 20 a 50. Hay algunos valores atípicos con montos mucho más altos, hasta 600, pero estos son escasos.

Relación entre qty_total_products y total_amount:

Existe una correlación positiva notable entre la cantidad de productos y el monto total de la orden, confirmando la relación lineal observada previamente.

Este análisis visual ayuda a identificar patrones y relaciones entre las variables, confirmando las observaciones obtenidas de la matriz de correlación de Pearson. Los datos parecen consistentes con los patrones esperados y no muestran relaciones inesperadas entre las variables.

[41]:
```
# Seleccionar las columnas numéricas para el pairplot
numerical_columns = ['hour', 'qty_total_products', 'total_amount']

# Crear el pairplot
sns.pairplot(df[numerical_columns], diag_kind='kde')

# Mostrar el plot
plt.show()
```

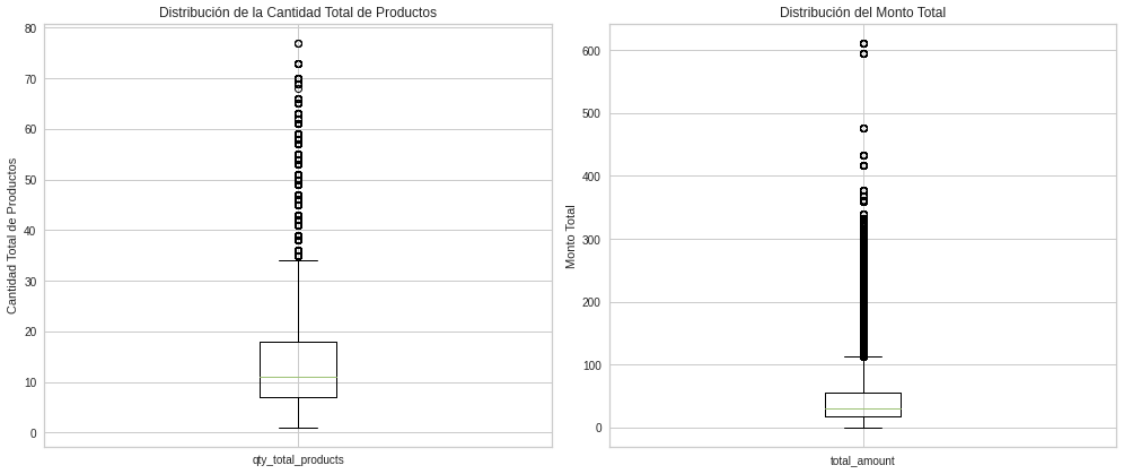


3.5 Tratamiento de Outliers

La eliminación de outliers en las categorías qty_total_products y total_amount se realizó para mejorar la calidad y la fiabilidad del análisis de datos. Los outliers, definidos como valores que superan el percentil 99, pueden distorsionar las estadísticas descriptivas y los modelos predictivos, llevando a conclusiones erróneas. Al eliminar estos valores extremos, se obtiene un conjunto de datos más representativo del comportamiento típico de los usuarios, permitiendo análisis más precisos y significativos. Esta limpieza de datos es crucial para garantizar la integridad de los resultados, facilitando la identificación de patrones y tendencias reales sin la influencia de valores atípicos.

En este caso, se eliminaron 12,516 registros de un total de 873,635, lo cual representa una pequeña fracción (aproximadamente el 1.43%). Aunque se pierden algunos datos, la reducción es mínima y no debería afectar significativamente los resultados generales.

[42]:
```
# Crear una figura con dos subplots
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Crear box plot para 'qty_total_products'
axs[0].boxplot(df['qty_total_products'])
axs[0].set_title('Distribución de la Cantidad Total de Productos')
axs[0].set_ylabel('Cantidad Total de Productos')
axs[0].set_xticks([1])
axs[0].set_xticklabels(['qty_total_products'])

# Crear box plot para 'total_amount'
axs[1].boxplot(df['total_amount'])
axs[1].set_title('Distribución del Monto Total')
axs[1].set_ylabel('Monto Total')
axs[1].set_xticks([1])
axs[1].set_xticklabels(['total_amount'])

# Ajustar el layout
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```



[43]:
```
# Para forzar que no se muestre en notación cientifica
pd.options.display.float_format = '{:.2f}'.format

df.describe(percentiles=[.25, .5, .75, 0.99])
```

[43]:

	hour	qty_total_products	total_amount
count	873635.00	873635.00	873635.00
mean	15.91	13.70	43.14

```
std        4.10              10.93         39.17
min        0.00               1.00          0.00
25%       12.00               7.00         17.05
50%       17.00              11.00         30.20
75%       19.00              18.00         55.12
99%       23.00              55.00        192.96
max       23.00              77.00        611.78
```

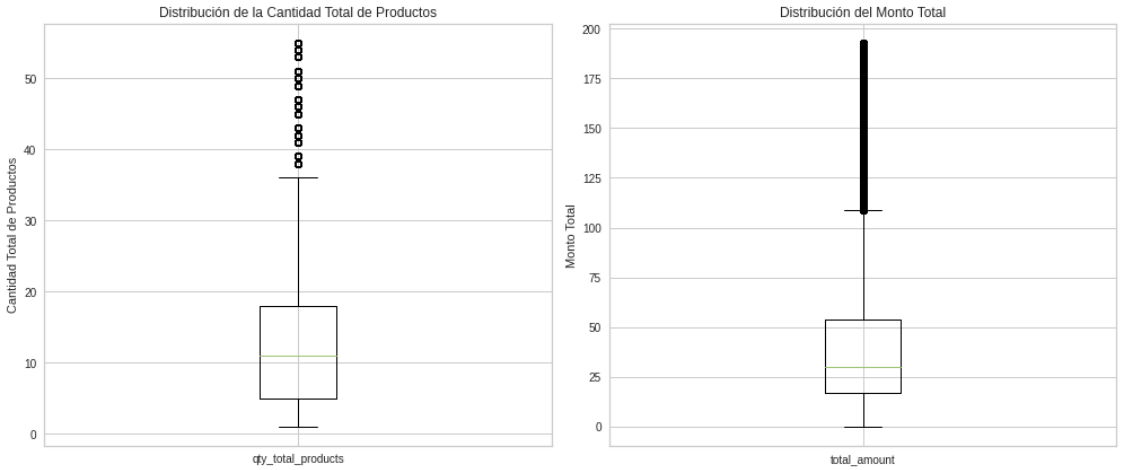
[44]:
```
# Listado de columnas a considerar para los percentiles
columns_to_consider = [ 'qty_total_products', 'total_amount']

# Cálculo del percentil 99 para cada una de estas columnas
percentiles_99 = df[columns_to_consider].quantile(0.99)

# Crear una columna que indique si alguna de estas columnas supera el percentil␣
 ↪99
df['above_99th_percentile'] = df.apply(lambda row: any(row[col] >␣
 ↪percentiles_99[col] for col in columns_to_consider), axis=1)

# Contar cuántas órdenes superan el percentil 99 en alguna de estas columnas
orders_above_99th = df['above_99th_percentile'].sum()

print(f"Total de órdenes que superan el percentil 99 en alguna categoría:␣
 ↪{orders_above_99th}")
```

```
Total de órdenes que superan el percentil 99 en alguna categoría: 12516
```

[45]:
```
# Cálculo del percentil 99.9 para qty_total_products y total_amount
percentile_99_qty_total_products = df['qty_total_products'].quantile(0.99)
percentile_99_total_amount = df['total_amount'].quantile(0.99)

# Filtrar las filas que no superen estos percentiles
df = df[(df['qty_total_products'] <= percentile_99_qty_total_products) &
                (df['total_amount'] <= percentile_99_total_amount)]
```

[46]:
```
# Crear una figura con dos subplots
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Crear box plot para 'qty_total_products'
axs[0].boxplot(df['qty_total_products'])
axs[0].set_title('Distribución de la Cantidad Total de Productos')
axs[0].set_ylabel('Cantidad Total de Productos')
axs[0].set_xticks([1])
axs[0].set_xticklabels(['qty_total_products'])

# Crear box plot para 'total_amount'
axs[1].boxplot(df['total_amount'])
```

```
axs[1].set_title('Distribución del Monto Total')
axs[1].set_ylabel('Monto Total')
axs[1].set_xticks([1])
axs[1].set_xticklabels(['total_amount'])

# Ajustar el layout
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```



[47]:
```
# Para forzar que no se muestre en notación cientifica
pd.options.display.float_format = '{:.2f}'.format

df.describe(percentiles=[.25, .5, .75, 0.99])
```

[47]:

	hour	qty_total_products	total_amount
count	861119.00	861119.00	861119.00
mean	15.92	13.12	40.74
std	4.10	9.81	33.07
min	0.00	1.00	0.00
25%	12.00	5.00	16.92
50%	17.00	11.00	29.70
75%	19.00	18.00	53.58
99%	23.00	47.00	158.03
max	23.00	55.00	192.96

4 Creacion y transormacion de variables

4.0.1 Creación de Variable "frecuencia de usuario"

En este proceso, se agrupa el DataFrame df por user_id para contar las órdenes únicas (order_id) realizadas por cada usuario, creando así una nueva columna order_count_usu que refleja la frecuencia de uso de cada usuario. Posteriormente, esta información se une de nuevo al DataFrame original, asegurando que cada registro de usuario contenga la cantidad de órdenes realizadas.

Data frame "df" creado a partir del merge del dataset1 y dataset2. Contabilización de número de órdenes realizada por cada usuario en df

[48]:
```
# Agrupar por "user_id" y contar los "order_id" únicos
order_counts = df.groupby('user_id')['order_id'].nunique().reset_index()
order_counts.columns = ['user_id', 'order_count_usu']

# Unir esta información de vuelta al DataFrame original sin generar duplicados
df = df.drop(columns=['order_count_usu'], errors='ignore')  # Elimina la␣
↪columna existente si existe
df = df.merge(order_counts, on='user_id', how='left')

# Ahora df_inicial tiene una nueva columna "order_count" que muestra cuántas␣
↪órdenes ha hecho cada usuario
df.head()
```

[48]:

	order_id	weekday	hour	business_type_name	partner_id	user_id	qty_total_products
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	Kiosks	vXtLI7qWnGDd4ZTQOiZJZ/wQXXY1GLgkEE+pErgYcCc=	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	1
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	Kiosks	jm4h8a5ugX13K8EyOwmt3XmFG+tC+GcN9Od0EX8TeLQ=	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	1
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	Kiosks	1JhXLeaP56TZ3S0s3L4ze2uF1z6TzLNNG4l+ZjrGGpc=	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1
3	GdvT/R8JqkIOzSzfwVrVag6okmBzO0ms88KxgAseht8=	Saturday	23	Kiosks	bmaUtn62nJYTQWQDGzL/E8Ka06Mx4EhRVVfk5OQwLZQ=	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	Kiosks	MLb861rH4PvXNX5DI2ODCQfEX82/0OnwHbRTaCBXD0U=	Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=	1

```
   total_amount  has_discount                                    product_id  \
0          8.05         False  CNOSao/6N6ofzazjxaKKpNX/uJJmYmRyEwsj0DC8FYM=
1          7.91         False  KnXqi39yRIczh1FaJo1LSzCJXK7K0ypgFu83U2qq8JY=
2          7.85         False  wm+wxfQbe8WeSI0D/2fTWWQ4PYtqLel1kNOx7Ltk/zY=
3          8.60         False  6ZvoEhv/vRYiuRQhx477vpGbwLxJYbWAeXyK/yy+kwM=
4          8.38         False  N6x+F9p/qU2YCJsTOvSq4JGVrtXKVGstCBjqBIl3Pb4=

           level_one                                    level_two  \
0  Smoking / Tobacco                                      Tobacco
1          Beverages                          Soft Drinks / Mixers
2          Beverages                          Soft Drinks / Mixers
3  Smoking / Tobacco                                      Tobacco
4     Packaged Foods  Cooking / Condiments / Baking / Herbs / Spices

   above_99th_percentile  order_count_usu
0                  False                1
1                  False                2
2                  False                7
3                  False                7
4                  False                1
```

Filtro el dataframe por usuario ("user_id") con 35mil filas, creando un nuevo DF llamado "df_usuarios" Realizamos este filtro para estudiar la frecuencia de compra de los usuarios

[49]:
```
# Filtrar el DataFrame por "user_id" únicos
df_usuarios = df.drop_duplicates(subset='user_id', keep='first')

# Mostrar el resultado para verificar
df_usuarios.head()
```

[49]:
```
                                order_id    weekday  hour  \
0  MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=   Thursday    14
1  GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=    Tuesday     0
2  KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=  Wednesday    22
4  jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=    Tuesday    14
5  hL+jTxAdNzCCuKxCpDNOlFAlPCHSE7NsX/GfIGS3+hA=     Friday    13

  business_type_name                                    partner_id  \
0              Kiosks  vXtLI7qWnGDd4ZTQOiZJZ/wQXXY1GLgkEE+pErgYcCc=
1              Kiosks  jm4h8a5ugX13K8EyOwmt3XmFG+tC+GcN9Od0EX8TeLQ=
2              Kiosks  1JhXLeaP56TZ3S0s3L4ze2uF1z6TzLNNG4l+ZjrGGpc=
4              Kiosks  MLb861rH4PvXNX5DI2ODCQfEX82/0OnwHbRTaCBXD0U=
5              Kiosks  GthUNOi4W/yXw2XRX/3XsGrwvW5Lj2pz7oAEnRPF7bs=

                                     user_id  qty_total_products  \
```

```
0  SwE0/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=                    1
1  Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZ0n+84SPBeEG4=                    1
2  Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=                    1
4  Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=                    1
5  SxNsmcGgwM+3KXeAbgmqCT+HCuuMBIW2RzgBx2gd3fI=                    1

   total_amount  has_discount                                  product_id  \
0          8.05         False  CNOSao/6N6ofzazjxaKKpNX/uJJmYmRyEwsj0DC8FYM=
1          7.91         False  KnXqi39yRIczh1FaJo1LSzCJXK7K0ypgFu83U2qq8JY=
2          7.85         False  wm+wxfQbe8WeSI0D/2fTWWQ4PYtqLel1kNOx7Ltk/zY=
4          8.38         False  N6x+F9p/qU2YCJsTOvSq4JGVrtXKVGstCBjqBIl3Pb4=
5         11.09         False  7gsiuBtVkcR6M0IUx0vqrZJE4Bf/3QzQB5ljOyV47+k=

           level_one                                      level_two  \
0  Smoking / Tobacco                                        Tobacco
1          Beverages                          Soft Drinks / Mixers
2          Beverages                          Soft Drinks / Mixers
4     Packaged Foods  Cooking / Condiments / Baking / Herbs / Spices
5             Frozen                          Ice Cream / Desserts

   above_99th_percentile  order_count_usu
0                  False                1
1                  False                2
2                  False                7
4                  False                1
5                  False               10
```

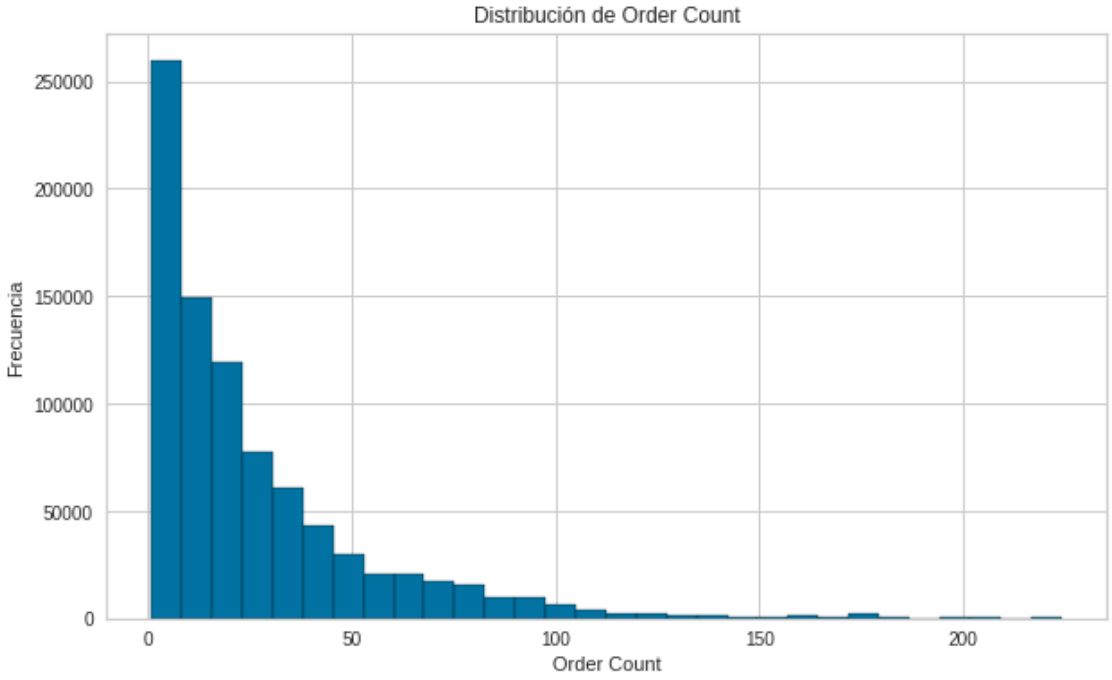
[50]:
```
# Obtener el número de filas en el DataFrame df_inicial
num_filas = df_usuarios.shape[0]

# Imprimir el resultado
print("Número de filas en el DataFrame df_inicial:", num_filas)
```

```
Número de filas en el DataFrame df_inicial: 35050
```

[51]:
```
# Estadísticas descriptivas
print(df_usuarios['order_count_usu'].describe())

# Histograma
plt.figure(figsize=(10, 6))
df['order_count_usu'].hist(bins=30, edgecolor='black')
plt.title('Distribución de Order Count')
plt.xlabel('Order Count')
plt.ylabel('Frecuencia')
plt.show()
```

```
count   35050.00
mean        5.61
```

```
std        10.70
min         1.00
25%         1.00
50%         2.00
75%         5.00
max       224.00
Name: order_count_usu, dtype: float64
```



Distribución de "order_count" en el dataframe "df_usuarios"

4.0.2 Estudiar la distribución de la frecuencia de órdenes agrupando por usuario (user_id) me sirve para determinar los rangos de si un usuario ordena con frecuencia "baja" , "media" y "alta".

4.0.3 Luego lo extrapolo al dataframe agrupado por "order_id" realizado un poco más adelate

Si el promedio de órdenes por usuario es 5.62 (redondeo en 6) entonces:

- **frecuencia "baja" menor o igual 3 ordenes**
- **frecuencia "media" = 4 a 7 ordenes**
- **frecuencia "alta" mayor a 8 ordenes**

Creacion variable "frec_compra_usu" en categorías "baja", "media" y "alta".

[52]:
```
def categorizar_frecuencia_usuario(X):
    if  X < 4:
        return 'Baja'
    elif X < 8:
        return 'Media'
    else:
        return 'Alta'
```

[53]:
```
df['frec_compra_usu'] = df['order_count_usu'].
↪apply(categorizar_frecuencia_usuario)
```

4.0.4 Creación de Variable "FIN DE SEMANA"

En este proceso, se añade una nueva columna al DataFrame df llamada es_fin_de_semana que indica si una orden fue realizada durante el fin de semana. Esta columna es booleana y toma el valor True si el weekday es Saturday o Sunday, y False en caso contrario.

[54]:
```
df['es_fin_de_semana'] = (df['weekday'] == 'Saturday') | (df['weekday'] ==␣
↪'Sunday')
df
```

[54]:

	order_id	weekday	hour	\
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	
3	GdvT/R8JqkIOzSzfwVrVag6okmBzOOms88KxgAseht8=	Saturday	23	
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	
…	…	…	…	
861114	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	
861115	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	
861116	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	
861117	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	
861118	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	

	business_type_name	partner_id	\
0	Kiosks	vXtLI7qWnGDd4ZTQOiZJZ/wQXXY1GLgkEE+pErgYcCc=	
1	Kiosks	jm4h8a5ugX13K8EyOwmt3XmFG+tC+GcN9Od0EX8TeLQ=	
2	Kiosks	1JhXLeaP56TZ3S0s3L4ze2uF1z6TzLNNG4l+ZjrGGpc=	
3	Kiosks	bmaUtn62nJYTQWQDGzL/E8Ka06Mx4EhRVVfk5OQwLZQ=	
4	Kiosks	MLb861rH4PvXNX5DI2ODCQfEX82/0OnwHbRTaCBXD0U=	
…	…	…	
861114	Market	w8O/LqbK5VCkpQtOieRFkAUljGjuQLLajrQpLWgWBtI=	
861115	Market	w8O/LqbK5VCkpQtOieRFkAUljGjuQLLajrQpLWgWBtI=	
861116	Market	w8O/LqbK5VCkpQtOieRFkAUljGjuQLLajrQpLWgWBtI=	
861117	Market	w8O/LqbK5VCkpQtOieRFkAUljGjuQLLajrQpLWgWBtI=	
861118	Market	w8O/LqbK5VCkpQtOieRFkAUljGjuQLLajrQpLWgWBtI=	

```
                                                user_id  qty_total_products  \
0       SwE0/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=                     1
1       Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZ0n+84SPBeEG4=                     1
2       Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=                     1
3       Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=                     1
4       Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=                     1
…                                                 …                   …
861114  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                    55
861115  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                    55
861116  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                    55
861117  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                    55
861118  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                    55

        total_amount  has_discount  \
0               8.05         False
1               7.91         False
2               7.85         False
3               8.60         False
4               8.38         False
…                …             …
861114        188.96          True
861115        188.96          True
861116        188.96          True
861117        188.96          True
861118        188.96          True

                                             product_id               level_one  \
0       CNOSao/6N6ofzazjxaKKpNX/uJJmYmRyEwsj0DC8FYM=       Smoking / Tobacco
1       KnXqi39yRIczh1FaJo1LSzCJXK7K0ypgFu83U2qq8JY=               Beverages
2       wm+wxfQbe8WeSIOD/2fTWWQ4PYtqLel1kNOx7Ltk/zY=               Beverages
3       6ZvoEhv/vRYiuRQhx477vpGbwLxJYbWAeXyK/yy+kwM=       Smoking / Tobacco
4       N6x+F9p/qU2YCJsTOvSq4JGVrtXKVGstCBjqBIl3Pb4=          Packaged Foods
…                                                 …                      …
861114  JUUVnWbr1jinwynRlQ+rlUcMVXnC1kgGYJDv7wGfo0g=  Dairy / Chilled / Eggs
861115  ClmfhpzVp8klHEsrvk2841wzeyrWn40LeYBal3vsDhQ=          Packaged Foods
861116  uHPCywTapMdvyfKa4yipz4fkDOIDn/5DUtLf2udaq5g=          Bread / Bakery
861117  c9swUHHVuB7jNLPPJ3hvlGeo011ATs4riQ2JTsLQR+s=          Packaged Foods
861118  c0/NrZqez55USRQyGHlY5xe4yaolhJJwLLOqN7ZDEeY=          Packaged Foods

                                          level_two  above_99th_percentile  \
0                                           Tobacco                  False
1                              Soft Drinks / Mixers                  False
2                              Soft Drinks / Mixers                  False
3                                           Tobacco                  False
4       Cooking / Condiments / Baking / Herbs / Spices                  False
…                                                 …                      …
```

```
861114                                  Dairy / Eggs                  False
861115                Canned / Jarred / Instant Meals                 False
861116                                         Bread                  False
861117                             Breakfast / Spreads                 False
861118                             Breakfast / Spreads                 False

        order_count_usu frec_compra_usu  es_fin_de_semana
0                     1            Baja             False
1                     2            Baja             False
2                     7           Media             False
3                     7           Media              True
4                     1            Baja             False
…                   …              …                …
861114               30            Alta             False
861115               30            Alta             False
861116               30            Alta             False
861117               30            Alta             False
861118               30            Alta             False

[861119 rows x 16 columns]
```

[55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 861119 entries, 0 to 861118
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   order_id               861119 non-null  object
 1   weekday                861119 non-null  category
 2   hour                   861119 non-null  int64
 3   business_type_name     861119 non-null  category
 4   partner_id             861119 non-null  object
 5   user_id                861119 non-null  object
 6   qty_total_products     861119 non-null  int64
 7   total_amount           861119 non-null  float64
 8   has_discount           861119 non-null  bool
 9   product_id             861119 non-null  object
 10  level_one              861119 non-null  object
 11  level_two              861119 non-null  category
 12  above_99th_percentile  861119 non-null  bool
 13  order_count_usu        861119 non-null  int64
 14  frec_compra_usu        861119 non-null  object
 15  es_fin_de_semana       861119 non-null  bool
dtypes: bool(3), category(3), float64(1), int64(3), object(6)
memory usage: 77.2+ MB
```

4.0.5 Creacion de la variable categorica 'periodo_dia' (mañana, tarde y noche)

[56]:
```
columns_to_convert = ["has_discount", "es_fin_de_semana", "frec_compra_usu"]
for column in columns_to_convert:
    if column in df.columns:
        df[column] = df[column].astype('category')
    else:
        print(f"La columna {column} no se encuentra en el dataframe")

# Verifica que las columnas hayan sido transformadas
print(df.dtypes)
```

```
order_id                  object
weekday                 category
hour                       int64
business_type_name      category
partner_id                object
user_id                   object
qty_total_products         int64
total_amount             float64
has_discount            category
product_id                object
level_one                 object
level_two               category
above_99th_percentile       bool
order_count_usu            int64
frec_compra_usu         category
es_fin_de_semana        category
dtype: object
```

[57]:
```
df['has_discount'].cat.categories
```

[57]:
```
Index([False, True], dtype='object')
```

[58]:
```
# Función para categorizar las horas en mañana, tarde y noche
def categorizar_hora(hora):
    if 5 <= hora < 12:
        return 'Matutino'
    elif 12 <= hora < 18:
        return 'Vespertino'
    else:
        return 'Nocturno'
```

[59]:
```
# Aplicar la función a la columna 'hour' para crear una nueva columna␣
↪'periodo_dia'
df['periodo_dia'] = df['hour'].apply(categorizar_hora)
```

[60]:
```
# Convertir la nueva columna a tipo categórico
df['periodo_dia'] = df['periodo_dia'].astype('category')
```

[61]:
```
# Verificar las nuevas categorías
df['periodo_dia'].cat.categories
```

[61]:
```
Index(['Matutino', 'Nocturno', 'Vespertino'], dtype='object')
```

[62]:
```
# Contar la cantidad de categorías distintas en la columna 'level_one'

print("Cantidad de categorías distintas en 'level_one':", df['level_one'].
 ↪nunique())
```

```
Cantidad de categorías distintas en 'level_one': 14
```

4.0.6 Eliminacion de Variables del DataFrame

La decisión de eliminar las variables partner_id, level_two, product_id, y business_type_name del DataFrame se basa en varios criterios estratégicos. En primer lugar, estas variables pueden no aportar valor adicional al análisis si ya tenemos variables más agregadas o representativas, como level_one en lugar de level_two. Mantener el DataFrame con las variables más relevantes facilita el análisis y la interpretación de los resultados, evitando la complejidad innecesaria y mejorando la claridad. Además, reducir el número de columnas mejora la eficiencia computacional, especialmente en conjuntos de datos grandes, acelerando el procesamiento y el análisis. En resumen, la eliminación de estas variables permite un análisis más centrado, eficiente y claro, dirigido a entender mejor el comportamiento del usuario y sus patrones de compra.

[63]:
```
df = df.
 ↪drop(['partner_id','level_two','product_id','business_type_name','above_99th_percentile'],␣
 ↪axis=1)

#'user_id'
```

[64]:
```
print("Shape of data orders:",df.shape)
df.info()
```

```
Shape of data orders: (861119, 12)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 861119 entries, 0 to 861118
Data columns (total 12 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   order_id            861119 non-null  object
 1   weekday             861119 non-null  category
 2   hour                861119 non-null  int64
 3   user_id             861119 non-null  object
 4   qty_total_products  861119 non-null  int64
```

```
 5   total_amount        861119 non-null  float64
 6   has_discount        861119 non-null  category
 7   level_one           861119 non-null  object
 8   order_count_usu     861119 non-null  int64
 9   frec_compra_usu     861119 non-null  category
 10  es_fin_de_semana    861119 non-null  category
 11  periodo_dia         861119 non-null  category
dtypes: category(5), float64(1), int64(3), object(3)
memory usage: 56.7+ MB
```

[65]:
```
df.describe()
```

[65]:

	hour	qty_total_products	total_amount	order_count_usu
count	861119.00	861119.00	861119.00	861119.00
mean	15.92	13.12	40.74	26.57
std	4.10	9.81	33.07	29.28
min	0.00	1.00	0.00	1.00
25%	12.00	5.00	16.92	7.00
50%	17.00	11.00	29.70	17.00
75%	19.00	18.00	53.58	36.00
max	23.00	55.00	192.96	224.00

[66]:
```
pd.isnull(df).describe()
```

[66]:

	order_id	weekday	hour	user_id	qty_total_products	total_amount	\
count	861119	861119	861119	861119	861119	861119	
unique	1	1	1	1	1	1	
top	False	False	False	False	False	False	
freq	861119	861119	861119	861119	861119	861119	

	has_discount	level_one	order_count_usu	frec_compra_usu	\
count	861119	861119	861119	861119	
unique	1	1	1	1	
top	False	False	False	False	
freq	861119	861119	861119	861119	

	es_fin_de_semana	periodo_dia
count	861119	861119
unique	1	1
top	False	False
freq	861119	861119

[67]:
```
df.tail()
```

[67]:

	order_id	weekday	hour	\
861114	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	
861115	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	

```
861116  6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=  Tuesday    19
861117  6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=  Tuesday    19
861118  6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=  Tuesday    19

                                            user_id  qty_total_products  \
861114  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55
861115  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55
861116  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55
861117  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55
861118  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55

        total_amount has_discount              level_one  order_count_usu  \
861114        188.96         True  Dairy / Chilled / Eggs               30
861115        188.96         True          Packaged Foods               30
861116        188.96         True          Bread / Bakery               30
861117        188.96         True          Packaged Foods               30
861118        188.96         True          Packaged Foods               30

       frec_compra_usu es_fin_de_semana periodo_dia
861114           Alta            False    Nocturno
861115           Alta            False    Nocturno
861116           Alta            False    Nocturno
861117           Alta            False    Nocturno
861118           Alta            False    Nocturno
```

4.0.7 Verificamos cantidad de ordenes por usuario filtrando por user_id

[68]:
```
# Filtrar el DataFrame para el 'order_id' específico
df_filtrado = df[df['user_id'] ==␣
↪'CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=']

# Agrupar por 'order_id' y 'user_id' sin operaciones de suma
df_agrupado = df_filtrado.groupby(['order_id', 'user_id'], as_index=False).
↪first()

# Mostrar el DataFrame filtrado
df_agrupado
```

[68]:
```
                                     order_id  \
0  6zsLlTL7kIryTsJCKC+F/bouhc88A7bnoCr3NZB7NAc=
1  APsZMJyg3BY19WEkvT4btMRHn0AYPC6OZ6/PsAN/62Y=
2  DNK0KzL3hRdF11EMqVucgM3J61y0yTjewHfJK/gkigs=
3  DZVmt+wbAQ0wZGkSdXv7pOxutVXug3iIU7Q63s6CUcY=
4  FaVZDRb9hxlB8/TcwkrGwPhEmmTuoUued5U6MrwhiVo=
5  GsTXfujblfGzj73Bn3WiHV6vecXHYEPmrTAO8aCoKUk=
```

```
6   KTBAWp5wyZyswGjpZSU6NTpNW6vOynZ6Rws1eoKzFTY=
7   UktjJ9dGT4c4Kx1feppDbZQZ+FJkqQUBQJKO6JWNOBk=
8   WcPBNq1Ylt1QDHI307o0Er/bQzuIVjBpA3uaAwp4L20=
9   d+aW5Lc/kMPNorm8CS2IaoLKAUKBn5I8l6awLywZ/0Y=
10  qcg6QmbAxf2HOMwpGkyxpab5kdjzztP8YN8fTvvheO0=
11  s+U5edsPvFvyCuDPHR5dtkNfZTTUd8CbS4PjBoshrzM=
12  sUfS7O2IaWBIe3tXt+4gab+z6nzHoJ0XU+eSs8MQLpY=
13  uIVcLOygeqRAEjPi7C7aB+zRe6ZHFqQ8iMI32Jqd9kU=
14  xlYiktoPbE/YFdfEeCNpbmbPGCbOEUC5TnXGtkgMoFo=

                                        user_id    weekday  hour  \
0   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=     Monday    20
1   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Saturday    10
2   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Thursday    15
3   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=     Monday    18
4   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=     Friday    17
5   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Saturday    16
6   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=     Monday    19
7   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Saturday    12
8   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=     Monday    20
9   CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=  Wednesday    11
10  CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=  Wednesday    14
11  CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Saturday    20
12  CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=   Saturday    16
13  CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=    Tuesday    16
14  CmjoaPL6j93XWLMqWiOmhIloFEqkCbgR9V5h+sDrAj8=  Wednesday    10

    qty_total_products  total_amount has_discount          level_one  \
0                   11         41.24         True  Smoking / Tobacco
1                   23        101.43         True         Home / Pet
2                   19         67.51         True             Snacks
3                    8         55.74         True         Home / Pet
4                   12         78.55         True             Frozen
5                   15         58.35         True            Produce
6                   18         83.34         True         Home / Pet
7                    7         30.45         True  Smoking / Tobacco
8                    5         24.18         True          Beverages
9                   11         60.47         True         Home / Pet
10                  14         67.06         True  Smoking / Tobacco
11                  26         87.55         True         Home / Pet
12                  11         42.41         True  Smoking / Tobacco
13                  11        113.51         True         Home / Pet
14                   5         27.01        False  Smoking / Tobacco

    order_count_usu frec_compra_usu es_fin_de_semana periodo_dia
0                15            Alta            False    Nocturno
1                15            Alta             True    Matutino
```

```
2                15           Alta            False  Vespertino
3                15           Alta            False    Nocturno
4                15           Alta            False  Vespertino
5                15           Alta             True  Vespertino
6                15           Alta            False    Nocturno
7                15           Alta             True  Vespertino
8                15           Alta            False    Nocturno
9                15           Alta            False    Matutino
10               15           Alta            False  Vespertino
11               15           Alta             True    Nocturno
12               15           Alta             True  Vespertino
13               15           Alta            False  Vespertino
14               15           Alta            False    Matutino
```

4.1 Se crean las columnas con porcentajes y cantidades para cada categoria de level_one, utilizamos un DF_PRUEBA para evitar modificar el df original

[69]:
```
DF_PRUEBA = df.copy()
```

[70]:
```
# Agrupar el DataFrame por 'order_id' y 'level_one'
grouped_df = DF_PRUEBA.groupby(['order_id', 'level_one'])

# Calcular el tamaño de cada grupo (número de filas en cada grupo)
group_sizes = grouped_df.size().reset_index(name='group_size')

# Calcular el tamaño de cada 'order_id' (total de filas por cada 'order_id')
order_sizes = DF_PRUEBA.groupby('order_id').size().
 ↪reset_index(name='order_size')

# Calcular los porcentajes de nivel dentro de cada grupo y redondear a 2␣
 ↪decimales
group_sizes['porcentaje_level_ONE'] = ((group_sizes['group_size'] /␣
 ↪group_sizes['order_id'].map(order_sizes.
 ↪set_index('order_id')['order_size'])) * 100).round(2)

# Eliminar la columna 'group_size' que ya no es necesaria
group_sizes.drop('group_size', axis=1, inplace=True)

# Fusionar los porcentajes calculados con el DataFrame original 'df' en función␣
 ↪del 'order_id'
DF_PRUEBA = DF_PRUEBA.merge(group_sizes.pivot(index='order_id',␣
 ↪columns='level_one', values='porcentaje_level_ONE'), on='order_id',␣
 ↪how='left')

# Eliminar duplicados de 'order_id' manteniendo solo el primer registro
```

```
DF_PRUEBA = DF_PRUEBA.drop_duplicates(subset=['order_id'])

# Completar NaN con 0 en las columnas generadas por la fusión
level_one_columns = group_sizes['level_one'].unique()
DF_PRUEBA[level_one_columns] = DF_PRUEBA[level_one_columns].fillna(0)

# Calcular cantidades según los porcentajes
for level in level_one_columns:
    DF_PRUEBA[level + '_qty'] = (DF_PRUEBA[level] / 100) *␣
 ↪DF_PRUEBA['qty_total_products']

# Suma de las cantidades calculadas por cada 'order_id'
sum_columns = [col + '_qty' for col in level_one_columns]
DF_PRUEBA['sum_qty'] = DF_PRUEBA[sum_columns].sum(axis=1)

# Ajustar las cantidades para que sumen exactamente 'qty_total_products'
for level in level_one_columns:
    DF_PRUEBA[level + '_qty'] *= (DF_PRUEBA['qty_total_products'] /␣
 ↪DF_PRUEBA['sum_qty'])

# Eliminar columnas auxiliares
DF_PRUEBA.drop(['sum_qty'], axis=1, inplace=True)

# Mostrar el DataFrame resultante
DF_PRUEBA
```

[70]:

	order_id	weekday	hour	\
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	
3	GdvT/R8JqkIOzSzfwVrVag6okmBzOOms88KxgAseht8=	Saturday	23	
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	
…	…	…	…	
860979	Ppsk/1iIoA++xYkWnLdOpAz9xnGqy/pStEmwX7hBfSY=	Wednesday	11	
861007	aFJuTDgA5K4j8CGuZIvhcPL2FEs6Umsj9mMO1eGr6Vo=	Sunday	18	
861037	MF2dm3i2fs87y04GYfIJ60pZJCwQ5nn02labBd03B+k=	Saturday	12	
861062	xHcIBn68bgDJbVcgOGxrnza6eiRl2TDhfbTjXLz2zGQ=	Friday	11	
861092	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	

	user_id	qty_total_products	\
0	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	1	
1	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	1	
2	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	
3	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	
4	Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=	1	
…	…	…	
860979	Ob2zUnYtV93bePpU43mfNRfyT3TOvZS/0xItVGz1rj8=	55	

```
861007  Qo7XOwW3phLImqmhdtTfbeWqJtnd6M+Vwfdcx8bBPT4=                  55
861037  S0Cb9TupUCxJ8GGEz5IYMKmnHJfaNNXGmSPpI2pAuHA=                  55
861062  Bode5766jynMCiSW9lhAiNaGRNfGscGezXnecnLHM5Q=                  55
861092  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                  55

        total_amount has_discount          level_one  order_count_usu  \
0               8.05        False  Smoking / Tobacco                1
1               7.91        False          Beverages                2
2               7.85        False          Beverages                7
3               8.60        False  Smoking / Tobacco                7
4               8.38        False     Packaged Foods                1
…                …           …                 …              …
860979        140.28         True            Produce               28
861007        112.39         True            Produce               32
861037        164.79         True     Packaged Foods                6
861062        107.30         True          Beverages               28
861092        188.96         True     Packaged Foods               30

       frec_compra_usu  … Produce_qty Beverages_qty  Snacks_qty  Frozen_qty  \
0                 Baja  …        0.00          0.00        0.00        0.00
1                 Baja  …        0.00          1.00        0.00        0.00
2                Media  …        0.00          1.00        0.00        0.00
3                Media  …        0.00          0.00        0.00        0.00
4                 Baja  …        0.00          0.00        0.00        0.00
…                 …  …          …             …          …          …
860979            Alta  …       13.75          3.93        5.89        0.00
861007            Alta  …        9.17          9.17       12.83        0.00
861037           Media  …        0.00          2.20        4.40        4.40
861062            Alta  …        0.00          7.33        0.00        0.00
861092            Alta  …       16.30          0.00        4.08        4.08

        Packaged Foods_qty  Smoking / Tobacco_qty  \
0                     0.00                   1.00
1                     0.00                   0.00
2                     0.00                   0.00
3                     0.00                   1.00
4                     1.00                   0.00
…                    …                      …
860979               19.64                   0.00
861007                9.17                   0.00
861037               17.60                   0.00
861062               18.33                   0.00
861092               18.33                   0.00

        Personal Care / Baby / Health_qty  Bread / Bakery_qty  \
0                                    0.00                0.00
1                                    0.00                0.00
```

```
2                                    0.00                0.00
3                                    0.00                0.00
4                                    0.00                0.00
…                                     …                   …
860979                               0.00                0.00
861007                               0.00                0.00
861037                               2.20                2.20
861062                              11.00                0.00
861092                               2.04                4.08

        General Merchandise_qty  Ready To Consume_qty
0                          0.00                  0.00
1                          0.00                  0.00
2                          0.00                  0.00
3                          0.00                  0.00
4                          0.00                  0.00
…                           …                     …
860979                     0.00                  0.00
861007                     0.00                  0.00
861037                     0.00                  6.60
861062                     0.00                  0.00
861092                     0.00                  0.00

[196456 rows x 40 columns]
```

[71]:
```
df = DF_PRUEBA.copy()
```

[72]:
```
df.columns
```

[72]:
```
Index(['order_id', 'weekday', 'hour', 'user_id', 'qty_total_products',
       'total_amount', 'has_discount', 'level_one', 'order_count_usu',
       'frec_compra_usu', 'es_fin_de_semana', 'periodo_dia', 'BWS',
       'Beverages', 'Bread / Bakery', 'Dairy / Chilled / Eggs', 'Frozen',
       'General Merchandise', 'Home / Pet', 'Meat / Seafood', 'Packaged Foods',
       'Personal Care / Baby / Health', 'Produce', 'Ready To Consume',
       'Smoking / Tobacco', 'Snacks', 'BWS_qty', 'Dairy / Chilled / Eggs_qty',
       'Home / Pet_qty', 'Meat / Seafood_qty', 'Produce_qty', 'Beverages_qty',
       'Snacks_qty', 'Frozen_qty', 'Packaged Foods_qty',
       'Smoking / Tobacco_qty', 'Personal Care / Baby / Health_qty',
       'Bread / Bakery_qty', 'General Merchandise_qty',
       'Ready To Consume_qty'],
      dtype='object')
```

[73]:
```
pd.isnull(df).describe()
```

[73]:
```
        order_id weekday    hour user_id qty_total_products total_amount  \
count     196456  196456  196456  196456             196456       196456
```

```
unique        1       1       1       1                  1            1
top       False   False   False   False              False        False
freq     196456  196456  196456  196456             196456       196456

       has_discount level_one order_count_usu frec_compra_usu  …  \
count        196456    196456          196456          196456  …
unique            1         1               1               1  …
top           False     False           False           False  …
freq         196456    196456          196456          196456  …

       Produce_qty Beverages_qty Snacks_qty Frozen_qty Packaged Foods_qty  \
count       196456        196456     196456     196456             196456
unique           1             1          1          1                  1
top          False         False      False      False              False
freq        196456        196456     196456     196456             196456

       Smoking / Tobacco_qty Personal Care / Baby / Health_qty  \
count                 196456                            196456
unique                     1                                 1
top                    False                             False
freq                  196456                            196456

       Bread / Bakery_qty General Merchandise_qty Ready To Consume_qty
count              196456                  196456               196456
unique                  1                       1                    1
top                 False                   False                False
freq               196456                  196456               196456

[4 rows x 40 columns]
```

4.1.1 Cantidad de categorias presentes de level one en la orden

[74]:
```
# Lista de columnas a evaluar
columnas_categorias = [
    'BWS', 'Beverages','Bread / Bakery', 'Dairy / Chilled / Eggs', 'Frozen',
    'General Merchandise', 'Home / Pet', 'Meat / Seafood', 'Packaged Foods',
    'Personal Care / Baby / Health', 'Produce', 'Ready To Consume','Smoking /␣
 ↪Tobacco', 'Snacks',
]

# Función para contar el número de categorías presentes en una fila
def contar_categorias_presentes(fila):
    return (fila[columnas_categorias] > 0).sum()

# Crear una nueva columna con el número total de categorías presentes
df['cant_cat_L1'] = df.apply(contar_categorias_presentes, axis=1)
```

[75]:
```
df
```

[75]:

	order_id	weekday	hour	\
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	
3	GdvT/R8JqkIOzSzfwVrVag6okmBzOOms88KxgAseht8=	Saturday	23	
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	
…	…	…	…	
860979	Ppsk/1iIoA++xYkWnLdOpAz9xnGqy/pStEmwX7hBfSY=	Wednesday	11	
861007	aFJuTDgA5K4j8CGuZIvhcPL2FEs6Umsj9mMO1eGr6Vo=	Sunday	18	
861037	MF2dm3i2fs87y04GYfIJ60pZJCwQ5nn02labBd03B+k=	Saturday	12	
861062	xHcIBn68bgDJbVcgOGxrnza6eiRl2TDhfbTjXLz2zGQ=	Friday	11	
861092	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	

	user_id	qty_total_products	\
0	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	1	
1	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	1	
2	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	
3	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	1	
4	Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=	1	
…	…	…	
860979	Ob2zUnYtV93bePpU43mfNRfyT3TOvZS/0xItVGz1rj8=	55	
861007	Qo7XOwW3phLImqmhdtTfbeWqJtnd6M+Vwfdcx8bBPT4=	55	
861037	SOCb9TupUCxJ8GGEz5IYMKmnHJfaNNXGmSPpI2pAuHA=	55	
861062	Bode5766jynMCiSW9lhAiNaGRNfGscGezXnecnLHM5Q=	55	
861092	D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=	55	

	total_amount	has_discount	level_one	order_count_usu	\
0	8.05	False	Smoking / Tobacco	1	
1	7.91	False	Beverages	2	
2	7.85	False	Beverages	7	

```
3              8.60        False  Smoking / Tobacco                7
4              8.38        False     Packaged Foods                1
…               …           …                 …                 …
860979       140.28         True            Produce               28
861007       112.39         True            Produce               32
861037       164.79         True     Packaged Foods                6
861062       107.30         True          Beverages               28
861092       188.96         True     Packaged Foods               30

       frec_compra_usu  … Beverages_qty Snacks_qty  Frozen_qty  \
0                 Baja  …          0.00       0.00        0.00
1                 Baja  …          1.00       0.00        0.00
2                Media  …          1.00       0.00        0.00
3                Media  …          0.00       0.00        0.00
4                 Baja  …          0.00       0.00        0.00
…                 …  …           …         …          …
860979            Alta  …          3.93       5.89        0.00
861007            Alta  …          9.17      12.83        0.00
861037           Media  …          2.20       4.40        4.40
861062            Alta  …          7.33       0.00        0.00
861092            Alta  …          0.00       4.08        4.08

        Packaged Foods_qty  Smoking / Tobacco_qty  \
0                     0.00                   1.00
1                     0.00                   0.00
2                     0.00                   0.00
3                     0.00                   1.00
4                     1.00                   0.00
…                    …                      …
860979               19.64                   0.00
861007                9.17                   0.00
861037               17.60                   0.00
861062               18.33                   0.00
861092               18.33                   0.00

        Personal Care / Baby / Health_qty  Bread / Bakery_qty  \
0                                    0.00                0.00
1                                    0.00                0.00
2                                    0.00                0.00
3                                    0.00                0.00
4                                    0.00                0.00
…                                   …                   …
860979                               0.00                0.00
861007                               0.00                0.00
861037                               2.20                2.20
861062                              11.00                0.00
861092                               2.04                4.08
```

```
        General Merchandise_qty  Ready To Consume_qty  cant_cat_L1
0                          0.00                  0.00            1
1                          0.00                  0.00            1
2                          0.00                  0.00            1
3                          0.00                  0.00            1
4                          0.00                  0.00            1
…                           …                     …          …
860979                     0.00                  0.00            6
861007                     0.00                  0.00            6
861037                     0.00                  6.60           10
861062                     0.00                  0.00            4
861092                     0.00                  0.00            7

[196456 rows x 41 columns]
```

4.2 Cantidad de dinero promedio por cada categoria de la orden, cada columna queda asignada con el prefijo peso_ antes del nombre de la columna.

4.2.1 Se define para cuantificar el costo monetario promedio de cada producto de acuerdo al costo total de la orden.

(suponiendo que cada articulo pesa lo mismo en el costo de la orden)

[76]:
```
# Identificar las columnas de porcentaje
columnas_porcentaje = []

# Calcular el peso en dinero para cada columna de porcentaje
for col in columnas_categorias:
    df[f'peso_{col}'] = df[col] / 100 * df['total_amount']
```

4.3 Filtramos por order_id para verificar la asignación de porcentajes por cada categoria de la órden

[77]:
```
# Filtrar el DataFrame para el 'order_id' específico
df_filtrado = df[df['order_id'] == '1HDYnLq4GrVCib7qF00S1yOXA4hA6HlN/
 ↪wK26V2cRMo=']

# Mostrar el DataFrame filtrado

columnas_a_mostrar = columnas_categorias + ['cant_cat_L1']
df_columnas = df_filtrado[columnas_a_mostrar]
df_columnas
```

[77]:
```
Empty DataFrame
Columns: [BWS, Beverages, Bread / Bakery, Dairy / Chilled / Eggs, Frozen,
General Merchandise, Home / Pet, Meat / Seafood, Packaged Foods, Personal Care /
Baby / Health, Produce, Ready To Consume, Smoking / Tobacco, Snacks,
cant_cat_L1]
Index: []
```

4.4 Filtramos por order_id para verificar la asignación de cantidades por cada categoria de la órden

[78]:
```
# Filtrar el DataFrame para el 'order_id' específico
df_filtrado = df[df['order_id'] == '1HDYnLq4GrVCib7qF0OS1yOXA4hA6HlN/
↪wK26V2cRMo=']

# Mostrar el DataFrame filtrado

columnas_a_mostrar = [f'peso_{col}' for col in columnas_categorias] +␣
↪['total_amount']
df_columnas = df_filtrado[columnas_a_mostrar]
df_columnas
```

[78]:
```
Empty DataFrame
Columns: [peso_BWS, peso_Beverages, peso_Bread / Bakery, peso_Dairy / Chilled /
Eggs, peso_Frozen, peso_General Merchandise, peso_Home / Pet, peso_Meat /
Seafood, peso_Packaged Foods, peso_Personal Care / Baby / Health, peso_Produce,
peso_Ready To Consume, peso_Smoking / Tobacco, peso_Snacks, total_amount]
Index: []
```

[79]:
```
df
```

[79]:

	order_id	weekday	hour	\
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	14	
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	0	
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	22	
3	GdvT/R8JqkIOzSzfwVrVag6okmBzOOms88KxgAseht8=	Saturday	23	
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	14	
…	…	…	…	
860979	Ppsk/1iIoA++xYkWnLd0pAz9xnGqy/pStEmwX7hBfSY=	Wednesday	11	
861007	aFJuTDgA5K4j8CGuZIvhcPL2FEs6Umsj9mMO1eGr6Vo=	Sunday	18	
861037	MF2dm3i2fs87y04GYfIJ60pZJCwQ5nn02labBd03B+k=	Saturday	12	
861062	xHcIBn68bgDJbVcgOGxrnza6eiRl2TDhfbTjXLz2zGQ=	Friday	11	
861092	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	19	

	user_id	qty_total_products	\
0	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	1	
1	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	1	

```
2       Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=                    1
3       Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=                    1
4       Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=                    1
…                                                    …                …
860979  Ob2zUnYtV93bePpU43mfNRfyT3TOvZS/0xItVGz1rj8=                   55
861007  Qo7XOwW3phLImqmhdtTfbeWqJtnd6M+Vwfdcx8bBPT4=                   55
861037  S0Cb9TupUCxJ8GGEz5IYMKmnHJfaNNXGmSPpI2pAuHA=                   55
861062  Bode5766jynMCiSW9lhAiNaGRNfGscGezXnecnLHM5Q=                   55
861092  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=                   55

        total_amount has_discount          level_one  order_count_usu  \
0               8.05        False  Smoking / Tobacco                1
1               7.91        False          Beverages                2
2               7.85        False          Beverages                7
3               8.60        False  Smoking / Tobacco                7
4               8.38        False     Packaged Foods                1
…                …           …                  …               …
860979        140.28         True            Produce               28
861007        112.39         True            Produce               32
861037        164.79         True     Packaged Foods                6
861062        107.30         True          Beverages               28
861092        188.96         True     Packaged Foods               30

       frec_compra_usu  … peso_Frozen peso_General Merchandise  \
0                 Baja  …        0.00                     0.00
1                 Baja  …        0.00                     0.00
2                Media  …        0.00                     0.00
3                Media  …        0.00                     0.00
4                 Baja  …        0.00                     0.00
…                 …  …         …                        …
860979            Alta  …        0.00                     0.00
861007            Alta  …        0.00                     0.00
861037           Media  …       13.18                     0.00
861062            Alta  …        0.00                     0.00
861092            Alta  …       14.00                     0.00

        peso_Home / Pet  peso_Meat / Seafood  peso_Packaged Foods  \
0                  0.00                 0.00                 0.00
1                  0.00                 0.00                 0.00
2                  0.00                 0.00                 0.00
3                  0.00                 0.00                 0.00
4                  0.00                 0.00                 8.38
…                  …                    …                    …
860979             5.01                 0.00                50.09
861007            11.24                 0.00                18.74
861037             0.00                 6.59                52.73
861062            35.76                 0.00                35.76
```

```
861092             0.00                  0.00                62.98

        peso_Personal Care / Baby / Health  peso_Produce  \
0                                     0.00          0.00
1                                     0.00          0.00
2                                     0.00          0.00
3                                     0.00          0.00
4                                     0.00          0.00
…                                      …             …
860979                                0.00         35.07
861007                                0.00         18.74
861037                                6.59          0.00
861062                               21.46          0.00
861092                                6.99         55.99

        peso_Ready To Consume  peso_Smoking / Tobacco  peso_Snacks
0                        0.00                    8.05         0.00
1                        0.00                    0.00         0.00
2                        0.00                    0.00         0.00
3                        0.00                    8.60         0.00
4                        0.00                    0.00         0.00
…                         …                       …            …
860979                   0.00                    0.00        15.02
861007                   0.00                    0.00        26.22
861037                  19.77                    0.00        13.18
861062                   0.00                    0.00         0.00
861092                   0.00                    0.00        14.00

[196456 rows x 55 columns]
```

[80]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 196456 entries, 0 to 861092
Data columns (total 55 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   order_id                            196456 non-null  object
 1   weekday                             196456 non-null  category
 2   hour                                196456 non-null  int64
 3   user_id                             196456 non-null  object
 4   qty_total_products                  196456 non-null  int64
 5   total_amount                        196456 non-null  float64
 6   has_discount                        196456 non-null  category
 7   level_one                           196456 non-null  object
 8   order_count_usu                     196456 non-null  int64
 9   frec_compra_usu                     196456 non-null  category
```

```
 10  es_fin_de_semana                    196456 non-null  category
 11  periodo_dia                         196456 non-null  category
 12  BWS                                 196456 non-null  float64
 13  Beverages                           196456 non-null  float64
 14  Bread / Bakery                      196456 non-null  float64
 15  Dairy / Chilled / Eggs              196456 non-null  float64
 16  Frozen                              196456 non-null  float64
 17  General Merchandise                 196456 non-null  float64
 18  Home / Pet                          196456 non-null  float64
 19  Meat / Seafood                      196456 non-null  float64
 20  Packaged Foods                      196456 non-null  float64
 21  Personal Care / Baby / Health       196456 non-null  float64
 22  Produce                             196456 non-null  float64
 23  Ready To Consume                    196456 non-null  float64
 24  Smoking / Tobacco                   196456 non-null  float64
 25  Snacks                              196456 non-null  float64
 26  BWS_qty                             196456 non-null  float64
 27  Dairy / Chilled / Eggs_qty          196456 non-null  float64
 28  Home / Pet_qty                      196456 non-null  float64
 29  Meat / Seafood_qty                  196456 non-null  float64
 30  Produce_qty                         196456 non-null  float64
 31  Beverages_qty                       196456 non-null  float64
 32  Snacks_qty                          196456 non-null  float64
 33  Frozen_qty                          196456 non-null  float64
 34  Packaged Foods_qty                  196456 non-null  float64
 35  Smoking / Tobacco_qty               196456 non-null  float64
 36  Personal Care / Baby / Health_qty   196456 non-null  float64
 37  Bread / Bakery_qty                  196456 non-null  float64
 38  General Merchandise_qty             196456 non-null  float64
 39  Ready To Consume_qty                196456 non-null  float64
 40  cant_cat_L1                         196456 non-null  int64
 41  peso_BWS                            196456 non-null  float64
 42  peso_Beverages                      196456 non-null  float64
 43  peso_Bread / Bakery                 196456 non-null  float64
 44  peso_Dairy / Chilled / Eggs         196456 non-null  float64
 45  peso_Frozen                         196456 non-null  float64
 46  peso_General Merchandise            196456 non-null  float64
 47  peso_Home / Pet                     196456 non-null  float64
 48  peso_Meat / Seafood                 196456 non-null  float64
 49  peso_Packaged Foods                 196456 non-null  float64
 50  peso_Personal Care / Baby / Health  196456 non-null  float64
 51  peso_Produce                        196456 non-null  float64
 52  peso_Ready To Consume               196456 non-null  float64
 53  peso_Smoking / Tobacco              196456 non-null  float64
 54  peso_Snacks                         196456 non-null  float64
dtypes: category(5), float64(43), int64(4), object(3)
memory usage: 77.4+ MB
```

[81]:
```
df.columns
```

[81]:
```
Index(['order_id', 'weekday', 'hour', 'user_id', 'qty_total_products',
       'total_amount', 'has_discount', 'level_one', 'order_count_usu',
       'frec_compra_usu', 'es_fin_de_semana', 'periodo_dia', 'BWS',
       'Beverages', 'Bread / Bakery', 'Dairy / Chilled / Eggs', 'Frozen',
       'General Merchandise', 'Home / Pet', 'Meat / Seafood', 'Packaged Foods',
       'Personal Care / Baby / Health', 'Produce', 'Ready To Consume',
       'Smoking / Tobacco', 'Snacks', 'BWS_qty', 'Dairy / Chilled / Eggs_qty',
       'Home / Pet_qty', 'Meat / Seafood_qty', 'Produce_qty', 'Beverages_qty',
       'Snacks_qty', 'Frozen_qty', 'Packaged Foods_qty',
       'Smoking / Tobacco_qty', 'Personal Care / Baby / Health_qty',
       'Bread / Bakery_qty', 'General Merchandise_qty', 'Ready To Consume_qty',
       'cant_cat_L1', 'peso_BWS', 'peso_Beverages', 'peso_Bread / Bakery',
       'peso_Dairy / Chilled / Eggs', 'peso_Frozen',
       'peso_General Merchandise', 'peso_Home / Pet', 'peso_Meat / Seafood',
       'peso_Packaged Foods', 'peso_Personal Care / Baby / Health',
       'peso_Produce', 'peso_Ready To Consume', 'peso_Smoking / Tobacco',
       'peso_Snacks'],
      dtype='object')
```

[82]:
```
numericas = []
otras = []
```

4.4.1 Creacion de un dataframe numerico:

[83]:
```
# Seleccionar solo las columnas numéricas
numeric_df = df.select_dtypes(include=[float, int])

df_otras = df.select_dtypes(exclude=[float, int])
```

[84]:
```
numeric_df = numeric_df.drop(columns=['hour'])
```

[85]:
```
numeric_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 196456 entries, 0 to 861092
Data columns (total 46 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   qty_total_products              196456 non-null  int64
 1   total_amount                    196456 non-null  float64
 2   order_count_usu                 196456 non-null  int64
 3   BWS                             196456 non-null  float64
 4   Beverages                       196456 non-null  float64
 5   Bread / Bakery                  196456 non-null  float64
```

```
 6   Dairy / Chilled / Eggs              196456 non-null  float64
 7   Frozen                              196456 non-null  float64
 8   General Merchandise                 196456 non-null  float64
 9   Home / Pet                          196456 non-null  float64
 10  Meat / Seafood                      196456 non-null  float64
 11  Packaged Foods                      196456 non-null  float64
 12  Personal Care / Baby / Health       196456 non-null  float64
 13  Produce                             196456 non-null  float64
 14  Ready To Consume                    196456 non-null  float64
 15  Smoking / Tobacco                   196456 non-null  float64
 16  Snacks                              196456 non-null  float64
 17  BWS_qty                             196456 non-null  float64
 18  Dairy / Chilled / Eggs_qty          196456 non-null  float64
 19  Home / Pet_qty                      196456 non-null  float64
 20  Meat / Seafood_qty                  196456 non-null  float64
 21  Produce_qty                         196456 non-null  float64
 22  Beverages_qty                       196456 non-null  float64
 23  Snacks_qty                          196456 non-null  float64
 24  Frozen_qty                          196456 non-null  float64
 25  Packaged Foods_qty                  196456 non-null  float64
 26  Smoking / Tobacco_qty               196456 non-null  float64
 27  Personal Care / Baby / Health_qty   196456 non-null  float64
 28  Bread / Bakery_qty                  196456 non-null  float64
 29  General Merchandise_qty             196456 non-null  float64
 30  Ready To Consume_qty                196456 non-null  float64
 31  cant_cat_L1                         196456 non-null  int64
 32  peso_BWS                            196456 non-null  float64
 33  peso_Beverages                      196456 non-null  float64
 34  peso_Bread / Bakery                 196456 non-null  float64
 35  peso_Dairy / Chilled / Eggs         196456 non-null  float64
 36  peso_Frozen                         196456 non-null  float64
 37  peso_General Merchandise            196456 non-null  float64
 38  peso_Home / Pet                     196456 non-null  float64
 39  peso_Meat / Seafood                 196456 non-null  float64
 40  peso_Packaged Foods                 196456 non-null  float64
 41  peso_Personal Care / Baby / Health  196456 non-null  float64
 42  peso_Produce                        196456 non-null  float64
 43  peso_Ready To Consume               196456 non-null  float64
 44  peso_Smoking / Tobacco              196456 non-null  float64
 45  peso_Snacks                         196456 non-null  float64
dtypes: float64(43), int64(3)
memory usage: 70.4 MB
```

[86]:
```python
# Recorrer el DataFrame para identificar columnas numéricas
for col in df.columns:
    if pd.api.types.is_numeric_dtype(df[col]):
        numericas.append(col)
```

[87]:
```
# Recorrer el DataFrame para identificar columnas categóricas
for col in df.columns:
    if not pd.api.types.is_numeric_dtype(df[col]):
        otras.append(col)
```

[88]:
```
numericas
```

[88]:
```
['hour',
 'qty_total_products',
 'total_amount',
 'order_count_usu',
 'BWS',
 'Beverages',
 'Bread / Bakery',
 'Dairy / Chilled / Eggs',
 'Frozen',
 'General Merchandise',
 'Home / Pet',
 'Meat / Seafood',
 'Packaged Foods',
 'Personal Care / Baby / Health',
 'Produce',
 'Ready To Consume',
 'Smoking / Tobacco',
 'Snacks',
 'BWS_qty',
 'Dairy / Chilled / Eggs_qty',
 'Home / Pet_qty',
 'Meat / Seafood_qty',
 'Produce_qty',
 'Beverages_qty',
 'Snacks_qty',
 'Frozen_qty',
 'Packaged Foods_qty',
 'Smoking / Tobacco_qty',
 'Personal Care / Baby / Health_qty',
 'Bread / Bakery_qty',
 'General Merchandise_qty',
 'Ready To Consume_qty',
 'cant_cat_L1',
 'peso_BWS',
 'peso_Beverages',
 'peso_Bread / Bakery',
 'peso_Dairy / Chilled / Eggs',
 'peso_Frozen',
 'peso_General Merchandise',
 'peso_Home / Pet',
```

```
 'peso_Meat / Seafood',
 'peso_Packaged Foods',
 'peso_Personal Care / Baby / Health',
 'peso_Produce',
 'peso_Ready To Consume',
 'peso_Smoking / Tobacco',
 'peso_Snacks']
```

[89]:
```
otras
```

[89]:
```
['order_id',
 'weekday',
 'user_id',
 'has_discount',
 'level_one',
 'frec_compra_usu',
 'es_fin_de_semana',
 'periodo_dia']
```

4.5 Observamos las frecuencias de las variables que no son numericas.

[90]:
```
# Una visión rápida de las variables categóricas
cat_vars = otras


for v in cat_vars:
  print('\n{}'.format(v))
  print(df[v].value_counts()) # frecuencia abosluta
  print(df[v].value_counts()/df[v].value_counts().sum()) # frecuencia relativa
```

```
order_id
MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=    1
6KNROzggJk7M3rME7esDKh8jGYJCXGYGReCvNqNeTRI=    1
buNl/ZOooKiCb1S9vIke5rqfNN6MUiOqhhipIonAdf4=    1
h386zbRlBt/IHD79CfcfON8uOWswOhR/g8e2HF/6/dk=    1
PjOWeyuYbYnrxPdKHb4NeY6AKrPFIalynasm/wWbqYs=    1
                                               ..
KxX/fcoCymW6XSvoxejWWkDwbR9zhd+UwstOOhmJnmg=    1
uAdwnEjPWplNWvfSZeeDeHVnAmSL8nfOYgPdKofPHrw=    1
Eglwq8FUCmOM76iC7u73MQVLENCuF8c+dTE4wKBYPt4=    1
5ugYdj39zVmA82zmkNqBuMFsRkLuvQvt/QaORjgXbyA=    1
6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=    1
Name: order_id, Length: 196456, dtype: int64
MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=   0.00
6KNROzggJk7M3rME7esDKh8jGYJCXGYGReCvNqNeTRI=   0.00
```

```
buNl/ZOooKiCb1S9vIke5rqfNN6MUiOqhhipIonAdf4=   0.00
h386zbRlBt/IHD79CfcfON8uOWswOhR/g8e2HF/6/dk=   0.00
PjOWeyuYbYnrxPdKHb4NeY6AKrPFIalynasm/wWbqYs=   0.00
                                              …
KxX/fcoCymW6XSvoxejWWkDwbR9zhd+UwstOOhmJnmg=   0.00
uAdwnEjPWplNWvfSZeeDeHVnAmSL8nfOYgPdKofPHrw=   0.00
Eglwq8FUCmOM76iC7u73MQVLENCuF8c+dTE4wKBYPt4=   0.00
5ugYdj39zVmA82zmkNqBuMFsRkLuvQvt/QaORjgXbyA=   0.00
6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=   0.00
Name: order_id, Length: 196456, dtype: float64

weekday
Saturday     29701
Tuesday      28698
Sunday       28586
Friday       28295
Monday       27284
Wednesday    27166
Thursday     26726
Name: weekday, dtype: int64
Saturday    0.15
Tuesday     0.15
Sunday      0.15
Friday      0.14
Monday      0.14
Wednesday   0.14
Thursday    0.14
Name: weekday, dtype: float64

user_id
rlw9AteZvxvrL6bJCNnohHyykqUwcz/FM2OdG1SDL1g=    224
j4QaIaQZoiT8FMBGrIFM6CoAMf0JygkJmOFORUdr3Ns=    203
lqAuWvkXRG7YgnUs2sFVW64OR8MFTKq6OZCBte/NFMU=    197
VF3KLyJW6qUiD2gNl2EKsr8iuysuJqlNhnSGemyugnc=    184
DW5m2gtmzlUDybv8ZI/I2Xg90xpF2e4g7L+cpIvGM1I=    178
                                               …
ibWEgCF66BzgLi2roX1RAEsv3dOWDIPKr+STXHmXXWQ=      1
ibMsdVUuIOd8keTLWb+FOPSoQYjEnLch78yJlyWFYgw=      1
iY1RZmJQ/IwHcowiHbIxIB5fIgU8R8304mgJ0uxQmk8=      1
iVjEB+OTUO+DIUJAbNpZ0+CbIxDZCQ42CByEiiBMcuY=      1
6jFLQeawB+NR8/huRqxuC6/n86AQE8BrV41iUDbdyMM=      1
Name: user_id, Length: 35050, dtype: int64
rlw9AteZvxvrL6bJCNnohHyykqUwcz/FM2OdG1SDL1g=   0.00
j4QaIaQZoiT8FMBGrIFM6CoAMf0JygkJmOFORUdr3Ns=   0.00
lqAuWvkXRG7YgnUs2sFVW64OR8MFTKq6OZCBte/NFMU=   0.00
VF3KLyJW6qUiD2gNl2EKsr8iuysuJqlNhnSGemyugnc=   0.00
DW5m2gtmzlUDybv8ZI/I2Xg90xpF2e4g7L+cpIvGM1I=   0.00
                                               …
```

```
ibWEgCF66BzgLi2roX1RAEsv3dOWDIPKr+STXHmXXWQ=   0.00
ibMsdVUuIOd8keTLWb+FOPSoQYjEnLch78yJlyWFYgw=   0.00
iY1RZmJQ/IwHcowiHbIxIB5fIgU8R8304mgJ0uxQmk8=   0.00
iVjEB+OTUO+DIUJAbNpZ0+CbIxDZCQ42CByEiiBMcuY=   0.00
6jFLQeawB+NR8/huRqxuC6/n86AQE8BrV41iUDbdyMM=   0.00
Name: user_id, Length: 35050, dtype: float64

has_discount
True     125906
False     70550
Name: has_discount, dtype: int64
True    0.64
False   0.36
Name: has_discount, dtype: float64

level_one
Snacks                          40321
Beverages                       33201
Packaged Foods                  24886
Home / Pet                      18690
Produce                         14962
Frozen                          12988
BWS                             11520
Smoking / Tobacco               10553
Dairy / Chilled / Eggs          10062
Meat / Seafood                   7328
Bread / Bakery                   7197
Personal Care / Baby / Health    4150
Ready To Consume                  300
General Merchandise               298
Name: level_one, dtype: int64
Snacks                          0.21
Beverages                       0.17
Packaged Foods                  0.13
Home / Pet                      0.10
Produce                         0.08
Frozen                          0.07
BWS                             0.06
Smoking / Tobacco               0.05
Dairy / Chilled / Eggs          0.05
Meat / Seafood                  0.04
Bread / Bakery                  0.04
Personal Care / Baby / Health   0.02
Ready To Consume                0.00
General Merchandise             0.00
Name: level_one, dtype: float64

frec_compra_usu
```

```
Alta     135110
Baja      34680
Media     26666
Name: frec_compra_usu, dtype: int64
Alta    0.69
Baja    0.18
Media   0.14
Name: frec_compra_usu, dtype: float64

es_fin_de_semana
False    138169
True      58287
Name: es_fin_de_semana, dtype: int64
False   0.70
True    0.30
Name: es_fin_de_semana, dtype: float64

periodo_dia
Nocturno      92443
Vespertino    72982
Matutino      31031
Name: periodo_dia, dtype: int64
Nocturno     0.47
Vespertino   0.37
Matutino     0.16
Name: periodo_dia, dtype: float64
```

[91]:
```
numeric_df.head()
```

[91]:

	qty_total_products	total_amount	order_count_usu	BWS	Beverages	\
0	1	8.05	1	0.00	0.00	
1	1	7.91	2	0.00	100.00	
2	1	7.85	7	0.00	100.00	
3	1	8.60	7	0.00	0.00	
4	1	8.38	1	0.00	0.00	

	Bread / Bakery	Dairy / Chilled / Eggs	Frozen	General Merchandise	\
0	0.00	0.00	0.00	0.00	
1	0.00	0.00	0.00	0.00	
2	0.00	0.00	0.00	0.00	
3	0.00	0.00	0.00	0.00	
4	0.00	0.00	0.00	0.00	

	Home / Pet	…	peso_Frozen	peso_General Merchandise	peso_Home / Pet	\
0	0.00	…	0.00	0.00	0.00	
1	0.00	…	0.00	0.00	0.00	
2	0.00	…	0.00	0.00	0.00	

```
3        0.00  …          0.00                      0.00             0.00
4        0.00  …          0.00                      0.00             0.00

   peso_Meat / Seafood  peso_Packaged Foods  \
0                 0.00                 0.00
1                 0.00                 0.00
2                 0.00                 0.00
3                 0.00                 0.00
4                 0.00                 8.38

   peso_Personal Care / Baby / Health  peso_Produce  peso_Ready To Consume  \
0                                0.00          0.00                   0.00
1                                0.00          0.00                   0.00
2                                0.00          0.00                   0.00
3                                0.00          0.00                   0.00
4                                0.00          0.00                   0.00

   peso_Smoking / Tobacco  peso_Snacks
0                    8.05         0.00
1                    0.00         0.00
2                    0.00         0.00
3                    8.60         0.00
4                    0.00         0.00

[5 rows x 46 columns]
```

[92]:
```
numeric_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 196456 entries, 0 to 861092
Data columns (total 46 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   qty_total_products                  196456 non-null  int64
 1   total_amount                        196456 non-null  float64
 2   order_count_usu                     196456 non-null  int64
 3   BWS                                 196456 non-null  float64
 4   Beverages                           196456 non-null  float64
 5   Bread / Bakery                      196456 non-null  float64
 6   Dairy / Chilled / Eggs              196456 non-null  float64
 7   Frozen                              196456 non-null  float64
 8   General Merchandise                 196456 non-null  float64
 9   Home / Pet                          196456 non-null  float64
 10  Meat / Seafood                      196456 non-null  float64
 11  Packaged Foods                      196456 non-null  float64
 12  Personal Care / Baby / Health       196456 non-null  float64
 13  Produce                             196456 non-null  float64
```

```
 14  Ready To Consume                    196456 non-null  float64
 15  Smoking / Tobacco                   196456 non-null  float64
 16  Snacks                              196456 non-null  float64
 17  BWS_qty                             196456 non-null  float64
 18  Dairy / Chilled / Eggs_qty          196456 non-null  float64
 19  Home / Pet_qty                      196456 non-null  float64
 20  Meat / Seafood_qty                  196456 non-null  float64
 21  Produce_qty                         196456 non-null  float64
 22  Beverages_qty                       196456 non-null  float64
 23  Snacks_qty                          196456 non-null  float64
 24  Frozen_qty                          196456 non-null  float64
 25  Packaged Foods_qty                  196456 non-null  float64
 26  Smoking / Tobacco_qty               196456 non-null  float64
 27  Personal Care / Baby / Health_qty   196456 non-null  float64
 28  Bread / Bakery_qty                  196456 non-null  float64
 29  General Merchandise_qty             196456 non-null  float64
 30  Ready To Consume_qty                196456 non-null  float64
 31  cant_cat_L1                         196456 non-null  int64
 32  peso_BWS                            196456 non-null  float64
 33  peso_Beverages                      196456 non-null  float64
 34  peso_Bread / Bakery                 196456 non-null  float64
 35  peso_Dairy / Chilled / Eggs         196456 non-null  float64
 36  peso_Frozen                         196456 non-null  float64
 37  peso_General Merchandise            196456 non-null  float64
 38  peso_Home / Pet                     196456 non-null  float64
 39  peso_Meat / Seafood                 196456 non-null  float64
 40  peso_Packaged Foods                 196456 non-null  float64
 41  peso_Personal Care / Baby / Health  196456 non-null  float64
 42  peso_Produce                        196456 non-null  float64
 43  peso_Ready To Consume               196456 non-null  float64
 44  peso_Smoking / Tobacco              196456 non-null  float64
 45  peso_Snacks                         196456 non-null  float64
dtypes: float64(43), int64(3)
memory usage: 70.4 MB
```

[93]:
```
numeric_df.describe()
```

[93]:

	qty_total_products	total_amount	order_count_usu	BWS	Beverages	\
count	196456.00	196456.00	196456.00	196456.00	196456.00	
mean	8.26	27.33	26.02	5.20	14.07	
std	6.92	22.98	30.41	18.29	24.74	
min	1.00	0.00	1.00	0.00	0.00	
25%	4.00	13.11	6.00	0.00	0.00	
50%	7.00	19.49	15.00	0.00	0.00	
75%	11.00	33.24	35.00	0.00	22.22	
max	55.00	192.96	224.00	100.00	100.00	

```
       Bread / Bakery  Dairy / Chilled / Eggs    Frozen  General Merchandise  \
count       196456.00                196456.00 196456.00            196456.00
mean             2.60                    13.79      5.57                 0.24
std              8.78                    22.24     17.31                 3.85
min              0.00                     0.00      0.00                 0.00
25%              0.00                     0.00      0.00                 0.00
50%              0.00                     0.00      0.00                 0.00
75%              0.00                    25.00      0.00                 0.00
max            100.00                   100.00    100.00               100.00

       Home / Pet  …  peso_Frozen  peso_General Merchandise  \
count   196456.00  …    196456.00                 196456.00
mean         7.58  …         1.48                      0.06
std         19.21  …         4.78                      0.98
min          0.00  …         0.00                      0.00
25%          0.00  …         0.00                      0.00
50%          0.00  …         0.00                      0.00
75%          0.00  …         0.00                      0.00
max        100.00  …       126.80                     62.23

       peso_Home / Pet  peso_Meat / Seafood  peso_Packaged Foods  \
count       196456.00             196456.00            196456.00
mean             2.47                  1.22                 4.20
std              7.06                  6.15                 8.69
min              0.00                  0.00                 0.00
25%              0.00                  0.00                 0.00
50%              0.00                  0.00                 0.00
75%              0.00                  0.00                 5.64
max            176.83                190.96               190.37

       peso_Personal Care / Baby / Health  peso_Produce  \
count                           196456.00     196456.00
mean                                 1.30          1.93
std                                  4.97          6.21
min                                  0.00          0.00
25%                                  0.00          0.00
50%                                  0.00          0.00
75%                                  0.00          0.00
max                                164.28        127.54

       peso_Ready To Consume  peso_Smoking / Tobacco  peso_Snacks
count              196456.00               196456.00    196456.00
mean                    0.09                    1.06         3.88
std                     1.22                    4.14         7.27
min                     0.00                    0.00         0.00
25%                     0.00                    0.00         0.00
50%                     0.00                    0.00         0.00
```

```
75%                    0.00                    0.00         5.83
max                  100.83                  171.93       161.77

[8 rows x 46 columns]
```

4.6 Eliminamos las columnas de porcentajes, decidimos quedarnos con la de cantidad (*qty) y las que contienen el costo promedio peso.*

[94]:
```
columns_to_drop = [
    'BWS',
    'Beverages',
    'Bread / Bakery',
    'Dairy / Chilled / Eggs',
    'Frozen',
    'General Merchandise',
    'Home / Pet',
    'Meat / Seafood',
    'Packaged Foods',
    'Personal Care / Baby / Health',
    'Produce',
    'Ready To Consume',
    'Smoking / Tobacco',
    'Snacks'
]

numeric_df = numeric_df.drop(columns=columns_to_drop)

numeric_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 196456 entries, 0 to 861092
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   qty_total_products              196456 non-null  int64
 1   total_amount                    196456 non-null  float64
 2   order_count_usu                 196456 non-null  int64
 3   BWS_qty                         196456 non-null  float64
 4   Dairy / Chilled / Eggs_qty      196456 non-null  float64
 5   Home / Pet_qty                  196456 non-null  float64
 6   Meat / Seafood_qty              196456 non-null  float64
 7   Produce_qty                     196456 non-null  float64
 8   Beverages_qty                   196456 non-null  float64
 9   Snacks_qty                      196456 non-null  float64
 10  Frozen_qty                      196456 non-null  float64
 11  Packaged Foods_qty              196456 non-null  float64
```

```
 12  Smoking / Tobacco_qty                196456 non-null  float64
 13  Personal Care / Baby / Health_qty    196456 non-null  float64
 14  Bread / Bakery_qty                   196456 non-null  float64
 15  General Merchandise_qty              196456 non-null  float64
 16  Ready To Consume_qty                 196456 non-null  float64
 17  cant_cat_L1                          196456 non-null  int64
 18  peso_BWS                             196456 non-null  float64
 19  peso_Beverages                       196456 non-null  float64
 20  peso_Bread / Bakery                  196456 non-null  float64
 21  peso_Dairy / Chilled / Eggs          196456 non-null  float64
 22  peso_Frozen                          196456 non-null  float64
 23  peso_General Merchandise             196456 non-null  float64
 24  peso_Home / Pet                      196456 non-null  float64
 25  peso_Meat / Seafood                  196456 non-null  float64
 26  peso_Packaged Foods                  196456 non-null  float64
 27  peso_Personal Care / Baby / Health   196456 non-null  float64
 28  peso_Produce                         196456 non-null  float64
 29  peso_Ready To Consume                196456 non-null  float64
 30  peso_Smoking / Tobacco               196456 non-null  float64
 31  peso_Snacks                          196456 non-null  float64
dtypes: float64(29), int64(3)
memory usage: 49.5 MB
```

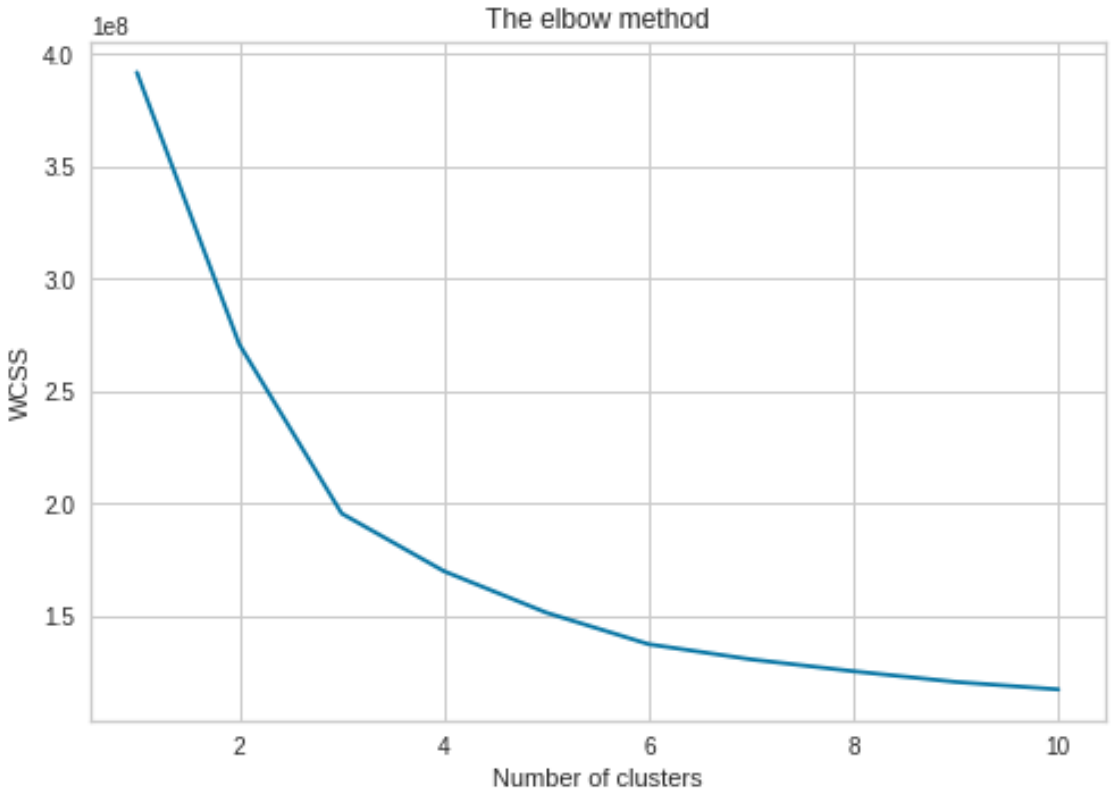
5 Aplicacion de los Modelos - K- Means y K-medoids clustering (PAM)

5.1 K- Means

[95]:
```
wcss = []

for i in range(1, tope_range):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = iteraciones,␣
 ↪n_init = 10, random_state = semilla)
    kmeans.fit(numeric_df)
    wcss.append(kmeans.inertia_)
```

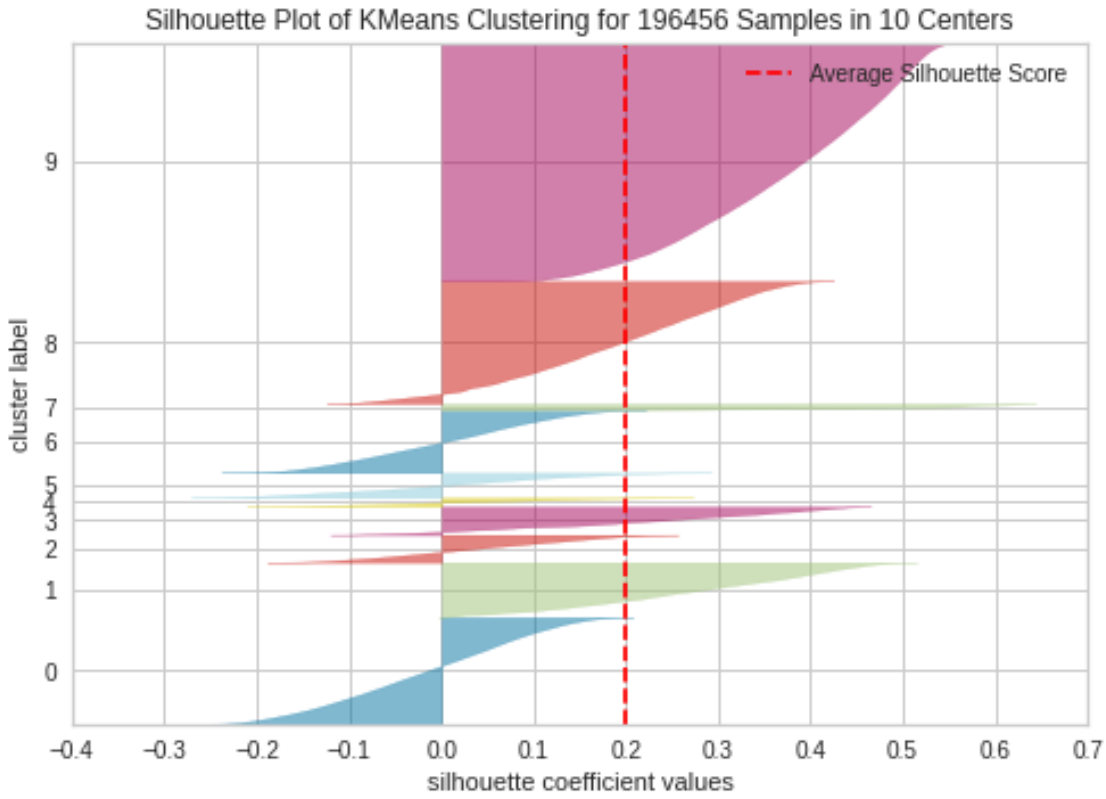
5.2 Utilizamos el Método del Códo - Elbow para visualizar el posible K óptimo

[96]:
```
plt.plot(range(1, tope_range), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```



5.3 También utilizamos Coeficiente de Silueta (Silhouette coefficient) para visualizar otro posible K óptimo

[97]:
```
model = KMeans(k_cluster)
visualizer = SilhouetteVisualizer(model, colors='yellowbrick')

visualizer.fit(numeric_df)
visualizer.show()
```



[97]:
```
<AxesSubplot:title={'center':'Silhouette Plot of KMeans Clustering for 196456
Samples in 10 Centers'}, xlabel='silhouette coefficient values', ylabel='cluster
label'>
```

5.4 Decidimos utilizar 3 clusters de acuerdo al método del codo.

[98]:
```
k_cluster = 3
```

5.4.1 Decidimos considerar el K que interpretamos de las gráficas, dejando comentado la opción de los coeficientes.

[99]:
```
'''
silhouette_scores = []

for k in range(2, tope_range):
    model = KMeans(n_clusters=k, random_state=semilla)
    model.fit(numeric_df)
    score = silhouette_score(numeric_df, model.labels_)
    print("Silhouette Score for k = ", k, "is", score)
```

```
    silhouette_scores.append(score)

'''
```

[99]:
```
'\nsilhouette_scores = []\n\nfor k in range(2, tope_range):\n    model =
KMeans(n_clusters=k, random_state=semilla)\n    model.fit(numeric_df)\n    score
= silhouette_score(numeric_df, model.labels_)\n    print("Silhouette Score for k
= ", k, "is", score)\n    silhouette_scores.append(score)\n\n'
```

[100]:
```
'''
plt.plot(range(2, tope_range), silhouette_scores, marker='o')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Silhouette score')
plt.savefig('silhouette plot.png')

'''
```

[100]:
```
"\nplt.plot(range(2, tope_range), silhouette_scores,
marker='o')\nplt.xlabel('Number of clusters (K)')\nplt.ylabel('Silhouette
score')\nplt.savefig('silhouette plot.png')\n\n"
```

[101]:
```
# Aplicamos el algoritmo de k-means
kmeans = KMeans(n_clusters=k_cluster, random_state=semilla).fit(numeric_df)
```

[102]:
```
# Miramos cuántas observaciones
cluster_sizes = np.bincount(kmeans.labels_)
print(cluster_sizes)
```

```
[143828  23989  28639]
```

[103]:
```
# Miramos los centroides de cada uno de los clusters
cluster_centers = kmeans.cluster_centers_
print(cluster_centers)
```

```
[[6.48582118e+00 2.00018238e+01 1.50593577e+01 3.12105718e-01
  9.64921056e-01 4.31407334e-01 1.01092743e-01 4.66769634e-01
  9.93127790e-01 1.32065294e+00 2.99910569e-01 9.13388663e-01
  2.48810085e-01 2.29688386e-01 1.74770729e-01 1.23358755e-02
  1.68396582e-02 2.22164583e+00 1.16455554e+00 2.77393875e+00
  5.60928262e-01 2.79746770e+00 1.20205373e+00 5.39753160e-02
  1.55392541e+00 6.81041207e-01 2.63643289e+00 9.21753249e-01
  1.20195410e+00 6.67114391e-02 1.08743090e+00 3.29963322e+00]
 [1.96174877e+01 7.58531993e+01 2.08846795e+01 5.37272146e-01
  3.79745169e+00 1.99791623e+00 6.30750422e-01 1.75365063e+00
  1.75232147e+00 2.52030697e+00 8.48092767e-01 4.00954768e+00
  9.12757620e-02 9.54140868e-01 6.32887465e-01 3.00181930e-02
  6.18554144e-02 4.34633658e+00 2.49401510e+00 6.09945460e+00
  2.42923598e+00 1.38833062e+01 3.66374954e+00 1.39489379e-01
```

```
  8.23897777e+00 4.17418467e+00 1.47293989e+01 4.15321795e+00
  6.38522519e+00 2.76925454e-01 5.21317027e-01 8.66403329e+00]
 [7.65351061e+00 2.34912472e+01 8.53208752e+01 3.82603052e-01
  1.22430697e+00 6.50811408e-01 2.22009447e-01 7.18125451e-01
  1.10679735e+00 1.11538735e+00 2.82519968e-01 1.13199936e+00
  3.42524393e-01 2.28593065e-01 2.14189464e-01 1.14009682e-02
  2.22423596e-02 2.52201982e+00 1.28548340e+00 2.99839534e+00
  6.64726108e-01 3.62987667e+00 1.05257762e+00 4.45563181e-02
  2.22634865e+00 1.42285043e+00 3.25773423e+00 8.43413329e-01
  1.85060064e+00 8.39189254e-02 1.37577649e+00 2.75496877e+00]]
```

[104]:
```
# Miramos la distribución de las observaciones clusterizadas
cluster_labels = kmeans.labels_
print(cluster_labels)
```

```
[0 0 0 … 1 1 1]
```

[105]:
```
# Calculamos el promedio de cada variable cuantitativa del dataset
promedio = numeric_df.mean()
print(promedio)
```

```
qty_total_products                  8.26
total_amount                       27.33
order_count_usu                    26.02
BWS_qty                             0.35
Dairy / Chilled / Eggs_qty          1.35
Home / Pet_qty                      0.65
Meat / Seafood_qty                  0.18
Produce_qty                         0.66
Beverages_qty                       1.10
Snacks_qty                          1.44
Frozen_qty                          0.36
Packaged Foods_qty                  1.32
Smoking / Tobacco_qty               0.24
Personal Care / Baby / Health_qty   0.32
Bread / Bakery_qty                  0.24
General Merchandise_qty             0.01
Ready To Consume_qty                0.02
cant_cat_L1                         2.52
peso_BWS                            1.34
peso_Beverages                      3.21
peso_Bread / Bakery                 0.80
peso_Dairy / Chilled / Eggs         4.27
peso_Frozen                         1.48
peso_General Merchandise            0.06
peso_Home / Pet                     2.47
peso_Meat / Seafood                 1.22
peso_Packaged Foods                 4.20
```

```
peso_Personal Care / Baby / Health    1.30
peso_Produce                          1.93
peso_Ready To Consume                 0.09
peso_Smoking / Tobacco                1.06
peso_Snacks                           3.88
dtype: float64
```

[106]:
```
len(cluster_sizes)
```

[106]:
```
3
```

[107]:
```
# Calculamos los promedios de cada cluster de cada una de las variables
cluster0 = numeric_df.loc[cluster_labels == 0].mean()
cluster1 = numeric_df.loc[cluster_labels == 1].mean()
cluster2 = numeric_df.loc[cluster_labels == 2].mean()

'''
cluster3 = numeric_df.loc[cluster_labels == 3].mean()
cluster4 = numeric_df.loc[cluster_labels == 4].mean()
cluster5 = numeric_df.loc[cluster_labels == 5].mean()
'''
```

[107]:
```
'\ncluster3 = numeric_df.loc[cluster_labels == 3].mean()\ncluster4 =
numeric_df.loc[cluster_labels == 4].mean()\ncluster5 =
numeric_df.loc[cluster_labels == 5].mean()\n'
```

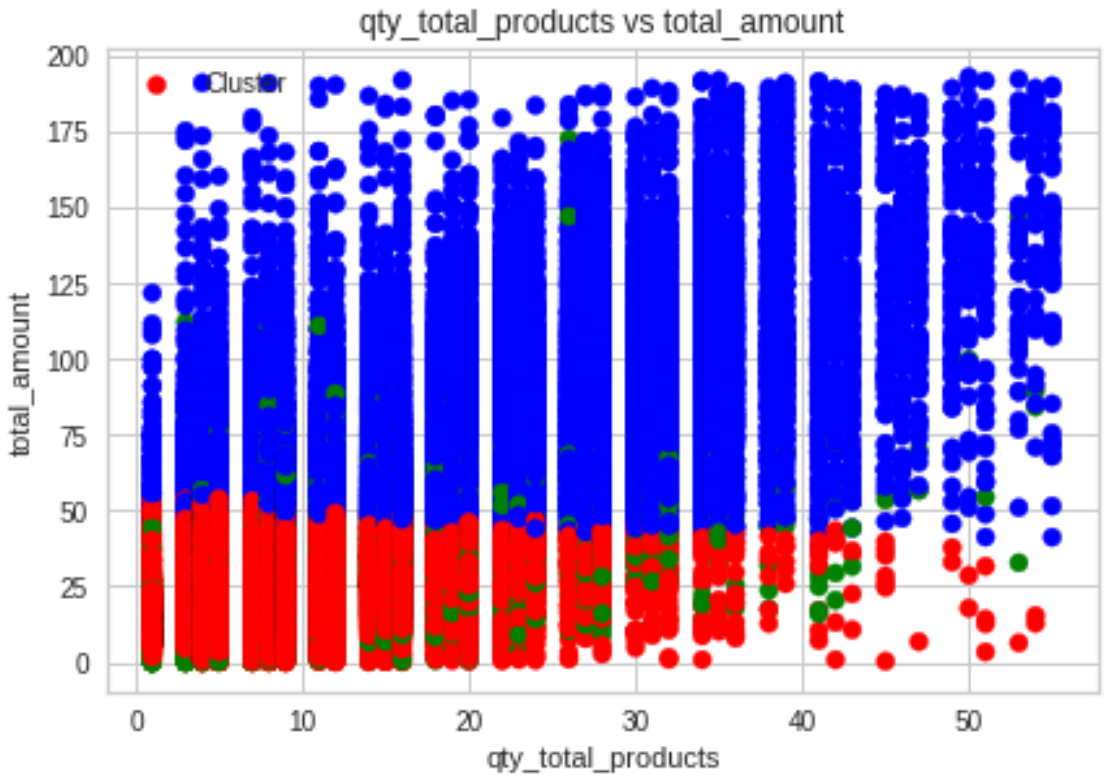
[108]:
```
# Construimos un DataFrame con los promedios de cada variable por cluster y el␣
↪promedio de todo el dataset
final = pd.DataFrame({'variables': numeric_df.columns, 'cluster0': cluster0,␣
↪'cluster1': cluster1,
                      'cluster2': cluster2 ,
                      'promedio': promedio}).reset_index(drop=True)

final
```

[108]:

	variables	cluster0	cluster1	cluster2	promedio
0	qty_total_products	6.49	19.62	7.65	8.26
1	total_amount	20.00	75.86	23.49	27.33
2	order_count_usu	15.06	20.88	85.34	26.02
3	BWS_qty	0.31	0.54	0.38	0.35
4	Dairy / Chilled / Eggs_qty	0.97	3.80	1.22	1.35
5	Home / Pet_qty	0.43	2.00	0.65	0.65
6	Meat / Seafood_qty	0.10	0.63	0.22	0.18
7	Produce_qty	0.47	1.75	0.72	0.66
8	Beverages_qty	0.99	1.75	1.11	1.10
9	Snacks_qty	1.32	2.52	1.12	1.44
10	Frozen_qty	0.30	0.85	0.28	0.36

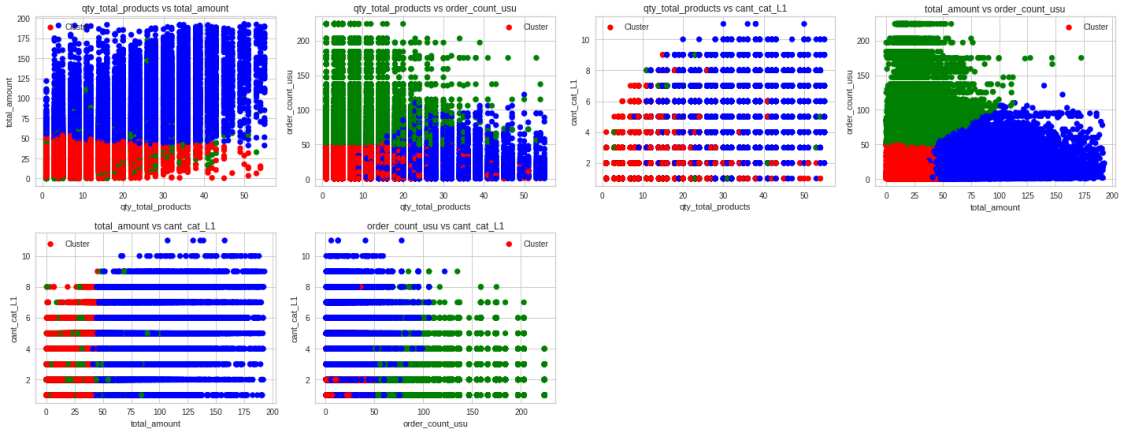
11	Packaged Foods_qty	0.91	4.01	1.13	1.32
12	Smoking / Tobacco_qty	0.25	0.09	0.34	0.24
13	Personal Care / Baby / Health_qty	0.23	0.95	0.23	0.32
14	Bread / Bakery_qty	0.17	0.63	0.21	0.24
15	General Merchandise_qty	0.01	0.03	0.01	0.01
16	Ready To Consume_qty	0.02	0.06	0.02	0.02
17	cant_cat_L1	2.22	4.35	2.52	2.52
18	peso_BWS	1.16	2.49	1.29	1.34
19	peso_Beverages	2.77	6.10	3.00	3.21
20	peso_Bread / Bakery	0.56	2.43	0.66	0.80
21	peso_Dairy / Chilled / Eggs	2.80	13.89	3.63	4.27
22	peso_Frozen	1.20	3.66	1.05	1.48
23	peso_General Merchandise	0.05	0.14	0.04	0.06
24	peso_Home / Pet	1.55	8.24	2.23	2.47
25	peso_Meat / Seafood	0.68	4.17	1.42	1.22
26	peso_Packaged Foods	2.64	14.73	3.26	4.20
27	peso_Personal Care / Baby / Health	0.92	4.15	0.84	1.30
28	peso_Produce	1.20	6.39	1.85	1.93
29	peso_Ready To Consume	0.07	0.28	0.08	0.09
30	peso_Smoking / Tobacco	1.09	0.52	1.38	1.06
31	peso_Snacks	3.30	8.67	2.76	3.88

[109]:
```
# Definir una paleta de colores personalizada
colors = ['red', 'blue', 'green', 'orange', 'purple', 'yellow', 'cyan',␣
 ↪'magenta', 'lime', 'pink', 'teal', 'brown', 'gray', 'olive', 'navy',␣
 ↪'salmon']


# Graficar
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.scatter(numeric_df['qty_total_products'], numeric_df['total_amount'],␣
 ↪c=[colors[label] for label in cluster_labels])
plt.title('qty_total_products vs total_amount')
plt.xlabel('qty_total_products')
plt.ylabel('total_amount')
plt.legend(['Cluster'])

plt.show()
```



[110]:
```
# Definir una paleta de colores personalizada
colors = ['red', 'blue', 'green', 'orange', 'purple', 'yellow', 'cyan',␣
 ↪'magenta', 'lime', 'pink', 'teal', 'brown', 'gray', 'olive', 'navy',␣
 ↪'salmon']

# Pares de variables a graficar
variable_pairs = [
    ('qty_total_products', 'total_amount'),
    ('qty_total_products', 'order_count_usu'),
    ('qty_total_products', 'cant_cat_L1'),
    ('total_amount', 'order_count_usu'),
    ('total_amount', 'cant_cat_L1'),
    ('order_count_usu', 'cant_cat_L1'),


]

# Graficar cada par de variables
plt.figure(figsize=(20, 15))

for i, (var1, var2) in enumerate(variable_pairs):
    plt.subplot(4, 4, i + 1)
```

```
    plt.scatter(numeric_df[var1], numeric_df[var2], c=[colors[label] for label␣
 ↪in cluster_labels])
    plt.title(f'{var1} vs {var2}')
    plt.xlabel(var1)
    plt.ylabel(var2)
    plt.legend(['Cluster'])

plt.tight_layout()
plt.show()
```



[111]:
```
df_otras
```

[111]:

	order_id	weekday	\
0	MtfRP1Q+Xx1Ox1JYEexxu7x9EOq7KfccMf+PdfY64n4=	Thursday	
1	GfUKK9Bh5wBGg+XJuy1E3yYr9kQ4GH+X1EVLe6YbzTg=	Tuesday	
2	KH4S+lzxCMKHd6UUOiIo7DuLlDaufLd3raqSHBeibAg=	Wednesday	
3	GdvT/R8JqkIOzSzfwVrVag6okmBzOOms88KxgAseht8=	Saturday	
4	jHBgND7nb06oC1b6Ky6aqdqR0bWIKvaBYKMnNrrzaaQ=	Tuesday	
…	…	…	
860979	Ppsk/1iIoA++xYkWnLd0pAz9xnGqy/pStEmwX7hBfSY=	Wednesday	
861007	aFJuTDgA5K4j8CGuZIvhcPL2FEs6Umsj9mMO1eGr6Vo=	Sunday	
861037	MF2dm3i2fs87y04GYfIJ60pZJCwQ5nn02labBd03B+k=	Saturday	
861062	xHcIBn68bgDJbVcgOGxrnza6eiRl2TDhfbTjXLz2zGQ=	Friday	
861092	6MQ1m73q6Mx8k+qhMpF2Zh/C7CJi8cDNLoAfNpBSIFY=	Tuesday	

	user_id	has_discount	\
0	SwEO/imkcz2Ws75JXdlGR117JNK3BiFLkwMkrJLbZbo=	False	
1	Sww2v+mY8dtWiMEtMPIEQLAkitFKxKZOn+84SPBeEG4=	False	
2	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	False	
3	Sx+OBqNkMZTtqkk7VUjbNa6brwNBsUzbjhLKFMz8yvM=	False	
4	Sx+s37QMSYEPCq+aeUFvWfJW87qJUZIEFmM4iqy5MXo=	False	
…	…	…	

```
860979  Ob2zUnYtV93bePpU43mfNRfyT3TOvZS/0xItVGz1rj8=         True
861007  Qo7XOwW3phLImqmhdtTfbeWqJtnd6M+Vwfdcx8bBPT4=         True
861037  SOCb9TupUCxJ8GGEz5IYMKmnHJfaNNXGmSPpI2pAuHA=         True
861062  Bode5766jynMCiSW9lhAiNaGRNfGscGezXnecnLHM5Q=         True
861092  D9mUJIqcA3+ZKJKMRZ9gHW5a7tUaHIk8sWy1a7ZidLE=         True
```

```
                level_one frec_compra_usu es_fin_de_semana periodo_dia
0       Smoking / Tobacco            Baja            False  Vespertino
1              Beverages            Baja            False    Nocturno
2              Beverages           Media            False    Nocturno
3       Smoking / Tobacco           Media             True    Nocturno
4          Packaged Foods            Baja            False  Vespertino
…                     …               …                …           …
860979            Produce            Alta            False    Matutino
861007            Produce            Alta             True    Nocturno
861037     Packaged Foods           Media             True  Vespertino
861062            Beverages            Alta            False    Matutino
861092     Packaged Foods            Alta            False    Nocturno

[196456 rows x 8 columns]
```

[112]:
```
# Miramos la distribución de las observaciones clusterizadas
cluster_labels = kmeans.labels_
print(cluster_labels)
```

```
[0 0 0 … 1 1 1]
```

[113]:
```
# Asegurémonos de que cluster_labels sea una serie de pandas
cluster_labels = pd.Series(cluster_labels)
```

[114]:
```
# Verifiquemos la dimensión de cluster_labels (debería ser 1)
print("Dimensiones de cluster_labels:", cluster_labels.ndim)
```

```
Dimensiones de cluster_labels: 1
```

[115]:
```
# Verifiquemos las primeras filas para asegurarnos de que los datos son␣
↪correctos
cluster_labels
```

[115]:
```
0         0
1         0
2         0
3         0
4         0
         ..
196451    1
196452    1
```

```
196453    1
196454    1
196455    1
Length: 196456, dtype: int32
```

[116]:
```
# Definimos las columnas categóricas que queremos analizar
columnas_categoricas = ['weekday', 'has_discount', 'frec_compra_usu',␣
↪'es_fin_de_semana', 'periodo_dia']

# Filtramos el DataFrame original para incluir solo las columnas categóricas
df_categoricas = df_otras[columnas_categoricas].copy()

# Añadimos cluster_labels como una columna adicional al DataFrame df_categoricas
df_categoricas['cluster_label'] = cluster_labels.values
```

[117]:
```
# Calculamos la tabla de frecuencias cruzadas entre las columnas categóricas y␣
↪cluster_label
cluster_class_freq = pd.crosstab(index=df_categoricas['cluster_label'],␣
↪columns=df_categoricas['weekday'])

# Imprimimos el resultado
cluster_class_freq
```

[117]:

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
cluster_label							
0	20616	19882	21813	21827	19196	20797	19697
1	3462	3333	3836	3076	3418	3597	3267
2	4217	4069	4052	3683	4112	4304	4202

[118]:
```
# Calculamos la tabla de frecuencias cruzadas entre las columnas categóricas y␣
↪cluster_label
cluster_class_freq = pd.crosstab(index=df_categoricas['cluster_label'],␣
↪columns=df_categoricas['has_discount'])

# Imprimimos el resultado
cluster_class_freq
```

[118]:

has_discount	False	True
cluster_label		
0	58519	85309
1	4554	19435
2	7477	21162

[119]:
```
# Calculamos la tabla de frecuencias cruzadas entre las columnas categóricas y␣
↪cluster_label
cluster_class_freq = pd.crosstab(index=df_categoricas['cluster_label'],␣
↪columns=df_categoricas['frec_compra_usu'])
```

```
# Imprimimos el resultado
cluster_class_freq
```

[119]:
frec_compra_usu	Alta	Baja	Media
cluster_label			
0	88283	32109	23436
1	18188	2571	3230
2	28639	0	0

[120]:
```
# Calculamos la tabla de frecuencias cruzadas entre las columnas categóricas y␣
↪cluster_label
cluster_class_freq = pd.crosstab(index=df_categoricas['cluster_label'],␣
↪columns=df_categoricas['es_fin_de_semana'])

# Imprimimos el resultado
cluster_class_freq
```

[120]:
es_fin_de_semana	False	True
cluster_label		
0	100188	43640
1	17077	6912
2	20904	7735

[121]:
```
# Calculamos la tabla de frecuencias cruzadas entre las columnas categóricas y␣
↪cluster_label
cluster_class_freq = pd.crosstab(index=df_categoricas['cluster_label'],␣
↪columns=df_categoricas['periodo_dia'])

# Imprimimos el resultado
cluster_class_freq
```

[121]:
periodo_dia	Matutino	Nocturno	Vespertino
cluster_label			
0	20688	71283	51857
1	4676	8944	10369
2	5667	12216	10756

6 K-medoids clustering (PAM)

[173]:
```
k_cluster = 3
```

[174]:
```
df_muestra = df.sample(n=10000, random_state=semilla)
```

[175]:
```
columns_to_drop = [
    'BWS',
    'Beverages',
    'Bread / Bakery',
    'Dairy / Chilled / Eggs',
    'Frozen',
    'General Merchandise',
    'Home / Pet',
    'Meat / Seafood',
    'Packaged Foods',
    'Personal Care / Baby / Health',
    'Produce',
    'Ready To Consume',
    'Smoking / Tobacco',
    'Snacks'
]

df_muestra = df_muestra.drop(columns=columns_to_drop)

df_muestra.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 370074 to 52364
Data columns (total 41 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   order_id                        10000 non-null  object
 1   weekday                         10000 non-null  category
 2   hour                            10000 non-null  int64
 3   user_id                         10000 non-null  object
 4   qty_total_products              10000 non-null  int64
 5   total_amount                    10000 non-null  float64
 6   has_discount                    10000 non-null  category
 7   level_one                       10000 non-null  object
 8   order_count_usu                 10000 non-null  int64
 9   frec_compra_usu                 10000 non-null  category
 10  es_fin_de_semana                10000 non-null  category
 11  periodo_dia                     10000 non-null  category
 12  BWS_qty                         10000 non-null  float64
 13  Dairy / Chilled / Eggs_qty      10000 non-null  float64
 14  Home / Pet_qty                  10000 non-null  float64
 15  Meat / Seafood_qty              10000 non-null  float64
 16  Produce_qty                     10000 non-null  float64
 17  Beverages_qty                   10000 non-null  float64
 18  Snacks_qty                      10000 non-null  float64
 19  Frozen_qty                      10000 non-null  float64
 20  Packaged Foods_qty              10000 non-null  float64
```

```
 21  Smoking / Tobacco_qty                10000 non-null  float64
 22  Personal Care / Baby / Health_qty    10000 non-null  float64
 23  Bread / Bakery_qty                   10000 non-null  float64
 24  General Merchandise_qty              10000 non-null  float64
 25  Ready To Consume_qty                 10000 non-null  float64
 26  cant_cat_L1                          10000 non-null  int64
 27  peso_BWS                             10000 non-null  float64
 28  peso_Beverages                       10000 non-null  float64
 29  peso_Bread / Bakery                  10000 non-null  float64
 30  peso_Dairy / Chilled / Eggs          10000 non-null  float64
 31  peso_Frozen                          10000 non-null  float64
 32  peso_General Merchandise             10000 non-null  float64
 33  peso_Home / Pet                      10000 non-null  float64
 34  peso_Meat / Seafood                  10000 non-null  float64
 35  peso_Packaged Foods                  10000 non-null  float64
 36  peso_Personal Care / Baby / Health   10000 non-null  float64
 37  peso_Produce                         10000 non-null  float64
 38  peso_Ready To Consume                10000 non-null  float64
 39  peso_Smoking / Tobacco               10000 non-null  float64
 40  peso_Snacks                          10000 non-null  float64
dtypes: category(5), float64(29), int64(4), object(3)
memory usage: 2.9+ MB
```

[176]:
```
df_gower = df_muestra.copy()
```

[177]:
```
df_gower.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 370074 to 52364
Data columns (total 41 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   order_id                             10000 non-null  object
 1   weekday                              10000 non-null  category
 2   hour                                 10000 non-null  int64
 3   user_id                              10000 non-null  object
 4   qty_total_products                   10000 non-null  int64
 5   total_amount                         10000 non-null  float64
 6   has_discount                         10000 non-null  category
 7   level_one                            10000 non-null  object
 8   order_count_usu                      10000 non-null  int64
 9   frec_compra_usu                      10000 non-null  category
 10  es_fin_de_semana                     10000 non-null  category
 11  periodo_dia                          10000 non-null  category
 12  BWS_qty                              10000 non-null  float64
 13  Dairy / Chilled / Eggs_qty           10000 non-null  float64
 14  Home / Pet_qty                       10000 non-null  float64
 15  Meat / Seafood_qty                   10000 non-null  float64
```

```
 16  Produce_qty                          10000 non-null  float64
 17  Beverages_qty                        10000 non-null  float64
 18  Snacks_qty                           10000 non-null  float64
 19  Frozen_qty                           10000 non-null  float64
 20  Packaged Foods_qty                   10000 non-null  float64
 21  Smoking / Tobacco_qty                10000 non-null  float64
 22  Personal Care / Baby / Health_qty    10000 non-null  float64
 23  Bread / Bakery_qty                   10000 non-null  float64
 24  General Merchandise_qty              10000 non-null  float64
 25  Ready To Consume_qty                 10000 non-null  float64
 26  cant_cat_L1                          10000 non-null  int64
 27  peso_BWS                             10000 non-null  float64
 28  peso_Beverages                       10000 non-null  float64
 29  peso_Bread / Bakery                  10000 non-null  float64
 30  peso_Dairy / Chilled / Eggs          10000 non-null  float64
 31  peso_Frozen                          10000 non-null  float64
 32  peso_General Merchandise             10000 non-null  float64
 33  peso_Home / Pet                      10000 non-null  float64
 34  peso_Meat / Seafood                  10000 non-null  float64
 35  peso_Packaged Foods                  10000 non-null  float64
 36  peso_Personal Care / Baby / Health   10000 non-null  float64
 37  peso_Produce                         10000 non-null  float64
 38  peso_Ready To Consume                10000 non-null  float64
 39  peso_Smoking / Tobacco               10000 non-null  float64
 40  peso_Snacks                          10000 non-null  float64
dtypes: category(5), float64(29), int64(4), object(3)
memory usage: 2.9+ MB
```

[178]:
```
df_gower = df_gower.drop(['order_id','user_id','level_one'], axis=1)
```

[179]:
```
df_gower_2 = df_gower.copy()
```

[180]:
```
# Convierte las columnas categóricas a tipo string
for col in df_gower_2.select_dtypes(['category']).columns:
    df_gower_2.loc[:, col] = df_gower_2[col].astype(str)
```

[181]:
```
df_gower_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 370074 to 52364
Data columns (total 38 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   weekday                              10000 non-null  object
 1   hour                                 10000 non-null  int64
 2   qty_total_products                   10000 non-null  int64
 3   total_amount                         10000 non-null  float64
 4   has_discount                         10000 non-null  object
```

```
 5   order_count_usu                      10000 non-null  int64
 6   frec_compra_usu                      10000 non-null  object
 7   es_fin_de_semana                     10000 non-null  object
 8   periodo_dia                          10000 non-null  object
 9   BWS_qty                              10000 non-null  float64
 10  Dairy / Chilled / Eggs_qty           10000 non-null  float64
 11  Home / Pet_qty                       10000 non-null  float64
 12  Meat / Seafood_qty                   10000 non-null  float64
 13  Produce_qty                          10000 non-null  float64
 14  Beverages_qty                        10000 non-null  float64
 15  Snacks_qty                           10000 non-null  float64
 16  Frozen_qty                           10000 non-null  float64
 17  Packaged Foods_qty                   10000 non-null  float64
 18  Smoking / Tobacco_qty                10000 non-null  float64
 19  Personal Care / Baby / Health_qty    10000 non-null  float64
 20  Bread / Bakery_qty                   10000 non-null  float64
 21  General Merchandise_qty              10000 non-null  float64
 22  Ready To Consume_qty                 10000 non-null  float64
 23  cant_cat_L1                          10000 non-null  int64
 24  peso_BWS                             10000 non-null  float64
 25  peso_Beverages                       10000 non-null  float64
 26  peso_Bread / Bakery                  10000 non-null  float64
 27  peso_Dairy / Chilled / Eggs          10000 non-null  float64
 28  peso_Frozen                          10000 non-null  float64
 29  peso_General Merchandise             10000 non-null  float64
 30  peso_Home / Pet                      10000 non-null  float64
 31  peso_Meat / Seafood                  10000 non-null  float64
 32  peso_Packaged Foods                  10000 non-null  float64
 33  peso_Personal Care / Baby / Health   10000 non-null  float64
 34  peso_Produce                         10000 non-null  float64
 35  peso_Ready To Consume                10000 non-null  float64
 36  peso_Smoking / Tobacco               10000 non-null  float64
 37  peso_Snacks                          10000 non-null  float64
dtypes: float64(29), int64(4), object(5)
memory usage: 3.0+ MB
```

[182]:
```
# Seleccionar las columnas numéricas
numerical_cols = df_gower_2.select_dtypes(include=['float64', 'int64']).columns

# Escalar las columnas numéricas usando scale de sklearn
df_gower_2[numerical_cols] = scale(df_gower_2[numerical_cols])
```

[183]:
```
df_gower_2.shape
```

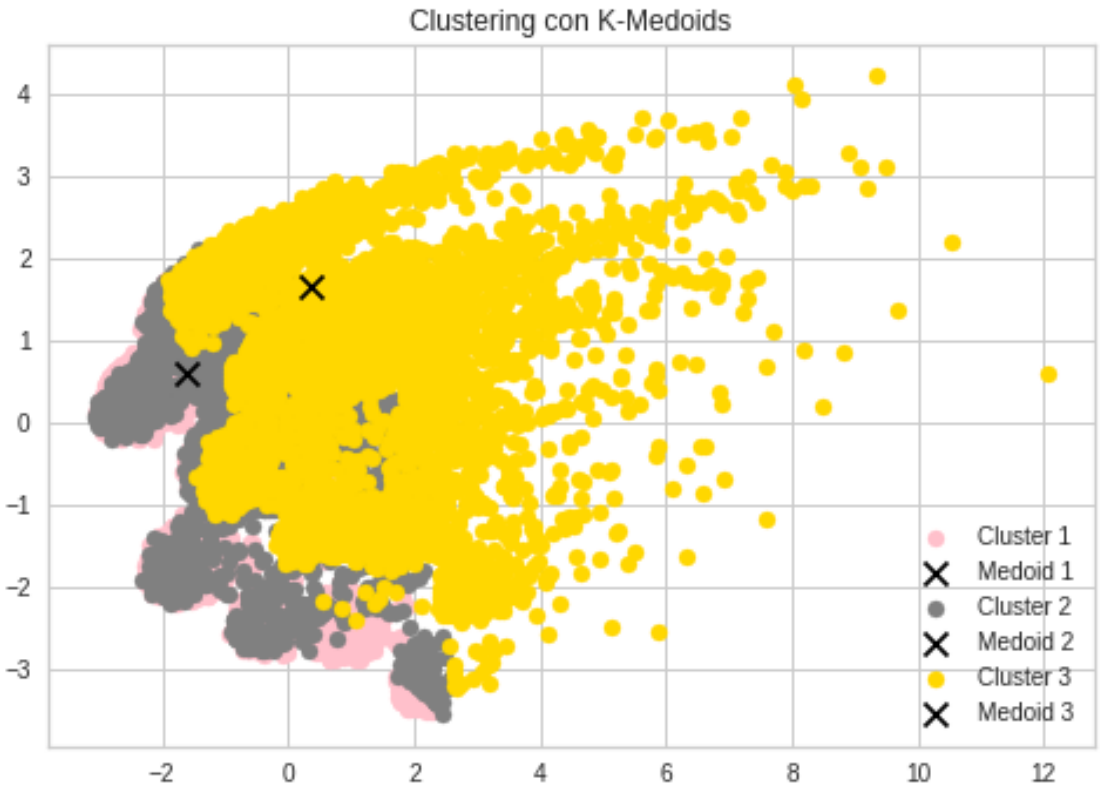
[183]:
```
(10000, 38)
```

[184]:
```
dist_gower = gower.gower_matrix(df_gower_2)
```

[185]:
```
# Configurar el algoritmo K-medioids
kmedoids = KMedoids(n_clusters=k_cluster, random_state=semilla)

# Ajustar el modelo K-medioids a los datos
km = kmedoids.fit(dist_gower)

# Obtener los labels (clusters asignados a cada punto)
labels = km.labels_

# Obtener los centroides
centroids = km.medoid_indices_

# Resultados
print("Cluster labels:", labels)

# Resultados
print("Cluster centroids:", centroids)
```

```
Cluster labels: [1 0 2 … 0 0 2]
Cluster centroids: [1824 4111 5282]
```

[198]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
X_pca = pca.fit_transform(dist_gower)

colors = ['pink', 'gray', 'gold', 'olive', 'teal']
for i in range(k_cluster):
    cluster_points = X_pca[labels == i]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], color=colors[i],␣
 ↪label=f'Cluster {i+1}')
    plt.scatter(X_pca[centroids[i], 0], X_pca[centroids[i], 1], marker='x',␣
 ↪color='black', s=100, label=f'Medoid {i+1}')

plt.title('Clustering con K-Medoids')
plt.legend()
plt.grid(True)
plt.show()
```


Clustering con K-Medoids
Cluster 1
Medoid 1
Cluster 2
Medoid 2
Cluster 3
Medoid 3