

Instituto Superior Politécnico Córdoba

Tecnicatura Superior en Innovación con Tecnologías 4.0

Analítica de Datos

Trabajo Práctico Final

Cohorte 2023

Profesor: Alejandro Mainero

Alumnos: Cristian Ghisiglieri

Naylui Osuna

Romina Peña

Correos: cristian.g2011@gmail.com

nayluiosuna@gmail.com

rominabpena@gmail.com

Tabla de Contenido

Introducción.....	3
1. Proceso de ETL.....	4
1.1. Extracción de Datos (Extract).....	4
1.2. Transformación de Datos (Transform).....	8
1.3. Limpieza de Datos (Load).....	16
2. Proceso de Análisis de Datos.....	18
1. Consultas.....	18
2. Análisis de Tendencias y Comportamientos.....	33
2.1. Perfil Demográfico y Comportamiento del Consumidor.....	33
2.2. Patrones de Consumo y Preferencias de Categoría.....	34
2.3. Métodos de pago y Transformación Digital.....	35
Conclusión.....	38
Bibliografía.....	40

Introducción

En el presente análisis nos sumergimos en el mundo del comportamiento del consumidor, utilizando un conjunto de datos detallado obtenido de Kaggle que comprende dos archivos complementarios: "sales_data.csv" y "customer_data.csv". A través de un riguroso proceso de ETL (Extracción, Transformación y Carga), hemos procesado más de 99,000 transacciones comerciales para descubrir patrones significativos en las compras minoristas. Los datos nos ofrecen una visión integral que combina información transaccional detallada, como números de factura, categorías de productos y ubicaciones de compra, con valiosos datos demográficos de los clientes, incluyendo género, edad y preferencias de pago. Esta combinación detallada de información nos permite profundizar en las tendencias de consumo y extraer insights significativos sobre cómo diferentes grupos de consumidores toman sus decisiones de compra.

1. Proceso de ETL

1.1. Extracción de Datos (Extract)

En este proyecto, realizaremos un proceso ETL (Extract, Transform, Load) para analizar el dataset "Sales and Customer Data" disponible en Kaggle.

Primero, extraemos los datos de los archivos "customer_data.csv" y "sales_data.csv".

A continuación, detallaremos los pasos a seguir:

1. Importamos las librerías necesarias. En este caso, utilizamos Pandas.

Código: `import pandas as pd`

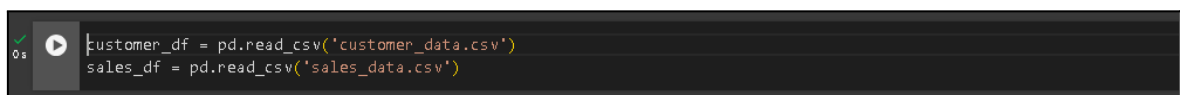
Figura 1. Importar Pandas



2. Luego, cargamos los datos de ventas y clientes desde el archivo CSV en dos DataFrames distintos.

Código: `customer_df = pd.read_csv('customer_data.csv')`
`sales_df = pd.read_csv('sales_data.csv')`

Figura 2. Carga de csv



El proceso de extracción implica la carga de datos desde archivos CSV a DataFrames de Pandas, que son estructuras de datos tabulares que facilitan la manipulación y análisis de datos. A través de las sentencias ejecutadas con

Pandas, se cargaron y prepararon los datos para visualizarlos en dos

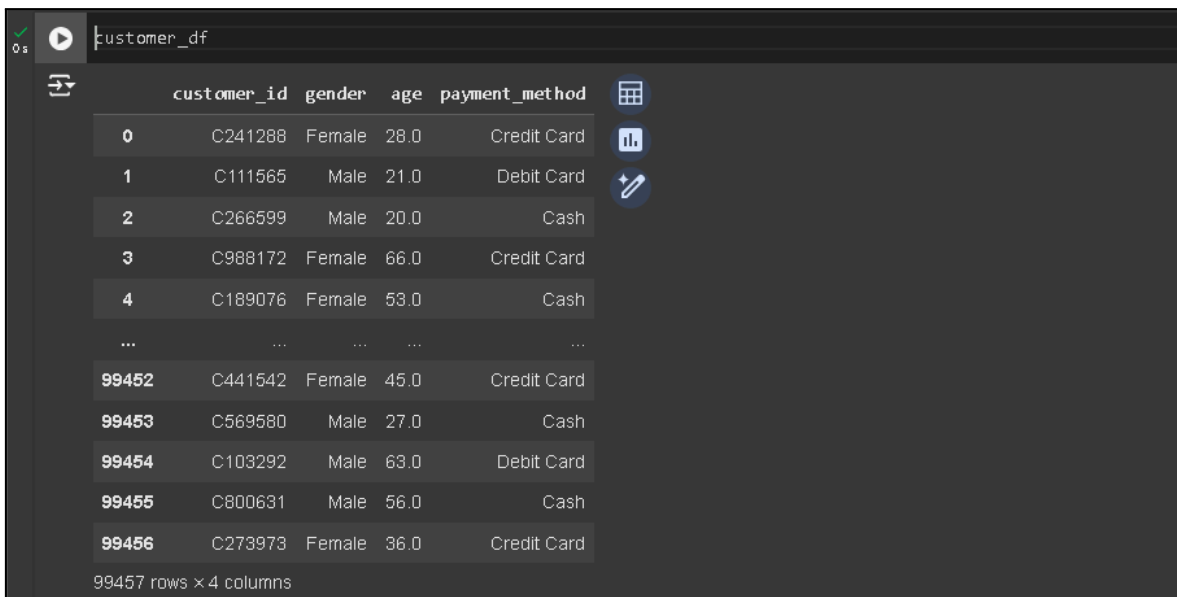
DataFrames: “customer_df” y “sales_df”.

Una vez cargados los datos en los DataFrames, se pueden explorar y manipular de diversas maneras.

Al colocar el nombre del DataFrame y ejecutarlo, se pueden visualizar los datos que contiene el mismo.

Código: customer_df

Figura 3. Visualización de contenido de DF Customer




	customer_id	gender	age	payment_method
0	C241288	Female	28.0	Credit Card
1	C111565	Male	21.0	Debit Card
2	C266599	Male	20.0	Cash
3	C988172	Female	66.0	Credit Card
4	C189076	Female	53.0	Cash
...
99452	C441542	Female	45.0	Credit Card
99453	C569580	Male	27.0	Cash
99454	C103292	Male	63.0	Debit Card
99455	C800631	Male	56.0	Cash
99456	C273973	Female	36.0	Credit Card

99457 rows x 4 columns

Al solicitar ver el DataFrame `customer_df`, además de poder visualizar los datos, obtenemos referencias importantes, como la cantidad de filas (99.457) y la cantidad de columnas (4).

Código: sales_df

Figura 4. Visualización de contenido de DF Sales



	invoice_no	customer_id	category	quantity	price	invoice_date	shopping_mall
0	I138884	C241288	Clothing	5	1500.40	05-08-2022	Kanyon
1	I317333	C111565	Shoes	3	1800.51	12-12-2021	Forum Istanbul
2	I127801	C266599	Clothing	1	300.08	09-11-2021	Metrocity
3	I173702	C988172	Shoes	5	3000.85	16-05-2021	Metropol AVM
4	I337046	C189076	Books	4	60.60	24-10-2021	Kanyon
...
99452	I219422	C441542	Souvenir	5	58.65	21-09-2022	Kanyon
99453	I325143	C569580	Food & Beverage	2	10.46	22-09-2021	Forum Istanbul
99454	I824010	C103292	Food & Beverage	2	10.46	28-03-2021	Metrocity
99455	I702964	C800631	Technology	4	4200.00	16-03-2021	Istinye Park
99456	I232867	C273973	Souvenir	3	35.19	15-10-2022	Mall of Istanbul

99457 rows x 7 columns

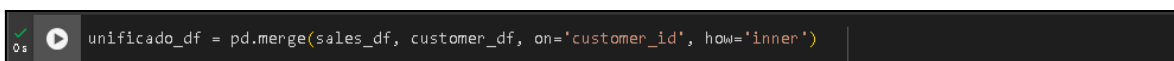
Al igual que con el DataFrame `customer_df`, podemos visualizar los datos, la cantidad de filas (99.457) y la cantidad de columnas (7) que tiene el DataFrame `sales_df`.

Además, utilizando los métodos `describe()`, `isnull()` e `info()`, es posible obtener una visión más detallada y completa de la información general contenida en los DataFrames.

- Concatenamos los dos DataFrames anteriores en uno final con información relevante.

Código: `unificado_df = pd.merge(sales_df, customer_df, on='customer_id', how='inner')`

Figura 5. Unificación de DF



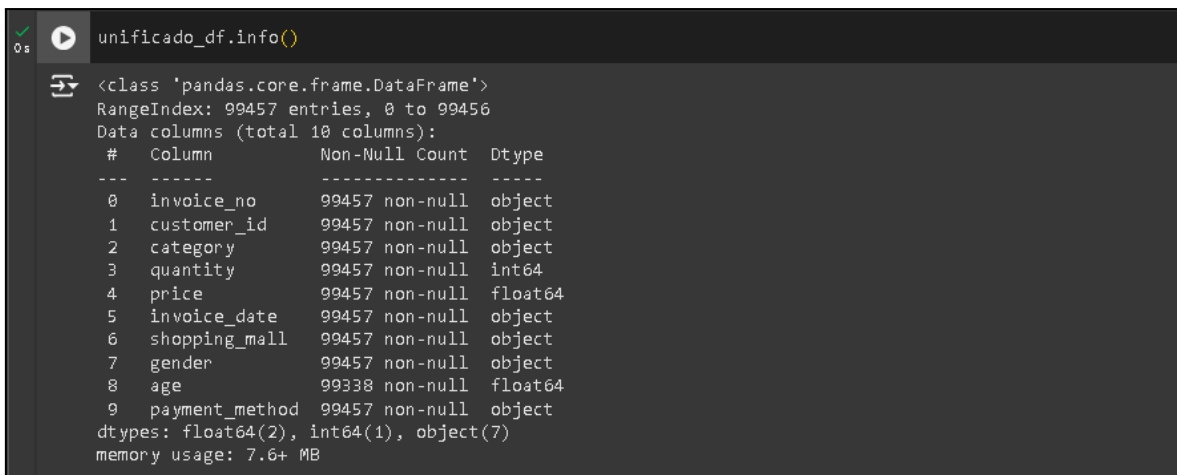
```
unificado_df = pd.merge(sales_df, customer_df, on='customer_id', how='inner')
```

Para evitar trabajar con dos archivos separados, unificamos los datos en un solo DataFrame (tabla) para facilitar su visualización y análisis. La concatenación de DataFrames se puede hacer de varias maneras dependiendo de cómo se relacionan los datos. En este punto, asumimos que `customer_id` es la clave común para unir los DataFrames de ventas y clientes.

4. Verificamos la fusión.

Código: `unificado_df.info()`

Figura 6. `unificado_df.info()`



```
unificado_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99457 entries, 0 to 99456
Data columns (total 10 columns):
#   Column             Non-Null Count  Dtype  
---  --
0   invoice_no         99457 non-null  object 
1   customer_id        99457 non-null  object 
2   category           99457 non-null  object 
3   quantity           99457 non-null  int64  
4   price              99457 non-null  float64 
5   invoice_date       99457 non-null  object 
6   shopping_mall      99457 non-null  object 
7   gender             99457 non-null  object 
8   age                99338 non-null  float64 
9   payment_method     99457 non-null  object 
dtypes: float64(2), int64(1), object(7)
memory usage: 7.6+ MB
```

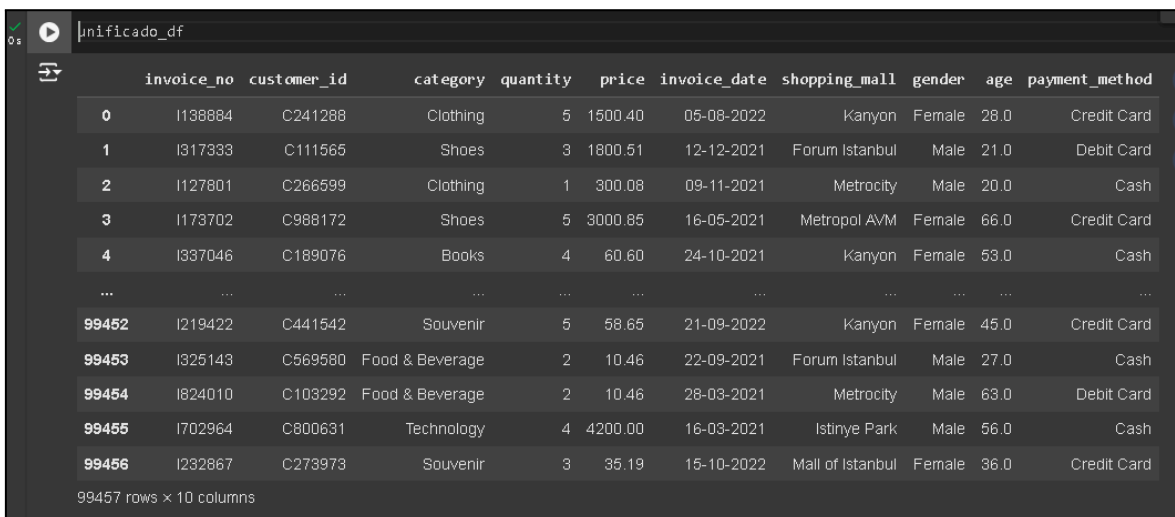
Corroboramos la fusión de los dos archivos .csv verificando la información sobre las columnas y los tipos de datos que contiene el DataFrame resultante utilizando el método `info()`. Observamos que la cantidad de filas se mantiene en 99.457, pero el número de columnas aumenta a 10. Esto se debe a que el archivo `sales_data.csv` aporta 7 columnas, mientras que el archivo `customer_data.csv` agrega solo 3, ya que la columna ID es común en ambos DataFrames. Asimismo,

identificamos los tipos de datos de cada columna, que incluyen float, int64 y object.

5. Visualizamos el nuevo Data Frame.

Código: `unificado_df`

Figura 7. Visualización de `unificado_df`



	invoice_no	customer_id	category	quantity	price	invoice_date	shopping_mall	gender	age	payment_method
0	I138884	C241288	Clothing	5	1500.40	05-08-2022	Kanyon	Female	28.0	Credit Card
1	I317333	C111565	Shoes	3	1800.51	12-12-2021	Forum Istanbul	Male	21.0	Debit Card
2	I127801	C266599	Clothing	1	300.08	09-11-2021	Metrocity	Male	20.0	Cash
3	I173702	C988172	Shoes	5	3000.85	16-05-2021	Metropol AVM	Female	66.0	Credit Card
4	I337046	C189076	Books	4	60.60	24-10-2021	Kanyon	Female	53.0	Cash
...
99452	I219422	C441542	Souvenir	5	58.65	21-09-2022	Kanyon	Female	45.0	Credit Card
99453	I325143	C569580	Food & Beverage	2	10.46	22-09-2021	Forum Istanbul	Male	27.0	Cash
99454	I824010	C103292	Food & Beverage	2	10.46	28-03-2021	Metrocity	Male	63.0	Debit Card
99455	I702964	C800631	Technology	4	4200.00	16-03-2021	Istinye Park	Male	56.0	Cash
99456	I232867	C273973	Souvenir	3	35.19	15-10-2022	Mall of Istanbul	Female	36.0	Credit Card

99457 rows x 10 columns

Al examinar el DataFrame resultante, observamos los datos correspondientes a las 10 columnas.

1.2. Transformación de Datos (Transform)

A continuación, daremos inicio a la fase de Transformación dentro del proceso ETL (Extract, Transform, Load). En esta etapa, limpiaremos, normalizaremos y estructuraremos los datos obtenidos previamente para mejorar su calidad y coherencia. Aplicaremos diversas técnicas, como el manejo de

valores nulos, la normalización de datos y la creación de nuevas variables derivadas, con el objetivo de preparar la información para su posterior análisis y carga en la base de datos destino. Este proceso es fundamental para garantizar que los datos sean precisos, completos y útiles para generar insights valiosos.

Con anterioridad, realizamos un análisis preliminar del dataframe mediante la función `unificado_df.info()` para visualizar la información de las columnas y sus tipos de datos. Notamos que en la columna "age" la cantidad de datos era menor al número total de filas del DataFrame. Ante esta situación, teníamos varias opciones:

- A. Eliminar filas con datos vacíos: Esta opción podría afectar futuros análisis, pero dado que los valores nulos sólo representaban el 1,19% del total, la eliminación de estas filas no comprometería la representatividad del dataset.
- B. Rellenar valores nulos con el promedio: Utilizar el promedio de la columna "age" para reemplazar los valores nulos puede ser adecuado en algunos casos, especialmente si los datos faltantes están distribuidos de manera uniforme.
- C. Rellenar valores nulos con la mediana: La mediana puede ser una mejor opción que el promedio si la distribución de "age" está sesgada, ya que no se verá influenciada por valores extremos.
- D. Rellenar valores nulos con un valor fijo: Dependiendo del contexto, puede ser útil reemplazar los valores nulos con un valor fijo que tenga sentido para el análisis específico.

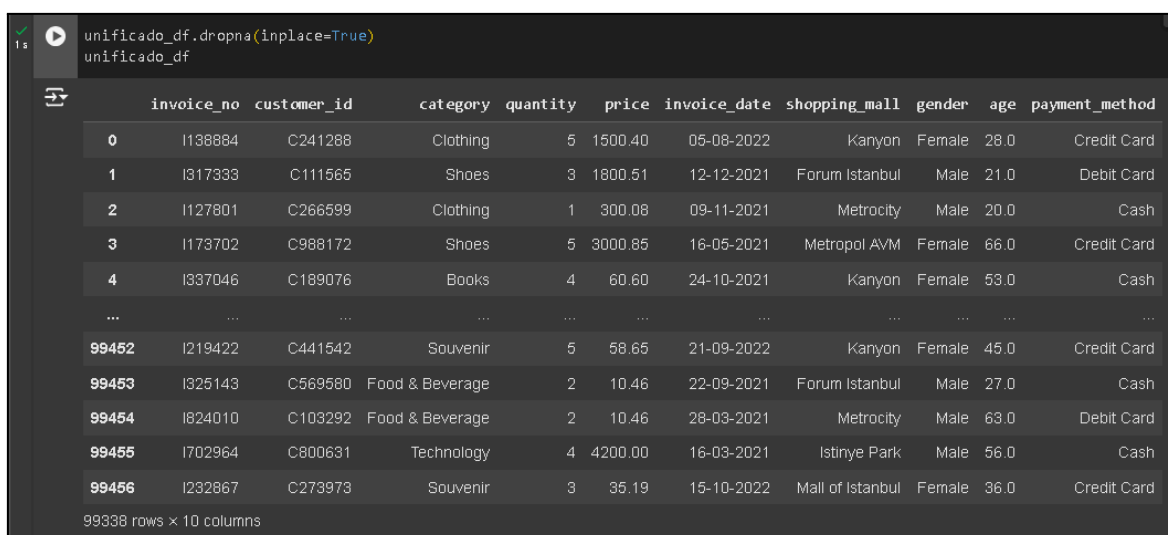
Optamos por eliminar las filas incompletas debido a que la proporción de datos faltantes era mínima (1,19%). Esta cantidad insignificante de filas eliminadas no afecta significativamente la calidad y representatividad del dataset, manteniendo la integridad de nuestro análisis.

1. Eliminamos las Filas con Valores Nulos

Código: `unificado_df.dropna(inplace=True)`

`unificado_df`

Figura 8. *Eliminación de valores nulos*



```
unificado_df.dropna(inplace=True)
unificado_df
```

	invoice_no	customer_id	category	quantity	price	invoice_date	shopping_mall	gender	age	payment_method
0	I138884	C241288	Clothing	5	1500.40	05-08-2022	Kanyon	Female	28.0	Credit Card
1	I317333	C111565	Shoes	3	1800.51	12-12-2021	Forum Istanbul	Male	21.0	Debit Card
2	I127801	C266599	Clothing	1	300.08	09-11-2021	Metrocity	Male	20.0	Cash
3	I173702	C988172	Shoes	5	3000.85	16-05-2021	Metropol AVM	Female	66.0	Credit Card
4	I337046	C189076	Books	4	60.60	24-10-2021	Kanyon	Female	53.0	Cash
...
99452	I219422	C441542	Souvenir	5	58.65	21-09-2022	Kanyon	Female	45.0	Credit Card
99453	I325143	C569580	Food & Beverage	2	10.46	22-09-2021	Forum Istanbul	Male	27.0	Cash
99454	I824010	C103292	Food & Beverage	2	10.46	28-03-2021	Metrocity	Male	63.0	Debit Card
99455	I702964	C800631	Technology	4	4200.00	16-03-2021	Istinye Park	Male	56.0	Cash
99456	I232867	C273973	Souvenir	3	35.19	15-10-2022	Mall of Istanbul	Female	36.0	Credit Card

99338 rows x 10 columns

Al analizar los datos del DataFrame con `unificado_df.info()`, observamos que la columna "age" tenía menos datos que las demás. Al aplicar una limpieza del DataFrame y eliminar los valores nulos, descubrimos que se eliminaron 119 filas, reduciendo el total de 99.457 a 99.338 filas.

Después, transformamos las fechas del formato dd/mm/yyyy a año/mes en una nueva columna para obtener mejores resultados en análisis futuros. La teoría detrás de este cambio es que, al agregar datos en intervalos de tiempo más amplios, como meses o años, se facilita la identificación de tendencias y patrones en los datos. Esto se debe a que trabajar con fechas más simplificadas reduce el nivel de detalle de los datos, lo que a su vez mejora la visualización y el análisis. Además, al tener menos datos únicos, se optimiza el rendimiento computacional, haciendo el proceso más eficiente y manejable.

2. Conversión de Formato de Fechas

```
Código:
unificado_df['invoice_date'] =
pd.to_date(unificado_df['invoice_date'], format='%d-%m-%Y')
unificado_df['year_month'] =
unificado_df['invoice_date'].dt.to_period('M').astype(str)
fecha_global_df = unificado_df['year_month']
unificado_df
```

Figura 9. Conversión de Formato de Fechas

```
unificado_df['invoice_date'] = pd.to_datetime(unificado_df['invoice_date'], format='%d-%m-%Y')
unificado_df['year_month'] = unificado_df['invoice_date'].dt.to_period('M').astype(str)
fecha_global_df = unificado_df['year_month']
unificado_df
```

	invoice_no	customer_id	category	quantity	price	invoice_date	shopping_mall	gender	age	payment_method	year_month
0	I138884	C241288	Clothing	5	1500.40	2022-08-05	Kanyon	Female	28.0	Credit Card	2022-08
1	I317333	C111565	Shoes	3	1800.51	2021-12-12	Forum Istanbul	Male	21.0	Debit Card	2021-12
2	I127801	C266599	Clothing	1	300.08	2021-11-09	Metrocity	Male	20.0	Cash	2021-11
3	I173702	C988172	Shoes	5	3000.85	2021-05-16	Metropol AVM	Female	66.0	Credit Card	2021-05
4	I337046	C189076	Books	4	60.60	2021-10-24	Kanyon	Female	53.0	Cash	2021-10
...
99452	I219422	C441542	Souvenir	5	58.85	2022-09-21	Kanyon	Female	45.0	Credit Card	2022-09
99453	I325143	C569580	Food & Beverage	2	10.46	2021-09-22	Forum Istanbul	Male	27.0	Cash	2021-09
99454	I824010	C103292	Food & Beverage	2	10.46	2021-03-28	Metrocity	Male	63.0	Debit Card	2021-03
99455	I702964	C800631	Technology	4	4200.00	2021-03-16	Istinye Park	Male	56.0	Cash	2021-03
99456	I232867	C273973	Souvenir	3	35.19	2022-10-15	Mall of Istanbul	Female	36.0	Credit Card	2022-10

99338 rows x 11 columns

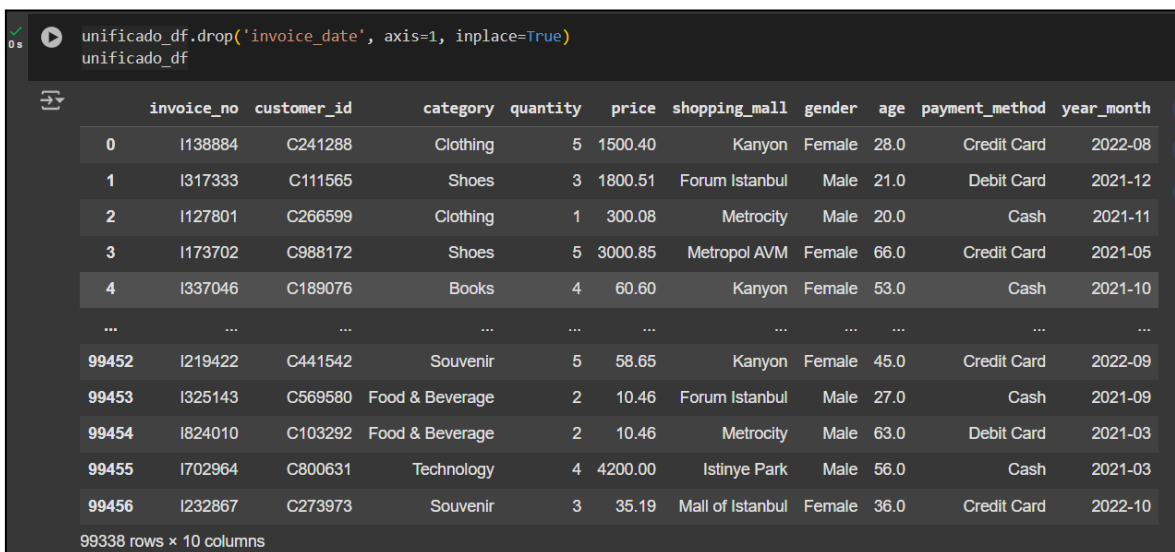
Tras un análisis preliminar de los datos, se observa que las fechas (año/mes) coinciden con las originales. Además, se nota un incremento en la cantidad de columnas, pasando a un total de 11.

A continuación, se procede a eliminar la columna "invoice_date", que contenía las fechas originales de los archivos .csv.

3. Limpieza de Fechas

Código: `unificado_df.drop('invoice_date', axis=1, inplace=True)`
`unificado_df`

Figura 10. Limpieza de fechas



```
unificado_df.drop('invoice_date', axis=1, inplace=True)
unificado_df
```

	invoice_no	customer_id	category	quantity	price	shopping_mall	gender	age	payment_method	year_month
0	I138884	C241288	Clothing	5	1500.40	Kanyon	Female	28.0	Credit Card	2022-08
1	I317333	C111565	Shoes	3	1800.51	Forum Istanbul	Male	21.0	Debit Card	2021-12
2	I127801	C266599	Clothing	1	300.08	Metrocity	Male	20.0	Cash	2021-11
3	I173702	C988172	Shoes	5	3000.85	Metropol AVM	Female	66.0	Credit Card	2021-05
4	I337046	C189076	Books	4	60.60	Kanyon	Female	53.0	Cash	2021-10
...
99452	I219422	C441542	Souvenir	5	58.65	Kanyon	Female	45.0	Credit Card	2022-09
99453	I325143	C569580	Food & Beverage	2	10.46	Forum Istanbul	Male	27.0	Cash	2021-09
99454	I824010	C103292	Food & Beverage	2	10.46	Metrocity	Male	63.0	Debit Card	2021-03
99455	I702964	C800631	Technology	4	4200.00	Istinye Park	Male	56.0	Cash	2021-03
99456	I232867	C273973	Souvenir	3	35.19	Mall of Istanbul	Female	36.0	Credit Card	2022-10

99338 rows x 10 columns

Se elimina la columna "invoice_date" y se visualizan los datos para corroborar.

Luego, creamos una nueva columna llamada "rango_etario" que clasifica los datos de la siguiente manera:

- Menor de 25 años (>24)
- De 25 a 35 años (25 a 35)
- De 36 a 45 años (36 a 45)
- De 46 a 60 años (46 a 60)
- Mayor de 60 años (<61)

4. Clasificación por Rango Etario

```
Código: bins = [0, 24, 35, 45, 60, float('inf')]

labels = ['>24', '25 a 35', '36 a 45', '46 a 60', '<61']

unificado_df['rango_etario'] = pd.cut(unificado_df['age'],
bins=bins, labels=labels)

unificado_df
```

Figura 11. Rango Etario

```
bins = [0, 24, 35, 45, 60, float('inf')]
labels = ['>24', '25 a 35', '36 a 45', '46 a 60', '<61']
unificado_df['rango_etario'] = pd.cut(unificado_df['age'], bins=bins, labels=labels)
unificado_df
```

	invoice_no	customer_id	category	quantity	price	shopping_mall	gender	age	payment_method	year_month	rango_etario
0	I138884	C241288	Clothing	5	1500.40	Kanyon	Female	28.0	Credit Card	2022-08	25 a 35
1	I317333	C111565	Shoes	3	1800.51	Forum Istanbul	Male	21.0	Debit Card	2021-12	>24
2	I127801	C266599	Clothing	1	300.08	Metrocity	Male	20.0	Cash	2021-11	>24
3	I173702	C988172	Shoes	5	3000.85	Metropol AVM	Female	66.0	Credit Card	2021-05	<61
4	I337046	C189076	Books	4	60.60	Kanyon	Female	53.0	Cash	2021-10	46 a 60
...
99452	I219422	C441542	Souvenir	5	58.65	Kanyon	Female	45.0	Credit Card	2022-09	36 a 45
99453	I325143	C569580	Food & Beverage	2	10.46	Forum Istanbul	Male	27.0	Cash	2021-09	25 a 35
99454	I824010	C103292	Food & Beverage	2	10.46	Metrocity	Male	63.0	Debit Card	2021-03	<61
99455	I702964	C800631	Technology	4	4200.00	Istinye Park	Male	56.0	Cash	2021-03	46 a 60
99456	I232867	C273973	Souvenir	3	35.19	Mall of Istanbul	Female	36.0	Credit Card	2022-10	36 a 45

99338 rows x 11 columns

Para clasificar a los consumidores según su edad, se genera una nueva columna llamada "rango_etario" utilizando la función `pd.cut()` de pandas. Primero, se definen los límites (bins) y se nombran los rangos (labels). La clasificación por rangos etarios es una técnica comúnmente utilizada en análisis

de datos para segmentar poblaciones y facilitar el análisis de tendencias y comportamientos dentro de cada grupo. Al agrupar las edades en categorías, se pueden identificar patrones específicos y realizar análisis más profundos y significativos.

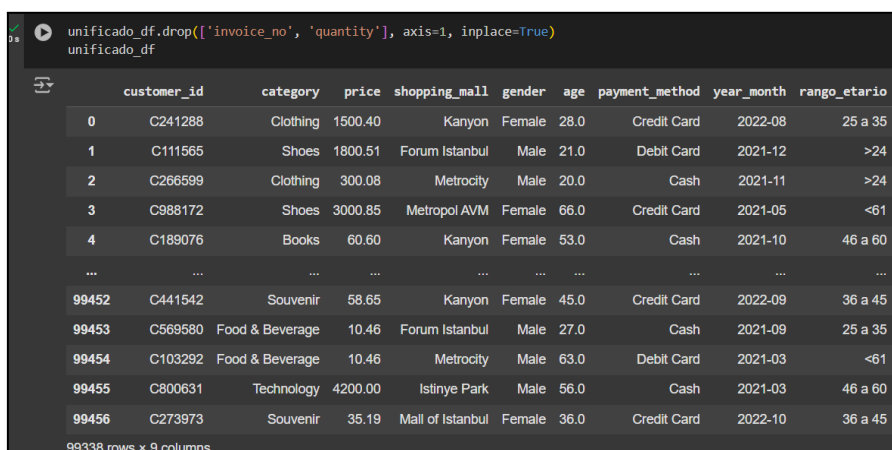
Después, se visualiza el DataFrame para un chequeo general. A simple vista, se verifica que las edades coinciden con los rangos asignados y que el DataFrame ahora contiene una columna adicional, totalizando 11 columnas. Este paso garantiza que la nueva columna se ha generado correctamente y que la clasificación de edades se ha realizado de manera precisa.

Luego, para optimizar y preparar el DataFrame para el proceso de carga, se eliminan las columnas "invoice_no" y "quantity", puesto que no se utilizarán en los análisis posteriores.

5. Limpieza y Preparación del DataFrame

Código: `unificado_df.drop(['invoice_no', 'quantity'], axis=1, inplace=True)`
`unificado_df`

Figura 12. Limpieza y preparación del DF



```
unificado_df.drop(['invoice_no', 'quantity'], axis=1, inplace=True)
unificado_df
```

	customer_id	category	price	shopping_mall	gender	age	payment_method	year_month	rango_etario
0	C241288	Clothing	1500.40	Kanyon	Female	28.0	Credit Card	2022-08	25 a 35
1	C111565	Shoes	1800.51	Forum Istanbul	Male	21.0	Debit Card	2021-12	>24
2	C266599	Clothing	300.08	Metrocity	Male	20.0	Cash	2021-11	>24
3	C988172	Shoes	3000.85	Metropol AVM	Female	66.0	Credit Card	2021-05	<61
4	C189076	Books	60.60	Kanyon	Female	53.0	Cash	2021-10	46 a 60
...
99452	C441542	Souvenir	58.65	Kanyon	Female	45.0	Credit Card	2022-09	36 a 45
99453	C569580	Food & Beverage	10.46	Forum Istanbul	Male	27.0	Cash	2021-09	25 a 35
99454	C103292	Food & Beverage	10.46	Metrocity	Male	63.0	Debit Card	2021-03	<61
99455	C800631	Technology	4200.00	Istinye Park	Male	56.0	Cash	2021-03	46 a 60
99456	C273973	Souvenir	35.19	Mall of Istanbul	Female	36.0	Credit Card	2022-10	36 a 45

99338 rows x 9 columns

Al eliminar las dos columnas mencionadas y visualizar el DataFrame, observamos que el número de columnas se reduce a 9.

Con todas las modificaciones y actualizaciones realizadas durante la etapa de Transformación, hemos generado un nuevo DataFrame limpio. Este paso finaliza el proceso de transformación y nos permite avanzar a la siguiente fase: la Carga (LOAD).

6. Finalización de la Transformación

```
Código: nuevo_df = unificado_df.copy()

nuevo_df.info()
```

Figura 13. *Finalización de la Transformación*

```
[22] nuevo_df = unificado_df.copy()
nuevo_df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 99338 entries, 0 to 99456
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   customer_id     99338 non-null  object
1   category        99338 non-null  object
2   price           99338 non-null  float64
3   shopping_mall   99338 non-null  object
4   gender          99338 non-null  object
5   age             99338 non-null  float64
6   payment_method  99338 non-null  object
7   year_month      99338 non-null  object
8   rango_etario    99338 non-null  category
dtypes: category(1), float64(2), object(6)
memory usage: 8.9+ MB
```

1.3. Limpieza de Datos (Load)

La etapa de carga (LOAD) en el proceso ETL (Extract, Transform, Load) es fundamental para trasladar los datos transformados y limpios a su destino final, como un almacén de datos, una base de datos o un sistema analítico. En esta

fase, nos aseguramos de que los datos estén en el formato adecuado y sean accesibles para futuras consultas y análisis.

Aplicamos restricciones de integridad para asegurar la validez de los datos. Dichas restricciones son fundamentales en el manejo de datos, ya que garantizan que la información cumpla con ciertos criterios de calidad y consistencia, lo que es esencial para obtener resultados de análisis precisos y fiables.

1. Aplicación de Restricciones de Integridad

Código:

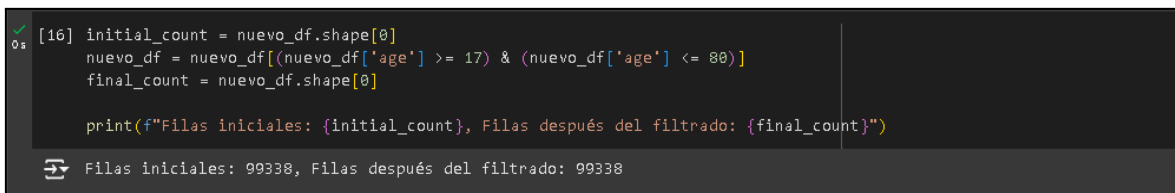
```
initial_count = nuevo_df.shape[0]

nuevo_df = nuevo_df[(nuevo_df['age'] >= 17) & (nuevo_df['age'] <=
80)]

final_count = nuevo_df.shape[0]

print (f"Filas iniciales: {initial_count}, Filas después del
filtro: {final_count}")
```

Figura 14. *Aplicación de Restricciones de Integridad*



```
[16] initial_count = nuevo_df.shape[0]
nuevo_df = nuevo_df[(nuevo_df['age'] >= 17) & (nuevo_df['age'] <= 80)]
final_count = nuevo_df.shape[0]

print(f"Filas iniciales: {initial_count}, Filas después del filtrado: {final_count}")

Filas iniciales: 99338, Filas después del filtrado: 99338
```

En este caso, aplicamos una restricción a la columna "edad". Primero, contamos el número inicial de filas. Luego, filtramos para mantener las edades entre 18 y 80 años y realizamos un conteo final. Al imprimir los resultados, verificamos que la cantidad de filas se mantiene consistente.

Filtrar las edades de 18 a 80 años en el dataframe tiene mucho sentido desde el punto de vista del análisis de datos:

1. **Relevancia:** Nos enfocamos en los consumidores económicamente activos y relevantes para el análisis de ventas.
2. **Calidad:** Eliminamos datos atípicos que podrían distorsionar los resultados.
3. **Precisión:** Ayuda a obtener insights más precisos y accionables.

Este filtro mejora la calidad del análisis y facilita la interpretación de los resultados.

2. Proceso de Análisis de Datos

El análisis de datos es una etapa elemental para extraer información valiosa y tomar decisiones informadas. En este proceso, utilizamos diversas técnicas y herramientas para explorar, visualizar y entender los datos disponibles. A lo largo del análisis, planteamos diferentes consultas y visualizaciones que nos permiten identificar patrones, tendencias y comportamientos clave dentro del dataset.

En este contexto, abordaremos diferentes aspectos como la distribución de género, las preferencias de pago y las categorías de productos comprados. También examinaremos cómo estos factores se relacionan con variables como la edad y el rango etario de los consumidores. Todo esto nos permitirá obtener insights significativos que nos ayuden a entender mejor a nuestros clientes y optimizar nuestras estrategias de negocio.

1. Consultas

Ahora, detallaremos las consultas y visualizaciones que realizaremos en el proceso de análisis.

a. Cantidades y porcentaje de mujeres y hombres que contiene el DataFrame.

```
Código: porcentaje_genero = (nuevo_df['gender'].value_counts(normalize=True)
* 100).round(2)

porcentaje = pd.DataFrame({'Porcentaje (%)': porcentaje_genero})

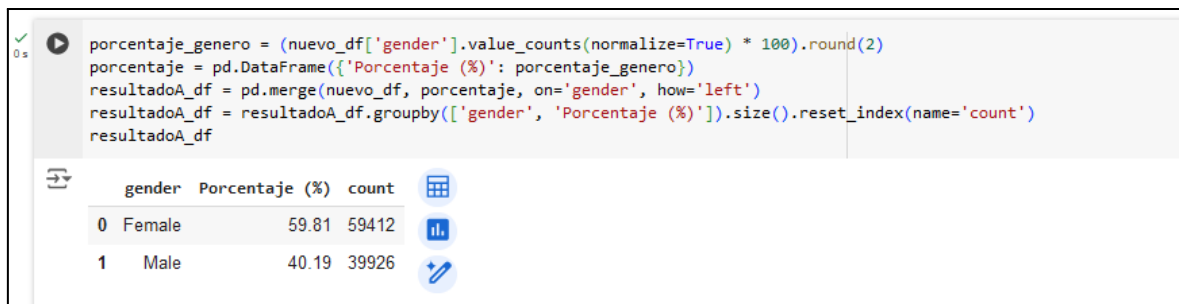
resultadoA_df = pd.merge(nuevo_df, porcentaje, on='gender',
```

```
how='left')

resultadoA_df = resultadoA_df.groupby(['gender', 'Porcentaje (%)']).size().reset_index(name='count')

resultadoA_df
```

Figura 15. Cantidades y porcentaje de mujeres y hombres



Para visualizar las cantidades con sus porcentajes correspondientes acorde al género del DataFrame, primero realizamos el conteo y transformamos ese número a porcentaje con 2 decimales. Generamos un DataFrame (porcentaje) asignando los valores del porcentaje para luego combinarlo (pd.merge) al DataFrame `nuevo_df`. Por último, agrupamos las columnas “gender” y “Porcentaje (%)” y realizamos el conteo para visualizarlo y se agrega en la columna “count”.

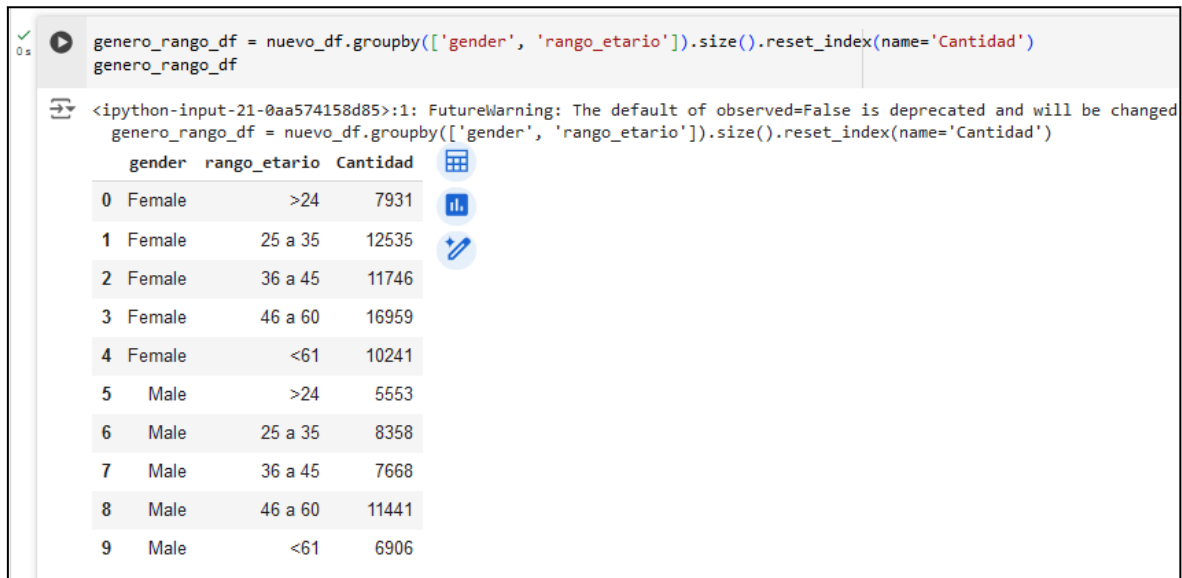
b. Cantidad de mujeres y hombres por rango etario

Código:

```
genero_rango_df = nuevo_df.groupby(['gender', 'rango_etario']).size().reset_index(name='Cantidad')

genero_rango_df
```

Figura 16. Cantidad de mujeres y hombres por rango etario



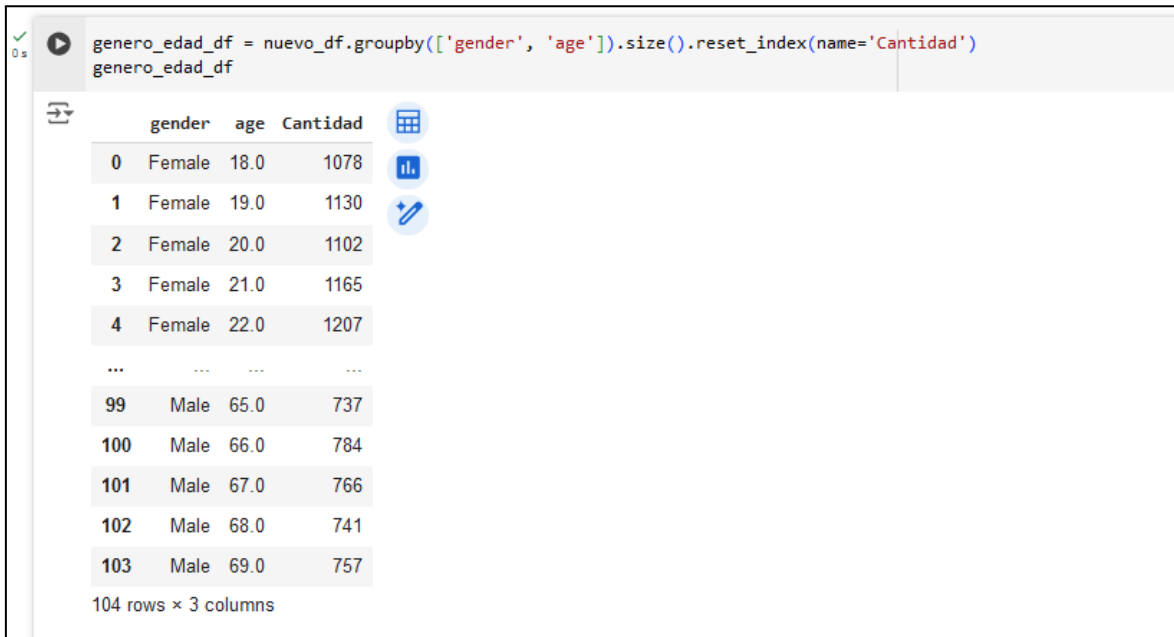
Para visualizar la cantidad de mujeres y hombres por rango etario, agrupamos las columnas "gender" y "rango_etario" y asignamos las cantidades a la columna "Cantidad".

c. Cantidad de consumidores por edad

Código:

```
genero_edad_df = nuevo_df.groupby(['gender',
'age']).size().reset_index(name='Cantidad')
genero_edad_df
```

Figura 17. Cantidad de consumidores por edad



```

genero_edad_df = nuevo_df.groupby(['gender', 'age']).size().reset_index(name='Cantidad')
genero_edad_df

```

	gender	age	Cantidad
0	Female	18.0	1078
1	Female	19.0	1130
2	Female	20.0	1102
3	Female	21.0	1165
4	Female	22.0	1207
...
99	Male	65.0	737
100	Male	66.0	784
101	Male	67.0	766
102	Male	68.0	741
103	Male	69.0	757

104 rows × 3 columns

Para visualizar la cantidad de consumidores por edad, agrupamos las columnas "gender" y "age" y asignamos las cantidades a la columna "Cantidad".

d. Modo de pago más frecuente de todos los consumidores categorizados por su género.

Código:

```

pago_genero_df = nuevo_df.groupby(['gender',
'payment_method']).size().reset_index(name='Cantidad')

pago_genero_df = pago_genero_df.sort_values(by='Cantidad',
ascending=False).reset_index(drop=True)

pago_genero_df

```

Figura 18. Modo de pago más frecuente categorizados por género

```

0 s
pago_genero_df = nuevo_df.groupby(['gender', 'payment_method']).size().reset_index(name='Cantidad')
pago_genero_df = pago_genero_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)
pago_genero_df

```

	gender	payment_method	Cantidad
0	Female	Cash	26479
1	Female	Credit Card	20994
2	Male	Cash	17918
3	Male	Credit Card	13904
4	Female	Debit Card	11939
5	Male	Debit Card	8104

Para visualizar el modo de pago más frecuente según el género de los consumidores, se agruparon las columnas “gender” y “payment_method” y se le asignaron las cantidades a la columna “Cantidad”. A su vez, se solicitó que se ordenen de modo descendente acorde a la cantidad de consumidores.

e. Método de pago más utilizado por el rango etario.

Código:

```

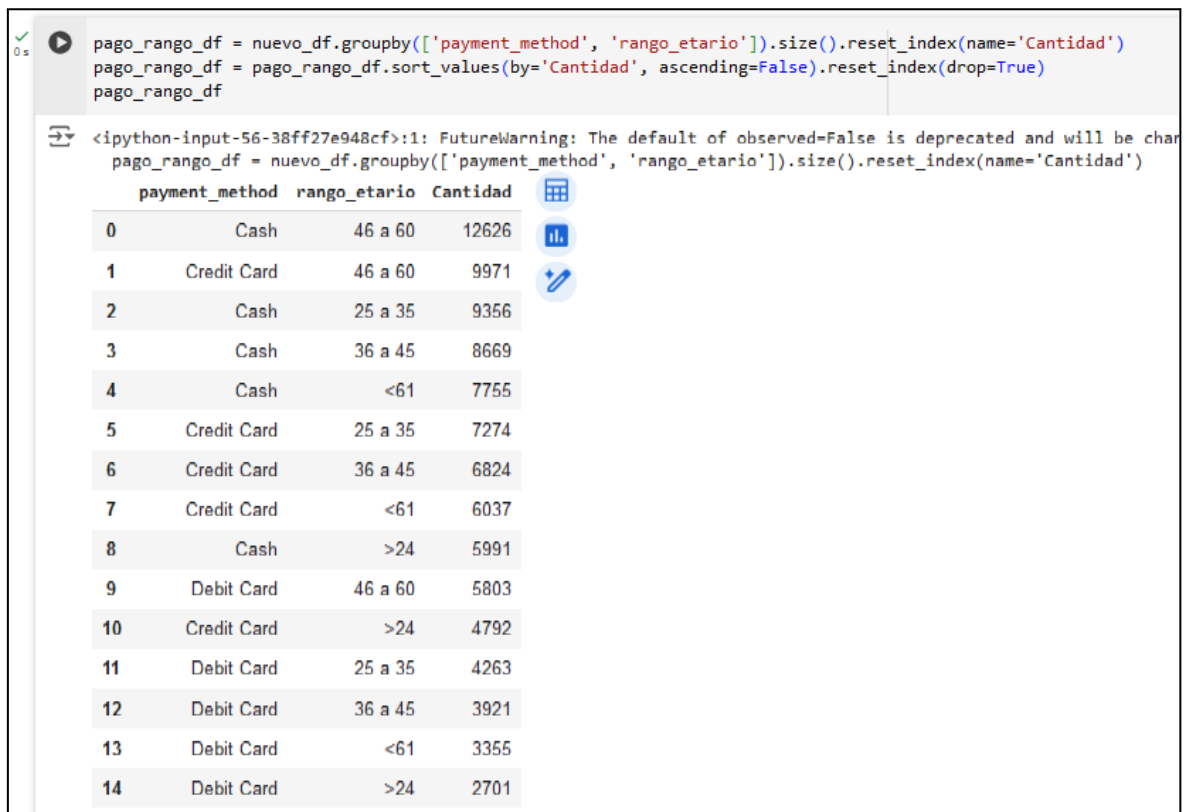
pago_rango_df = nuevo_df.groupby(['payment_method',
'rango_etario']).size().reset_index(name='Cantidad')

pago_rango_df = pago_rango_df.sort_values(by='Cantidad',
ascending=False).reset_index(drop=True)

pago_rango_df

```

Figura 19. Método de pago más utilizado por el rango etario



Para visualizar el modo de pago más utilizado por rango etario, agrupamos las columnas "rango_etario" y "payment_method" y asignamos las cantidades a la columna "Cantidad". Luego, ordenamos de forma descendente según la cantidad de consumidores.

f. Queremos determinar los precios de los productos por categoría

Código:

```

precio_cat_df = nuevo_df.groupby(['category',
'price']).size().reset_index(name='Cantidad')
precio_cat_df

```

Figura 20. Precios de los productos por categoría

```
precio_cat_df = nuevo_df.groupby(['category', 'price']).size().reset_index(name='Cantidad')
precio_cat_df
```

	category	price	Cantidad
0	Books	15.15	984
1	Books	30.30	1007
2	Books	45.45	1000
3	Books	60.60	958
4	Books	75.75	1027
5	Clothing	300.08	6834
6	Clothing	600.16	6939
7	Clothing	900.24	6899
8	Clothing	1200.32	6838
9	Clothing	1500.40	6935
10	Cosmetics	40.66	2983
11	Cosmetics	81.32	2971
12	Cosmetics	121.98	3069
13	Cosmetics	162.64	3010
14	Cosmetics	203.30	3051
15	Food & Beverage	5.23	2999
16	Food & Beverage	10.46	2906
17	Food & Beverage	15.69	2990
18	Food & Beverage	20.92	2879

Para visualizar los precios de los productos por categoría, agrupamos las columnas "category" y "price" y asignamos las cantidades a la columna "Cantidad". Luego, mostramos el precio mínimo, máximo y promedio por categoría.

g. Solicitamos que nos muestre el precio mínimo, máximo y el promedio por categoría.

Código:

```
precios_por_categoria_df =
nuevo_df.groupby('category')['price'].agg(['min', 'max',
'mean']).reset_index()
precios_por_categoria_df.rename(columns={'min': 'precio_minimo',
```



```
'max': 'precio_maximo', 'mean': 'precio_promedio'}, inplace=True)

precios_por_categoria_df
```

Figura 21. Precios Máximo, Mínimo y Promedio

```
[28] precios_por_categoria_df = nuevo_df.groupby('category')['price'].agg(['min', 'max', 'mean']).reset_index()
      precios_por_categoria_df.rename(columns={'min': 'precio_minimo', 'max': 'precio_maximo', 'mean': 'precio_promedio'}, inplace=True)
      precios_por_categoria_df
```

	category	precio_minimo	precio_maximo	precio_promedio
0	Books	15.15	75.75	45.562651
1	Clothing	300.08	1500.40	901.119898
2	Cosmetics	40.66	203.30	122.451725
3	Food & Beverage	5.23	26.15	15.671930
4	Shoes	600.17	3000.85	1807.281763
5	Souvenir	11.73	58.65	34.884470
6	Technology	1050.00	5250.00	3157.147147
7	Toys	35.84	179.20	107.754876

h. Solicitamos visualizar las categorías de los productos comprados por género y método de pago.

```
Código: categoria_genero_df = nuevo_df.groupby(['category', 'gender',
'payment_method']).size().reset_index(name='Cantidad')

categoria_genero_df

categoria_genero_df.sort_values(by='Cantidad',
ascending=False).reset_index(drop=True)

categoria_genero_df
```

Figura 22. Categorías de los productos comprados por género y método de pago.

```

categoria_genero_df = nuevo_df.groupby(['category', 'gender', 'payment_method']).size().reset_index(name='Cantidad')
categoria_genero_df = categoria_genero_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)
categoria_genero_df

```

	category	gender	payment_method	Cantidad
0	Clothing	Female	Cash	9266
1	Clothing	Female	Credit Card	7175
2	Clothing	Male	Cash	6170
3	Clothing	Male	Credit Card	4840
4	Clothing	Female	Debit Card	4190
5	Food & Beverage	Female	Cash	3911
6	Cosmetics	Female	Cash	3903
7	Cosmetics	Female	Credit Card	3281
8	Food & Beverage	Female	Credit Card	3127
9	Clothing	Male	Debit Card	2804
10	Cosmetics	Male	Cash	2766
11	Toys	Female	Cash	2729
12	Food & Beverage	Male	Cash	2671
13	Shoes	Female	Cash	2652
14	Toys	Female	Credit Card	2154
15	Shoes	Female	Credit Card	2136

Para visualizar las categorías de los productos comprados por el género y el método de pago más utilizado de los consumidores, se agruparon las columnas “category”, “gender” y “payment_method” y se le asignaron las cantidades a la columna “Cantidad”. A su vez, se solicitó que se ordenen de modo descendente acorde a la cantidad de consumidores.

i. Solicitamos visualizar la cantidad de compras por categoría según la fecha (año/mes).

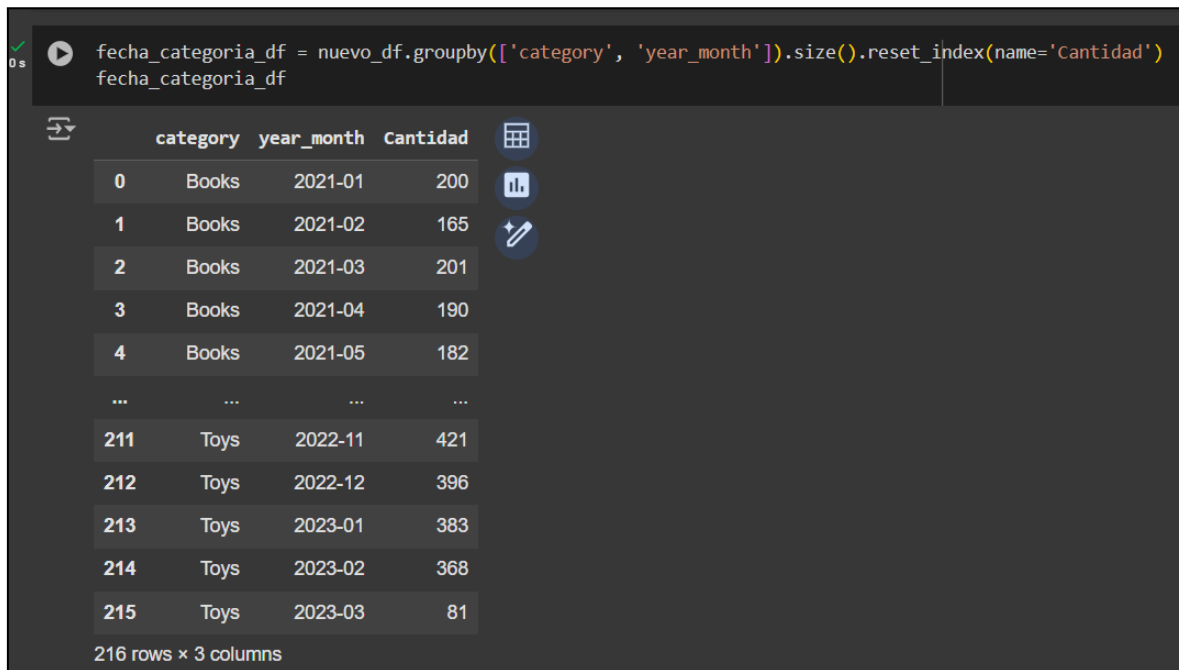
Código:

```

fecha_categoria_df = nuevo_df.groupby(['category',
'year_month']).size().reset_index(name='Cantidad')
fecha_categoria_df

```

Figura 23. Cantidad de compras por categoría según la fecha (año/mes)



j. Solicitamos visualizar el método de pago acorde al precio.

Código:

```

pago_precio_df = nuevo_df.groupby(['price',
'payment_method']).size().reset_index(name='Cantidad')

pago_precio_df = pago_precio_df.sort_values(by='Cantidad',
ascending=False).reset_index(drop=True)

pago_precio_df

```

Figura 24. Método de pago acorde al precio

```

pagoPrecio_df = nuevo_df.groupby(['price', 'payment_method']).size().reset_index(name='Cantidad')
pagoPrecio_df = pagoPrecio_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)
pagoPrecio_df

```

	price	payment_method	Cantidad
0	1500.40	Cash	3146
1	600.16	Cash	3123
2	900.24	Cash	3117
3	1200.32	Cash	3036
4	300.08	Cash	3014
...
115	4200.00	Debit Card	198
116	60.60	Debit Card	196
117	46.92	Debit Card	194
118	30.30	Debit Card	192
119	3150.00	Debit Card	180

120 rows × 3 columns

Para visualizar el método de pago por los precios de los productos, se agruparon las columnas “price” y “payment_method” y se le asignaron las cantidades a la columna “Cantidad”. A su vez, se solicitó que se ordenen de modo descendente acorde a la cantidad de consumos.

k. Solicitamos visualizar el lugar de compra (shopping_mall) acorde al género y rango etario.

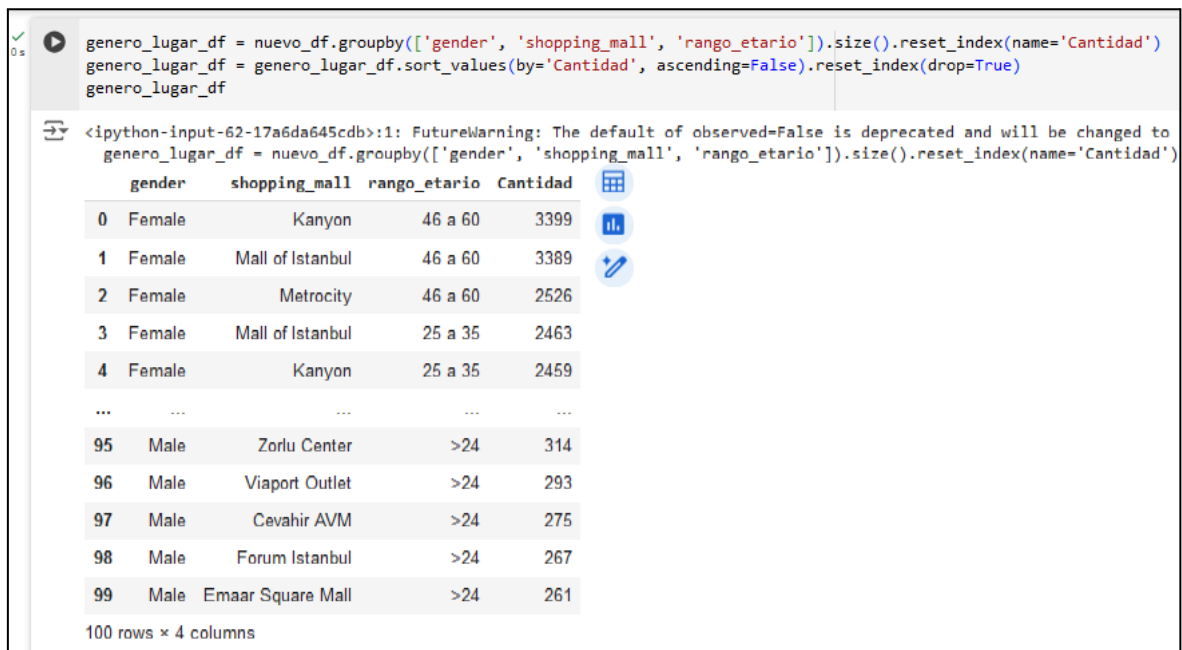
Código:

```

genero_lugar_df = nuevo_df.groupby(['gender', 'shopping_mall', 'rango_etario']).size().reset_index(name='Cantidad')
genero_lugar_df = genero_lugar_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)
genero_lugar_df

```

Figura 25. Lugar de compra (shopping_mall) acorde al género y rango etario.



Para visualizar el lugar de compra teniendo en cuenta el género y rango etario de los consumidores, se agruparon las columnas “gender”, “shopping_mall” y “rango_etario” y se le asignaron las cantidades a la columna “Cantidad”. A su vez, se solicitó que se ordenen de modo descendente acorde a la cantidad de consumos.

I. Solicitamos visualizar el método de pago acorde al lugar de compra (shopping_mall).

Código:

```

pago_lugar_df = nuevo_df.groupby(['payment_method', 'shopping_mall']).size().reset_index(name='Cantidad')

pago_lugar_df = pago_lugar_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)

pago_lugar_df

```

Figura 26. Método de pago acorde al lugar de compra (shopping_mall)

```

pago_lugar_df = nuevo_df.groupby(['payment_method', 'shopping_mall']).size().reset_index(name='Cantidad')
pago_lugar_df = pago_lugar_df.sort_values(by='Cantidad', ascending=False).reset_index(drop=True)
pago_lugar_df

```

	payment_method	shopping_mall	Cantidad
0	Cash	Mall of Istanbul	8882
1	Cash	Kanyon	8845
2	Credit Card	Mall of Istanbul	7012
3	Credit Card	Kanyon	6910
4	Cash	Metrocity	6619
5	Credit Card	Metrocity	5342
6	Cash	Metropol AVM	4553
7	Cash	Istinye Park	4430
8	Debit Card	Kanyon	4050
9	Debit Card	Mall of Istanbul	4020
10	Credit Card	Metropol AVM	3519
11	Credit Card	Istinye Park	3415
12	Debit Card	Metrocity	3035
13	Cash	Zorlu Center	2320
14	Cash	Cevahir AVM	2228

Para visualizar el método de pago acorde al lugar de compra, se agruparon las columnas “payment_method” y “shopping_mall” y se le asignaron las cantidades a la columna “Cantidad”. A su vez, se solicitó que se ordenen de modo descendente acorde a la cantidad de consumos.

m. Solicitamos visualizar los precios de compra acorde al género de la persona.

Código:

```

genero_precio_df = nuevo_df.groupby(['gender',
'price']).size().reset_index(name='Cantidad')
genero_precio_df

```

Figura 27. Precios de compra acorde al género de la persona

✓ 0 s

```
genero_precio_df = nuevo_df.groupby(['gender', 'price']).size().reset_index(name='Cantidad')
genero_precio_df
```

	gender	price	Cantidad
0	Female	5.23	1803
1	Female	10.46	1700
2	Female	11.73	604
3	Female	15.15	567
4	Female	15.69	1780
...
75	Male	2400.68	806
76	Male	3000.85	820
77	Male	3150.00	435
78	Male	4200.00	402
79	Male	5250.00	392

80 rows × 3 columns

n. Solicitamos visualizar las categorías de compra acorde a la fecha (año/mes).

Código:

```
nuevo_df['year_month'] = pd.to_datetime(nuevo_df['year_month'],
format='%Y-%m')

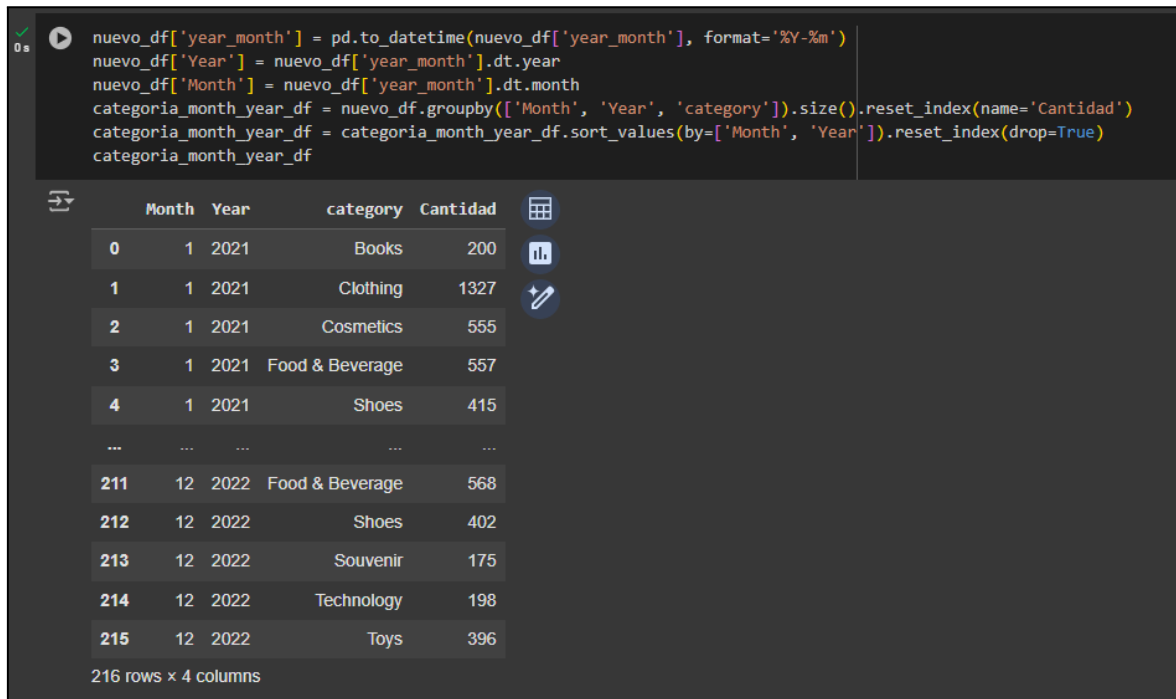
nuevo_df['Year'] = nuevo_df['year_month'].dt.year
nuevo_df['Month'] = nuevo_df['year_month'].dt.month

categoria_month_year_df = nuevo_df.groupby(['Month', 'Year',
'category']).size().reset_index(name='Cantidad')

categoria_month_year_df =
categoria_month_year_df.sort_values(by=['Month',
'Year']).reset_index(drop=True)

categoria_month_year_df
```

Figura 28. Categorías de compra acorde a la fecha (año/mes)



Para visualizar la categoría de los consumos acorde a la fechas de dichos consumos, primero realizamos algunas modificaciones para una mejor visualización.

El primer paso fue convertir la columna “year_month” a formato fecha, por las dudas que tuviese otro formato. Segundo, separamos los meses y los años en columnas distintas, para facilitar el filtrado y visualización. Luego, agrupamos los datos por mes, año y categoría e incluimos la columna Cantidad con los valores de los consumos. Finalmente, ordenamos los datos para poder visualizarlos de manera ordenada por mes.

2. Análisis de Tendencias y Comportamientos

Este análisis examina detalladamente las tendencias de ventas minoristas durante el período 2021-2023, basándose en un conjunto de datos que comprende 99.338 transacciones. El estudio revela patrones significativos en el comportamiento de compra, preferencias demográficas y oportunidades de optimización comercial que pueden resultar fundamentales para la toma de decisiones estratégicas en el sector.

2.1. Perfil Demográfico y Comportamiento del Consumidor

El análisis de los datos de consumidores revela información clave sobre su perfil demográfico y comportamientos de compra, lo que es esencial para diseñar estrategias comerciales efectivas. La mayoría de los consumidores son mujeres, representando el 59.81% del total, con una concentración significativa en los rangos etarios de 25 a 35 años y de 46 a 60 años. Por otro lado, los hombres constituyen el 40.19% de la muestra, destacándose también en estos mismos rangos etarios. Este patrón sugiere que tanto hombres como mujeres de mediana edad son compradores activos, lo que implica que las campañas de marketing deben enfocarse especialmente en estos segmentos.

El análisis de métodos de pago muestra que el efectivo es el método más utilizado, especialmente entre los consumidores de 46 a 60 años, quienes realizaron 12,626 transacciones en efectivo. Esto resalta una preferencia por las transacciones físicas en este grupo etario, posiblemente reflejando una mayor comodidad con el manejo de efectivo. Sin embargo, las tarjetas de crédito también tienen una presencia significativa, especialmente entre las mujeres de 25

a 35 años, donde se registran 7,274 transacciones. Este hallazgo indica una creciente preferencia por las compras en línea y métodos digitales entre este grupo.

En cuanto a las preferencias por categoría de producto, las mujeres predominan en categorías como "Ropa" y "Cosméticos", utilizando principalmente efectivo para estas compras. En contraste, los hombres tienden a gastar más en "Ropa" y "Alimentos y Bebidas", mostrando un uso notable de tarjetas para productos de mayor valor, como "Tecnología" y "Libros". Este perfil demográfico y de comportamiento sugiere que las marcas deben implementar campañas específicas que aborden las preferencias de compra de diferentes grupos. Ofrecer incentivos para pagos digitales y enfocarse en las categorías más populares para cada género puede mejorar la satisfacción del consumidor y aumentar las ventas.

2.2. Patrones de Consumo y Preferencias de Categoría

El estudio de los patrones de consumo y preferencias de categoría revela información esencial sobre el comportamiento de compra de los consumidores. La categoría de "Clothing" (Ropa) se destaca como la más popular, con una cantidad significativa de transacciones y un precio promedio de 901.12, lo que indica que los consumidores están dispuestos a invertir en ropa de calidad.

La categoría de "Technology" (Tecnología) presenta un rango de precios más alto, con un precio promedio de 3157.15. Aunque la cantidad de transacciones es menor en comparación con la ropa, el valor de cada compra es significativamente mayor, lo que sugiere que los consumidores son selectivos en sus compras tecnológicas.

Las categorías de "Cosmetics" (Cosméticos) y "Toys" (Juguetes) también muestran patrones interesantes. Los cosméticos tienen un precio promedio de 122.45, indicando que los consumidores buscan productos accesibles pero de calidad. Los juguetes, con un precio promedio de 107.75, muestran que los padres están dispuestos a invertir en productos que estimulan el desarrollo de sus hijos.

La categoría de "Food & Beverage" (Alimentos y Bebidas) tiene precios más bajos, con un promedio de 15.67, sugiriendo que los consumidores tienden a elegir productos cotidianos y accesibles en esta categoría. Las categorías de "Souvenirs" (Recuerdos) tienen un precio promedio de 34.88, lo que indica una tendencia a comprar recuerdos a precios razonables.

2.3. Métodos de pago y Transformación Digital

El análisis de los métodos de pago revela importantes tendencias en el comportamiento de compra de los consumidores, especialmente en el contexto de la transformación digital. El uso del efectivo sigue siendo predominante en los centros comerciales, con el "Mall of Istanbul" y "Kanyon" liderando en transacciones en efectivo, con 8,882 y 8,845 respectivamente. Esto sugiere que, a pesar del avance de las tecnologías digitales, muchos consumidores aún prefieren pagar en efectivo, lo que podría reflejar mayor comodidad o confianza en este método tradicional.

Sin embargo, las tarjetas de crédito también muestran una considerable aceptación. En el "Mall of Istanbul", se registraron 7,012 transacciones con tarjetas de crédito, lo que indica una tendencia creciente hacia la digitalización en los pagos. Este cambio sugiere que los consumidores están adoptando métodos más modernos, lo cual es esencial para las empresas que buscan adaptarse a un entorno de compra en evolución.

Las tarjetas de débito, aunque utilizadas en menor medida, todavía representan una parte importante de las transacciones, especialmente en centros comerciales como Kanyon y Metrocity. Esto podría indicar que, aunque los consumidores optan por métodos de pago digitales, muchos prefieren la seguridad y el control que ofrecen las tarjetas de débito.

En cuanto a los precios, las transacciones en efectivo abarcan desde compras pequeñas hasta inversiones más significativas. Por ejemplo, hay transacciones en efectivo de hasta 3,000.85, lo que demuestra que los consumidores están dispuestos a utilizar este método incluso para compras de alto valor. Por otro lado, las tarjetas de crédito también se utilizan para compras relativamente altas, resaltando la importancia de ofrecer opciones de pago que se alineen con las expectativas de los consumidores modernos.

Este panorama de métodos de pago refleja las preferencias actuales de los consumidores y subraya la necesidad de que las empresas adapten sus estrategias de pago para incluir una combinación de métodos tradicionales y digitales. La transformación digital en el sector minorista debe tener en cuenta la diversidad de preferencias de pago, garantizando así una experiencia de compra

fluida y satisfactoria para todos los clientes. Esta adaptabilidad será clave para mantener la competitividad en un mercado que avanza hacia la digitalización y la innovación.

Conclusión

El análisis de los datos ha revelado patrones interesantes en el comportamiento de consumo de nuestra base de clientes. Las mujeres son el principal motor de las transacciones, representando aproximadamente el 60% de las compras, especialmente en el grupo de edad de 46 a 60 años. Este hallazgo es clave para futuras estrategias de marketing y desarrollo de productos.

En cuanto a los hábitos de pago, el efectivo sigue siendo el método principal, seguido por las tarjetas de crédito y débito. Esta preferencia se mantiene constante en todas las categorías de productos y rangos de precios, reflejando un hábito de consumo arraigado en nuestros clientes.

La distribución geográfica de las ventas muestra que, aunque el estudio abarcó diez ubicaciones, tres centros comerciales se destacaron en volumen de transacciones. Sin embargo, los patrones de comportamiento de compra fueron consistentes en todas las ubicaciones, sugiriendo que los factores demográficos son más determinantes que la ubicación geográfica en las decisiones de compra.

Un descubrimiento significativo es la estabilidad temporal en los patrones de consumo. El análisis mes a mes y año a año muestra variaciones mínimas, inferiores al 5%. Esta estabilidad sugiere una base de clientes consolidada y un negocio resistente a fluctuaciones estacionales.

Estos hallazgos nos proporcionan una comprensión más profunda de nuestros clientes y una base sólida para la toma de decisiones estratégicas futuras. La consistencia en los patrones observados sugiere que podemos confiar

en estas tendencias para informar decisiones sobre gestión de inventario, estrategias de marketing y optimización de métodos de pago. Sin embargo, será vital monitorear continuamente estos patrones para detectar cambios emergentes y adaptar nuestras estrategias según sea necesario.

Bibliografía

Pandas - DataFrame reference. (n.d.). W3schools.com. Retrieved November 6, 2024, from https://www.w3schools.com/python/pandas/pandas_ref_dataframe.asp

Zero to hero pandas CheetSheet beginners. (2020, octubre 2). Kaggle.com;

Kaggle.

<https://www.kaggle.com/code/narendrageek/zero-to-hero-pandas-cheetsheet-beginners>