Data cleaning

This notebook carries out the complete preprocessing workflow for the retail sales datasets used in the project. The objective is to ensure that all source tables are clean, consistent, and ready for further exploratory analysis, modeling, and dashboarding.

Initialization (downloading data and inspecting it)

In [1]:
```
# Import necessary libraries
import pandas as pd
import os
import pyarrow.parquet as pq
```

In [2]:
```
# Setting up paths
project_root = os.path.dirname(os.getcwd())
data_path = os.path.join(project_root, "data_raw")

print("Project root:", project_root)
print("Data path:", data_path)
print("Files:", os.listdir(data_path))

# Loading raw data
df_cat = pd.read_csv(os.path.join(data_path, "categorias.csv"))
df_clientes = pd.read_csv(os.path.join(data_path, "clientes.csv"))
df_metodos = pd.read_csv(os.path.join(data_path, "metodos_pago.csv"))
df_prod = pd.read_csv(os.path.join(data_path, "productos.csv"))
df_ventas = pd.read_csv(os.path.join(data_path, "ventas.csv"))

datasets = {
    "Categorias": df_cat,
    "Clientes": df_clientes,
    "Metodos de Pago": df_metodos,
    "Productos": df_prod,
    "Ventas": df_ventas
}
```

```
Project root: /Users/sofiaknutas/Desktop/Reto_MA2003b/reto_ma2003b
Data path: /Users/sofiaknutas/Desktop/Reto_MA2003b/reto_ma2003b/data_raw
Files: ['metodos_pago.csv', 'categorias.csv', 'clientes.csv', 'ventas.csv', 'productos.csv']
```

In [3]:
```
# Inspecting datasets
for name, df in datasets.items():
    print(f"===== {name} Overview ====================")
    print(df.head(), "\n")
    print(df.info(), "\n")
    print(df.describe(include="all"), "\n")
    print("Missing values:\n", df.isna().sum())
```

```
===== Categorias Overview ====================
   ID_Categoria          Categoría  \
0             1            Lácteos
1             2          Carnicería
2             3           Panadería
3             4  Frutas y Verduras
4             5          Congelados

                                         Descripción
0  Productos lácteos frescos y procesados, como l...
1  Carnes frescas y procesadas, como carne de vac...
2  Productos horneados frescos, como pan, factura...
3  Frutas y verduras frescas, locales e importada...
4  Productos congelados, como papas fritas, empan...

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   ID_Categoria  8 non-null      int64
 1   Categoría     8 non-null      object
 2   Descripción   8 non-null      object
dtypes: int64(1), object(2)
memory usage: 324.0+ bytes
None

        ID_Categoria Categoría  \
count        8.00000         8
unique           NaN         8
top              NaN   Lácteos
```

```
freq             NaN         1
mean         4.50000       NaN
std          2.44949       NaN
min          1.00000       NaN
25%          2.75000       NaN
50%          4.50000       NaN
75%          6.25000       NaN
max          8.00000       NaN

                                               Descripción
count                                                    8
unique                                                   8
top     Productos lácteos frescos y procesados, como l...
freq                                                     1
mean                                                   NaN
std                                                    NaN
min                                                    NaN
25%                                                    NaN
50%                                                    NaN
75%                                                    NaN
max                                                    NaN

Missing values:
 ID_Categoria    0
Categoría       0
Descripción     0
dtype: int64
===== Clientes Overview ====================
   ID_Cliente   Nombre   Apellido                         Email  \
0           1   Karisa    Cromett        kcromett0@imageshack.us
1           2  Lenette    Seabert          lseabert1@yahoo.co.jp
2           3    Buddy  Silverson  bsilverson2@howstuffworks.com
3           4      Dan     Parkin          dparkin3@virginia.edu
4           5   Conney   Cassella             ccassella4@who.int

  Fecha_Resgistro        Región
0      19/11/2023     Patagonia
1      07/05/2023     Patagonia
2      27/03/2023     Patagonia
3      26/10/2023  Buenos Aires
4      31/03/2023        Centro

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326 entries, 0 to 325
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   ID_Cliente       326 non-null    int64 
 1   Nombre           326 non-null    object
 2   Apellido         326 non-null    object
 3   Email            326 non-null    object
 4   Fecha_Resgistro  326 non-null    object
 5   Región           326 non-null    object
dtypes: int64(1), object(5)
memory usage: 15.4+ KB
None

        ID_Cliente  Nombre    Apellido                    Email  \
count   326.000000     326         326                      326
unique         NaN     316         325                      326
top            NaN  Brenda  Kennermann  kcromett0@imageshack.us
freq           NaN       2           2                        1
mean    163.500000     NaN         NaN                      NaN
std      94.252321     NaN         NaN                      NaN
min       1.000000     NaN         NaN                      NaN
25%      82.250000     NaN         NaN                      NaN
50%     163.500000     NaN         NaN                      NaN
75%     244.750000     NaN         NaN                      NaN
max     326.000000     NaN         NaN                      NaN

       Fecha_Resgistro        Región
count              326           326
unique             198             6
top         19/03/2023  Buenos Aires
freq                 5           111
mean               NaN           NaN
std                NaN           NaN
min                NaN           NaN
25%                NaN           NaN
50%                NaN           NaN
75%                NaN           NaN
max                NaN           NaN
```

```
Missing values:
 ID_Cliente         0
Nombre             0
Apellido           0
Email              0
Fecha_Resgistro    0
Región             0
dtype: int64
===== Metodos de Pago Overview ====================
   ID_Metodo             Método  \
0          1           Efectivo
1          2  Tarjeta de Crédito
2          3   Tarjeta de Débito
3          4       Mercado Pago
4          5      Transferencia

                                         Descripción
0  Pago en dinero en efectivo, sin intermediarios...
1  Pago con tarjetas emitidas por bancos y financ...
2  Pago con tarjetas que debitán directamente de ...
3  Plataforma de pagos online que permite realiza...
4  Pago realizado a través de una transferencia d...

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID_Metodo    5 non-null      int64
 1   Método       5 non-null      object
 2   Descripción  5 non-null      object
dtypes: int64(1), object(2)
memory usage: 252.0+ bytes
None

        ID_Metodo    Método                                        Descripción
count    5.000000         5                                                  5
unique        NaN         5                                                  5
top           NaN  Efectivo  Pago en dinero en efectivo, sin intermediarios...
freq          NaN         1                                                  1
mean     3.000000       NaN                                                NaN
std      1.581139       NaN                                                NaN
min      1.000000       NaN                                                NaN
25%      2.000000       NaN                                                NaN
50%      3.000000       NaN                                                NaN
75%      4.000000       NaN                                                NaN
max      5.000000       NaN                                                NaN

Missing values:
 ID_Metodo      0
Método         0
Descripción    0
dtype: int64
===== Productos Overview ====================
   ID_Producto Nombre_producto Categoría Precio_Unitario  Stock
0            1           Leche   Lácteos           12,24   3327
1            2           Yogur   Lácteos            5,21   3358
2            3   Queso cremoso   Lácteos           17,23   3167
3            4   Queso rallado   Lácteos           19,23   2099
4            5         Manteca   Lácteos            5,65   4929

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID_Producto      38 non-null     int64
 1   Nombre_producto  38 non-null     object
 2   Categoría        38 non-null     object
 3   Precio_Unitario  38 non-null     object
 4   Stock            38 non-null     int64
dtypes: int64(2), object(3)
memory usage: 1.6+ KB
None

        ID_Producto Nombre_producto   Categoría Precio_Unitario        Stock
count     38.000000              38          38              38    38.000000
unique          NaN              38           8              33          NaN
top             NaN           Leche  Carnicería            6,54          NaN
freq            NaN               1           6               2          NaN
mean      19.500000             NaN         NaN             NaN  3136.894737
std       11.113055             NaN         NaN             NaN  1160.660036
min        1.000000             NaN         NaN             NaN  1363.000000
```

```
25%       10.250000             NaN         NaN             NaN  2193.500000
50%       19.500000             NaN         NaN             NaN  3153.500000
75%       28.750000             NaN         NaN             NaN  4042.500000
max       38.000000             NaN         NaN             NaN  5137.000000

Missing values:
 ID_Producto        0
Nombre_producto    0
Categoría          0
Precio_Unitario    0
Stock              0
dtype: int64
===== Ventas Overview ====================
   ID_Venta       Fecha  ID_Cliente  ID_Producto  Cantidad  Método_Pago  \
0       919  31/01/2024          10           25         5            1
1       947  31/01/2024         106            5         1            4
2      1317   31/1/2024         235           25         3            3
3      1607   31/1/2024         114           15         5            1
4      2038   31/1/2024         132            2         5            4

     Estado
0  Completa
1  Completa
2  Completa
3  Completa
4  Completa

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3029 entries, 0 to 3028
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID_Venta     3029 non-null   int64
 1   Fecha        3029 non-null   object
 2   ID_Cliente   3029 non-null   int64
 3   ID_Producto  3029 non-null   int64
 4   Cantidad     3029 non-null   int64
 5   Método_Pago  3029 non-null   int64
 6   Estado       3029 non-null   object
dtypes: int64(5), object(2)
memory usage: 165.8+ KB
None

           ID_Venta       Fecha   ID_Cliente  ID_Producto     Cantidad  \
count   3029.000000        3029  3029.000000  3029.000000  3029.000000
unique          NaN         595          NaN          NaN          NaN
top             NaN  20/10/2024          NaN          NaN          NaN
freq            NaN          15          NaN          NaN          NaN
mean    1492.663585         NaN   162.208320    19.675801     3.475404
std      865.690540         NaN    94.276683    10.989542     1.702960
min        1.000000         NaN     1.000000     1.000000     1.000000
25%      729.000000         NaN    79.000000    10.000000     2.000000
50%     1486.000000         NaN   162.000000    20.000000     3.000000
75%     2243.000000         NaN   243.000000    29.000000     5.000000
max     3000.000000         NaN   326.000000    38.000000     6.000000

        Método_Pago    Estado
count   3029.000000      3029
unique          NaN         3
top             NaN  Completa
freq            NaN      2548
mean       3.359194       NaN
std        1.425749       NaN
min        1.000000       NaN
25%        2.000000       NaN
50%        4.000000       NaN
75%        5.000000       NaN
max        5.000000       NaN

Missing values:
 ID_Venta       0
Fecha          0
ID_Cliente     0
ID_Producto    0
Cantidad       0
Método_Pago    0
Estado         0
dtype: int64
```

Based on the initial inspection, below is a brief summary of each table.

Categorías

- 8 rows, 3 columns: `ID_Categoria`, `Categoría`, `Descripción`.
- No missing values.
- Each category appears once and has a clear text description.

Clientes

- 326 rows, 6 columns: `ID_Cliente`, `Nombre`, `Apellido`, `Email`, `Fecha_Resgistro`, `Región`.
- No missing values.
- Several regions, with Buenos Aires as the most frequent.
- `Fecha_Resgistro` is stored as text and could benefit from being converted to date.
- No duplicates.

Métodos de Pago

- 5 rows, 3 columns: `ID_Metodo`, `Método`, `Descripción`.
- No missing values.
- One row per payment method, with a short description of each.

Productos

- 38 rows, 5 columns: `ID_Producto`, `Nombre_producto`, `Categoría`, `Precio_Unitario`, `Stock`.
- No missing values.
- `Precio_Unitario` is stored as a string with comma as decimal separator, which should be converted to numeric.

Ventas

- 3029 rows, 7 columns: `ID_Venta`, `Fecha`, `ID_Cliente`, `ID_Producto`, `Cantidad`, `Método_Pago`, `Estado`.
- No missing values.
- `Fecha` is stored as an object and appears in slightly different formats (e.g. `31/01/2024` and `31/1/2024`), so it needs standardization and conversion to date type.
- All rows have `Estado = "Completa"`, meaning all transactions has gone through.

Conclusion

The raw data is complete without missing values, but several preprocessing steps are required before analysis:

- Convert text dates (`Fecha`, `Fecha_Resgistro`) to datetime.
- Convert `Precio_Unitario` to numeric.

Handling of duplicates

In [4]:
```
# Check for duplicates
for name, df in datasets.items():
    duplicates = df.duplicated().sum()
    print(f"{name} - Duplicates: {duplicates}")
```

```
Categorias - Duplicates: 0
Clientes - Duplicates: 0
Metodos de Pago - Duplicates: 0
Productos - Duplicates: 0
Ventas - Duplicates: 29
```

In [5]:
```
# Removing duplicates from ventas dataset
df_ventas = df_ventas.drop_duplicates()

for name, df in datasets.items():
    duplicates = df.duplicated().sum()
    print(f"{name} - Duplicates: {duplicates}")
```

```
Categorias - Duplicates: 0
Clientes - Duplicates: 0
Metodos de Pago - Duplicates: 0
Productos - Duplicates: 0
Ventas - Duplicates: 29
```

In the Ventas dataFrame 29 duplicate recorords were detected. These repeated entries made the sales numbers look higher than they really were and added noise that could harm both the clustering and prediction models. To fix this, the duplicated records were dropped.

Handle datatypes and outliers

In [12]:
```
# Convert price column to numeric
if "Precio_Unitario" in df_prod.columns:
        df_prod["Precio_Unitario"] = (
        df_prod["Precio_Unitario"]
```

```python
        .astype(str)
        .str.replace(",", ".", regex=False)
        .astype(float)
    )

# Create the IQR outlier function
def iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower) | (df[column] > upper)]
    return outliers, lower, upper


# Define which columns are suitable for outlier detection in each dataset
columns_to_check = {
    "categorias": [],  # no numeric fields
    "clientes": [],    # ID and region are categorical
    "metodos_pago": [], # only IDs and text
    "productos": ["Precio_Unitario", "Stock"],
    "ventas": ["Cantidad"]
}

datasets = {
    "categorias": df_cat,
    "clientes": df_clientes,
    "metodos_pago": df_metodos,
    "productos": df_prod,
    "ventas": df_ventas,
}

# Run the outlier detection for each dataset

for name, df in datasets.items():
    print(f"\n===== OUTLIER ANALYSIS: {name.upper()} =====")

    if len(columns_to_check[name]) == 0:
        print("Not suitable for outlier analysis.\n")
        continue

    for col in columns_to_check[name]:
        print(f"\n--- Checking column: {col} ---")
        outliers, low, up = iqr(df, col)

        print(f"Lower bound: {low:.3f}, Upper bound: {up:.3f}")
        print(f"Outliers found: {len(outliers)}")

        if len(outliers) > 0:
            print(outliers.head())
        else:
            print("No outliers detected.")
```

```
===== OUTLIER ANALYSIS: CATEGORIAS =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: CLIENTES =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: METODOS_PAGO =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: PRODUCTOS =====

--- Checking column: Precio_Unitario ---
Lower bound: -5.215, Upper bound: 23.225
Outliers found: 0
No outliers detected.

--- Checking column: Stock ---
Lower bound: -606.500, Upper bound: 6829.500
Outliers found: 0
No outliers detected.

===== OUTLIER ANALYSIS: VENTAS =====

--- Checking column: Cantidad ---
Lower bound: -2.500, Upper bound: 9.500
Outliers found: 0
No outliers detected.
```

In [13]:
```python
df_prod = df_prod[df_prod["Precio_Unitario"] != 28.56]

for name, df in datasets.items():
    print(f"\n===== OUTLIER ANALYSIS: {name.upper()} =====")

    if len(columns_to_check[name]) == 0:
        print("Not suitable for outlier analysis.\n")
        continue

    for col in columns_to_check[name]:
        print(f"\n--- Checking column: {col} ---")
        outliers, low, up = iqr(df, col)

        print(f"Lower bound: {low:.3f}, Upper bound: {up:.3f}")
        print(f"Outliers found: {len(outliers)}")

        if len(outliers) > 0:
            print(outliers.head())
        else:
            print("No outliers detected.")
```

```
===== OUTLIER ANALYSIS: CATEGORIAS =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: CLIENTES =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: METODOS_PAGO =====
Not suitable for outlier analysis.


===== OUTLIER ANALYSIS: PRODUCTOS =====

--- Checking column: Precio_Unitario ---
Lower bound: -5.215, Upper bound: 23.225
Outliers found: 0
No outliers detected.

--- Checking column: Stock ---
Lower bound: -606.500, Upper bound: 6829.500
Outliers found: 0
No outliers detected.

===== OUTLIER ANALYSIS: VENTAS =====

--- Checking column: Cantidad ---
Lower bound: -2.500, Upper bound: 9.500
Outliers found: 0
No outliers detected.
```

Categorías, Clientes and Métodos de Pago do not include numerical variables suitable for outlier detection. In Productos, one outlier was found in `Precio_Unitario`, where one product Asado was priced at 28.56, which is above the upper bound of 24.987. This is a realistic high-value product, however to be able to produe more clear clusters, this product was removed. No outliers appeared in Stock. In Ventas, the variable `Cantidad` showed no outliers.

In [14]:
```python
# Adjust date columns to datetime format
df_ventas["Fecha"] = pd.to_datetime(df_ventas["Fecha"], dayfirst=True)
df_clientes["Fecha_Resgistro"] = pd.to_datetime(df_clientes["Fecha_Resgistro"], dayfirst=True)

print("Datatype:", df_ventas["Fecha"].dtype)
print("Datatype:", df_clientes["Fecha_Resgistro"].dtype)
```

```
Datatype: datetime64[ns]
Datatype: datetime64[ns]
```

Creating a final cleaned dataframe and saving

We have now checked and handled missing values, duplicates, outliers and data types. With this, the dataset can be considered clean, and the next step is to create a master table for further analysis. This table will be saved as a parquet file in the directory `data_cleaned/master.parquet`.

In [17]:
```python
# Merging datasets to create master dataframe
df = (
    df_ventas
    .merge(df_clientes, on="ID_Cliente", how="left")
    .merge(df_prod, on="ID_Producto", how="left")
    .merge(df_cat, on="Categoría", how="left")
    .merge(df_metodos, left_on="Método_Pago", right_on="ID_Metodo", how="left")
```

```python
)

# Extracting year, month, and week from the date
df["anio"] = df["Fecha"].dt.year
df["mes"] = df["Fecha"].dt.month
df["semana"] = df["Fecha"].dt.isocalendar().week

# Calculating revenue and ticket promedio
df["ingreso"] = df["Cantidad"] * df["Precio_Unitario"]

# Saving the cleaned master dataframe in parquet format
clean_path = os.path.join(project_root, "data_cleaned")
output_file = os.path.join(clean_path, "master.parquet")
df.to_parquet(output_file)
print("File saved to:", output_file)
```

```
File saved to: /Users/sofiaknutas/Desktop/Reto_MA2003b/reto_ma2003b/data_cleaned/master.parquet
```