# Projet SQL E-commerce

*Analyse des ventes et recommandations business*

## 1. Import des librairies

In [2]:
```
import pandas as pd
import sqlite3
```

## 2. Chargement des fichiers CSV

In [17]:
```
# Chemin à adapter à ton PC
customers = pd.read_csv(r"C:\Users\thoma\Desktop\Projet Data 2025\Projet 4\data\
products = pd.read_csv(r"C:\Users\thoma\Desktop\Projet Data 2025\Projet 4\data\p
orders = pd.read_csv(r"C:\Users\thoma\Desktop\Projet Data 2025\Projet 4\data\ord
```

## 3. Création de la base SQLite et insertion des données

In [18]:
```
conn = sqlite3.connect('ecommerce.db')
customers.to_sql('customers', conn, if_exists='replace', index=False)
products.to_sql('products', conn, if_exists='replace', index=False)
orders.to_sql('orders', conn, if_exists='replace', index=False)
```

Out[18]:
```
400
```

## 4. Vérification du chargement des données

In [7]:
```
print(customers.head())
print(products.head())
print(orders.head())
```

```
  customer_id customer_name     region signup_date
0    ae0c2c0c    Customer_0   Bordeaux  2023-11-30
1    bd78a9bf    Customer_1      Lille  2023-11-07
2    ea943533    Customer_2  Marseille  2023-06-14
3    90dd0796    Customer_3      Lille  2023-02-04
4    823e8cd5    Customer_4      Lille  2023-05-16
  product_id product_name     category  unit_price
0   d59e4097    Product_0     Clothing         113
1   e845b41a    Product_1         Home         263
2   0f827565    Product_2  Electronics         236
3   11ab5ae8    Product_3  Electronics         121
4   d814aefd    Product_4        Books         108
   order_id customer_id product_id  quantity  order_date  unit_price  \
0  8667eaab    13bd973e   d59e4097         4  2023-11-13         113
1  a3f79690    2bebdc40   51853bbd         4  2023-02-04         185
2  52e63739    09c29019   6db32e9b         4  2023-01-17         137
3  6b76d219    b2ebdf80   3b64e9bb         2  2023-02-15          36
4  19d04a7c    8fdcff9b   0f827565         4  2023-08-21         236

   total_price
0          452
1          740
2          548
3           72
4          944
```

# Analyses et requêtes SQL

## 5. Top 5 produits les plus vendus (en quantité)

In [10]:
```
# Top 5 produits les plus vendus (en quantité)
top_products = pd.read_sql_query("""
SELECT
    p.product_id,
    p.product_name,
    SUM(o.quantity) AS total_quantity
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_quantity DESC
LIMIT 5;
""", conn)
display(top_products)
```

| | product_id | product_name | total_quantity |
|---|---|---|---|
| **0** | a2349096 | Product_28 | 49 |
| **1** | 5f4337a3 | Product_22 | 48 |
| **2** | 51853bbd | Product_18 | 47 |
| **3** | 7b0a0689 | Product_25 | 47 |
| **4** | 6db32e9b | Product_16 | 45 |

On observe que **Product_28** est le best-seller avec 49 ventes, suivi de près par **Product_22** (48 ventes), etc.

# 6. Top 5 clients les plus rentables (CA total)

In [11]:
```
# Top 5 clients les plus rentables (CA total)
top_clients = pd.read_sql_query("""
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.quantity * o.unit_price) AS total_revenue
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_revenue DESC
LIMIT 5;
""", conn)
display(top_clients)
```

| | customer_id | customer_name | total_revenue |
|---|---|---|---|
| **0** | 823e8cd5 | Customer_4 | 4336 |
| **1** | 8fdcff9b | Customer_5 | 4175 |
| **2** | cdf0e1fc | Customer_35 | 4008 |
| **3** | 56b989c9 | Customer_68 | 3866 |
| **4** | 094442f1 | Customer_53 | 3784 |

**Customer_4** est le client ayant le plus dépensé (4336 €), suivi par **Customer_5** (4175 €), etc.
Ces clients sont stratégiques pour l'activité commerciale.

# 7. Chiffre d'affaires par mois

In [12]:
```
# Chiffre d'affaires total par mois
ca_mois = pd.read_sql_query("""
SELECT
    STRFTIME('%Y-%m', o.order_date) AS month,
    SUM(o.quantity * o.unit_price) AS total_revenue
FROM orders o
GROUP BY month
ORDER BY month;
""", conn)
display(ca_mois)
```

| | month | total_revenue |
|---|---|---|
| **0** | 2023-01 | 17621 |
| **1** | 2023-02 | 10643 |
| **2** | 2023-03 | 13261 |
| **3** | 2023-04 | 15980 |
| **4** | 2023-05 | 15572 |
| **5** | 2023-06 | 13784 |
| **6** | 2023-07 | 9611 |
| **7** | 2023-08 | 15636 |
| **8** | 2023-09 | 16715 |
| **9** | 2023-10 | 12592 |
| **10** | 2023-11 | 13201 |
| **11** | 2023-12 | 16545 |

CA mensuel de l'entreprise sur l'année 2023.
On remarque de fortes variations : janvier est le mois record (17 621 €), suivi de septembre (16 715 €) et décembre (16 545 €).
Ces pics de ventes peuvent correspondre à des périodes de promotions, fêtes ou événements spéciaux.

# 8. Nombre de commandes par région

In [13]:
```
# Nombre de commandes par région
cmd_region = pd.read_sql_query("""
SELECT
    c.region,
    COUNT(o.order_id) AS nb_orders
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.region
ORDER BY nb_orders DESC;
""", conn)
display(cmd_region)
```

| | region | nb_orders |
|---|---|---|
| **0** | Bordeaux | 94 |
| **1** | Lyon | 89 |
| **2** | Paris | 80 |
| **3** | Lille | 77 |
| **4** | Marseille | 60 |

**Bordeaux** est la région la plus dynamique avec 94 commandes, suivie par **Lyon** (89) et **Paris** (80).
Cela met en avant les régions les plus stratégiques pour l'activité.

## 9. Commandes avec incohérences (quantité = 0 ou client inexistant)

In [14]:
```
# Commandes avec incohérences (quantité = 0 ou client inexistant)
incoherences = pd.read_sql_query("""
SELECT *
FROM orders o
LEFT JOIN customers c ON o.customer_id = c.customer_id
WHERE o.quantity = 0 OR c.customer_id IS NULL;
""", conn)
display(incoherences)
```

| order_id | customer_id | product_id | quantity | order_date | unit_price | total_price | custom |
|---|---|---|---|---|---|---|---|

## 10. Panier moyen par client

In [15]:
```
# Panier moyen par client
panier_moyen = pd.read_sql_query("""
SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.quantity * o.unit_price) / COUNT(DISTINCT o.order_id) AS avg_basket
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY avg_basket DESC;
""", conn)
display(panier_moyen)
```

| | customer_id | customer_name | avg_basket |
|---|---|---|---|
| **0** | 1fab1c13 | Customer_21 | 763 |
| **1** | 11f4fc23 | Customer_75 | 755 |
| **2** | bd78a9bf | Customer_1 | 752 |
| **3** | 2fab5e0a | Customer_78 | 741 |
| **4** | b4e43028 | Customer_7 | 693 |
| **...** | ... | ... | ... |
| **90** | 0ccfd0a4 | Customer_65 | 167 |
| **91** | 77a6e7d9 | Customer_96 | 156 |
| **92** | 34398ff9 | Customer_69 | 154 |
| **93** | b8af3c52 | Customer_76 | 144 |
| **94** | 792b3b0d | Customer_66 | 113 |

95 rows × 3 columns

Les clients les plus "rentables" ont un panier moyen élevé (ex : **Customer_21** avec 763 €, **Customer_75** avec 755 €, etc).
Identifier ces clients permet d'adapter des stratégies marketing ciblées (fidélisation, offres spéciales...).

# 11. Évolution des ventes par catégorie de produit

In [16]:
```
# Évolution des ventes par catégorie de produit
evol_cat = pd.read_sql_query("""
SELECT
    STRFTIME('%Y-%m', o.order_date) AS month,
    p.category,
    SUM(o.quantity) AS total_quantity
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY month, p.category
ORDER BY month, p.category;
""", conn)
display(evol_cat)
```

| | month | category | total_quantity |
|---|---|---|---|
| **0** | 2023-01 | Beauty | 7 |
| **1** | 2023-01 | Books | 14 |
| **2** | 2023-01 | Clothing | 30 |
| **3** | 2023-01 | Electronics | 36 |
| **4** | 2023-01 | Home | 15 |
| **5** | 2023-02 | Beauty | 15 |
| **6** | 2023-02 | Books | 5 |
| **7** | 2023-02 | Clothing | 31 |
| **8** | 2023-02 | Electronics | 13 |
| **9** | 2023-02 | Home | 13 |
| **10** | 2023-03 | Beauty | 9 |
| **11** | 2023-03 | Books | 8 |
| **12** | 2023-03 | Clothing | 16 |
| **13** | 2023-03 | Electronics | 26 |
| **14** | 2023-03 | Home | 15 |
| **15** | 2023-04 | Beauty | 7 |
| **16** | 2023-04 | Books | 11 |
| **17** | 2023-04 | Clothing | 15 |
| **18** | 2023-04 | Electronics | 29 |
| **19** | 2023-04 | Home | 24 |
| **20** | 2023-05 | Beauty | 4 |
| **21** | 2023-05 | Books | 13 |
| **22** | 2023-05 | Clothing | 30 |
| **23** | 2023-05 | Electronics | 21 |
| **24** | 2023-05 | Home | 18 |
| **25** | 2023-06 | Beauty | 2 |
| **26** | 2023-06 | Books | 6 |
| **27** | 2023-06 | Clothing | 9 |
| **28** | 2023-06 | Electronics | 19 |
| **29** | 2023-06 | Home | 30 |
| **30** | 2023-07 | Beauty | 1 |
| **31** | 2023-07 | Books | 4 |
| **32** | 2023-07 | Clothing | 17 |

| | month | category | total_quantity |
|---|---|---|---|
| **33** | 2023-07 | Electronics | 17 |
| **34** | 2023-07 | Home | 12 |
| **35** | 2023-08 | Beauty | 10 |
| **36** | 2023-08 | Books | 1 |
| **37** | 2023-08 | Clothing | 27 |
| **38** | 2023-08 | Electronics | 31 |
| **39** | 2023-08 | Home | 21 |
| **40** | 2023-09 | Beauty | 6 |
| **41** | 2023-09 | Books | 9 |
| **42** | 2023-09 | Clothing | 15 |
| **43** | 2023-09 | Electronics | 33 |
| **44** | 2023-09 | Home | 22 |
| **45** | 2023-10 | Beauty | 2 |
| **46** | 2023-10 | Books | 15 |
| **47** | 2023-10 | Clothing | 19 |
| **48** | 2023-10 | Electronics | 12 |
| **49** | 2023-10 | Home | 30 |
| **50** | 2023-11 | Clothing | 22 |
| **51** | 2023-11 | Electronics | 24 |
| **52** | 2023-11 | Home | 29 |
| **53** | 2023-12 | Beauty | 9 |
| **54** | 2023-12 | Books | 10 |
| **55** | 2023-12 | Clothing | 24 |
| **56** | 2023-12 | Electronics | 20 |
| **57** | 2023-12 | Home | 29 |

On observe que la catégorie **"Electronics"** domine régulièrement, avec des pics notables certains mois.
Les catégories **"Clothing"** et **"Home"** sont également bien représentées, tandis que **"Beauty"** et **"Books"** restent plus stables.
Cette analyse permet d'orienter les efforts marketing par catégorie au fil de l'année.