PROJET 1 : Retail — "DataLendo"

Contexte Entreprise

Vous venez d'être engagé comme **Analyste de Données** chez **DataLendo**, une entreprise qui vend des **produits électroniques** (smartphones, laptops, TV, accessoires) à travers :

- un **site e-commerce**
- des **boutiques physiques**

Les clients viennent majoritairement de pays **francophones** : 🇧🇪 Belgique, 🇨🇦 Canada (Québec), 🇫🇷 France, 🇨🇩 RD Congo, 🇨🇲 Cameroun, 🇨🇮 Côte d'Ivoire, 🇸🇳 Sénégal, 🇬🇦 Gabon et quelques autres pays francophones occasionnels.

DataLendo a une croissance rapide mais manque de **visibilité analytique** sur : son **CA (chiffre d'affaires),** ses **clients,** ses **produits,** ses **régions et** ses **segments.**

Mission reçue par e-mail

Objet : Besoin d'analyse ventes Q2 + segmentation clients

Bonjour,

Bienvenue dans l'équipe Data de DataLendo !

Nous travaillons sur un projet stratégique pour le département Retail concernant nos performances commerciales sur l'année écoulée. La Direction souhaite mieux comprendre nos ventes, nos produits et nos clients, ainsi que les indicateurs clés liés au paiement et au revenu.

Ta mission :

1. Explorer et nettoyer les données
2. Identifier les indicateurs clés (KPIs Retail)
3. Segmenter nos clients (RFM simplifié)
4. Analyser la performance produit
5. Formuler des insights business actionnables
6. Préparer une synthèse pour le comité Direction

Tu trouveras en pièces jointes les datasets suivants :

clients.csv

paiements.csv

produits.csv

ventes.csv

Merci de revenir vers nous d'ici vendredi avec un rapport analytique structuré, incluant les visualisations nécessaires, les insights et des recommandations business.

Deadline : dans 72h.

Merci,
Manager Data
DataLendo

Comment on va aborder le projet (Jour 4 Workflow)

Comme un vrai Data Analyst, avec logique + narration + exécution + business :

1. **Introduction + Contexte + Mission**
2. **Exploration Dataset**
3. **Nettoyage & préparation**
4. **KPIs Retail**
5. **Segmentation Clients**
6. **Analyse Produits**
7. **RFM simplifié**
8. **Insights Business**
9. **Correction + bonnes pratiques Data**
10. **Versioning GitHub** (Livrable + Notebook + README + dataset)

Section 1 - 15 Questions Business Demandées par le Manager

Ces questions guident l'analyse (KPIs, produits, clients, paiements, insights) :

Bloc KPIs / Performance Commerciale

1. Quel est le chiffre d'affaires total sur la période ?

2. Quelle est l’évolution mensuelle des ventes ?
3. Quel jour / mois est le meilleur en termes de revenu ?
4. Quel moyen de paiement est le plus utilisé et lequel rapporte le plus ?

Bloc Produits

5. Quels sont les produits Best Sellers (en volume) ?
6. Quels sont les produits Top Revenue (en valeur) ?
7. Quels produits se vendent mal ou sont en fin de cycle ?
8. Quel est le panier moyen par produit ?

Bloc Clients

9. Quels segments de clients génèrent le plus de revenus ?
10. Comment se répartissent les achats par localisation (pays/ville) ?
11. Quel est l’âge moyen des clients acheteurs ?

Bloc RFM simplifié

12. Quels sont les clients les plus récents (R) ?
13. Quels clients achètent le plus fréquemment (F) ?
14. Quels clients dépensent le plus (M) ?

Bloc Insights / Business

15. Sur la base de l’analyse, quelles actions recommandes-tu à la Direction ?

SECTION 2 — EXPLORATION DES DONNÉES (SQL Analyst Mindset)

L’objectif de l’exploration est de répondre à 4 questions simples, avant tout KPI :

1. **Qu’est-ce que j’ai comme données ?** (tables + relations)
2. **Quelle est la qualité des données ?**

3. **Quels volumes ?** (granularité, taille, dates)
4. **De quoi ai-je besoin pour répondre aux questions business ?**

2.1 — Inventaire des tables

On liste mentalement les fichiers fournis + leur fonction :

Table	Rôle
clients	infos clients (démographie + localisation)
produits	catalogue (catégorie + prix)
ventes	transactions (grain = ligne de vente)
paiements	encaissement + moyen de paiement

➡️ Important pour l'apprenant : comprendre le **grain** (= niveau de détail), car c'est une notion clé en SQL.

2.2 — Vérification des relations (PK/FK)

En projet réel, c'est toujours un des premiers réflexes.

Relations attendues :

```
clients.client_id = ventes.client_id
produits.produit_id = ventes.produit_id
ventes.vente_id = paiements.vente_id
```

➡️ Ce mapping permet de remonter du **paiement** → **vente** → **client / produit**

2.3 — Exploration SQL

Ici on apprend les commandes "exploratoires" que tout Junior devrait maîtriser.

a. Voir les 10 premières lignes (sanity check)

Pour `clients` :

```
SELECT * FROM clients LIMIT 10;
```

Pareil pour `produits`, `ventes`, `paiements`.

But :

- détecter colonnes évidentes
- structure
- types implicites
- anomalies visibles

b. Volume des tables

```
SELECT COUNT(*) FROM clients;
SELECT COUNT(*) FROM produits;
SELECT COUNT(*) FROM ventes;
SELECT COUNT(*) FROM paiements;
```

Ce check est critique car :

- retail = beaucoup d'évènements (ventes/paiements >> clients/produits)

c. Structure des colonnes

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'ventes';
```

Intérêt pédagogique :
➡️ introduit `information_schema`

d. Vérifier les dates disponibles

```
SELECT MIN(date_vente), MAX(date_vente)
FROM ventes;
```

Cela permet de :

✔ définir la période couverte
✔ estimer saisonnalité
✔ préparer KPIs temporels

e. Vérifier les doublons PK potentiels

```
SELECT vente_id, COUNT(*)
FROM ventes
GROUP BY vente_id
HAVING COUNT(*) > 1;
```

Ce genre de contrôle montre la **qualité data**.

2.4 — Premières observations analytiques

Quand un analyst fait l'exploration, il commence déjà à "penser KPI".

Dans Retail typique :

- `ventes` = transactionnel → base des KPIs
- `paiements` = cashflow → moyens de paiement
- `produits` = pricing + catégorisation → segmentation produit
- `clients` = segmentation → demographie + RFM

On connecte ça à la mission du manager (jour 1).

2.5 — Ce que vous devez retenir

➤ **Le mindset Data Analyst SQL**

Avant d'analyser, je comprends mes données.

➤ **La notion de grain**

`ventes` = par article vendu
`paiements` = par vente encaissée

➢ **Les relations business**

pas de KPI sans JOIN

Transition Section 3 — Nettoyage

Dans Retail, nettoyage = principalement :

✔ types + dates
✔ null + manquants
✔ cohérence PK/FK
✔ cohérence prix/quantité/montant
✔ cohérence paiement vs vente

C'est ce qu'on va aborder maintenant.

SECTION 3 — NETTOYAGE & PRÉPARATION (SQL Data Cleaning)

Dans un vrai job Data, **avant tous les KPIs**, on doit garantir :

✔ cohérence des données
✔ qualité des dates
✔ qualité des clés
✔ qualité des montants
✔ absence de doublons
✔ relations JOINable
✔ granularité claire

Dans Retail E-commerce, ce sont les aspects critiques.

3.1 Validation des clés PK/FK

a. PK uniques

```
SELECT vente_id, COUNT(*)
FROM ventes
GROUP BY vente_id
HAVING COUNT(*) > 1;
```

C'est **la première barrière** : si PK est cassé, tout explose.

b. FK référentielles

On vérifie que `ventes.client_id` existe dans `clients`

```
SELECT v.client_id
FROM ventes v
LEFT JOIN clients c ON v.client_id = c.client_id
WHERE c.client_id IS NULL;
```

Même logique pour `produits` et `paiements`.

Ces checks répondent à :

> "Peut-on JOIN ?"

3.2 Vérification des Dates

Retail = **temporal analytics**, donc dates = critique.

```
SELECT
    MIN(date_vente),
    MAX(date_vente)
FROM ventes;
```

Points à vérifier :

✔ présence de dates
✔ période couverte
✔ anomalies (future, trop anciennes)

Rapport analyste typique :

> “Dataset couvre Jan 2023 → Déc 2023 (1 an complet). Cohérent.”

3.3 Vérification Montants / Quantités

Retail exige cohérence :

```
montant = quantité * prix_unitaire
```

On valide :

```
SELECT *
FROM ventes
WHERE montant <> quantite * prix_unitaire;
```

S’il y a des erreurs :

✔ arrondi
✔ taxe
✔ remise
✔ erreur systeme

On introduit une notion réaliste de Data Cleaning :

> Tous les datasets Retail contiennent des cas marginalement incohérents.

3.4 Vérifier Nulls Critiques

Cas critique : un `client_id NULL` dans `ventes` = transaction impossible à attribuer.

```
SELECT COUNT(*) AS ventes_sans_client
FROM ventes
```

```
WHERE client_id IS NULL;
```

Même chose sur `produits`.

Impact business :

➜ segmentation impossible
➜ RFM impossible
➜ panier moyen distordu

3.5 Vérifier Paiement vs Vente

Très important car retail = cashflow

1 vente → 1 paiement (chez nous)
ou **n paiements** (dans monde réel : split, crédit, etc.)

On contrôle :

```
SELECT vente_id, COUNT(*) AS n
FROM paiements
GROUP BY vente_id
HAVING COUNT(*) > 1;
```

et ventes sans paiement :

```
SELECT v.vente_id
FROM ventes v
LEFT JOIN paiements p ON v.vente_id = p.vente_id
WHERE p.vente_id IS NULL;
```

3.6 Détection Doublons de ventes

Doublons transactionnels = piège retail classique.

3.7 Uniformisation des champs

Exemples utiles :

- minuscules/Majuscules clients
- normalisation catégories produits
- pays/ville standard

ex manipulation textuelle :

```
UPDATE clients
SET pays = INITCAP(pays);
```

3.8 Renommage colonnes (selon standard BI)

Pourquoi ?

✔ lisibilité
✔ compatibilité KPIs
✔ standardisation métiers

Exemple :

`date_vente` → `date`
`montant` → `revenue`

3.9 Enrichissements simples

On peut préparer des colonnes analytiques

Ex :

```
ALTER TABLE ventes
ADD COLUMN annee INT GENERATED ALWAYS AS (EXTRACT(YEAR FROM
date_vente)) STORED;
```

ou `mois`, `jour_semaine`

En retail c'est essentiel.

3.10 Préparation pour les JOINS

On prépare le futur JOIN universel :

```
SELECT *
FROM ventes v
JOIN clients c ON v.client_id = c.client_id
JOIN produits p ON v.produit_id = p.produit_id
JOIN paiements pay ON v.vente_id = pay.vente_id;
```

Avant KPI, l'analyste doit garantir :

> "Je peux JOIN, j'ai des dates cohérentes, des montants cohérents et du cashflow présent."

On prépare ici le **jeu d'analyse complet**.

On va maintenant passer à :

✔ Revenue
✔ Panier moyen
✔ Fréquence d'achat
✔ GMV
✔ Contribution produit

✔ Volume
✔ Marge (si dispo)

SECTION 4 — CALCUL & INTERPRETATION DES KPIs RETAIL

4.1 Liste des KPIs Retail Essentiels

Nous allons calculer :

✔ Revenue (CA / GMV)
✔ Nombre de transactions
✔ Nombre de clients actifs
✔ Panier moyen (AOV)
✔ Fréquence d'achat
✔ Lifetime Value (préliminaire)
✔ Revenue par pays
✔ Revenue par catégorie
✔ Top produits (quantité + revenue)
✔ Take rate / % paiement succès
✔ Taux de refund (si existait)
✔ Mix produits
✔ Mix géographique
✔ Mix clientèle

On garde **réaliste mais pas surchargé**.

Table centrale

Pour rappel **table ventes = table centrale**

Parce que retail analytics = **event → revenue**

4.2 KPI 1 — Revenue global (GMV)

GMV = Gross Merchandise Value = somme des ventes

```
SELECT SUM(montant) AS revenue_global
FROM ventes;
```

Interprétation Analyste :

> “Datalendo a généré 2.4M€ de GMV en 2023.”

C’est le KPI roi.

4.3 KPI 2 — Nombre de transactions

```
SELECT COUNT(*) AS nombre_transactions
FROM ventes;
```

Utilité business :

✔ volume
✔ activité
✔ conversion

👥 4.4 KPI 3 — Clients actifs

Client actif = client ayant acheté ≥ 1 fois sur période

```
SELECT COUNT(DISTINCT client_id) AS clients_actifs
FROM ventes;
```

Couplé avec `clients`, on peut faire :

```
SELECT
```

```
    (COUNT(DISTINCT client_id) * 100.0 / (SELECT COUNT(*) FROM
clients)) AS taux_activation
FROM ventes;
```

KPIs très utilisés en CRM/Marketing.

4.5 KPI 4 — Panier Moyen (AOV / Average Order Value)

```
SELECT
    SUM(montant) / COUNT(*) AS panier_moyen
FROM ventes;
```

Interprétation :

> “Les clients dépensent en moyenne 87€ par transaction.”

Si le panier moyen augmente → stratégie pricing/merchandising réussie.

🔁 4.6 KPI 5 — Fréquence d’Achat

Approche simple pour un premier projet :

```
SELECT
    client_id,
    COUNT(*) AS freq_achat
FROM ventes
GROUP BY client_id;
```

Insight marketing :

✔ upsell
✔ fidélité
✔ segment VIP

4.7 KPI 6 — Revenue par pays

Dans retail, **géographie = segment puissant**

```
SELECT c.pays, SUM(v.montant) AS revenue
FROM ventes v
JOIN clients c USING (client_id)
GROUP BY c.pays
ORDER BY revenue DESC;
```

Dans notre dataset, attendu :

🇫🇷 🇨🇦 🇧🇪 + Afrique francophone

Ce KPI dit **où pousser l'acquisition**

4.8 KPI 7 — Revenue par catégorie produits

```
SELECT p.categorie, SUM(v.montant) AS revenue
FROM ventes v
JOIN produits p USING (produit_id)
GROUP BY p.categorie
ORDER BY revenue DESC;
```

Insight typique :

> "Smartphones + Laptops = 65% du revenue → focus merchandising + stock"

4.9 KPI 8 — Top 10 produits (Revenue & Quantité)

```
SELECT p.nom_produit,
       SUM(v.quantite) AS qty,
       SUM(v.montant) AS revenue
FROM ventes v
JOIN produits p USING (produit_id)
GROUP BY p.nom_produit
ORDER BY revenue DESC
LIMIT 10;
```

Très utilisé par :

✔ équipes produit
✔ pricing
✔ stock/supply chain

4.10 KPI 9 — Paiement success rate

Retail = paiement = nerf de la guerre

```
SELECT
    SUM(CASE WHEN statut='success' THEN 1 ELSE 0 END) * 100.0 /
COUNT(*) AS taux_succes
FROM paiements;
```

Dans le monde réel :

➜ Stripe, PayPal, Mobile Money
➜ problème pays vs méthode
➜ abandons panier

4.11 KPI 10 — Mix Produit / Mix Client / Mix Geo

Le mix dit : **qu'est-ce qui compose le CA**

Ex mix géographique :

```
SELECT pays,
       SUM(montant) * 100.0 / (SELECT SUM(montant) FROM ventes) AS
pct_mix
FROM ventes v
JOIN clients USING (client_id)
GROUP BY pays;
```

Interprétation et Narration Analyste

Pour ce projet, après chaque bloc SQL :

➜ KPI → Insight → Action

Example narration :

> "La France représente 42% du CA, suivie du Canada (21%).
> Opportunité : cibler Cote d'Ivoire + Sénégal avec promos mobiles."

C'est **exactement** ce qu'on demande en entreprise.

Maintenant qu'on a :

✔ revenue
✔ fréquence
✔ activité

On va segmenter les clients.

C'est ce qui permet :

➜ RFM
➜ VIP
➜ dormant
➜ churn risk
➜ marketing ciblé

SECTION 5 — SEGMENTATION CLIENT & SCORING

Objectif : transformer des **KPIs clients** en **segments exploitables** & **insights marketing / business**

Pourquoi c'est crucial ?

✔ aucun business retail ne traite tous les clients pareil
✔ acquisition ≠ rétention ≠ upsell
✔ le marketing/clustering = ROI direct
✔ SQL est excellent pour ça

5.1 Rappel Business (Rapide mais Fondamental)

Dans retail / e-commerce :

➡ "Clients" = **portefeuille**
➡ "Segments" = **stratégies**

Cas réels :

- VIP → programme fidélité
- Low buyers → promos
- Dormants → réactivation
- High Frequency → cross-sell
- High Basket → premium offers

5.2 Construction du Dataset Client

On prend la table **ventes (event)** et on l'agrège par **client**

```
CREATE VIEW client_metrics AS
SELECT
    client_id,
    COUNT(*) AS transactions,
    SUM(montant) AS revenue,
    SUM(montant) / COUNT(*) AS panier_moyen,
    MAX(date_vente) AS derniere_vente,
    MIN(date_vente) AS premiere_vente
FROM ventes
GROUP BY client_id;
```

Ce view nous servira tout au long du jour.

5.3 Extraction de la Fréquence

Très utilisé en CRM :

```
SELECT client_id, transactions
FROM client_metrics;
```

5.4 Extraction du panier moyen

```
SELECT client_id, panier_moyen
FROM client_metrics;
```

5.5 Revenue Client (LTV simplifié)

LTV = Lifetime Value (simplifié car sans marge/coût)

```
SELECT client_id, revenue
FROM client_metrics;
```

5.6 Normalisation des Variables (Business Thinking)

Avant de scorer → on se pose les bonnes questions :

➡ Quel KPI distingue vraiment les clients ?
➡ Quel KPI est actionnable ?
➡ Quel KPI est stable ?

Exemple typique :

- transactions → fidélité
- revenue → valeur
- panier → pouvoir d'achat

5.7 Construction Scoring Simple (3 niveaux)

On fait un **3-tier segmentation**, très classique retail :

KPI	seuil bas	moyen	haut
transactions	1-2	3-5	>5
panier moyen	<50	50-150	>150
revenue	<200	200-500	>500

(Pure provocation business, à adapter selon dataset IRL)

Implémentation SQL (CASE) :

```
SELECT
    client_id,
    CASE
        WHEN transactions > 5 THEN 'Fidèle'
        WHEN transactions BETWEEN 3 AND 5 THEN 'Occasionnel'
        ELSE 'Rare'
    END AS segment_freq,
    CASE
        WHEN panier_moyen > 150 THEN 'Premium'
        WHEN panier_moyen BETWEEN 50 AND 150 THEN 'Standard'
        ELSE 'Bas'
    END AS segment_panier,
    CASE
        WHEN revenue > 500 THEN 'High Value'
        WHEN revenue BETWEEN 200 AND 500 THEN 'Medium Value'
        ELSE 'Low Value'
    END AS segment_value
FROM client_metrics;
```

5.8 Création Segmentation Mixée

On combine :

➡ fidélité + panier + valeur

```
SELECT client_id,
       CONCAT(segment_freq, '_', segment_value) AS segment_mix
FROM (... vue précédente ...)
```

Segments typiques obtenus :

- Fidèle_HighValue (VIP)
- Fidèle_MediumValue
- Occasionnel_HighValue
- Rare_LowValue

- etc.

5.9 Mapping Business

Dans une boîte retail réelle, ça sert à :

✔ Prioriser promos
✔ Programmes VIP
✔ Réactivation
✔ Cross-sell produit
✔ Pricing dynamique

Exemple décision :

> “Fidèle_HighValue → envoyer offres premium + early access”

5.10 Résultat & Narration Analyste

Ce qu’on veut produire côté apprentissage :

```
40% clients = Rare_LowValue
25% = Occasionnel_Standard
10% = Fidèle_HighValue (groupe extrêmement rentable)
```

Narration business :

> “10% des clients génèrent 45% du revenue → churn sur cette population = risque majeur.”

SECTION 6 — ANALYSE PRODUIT & TOP VENTES

Objectif du jour :
Passer d'une vision **Client** → **Produit**, ce que font 100% des boîtes e-commerce, retail et marketplace.

Car dans le business, les bonnes décisions sont souvent sur les produits :

✔ ce qu'il faut promouvoir
✔ ce qu'il faut déstocker
✔ ce qu'il faut arrêter
✔ ce qu'il faut re-pricer
✔ ce qu'il faut acheter plus

6.1 Structure Produit (Rappel)

On repart du dataset :

produits

- id_produit
- nom
- categorie
- prix
- stock

ventes

- produit_id
- quantite
- montant
- date

Lien clé : `ventes.produit_id → produits.id_produit`

6.2 Exploration Produit (SQL de base)

Toujours commencer simple :

```
SELECT *
FROM produits
LIMIT 10;
```

Puis :

```
SELECT categorie, COUNT(*) AS nb_produits
FROM produits
GROUP BY categorie;
```

Insight typique : volume par catégorie.

6.3 Top Ventes (quantité)

Première métrique classique retail :

```
SELECT p.nom, p.categorie,
       SUM(v.quantite) AS quantite_totale
FROM ventes v
JOIN produits p ON v.produit_id = p.id_produit
GROUP BY p.nom, p.categorie
ORDER BY quantite_totale DESC
LIMIT 10;
```

KPIs extraits :

✔ top sellers
✔ catégories dominantes
✔ effet saisonnier futur (jour 7)

6.4 Top Revenue (CA produit)

Plus concret pour le business :

```
SELECT p.nom, p.categorie,
       SUM(v.montant) AS revenue
FROM ventes v
JOIN produits p ON v.produit_id = p.id_produit
GROUP BY p.nom, p.categorie
ORDER BY revenue DESC
LIMIT 10;
```

Ici on identifie les produits **rentables** et pas seulement vendus.

6.5 Panier Produit (rev / qty)

On reconstruit un pseudo panier moyen par produit :

```
SELECT p.nom,
       SUM(v.montant) / SUM(v.quantite) AS prix_effectif
FROM ventes v
JOIN produits p ON v.produit_id = p.id_produit
GROUP BY p.nom;
```

Utilité : voir si prix catalogue ≠ prix réellement payé
(retour promo, bundle etc.)

6.6 Analyse par Catégorie

Très important en Retail Analytics :

```
SELECT p.categorie,
       SUM(v.quantite) AS quantite_totale,
       SUM(v.montant) AS revenue_total
FROM ventes v
JOIN produits p ON v.produit_id = p.id_produit
GROUP BY p.categorie
```

```
ORDER BY revenue_total DESC;
```

Typiquement → décision assortiments & marge.

6.7 Stock vs Vente (mini Supply Chain)

Retail = toujours TENSION **stock ↔ vente**

```
SELECT p.nom, p.stock,
       SUM(v.quantite) AS ventes
FROM produits p
LEFT JOIN ventes v ON p.id_produit = v.produit_id
GROUP BY p.nom, p.stock
ORDER BY stock ASC;
```

Usage Business :

✔ détecter ruptures potentielles
✔ détecter surstock
✔ déstocker bas ventes

6.8 Produits Dormants (no sales)

En retail réel c'est un enfer coûteux :

```
SELECT p.nom, p.categorie, p.stock
FROM produits p
LEFT JOIN ventes v ON p.id_produit = v.produit_id
WHERE v.produit_id IS NULL;
```

Décisions :

✔ déstockage
✔ promotions
✔ arrêt produit

6.9 Matrice Produit (insight simple)

On combine 2 axes :

➡ Ventes (qty)
➡ Revenue (CA)

```
- High qty / High CA  → Champions (A garder + pousser)
- Low qty / High CA   → Premium (pricing sensible)
- High qty / Low CA   → Low margin (à optimiser)
- Low qty / Low CA    → Suppression / déstock
```

Ce quadrant est ultra business, appliqué partout (Zara, Amazon, Carrefour…).

6.10 Narration Analyste

Ce qu'un bon analyste dirait :

> "Les laptops génèrent 42% du CA mais seulement 12% du volume — segment premium, sensible au pricing"

Ou :

> "Les écouteurs dominent en volume, marge faible — utiles en acquisition"

Maintenant qu'on sait :

✔ Qui achète (Section 5)
✔ Quoi ils achètent (Section 6)

On passe à :

➡ RFM simplifié = **fréquence + récence + valeur**

C'est une des analyses les plus utilisées dans le commerce moderne (e-commerce, SaaS, retail, marketplaces, etc.)

SECTION 7 — RFM SIMPLIFIÉE & INSIGHTS AVANCÉS

Objectif :
Passer d'une vision produit et client séparée à une **analyse combinée client** qui permet de segmenter, prioriser et agir, exactement comme en entreprise.

RFM = **Recency (récence), Frequency (fréquence), Monetary (montant)**

7.1 Rappel données

Datasets déjà préparés :

- **clients** : id_client, nom, pays, date_inscription
- **ventes** : id_vente, client_id, produit_id, quantite, montant, date
- **produits** : id_produit, nom, categorie, prix, stock

Table centrale pour RFM : **ventes** → on relie à **clients**.

7.2 Calcul Récence

Récence = combien de jours depuis la dernière commande d'un client

```
SELECT c.id_client,
       c.nom,
       MAX(v.date) AS derniere_vente,
       CURRENT_DATE - MAX(v.date) AS jours_depuis_derniere_vente
FROM clients c
JOIN ventes v ON c.id_client = v.client_id
GROUP BY c.id_client, c.nom
ORDER BY jours_depuis_derniere_vente;
```

💡 Insight :

- Plus le nombre de jours est petit → client actif
- Plus il est grand → client dormant

7.3 Calcul Fréquence

Fréquence = combien de commandes dans une période

```
SELECT c.id_client,
       c.nom,
       COUNT(v.id_vente) AS nb_commandes
FROM clients c
JOIN ventes v ON c.id_client = v.client_id
GROUP BY c.id_client, c.nom
ORDER BY nb_commandes DESC;
```

💡 Insight :

- Clients fidèles = fréquence élevée
- Clients occasionnels = fréquence faible

7.4 Calcul Montant (Monetary)

Montant = total dépensé par client

```
SELECT c.id_client,
       c.nom,
       SUM(v.montant) AS total_depense
FROM clients c
JOIN ventes v ON c.id_client = v.client_id
GROUP BY c.id_client, c.nom
ORDER BY total_depense DESC;
```

💡 Insight :

- Top clients en valeur réelle → focus marketing / fidélisation
- Petit montant → upsell / cross-sell

7.5 Combinaison RFM

On combine Récence + Fréquence + Montant pour segmenter :

```
WITH rfm AS (
    SELECT c.id_client,
           c.nom,
           CURRENT_DATE - MAX(v.date) AS recence,
           COUNT(v.id_vente) AS frequence,
           SUM(v.montant) AS monetary
    FROM clients c
    JOIN ventes v ON c.id_client = v.client_id
    GROUP BY c.id_client, c.nom
)
SELECT *,
       CASE
           WHEN recence <= 30 AND frequence >= 5 AND monetary >= 500
THEN 'Top Client'
           WHEN recence <= 60 AND frequence >= 3 AND monetary >= 200
THEN 'Client Régulier'
           WHEN recence > 60 AND frequence < 3 THEN 'Client Dormant'
           ELSE 'Client Moyen'
       END AS segment
FROM rfm
ORDER BY monetary DESC;
```

💡 Insight business :

- **Top Client** → fidélisation / premium
- **Client Régulier** → maintenance / up-sell
- **Client Dormant** → réactivation par campagne email / promo

7.6 Analyse Avancée / Cross Insights

Combiner avec produit ou catégorie :

```
SELECT p.categorie,
       r.segment,
       COUNT(v.id_vente) AS ventes,
       SUM(v.montant) AS revenue
FROM ventes v
JOIN produits p ON v.produit_id = p.id_produit
JOIN (
    -- RFM calculé ci-dessus
    SELECT c.id_client,
           CASE
               WHEN CURRENT_DATE - MAX(v.date) <= 30 AND
COUNT(v.id_vente) >= 5 AND SUM(v.montant) >= 500 THEN 'Top Client'
               WHEN CURRENT_DATE - MAX(v.date) <= 60 AND
COUNT(v.id_vente) >= 3 AND SUM(v.montant) >= 200 THEN 'Client
Régulier'
               WHEN CURRENT_DATE - MAX(v.date) > 60 AND
COUNT(v.id_vente) < 3 THEN 'Client Dormant'
               ELSE 'Client Moyen'
           END AS segment
    FROM clients c
    JOIN ventes v ON c.id_client = v.client_id
    GROUP BY c.id_client
) r ON v.client_id = r.id_client
GROUP BY p.categorie, r.segment
ORDER BY revenue DESC;
```

💡 Insight :

- Quels segments achètent quelles catégories
- Priorité marketing et promotion
- Ajustement des stocks selon le segment

7.7 Narration Analyste

Exemples de phrases business :

- “Les **Top Clients** achètent majoritairement des **laptops et smartphones**, générant 45% du CA”
- “Les **Clients Dormants** représentent 20% du fichier mais moins de 5% du CA, campagne de réactivation nécessaire”
- “Les **Clients Moyens** achètent surtout accessoires et bundles → opportunité upsell”

Maintenant que nous savons :

✔ qui achète (RFM)
✔ quoi ils achètent (produits / top ventes)
✔ segment client

On est prêt pour **SECTION 8 — Présentation Business & Recommandations**, où l’on va **mettre tout ensemble**, produire **insights clairs et actionnables**, exactement comme un **analyste professionnel**.

SECTION 8 — PRÉSENTATION BUSINESS & RECOMMANDATIONS

Objectif :
Transformer toutes les analyses SQL réalisées jusque-là en **insights exploitables** pour le business. On va produire un **résumé clair, structuré et argumenté** comme le ferait un Data Analyst dans une entreprise réelle.

8.1 Synthèse des KPIs

À partir des sections précédentes (KPIs globaux, segmentation clients, analyse produits, RFM), on résume :

KPI	Résultat / Insight
CA total	1 245 000 €
CA par pays	France : 450 000 €, Belgique : 180 000 €, Cameroon : 120 000 €, etc.
Clients actifs	4 500 clients sur 6 000 inscrits
Top produits	Laptop X200, Smartphone Z10, Casque Audio Q5
Segment Top Client	350 clients (7,8% total) génèrent 45% du CA
Segment Client Dormant	1 200 clients → priorité réactivation

💡 Conseil : toujours présenter les KPIs avec **chiffres clairs**, **tableaux ou graphiques** simples.

8.2 Recommandations clients

1. **Top Clients**
    - Stratégie : fidélisation premium
    - Actions : offres exclusives, early access produits, programmes VIP
2. **Clients Réguliers**
    - Stratégie : maintenance & upsell
    - Actions : campagnes ciblées, bundle promotions
3. **Clients Dormants**
    - Stratégie : réactivation
    - Actions : emails personnalisés, coupons, relances SMS
4. **Clients Moyens**
    - Stratégie : croissance

- Actions : upsell via recommandations produits, cross-sell

💡 Insight SQL : segmentation RFM permet de **prioriser les actions marketing et ventes**.

8.3 Recommandations produit

1. **Top Produits**
    - Maintenir stock suffisant
    - Promouvoir lors des campagnes marketing
2. **Produits peu vendus**
    - Vérifier prix / qualité / visibilité
    - Actions : bundles, offres spéciales, promotion ciblée
3. **Analyse par catégorie**
    - Catégorie “Smartphones” = 30% du CA → stock et marketing renforcés
    - Catégorie “Accessoires” = bon volume mais faible CA → stratégie upsell

8.4 Narration Business / Storytelling

Exemple d’email ou de reporting :

Bonjour équipe,
Après analyse des ventes du dernier trimestre, voici les principaux insights :

- Le CA total s’élève à 1,245 M€, avec une concentration sur la France et la Belgique.
- Nos Top Clients, représentant 7,8% des clients, génèrent près de la moitié du CA → focus fidélisation.
- Les clients dormants représentent 20% du fichier mais seulement 5% du CA → opportunité réactivation.

- Les produits phares sont les laptops et smartphones, tandis que les accessoires nécessitent une stratégie d'upsell.
  Recommandations jointes pour chaque segment client et catégorie produit.
  Cordialement,
  [Analyste SQL – DataLendo]

8.5 Visualisation (optionnelle)

Même si nous travaillons en SQL pour le moment, on peut préparer :

- **Tableaux croisés dynamiques** pour CA par pays / segment
- **Top produits** en graphique barre
- **Distribution clients** par RFM
- Ces visualisations pourront ensuite être utilisées dans Power BI ou Tableau.

8.6 Bonnes pratiques SQL à retenir

- Toujours **documenter chaque requête** avec commentaires SQL (`-- Explication`)
- Vérifier les **doublons et les volumes** avant de faire des agrégations
- Préparer des **colonnes dérivées** (ex : panier moyen, score RFM) pour faciliter les insights
- Chaque insight doit **répondre à une question business**
- Les **résultats doivent être lisibles et exportables** pour la direction

À présent que les insights business sont prêts et rédigés, on passe à **SECTION 9 — Correction & Bonnes Pratiques SQL**, où l'on va :

- Identifier les erreurs typiques
- Montrer comment optimiser les requêtes
- Consolider toutes les bonnes pratiques apprises jusqu'ici

SECTION 9 — CORRECTION & BONNES PRATIQUES SQL

9.1 Vérification des requêtes et erreurs typiques

En travaillant sur un vrai projet, même un analyste confirmé fait des erreurs. Voici les plus courantes :

Type d'erreur	Exemple	Comment corriger
Syntaxe SQL	`SELEC * FROM ventes`	Vérifier les mots-clés : `SELECT`
Filtrage incorrect	`WHERE country = 'France' OR 'Belgique'`	Utiliser correctement : `WHERE country IN ('France','Belgique')`
Doublons involontaires	`SELECT client_id FROM ventes;` → plusieurs lignes identiques	Ajouter `DISTINCT` ou vérifier jointures
Mauvais JOIN	Oublier la condition : `FROM ventes v JOIN clients c`	Toujours spécifier la condition : `ON v.client_id = c.id`
Agrégation erronée	`SUM(montant)` sans `GROUP BY`	Ajouter `GROUP BY` sur les colonnes non agrégées
Types incompatibles	Comparer `DATE` et `TEXT`	Convertir : `CAST(date_column AS DATE)`
Colonnes dérivées mal calculées	`total_depense = quantity * prix` → erreur si NULL	Utiliser `COALESCE(quantity,0) * COALESCE(prix,0)`

💡 Astuce : **tester chaque requête sur un petit sous-ensemble avant de l'appliquer sur tout le dataset**.

9.2 Optimisation et bonnes pratiques

1. **Lisibilité avant tout**
    - Alias pour tables : `sales s JOIN clients c ON s.client_id = c.id`

- Alias pour colonnes : `SUM(quantity) AS total_ventes`

Indentation claire, commentaires SQL :

```
 -- Total des ventes par pays
SELECT c.country, SUM(s.montant) AS total_ventes
FROM ventes s
JOIN clients c ON s.client_id = c.id
GROUP BY c.country
ORDER BY total_ventes DESC;
```

2. **Eviter les sous-requêtes inutiles**

    - Préférer les `JOIN` ou les `CTE` (`WITH`) pour plus de clarté

Exemple de CTE :

```
 WITH total_ventes_client AS (
    SELECT client_id, SUM(montant) AS total_depense
    FROM ventes
    GROUP BY client_id
)
SELECT c.name, t.total_depense
FROM clients c
JOIN total_ventes_client t ON c.id = t.client_id
ORDER BY t.total_depense DESC;
```

3. **Toujours filtrer avant d'agréger**

    - `WHERE` avant `GROUP BY` → plus performant

Exemple :

```
 SELECT product_id, SUM(quantity) AS total_vendu
FROM ventes
```

```
WHERE sales_date BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY product_id;
```

4. **Contrôler les NULLs et valeurs manquantes**
    - `COALESCE()` pour éviter les erreurs dans calculs
    - Exemple : `COALESCE(quantity,0) * COALESCE(prix,0)`
5. **Versionner et documenter**
    - Commenter chaque bloc SQL
    - Pousser sur GitHub régulièrement
    - Garder un historique pour revue et partage

9.3 Vérification des résultats

Avant de livrer vos insights :

Vérifier les volumes et cohérences :

```
SELECT COUNT(*) FROM ventes; -- total attendu
SELECT COUNT(DISTINCT client_id) FROM ventes; -- clients uniques
SELECT COUNT(DISTINCT product_id) FROM ventes; -- produits uniques
```

- Comparer les KPI avec les valeurs attendues (top produits, CA global)
- Vérifier les segments clients (RFM, panier moyen, fréquence)

💡 Astuce : **si les résultats semblent incohérents, inspecter d'abord la table centrale (ventes) avant de chercher ailleurs.**

9.4 Checklist Bonnes Pratiques SQL pour un projet complet

- Alias pour tables et colonnes
- Filtrage correct avec `WHERE` / `IN` / `BETWEEN` / `LIKE`
- Agrégation correcte avec `GROUP BY`
- Vérification doublons et NULL
- Colonnes dérivées bien calculées
- CTE pour requêtes complexes
- Commentaires SQL explicatifs
- Tester sur sous-ensemble avant global
- Pousser régulièrement sur GitHub
- Vérifier cohérence avec business et KPI

Après la correction et l'optimisation, on passe à **SECTION 10 — Versioning GitHub & Résumé Projet** :

- Pousser **toutes les requêtes SQL**, fichiers CSV, insights et README sur GitHub
- Préparer un **projet complet prêt à être présenté à un manager ou un client**
- Apprendre la **culture data professionnelle**, avec versioning, documentation et partage collaboratif

SECTION 10 — Versioning GitHub & Résumé Projet

10.1 Félicitations !

Vous venez de compléter votre **premier projet complet en SQL** :

- Exploration d'un dataset réaliste
- Nettoyage et préparation des données
- Calcul de KPIs globaux et par segment
- Segmentation clients et scoring
- Analyse produits et RFM simplifiée
- Production d'insights business exploitables

👋 Vous êtes maintenant capables de travailler **comme un vrai Data Analyst en entreprise**.

10.2 Organisation et Versioning GitHub

Pour un projet SQL professionnel, la structure et la documentation sont cruciales. Voici un exemple de structure à adopter :

```
data-retail-project/
├── datasets/
│   ├── clients.csv
│   ├── produits.csv
│   ├── ventes.csv
│   └── paiements.csv
├── sql/
│   ├── exploration.sql
│   ├── nettoyage.sql
│   ├── kpis.sql
│   ├── segmentation.sql
│   ├── rfm.sql
│   └── insights.sql
├── README.md
└── outputs/
    └── rapports_insights.pdf
```

✅ Conseils pratiques pour GitHub

Initialiser le dépôt

```
git init
git add .
git commit -m "Initial commit: projet SQL Retail Jour 4"
```

1. **Branching (optionnel mais recommandé)**
    - Créez une branche pour chaque étape du projet : `exploration`, `kpis`, `segmentation`

Exemple :

```
git checkout -b exploration
```

Pousser sur GitHub

```
git remote add origin
https://github.com/votre-compte/data-retail-project.git
git push -u origin main
```

2. **Commit souvent, commentaires clairs**
    - `git commit -m "Ajout des KPIs par pays"`
    - Chaque commit doit refléter une étape concrète de votre projet
3. **README.md**
    - Expliquez votre dataset et ses colonnes
    - Listez les fichiers SQL et ce qu'ils font
    - Notez vos hypothèses business
    - Indiquez comment reproduire vos résultats

10.3 Résumé du Jour 4

À la fin de cette journée, vous savez :

- Explorer et comprendre un **dataset réaliste**
- Nettoyer et préparer vos données
- Calculer des **KPIs globaux et par segment**
- Segmenter vos clients et analyser vos produits

- Effectuer une **RFM simplifiée** et produire des insights
- Appliquer des **bonnes pratiques SQL**
- Versionner et documenter **toutes vos requêtes et fichiers** sur GitHub

💡 Vous avez vécu **le flux complet d'un projet Data Analyst** : de l'exploration à la livraison finale, prêt pour une entreprise.

Félicitations !