

TP Cloud : Pipeline Moderne

Étudiant(e) : Khadija Nachid Idrissi

: Rajaе Fdili

: Aya Hamim

Filière : Data Analytics

Contents

| | | |
|----------|--|----------|
| 1 | Introduction au Modern Data Stack | 3 |
| 1.1 | Contexte : L'évolution vers le Cloud | 3 |
| 1.2 | Le Modern Data Stack | 3 |
| 1.3 | Installation et configuration de PostgreSQL | 4 |
| 1.3.1 | Contexte dans le pipeline | 4 |
| 1.3.2 | Création de la base de données <code>shopstream</code> | 4 |
| 1.3.3 | Création des tables OLTP | 4 |
| 1.3.4 | Vérification de la création des tables | 6 |
| 1.4 | Génération de données de démonstration | 6 |
| 1.4.1 | Exécution du script de génération | 6 |
| 1.5 | Vérification des données dans pgAdmin | 8 |
| 1.5.1 | Vérification de la table <code>users</code> | 8 |
| 1.5.2 | Vérification de la table <code>orders</code> | 8 |
| 1.5.3 | Vérification de la table <code>products</code> | 8 |
| 1.6 | Création du Data Lake sur AWS S3 | 8 |
| 1.6.1 | Création d'un compte AWS | 8 |
| 1.6.2 | Accès à la console AWS | 8 |
| 1.6.3 | Création du bucket S3 | 10 |
| 1.6.4 | Création de la structure de dossiers | 10 |
| 1.6.5 | Création d'un utilisateur IAM avec accès S3 | 10 |
| 1.6.6 | Configuration d'AWS CLI | 12 |
| 1.7 | Export des données PostgreSQL vers Amazon S3 | 13 |
| 1.7.1 | Script Python d'export | 13 |
| 1.7.2 | Exécution du script | 13 |
| 1.8 | Configuration de Snowflake | 13 |
| 1.8.1 | Création des schémas | 13 |
| 1.8.2 | Création des Virtual Warehouses | 16 |
| 1.8.3 | Création de la Storage Integration (S3) | 16 |
| 1.8.4 | Création du Stage externe | 16 |
| 1.8.5 | Création des tables STAGING | 16 |
| 1.8.6 | Chargement des données depuis S3 | 16 |
| 1.9 | Installation et utilisation de dbt | 16 |
| 1.9.1 | Installation de dbt | 16 |
| 1.9.2 | Initialisation et configuration du projet dbt | 16 |
| 1.10 | Exécution de dbt | 21 |
| 1.10.1 | Exécution des modèles dbt | 21 |
| 1.10.2 | Exécution des tests dbt | 22 |
| 1.10.3 | Génération de la documentation | 22 |

CONTENTS

| | | |
|--------|---|----|
| 1.10.4 | Vérification dans Snowflake | 23 |
| 1.10.5 | Récapitulatif dbt | 23 |
| 1.11 | Création des Data Marts | 24 |
| 1.11.1 | Data Mart : Performance Produits | 24 |
| 1.11.2 | Data Mart : Customer Lifetime Value (CLV) | 25 |
| 1.11.3 | Résumé Data Marts | 26 |
| 1.12 | Connexion Power BI et création de dashboards | 26 |
| 1.12.1 | Pourquoi Power BI ? | 26 |
| 1.12.2 | Connexion à Snowflake | 27 |
| 1.12.3 | Création du dashboard "Vue d'ensemble des ventes" | 27 |
| 1.12.4 | Création de mesures DAX | 29 |
| 1.12.5 | Création d'une deuxième page : Performance Produits | 31 |
| 1.12.6 | Création d'une troisième page : Analyse Clients | 33 |
| 1.12.7 | Récapitulatif | 34 |

1. Introduction au Modern Data Stack

1.1 Contexte : L'évolution vers le Cloud

Depuis une dizaine d'années, les entreprises migrent leurs infrastructures de données vers le cloud. Cette évolution s'explique par plusieurs facteurs :

- **Scalabilité élastique** : Les ressources s'adaptent automatiquement aux besoins (pics de charge, croissance des données).
- **Coûts optimisés** : Modèle *pay-as-you-go*, pas d'investissement matériel initial.
- **Maintenance réduite** : Fini les serveurs à gérer, patches, sauvegardes physiques.
- **Innovation rapide** : Accès aux dernières technologies (Machine Learning, IA, streaming).
- **Collaboration facilitée** : Accès global, travail à distance, partage de données.

1.2 Le Modern Data Stack

Le terme "**Modern Data Stack**" désigne un ensemble d'outils *cloud-native*, modulaires et intégrés, qui permettent de construire rapidement des pipelines de données robustes et scalables.

Un pipeline typique comprend plusieurs composants :

- **PostgreSQL** : base de données source.
- **Amazon S3** : stockage cloud des données brutes.
- **Snowflake** : entrepôt de données pour stocker et structurer les données.
- **dbt** : transformation des données pour les rendre prêtes à l'analyse.
- **Airflow** : orchestration et automatisation du pipeline.
- **Power BI** : visualisation et création de dashboards interactifs.

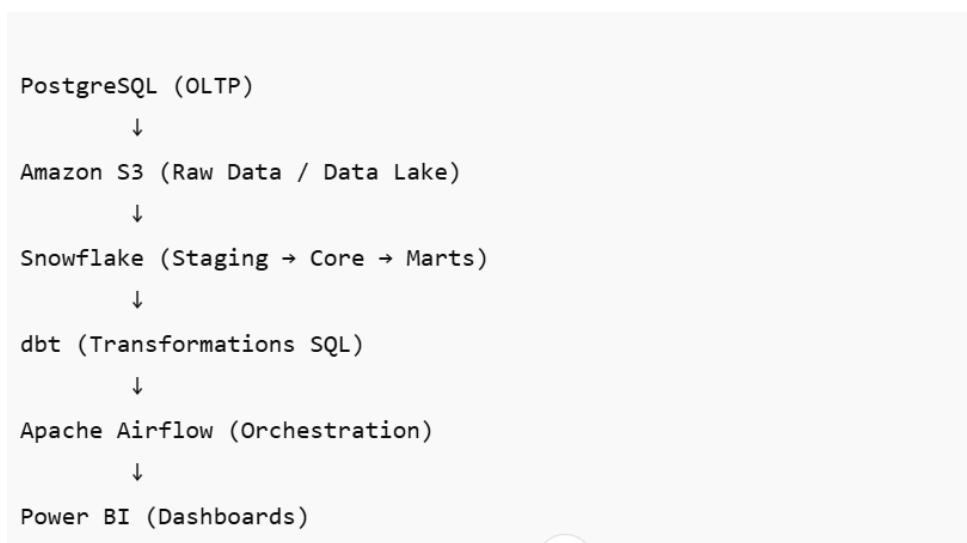


Figure 1.1: Architecture simplifiée d'un Modern Data Stack

1.3 Installation et configuration de PostgreSQL

1.3.1 Contexte dans le pipeline

La première étape du pipeline de données consiste à mettre en place une source de données transactionnelle (OLTP). Dans notre architecture *ShopStream*, PostgreSQL joue le rôle de base de données opérationnelle à partir de laquelle les données seront extraites et intégrées dans le Data Lake.

PostgreSQL (OLTP) → S3 (Data Lake) → Snowflake (DWH) → dbt (Transform) → Airflow (Orchestration) → Power BI (BI)

PostgreSQL a été choisi pour sa robustesse, sa conformité ACID et sa compatibilité native avec les outils Big Data et analytiques.

1.3.2 Création de la base de données shopstream

Via pgAdmin

1.3.3 Création des tables OLTP

Les tables suivantes modélisent l'activité transactionnelle de la plateforme *ShopStream* :

- users
- products
- orders
- order_items
- events
- crm_contacts

Le script SQL exécuté permet de créer les tables, leurs clés primaires, clés étrangères et index afin d'optimiser les performances OLTP.

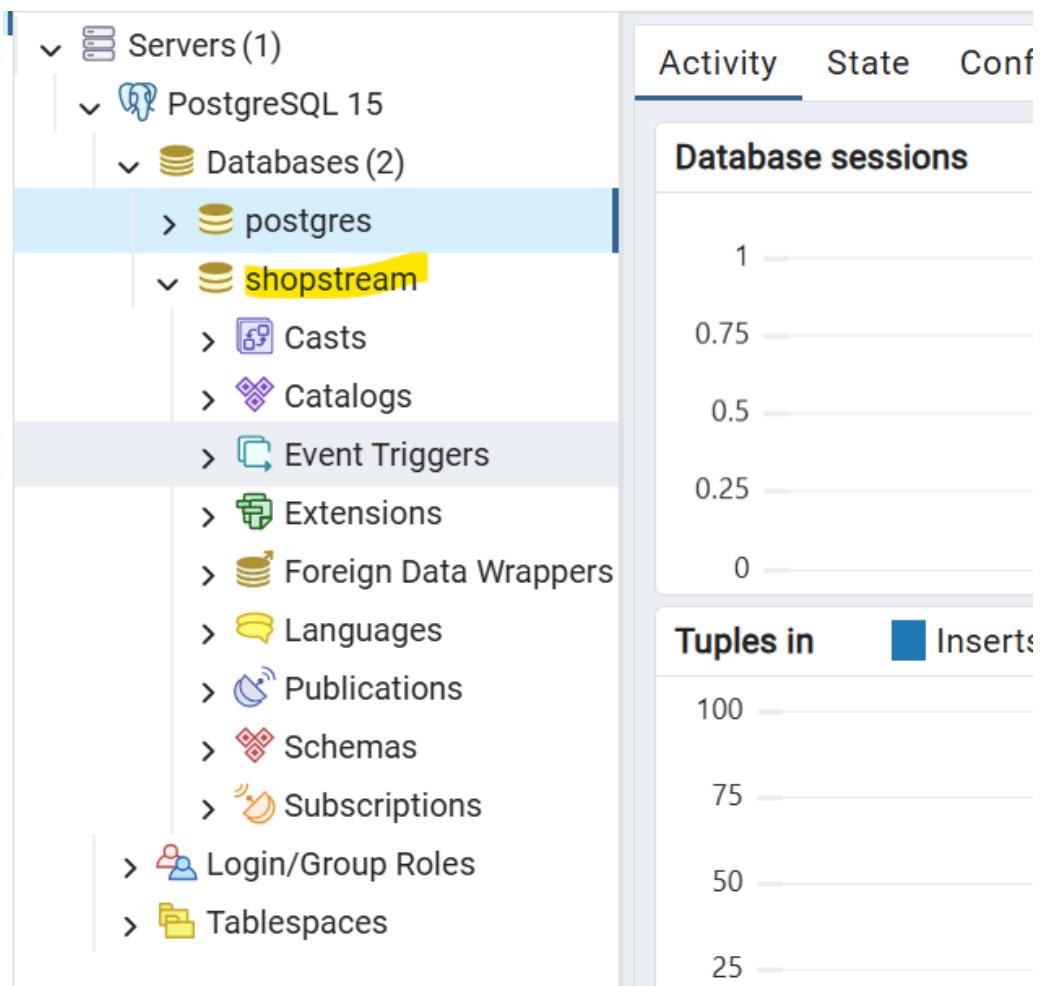


Figure 1.2: Création de la base de données shopstream

1.3.4 Vérification de la création des tables

Après l'exécution du script de création de la base de données, les tables sont visibles dans l'outil **pgAdmin**, ce qui confirme la bonne mise en place du schéma OLTP de la base `shopstream`.

Les tables suivantes ont été créées :

- users
- products
- orders
- order_items
- events
- crm_contacts

Conclusion

La base de données PostgreSQL `shopstream` est désormais correctement installée, configurée et structurée selon un modèle OLTP cohérent. Elle constitue une source de données fiable pour la suite du pipeline, notamment l'ingestion vers le Data Lake (couche Bronze).

1.4 Génération de données de démonstration

Cette étape consiste à générer des données de démonstration afin d'alimenter la base de données `ShopStream` pour les tests et les analyses ultérieures.

1.4.1 Exécution du script de génération

Une invite de commande (CMD ou Terminal) est ouverte, puis l'utilisateur se place dans le dossier `scripts` du projet.

- Windows :

```
cd C:\Users\VotreNom\ShopStreamTP\scripts
```

- macOS / Linux :

```
cd ~/ShopStreamTP/scripts
```

Le script de génération est ensuite exécuté avec la commande suivante :

```
python generate_data.py
```

Le script démarre et affiche les messages de progression indiquant la génération des utilisateurs, produits, commandes, événements et contacts CRM. Après quelques minutes, le message **GENERATION TERMINEE AVEC SUCCES** confirme que toutes les données ont été insérées correctement dans PostgreSQL.

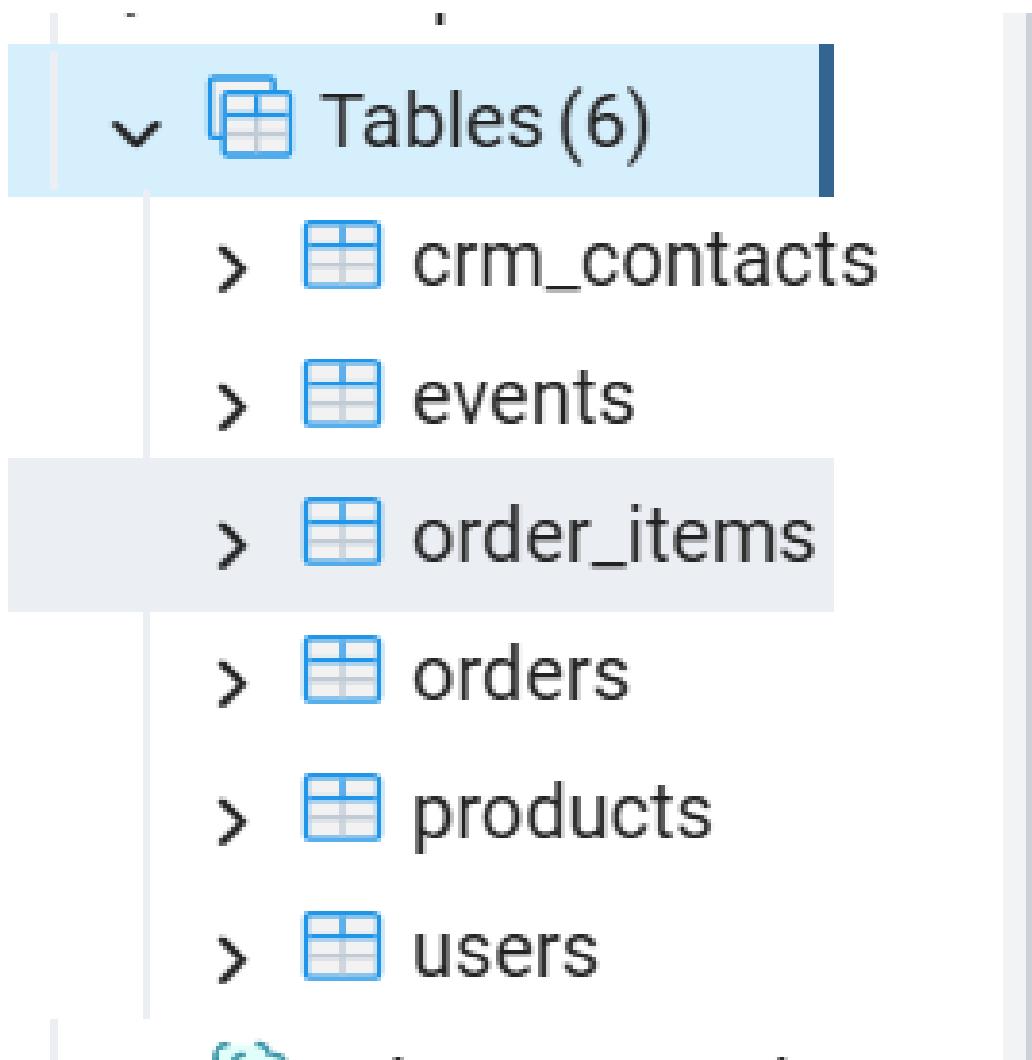


Figure 1.3: Liste des tables OLTP dans la base shopstream

Récapitulatif :
- Users : 1000
- Products : 200
- Orders : 5000
- Order Items : 15112
- Events : 10000
- CRM Contacts : 500

Figure 1.4: Vérification des données générées

1.5 Vérification des données dans pgAdmin

Après la génération des données, une vérification est effectuée à l'aide de **pgAdmin** afin de s'assurer que chaque table contient bien des enregistrements.

1.5.1 Vérification de la table users

Dans le *Query Tool* de pgAdmin, la requête suivante est exécutée :

```
SELECT * FROM users LIMIT 10;
```

Le résultat affiche les dix premiers utilisateurs générés, confirmant que la table `users` a été correctement alimentée.

1.5.2 Vérification de la table orders

La table `orders` est vérifiée à l'aide de la requête suivante :

```
SELECT * FROM orders LIMIT 10;
```

Les résultats affichent les premières commandes générées, incluant les identifiants clients, les dates de commande et les montants associés.

1.5.3 Vérification de la table products

La table `products` est ensuite vérifiée avec la requête suivante :

```
SELECT * FROM products LIMIT 10;
```

Les résultats affichent correctement les produits générés avec leurs caractéristiques principales, confirmant la cohérence des données insérées.

Cette étape valide le bon fonctionnement du script de génération et garantit que la base de données est prête pour les traitements suivants, notamment l'ingestion vers le Data Lake et la mise en place du pipeline analytique.

1.6 Crédation du Data Lake sur AWS S3

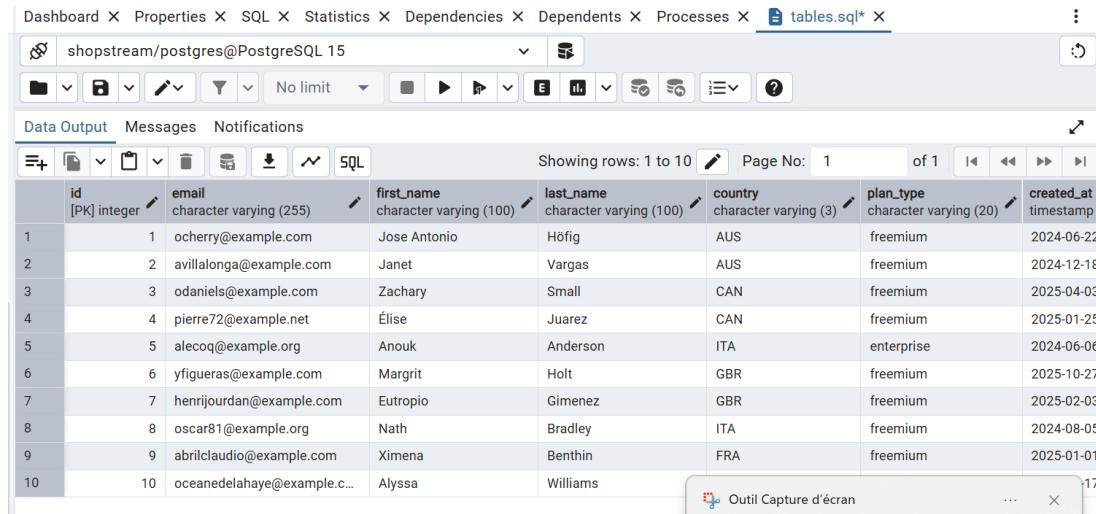
1.6.1 Crédation d'un compte AWS

Un compte AWS est créé via le site officiel <https://aws.amazon.com/>. L'inscription nécessite une adresse email valide, des informations personnelles et une carte bancaire. Le plan **AWS Free Tier** est sélectionné, ce qui permet d'utiliser les services S3 gratuitement dans les limites offertes.

Une fois l'inscription terminée, l'utilisateur accède à la **console AWS**.

1.6.2 Accès à la console AWS

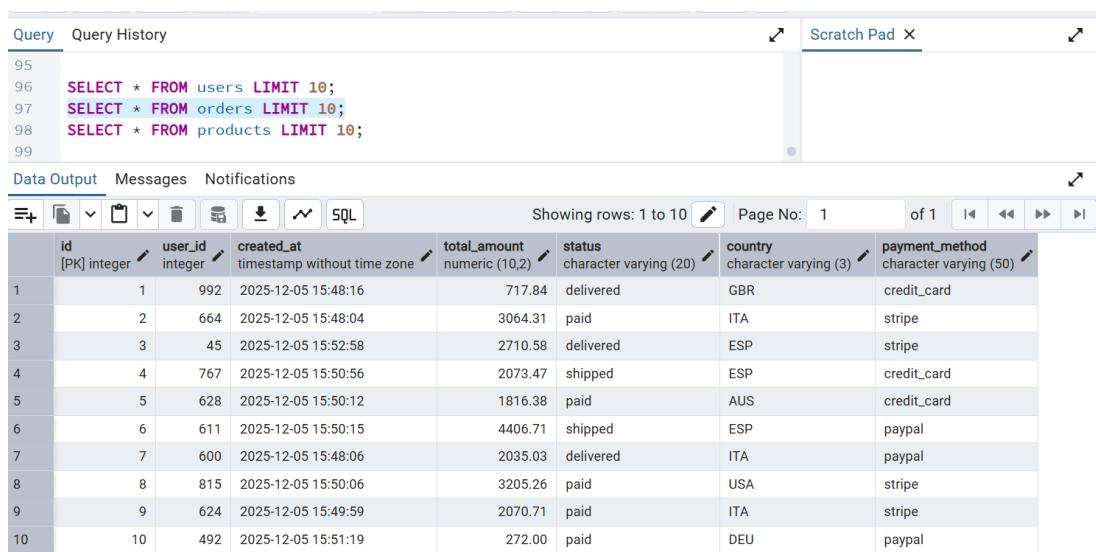
La connexion se fait via <https://console.aws.amazon.com/>. Après authentification, la région AWS est vérifiée et définie sur : **Europe (Paris) – eu-west-3**.



The screenshot shows a PostgreSQL database interface with the 'users' table selected. The table has columns: id, email, first_name, last_name, country, plan_type, and created_at. The data shows 10 rows of user information, including names like Jose Antonio Höfig and Janet Vargas, and email addresses like ocherry@example.com and avillalonga@example.com.

| | <code>id</code> [PK] integer | <code>email</code> character varying (255) | <code>first_name</code> character varying (100) | <code>last_name</code> character varying (100) | <code>country</code> character varying (3) | <code>plan_type</code> character varying (20) | <code>created_at</code> timestamp |
|----|---------------------------------|---|--|---|---|--|--------------------------------------|
| 1 | 1 | ocherry@example.com | Jose Antonio | Höfig | AUS | freemium | 2024-06-22 |
| 2 | 2 | avillalonga@example.com | Janet | Vargas | AUS | freemium | 2024-12-18 |
| 3 | 3 | odaniels@example.com | Zachary | Small | CAN | freemium | 2025-04-03 |
| 4 | 4 | pierre72@example.net | Élise | Juarez | CAN | freemium | 2025-01-25 |
| 5 | 5 | alecoq@example.org | Anouk | Anderson | ITA | enterprise | 2024-06-06 |
| 6 | 6 | yfigueras@example.com | Margrit | Holt | GBR | freemium | 2025-10-27 |
| 7 | 7 | henrijourdan@example.com | Eutropio | Gimenez | GBR | freemium | 2025-02-03 |
| 8 | 8 | oscar81@example.org | Nath | Bradley | ITA | freemium | 2024-08-05 |
| 9 | 9 | abrilclaudio@example.com | Ximena | Benthin | FRA | freemium | 2025-01-01 |
| 10 | 10 | oceannedelahaye@example.c... | Alyssa | Williams | | | |

Figure 1.5: Vérification des données générées dans la table users



The screenshot shows a PostgreSQL database interface with the 'orders' table selected. The table has columns: id, user_id, created_at, total_amount, status, country, and payment_method. The data shows 10 rows of order information, including details like a total amount of 717.84 and a status of delivered.

| | <code>id</code> [PK] integer | <code>user_id</code> integer | <code>created_at</code> timestamp without time zone | <code>total_amount</code> numeric (10,2) | <code>status</code> character varying (20) | <code>country</code> character varying (3) | <code>payment_method</code> character varying (50) |
|----|---------------------------------|---------------------------------|--|---|---|---|---|
| 1 | 1 | 992 | 2025-12-05 15:48:16 | 717.84 | delivered | GBR | credit_card |
| 2 | 2 | 664 | 2025-12-05 15:48:04 | 3064.31 | paid | ITA | stripe |
| 3 | 3 | 45 | 2025-12-05 15:52:58 | 2710.58 | delivered | ESP | stripe |
| 4 | 4 | 767 | 2025-12-05 15:50:56 | 2073.47 | shipped | ESP | credit_card |
| 5 | 5 | 628 | 2025-12-05 15:50:12 | 1816.38 | paid | AUS | credit_card |
| 6 | 6 | 611 | 2025-12-05 15:50:15 | 4406.71 | shipped | ESP | paypal |
| 7 | 7 | 600 | 2025-12-05 15:48:06 | 2035.03 | delivered | ITA | paypal |
| 8 | 8 | 815 | 2025-12-05 15:50:06 | 3205.26 | paid | USA | stripe |
| 9 | 9 | 624 | 2025-12-05 15:49:59 | 2070.71 | paid | ITA | stripe |
| 10 | 10 | 492 | 2025-12-05 15:51:19 | 272.00 | paid | DEU | paypal |

Figure 1.6: Vérification des données générées dans la table orders

1.6.3 Creation du bucket S3

Un bucket Amazon S3 est cre  afin de servir de **Data Lake** pour le projet ShopStream.

- Nom du bucket : `shopstream-datalake-votreprenom`
- Region : Europe (Paris)
- Accès public : bloqu  (param tres par defaut)
- Chiffrement : activ  (SSE-S3)

La creation du bucket est valid e par un message de confirmation affich  dans la console AWS.

1.6.4 Creation de la structure de dossiers

Une structure de dossiers est cre    l'int rieur du bucket afin d'organiser les donn es brutes (couche **Raw / Bronze**).

La structure finale du Data Lake est la suivante :

```
shopstream-datalake-votreprenom/
  raw/
    postgres/
      users/
      products/
      orders/
      order_items/
    events/
  crm/
```

Cette organisation permet de s parer clairement les sources de donn es et facilite les traitements ult rieurs.

Cette ´tape valide la mise en place du Data Lake sur AWS S3 et pr pare l'environnement pour l'ingestion des donn es.

1.6.5 Creation d'un utilisateur IAM avec acc s S3

Afin de permettre aux scripts Python (et ult rieurement   Snowflake) d'acc der au Data Lake S3, un utilisateur **IAM** d di  est cre  avec des droits sur Amazon S3.

Dans la console AWS, le service **IAM** est ouvert   partir de la barre de recherche, puis l'option **Utilisateurs** est s lectionn e .

Un nouvel utilisateur est cre  avec les param tres suivants :

- Nom d'utilisateur : `shopstream-s3-user`
- Type d'autorisation : Attacher directement des strat gies
- Strat gie associ e : `AmazonS3FullAccess`

CHAPTER 1. INTRODUCTION AU MODERN DATA STACK

The screenshot shows a PostgreSQL database interface. In the top-left, there's a 'Query History' section with the following SQL code:

```

95
96   SELECT * FROM users LIMIT 10;
97   SELECT * FROM orders LIMIT 10;
98   SELECT * FROM products LIMIT 10
99

```

In the top-right, there's a 'Scratch Pad' tab. Below these tabs is a navigation bar with 'Data Output', 'Messages', and 'Notifications'.

The main area displays the results of the query in a table format:

| | id [PK] integer | merchant_id integer | name character varying (255) | description text |
|----|---------------------------|-------------------------------|--|---|
| 1 | 1 | 40 | Reverse-engineered bandwidth-monitored time... | Den schenken Musik anfangen bleiben Garten. Auf schwer waschen Zimmer tun Kl... |
| 2 | 2 | 14 | Le confort de changer à la pointe | Importancia precisamente evitar ayer. Producción encuentra hermano hechos otros |
| 3 | 3 | 67 | Vision-oriented national monitoring | Although describe culture. Talk various he rise believe. Goal know through month. |
| 4 | 4 | 62 | Fully-configurable client-driven methodology | Realidad generales elementos les. Paz fuerzas ni yo e habla. |
| 5 | 5 | 41 | Persevering client-driven moderator | Direction désir clef eh dire. Eau verser haut. Corde lorsque rouge. |
| 6 | 6 | 41 | L'assurance de changer à l'état pur | Elementos asociación europa bastante etapa fin unos. Aparece medida te. Jefe pa... |
| 7 | 7 | 26 | L'art de concrétiser vos projets en toute tranqui... | List lose goal student. Our ahead hand drop amount front others. Single tax speech |
| 8 | 8 | 19 | Multi-lateral discrete initiative | Laugh article base talk. Identify week behind painting far. Dog civil major imagine R |
| 9 | 9 | 81 | Profit-focused mobile interface | Phone type suffer three. Pick society run statement. Wind fish study teacher and o... |
| 10 | 10 | 84 | User-centric static utilization | Jour depuis jeune. Comme différent couleur beauté essayer agir réduire. Discours r... |

Figure 1.7: Vérification des données générées dans la table products

The screenshot shows the AWS S3 console. At the top, it says 'Amazon S3 > Compartiments'. A green banner at the top indicates that a compartment named 'shopstream-datalake-khadija' has been created successfully. Below this, there's a search bar and a table for managing compartments. On the right, there's a sidebar with information about Storage Lens and a summary of external access.

Figure 1.8: Crédation du bucket S3 pour le Data Lake ShopStream

The screenshot shows the AWS S3 console with the path 'Amazon S3 > Compartiments > shopstream-datalake-khadija > raw/ > postgres/'. It displays a list of objects (4) under the 'Objects' tab. The table includes columns for Nom, Type, Dernière modification, Taille, and Classe de stockage. The objects listed are 'order_items/', 'orders/', 'products/', and 'users/'.

Figure 1.9: Vérification de la structure des sous-dossiers dans le bucket S3

Une fois l'utilisateur créé, des **clés d'accès** sont générées afin de permettre l'authentification depuis une application externe à AWS.

Le cas d'utilisation sélectionné est : *Application s'exécutant en dehors d'AWS.*

Deux informations sont alors fournies :

- Access Key ID
- Secret Access Key

Ces clés sont téléchargées sous forme de fichier .csv et stockées de manière sécurisée.

Bonne pratique de sécurité

Les clés d'accès AWS sont strictement confidentielles :

- Elles ne doivent jamais être partagées
- Elles ne doivent jamais être versionnées dans Git
- Elles ne doivent jamais être publiées en ligne

Elles doivent être conservées dans un gestionnaire de mots de passe ou un fichier sécurisé local.

Cette étape permet d'assurer un accès sécurisé et contrôlé au Data Lake S3 depuis les outils externes du projet.

1.6.6 Configuration d'AWS CLI

Afin de permettre l'interaction entre la machine locale et Amazon S3, l'outil **AWS CLI (Command Line Interface)** est installé puis configuré avec les clés IAM précédemment créées.

Installation d'AWS CLI

L'installation d'AWS CLI dépend du système d'exploitation utilisé :

- **Windows** : téléchargement du programme d'installation officiel (`AWSCLIV2.msi`) et installation avec les paramètres par défaut.
- **macOS** : installation via Homebrew avec la commande `brew install awscli`.
- **Linux** : téléchargement de l'archive, décompression et installation via le script fourni par AWS.

Une fois l'installation terminée, l'invite de commande est fermée puis rouverte afin de prendre en compte l'installation.

Configuration des identifiants AWS

Dans une invite de commande (CMD ou Terminal), la commande suivante est exécutée :

```
aws configure
```

Les informations demandées sont renseignées à l'aide des clés IAM générées précédemment :

- AWS Access Key ID
- AWS Secret Access Key
- Région par défaut : `eu-west-3`
- Format de sortie : `json`

Cette étape permet de lier la machine locale au compte AWS de manière sécurisée.

Vérification de la connexion à S3

Pour vérifier que la configuration est correcte, la commande suivante est exécutée :

```
aws s3 ls
```

La liste des buckets S3 du compte AWS s'affiche, incluant le bucket créé précédemment pour le Data Lake. L'affichage du bucket confirme que la connexion entre AWS CLI et le compte AWS fonctionne correctement.

1.7 Export des données PostgreSQL vers Amazon S3

Cette étape consiste à exporter les données des tables PostgreSQL vers le bucket S3 créé précédemment. Les fichiers sont stockés au format CSV dans la structure de dossiers `raw/postgres`.

1.7.1 Script Python d'export

Le script Python `export_to_s3.py` se connecte à PostgreSQL, lit les tables et les exporte vers S3 l'aide de `boto3`.

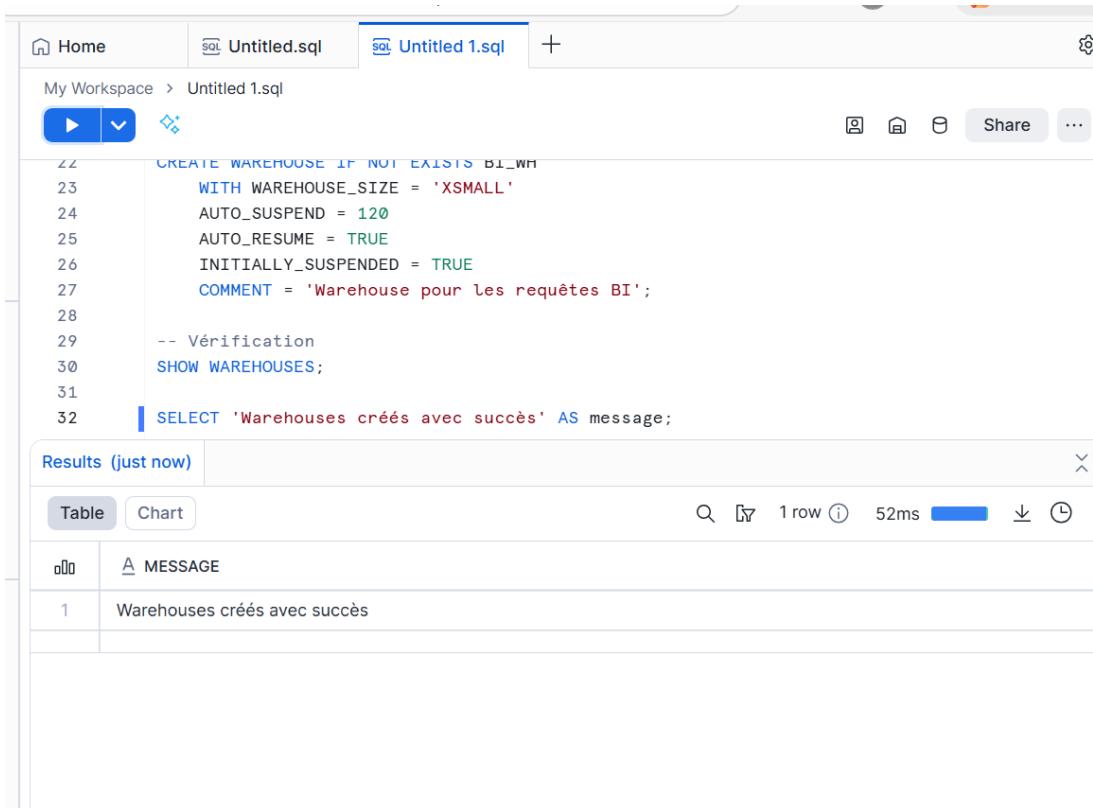
1.7.2 Exécution du script

1.8 Configuration de Snowflake

Cette étape consiste à configurer Snowflake pour accueillir le Data Warehouse ShopStream, créer les schémas, les warehouses, l'intégration S3 et charger les données.

1.8.1 Création des schémas

Les schémas suivants sont créés dans Snowflake : RAW, STAGING, CORE et MARTS.



The screenshot shows a SQL editor interface with two tabs: 'Untitled.sql' and 'Untitled 1.sql'. The 'Untitled 1.sql' tab is active, displaying the following SQL code:

```

22 CREATE WAREHOUSE IF NOT EXISTS BI_WH
23   WITH WAREHOUSE_SIZE = 'XSMALL'
24     AUTO_SUSPEND = 120
25     AUTO_RESUME = TRUE
26     INITIALLY_SUSPENDED = TRUE
27     COMMENT = 'Warehouse pour les requêtes BI';
28
29 -- Vérification
30 SHOW WAREHOUSES;
31
32 SELECT 'Warehouses créés avec succès' AS message;

```

Below the code, the results pane shows a single row of data:

| | MESSAGE |
|---|------------------------------|
| 1 | Warehouses créés avec succès |

Figure 1.10: Création de l'utilisateur IAM et génération des clés d'accès S3

```

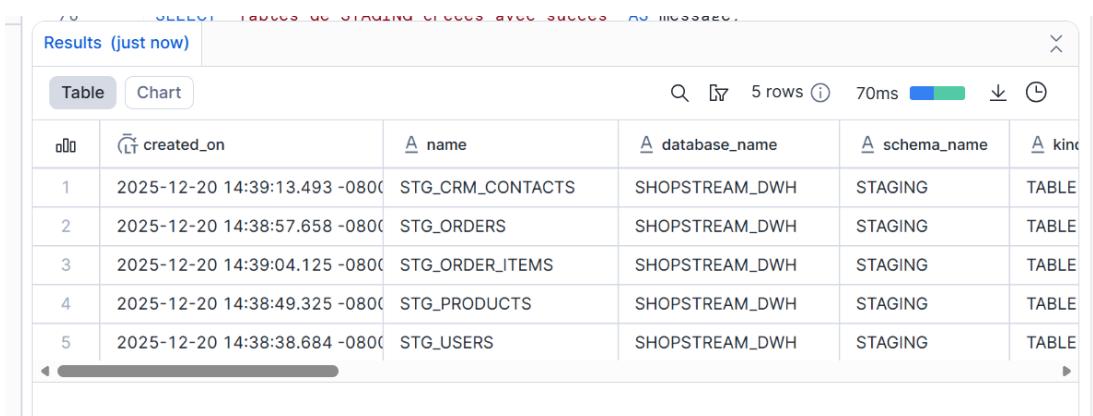
C:\Users\LENOVO>aws configure
AWS Access Key ID [None]: AKIASCF3P4D5TCQ4C2W5
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-3
Default output format [None]: json

C:\Users\LENOVO>aws s3 ls
2025-12-18 23:42:19 shopstream-datalake-khadija

C:\Users\LENOVO>

```

Figure 1.11: Configuration d'AWS CLI avec la commande aws configure et la commande aws s3 ls



The screenshot shows a SQL editor interface with a results pane titled 'Results (just now)'. The results are presented in a table format:

| | created_on | name | database_name | schema_name | kind |
|---|-------------------------------|------------------|----------------|-------------|-------|
| 1 | 2025-12-20 14:39:13.493 -0800 | STG_CRM_CONTACTS | SHOPSTREAM_DWH | STAGING | TABLE |
| 2 | 2025-12-20 14:38:57.658 -0800 | STG_ORDERS | SHOPSTREAM_DWH | STAGING | TABLE |
| 3 | 2025-12-20 14:39:04.125 -0800 | STG_ORDER_ITEMS | SHOPSTREAM_DWH | STAGING | TABLE |
| 4 | 2025-12-20 14:38:49.325 -0800 | STG_PRODUCTS | SHOPSTREAM_DWH | STAGING | TABLE |
| 5 | 2025-12-20 14:38:38.684 -0800 | STG_USERS | SHOPSTREAM_DWH | STAGING | TABLE |

```

Connexion S3 réussie
Uploadé vers s3://shopstream-datalake-khadija/raw/postgres/orders/2025-12-19/orders_20251219.csv

Export de la table 'order_items'...
Connexion à PostgreSQL...
Connexion PostgreSQL réussie
15112 lignes extraites de PostgreSQL
Connexion à AWS S3...
Connexion S3 réussie
Uploadé vers s3://shopstream-datalake-khadija/raw/postgres/order_items/2025-12-19/order_items_20251219.csv

Export de la table 'crm_contacts'...
Connexion à PostgreSQL...
Connexion PostgreSQL réussie
500 lignes extraites de PostgreSQL
Connexion à AWS S3...
Connexion S3 réussie
Uploadé vers s3://shopstream-datalake-khadija/raw/postgres/crm_contacts/2025-12-19/crm_contacts_20251219.csv

Export de la table 'events'...
Connexion à PostgreSQL...
Connexion PostgreSQL réussie
C:\Users\LENOVO\Desktop\ShopStreamTP\scripts\export_to_s3.py:112: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
df = pd.read_sql(query, conn)
10000 lignes extraites de PostgreSQL
Connexion à AWS S3...
Connexion S3 réussie
Uploadé vers s3://shopstream-datalake-khadija/raw/events/2025-12-19/events_20251219.json

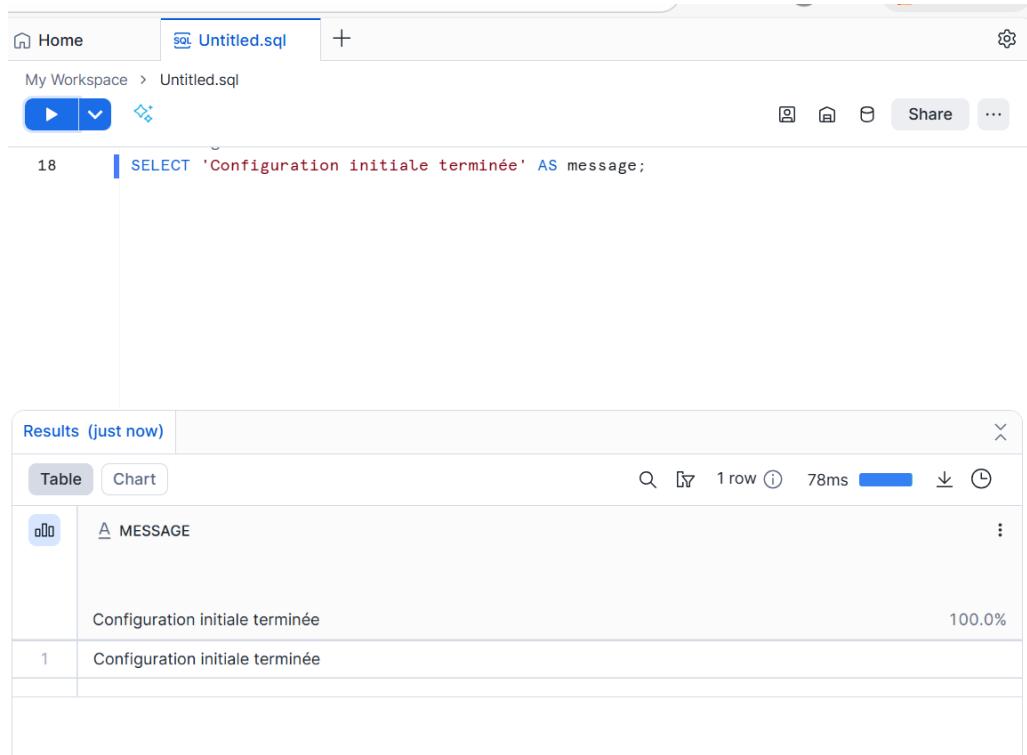
=====
EXPORT TERMINE AVEC SUCCES
=====
```

```

=====
EXPORT TERMINE AVEC SUCCES
=====

Vérifiez dans S3 : https://s3.console.aws.amazon.com/s3/buckets/shopstream-datalake-khadija
C:\Users\LENOVO\Desktop\ShopStreamTP\scripts>
```

Figure 1.12: Exécution du script Python d'export PostgreSQL vers S3



The screenshot shows the Snowflake SQL interface. At the top, there's a navigation bar with 'Home', a file tab labeled 'Untitled.sql', and a '+' button. Below the navigation is a toolbar with icons for play, dropdown, and refresh, followed by 'Share' and '...' buttons. The main area contains a code editor with the following SQL query:

```
18     SELECT 'Configuration initiale terminée' AS message;
```

Below the code editor is a results panel titled 'Results (just now)'. It has tabs for 'Table' and 'Chart'. The 'Table' tab is selected, showing a single row with the following data:

| | MESSAGE | |
|-----|---------------------------------|--------|
| 000 | Configuration initiale terminée | 100.0% |
| 1 | Configuration initiale terminée | |

Figure 1.13: Vérification des schémas créés dans Snowflake

1.8.2 Création des Virtual Warehouses

Trois Virtual Warehouses sont créés pour gérer les différents besoins :

- `LOADING_WH` : chargement des données depuis S3
- `TRANSFORM_WH` : transformations dbt
- `BI_WH` : requêtes BI

1.8.3 Création de la Storage Integration (S3)

Une Storage Integration permet à Snowflake de se connecter au bucket S3 et de lire les fichiers CSV générés précédemment.

1.8.4 Création du Stage externe

Le Stage externe pointe vers le bucket S3 et permet de lister et charger les fichiers CSV dans Snowflake.

1.8.5 Création des tables STAGING

Les tables STAGING sont créées pour accueillir les données brutes venant de S3, prêtes pour les transformations ultérieures.

1.8.6 Chargement des données depuis S3

Les données CSV des tables PostgreSQL et CRM sont chargées dans les tables STAGING à l'aide de la commande `COPY INTO`.

1.9 Installation et utilisation de dbt

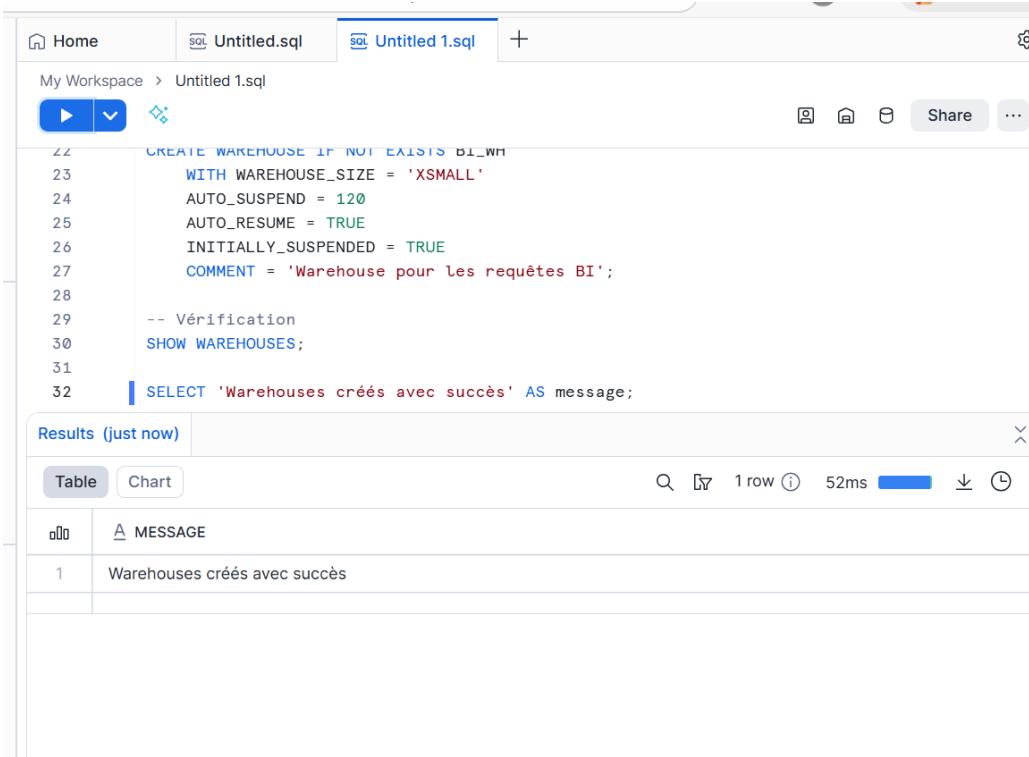
dbt (data build tool) permet de transformer les données brutes dans Snowflake (STAGING) en dimensions, faits et data marts optimisés pour l'analyse. Il utilise du SQL déclaratif et gère les dépendances entre modèles.

1.9.1 Installation de dbt

Résumé : Installation de dbt et de l'adaptateur Snowflake via pip. Vérification avec `dbt -version`.

1.9.2 Initialisation et configuration du projet dbt

Résumé : Crédit à la création du projet dbt (`dbt init shopstream_dbt`) et configuration de la connexion Snowflake via `profiles.yml`. Test de connexion avec `dbt debug`.



The screenshot shows the Snowflake SQL interface. In the top navigation bar, there are tabs for 'Home', 'Untitled.sql' (which is currently selected), and 'Untitled 1.sql'. Below the tabs, the breadcrumb path 'My Workspace > Untitled 1.sql' is visible. On the left, a code editor displays the following SQL script:

```

22 CREATE WAREHOUSE IF NOT EXISTS BI_WH
23   WITH WAREHOUSE_SIZE = 'XSMALL'
24     AUTO_SUSPEND = 120
25     AUTO_RESUME = TRUE
26     INITIALLY_SUSPENDED = TRUE
27     COMMENT = 'Warehouse pour les requêtes BI';
28
29 -- Vérification
30 SHOW WAREHOUSES;
31
32 SELECT 'Warehouses créés avec succès' AS message;

```

In the bottom right corner of the code editor, there are sharing and more options buttons. Below the code editor, the results pane is titled 'Results (just now)'. It shows a table with one row, labeled 'MESSAGE', containing the text 'Warehouses créés avec succès'. The results pane also includes a search bar, a refresh button, and performance metrics: 1 row, 52ms.

Figure 1.14: Vérification des Virtual Warehouses créés dans Snowflake

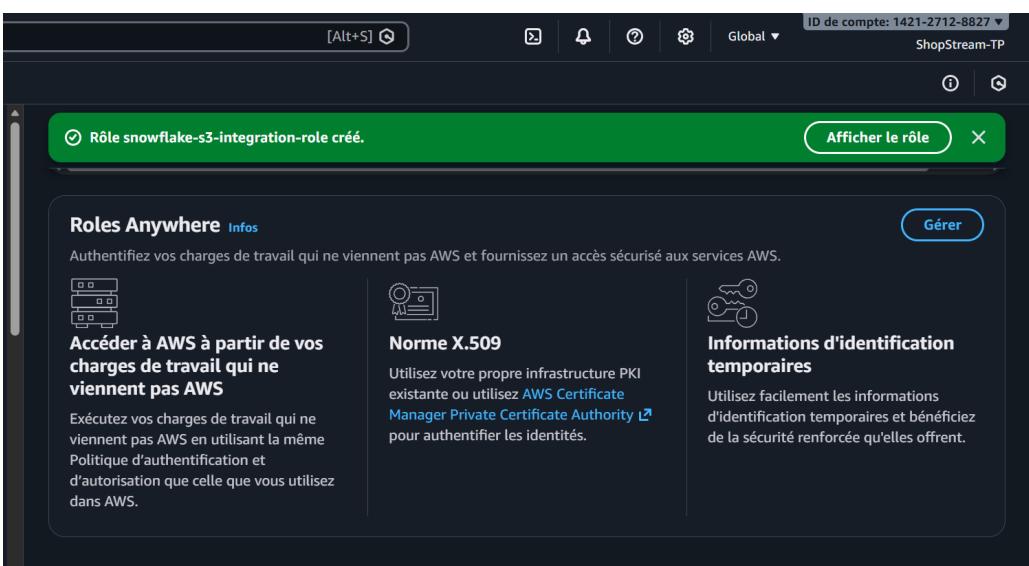
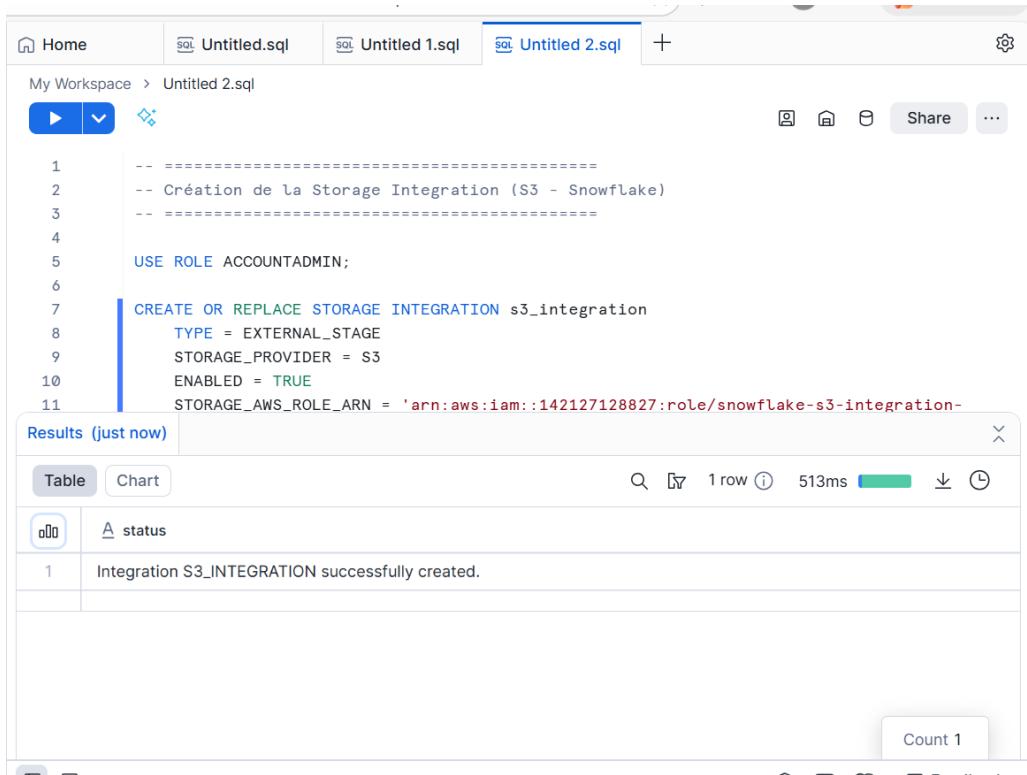


Figure 1.15: Étape A : Crédit d'un rôle IAM pour Snowflake



The screenshot shows the Snowflake SQL editor interface. The top navigation bar has tabs for 'Home', 'Untitled.sql', 'Untitled 1.sql', and 'Untitled 2.sql' (which is currently selected). Below the tabs, the breadcrumb path 'My Workspace > Untitled 2.sql' is visible. The main area contains the following SQL code:

```

1 -- =====
2 -- Création de la Storage Integration (S3 - Snowflake)
3 -- =====
4
5 USE ROLE ACCOUNTADMIN;
6
7 CREATE OR REPLACE STORAGE INTEGRATION s3_integration
8   TYPE = EXTERNAL_STAGE
9   STORAGE_PROVIDER = S3
10  ENABLED = TRUE
11  STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::142127128827:role/snowflake-s3-integration'

```

Below the code, the results pane shows a single row of data:

| Results (just now) | |
|--------------------|--|
| Table | Chart |
| | status |
| 1 | Integration S3_INTEGRATION successfully created. |

A 'Count 1' button is located at the bottom right of the results pane.

Figure 1.16: Étape B : Crédit de la Storage Integration dans Snowflake

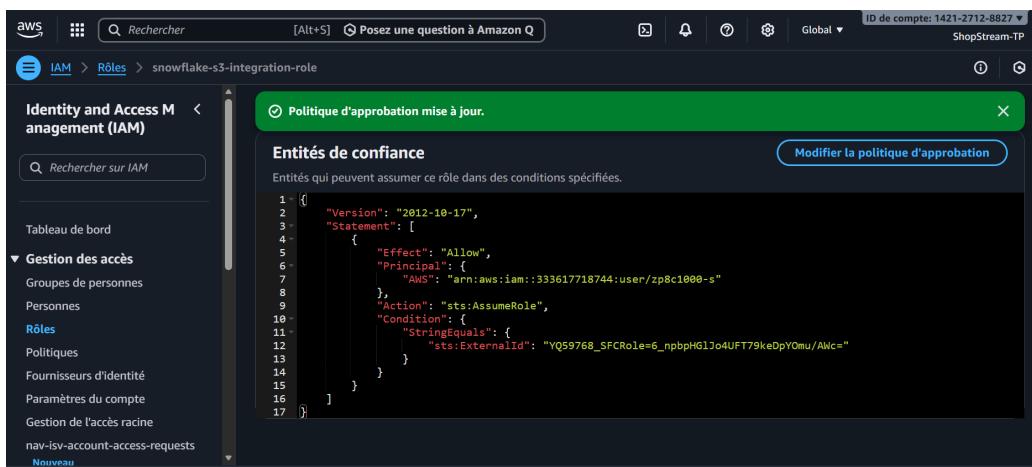
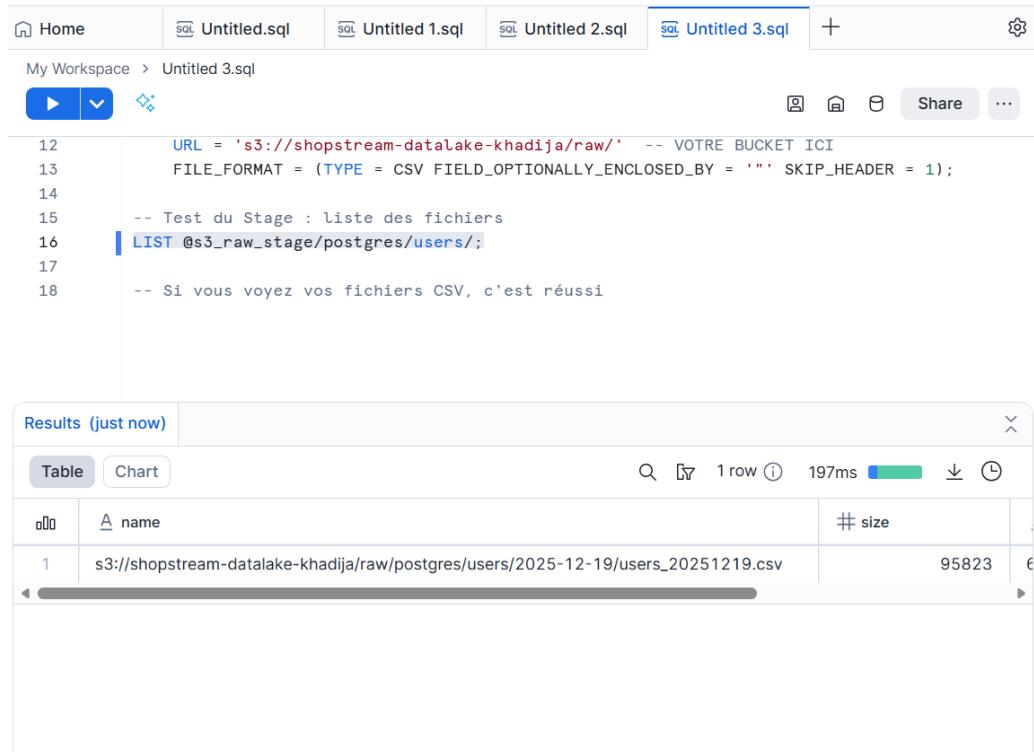


Figure 1.17: Étape C : Configuration de la relation de confiance dans AWS

CHAPTER 1. INTRODUCTION AU MODERN DATA STACK

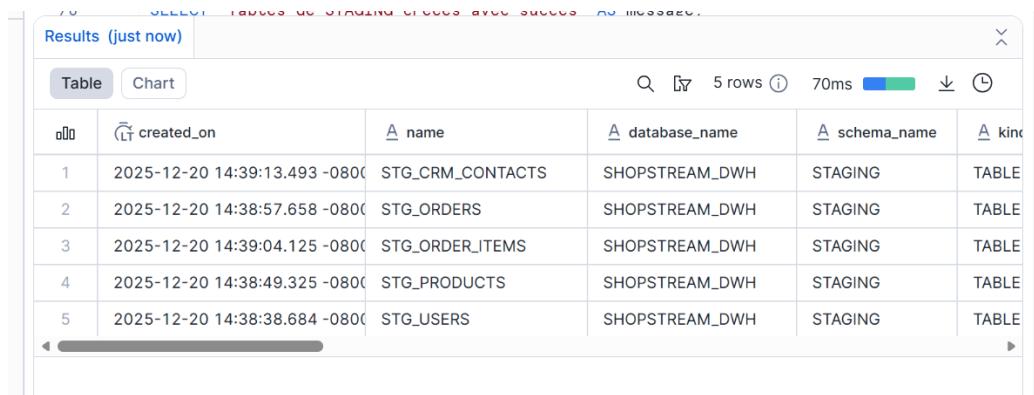


The screenshot shows a SQL editor interface with several tabs at the top: Home, Untitled.sql, Untitled 1.sql, Untitled 2.sql, Untitled 3.sql (which is selected), and a plus sign icon. Below the tabs, the path "My Workspace > Untitled 3.sql" is displayed. The main area contains the following SQL code:

```
12     URL = 's3://shopstream-datalake-khadija/raw/' -- VOTRE BUCKET ICI
13     FILE_FORMAT = (TYPE = CSV FIELD_OPTIONALLY_ENCLOSED_BY = '''' SKIP_HEADER = 1);
14
15     -- Test du Stage : liste des fichiers
16     LIST @s3_raw_stage/postgres/users/;
17
18     -- Si vous voyez vos fichiers CSV, c'est réussi
```

Below the code, a results panel titled "Results (just now)" is shown. It has a "Table" tab selected, displaying a single row of data from a file named "users_20251219.csv". The table has two columns: "name" and "size". The "name" column contains the file path "s3://shopstream-datalake-khadija/raw/postgres/users/2025-12-19/users_20251219.csv", and the "size" column shows "95823".

Figure 1.18: Vérification du Stage externe pointant vers S3



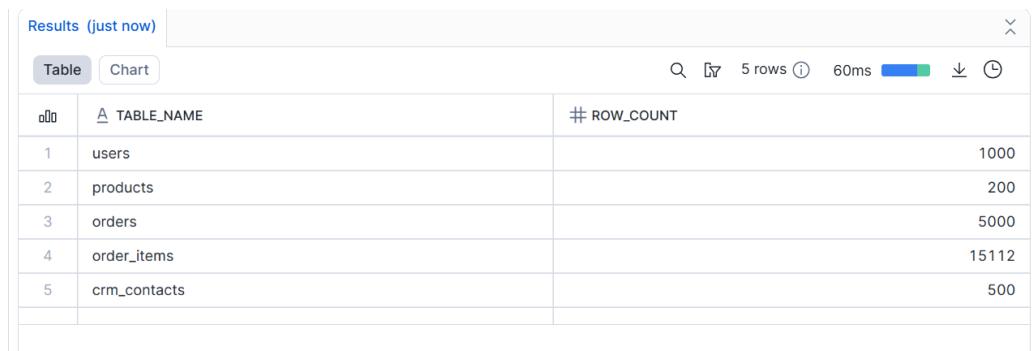
The screenshot shows a SQL editor interface with several tabs at the top: Home, Untitled.sql, Untitled 1.sql, Untitled 2.sql, Untitled 3.sql (which is selected), and a plus sign icon. Below the tabs, the path "My Workspace > Untitled 3.sql" is displayed. The main area contains the following SQL code:

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'users';
```

Below the code, a results panel titled "Results (just now)" is shown. It has a "Table" tab selected, displaying five rows of data from a table named "users". The table has six columns: "created_on", "name", "database_name", "schema_name", "kind", and "type". The data includes:

| | created_on | name | database_name | schema_name | kind |
|---|-------------------------------|------------------|----------------|-------------|-------|
| 1 | 2025-12-20 14:39:13.493 -0800 | STG_CRM_CONTACTS | SHOPSTREAM_DWH | STAGING | TABLE |
| 2 | 2025-12-20 14:38:57.658 -0800 | STG_ORDERS | SHOPSTREAM_DWH | STAGING | TABLE |
| 3 | 2025-12-20 14:39:04.125 -0800 | STG_ORDER_ITEMS | SHOPSTREAM_DWH | STAGING | TABLE |
| 4 | 2025-12-20 14:38:49.325 -0800 | STG_PRODUCTS | SHOPSTREAM_DWH | STAGING | TABLE |
| 5 | 2025-12-20 14:38:38.684 -0800 | STG_USERS | SHOPSTREAM_DWH | STAGING | TABLE |

Figure 1.19: Vérification des tables STAGING dans Snowflake



The screenshot shows a SQL editor interface with several tabs at the top: Home, Untitled.sql, Untitled 1.sql, Untitled 2.sql, Untitled 3.sql (which is selected), and a plus sign icon. Below the tabs, the path "My Workspace > Untitled 3.sql" is displayed. The main area contains the following SQL code:

```
SELECT TABLE_NAME, COUNT(*) AS ROW_COUNT
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE' AND SCHEMA_NAME = 'STAGING';
```

Below the code, a results panel titled "Results (just now)" is shown. It has a "Table" tab selected, displaying five rows of data from a table named "STAGING". The table has two columns: "TABLE_NAME" and "ROW_COUNT". The data includes:

| TABLE_NAME | ROW_COUNT |
|--------------|-----------|
| users | 1000 |
| products | 200 |
| orders | 5000 |
| order_items | 15112 |
| crm_contacts | 500 |

Figure 1.20: Vérification du chargement des données STAGING depuis S3

```
C:\Users\LENOVO>dbt --version
Core:
  - installed: 1.11.2
  - latest:    1.11.2 - Up to date!

Plugins:
  - snowflake: 1.11.0 - Up to date!
```

Figure 1.21: Vérification de l'installation de dbt

```
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>dbt debug
18:10:44 Running with dbt=1.11.2
18:10:44 dbt version: 1.11.2
18:10:44 python version: 3.12.0
18:10:44 python path: C:\Users\LENOVO\AppData\Local\Programs\Python\Python312\python.exe
18:10:44 os info: Windows-11-10.0.26100-SP0
18:10:45 Using profiles dir at C:\Users\LENOVO\.dbt
18:10:45 Using profiles.yml file at C:\Users\LENOVO\.dbt\profiles.yml
18:10:45 Using dbt_project.yml file at C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt\dbt_project.yml
18:10:45 adapter type: snowflake
18:10:45 adapter version: 1.11.0
18:10:45 Configuration:
18:10:45   profiles.yml file [OK found and valid]
18:10:45   dbt_project.yml file [OK found and valid]
18:10:45 Required dependencies:
18:10:45   - git [ERROR]

18:10:45 Connection:
18:10:45   account: qhcvezkr-xk29245
18:10:45   user: KHADJIA
18:10:45   database: SHOPSTREAM_DWH
18:10:45   warehouse: TRANSFORM_WH
18:10:45   role: ACCOUNTADMIN
18:10:45   schema: CORE
18:10:45   authenticator: None
18:10:45   oauth_client_id: None
18:10:45   query_tag: None
18:10:45   client_session_keep_alive: False
18:10:45   host: None
18:10:45   port: None
18:10:45   proxy_host: None
18:10:45   proxy_port: None
```

```
18:10:45 protocol: None
18:10:45 connect_retries: 1
18:10:45 connect_timeout: None
18:10:45 retry_on_database_errors: False
18:10:45 retry_all: False
18:10:45 insecure_mode: False
18:10:45 reuse_connections: True
18:10:45 s3_stage_vpce_dns_name: None
18:10:45 platform_detection_timeout_seconds: 0.0
18:10:45 Registered adapter: snowflake=1.11.0
18:10:47 Connection test: [OK connection ok]
```

Figure 1.22: Test de connexion dbt avec Snowflake

| Étape | Description |
|--|---|
| 8.3 Configuration du projet dbt | Configuration du projet via le fichier <code>dbt_project.yml</code> : dossiers, matérialisation (view/table), schémas, variables globales. |
| Structure de dossiers | Organisation des modèles en <code>staging</code> , <code>core</code> (dimensions + faits) et <code>marts</code> pour un projet modulable et clair. |
| 8.4 Modèles de staging | Déclaration des sources dans <code>_staging__sources.yml</code> et création de vues SQL nettoyées comme <code>stg_orders.sql</code> . |
| 8.5 Dimensions | Création des tables dimensionnelles : <code>dim_customers.sql</code> (clients) et <code>dim_products.sql</code> (produits) avec métriques et segmentations. |
| 8.6 Table de faits | Création de <code>fact_orders.sql</code> : granularité ligne de commande, clés étrangères vers les dimensions, métriques et flags. |
| 8.7 Data Mart | Création de <code>mart_sales_overview.sql</code> : agrégations par jour, pays, catégorie, et métriques clés pour reporting/BI. |

Table 1.1: Récapitulatif des étapes de configuration et création des modèles dans dbt

1.10 Exécution de dbt

1.10.1 Exécution des modèles dbt

Objectif : Exécuter tous les modèles dbt et créer les tables et vues dans Snowflake.

1. Dans votre terminal, assurez-vous d'être dans le dossier dbt :

```
cd ~/ShopStreamTP/shopstream_dbt
```

2. Exédez tous les modèles :

```
dbt run
```

3. dbt compile et exécute les modèles dans l'ordre des dépendances.

Résultat attendu : Tous les modèles sont créés dans Snowflake, par exemple :

- `stg_orders` (vue)
- `dim_customers` (table)
- `dim_products` (table)
- `fact_orders` (table)
- `mart_sales_overview` (table)

```
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>dbt run
18:36:15  Running with dbt=1.11.2
18:36:16  Registered adapter: snowflake=1.11.0
18:36:17  Found 5 models, 13 data tests, 4 sources, 520 macros
18:36:17  Concurrency: 4 threads (target='dev')
18:36:17
18:36:20  1 of 5 START sql table model CORE_core.dim_customers ..... [RUN]
18:36:20  2 of 5 START sql table model CORE_core.dim_products ..... [RUN]
18:36:20  3 of 5 START sql table model CORE_core.fact_orders ..... [RUN]
18:36:20  4 of 5 START sql view model CORE_staging.stg_orders ..... [RUN]
18:36:21  4 of 5 OK created sql view model CORE_staging.stg_orders ..... [SUCCESS 1 in 1.11s]
18:36:22  2 of 5 OK created sql table model CORE_core.dim_products ..... [SUCCESS 1 in 2.05s]
18:36:22  1 of 5 OK created sql table model CORE_core.dim_customers ..... [SUCCESS 1 in 2.06s]
18:36:22  3 of 5 OK created sql table model CORE.core.fact_orders ..... [SUCCESS 1 in 2.13s]
18:36:22  5 of 5 START sql table model CORE_marts.mart_sales_overview ..... [RUN]
18:36:23  5 of 5 OK created sql table model CORE_marts.mart_sales_overview ..... [SUCCESS 1 in 1.19s]
18:36:26
18:36:26  Finished running 4 table models, 1 view model in 0 hours 0 minutes and 8.77 seconds (8.77s).
18:36:26
18:36:26  Completed successfully
18:36:26
18:36:26  Done. PASS=5 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=5
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>
```

Figure 1.23: Exécution des modèles dbt avec succès

1.10.2 Exécution des tests dbt

Objectif : Vérifier la qualité des données avec les tests définis dans les fichiers YAML.

dbt test

```
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>dbt test
18:42:39  Running with dbt=1.11.2
18:42:40  Registered adapter: snowflake=1.11.0
18:42:41  Found 5 models, 13 data tests, 4 sources, 520 macros
18:42:41  Concurrency: 4 threads (target='dev')
18:42:41
18:42:43  1 of 13 START test source_not_null_staging_stg_order_items_id ..... [RUN]
18:42:43  2 of 13 START test source_not_null_staging_stg_orders_id ..... [RUN]
18:42:43  3 of 13 START test source_not_null_staging_stg_orders_user_id ..... [RUN]
18:42:43  4 of 13 START test source_not_null_staging_stg_products_id ..... [RUN]
18:42:43  2 of 13 PASS source_not_null_staging_stg_orders_id ..... [PASS in 1.58s]
18:42:44  3 of 13 PASS source_not_null_staging_stg_orders_user_id ..... [PASS in 1.58s]
18:42:44  1 of 13 PASS source_not_null_staging_stg_order_items_id ..... [PASS in 1.59s]
18:42:44  5 of 13 START test source_not_null_staging_stg_users_email ..... [RUN]
18:42:44  6 of 13 START test source_not_null_staging_stg_users_id ..... [RUN]
18:42:44  7 of 13 START test source_relationships_staging_stg_order_items_order_id_id_source_staging_stg_orders_ ..... [RUN]
18:42:44  6 of 13 PASS source_not_null_staging_stg_users_id ..... [PASS in 0.19s]
18:42:44  5 of 13 PASS source_not_null_staging_stg_users_email ..... [PASS in 0.19s]
18:42:44  8 of 13 START test source_relationships_staging_stg_orders_user_id_id_source_staging_stg_users_ ..... [RUN]
18:42:44  9 of 13 START test source_unique_staging_stg_order_items_id ..... [RUN]
18:42:45  7 of 13 PASS source_relationships_staging_stg_order_items_order_id_id_source_staging_stg_orders_ ..... [PASS in 0.34s]
18:42:45  10 of 13 START test source_unique_staging_stg_orders_id ..... [RUN]
18:42:45  9 of 13 PASS source_unique_staging_stg_order_items_id ..... [PASS in 0.17s]
18:42:45  11 of 13 START test source_unique_staging_stg_products_id ..... [RUN]
18:42:45  8 of 13 PASS source_relationships_staging_stg_orders_user_id_id_source_staging_stg_users_ ..... [PASS in 0.24s]
18:42:45  12 of 13 START test source_unique_staging_stg_users_email ..... [RUN]
18:42:45  4 of 13 PASS source_not_null_staging_stg_products_id ..... [PASS in 2.06s]
18:42:45  13 of 13 START test source_unique_staging_stg_users_id ..... [RUN]
18:42:45  10 of 13 PASS source_unique_staging_stg_orders_id ..... [PASS in 0.18s]
18:42:45  12 of 13 PASS source_unique_staging_stg_users_email ..... [PASS in 0.17s]
18:42:45  11 of 13 PASS source_unique_staging_stg_products_id ..... [PASS in 0.24s]
18:42:45  13 of 13 PASS source_unique_staging_stg_users_id ..... [PASS in 0.20s]
```

Figure 1.24: Exécution des tests de qualité des données

Résultat attendu : Tous les tests passent avec succès. Exemples :

- not_null_stg_users_id
- unique_stg_users_email
- relations entre commandes et utilisateurs

1.10.3 Génération de la documentation

Objectif : Générer et visualiser la documentation du projet dbt.

1. Génération de la documentation :

```
dbt docs generate
```

2. Lancement du serveur de documentation :

```
dbt docs serve
```

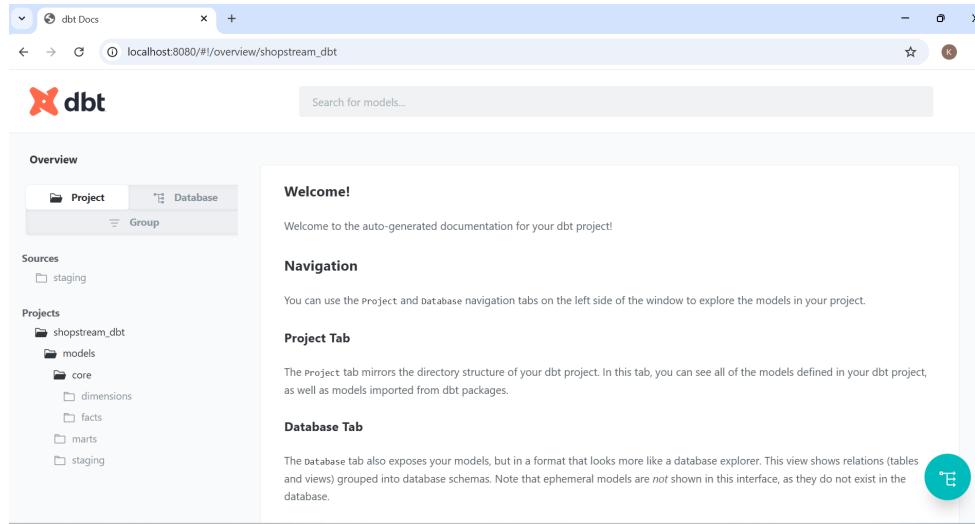


Figure 1.25: Documentation complète du projet dbt (lineage graph, description des modèles, tests, code SQL)

1.10.4 Vérification dans Snowflake

Objectif : Vérifier que les transformations dbt sont bien présentes dans Snowflake.

```
USE DATABASE SHOPSTREAM_DWH;
USE SCHEMA MARTS;
```

```
SELECT * FROM mart_sales_overview
LIMIT 10;
```

1.10.5 Récapitulatif dbt

Après ces étapes, vous avez réussi à :

- Installer dbt avec l'adaptateur Snowflake
- Configurer la connexion Snowflake
- Créeer des modèles de staging
- Créeer des dimensions (dim_customers, dim_products)

The screenshot shows a Snowflake query results interface. At the top, there's a header bar with 'Results (just now)' and various icons for search, refresh, and settings. Below it is a toolbar with 'Table' and 'Chart' buttons, and a progress bar indicating '30ms'. The main area is a table with the following data:

| # | SALE_DATE | COUNTRY_CODE | PRODUCT_CATEGORY | CUSTOMER_SEGMENT | TOTAL_ORDERS | UNIQUE_CUSTOMERS |
|----|------------|--------------|------------------|------------------|--------------|------------------|
| 1 | 2025-12-05 | ESP | Fashion | Freemium | 151 | 100 |
| 2 | 2025-12-05 | ESP | Home | Freemium | 168 | 100 |
| 3 | 2025-12-05 | DEU | Toys | Premium | 34 | 100 |
| 4 | 2025-12-05 | ITA | Electronics | Premium | 51 | 100 |
| 5 | 2025-12-05 | ESP | Sports | Premium | 31 | 100 |
| 6 | 2025-12-05 | GBR | Beauty | Freemium | 145 | 100 |
| 7 | 2025-12-05 | CAN | Toys | Freemium | 181 | 100 |
| 8 | 2025-12-05 | CAN | Books | Freemium | 113 | 100 |
| 9 | 2025-12-05 | AUS | Electronics | Freemium | 183 | 100 |
| 10 | 2025-12-05 | GBR | Sports | Freemium | 130 | 100 |

Figure 1.26: Vérification des données agrégées dans Snowflake

- Créer une table de faits (fact_orders)
- Créer un data mart (mart_sales_overview)
- Exécuter les transformations avec dbt run
- Tester la qualité des données avec dbt test
- Générer la documentation avec dbt docs

1.11 Création des Data Marts

Contexte : Nous restons sur l'étape dbt pour créer des data marts supplémentaires. Ces tables sont optimisées pour répondre à des questions métier spécifiques.

PostgreSQL (OLTP) → S3 (Data Lake) → Snowflake (DWH) → dbt (Transform)
→ Airflow (Orchestration) → Power BI (BI)

1.11.1 Data Mart : Performance Produits

Objectif : Créer le mart mart_product_performance pour analyser la performance des produits (analyse ABC).

1. Créez le fichier models/marts/mart_product_performance.sql et collez le code SQL fourni.
2. Exéutez dbt pour créer ce mart :

```
dbt run --select mart_product_performance
```

Résultat attendu : La table est créée dans Snowflake et contient :

- Total de commandes par produit

```
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>dbt run --select mart_product_performance
12:05:16  Running with dbt=1.11.2
12:05:23  Registered adapter: snowflake=1.11.0
12:05:28  Found 6 models, 13 data tests, 4 sources, 520 macros
12:05:28
12:05:28  Concurrency: 4 threads (target='dev')
12:05:28
12:05:38  1 of 1 START sql table model CORE_marts.mart_product_p
erformance ..... [RUN]
12:05:41  1 of 1 OK created sql table model CORE_marts.mart_prod
uct_performance ..... [SUCCESS 1 in 2.73s]
12:05:42
12:05:42  Finished running 1 table model in 0 hours 0 minutes an
d 13.94 seconds (13.94s).
12:05:42
12:05:42  Completed successfully
12:05:42
12:05:42  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=1

C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>
```

Figure 1.27: Exécution du mart `mart_product_performance` dans dbt

- Quantité totale vendue
- Revenus totaux et marge
- Classement par revenu et classification ABC
- Performance (Top 10, Top 50, Long Tail)

1.11.2 Data Mart : Customer Lifetime Value (CLV)

Objectif : Créer le mart `mart_customer_ltv` pour analyser la valeur à vie des clients et la segmentation RFM.

1. Créez le fichier `models/marts/mart_customer_ltv.sql` et collez le code SQL fourni.
2. Exéutez dbt pour créer ce mart :

```
dbt run --select mart_customer_ltv
```

Résultat attendu : La table est créée dans Snowflake et contient :

- Dernière commande et récence (Recency)
- Fréquence des commandes (Frequency)
- Montant total et moyen (Monetary)
- Scores RFM (1-5) et segmentation
- Risque de churn (Low, Medium, High)

```
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>dbt run --select mart_customer_ltv
12:08:59  Running with dbt=1.11.2
12:09:00  Registered adapter: snowflake=1.11.0
12:09:02  Found 7 models, 13 data tests, 4 sources, 520 macros
12:09:02
12:09:02  Concurrency: 4 threads (target='dev')
12:09:02
12:09:04  1 of 1 START sql table model CORE_marts.mart_customer_
12:09:04  ltv ..... [RUN]
12:09:07  1 of 1 OK created sql table model CORE_marts.mart_cust
12:09:07  omer_ltv ..... [SUCCESS 1 in 2.24s]
12:09:07
12:09:07  Finished running 1 table model in 0 hours 0 minutes an
12:09:07  d 5.77 seconds (5.77s).
12:09:07
12:09:07  Completed successfully
12:09:07
12:09:07  Done. PASS=1 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=1
C:\Users\LENOVO\Desktop\ShopStreamTP\shopstream_dbt>
```

Figure 1.28: Exécution du mart `mart_customer_ltv` dans dbt

1.11.3 Résumé Data Marts

Après ces étapes, vous avez réussi à :

- Créer le mart `mart_product_performance` pour analyser les produits
- Créer le mart `mart_customer_ltv` pour analyser les clients et la segmentation RFM
- Exécuter les transformations avec `dbt run -select <mart>`
- Vérifier la présence des données dans Snowflake

1.12 Connexion Power BI et création de dashboards

Contexte : Nous arrivons à la dernière étape technique : Power BI. Les données sont maintenant transformées et disponibles dans les data marts de Snowflake. Nous allons les connecter à Power BI pour créer des dashboards interactifs destinés aux utilisateurs métier.

PostgreSQL (OLTP) → S3 (Data Lake) → Snowflake (DWH) → dbt (Transform)
→ Airflow (Orchestration) → Power BI (BI)

1.12.1 Pourquoi Power BI ?

Power BI est l'outil de BI de Microsoft, largement adopté en entreprise. Il offre une intégration native avec l'écosystème Microsoft et une courbe d'apprentissage douce pour les utilisateurs métier.

| Outil | Avantages | Inconvénients |
|----------|---|--------------------------------------|
| Power BI | Intégration Microsoft, prix attractif, DAX puissant | Moins flexible que Tableau |
| Tableau | Visualisations riches, communauté active | Coût élevé |
| Looker | LookML (code-first), excellent lineage | Nécessite des compétences techniques |
| Metabase | Open-source, simple, gratuit | Moins de fonctionnalités |

1.12.2 Connexion à Snowflake

1. Dans Power BI Desktop, allez sur "Accueil" → "Obtenir des données" → "Snowflake".
2. Remplissez Serveur et Warehouse (ex : BI_WH) et cliquez sur OK.
3. Authentifiez-vous avec vos identifiants Snowflake.
4. Sélectionnez les bases de données et tables de Data Marts à importer.

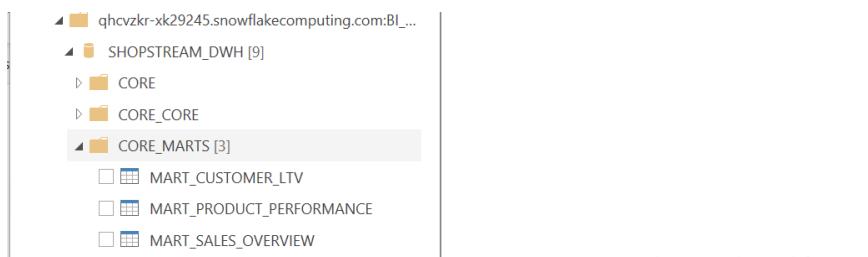


Figure 1.29: Connexion à Snowflake depuis Power BI

1.12.3 Crédit au dashboard "Vue d'ensemble des ventes"

Étape 4 : Création de la page de rapport

1. Dans la vue "Rapport" de Power BI, double-cliquez sur "Page 1" en bas et renommez-la en "Vue d'ensemble".

Étape 5 : Ajout d'une carte KPI - Chiffre d'affaires total

1. Cliquez sur l'icône "Carte" dans le panneau Visualisations.
2. Cochez le champ TOTAL_REVENU dans MART_SALES_OVERVIEW.
3. Formatez le montant en € avec 0 décimale.

Étape 6 : Ajout d'une carte KPI - Nombre de commandes

1. Cliquez sur une zone vide et sur l'icône "Carte".
2. Cochez le champ TOTAL_ORDERS dans MART_SALES_OVERVIEW.

Étape 7 : Ajout d'un graphique en courbes - Évolution du CA

1. Cliquez sur "Graphique en courbes" dans Visualisations.

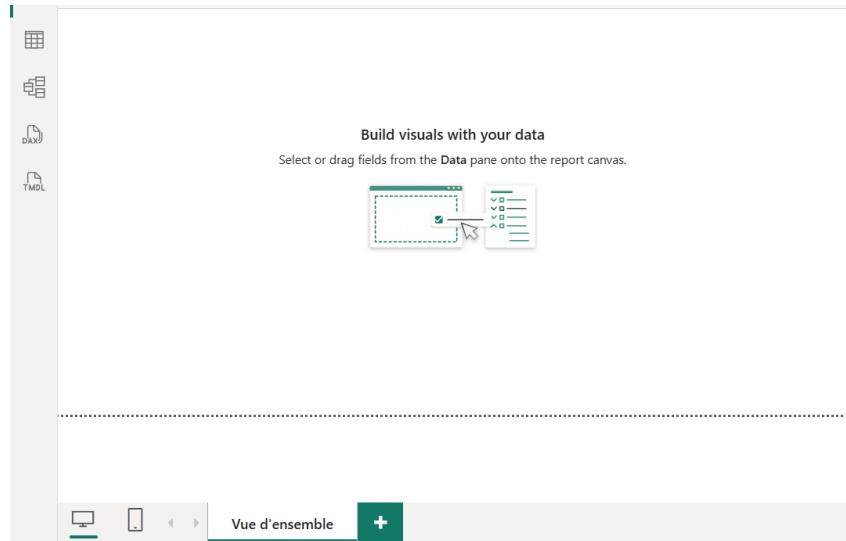


Figure 1.30: Renommage de la page en "Vue d'ensemble"



Figure 1.31: Carte KPI : Chiffre d'affaires total

2. Déposez SALE_DATE dans Axe X et TOTAL_REVENUE dans Axe Y.

Étape 8 : Ajout d'un graphique en barres - CA par pays

1. Cliquez sur "Graphique à barres empilées".

2. Déposez COUNTRY_CODE dans Axe Y et TOTAL_REVENUE dans Axe X.

Étape 9 : Ajout d'un graphique en secteurs - CA par catégorie

1. Cliquez sur "Graphique en secteurs".

2. Déposez PRODUCT_CATEGORY dans Légende et TOTAL_REVENUE dans Valeurs.

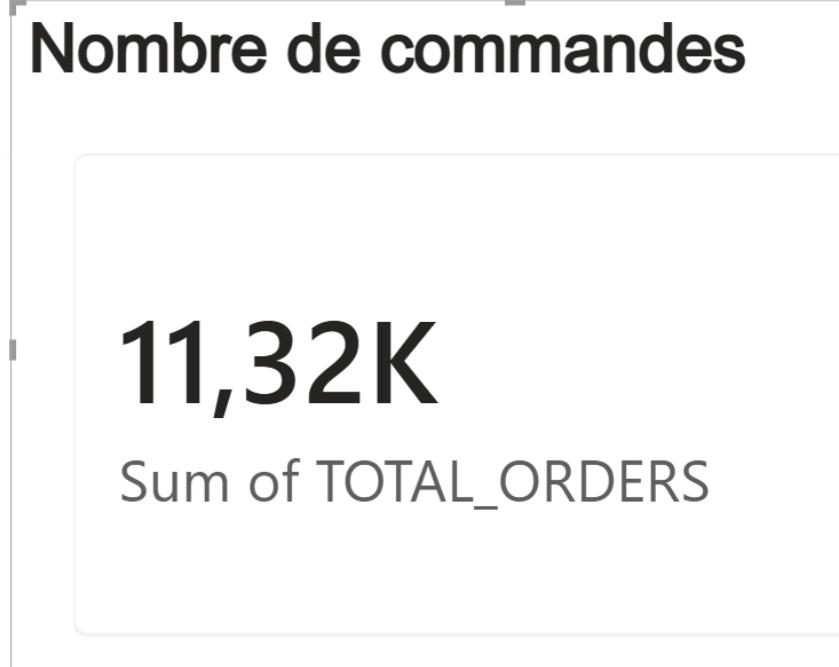


Figure 1.32: Carte KPI : Nombre de commandes

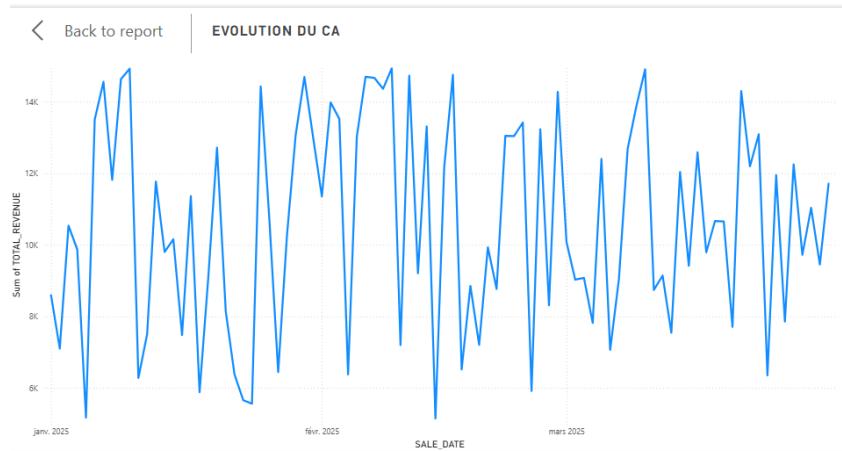


Figure 1.33: Graphique en courbes : Évolution du chiffre d'affaires

1.12.4 Crédation de mesures DAX

Étape 10 : Crédation de la mesure "Panier moyen"

1. Clic droit sur MART_SALES_OVERVIEW → Nouvelle mesure.
2. Tapez la formule DAX suivante :

```
Panier Moyen = DIVIDE(
    SUM(MART_SALES_OVERVIEW[TOTAL_REVENUE]),
    SUM(MART_SALES_OVERVIEW[TOTAL_ORDERS])
)
```

Étape 11 : Crédation de la mesure "CA du mois précédent"

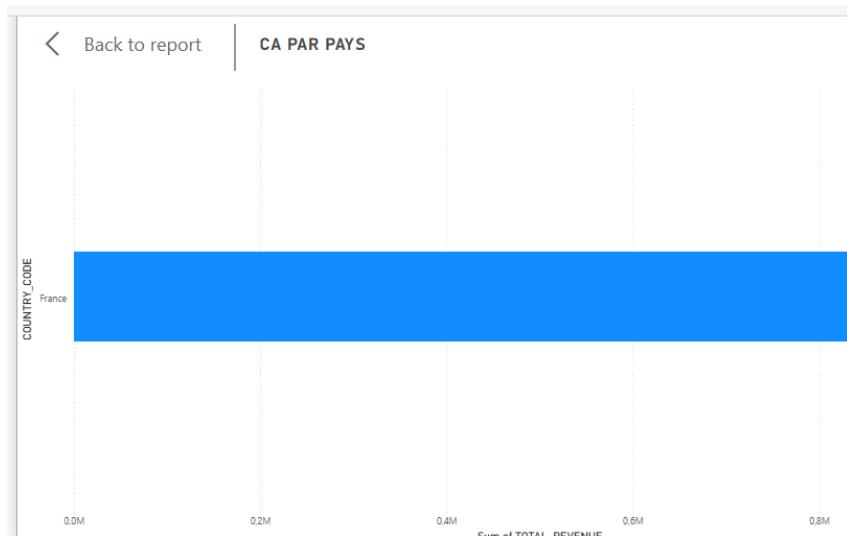


Figure 1.34: Graphique en barres : CA par pays

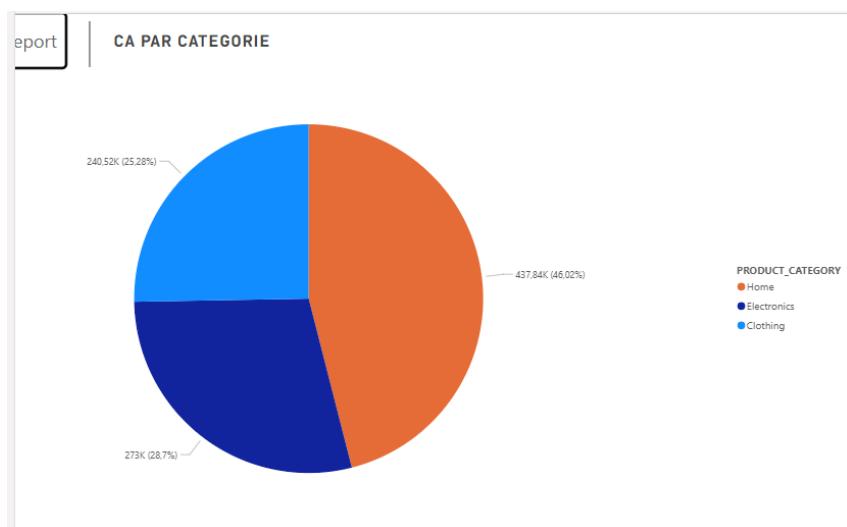


Figure 1.35: Graphique en secteurs : CA par catégorie

1. Nouvelle mesure DAX :

```
CA Mois Précédent = CALCULATE(
    SUM(MART_SALES_OVERVIEW[TOTAL_REVENUE]),
    DATEADD(MART_SALES_OVERVIEW[SALE_DATE], -1, MONTH)
)
```

Étape 12 : Création de la mesure "Croissance mensuelle"

1. Nouvelle mesure DAX :

```
Croissance Mensuelle = DIVIDE(
    SUM(MART_SALES_OVERVIEW[TOTAL_REVENUE]) - [CA Mois Précédent],
    [CA Mois Précédent],
    0
)
```

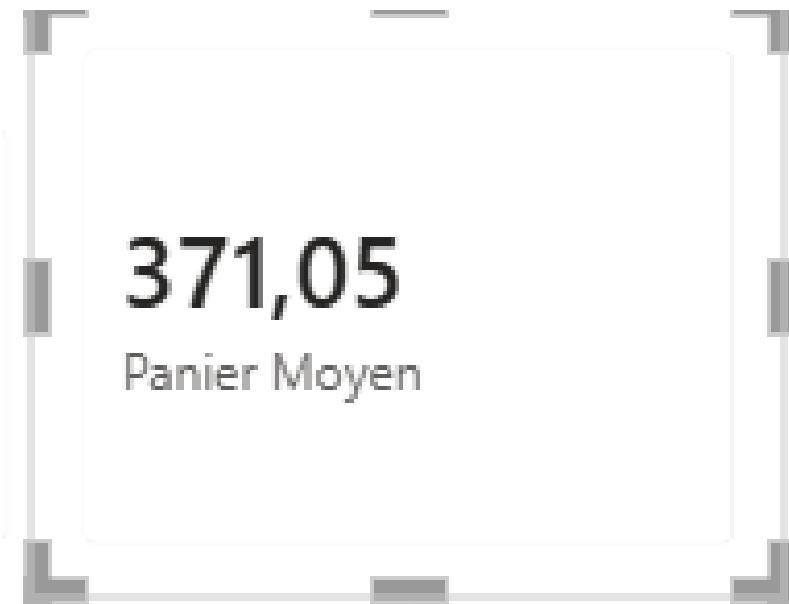


Figure 1.36: Création de la mesure DAX : Panier Moyen

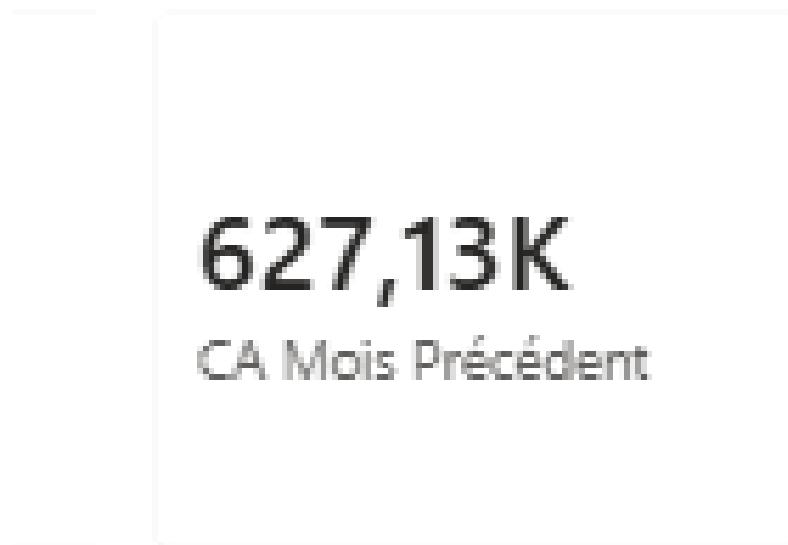


Figure 1.37: Création de la mesure DAX : CA du mois précédent

2. Formatez la mesure en pourcentage.

1.12.5 Crédation d'une deuxième page : Performance Produits

Étape 13 : Crédation de la page "Performance Produits"

1. Cliquez sur le bouton "+" en bas de l'écran pour ajouter une nouvelle page.
2. Renommez la page en "Performance Produits".

51,70 %
Croissance Mensuelle

Figure 1.38: Création de la mesure DAX : Croissance mensuelle

Création du tableau avec MART_PRODUCT_PERFORMANCE

1. Dans "Visualisations", cliquez sur "Tableau".
2. Ajoutez les colonnes : PRODUCT_NAME, PRODUCT_CATEGORY, TOTAL_REVENUE, TOTAL_ORDERS, ABC_CLASS, PERFORMANCE_TIER.
3. Redimensionnez le tableau pour qu'il occupe toute la page.
4. Triez le tableau par TOTAL_REVENUE décroissant.

| PRODUCT_CATEGORY | PRODUCT_NAME | Sum of TOTAL_REVENUE | Sum of TOTAL_ORDERS | ABC_CLASS |
|------------------|---|----------------------|---------------------|-----------|
| Books | Visionary responsive system engine | 80 304,24 | 82,00 A | |
| Home | L'avantage d'avancer à sa source | 79 539,90 | 81,00 A | |
| Home | Versatile composite architecture | 77 518,30 | 83,00 A | |
| Fashion | Organic contextually-based matrix | 76 683,12 | 81,00 A | |
| Fashion | Configurable multimedia groupware | 74 518,08 | 73,00 A | |
| Home | User-centric static utilization | 70 707,90 | 71,00 A | |
| Sports | Down-sized next generation structure | 70 551,81 | 79,00 A | |
| Fashion | Total stable architecture | 68 035,38 | 74,00 A | |
| Electronics | Virtual secondary framework | 68 013,30 | 70,00 A | |
| Toys | L'avantage de changer à sa source | 67 578,70 | 68,00 A | |
| Electronics | Synergized methodical approach | 67 106,88 | 70,00 A | |
| Beauty | Le plaisir de louer à sa source | 66 795,72 | 71,00 A | |
| Fashion | Quality-focused bandwidth-monitored task-force | 66 654,45 | 68,00 A | |
| Beauty | Intuitive 4th-generation functionalities | 66 142,40 | 69,00 A | |
| Books | Focused dedicated hardware | 65 685,96 | 75,00 A | |
| Home | Devolved 4th-generation approach | 65 507,04 | 73,00 A | |
| Toys | Front-line dynamic projection | 65 341,90 | 70,00 A | |
| Beauty | Innovative fresh-thinking neural-net | 64 183,24 | 70,00 A | |
| Beauty | L'assurance de rouler autrement | 63 646,71 | 76,00 A | |
| Toys | Function-based coherent software | 62 730,72 | 69,00 A | |
| Sports | Progressive grid-enabled moratorium | 62 267,35 | 74,00 A | |
| Books | De-engineered secondary workforce | 62 125,96 | 76,00 A | |
| Sports | Vision-oriented tertiary application | 61 294,01 | 61,00 A | |
| Toys | La liberté d'avancer à l'état pur | 60 088,20 | 65,00 A | |
| Sports | Synergistic cohesive framework | 59 994,48 | 81,00 A | |
| Toys | Customer-focused next generation contingency | 59 816,80 | 73,00 A | |
| Toys | L'assurance d'atteindre vos buts à l'état pur | 59 293,76 | 66,00 A | |
| Beauty | Team-oriented even keeled paradigm | 59 136,22 | 77,00 A | |
| Electronics | Reverse-engineered bandwidth-monitored time-frame | 58 887,24 | 71,00 A | |
| Total | | 7 320 915,45 | 13 637,00 | |

Figure 1.39: Tableau des performances produits

1.12.6 Crédation d'une troisième page : Analyse Clients

Étape 14 : Crédation de la page "Analyse Clients"

1. Ajoutez une nouvelle page.
2. Renommez-la en "Analyse Clients".

Crédation d'un graphique en anneau - Segmentation RFM

1. Dans "Visualisations", cliquez sur "Graphique en anneau".
2. Légende : RFM_SEGMENT
3. Valeurs : Count of CUSTOMER_KEY
4. Positionnez le graphique en haut à gauche de la page.

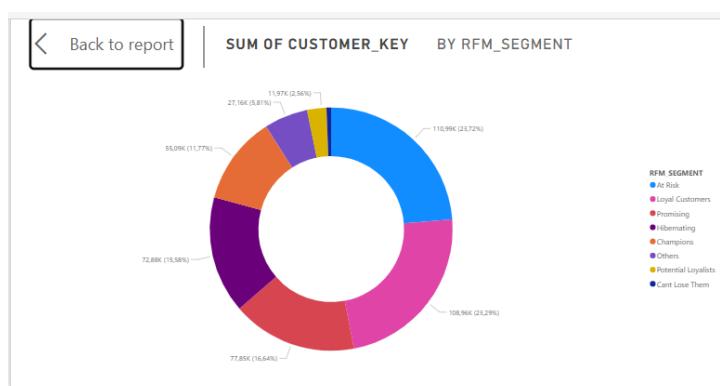


Figure 1.40: Graphique en anneau : segmentation RFM

Crédation d'un graphique en barres empilées - Valeur à vie par segment

1. Dans "Visualisations", cliquez sur "Graphique à barres empilées".
2. Axe Y : RFM_SEGMENT
3. Axe X : LIFETIME_VALUE (somme)
4. Positionnez le graphique en haut à droite de la page.

Crédation du tableau des top clients

1. Dans "Visualisations", cliquez sur "Tableau".
2. Ajoutez les colonnes : FULL_NAME, EMAIL, COUNTRY_CODE, LIFETIME_VALUE, TOTAL_ORDERS, RFM_SEGMENT, CHURN_RISK.
3. Positionnez le tableau en bas de la page.
4. Triez par LIFETIME_VALUE décroissant.

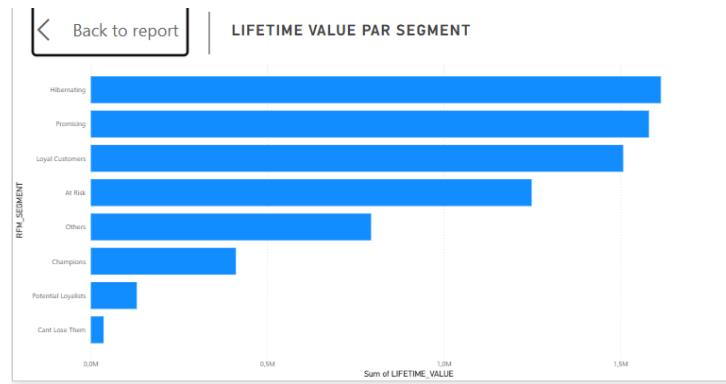


Figure 1.41: Graphique en barres empilées : valeur à vie par segment

| FULL NAME | EMAIL | COUNTRY_CODE | Sum of LIFETIME VALUE | Sum of TOTAL_ORDERS | RFM SEGMENT | CHURN RISK |
|------------------|--------------------------------|--------------|-----------------------|---------------------|-----------------|------------|
| Idrienne Bonneau | sylviebertrand@example.net | DEU | 10 228,26 | 6,00 | Promising | LOW |
| Idrienne Ramirez | reyessidora@example.org | DEU | 6 240,08 | 6,00 | Promising | LOW |
| Ifrica Weaver | roselliane@example.com | GBR | 9 734,14 | 7,00 | Hibernating | LOW |
| Ignathe Bender | christellenod@example.com | AUS | 11 058,65 | 6,00 | Hibernating | LOW |
| Ignès Bonnin | hallen@example.net | AUS | 9 027,82 | 6,00 | Hibernating | LOW |
| Ignès Brooks | gjulia@example.com | CAN | 10 044,43 | 5,00 | Hibernating | LOW |
| Urimé Hänel | bernhardrosenow@example.com | AUS | 13 532,57 | 7,00 | Hibernating | LOW |
| Umée Menendez | mahemangot@example.com | ESP | 15 613,57 | 11,00 | Promising | LOW |
| Umoa Brown | manola34@example.net | GBR | 11 395,01 | 5,00 | Loyal Customers | LOW |
| Utna Moon | cmaury@example.net | DEU | 7 260,44 | 6,00 | Hibernating | LOW |
| Ubano Young | operrot@example.com | CAN | 11 737,55 | 6,00 | Hibernating | LOW |
| Uberto Müller | jeronome3@example.org | AUS | 5 375,07 | 4,00 | Champions | LOW |
| Ubino Barba | nicolasebastien@example.org | AUS | 2 332,58 | 2,00 | Loyal Customers | LOW |
| Ubino Pons | suekrue07@example.org | CAN | 8 180,23 | 6,00 | Hibernating | LOW |
| Uejandro Bonbach | mrkranz@example.net | ESP | 6 097,96 | 6,00 | Promising | LOW |
| Ulex Chauvet | ahmed05@example.org | CAN | 8 872,38 | 5,00 | Loyal Customers | LOW |
| Ulex Couturier | stephanegonzalez@example.com | CAN | 6 714,49 | 4,00 | Loyal Customers | LOW |
| Ulex Johnson | joanna84@example.org | FRA | 8 893,57 | 4,00 | Loyal Customers | LOW |
| Ulex Levy | brianmeyers@example.com | DEU | 2 130,52 | 3,00 | Champions | LOW |
| Ulex Mora | lilianegnatz@example.net | ITA | 11 779,50 | 7,00 | Promising | LOW |
| Ulexander Allen | hectorsacristan@example.net | DEU | 7 346,15 | 6,00 | Promising | LOW |
| Ulexander Sureda | elke41@example.net | GBR | 19 951,53 | 13,00 | Others | LOW |
| Ulexander Techer | barthelemylaurence@example.com | ITA | 9 010,07 | 5,00 | Loyal Customers | LOW |
| Total | | | 7 320 915,45 | 4 515,00 | | |

Figure 1.42: Tableau des top clients : analyse clients

1.12.7 Récapitulatif

- Installation de Power BI Desktop
- Connexion à Snowflake
- Import des tables data marts
- Crédation de 3 dashboards :
 - Vue d'ensemble des ventes
 - Performance produits
 - Analyse clients
- Crédation de mesures DAX
- Ajout de filtres interactifs
- Mise en forme conditionnelle
- Publication et rafraîchissement automatique