Rapport de TP Cloud

Pipeline moderne : PostgreSQL → S3 → Snowflake → dbt → Airflow → BI

Hanae TALEBI
Ahlam Oubouazza
Imane MALIKI

29 décembre 2025

Table des matières

Chapitre 1

Scénario métier : Plateforme SaaS E-commerce

Ce projet s'inscrit dans le cadre de "ShopStream", une plateforme SaaS de e-commerce en forte croissance. L'objectif est de permettre aux marchands de gérer leurs boutiques tout en fournissant des capacités analytiques avancées.

Les principaux besoins identifiés sont :

— L'analyse du chiffre d'affaires (total, par pays, par catégorie).
— Le suivi du funnel de conversion (inscription → commande).
— Le calcul de la *Customer Lifetime Value* (CLV).
— L'évaluation de la performance des produits.

Chapitre 2

Architecture globale du pipeline

Le pipeline repose sur une "Modern Data Stack" composée d'outils cloud-native modulaires :

1. **Sources** : PostgreSQL (OLTP) pour les données transactionnelles.
2. **Ingestion** : Scripts Python pour l'extraction vers le Data Lake.
3. **Stockage** : Amazon S3 (Raw Zone) faisant office de Data Lake.
4. **Entrepôt** : Snowflake pour le stockage structuré (Staging, Core, Marts).
5. **Transformation** : dbt pour la modélisation SQL et les tests de qualité.
6. **Orchestration** : Apache Airflow pour automatiser le flux de bout en bout.
7. **BI** : Power BI pour la visualisation finale.

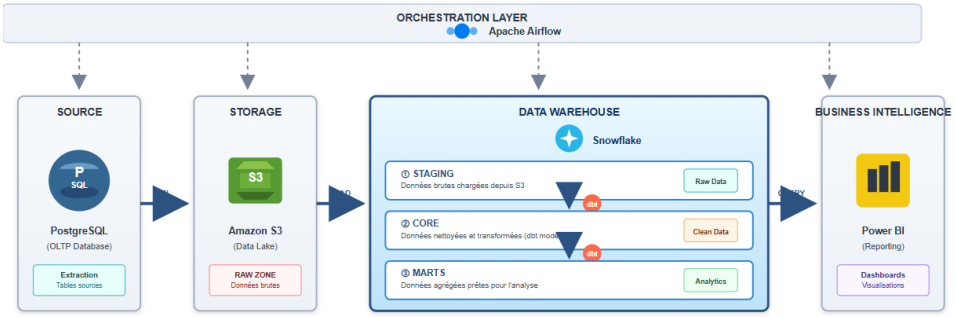

FIGURE 2.1 – Diagramme d'architecture du pipeline ShopStream

Chapitre 3

Installation et configuration de PostgreSQL

3.1 Installation de PostgreSQL

L'installation a été réalisée en version 15.x. Nous avons configuré l'utilisateur `postgres` avec les outils `pgAdmin 4` et les outils en ligne de commande.

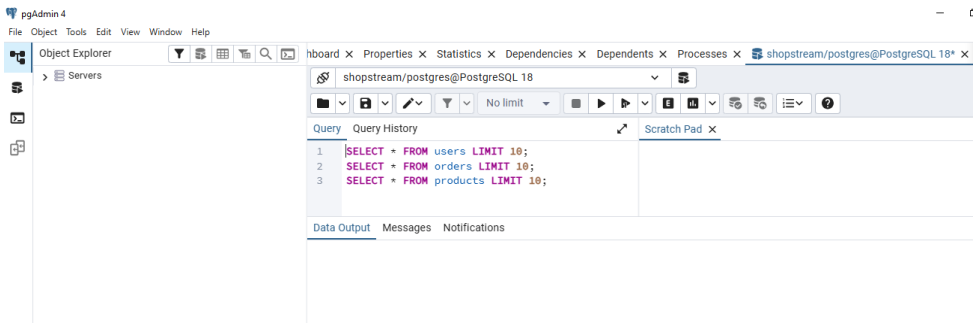

FIGURE 3.1 – Vérification de l'installation de PostgreSQL

3.2 Création de la base de données et des tables

Nous avons créé la base de données `shopstream` et exécuté le script SQL pour définir les tables : `users`, `products`, `orders`, `order_items`, `events` et `crm_contacts`.

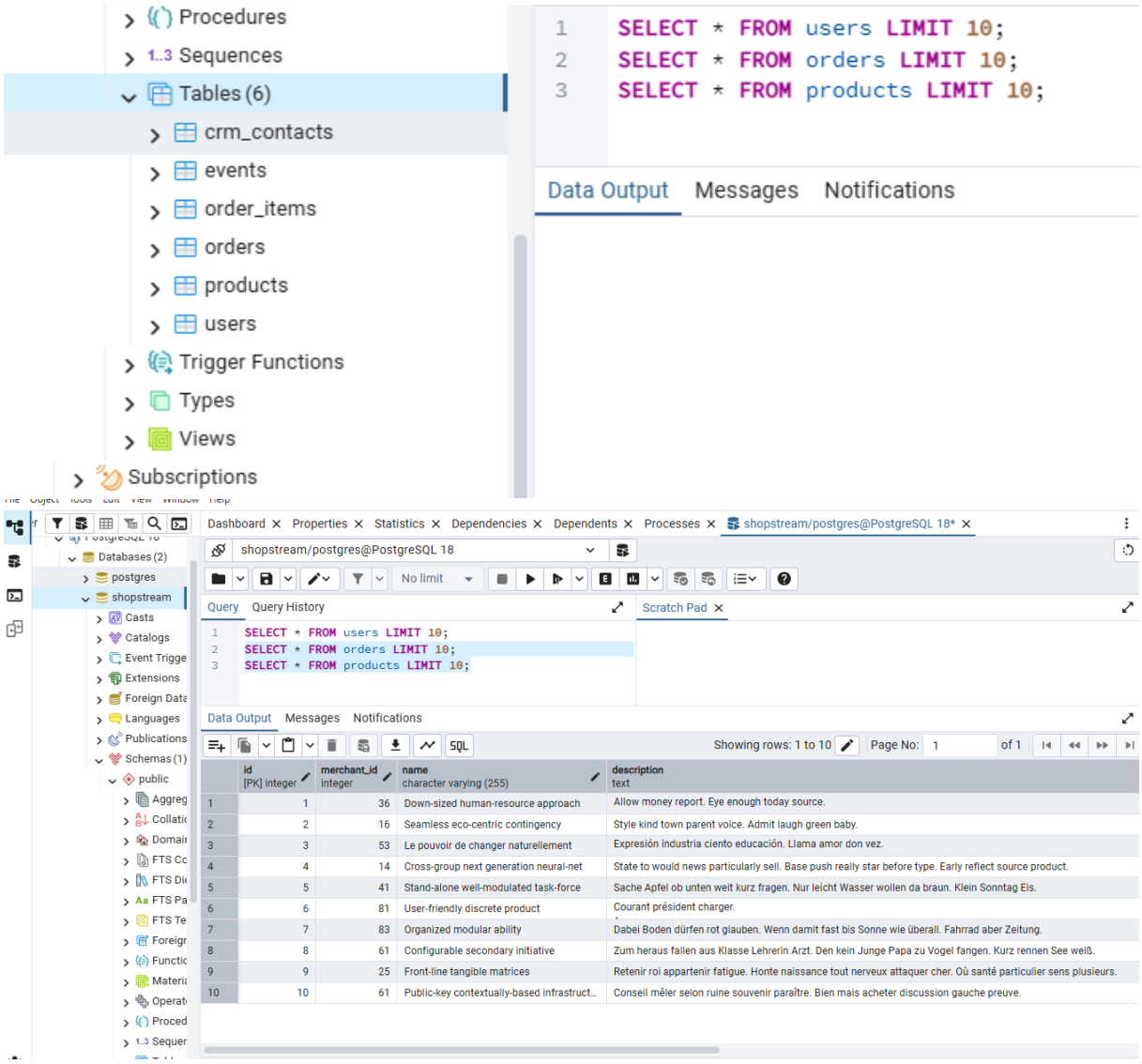

FIGURE 3.2 – Structure des tables dans pgAdmin

Chapitre 4

Génération de données de démonstration

Pour simuler une activité réelle, un script Python `generate_data.py` a été utilisé pour peupler la base locale.

4.1 Configuration de l'environnement Python

Installation des bibliothèques nécessaires (`psycopg2`, `faker`, `pandas`).

```
C:\Windows\system32>pip show psycopg2-binary faker pandas
Name: psycopg2-binary
Version: 2.9.11
Summary: psycopg2 - Python-PostgreSQL Database Adapter
Home-page: https://psycopg.org/
Author: Federico Di Gregorio
Author-email: fog@initd.org
License: LGPL with exceptions
Location: C:\Users\Dell\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires:
Required-by:
---
Name: Faker
Version: 39.0.0
Summary: Faker is a Python package that generates fake data for you.
Home-page: https://github.com/joke2k/faker
Author: joke2k
Author-email: joke2k@gmail.com
License: MIT License
Location: C:\Users\Dell\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires: tzdata
Required-by:
---
Name: pandas
Version: 2.3.3
Summary: Powerful data structures for data analysis, time series, and statistics
Home-page:
Author:
Author-email: The Pandas Development Team <pandas-dev@python.org>
License: BSD 3-Clause License
```

FIGURE 4.1 – Verification de l'installation des bibliothéque

4.2 Exécution du script de génération(`generate_data.py`)

Le script a généré environ 1000 utilisateurs, 200 produits et 5000 commandes.

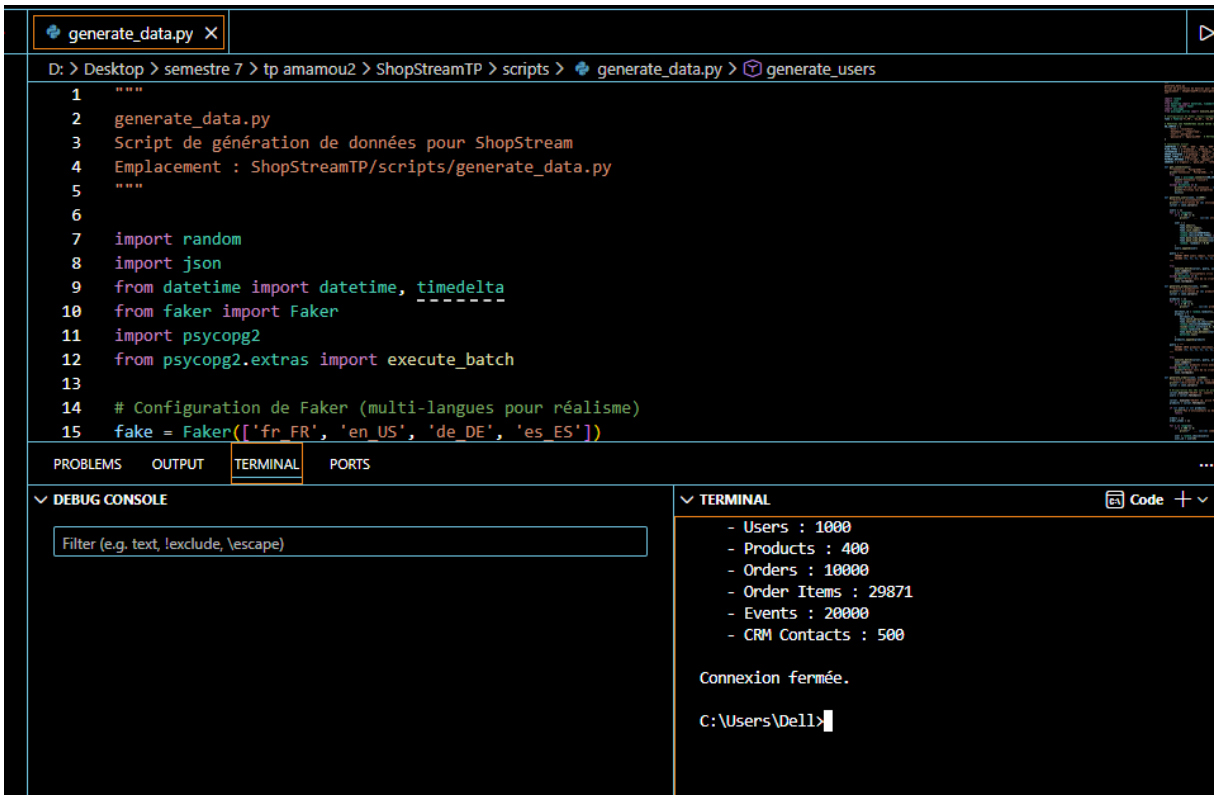

FIGURE 4.2 – Exécution du script generate_data.py

Chapitre 5

Configuration d'Amazon S3 (Data Lake)

5.1 Création du bucket et structure

Un bucket `shopstream-datalake` a été créé sur AWS. Nous y avons organisé les dossiers selon une zone `raw/` segmentée par source (`postgres`, `events`, `crm`).

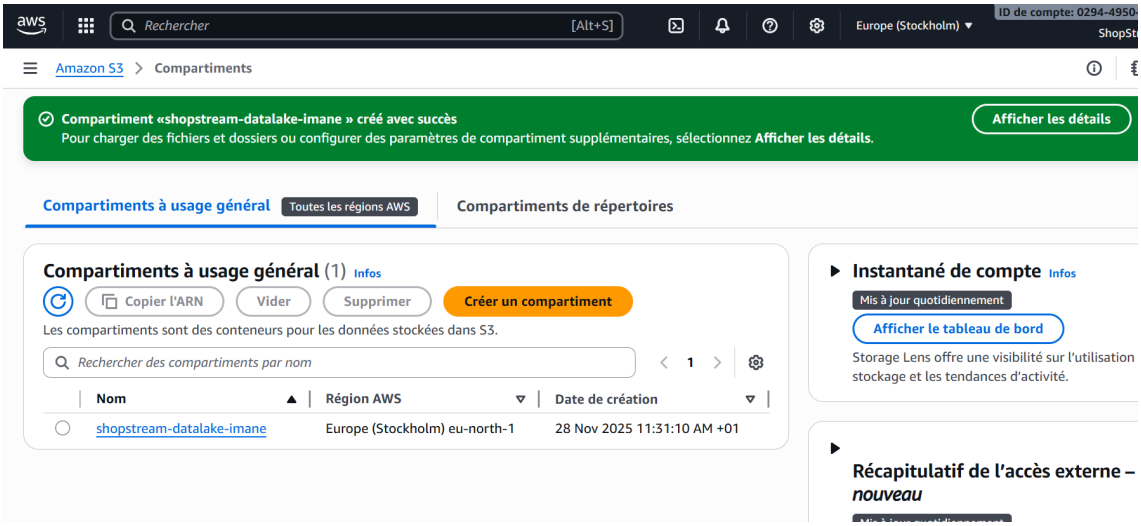

FIGURE 5.1 – Bucket S3 créé sur la console AWS

5.2 Configuration IAM et CLI

Un utilisateur IAM `shopstream-s3-user` avec les droits `AmazonS3FullAccess` a été configuré pour permettre l'accès programmatique via l'AWS CLI.

FIGURE 5.2 – Configuration de l'AWS CLI avec 'aws configure'

Chapitre 6

Export PostgreSQL vers S3

6.1 Script Python export_to_s3.py

Le script utilise `psycopg2` pour lire les données de PostgreSQL et `boto3` pour les téléverser vers S3 au format CSV.

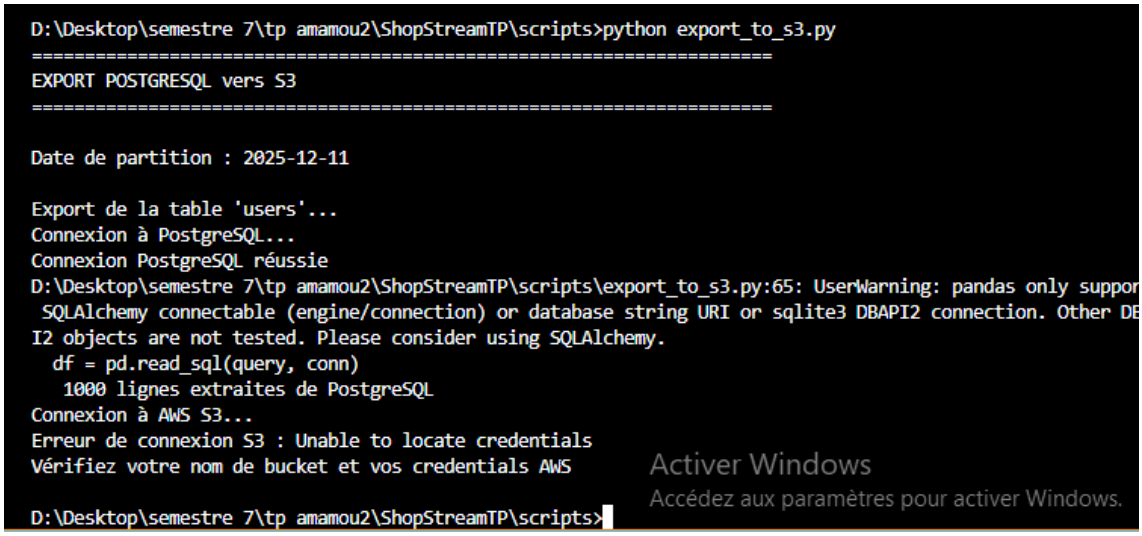

FIGURE 6.1 – Exécution de l'export et confirmation du chargement vers S3

6.2 Vérification des fichiers dans S3

	property	property_type	property_value	property_default
1	ENABLED	Boolean	true	false
2	STORAGE_PROVIDER	String	S3	
3	STORAGE_ALLOWED_LOCAT	List	s3://shopstream-datalake-imane/	[]
4	STORAGE_BLOCKED_LOCAT	List		[]
5	STORAGE_AWS_IAM_USER_A	String	arn:aws:iam::086325458992:use	
6	STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::029449501819:role	
7	STORAGE_AWS_EXTERNAL_I	String	WN69934_SFCRole=2_ICQpG7Y	
8	USE_PRIVATELINK_ENDPOIN	Boolean	false	false
9	COMMENT	String		

FIGURE 6.2 – Visualisation des fichiers CSV dans le bucket S3

Chapitre 7

Configuration de Snowflake

7.1 Mise en place de l'environnement

Nous avons créé les bases, schémas (`RAW`, `STAGING`, `CORE`, `MARTS`) et les *Warehouses* nécessaires au traitement.

7.2 Chargement via Storage Integration

L'intégration entre S3 et Snowflake a été sécurisée par une *Storage Integration* et un rôle IAM spécifique. Les données ont été chargées dans les tables de staging via la commande `COPY INTO`.

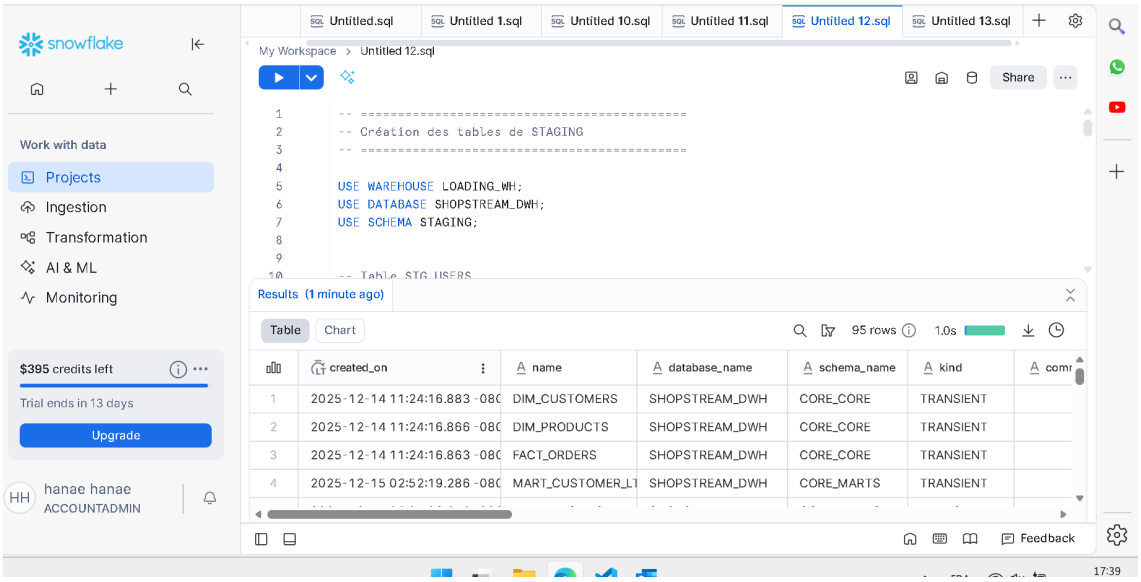

FIGURE 7.1 – Confirmation du chargement des données dans les tables STAGING

Chapitre 8

Transformations avec dbt

8.1 Initialisation et Tests

Après l'installation de `dbt-snowflake`, nous avons configuré le fichier `profiles.yml` pour connecter dbt à notre entrepôt Snowflake.

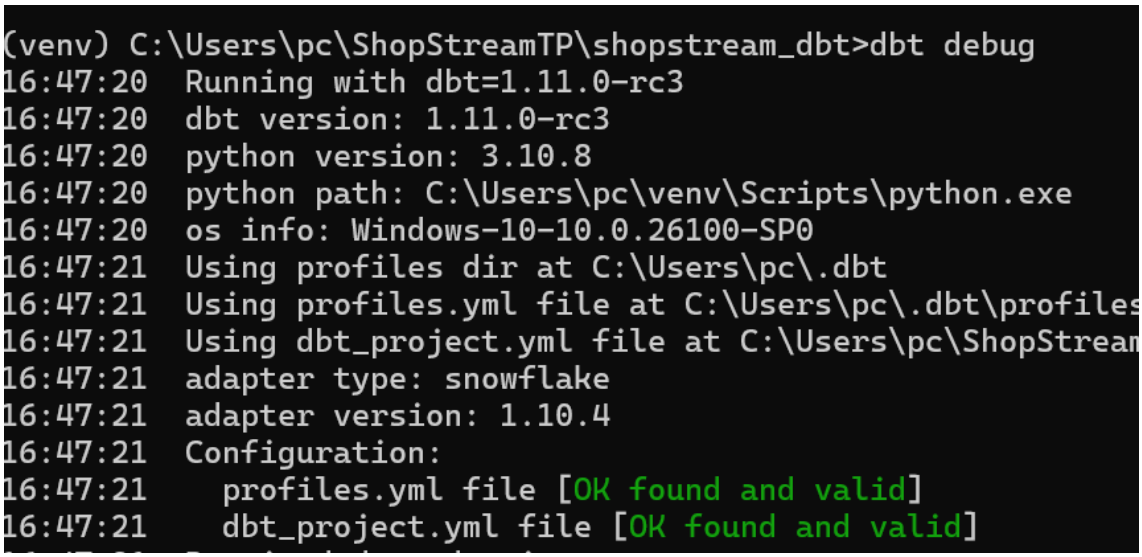

FIGURE 8.1 – Résultat de 'dbt debug' : Connexion réussie

8.2 Modélisation et Lineage

Nous avons développé les modèles pour les dimensions (`dim_customers`, `dim_products`) et les faits (`fact_orders`). Le *lineage graph* permet de visualiser les dépendances entre ces tables.

FIGURE 8.2 – Graphe de lignage (Lineage Graph) généré par dbt

8.3 Exécution des transformations

```
(venv) C:\Users\pc\ShopStreamTP\shopstream_dbt>dbt run
19:24:07  Running with dbt=1.7.9
19:24:11  Registered adapter: snowflake=1.7.2
19:24:11  Found 5 models, 13 tests, 4 sources, 0 exposures, 0 metrics, 430 macros, 0 groups, 0 semantic models
19:24:11
19:24:16  Concurrency: 4 threads (target='dev')
19:24:16
19:24:16  1 of 5 START sql table model CORE_core.dim_customers ........................... [RUN]
19:24:16  2 of 5 START sql table model CORE_core.dim_products ............................ [RUN]
19:24:16  3 of 5 START sql table model CORE_core.fact_orders ............................. [RUN]
19:24:16  4 of 5 START sql view model CORE_staging.stg_orders ............................ [RUN]
19:24:17  4 of 5 OK created sql view model CORE_staging.stg_orders ....................... [SUCCESS 1 in 1.67s]
19:24:17  1 of 5 OK created sql table model CORE_core.dim_customers ...................... [SUCCESS 1 in 1.75s]
19:24:17  3 of 5 OK created sql table model CORE_core.fact_orders ........................ [SUCCESS 1 in 1.77s]
19:24:17  2 of 5 OK created sql table model CORE_core.dim_products ....................... [SUCCESS 1 in 1.78s]
19:24:17  5 of 5 START sql table model CORE_marts.mart_sales_overview .................... [RUN]
19:24:19  5 of 5 OK created sql table model CORE_marts.mart_sales_overview ............... [SUCCESS 1 in 1.41s]
19:24:19
19:24:19  Finished running 1 view model, 4 table models in 0 hours 0 minutes and 7.82 seconds (7.82s).
19:24:19
19:24:19  Completed successfully
19:24:19
19:24:19  Done. PASS=5 WARN=0 ERROR=0 SKIP=0 TOTAL=5

(venv) C:\Users\pc\ShopStreamTP\shopstream_dbt>
```

FIGURE 8.3 – Exécution de 'dbt run' : Création des tables dans Snowflake

Chapitre 9

Orchestration avec Apache Airflow

Airflow orchestre le pipeline en automatisant l'extraction, le chargement et la transformation des données.

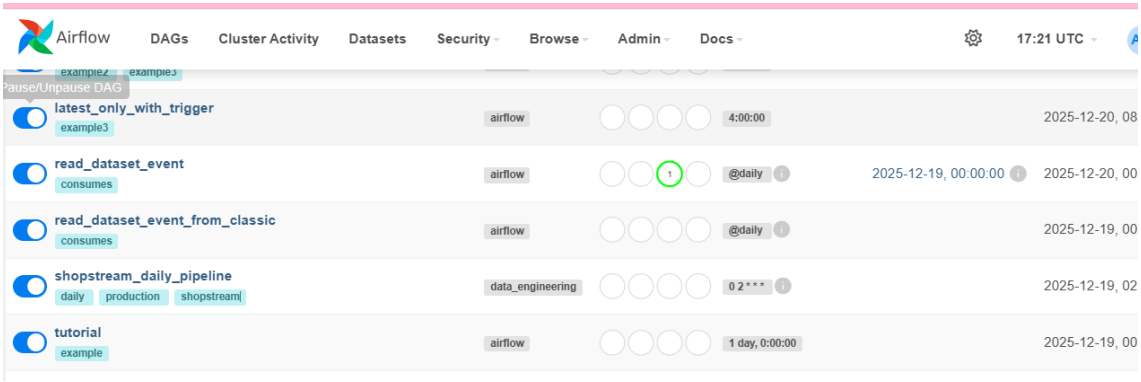

FIGURE 9.1 – Visualisation du DAG ShopStream dans Airflow

Chapitre 10

Business Intelligence avec Power BI

Le rapport final se connecte aux Data Marts de Snowflake (`mart_sales_overview`, `mart_product_performance`).

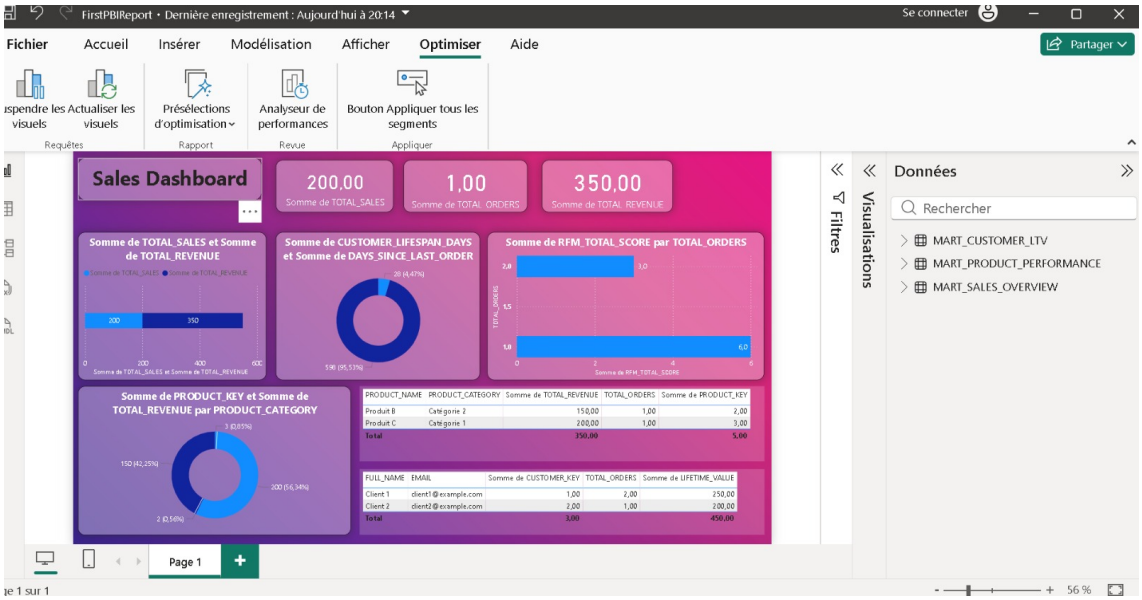

FIGURE 10.1 – Dashboard final : Analyse des ventes et performance produits

Chapitre 11

Conclusion

Ce TP a permis de mettre en œuvre une chaîne de traitement complète, illustrant le passage d'une donnée brute transactionnelle à une information décisionnelle exploitable.