

Rapport de TP Cloud

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

Étudiants : Aya Hassan, Fatima zahra Tliji, Salma Mouj
Université Euro-Med

decembre 2025

Table des matières

1	Introduction au Modern Data Stack	2
1.1	Contexte	2
1.2	Architecture globale	2
2	PostgreSQL : Source de données (OLTP)	3
2.1	Installation et Configuration	3
2.2	Création du Schéma	4
2.3	Génération de données	6
3	Amazon S3 : Data Lake	9
3.1	Création du Bucket	9
3.2	Export vers S3	9
4	Snowflake : Data Warehouse	12
4.1	Configuration de l'environnement	12
4.2	Intégration S3 (Storage Integration)	13
5	dbt : Transformation de données	14
5.1	Installation et Initialisation	14
5.2	Modélisation (Lineage)	15
5.3	Exécution et Tests	15
6	Apache Airflow : Orchestration	18
6.1	Installation et Configuration	18
7	Power BI : Business Intelligence	20
7.1	Connexion et Modélisation	20
7.2	Tableau de bord : Vue d'ensemble	20
7.3	Tableau de bord : Analyse Clients (RFM)	20
8	Conclusion	22

Chapitre 1

Introduction au Modern Data Stack

1.1 Contexte

Dans le cadre de ce TP, nous mettons en place une architecture de données complète basée sur le cloud pour l'entreprise fictive "**ShopStream**". L'objectif est de migrer d'une base de données transactionnelle (OLTP) vers un entrepôt de données analytique (OLAP) pour permettre la Business Intelligence.

1.2 Architecture globale

Le pipeline mis en place suit le flux suivant :

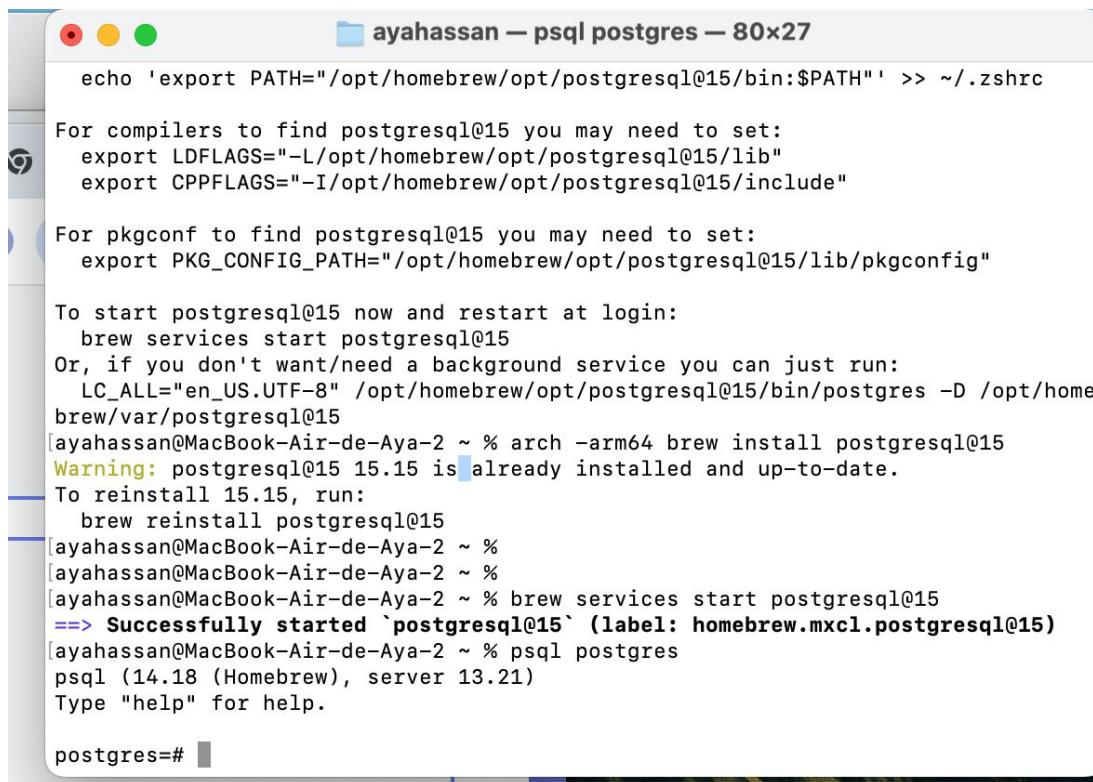
1. **Source** : PostgreSQL (Données transactionnelles).
2. **Ingestion** : Scripts Python vers Amazon S3 (Data Lake).
3. **Storage** : Snowflake (Data Warehouse).
4. **Transformation** : dbt (Data Build Tool).
5. **Orchestration** : Apache Airflow.
6. **Visualisation** : Power BI.

Chapitre 2

PostgreSQL : Source de données (OLTP)

2.1 Installation et Configuration

Nous avons installé PostgreSQL 15 et configuré l'outil pgAdmin 4. Une base de données nommée `shopstream` a été créée.



```
ayahassan — psql postgres — 80x27
echo 'export PATH="/opt/homebrew/opt/postgresql@15/bin:$PATH"' >> ~/.zshrc

For compilers to find postgresql@15 you may need to set:
  export LDFLAGS="-L/opt/homebrew/opt/postgresql@15/lib"
  export CPPFLAGS="-I/opt/homebrew/opt/postgresql@15/include"

For pkgconf to find postgresql@15 you may need to set:
  export PKG_CONFIG_PATH="/opt/homebrew/opt/postgresql@15/lib/pkgconfig"

To start postgresql@15 now and restart at login:
  brew services start postgresql@15
Or, if you don't want/need a background service you can just run:
  LC_ALL="en_US.UTF-8" /opt/homebrew/opt/postgresql@15/bin/postgres -D /opt/homebrew/var/postgresql@15
[ayahassan@MacBook-Air-de-Aya-2 ~ % arch -arm64 brew install postgresql@15
Warning: postgresql@15 15.15 is already installed and up-to-date.
To reinstall 15.15, run:
  brew reinstall postgresql@15
[ayahassan@MacBook-Air-de-Aya-2 ~ %
[ayahassan@MacBook-Air-de-Aya-2 ~ %
[ayahassan@MacBook-Air-de-Aya-2 ~ % brew services start postgresql@15
==> Successfully started `postgresql@15` (label: homebrew.mxcl.postgresql@15)
[ayahassan@MacBook-Air-de-Aya-2 ~ % psql postgres
psql (14.18 (Homebrew), server 13.21)
Type "help" for help.

postgres=#
```

FIGURE 2.1 – Capture 1 d'écran de pgAdmin montrant la connexion au serveur

```
[ayahassan@MacBook-Air-de-Aya-2 ~ % source ~/.zshrc
[ayahassan@MacBook-Air-de-Aya-2 ~ % psql --version
psql (PostgreSQL) 15.15 (Homebrew)
ayahassan@MacBook-Air-de-Aya-2 ~ % ]
```

FIGURE 2.2 – Capture 2 d'écran de pgAdmin montrant la connexion au serveur

2.2 Cr éation du Sch éma

Le script SQL suivant a été exécuté pour créer les tables relationnelles (`users`, `products`, `orders`, etc.).

```
1 -- Cr ation de la table USERS (extrait)
2 CREATE TABLE users (
3     id SERIAL PRIMARY KEY,
4     email VARCHAR(255) UNIQUE NOT NULL,
5     first_name VARCHAR(100),
6     last_name VARCHAR(100),
7     country VARCHAR(3),
8     plan_type VARCHAR(20) DEFAULT 'freemium',
9     created_at TIMESTAMP DEFAULT NOW(),
10    last_login TIMESTAMP,
11    is_active BOOLEAN DEFAULT TRUE
12 );
13 -- (Le reste du script a t ex cut comme indiqu dans le TP)
```

Listing 2.1 – Cr éation des tables PostgreSQL

```
^
postgres=# \q
ayahassan@MacBook-Air-de-Aya-2 ~ % createdb shopstream
ayahassan@MacBook-Air-de-Aya-2 ~ % psql shopstream
psql (15.15 (Homebrew), serveur 13.21)
Saisissez « help » pour l'aide.

shopstream=#

shopstream=# -- 1. Table USERS
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    country VARCHAR(3),
    plan_type VARCHAR(20) DEFAULT 'freemium',
    created_at TIMESTAMP DEFAULT NOW(),
    last_login TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE
);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_country ON users(country);
CREATE INDEX idx_users_plan_type ON users(plan_type);

-- 2. Table PRODUCTS
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    merchant_id INT NOT NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    category VARCHAR(100),
    price DECIMAL(10,2) NOT NULL,
    stock_quantity INT DEFAULT 0,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
CREATE INDEX idx_products_merchant ON products(merchant_id);
CREATE INDEX idx_products_category ON products(category);
```

```

-- 3. Table ORDERS
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    user_id INT NOT NULL REFERENCES users(id),
    created_at TIMESTAMP DEFAULT NOW(),
    total_amount DECIMAL(10,2) NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    country VARCHAR(3),
    payment_method VARCHAR(50)
);

CREATE INDEX idx_orders_user_id ON orders(user_id);
CREATE INDEX idx_orders_created_at ON orders(created_at);
CREATE INDEX idx_orders_status ON orders(status);

-- 4. Table ORDER_ITEMS
CREATE TABLE order_items (
    id SERIAL PRIMARY KEY,
    order_id INT NOT NULL REFERENCES orders(id),
    product_id INT NOT NULL REFERENCES products(id),
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    line_total DECIMAL(10,2) NOT NULL
);

CREATE INDEX idx_order_items_order_id ON order_items(order_id);
CREATE INDEX idx_order_items_product_id ON order_items(product_id);

-- 5. Table EVENTS
CREATE TABLE events (
    id BIGSERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id),
    event_type VARCHAR(50) NOT NULL,
    event_ts TIMESTAMP DEFAULT NOW(),
    metadata JSONB
);

CREATE INDEX idx_events_user_id ON events(user_id);
CREATE INDEX idx_events_event_type ON events(event_type);
CREATE INDEX idx_events_event_ts ON events(event_ts);

CREATE INDEX idx_events_event_ts ON events(event_ts);

-- 6. Table CRM_CONTACTS
CREATE TABLE crm_contacts (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    source VARCHAR(100),
    campaign_id VARCHAR(100),
    created_at TIMESTAMP DEFAULT NOW(),
    converted BOOLEAN DEFAULT FALSE,
    converted_at TIMESTAMP
);

CREATE INDEX idx_crm_contacts_email ON crm_contacts(email);
CREATE INDEX idx_crm_contacts_source ON crm_contacts(source);
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE TABLE
CREATE INDEX
shopstream=#

```

```
CREATE INDEX
[shopstream=# \dt
      Liste des relations
Schéma |     Nom      | Type | Propriétaire
-----+-----+-----+-----+
public | crm_contacts | table | ayahassan
public | events       | table | ayahassan
public | order_items  | table | ayahassan
public | orders        | table | ayahassan
public | products      | table | ayahassan
public | users         | table | ayahassan
(6 lignes)

shopstream=#
```

2.3 Génération de données

Un script Python utilisant la librairie Faker a été utilisé pour générer des données réalistes.

```
1 python generate_data.py
```

Listing 2.2 – Commande d'exécution du script de génération

```
[ayahassan@MacBook-Air-de-Aya-2 scripts % open -aTextEdit generate_data.py
[ayahassan@MacBook-Air-de-Aya-2 scripts % psql shopstream
psql (15.15 (Homebrew), serveur 13.21)
Saisissez « help » pour l'aide.

shopstream=# TRUNCATE TABLE order_items, orders, events, crm_contacts, products,
users RESTART IDENTITY CASCADE;
[\q
TRUNCATE TABLE
[ayahassan@MacBook-Air-de-Aya-2 scripts % python3.10 generate_data.py
=====
🚀 GENERATION DE DONNEES SHOPSTREAM
=====
Connexion à PostgreSQL...
✓ Connexion réussie

📊 Génération de 1000 utilisateurs...
... 0/1000 utilisateurs générés
... 100/1000 utilisateurs générés
... 200/1000 utilisateurs générés
... 300/1000 utilisateurs générés
... 400/1000 utilisateurs générés
... 500/1000 utilisateurs générés
... 600/1000 utilisateurs générés
... 700/1000 utilisateurs générés
... 800/1000 utilisateurs générés
... 900/1000 utilisateurs générés
✓ 1000 utilisateurs créés avec succès

📦 Génération de 200 produits...
... 0/200 produits générés
... 50/200 produits générés
... 100/200 produits générés
... 150/200 produits générés
✓ 200 produits créés avec succès

🛒 Génération de 5000 commandes...
... 0/5000 commandes générées
... 500/5000 commandes générées
... 1000/5000 commandes générées
... 1500/5000 commandes générées
... 2000/5000 commandes générées
... 2500/5000 commandes générées
... 3000/5000 commandes générées
... 3500/5000 commandes générées
... 4000/5000 commandes générées
... 4500/5000 commandes générées
✓ 5000 commandes créées
✓ 14989 lignes de commande créées
```

FIGURE 2.3 – Terminal1 montrant la sortie du script `generate_data.py`

```

❸ Génération de 500 contacts CRM...
... 0/500 contacts générés
... 100/500 contacts générés
... 200/500 contacts générés
... 300/500 contacts générés
... 400/500 contacts générés
✓ 500 contacts CRM créés

=====
✓ GENERATION TERMINEE AVEC SUCCES
=====

📊 Récapitulatif :
- Users : 1000
- Products : 200
- Orders : 5000
- Order Items : 14989
- Events : 10000
- CRM Contacts : 500

⚡ Connexion fermée.
ayahassan@MacBook-Air-de-Aya-2 scripts %

```

FIGURE 2.4 – Terminal2 montrant la sortie du script generate_data.py

```

[ayahassan@MacBook-Air-de-Aya-2 scripts % psql shopstream
psql (15.15 (Homebrew), serveur 13.21)
Saisissez « help » pour l'aide.

shopstream=# -- Voir les premiers utilisateurs
SELECT * FROM users LIMIT 5;

-- Voir les produits par catégorie
SELECT category, COUNT(*) FROM products GROUP BY category;

-- Voir le CA total
SELECT SUM(total_amount) AS chiffre_affaires FROM orders;

-- Voir les commandes par pays
SELECT country, COUNT(*), SUM(total_amount)
FROM orders
GROUP BY country
ORDER BY SUM(total_amount) DESC;

-- Voir les événements les plus fréquents
SELECT event_type, COUNT(*)
FROM events
GROUP BY event_type
ORDER BY COUNT(*) DESC;

-- Quitter
[\q
      id |           email          | first_name | last_name | country | plan_type
e  | created_at        | last_login   | is_active
-----+-----+-----+-----+
      +-----+
      1 | johnsonjustin@example.net | Marie       | Hayes     | USA      | freemium
      | 2025-05-15 16:10:58 | 2025-11-20 21:21:01 | t
      2 | danielle71@example.org   | Arthur      | Buchholz  | FRA      | freemium
      | 2025-01-06 00:19:05 |                   | t
      3 | jennyhernandez@example.net | Céline     | Scott     | USA      | freemium
      | 2024-11-07 13:04:55 | 2025-11-05 09:24:00 | t
      4 | mitschkecaterina@example.org | Ewa        | Klein     | USA      | freemium
      | 2024-08-31 12:54:14 | 2025-11-04 20:56:16 | t
      5 | anastasiopuig@example.org | Noémie    | Garcia    | ITA      | enterprise
se | 2025-10-05 18:34:08 |                   | t
(5 lignes)

(END)]]

```

FIGURE 2.5 – pgAdmin : SELECT * FROM users montrant les données insérées

Chapitre 3

Amazon S3 : Data Lake

3.1 Crédation du Bucket

Un bucket S3 nommé `shopstream-datalake-votreprenom` a été créé dans la région `eu-west-3`.

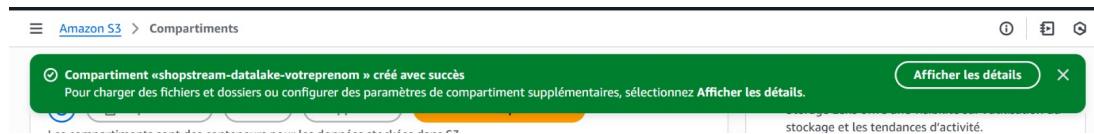


FIGURE 3.1 – Console AWS S3 montrant le bucket

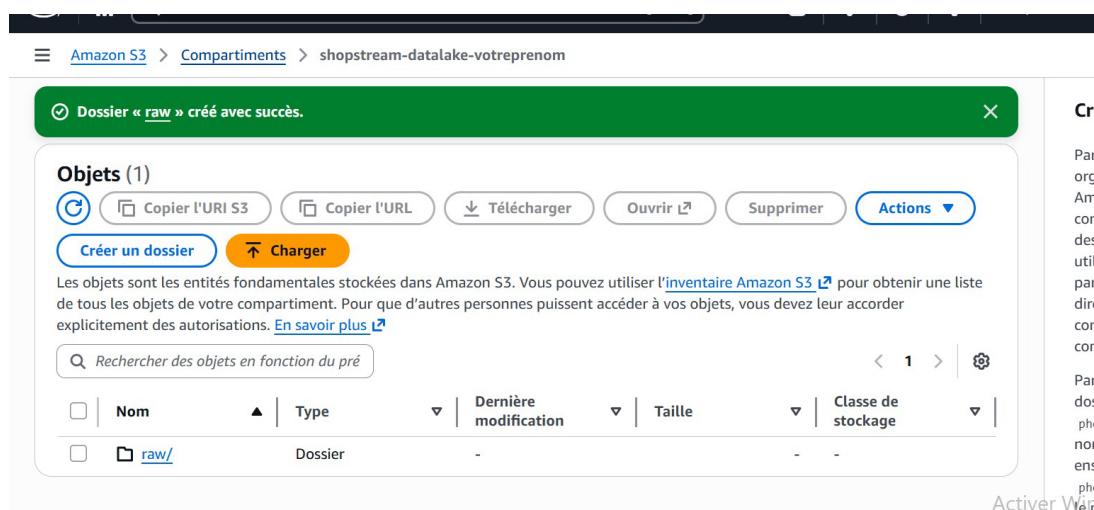


FIGURE 3.2 – Console AWS S3 montrant l’arborescence des dossiers raw

3.2 Export vers S3

Le script `export_to_s3.py` a été utilisé pour extraire les données de PostgreSQL et les uploader en CSV vers S3.

```

1 def export_table_to_s3(table_name, date_partition):
2     # ... connexion DB ...
3     df = pd.read_sql(query, conn)
4     # ... upload S3 ...
5     s3_client.put_object(
6         Bucket=S3_CONFIG['bucket'],
7         Key=s3_key,
8         Body=csv_buffer.getvalue()
9     )

```

Listing 3.1 – Extrait du script `export_to_s3.py`

```

[ayahassan@MacBook-Air-de-Aya-2 scripts % aws --version
aws-cli/1.43.5 Python/3.10.17 Darwin/24.0.0 botocore/1.41.5
[ayahassan@MacBook-Air-de-Aya-2 scripts % aws configure
AWS Access Key ID [None]: AKIAQQH6ZXKI2DJQ2U32
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-3
Default output format [None]: json
[ayahassan@MacBook-Air-de-Aya-2 scripts % python3.10 export_to_s3.py
=====
🚀 EXPORT POSTGRESQL vers S3
=====

📅 Date : 2025-11-28

📍 Export de la table 'users'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
    /Users/ayahassan/ShopStreamTP/scripts/export_to_s3.py:62: 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)
            ✓ 1000 lignes extraites
    🌐 Connexion à AWS S3...
    ✓ Connexion S3 réussie
        ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/users/2025-11-28/users_20251128.csv

📍 Export de la table 'products'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
        ✓ 200 lignes extraites
    🌐 Connexion à AWS S3...
    ✓ Connexion S3 réussie
        ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/products/2025-11-28/products_20251128.csv

📍 Export de la table 'orders'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
        ✓ 5000 lignes extraites
    🌐 Connexion à AWS S3...
    ✓ Connexion S3 réussie
        ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/orders/2025-11-28/orders_20251128.csv

📍 Export de la table 'order_items'...
    Connexion à PostgreSQL...

```

FIGURE 3.3 – Terminal1 montrant l'exécution réussie de `export_to_s3.py`

```

    📈 Export de la table 'orders'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
      ✓ 5000 lignes extraites
    🌃 Connexion à AWS S3...
    ✓ Connexion S3 réussie
      ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/orders/2025-11-28/orders_20251128.csv

    📈 Export de la table 'order_items'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
      ✓ 14989 lignes extraites
    🌃 Connexion à AWS S3...
    ✓ Connexion S3 réussie
      ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/order_items/2025-11-28/order_items_20251128.csv

    📈 Export de la table 'crm_contacts'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
      ✓ 500 lignes extraites
    🌃 Connexion à AWS S3...
    ✓ Connexion S3 réussie
      ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/postgres/crm_contacts/2025-11-28/crm_contacts_20251128.csv

    📈 Export de la table 'events'...
    Connexion à PostgreSQL...
    ✓ Connexion PostgreSQL réussie
    /Users/ayahassan/ShopStreamTP/scripts/export_to_s3.py:102: 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
    🌃 Connexion à AWS S3...
    ✓ Connexion S3 réussie
      ✓ Uploadé : s3://shopstream-datalake-votrerenom/raw/events/2025-11-28/events_20251128.json

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

🔗 Vérifiez : https://s3.console.aws.amazon.com/s3/buckets/shopstream-datalake-votrerenom
ayahassan@MacBook-Air-de-Aya-2 scripts %
  
```

FIGURE 3.4 – Terminal2 montrant l'exécution réussie de `export_to_s3.py`

Chapitre 4

Snowflake : Data Warehouse

4.1 Configuration de l'environnement

Nous avons créé la base de données `SHOPSTREAM_DWH` et les entrepôts virtuels (Warehouses).

```
1 CREATE WAREHOUSE IF NOT EXISTS LOADING_WH WITH WAREHOUSE_SIZE = 'XSMALL'
      ;
2 CREATE WAREHOUSE IF NOT EXISTS TRANSFORM_WH WITH WAREHOUSE_SIZE = 'SMALL
      ';
3 CREATE WAREHOUSE IF NOT EXISTS BI_WH WITH WAREHOUSE_SIZE = 'XSMALL';
```

Listing 4.1 – Crédit des Warehouses

The screenshot shows the Snowflake web interface. On the left, the sidebar includes sections for Work with data (Projects, Ingestion, Transformation, AI & ML, Monitoring, Marketplace), Horizon Catalog (Catalog, Data sharing, Governance & security), and a trial status (\$400 credits left, Trial ends in 30 days, Upgrade button). The main area has tabs for Worksheets, Database Explorer, and Results (just now). The Worksheets tab shows a code editor with the following SQL commands:

```
-- Configuration initiale Snowflake pour ShopStream
USE DATABASE SHOPSTREAM_DWH;

CREATE SCHEMA IF NOT EXISTS RAW;
CREATE SCHEMA IF NOT EXISTS STAGING;
CREATE SCHEMA IF NOT EXISTS CORE;
CREATE SCHEMA IF NOT EXISTS MARTS;

SHOW SCHEMAS;

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

The Database Explorer shows the `SHOPSTREAM_DWH` database with the `INFORMATION_SCHEMA` schema expanded, displaying tables like `Views` (53) and `PUBLIC`. The Results section shows a single message row: "Configuration initiale terminée".

FIGURE 4.1 – Snowflake : Résultat de la commande SHOW WAREHOUSE

4.2 Intégration S3 (Storage Integration)

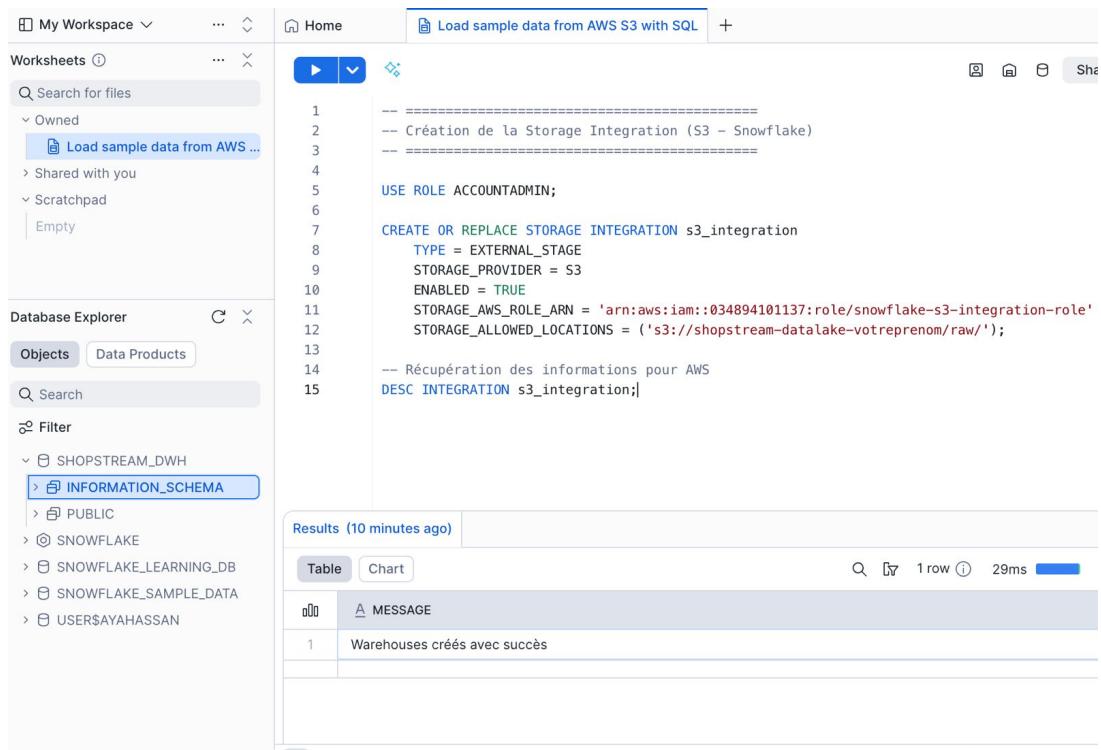
Une Storage Integration a été configurée pour permettre à Snowflake de lire les données sécurisées sur S3.

```

1 CREATE OR REPLACE STAGE s3_raw_stage
2   STORAGE_INTEGRATION = s3_integration
3   URL = 's3://shopstream-datalake-votrerenom/raw/'
4   FILE_FORMAT = (TYPE = CSV FIELD_OPTIONALLY_ENCLOSED_BY = '',
5   SKIP_HEADER = 1);

```

Listing 4.2 – Crédit du Stage Externe



```

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::034894101137:role/snowflake-s3-integration-role'
12  STORAGE_ALLOWED_LOCATIONS = ('s3://shopstream-datalake-votrerenom/raw/');
13
14  -- Récupération des informations pour AWS
15  DESC INTEGRATION s3_integration;

```

FIGURE 4.2 – Crédit du Stage Externe

Chapitre 5

dbt : Transformation de données

5.1 Installation et Initialisation

Le projet `shopstream_dbt` a été initialisé et configuré pour se connecter à Snowflake via le fichier `profiles.yml`.

```
(dbt_env) ayahassan@MacBook-Air-de-Aya-2 shopstream_dbt % nano ~/.dbt/profiles.yml
[(dbt_env) ayahassan@MacBook-Air-de-Aya-2 shopstream_dbt % dbt debug]
20:21:18 Running with dbt=1.11.0-rc1
20:21:18 dbt version: 1.11.0-rc1
20:21:18 python version: 3.13.3
20:21:18 python path: /Users/ayahassan/ShopStreamTP/dbt_env/bin/python3.13
20:21:18 os info: macOS-15.0.1-arm64-i386-64bit-Mach-O
20:21:18 Using profiles dir at /Users/ayahassan/.dbt
20:21:18 Using profiles.yml file at /Users/ayahassan/.dbt/profiles.yml
20:21:18 Using dbt_project.yml file at /Users/ayahassan/ShopStreamTP/shopstream_dbt/dbt_project.yml
20:21:18 adapter type: snowflake
20:21:18 adapter version: 1.10.3
20:21:18 Configuration:
20:21:18   profiles.yml file [OK found and valid]
20:21:18   dbt_project.yml file [OK found and valid]
20:21:18 Required dependencies:
20:21:18   - git [OK found]

20:21:18 Connection:
20:21:18   account: ekxwane-ri30251
20:21:18   user: AYAHASSAN
20:21:18   database: SHOPSTREAM_DWH
20:21:18   warehouse: TRANSFORM_WH
20:21:18   role: ACCOUNTADMIN
20:21:18   schema: CORE
20:21:18   authenticator: None
20:21:18   oauth_client_id: None
20:21:18   query_tag: None
20:21:18   client_session_keep_alive: False
20:21:18   host: None
20:21:18   port: None
20:21:18   proxy_host: None
20:21:18   proxy_port: None
20:21:18   protocol: None
20:21:18   connect_retries: 1
20:21:18   connect_timeout: None
20:21:18   retry_on_database_errors: False
20:21:18   retry_all: False
20:21:18   insecure_mode: False
20:21:18   reuse_connections: True
20:21:18   s3_stage_vpce_dns_name: None
20:21:18   platform_detection_timeout_seconds: 0.0
20:21:18 Registered adapter: snowflake=1.10.3
20:21:20   Connection test: [OK connection ok]

20:21:20 All checks passed!
(dbt_env) ayahassan@MacBook-Air-de-Aya-2 shopstream_dbt %
```

FIGURE 5.1 – Terminal : Résultat de la commande `dbt debug` (All checks passed)

5.2 Modélisation (Lineage)

Nous avons créé trois couches de modèles :

- **Staging** : Nettoyage des données brutes.
- **Core** : Création des dimensions et faits.
- **Marts** : Tables agrégées pour le métier.

```
1 WITH fact_orders AS (
2     SELECT * FROM {{ ref('fact_orders') }}
3 ),
4 -- ... jointures ...
5 daily_sales AS (
6     SELECT
7         f.date_key AS sale_date,
8         SUM(f.line_revenue) AS total_revenue
9         -- ...
10    FROM fact_orders f
11    -- ...
12 )
13 SELECT * FROM daily_sales
```

Listing 5.1 – Exemple : models/marts/mart_sales_overview.sql

5.3 Exécution et Tests

Les modèles ont été exécutés et testés.

```
(dbt_env) ayahassan@MacBook-Air-de-Aya-2 shopstream_dbt % cd ~/ShopStreamTP/shopstream_dbt
[dbt run
20:51:28  Running with dbt=1.11.0-rc1
20:51:28  Registered adapter: snowflake=1.10.3
20:51:28  Found 8 models, 5 sources, 504 macros
20:51:28  Concurrency: 4 threads (target='dev')
20:51:28
20:51:31  2 of 8 START sql view model CORE_staging.stg_orders .....
..... [RUN]
20:51:31  1 of 8 START sql view model CORE_staging.stg_order_items .....
..... [RUN]
20:51:31  4 of 8 START sql view model CORE_staging.stg_users .....
..... [RUN]
20:51:31  3 of 8 START sql view model CORE_staging.stg_products .....
..... [RUN]
20:51:31  1 of 8 OK created sql view model CORE_staging.stg_order_items .....
..... [SUCCESS 1 in 0.79s]
20:51:31  4 of 8 OK created sql view model CORE_staging.stg_users .....
..... [SUCCESS 1 in 0.79s]
20:51:31  2 of 8 OK created sql view model CORE_staging.stg_orders .....
..... [SUCCESS 1 in 0.80s]
20:51:31  3 of 8 OK created sql view model CORE_staging.stg_products .....
..... [SUCCESS 1 in 0.79s]
20:51:31  5 of 8 START sql table model CORE_core.dim_users .....
..... [RUN]
20:51:31  6 of 8 START sql table model CORE_core.fact_orders .....
..... [RUN]
20:51:31  7 of 8 START sql table model CORE_core.dim_products .....
..... [RUN]
20:51:33  7 of 8 OK created sql table model CORE_core.dim_products .....
..... [SUCCESS 1 in 1.55s]
20:51:33  5 of 8 OK created sql table model CORE_core.dim_users .....
..... [SUCCESS 1 in 1.61s]
20:51:33  6 of 8 OK created sql table model CORE_core.fact_orders .....
..... [SUCCESS 1 in 1.63s]
20:51:33  8 of 8 START sql table model CORE_marts.mart_sales_overview .....
..... [RUN]
20:51:34  8 of 8 OK created sql table model CORE_marts.mart_sales_overview .....
..... [SUCCESS 1 in 1.21s]
20:51:34
20:51:34  Finished running 4 table models, 4 view models in 0 hours 0 minutes and
6.22 seconds (6.22s).
20:51:34
20:51:34  Completed successfully
20:51:34
20:51:34  Done. PASS=8 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=8
```

FIGURE 5.2 – Terminal : Résultat de la commande dbt run

The screenshot shows the dbt web interface at localhost:8080/#/model/model.shopstream_dbt.dim_products. The left sidebar lists various database schemas and tables, including `CORE_core`, `dim_products`, `fact_orders`, `CORE_marts`, `CORE_staging`, `STAGING`, and `dim_users`. The right panel displays detailed information for the `dim_products` table, such as its owner (`ACCOUNTADMIN`), type (`table`), package (`shopstream_dbt`), language (`sql`), and relation (`SHOPSTREAM_DWH.CORE_core.dim_products`). It also shows approximate size (25 KB), last modified (2025-11-29 20:51 UTC), and row count (200). The `Description` section notes that the model is not currently documented. The `Columns` section lists three columns: `product_id` (NUMBER), `merchant_id` (NUMBER), and `product_name` (TEXT).

COLUMN	TYPE	DESCRIPTION	CONSTRAINTS	DATA TESTS	MORE?
<code>product_id</code>	NUMBER				
<code>merchant_id</code>	NUMBER				
<code>product_name</code>	TEXT				

FIGURE 5.3 – Navigateur Web

Chapitre 6

Apache Airflow : Orchestration

6.1 Installation et Configuration

Airflow a été installé localement. Les connexions vers AWS, Snowflake et PostgreSQL ont été configurées dans l'interface Admin.

```
[ayahassan@MacBook-Air-de-Aya-2 ShopStreamTP % airflow db init           ]
[~/opt/homebrew/lib/python3.10/site-packages/airflow/cli/commands/db_command.py:47 D
eprecationWarning: `db init` is deprecated. Use `db migrate` instead to migrate t
he db and/or airflow connections create-default-connections to create the default
connections
DB: sqlite:///Users/ayahassan/ShopStreamTP/airflow/airflow.db
[2025-11-29T22:33:04.194+0100] {migration.py:213} INFO - Context impl SQLiteImpl.
[2025-11-29T22:33:04.195+0100] {migration.py:216} INFO - Will assume non-transacti
onal DDL.
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running stamp_revision -> 10b52ebd31f7
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored e
ncrypted.
Initialization done
[ayahassan@MacBook-Air-de-Aya-2 ShopStreamTP % ls -la ~/ShopStreamTP/airflow/      ]
total 1080
drwxr-xr-x@ 5 ayahassan  staff     160 Nov 29 22:33 .
drwxr-xr-x  6 ayahassan  staff     192 Nov 29 22:33 ..
-rw-----@ 1 ayahassan  staff    73620 Nov 29 22:33 airflow.cfg
-rw-r--r--@ 1 ayahassan  staff   479232 Nov 29 22:33 airflow.db
drwxr-xr-x@ 3 ayahassan  staff     96 Nov 29 22:33 logs
ayahassan@MacBook-Air-de-Aya-2 ShopStreamTP % mkdir -p ~/ShopStreamTP/airflow/dags
mkdir -p ~/ShopStreamTP/airflow/plugins
[ayahassan@MacBook-Air-de-Aya-2 ShopStreamTP % ls -la ~/ShopStreamTP/airflow/      ]
total 1080
drwxr-xr-x@ 7 ayahassan  staff    224 Nov 29 22:34 .
drwxr-xr-x  6 ayahassan  staff    192 Nov 29 22:33 ..
-rw-----@ 1 ayahassan  staff    73620 Nov 29 22:33 airflow.cfg
-rw-r--r--@ 1 ayahassan  staff   479232 Nov 29 22:33 airflow.db
drwxr-xr-x  2 ayahassan  staff     64 Nov 29 22:34 dags
drwxr-xr-x@ 3 ayahassan  staff     96 Nov 29 22:33 logs
drwxr-xr-x  2 ayahassan  staff     64 Nov 29 22:34 plugins
ayahassan@MacBook-Air-de-Aya-2 ShopStreamTP % ]
```

FIGURE 6.1 – Configuration du dossier Airflow

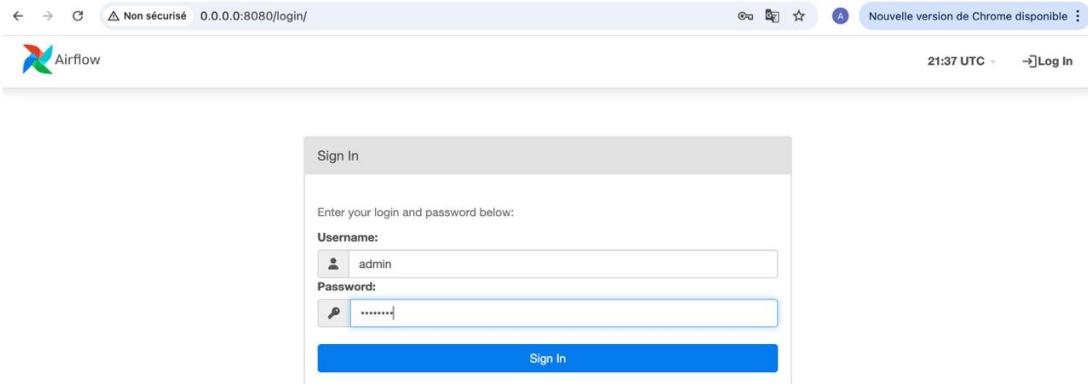


FIGURE 6.2 – Démarrage d’Airflow

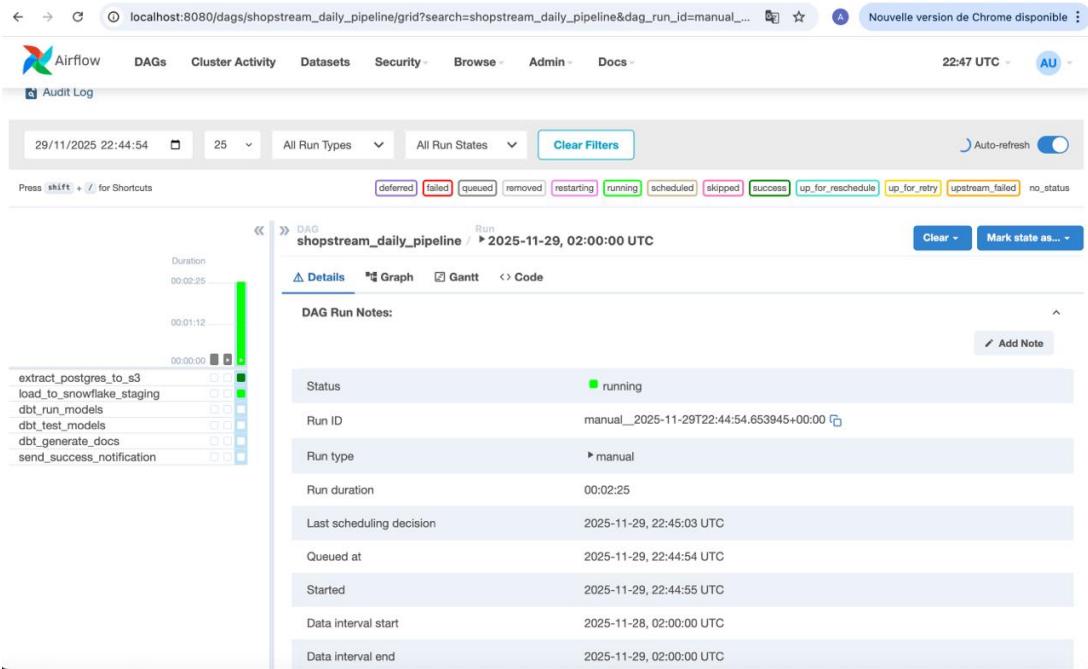


FIGURE 6.3 – Activation et exécution du DAG

Chapitre 7

Power BI : Business Intelligence

7.1 Connexion et Modélisation

Power BI a été connecté à Snowflake (Warehouse BI_WH) pour importer les tables de la couche *Marts*.

7.2 Tableau de bord : Vue d'ensemble

Ce tableau de bord présente les KPIs principaux : CA Total, Nombre de commandes, et évolution temporelle.

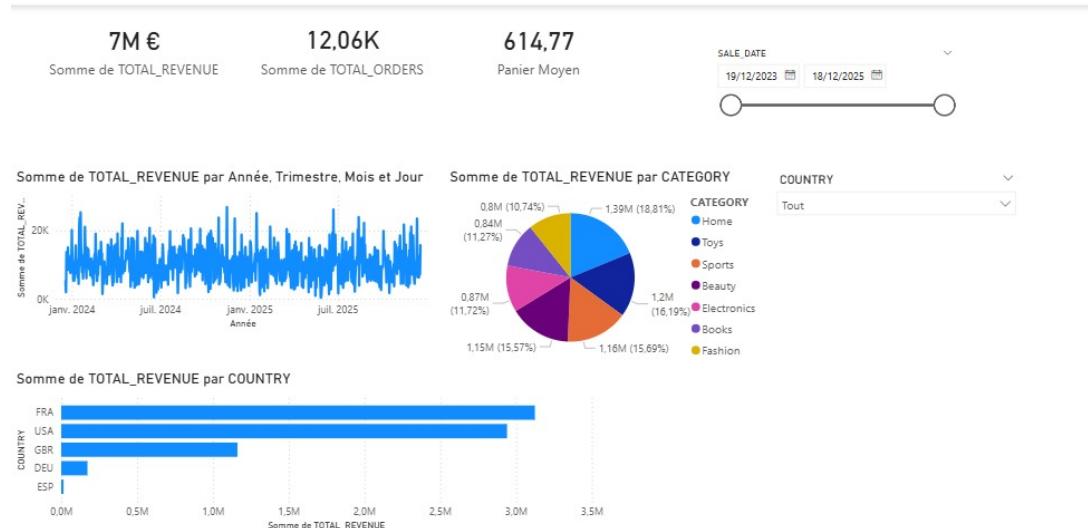


FIGURE 7.1 – Power BI : Capture du Dashboard "Vue d'ensemble des ventes"

7.3 Tableau de bord : Analyse Clients (RFM)

Ce rapport visualise la segmentation client et la Lifetime Value.

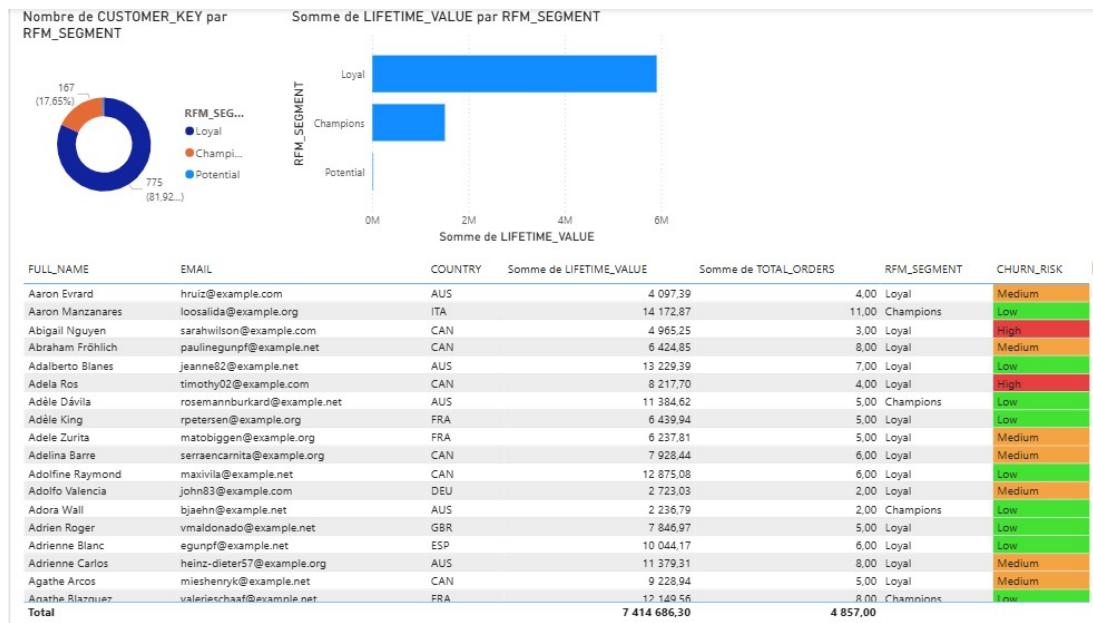


FIGURE 7.2 – Power BI : Capture du Dashboard "Analyse Clients"

PRODUCT_NAME	CATEGORY	Somme de TOTAL_ORDERS	Somme de TOTAL_REVENUE	ABC_CLASS	PERFORMANCE_TIER
Triple-buffered static utilization	Fashion	69,00	776,79 C	Long Tail	
Configurable multi-state analyzer	Sports	62,00	1 103,30 C	Long Tail	
Down-sized fault-tolerant hardware	Beauty	66,00	1 210,32 C	Long Tail	
Le confort d'innover plus rapidement	Books	64,00	1 523,97 C	Long Tail	
Customer-focused 5thgeneration installation	Electronics	56,00	1 719,86 C	Long Tail	
Operative homogeneous moderator	Sports	74,00	2 083,14 C	Long Tail	
Reduced encompassing customer loyalty	Electronics	65,00	2 806,02 C	Long Tail	
Balanced interactive secured line	Books	77,00	3 148,50 C	Long Tail	
Customer-focused asymmetric Internet solution	Home	70,00	3 912,75 C	Long Tail	
Triple-buffered context-sensitive focus group	Beauty	79,00	3 932,10 C	Long Tail	
Monitored explicit instruction set	Home	73,00	4 658,19 C	Long Tail	
Robust asynchronous matrices	Electronics	58,00	5 125,12 C	Long Tail	
Ameliorated dedicated utilization	Sports	74,00	5 841,63 C	Long Tail	
Reactive mission-critical toolset	Electronics	63,00	5 956,47 C	Long Tail	
La liberté de concrétiser vos projets plus facilement	Fashion	65,00	6 298,05 C	Long Tail	
Virtual full-range analyzer	Books	60,00	6 914,40 C	Long Tail	
Ergonomic Sthegeneration project	Fashion	80,00	7 476,81 C	Long Tail	
Le pouvoir d'avancer à la pointe	Beauty	57,00	7 754,06 C	Long Tail	
Le droit d'évoluer autrement	Fashion	75,00	8 737,83 C	Long Tail	
Profit-focused multimedia projection	Home	58,00	8 795,20 C	Long Tail	
De-engineered content-based focus group	Electronics	73,00	9 288,52 C	Long Tail	
Diverse secondary definition	Toys	69,00	9 339,84 C	Long Tail	
Total fresh-thinking task-force	Toys	71,00	9 380,00 C	Long Tail	
Monitored actuating algorithm	Sports	63,00	9 437,22 C	Long Tail	
Cross-group bifurcated adapter	Toys	71,00	9 854,41 C	Long Tail	
Intuitive full-range open system	Beauty	84,00	10 226,72 C	Long Tail	
Le plaisir d'évoluer avant-tout	Toys	59,00	10 254,44 C	Long Tail	
L'avantage d'innover à sa source	Toys	63,00	11 320,38 C	Long Tail	
Cloned national success	Electronics	70,00	11 468,88 C	Long Tail	
Implemented bottom-line moratorium	Electronics	73,00	11 494,30 C	Long Tail	
Expanded zero-defect parallelism	Fashion	66,00	12 438,80 C	Long Tail	
Total		14 559,00	7 414 686,30		

FIGURE 7.3 – Power BI : Capture du Dashboard "Performance produit"

Chapitre 8

Conclusion

Ce travail pratique a permis de mettre en œuvre une chaîne de traitement de données complète et moderne. En partant d'une base de données opérationnelle, nous avons réussi à :

- Centraliser les données dans un Data Lake S3.
- Exploiter la puissance de calcul de Snowflake.
- Industrialiser les transformations avec dbt et le SQL modulaire.
- Automatiser le tout via l'orchestrateur Airflow.
- Valoriser les données via des tableaux de bord Power BI.

Cette architecture garantit la scalabilité, la maintenabilité et la qualité des données pour l'entreprise ShopStream.