

TP de Business Intelligence

TP Complet

Pipeline OLTP → ETL → DWH → Power BI

Exemple : TechStore

Réalisé par :

Imane Maliki , Hanae Talebi

Formation : Big Data Analysis

Date : 28–11–2025

Année Universitaire 2025–2026

Résumé Exécutif

Ce TP présente la mise en place complète d'un pipeline Business Intelligence pour l'entreprise fictive TechStore. L'objectif principal est d'extraire, transformer et charger les données provenant de la base transactionnelle (OLTP) vers un Data Warehouse modélisé en schéma en étoile, afin de permettre l'analyse et la visualisation via Power BI.

Le processus repose sur :

- une base OLTP MySQL contenant les données opérationnelles (clients, produits, commandes),
- un pipeline ETL construit avec Pentaho Data Integration,
- un Data Warehouse MySQL structuré autour d'une table de faits et trois dimensions,
- un reporting final sous Power BI.

Ce rapport détaille l'architecture technique, les modèles de données OLTP et DWH, la génération des datasets, ainsi que l'ensemble du processus ETL utilisé pour alimenter le Data Warehouse.

Chapitre 1

Introduction

TechStore est une entreprise spécialisée dans la vente en ligne de produits électroniques.

La direction souhaite analyser :

- le chiffre d'affaires par ville,
- les ventes par catégorie de produits,
- l'évolution des ventes au fil du temps,
- les comportements d'achat des clients.

La base de données opérationnelle existante (OLTP) n'est pas adaptée à la création de rapports décisionnels. Un pipeline BI complet doit donc être mis en place pour structurer et analyser les données.

Objectifs du projet

- centraliser et nettoyer les données opérationnelles,
- concevoir un Data Warehouse optimisé pour l'analyse,
- automatiser le chargement via un processus ETL,
- produire des tableaux de bord interactifs.

Chapitre 2

Architecture Technique

L'architecture globale du projet comprend quatre composants principaux :

1. **Base OLTP MySQL** : stockage transactionnel normalisé.
2. **Pentaho Data Integration** : extraction, transformation et chargement (ETL).
3. **Data Warehouse MySQL** : schéma en étoile optimisé pour l'analyse.
4. **Power BI** : reporting et visualisation.

Diagramme d'Architecture

```
TP_Pipeline_OLTP_ETL_DWH_PowerBI/
|   └── sql/
|       ├── 01_create_oltp.sql
|       ├── 02_create_dwh.sql
|       ├── 03_queries_olap.sql
|       └── 04_verification.sql
|
|   └── data/
|       ├── clients.csv
|       ├── produits.csv
|       ├── commandes.csv
|       └── lignes_commandes.csv
|
|   └── scripts/
|       └── generate_data.py
|
|   └── pentaho/
|       ├── transformations/
|       |   ├── dim_client.ktr
|       |   ├── dim_produit.ktr
|       |   ├── dim_date.ktr
|       |   └── fact_ventes.ktr
|       └── jobs/
|           └── job_etl_complet.kjb
|
|   └── powerbi/
|       └── Dashboard_Ventes_TechStore.pbix
|
|   └── documentation/
|       ├── Dashboard_Ventes_TechStore.pdf
|       ├── Rapport_TP_BI.pdf
|       └── Captures_Ecran/
|           ├── pentaho_dim_client.png
|           ├── pentaho_fact_ventes.png
|           ├── powerbi_dashboard.png
|           └── ...
|
└── README.md
```

Chapitre 3

Modélisation OLTP

La base OLTP comprend quatre tables principales :

- **clients** : informations personnelles des clients,
- **produits** : catalogue des produits,
- **commandes** : en-tête des commandes,
- **lignes_commandes** : détail des articles commandés.

Description des tables

Table : clients

Colonne	Type	Description
id_client	INT	Clé primaire
nom, prenom	VARCHAR	Identité du client
email	VARCHAR	Identifiant unique
ville	VARCHAR	Ville de résidence
date_inscription	DATE	Date d'inscription

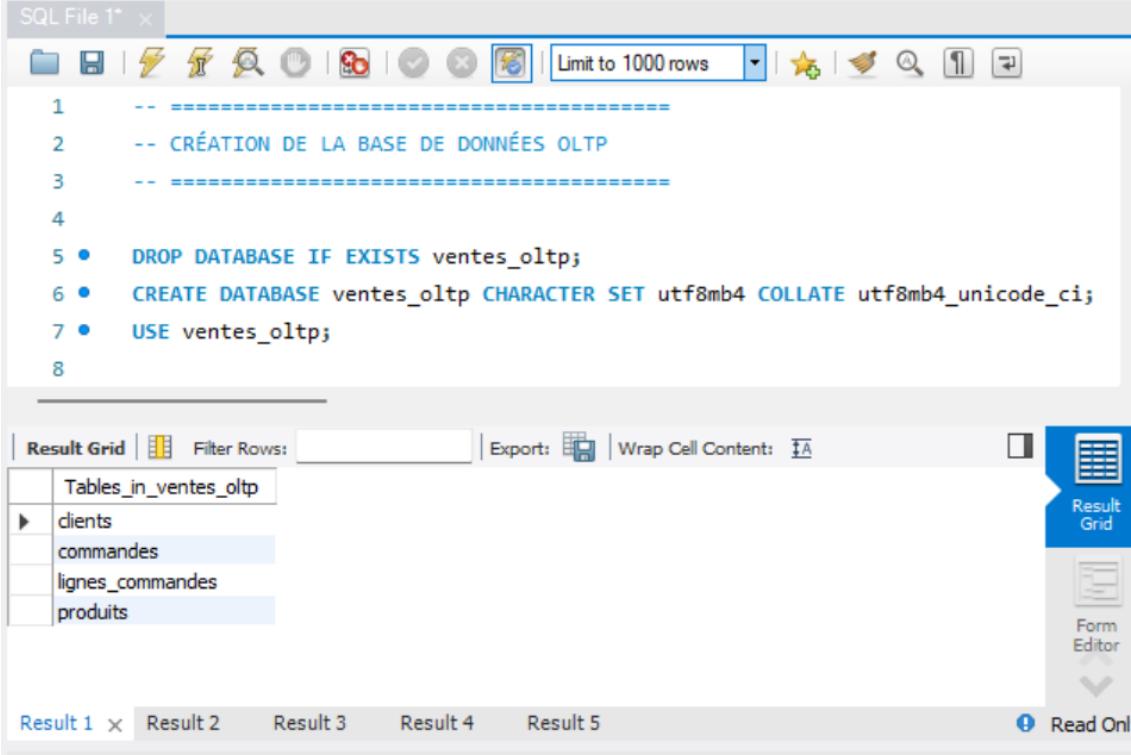
Relations du modèle

- Un client peut passer plusieurs commandes ;
- Une commande contient une ou plusieurs lignes ;
- Un produit peut apparaître dans plusieurs lignes.

Script SQL Complet - Base OLTP

- Création des tables dans la base de données ;

- **Résultat : ventes_oltp** : quatre tables ont été créées, et les relations entre elles sont définies via les clés étrangères



SQL File 1*

```

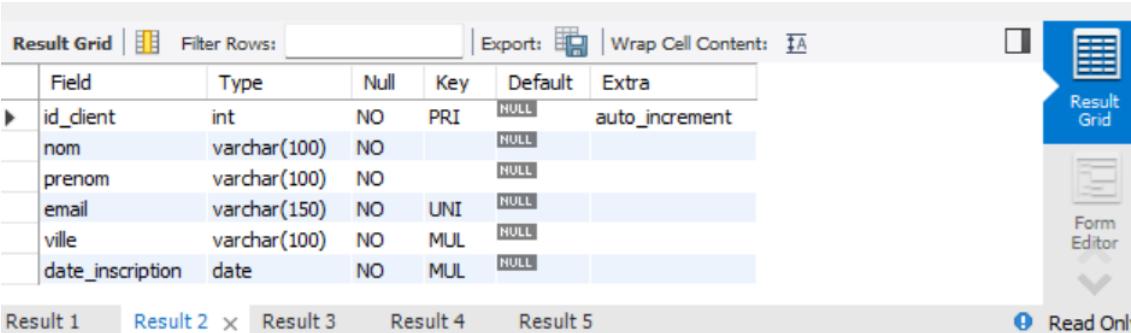
1 -- =====
2 -- CRÉATION DE LA BASE DE DONNÉES OLTP
3 -- =====
4
5 • DROP DATABASE IF EXISTS ventes_oltp;
6 • CREATE DATABASE ventes_oltp CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
7 • USE ventes_oltp;
8

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 1 × Result 2 Result 3 Result 4 Result 5 Read Only

Tables_in_ventes_oltp
clients
commandes
lignes_commandes
produits

FIGURE 3.1 – Capture 1 : Structure OLTP



Result Grid | Filter Rows: Export: Wrap Cell Content: Result 1 × Result 2 Result 3 Result 4 Result 5 Read Only

Field	Type	Null	Key	Default	Extra
id_client	int	NO	PRI	NULL	auto_increment
nom	varchar(100)	NO		NULL	
prenom	varchar(100)	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
ville	varchar(100)	NO	MUL	NULL	
date_inscription	date	NO	MUL	NULL	

FIGURE 3.2 – Capture 2 :Table Client

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Field	Type	Null	Key	Default	Extra
id_produit	int	NO	PRI	NULL	auto_increment
nom_produit	varchar(200)	NO		NULL	
categorie	varchar(100)	NO	MUL	NULL	
prix_unitaire	decimal(10,2)	NO		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Read Only

FIGURE 3.3 – Capture 3 :Table Produit

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Field	Type	Null	Key	Default	Extra
id_commande	int	NO	PRI	NULL	auto_increment
id_client	int	NO	MUL	NULL	
date_commande	date	NO	MUL	NULL	
montant_total	decimal(10,2)	NO		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Read Only

FIGURE 3.4 – Capture 4 :Table Commandes

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Field	Type	Null	Key	Default	Extra
id_ligne	int	NO	PRI	NULL	auto_increment
id_commande	int	NO	MUL	NULL	
id_produit	int	NO	MUL	NULL	
quantite	int	NO		NULL	
prix_unitaire	decimal(10,2)	NO		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 Read Only

FIGURE 3.5 – Capture 5 : Table lignes de Commandes

Chapitre 4

Génération des données : Méthodologie, volumes et répartition

Dans cette partie, nous générerons un jeu de données complet et réaliste afin d'alimenter notre base OLTP. L'objectif est de créer un dataset de **100 000 lignes**, réparti entre 4 tables : Clients, Produits, Commandes et Lignes de commandes.

Méthodologie

Nous utilisons Python et la bibliothèque **Faker** pour produire des données synthétiques cohérentes. Les volumes générés sont les suivants :

- **10 000** clients ;
- **500** produits répartis en catégories ;
- **20 000** commandes ;
- **100 000** lignes de commandes.

Les données sont réparties sur une période de 3 ans (2022–2024) afin de simuler un historique réaliste.

Extrait du Script Python de génération

```
import pandas as pd
from faker import Faker
import random
from datetime import datetime, timedelta

# Initialisation
fake = Faker('fr_FR')
random.seed(42)
Faker.seed(42)

print("Génération des données en cours...")

# =====
# GÉNÉRATION DES CLIENTS (10 000)
# =====
print("Génération de 10 000 clients...")

villes = ['Paris', 'Lyon', 'Marseille', 'Toulouse', 'Nice', 'Nantes',
          'Strasbourg', 'Montpellier', 'Bordeaux', 'Lille', 'Rennes', 'Reims']

clients_data = []
for i in range(1, 10001):
    clients_data.append({
        'id_client': i,
        'nom': fake.last_name(),
        'prenom': fake.first_name(),
        'email': f"client{i}@{fake.free_email_domain()}",
        'ville': random.choice(villes),
```

— Après Execution du code :

```
C:\Users\pc\Documents\S7Eidia\systemeinfo\Tpapoltp>python generate_data.py
Génération des données en cours...
Génération de 10 000 clients...
Fichier clients.csv créé : 10000 lignes
Génération de 500 produits...
Fichier produits.csv créé : 500 lignes
Génération de 20 000 commandes...
Génération de 100 000 lignes de commandes...
Fichier lignes_commandes.csv créé : 100000 lignes
Fichier commandes.csv créé : 20000 lignes

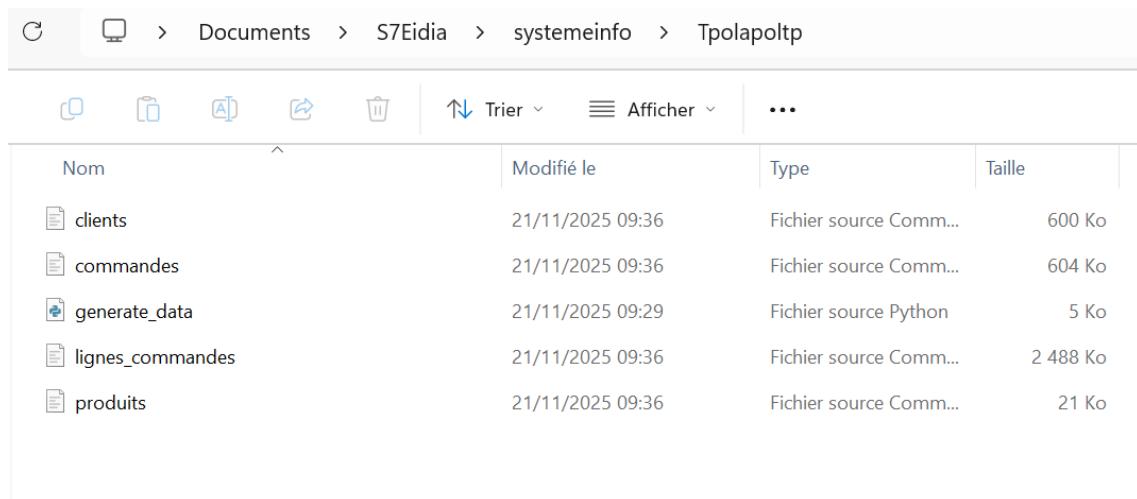
Génération terminée avec succès !
Fichiers créés : clients.csv, produits.csv, commandes.csv, lignes_commandes.csv

C:\Users\pc\Documents\S7Eidia\systemeinfo\Tpapoltp>
```

Résultat : Fichiers générés

L'exécution du script produit automatiquement les fichiers suivants :

- clients.csv
- produits.csv
- commandes.csv
- lignes_commandes.csv



A screenshot of a Windows File Explorer window. The path is shown in the address bar: C:\ > Documents > S7Eidia > systemeinfo > Tpolapoltp. Below the address bar is a toolbar with icons for back, forward, search, and file operations. To the right of the toolbar are buttons for 'Trier' (Sort) and 'Afficher' (View). A '...' button is also present. The main area is a table listing files:

Nom	Modifié le	Type	Taille
clients	21/11/2025 09:36	Fichier source Comm...	600 Ko
commandes	21/11/2025 09:36	Fichier source Comm...	604 Ko
generate_data	21/11/2025 09:29	Fichier source Python	5 Ko
lignes_commandes	21/11/2025 09:36	Fichier source Comm...	2 488 Ko
produits	21/11/2025 09:36	Fichier source Comm...	21 Ko

Aperçu des données générées

Ci-dessous un aperçu des premières lignes des fichiers :

Extrait de clients.csv

```
id_client,nom,prenom,email,ville,date_inscription
1,Riou,Frédéric,client1@voila.fr,Rennes,2024-01-22
2,Moulin,Maurice,client2@laposte.net,Lyon,2023-04-29
3,Labbé,Nicole,client3@hotmail.fr,Paris,2025-05-26
4,Briand,Arthur,client4@voila.fr,Reims,2023-04-15
5,Carpentier,Nicolas,client5@wanadoo.fr,Nice,2025-06-13
6,Michel,Gabrielle,client6@sfr.fr,Toulouse,2025-08-26
7,Boutin,Émilie,client7@noos.fr,Toulouse,2023-10-29
8,Pons,Jacqueline,client8@bouygtel.fr,Marseille,2022-12-01
9,Thierry,Susan,client9@free.fr,Reims,2025-11-09
```

Extrait de produits.csv

```
id_produit,nom_produit,categorie,prix_unitaire
1,MacBook Pro Ultra,Ordinateurs,100.63
2,MacBook Pro Standard,Ordinateurs,778.44
3,MacBook Pro Ultra,Ordinateurs,349.86
4,MacBook Pro Standard,Ordinateurs,1669.86
5,Lenovo ThinkPad Standard,Ordinateurs,1275.3
6,HP Pavilion Lite,Ordinateurs,1333.11
7,HP Pavilion Pro,Ordinateurs,1339.54
```

Extrait de commandes.csv

```
id_commande,id_client,date_commande,montant_total
1,69,2023-01-24,13063.92
2,8204,2022-01-27,24348.16
3,8052,2024-02-06,8588.92
4,6813,2024-04-08,8465.77
5,6114,2024-03-19,14608.4
6,8021,2024-02-26,35228.03
7,9639,2023-09-23,17034.95
```

Extrait de lignes_commandes.csv

```
id_produit,nom_produit,categorie,prix_unitaire
1,MacBook Pro Ultra,Ordinateurs,100.63
2,MacBook Pro Standard,Ordinateurs,778.44
3,MacBook Pro Ultra,Ordinateurs,349.86
4,MacBook Pro Standard,Ordinateurs,1669.86
5,Lenovo ThinkPad Standard,Ordinateurs,1275.3
6,HP Pavilion Lite,Ordinateurs,1333.11
7,HP Pavilion Pro,Ordinateurs,1329.54
```

Chapitre 5

Modélisation du Data Warehouse

Le Data Warehouse est construit selon un schéma en étoile comprenant :

- **FactVentes** (table centrale),
- **DimClient**,
- **DimProduit**,
- **DimDate**.

Avantages du schéma en étoile

- performances optimisées pour l'analyse,
- structure claire et intuitive,
- peu de jointures,
- compatible avec les outils de BI.

5.0.1 Méthode 1 : Importation via MySQL Workbench

Étape 1 : Ouvrir MySQL Workbench Lancez MySQL Workbench et connectez-vous à votre serveur MySQL.

Étape 2 : Sélectionner la base de données et Importation de toutes les bases de données .csv - Les opérations se fait pour tout les fichier csv

SCHEMAS

ventes_oltp

▼ ventes_oltp
▶ Tables
 Views
 Stored Procedures
 Functions

Showing loaded schemas only

Administration Schemas

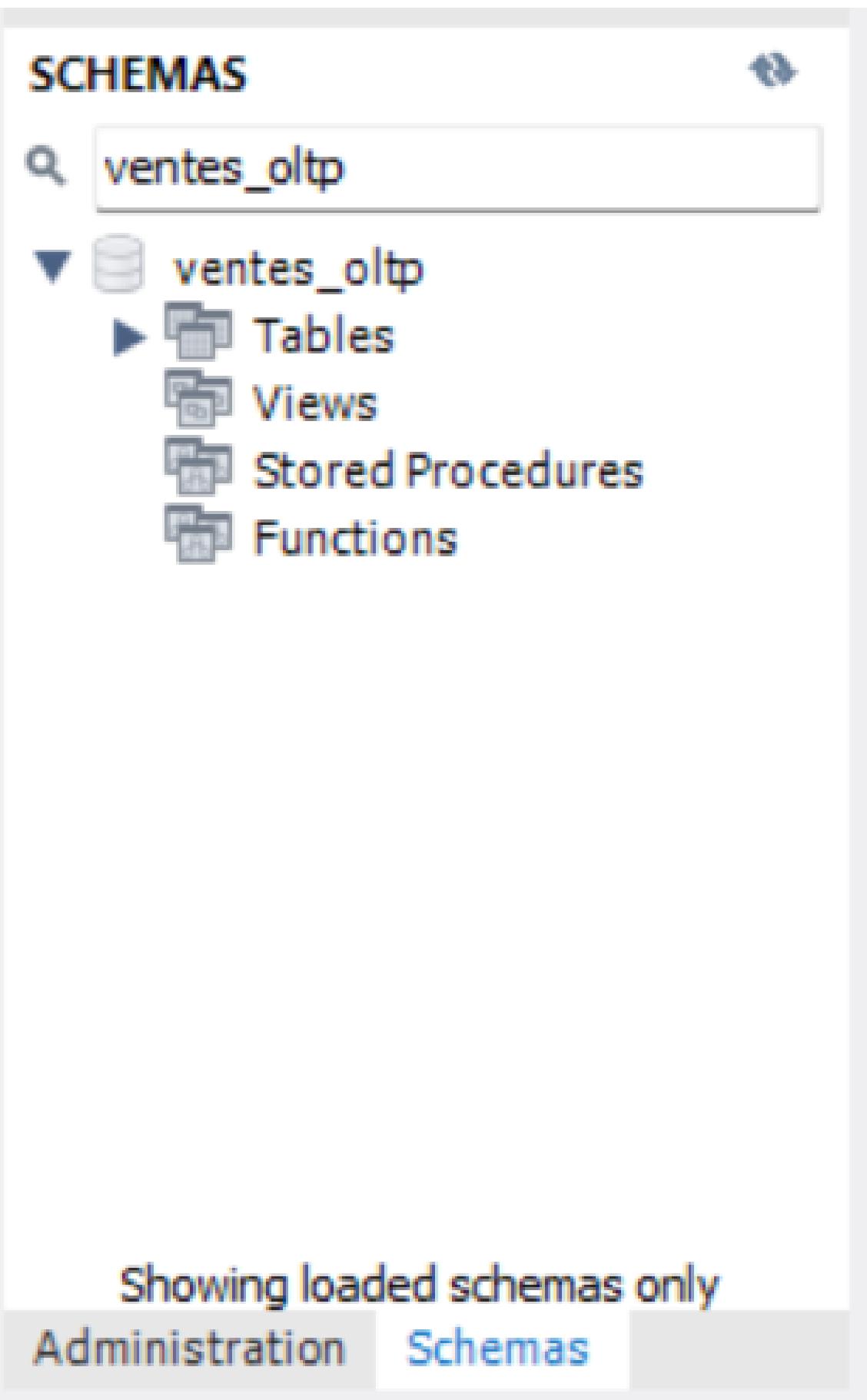


FIGURE 5.1 – Affichage du schéma ventes_oltp

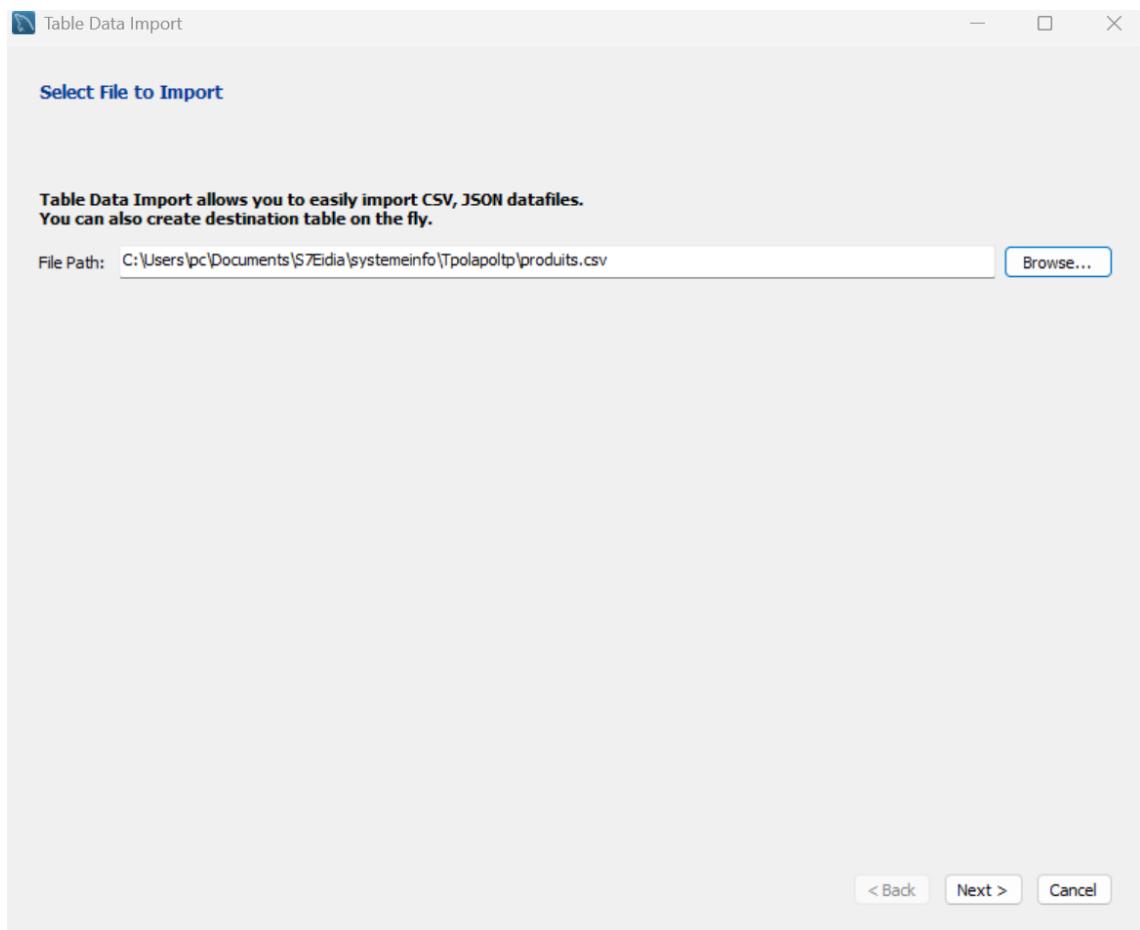


FIGURE 5.2 – Import de données dans MySQL Workbench

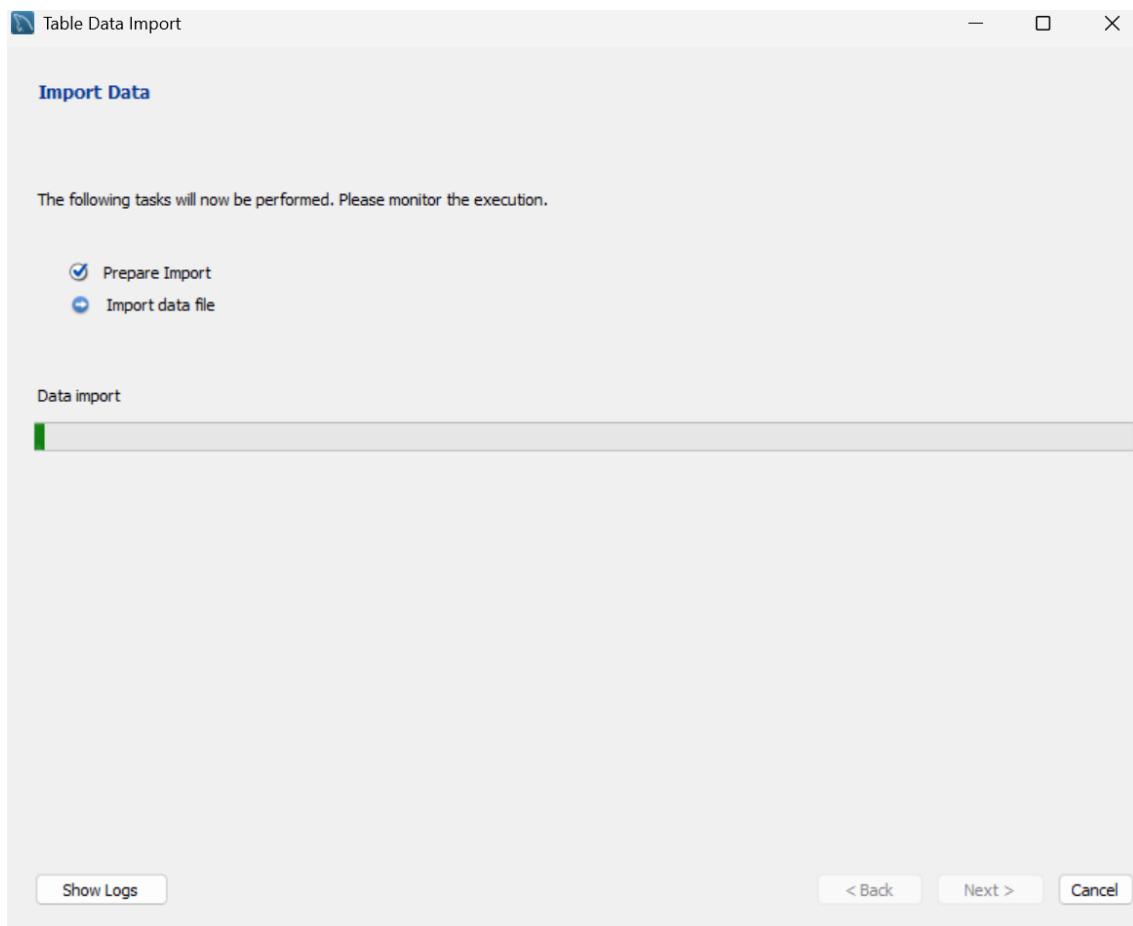


FIGURE 5.3 – Import de données dans MySQL Workbench steps

Étape 3 : Script SQL Complet du Data Warehouse

- Exécution du code SQL pour créer la structure du Data Warehouse. Ce script prépare les tables vides qui seront alimentées par Pentaho.

SQL File 1* clients - Table SQL File 2* x

73 --- =====

74 • SHOW TABLES;

75 • DESCRIBE DimClient;

76 • DESCRIBE DimProduit;

77 • DESCRIBE DimDate;

78 • DESCRIBE FactVentes;|

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid

Field	Type	Null	Key	Default	Extra
id_vente	int	NO	PRI	NULL	auto_increment
id_client_dim	int	NO	MUL	NULL	
id_produit_dim	int	NO	MUL	NULL	
id_date_dim	int	NO	MUL	NULL	
quantité	int	NO		NULL	

Result 1 Result 2 Result 3 Result 4 Result 5 x Read Only Context Help Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
47	10:04:05	SHOW TABLES	4 row(s) returned	0.016 sec / 0.000 sec
48	10:04:05	DESCRIBE DimClient	5 row(s) returned	0.015 sec / 0.000 sec
49	10:04:05	DESCRIBE DimProduit	4 row(s) returned	0.000 sec / 0.000 sec
50	10:04:05	DESCRIBE DimDate	9 row(s) returned	0.000 sec / 0.000 sec
51	10:04:05	DESCRIBE FactVentes	7 row(s) returned	0.000 sec / 0.000 sec

SQLAdditions: | Jump to | Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

FIGURE 5.4 – Création des tables pour PDI

Pentaho PDI - Configuration et Transformations

Nous entrons maintenant dans la partie centrale du TP : l'utilisation de Pentaho Data Integration pour extraire, transformer et charger les données depuis OLTP vers le DWH.

Étape 1 : Configuration des connexions à la base de données

Texte explicatif sur les connexions OLTP et DWH.

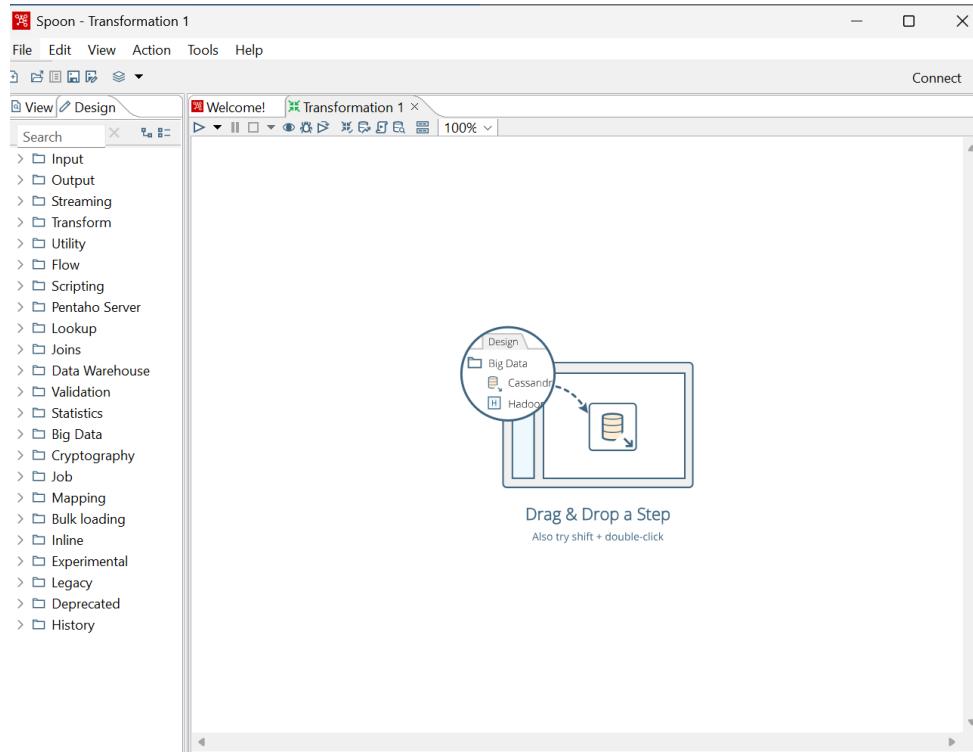


FIGURE 5.5 – Interface Pentaho

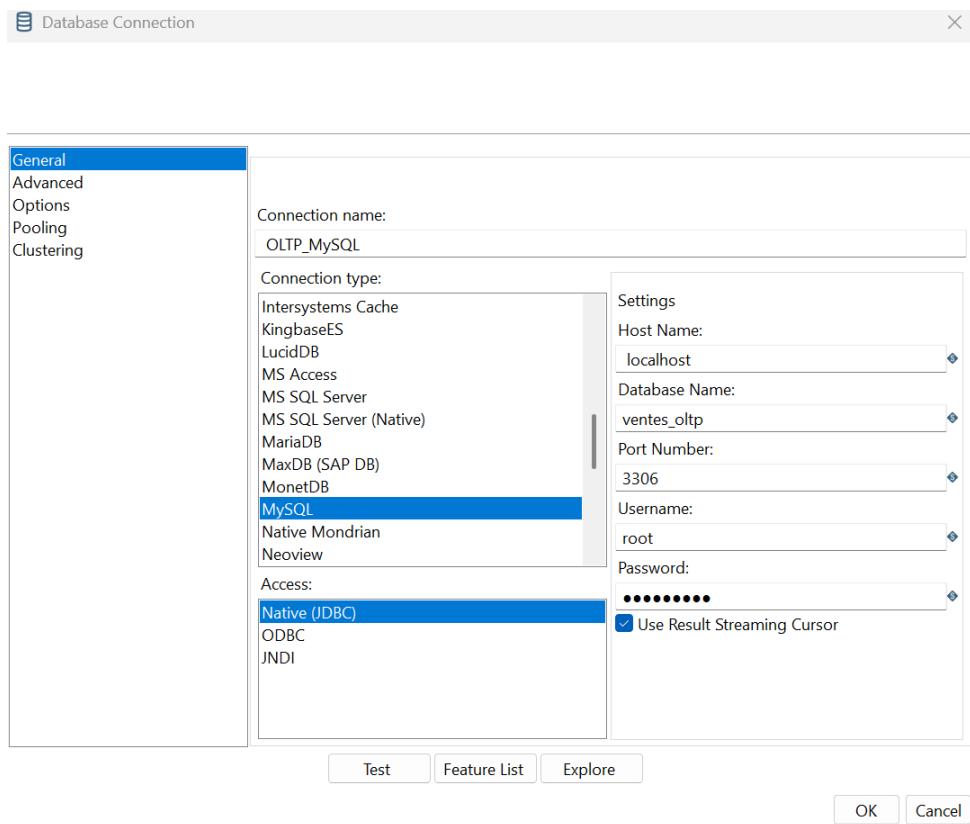


FIGURE 5.6 – Création de la connexion OLTP dans Pentaho Spoon

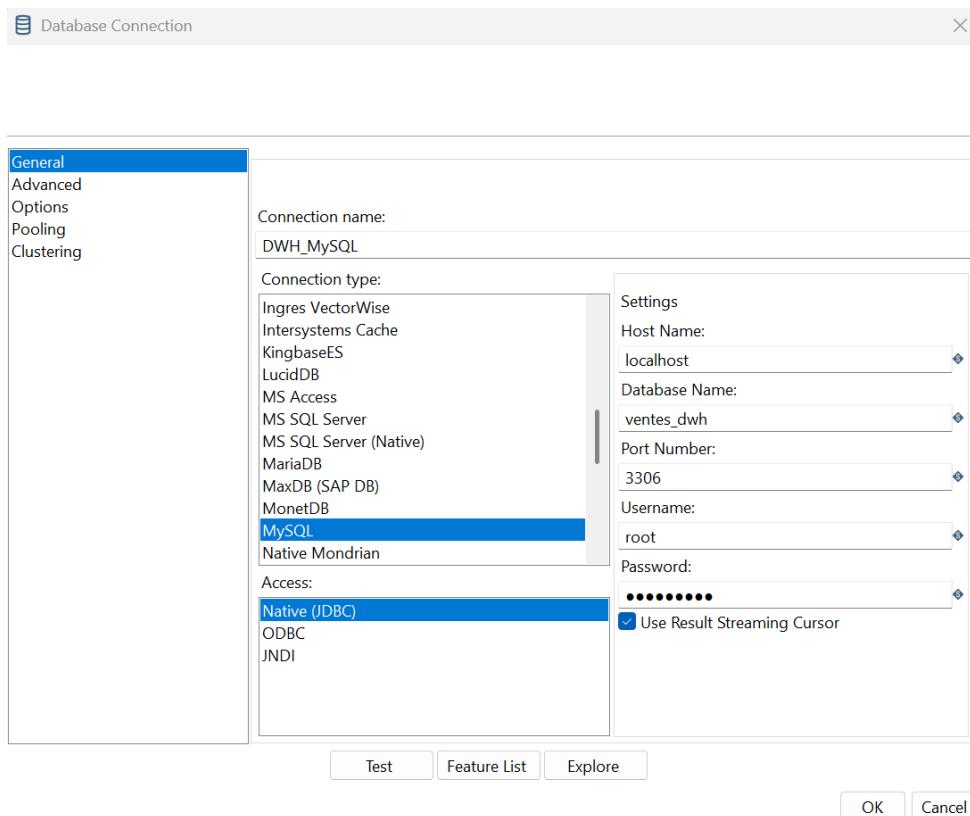


FIGURE 5.7 – Création de la connexion DWH dans Pentaho Spoon

Étape 2 : Transformation 2 - Charger DimProduit

Après l'application du texte explicatif sur la transformation DimProduit.

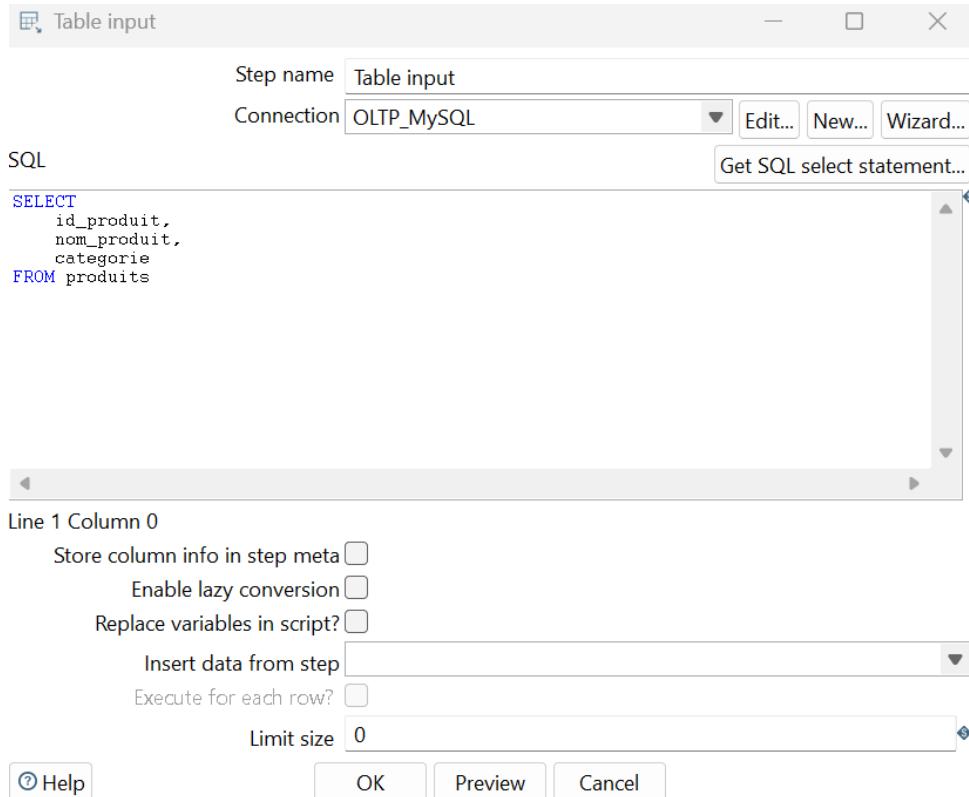


FIGURE 5.8 – Table input configuration

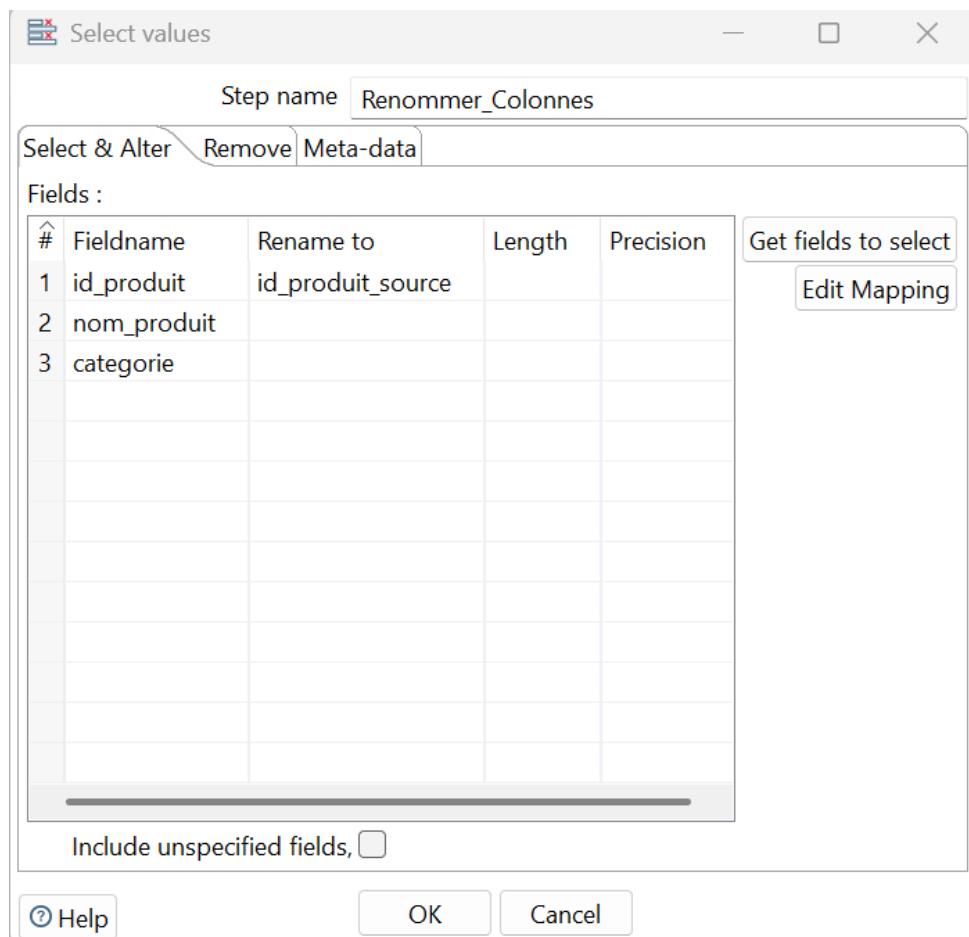


FIGURE 5.9 – Select values configuration

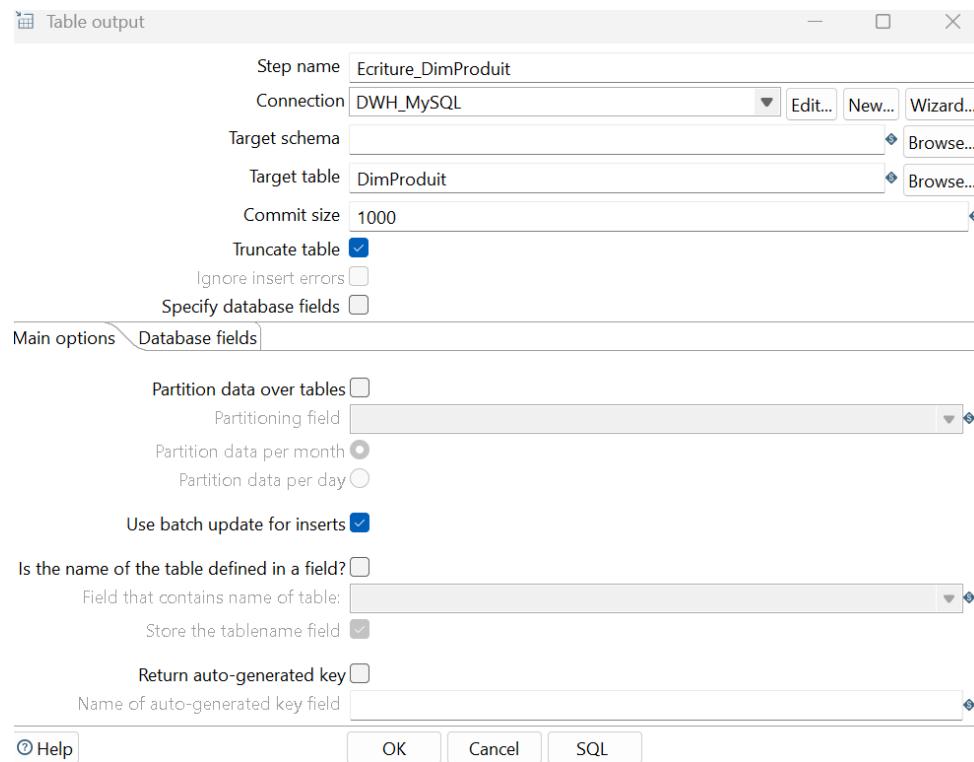


FIGURE 5.10 – Output Table

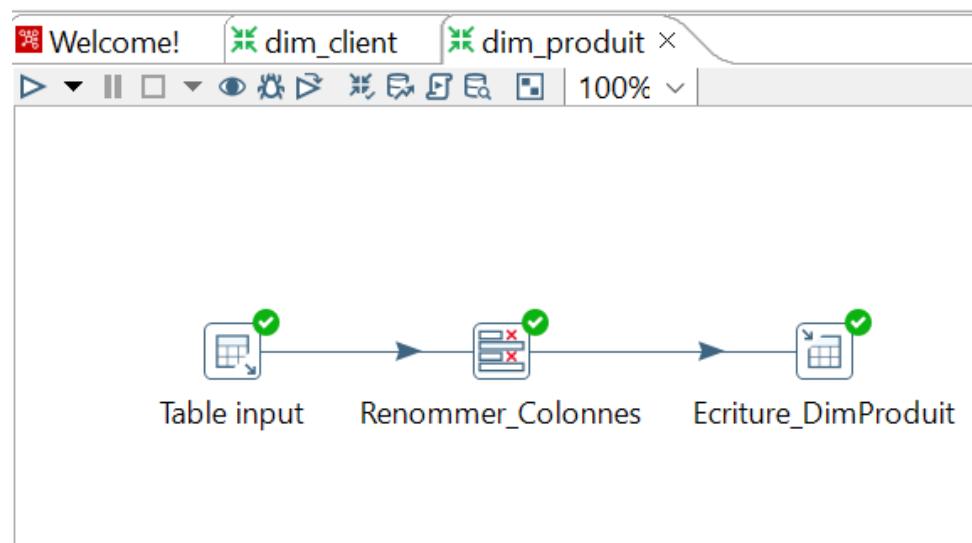


FIGURE 5.11 – Exécution de la transformation DimProduit et aperçu des lignes chargées

- On remarque comme résultat dans le panneau du bas :

```
11:31:49.311 - Spoon - Started the transformation execution.  
11:31:49.598 - dim_produit - Dispatching started for transformation [dim_produit]  
11:31:49.602 - Ecriture_DimProduit.0 - Connected to database [DWH_MySQL] (commit=1000)  
11:31:49.624 - Table input.0 - Finished reading query, closing connection  
11:31:49.626 - Table input.0 - Finished processing (I=500, O=0, R=0, W=500, U=0, E=0)  
11:31:49.644 - Renommer_Colones.0 - Finished processing (I=0, O=0, R=500, W=500, U=0, E=0)  
11:31:49.856 - Ecriture_DimProduit.0 - Finished processing (I=0, O=500, R=500, W=500, U=0, E=0)  
11:31:49.858 - Spoon - The transformation has finished!!
```

FIGURE 5.12 – Résultat d'exécution

Étape 4 : Transformation 3 - Générer DimDate

Après l'application du texte explicatif sur la transformation DimDate.

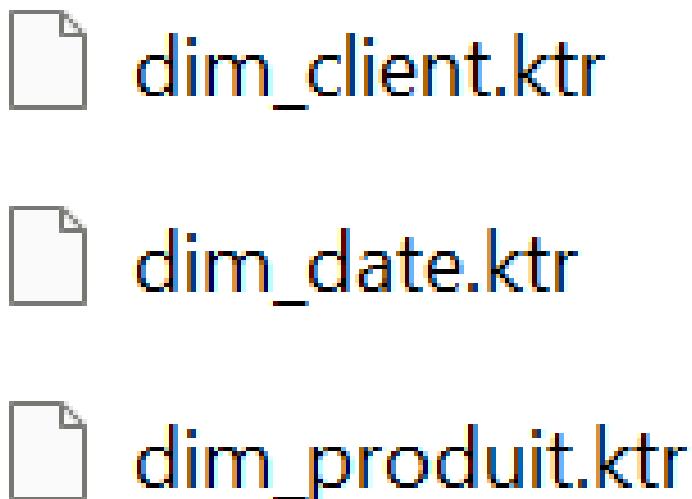


FIGURE 5.13 – Crédit de la création du DimDate.ktr

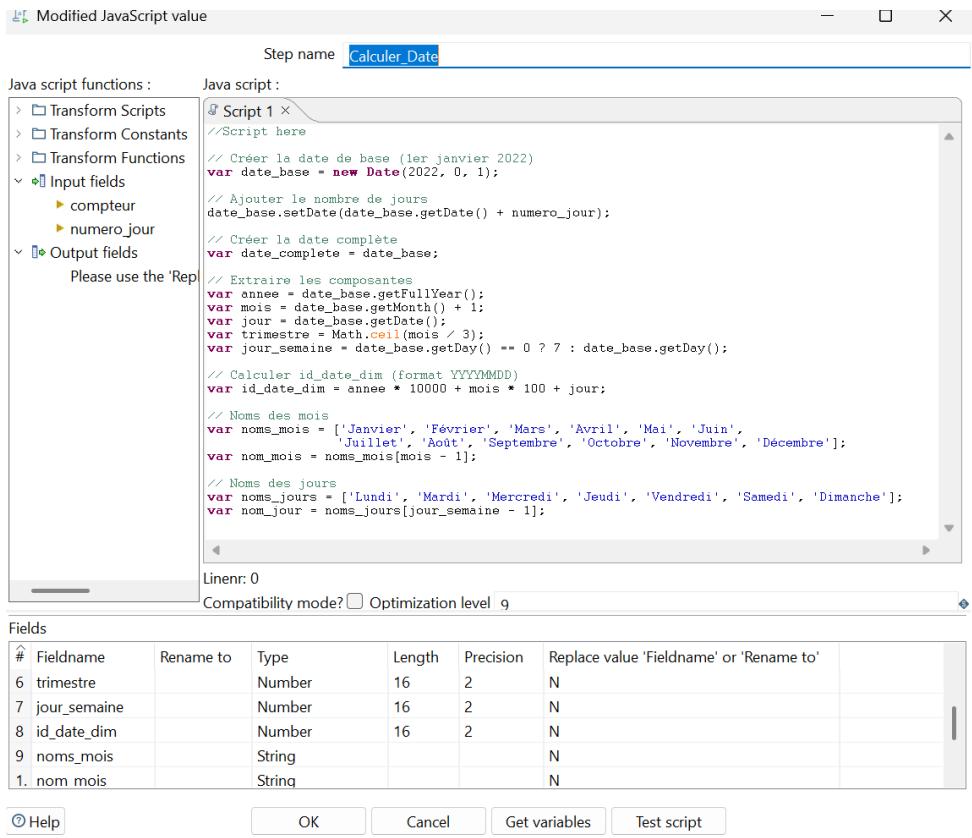


FIGURE 5.14 – Step "Modified Java Script Value" pour générer les dates complètes et les composantes

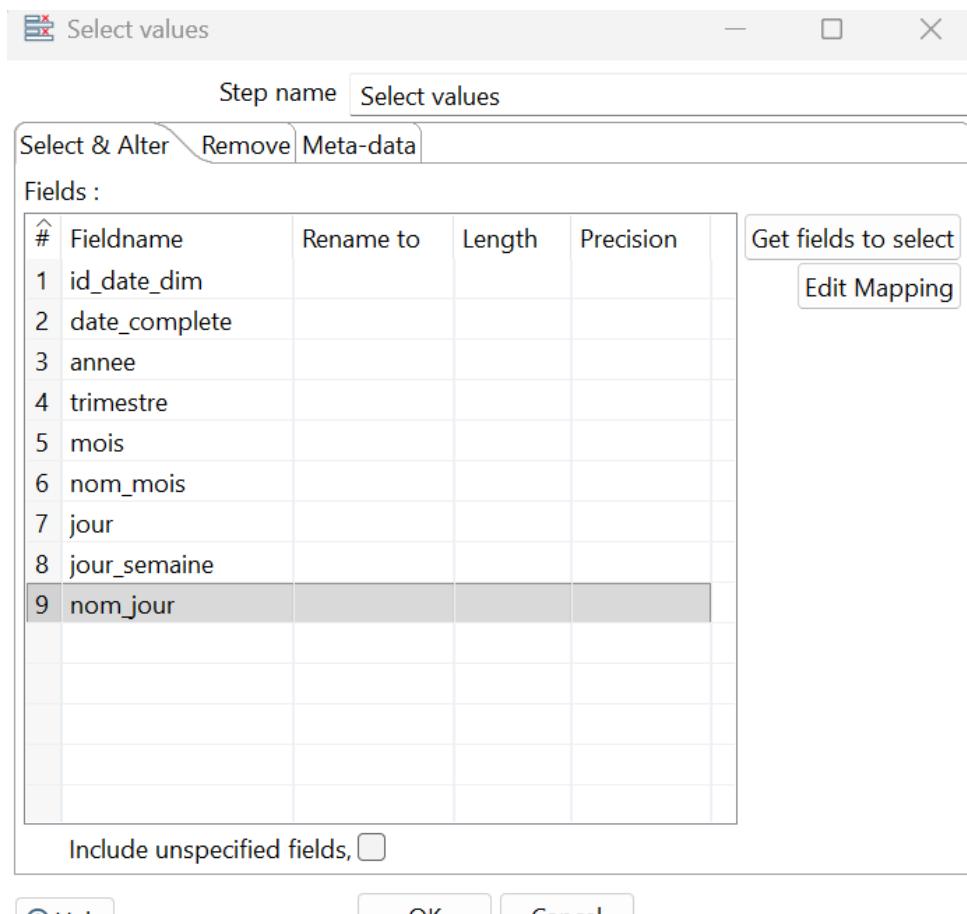


FIGURE 5.15 – Step "Select values" pour choisir les champs finaux

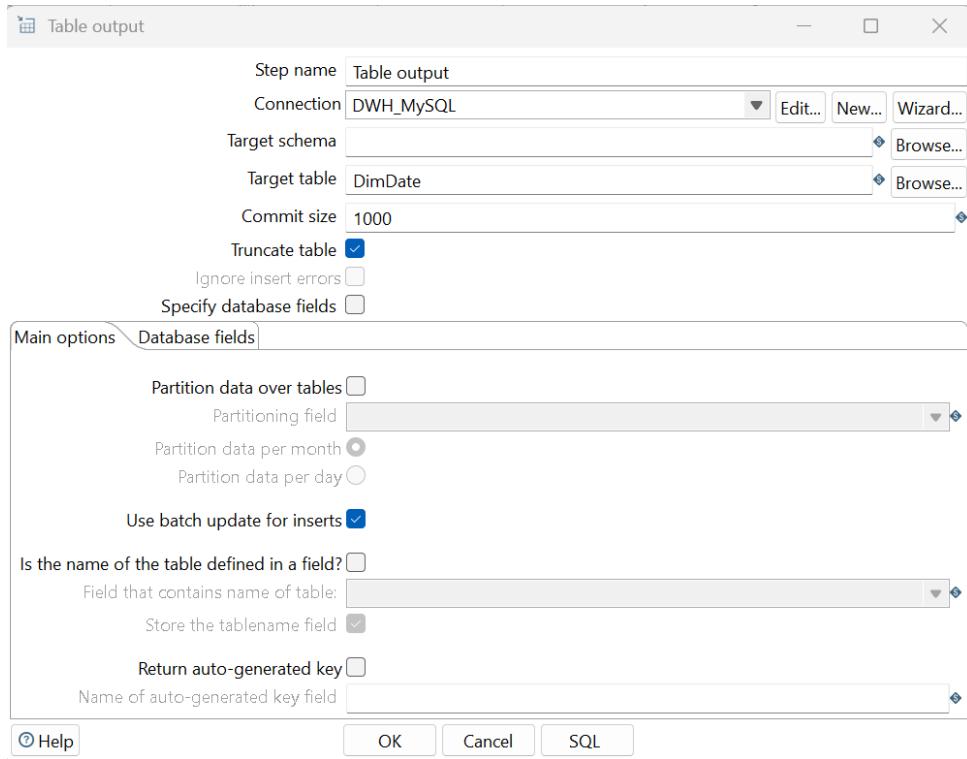
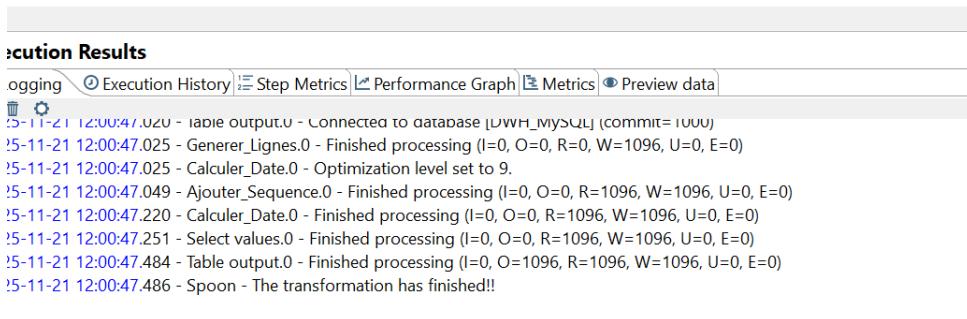
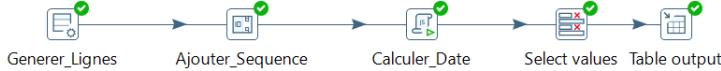


FIGURE 5.16 – Step "Table Output" pour charger DimDate dans le DWH



- Résultat attendu : La table DimDate contient 1096 lignes couvrant la période 2022-2024.

Étape 5 : Transformation 4 - Charger FactVentes

Après l'application du texte explicatif sur la transformation FactVentes.

- 📄 dim_client.ktr
- 📄 dim_date.ktr
- 📄 dim_produit.ktr
- 📄 fact_ventes.ktr

FIGURE 5.18 – Création du FactVentes.ktr

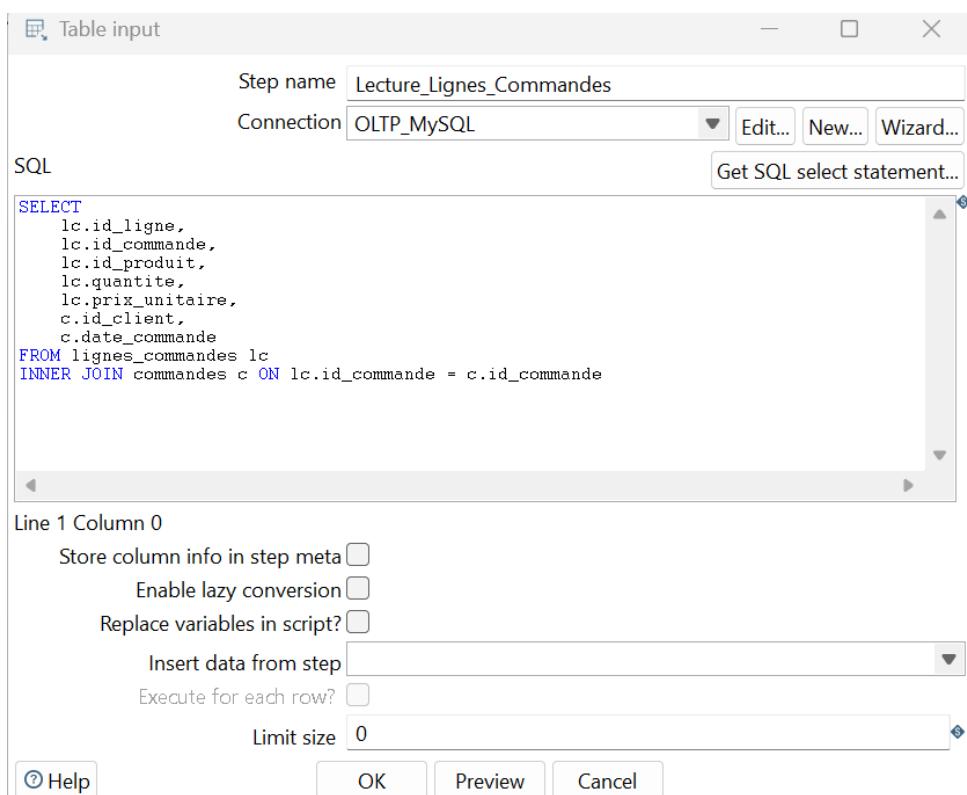


FIGURE 5.19 – Step "Table Input" pour extraire les lignes de commandes

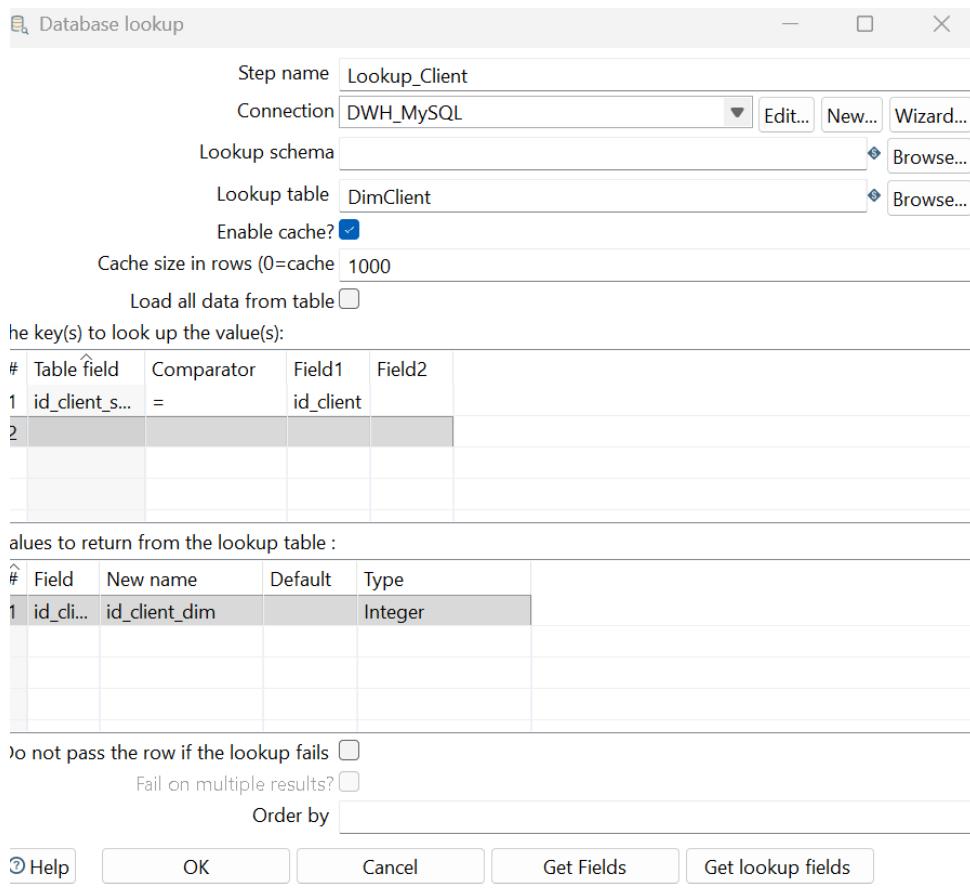


FIGURE 5.20 – Lookup DimClient pour récupérer id_client_dim

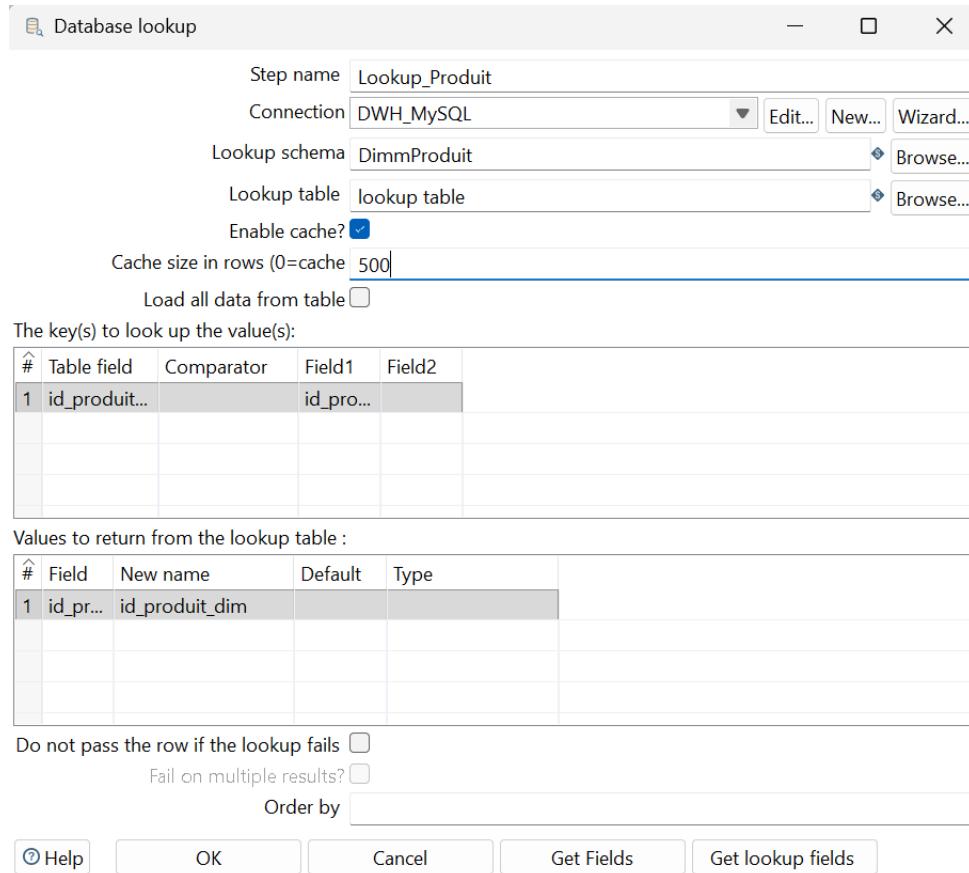


FIGURE 5.21 – Lookup DimProduit pour récupérer id_produit_dim

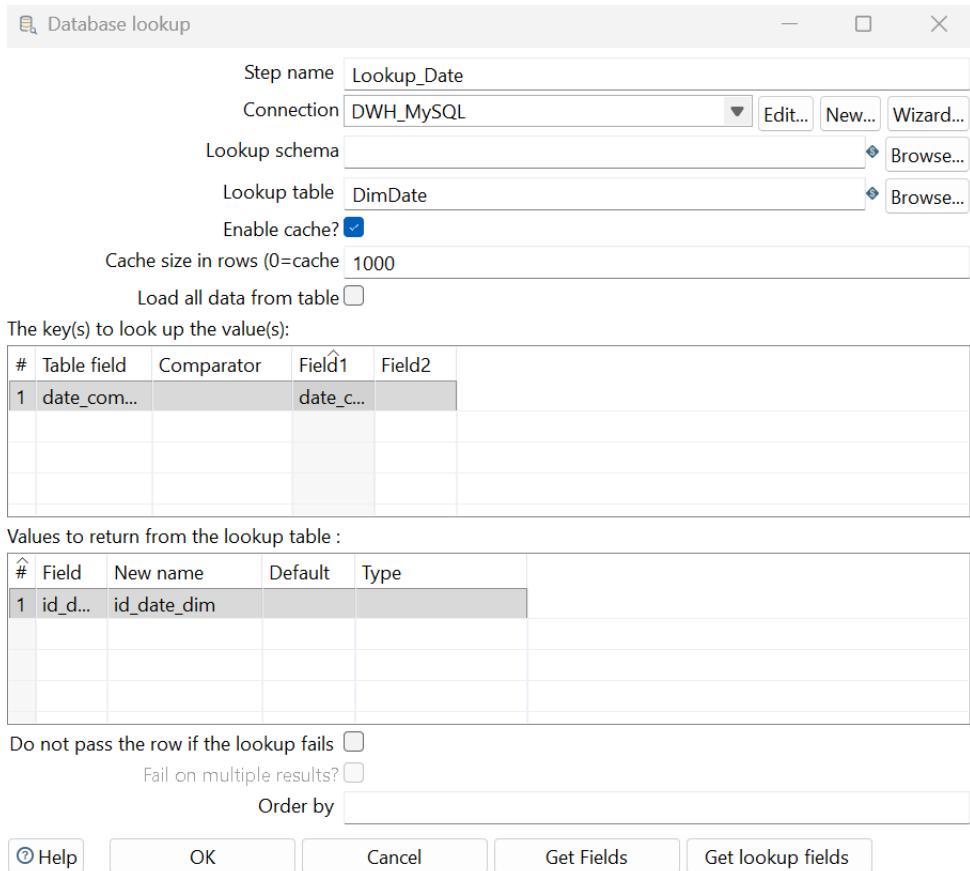


FIGURE 5.22 – Lookup DimDate pour récupérer id_date_dim

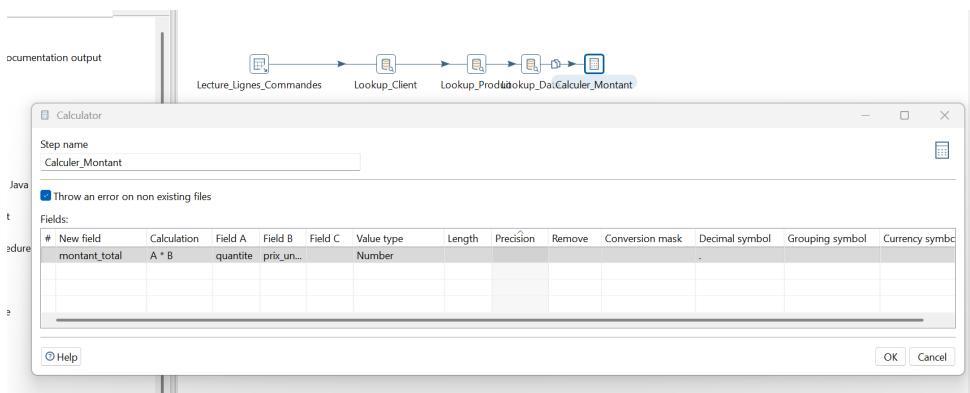


FIGURE 5.23 – Step "Calculator" pour calculer le montant total

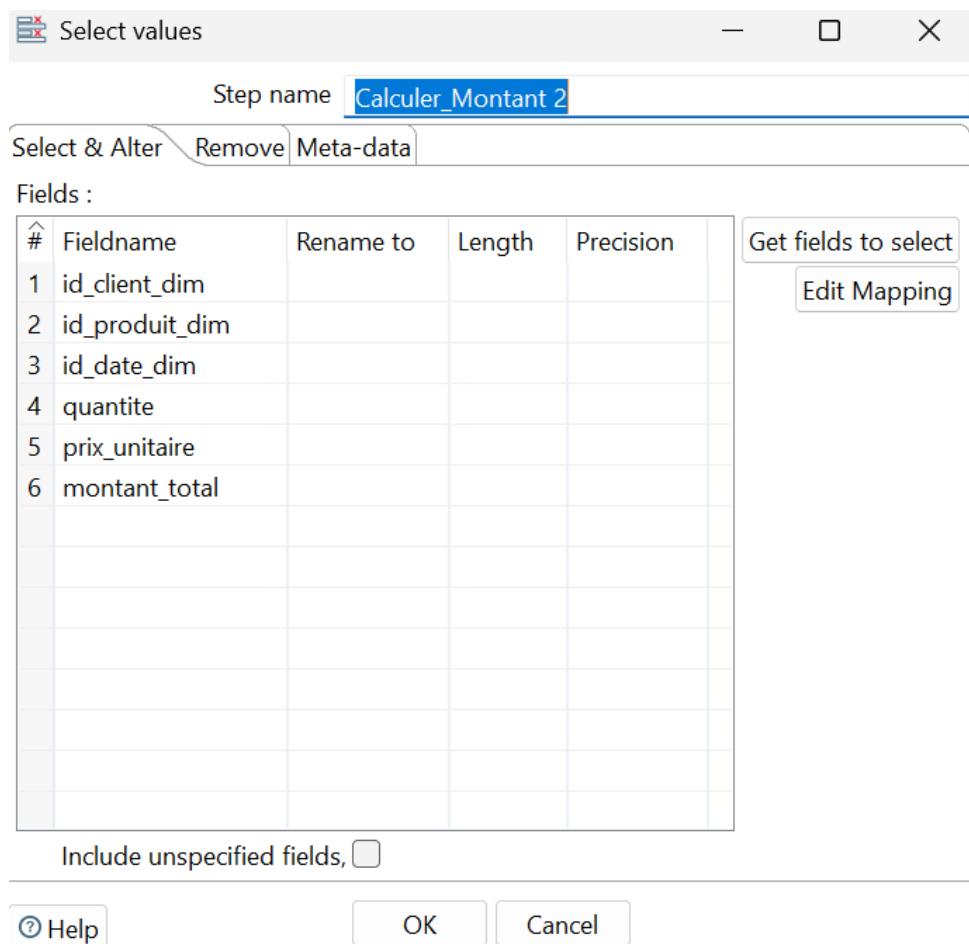


FIGURE 5.24 – Step "Select values" pour choisir les champs finaux

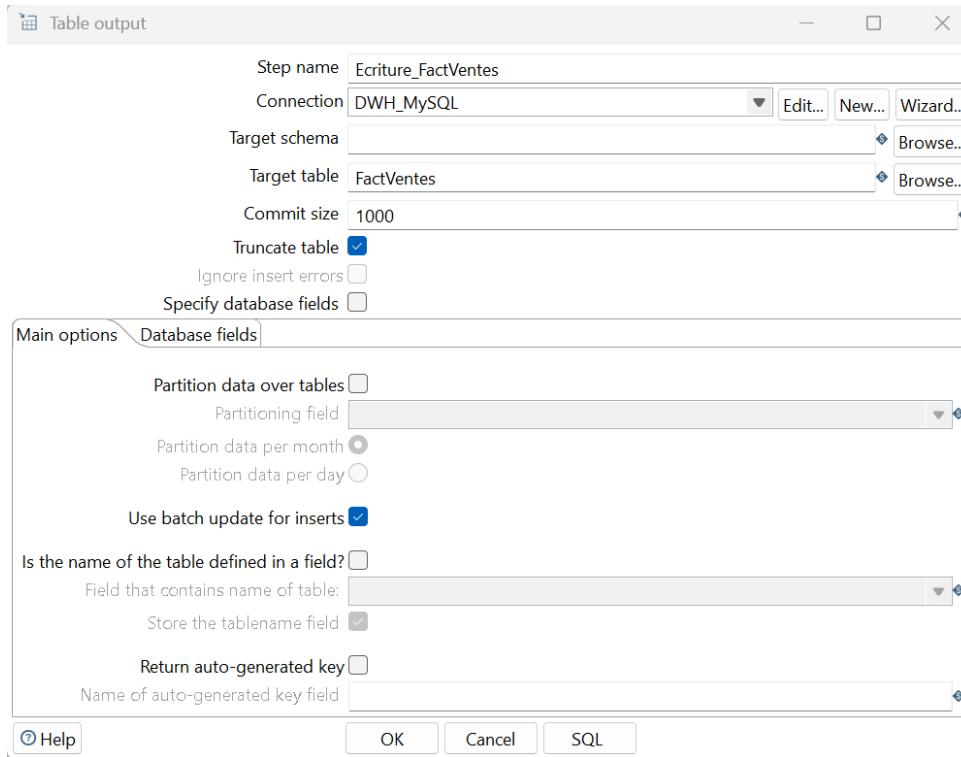


FIGURE 5.25 – Step "Table Output" pour écrire dans FactVentes



FIGURE 5.26 – Exécution de la transformation FactVentes et aperçu des lignes chargées

- Résultat attendu : La table FactVentes contient environ 100 000 lignes.

```

2025-11-22 00:28:19.522 - Calculer_Montant.0 - Linenr 100000
2025-11-22 00:28:19.538 - Calculer_Montant.0 - Finished processing (I=0, O=0, R=100000, W=100000, U=0, E=0)
2025-11-22 00:28:21.661 - Calculer_Montant 2.0 - linenr 100000
2025-11-22 00:28:21.694 - Calculer_Montant 2.0 - Finished processing (I=0, O=0, R=100000, W=100000, U=0, E=0)
2025-11-22 00:28:23.835 - Ecriture_FactVentes.0 - linenr 100000
2025-11-22 00:28:23.864 - Ecriture_FactVentes.0 - Finished processing (I=0, O=100000, R=100000, W=100000, U=0, E=0)
2025-11-22 00:28:23.866 - Spoon - The transformation has finished!!

```

FIGURE 5.27 – Résultat d'exécution

Job d'orchestration Pentaho : job_etl_complet.kjb

Maintenant que nous avons créé les 4 transformations, nous allons les orchestrer dans un Job qui les exécutera dans le bon ordre.

Pourquoi un Job ?

Un Job permet de :

- Exécuter plusieurs transformations dans un ordre précis
- Gérer les dépendances entre transformations
- Gérer les erreurs et les chemins d'exécution conditionnels
- Planifier l'exécution automatique (via cron ou Task Scheduler)

Étape 1 : Créez un nouveau Job

Dans Pentaho Spoon, cliquez sur **File → New → Job** Sauvegardez immédiatement : **job_etl_complet.kjb**



FIGURE 5.28 – Création d'un nouveau Job dans Pentaho Spoon

Étape 2 : Ajouter l'entrée START

Sous **General**, trouvez **START**

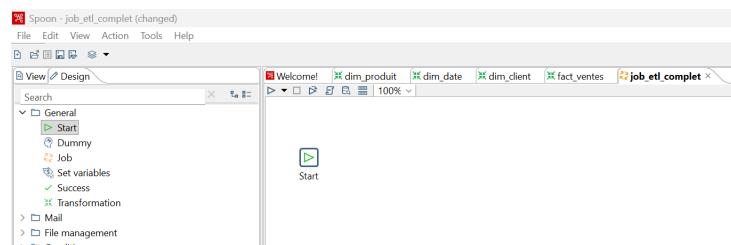


FIGURE 5.29 – Step START ajouté au Job

Étape 3 : Ajouter la transformation DimClient

Sous **General**, trouvez **Transformation** Glissez-déposez sur le canvas et configuez :

- Job entry name : **Charger_DimClient**
- Sélectionner le fichier **dim_client.ktr**

Créez un lien depuis **START** vers **Charger_DimClient** (type : **Unconditional**)

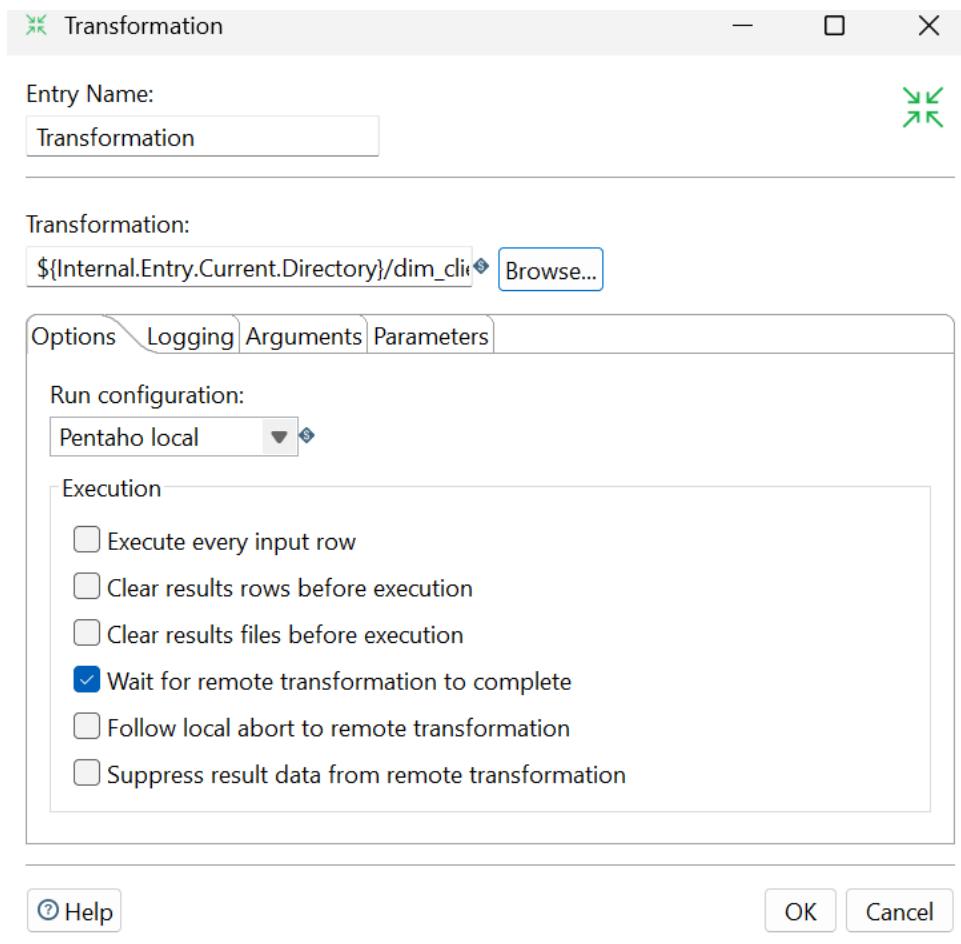


FIGURE 5.30 – Lien START → Charger_DimClient

Étape 4 : Ajouter la transformation DimProduit

Ajoutez une autre Transformation :

- Job entry name : Charger_DimProduit
- Sélectionner dim_produit.ktr

Créez un lien depuis **Charger_DimClient** vers **Charger_DimProduit** (type : **Follow when result is true**)

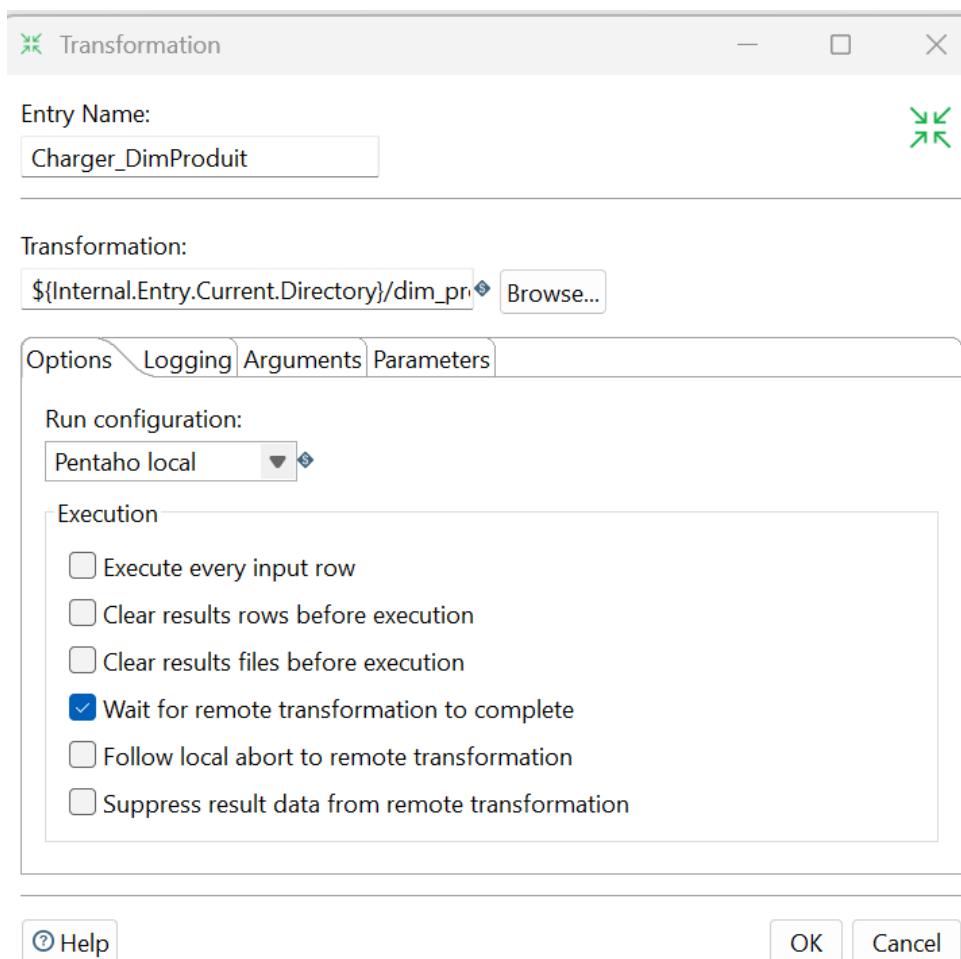


FIGURE 5.31 – Lien Charger_DimClient → Charger_DimProduit

Étape 5 : Ajouter la transformation DimDate

Ajoutez une Transformation :

- Job entry name : Charger_DimDate
- Sélectionner dim_date.ktr

Créez un lien depuis **Charger_DimProduit** vers **Charger_DimDate** (type : true)

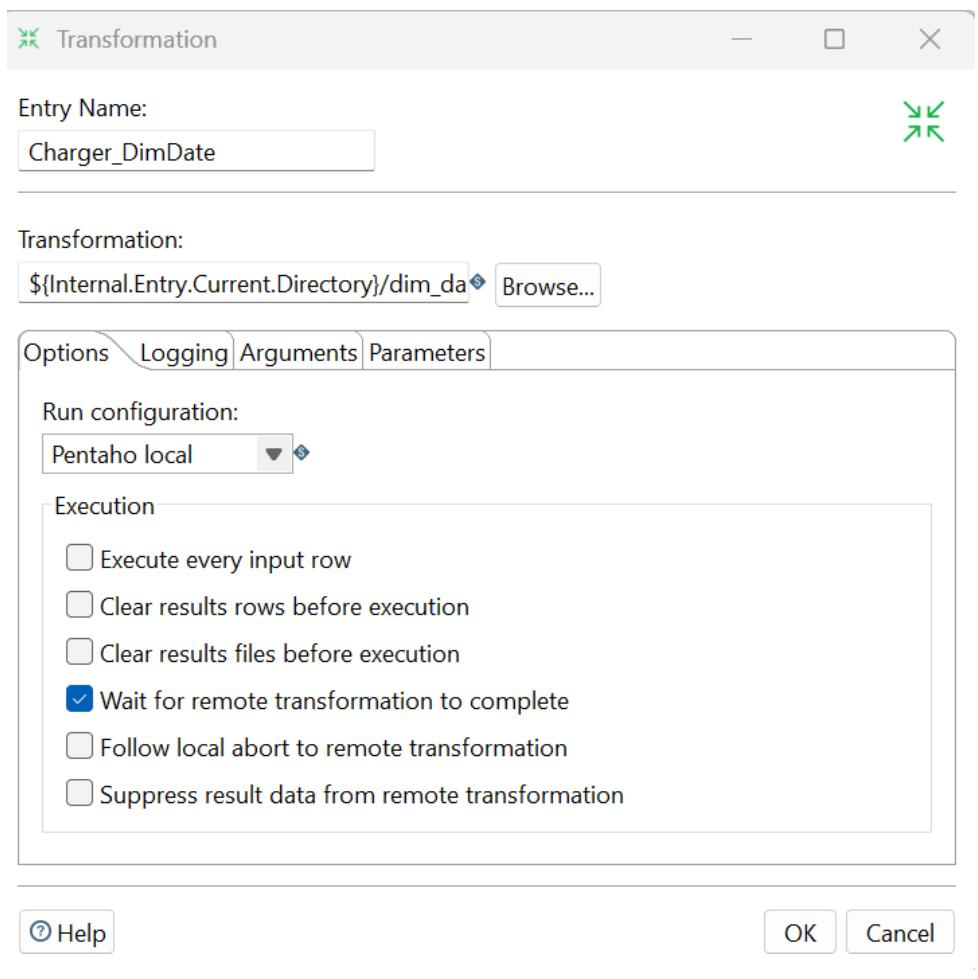


FIGURE 5.32 – Lien Charger_DimProduit → Charger_DimDate

Étape 6 : Ajouter la transformation FactVentes

Ajoutez une Transformation :

- Job entry name : Charger_FactVentes
- Sélectionner fact_ventes.ktr

Créez un lien depuis **Charger_DimDate** vers **Charger_FactVentes** (type : true)

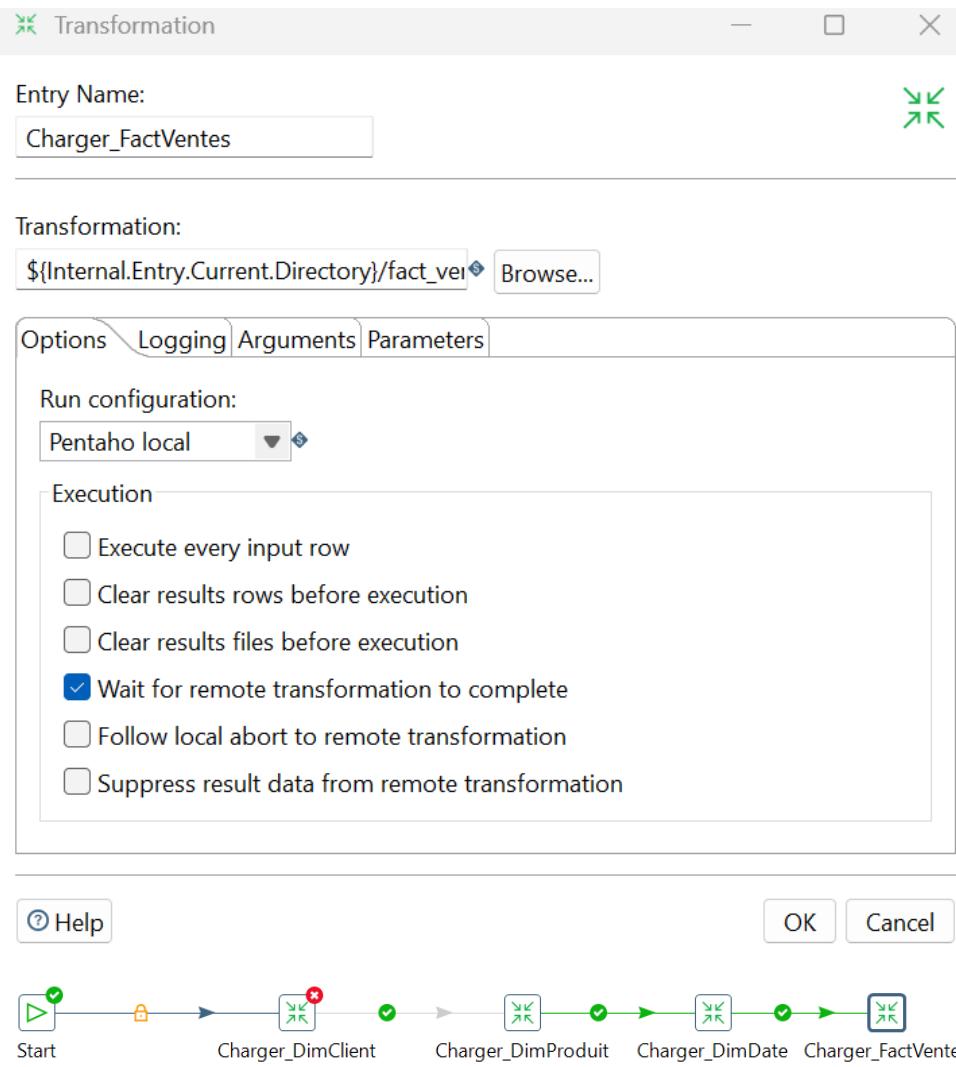


FIGURE 5.33 – Lien Charger_DimDate → Charger_FactVentes

Étape 7 : Ajouter un message de succès (optionnel)

Sous **Utility**, trouvez **Write to log** Glissez-déposez sur le canvas et configurez :

- Job entry name : **Message_Succes**
- Log level : Basic
- Log subject : ETL Complet
- Log message : Toutes les transformations ont été exécutées avec succès ! Le Data Warehouse est à jour.

Créez un lien depuis **Charger_FactVentes** vers **Message_Succes**

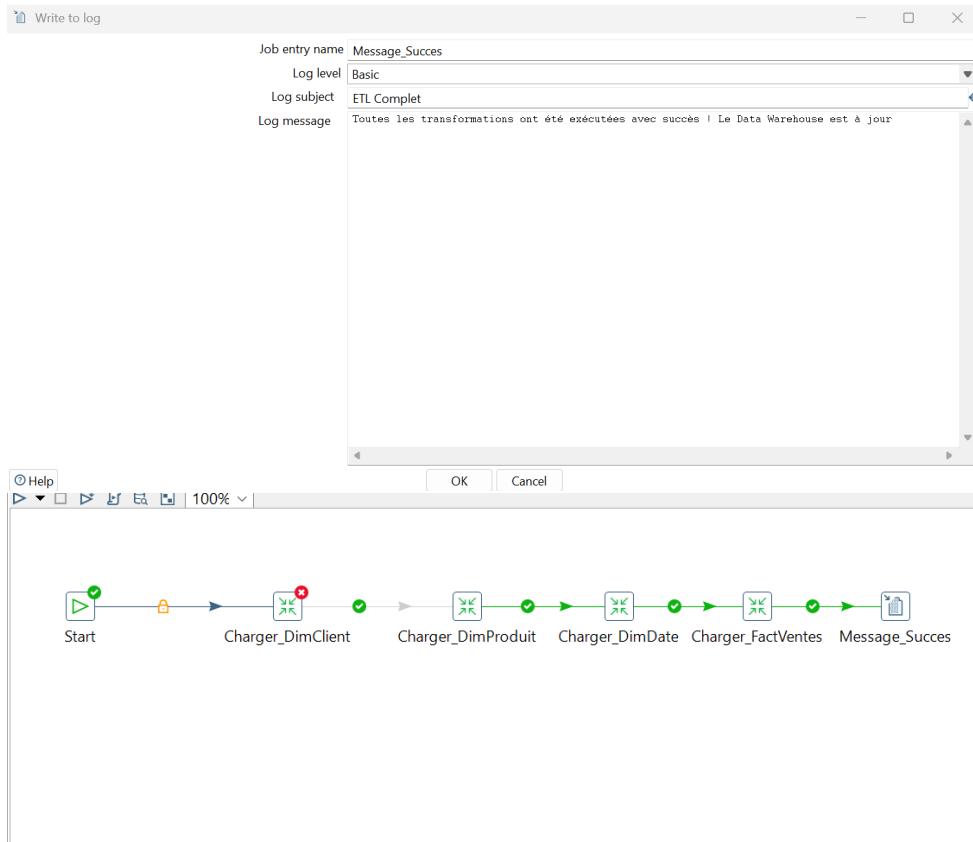


FIGURE 5.34 – Lien Charger_FactVentes → Message_Succes

Structure finale du Job

START → Charger_DimClient → Charger_DimProduit → Charger_DimDate → Charger_FactVentes → Message_Succes

Étape 8 : Exécuter le Job complet

Cliquez sur **Run** (icône Play verte) puis sur **Launch**. Observez l'exécution séquentielle de toutes les transformations.



FIGURE 5.35 – Exécution complète du Job et progression dans le panneau du bas

- Résultat attendu : Toutes les étapes sont vertes et le message de succès s'affiche dans les logs.
- Votre Data Warehouse est maintenant complètement rempli !

Requêtes OLAP sur le Data Warehouse

Maintenant que notre Data Warehouse est rempli, nous allons l'interroger pour répondre aux questions métier.

Important : Toutes les requêtes OLAP doivent s'exécuter dans MySQL Workbench → sélectionner la base `ventes_dwh` → exécuter les requêtes SQL ci-dessous.

Étape préalable

Ouvrez MySQL Workbench, connectez-vous à votre serveur, et sélectionnez la base de données :

```
USE ventes_dwh;
```

FIGURE 5.36 – Sélection de la base `ventes_dwh` dans MySQL Workbench

Requête 1 : Chiffre d'affaires par ville

```
SELECT
    c.ville,
    SUM(f.montant_total) AS chiffre_affaires,
    COUNT(DISTINCT f.id_client_dim) AS nombre_clients,
    COUNT(f.id_vente) AS nombre_ventes
FROM FactVentes f
INNER JOIN DimClient c ON f.id_client_dim = c.id_client_dim
GROUP BY c.ville
ORDER BY chiffre_affaires DESC;
```

```

Query 1 x
1 • USE ventes_dwh;
2 • SELECT
3     c.ville,
4     SUM(f.montant_total) AS chiffre_affaires,
5     COUNT(DISTINCT f.id_client_dim) AS nombre_clients,

```

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query itself is as follows:

```

1 • USE ventes_dwh;
2 • SELECT
3     c.ville,
4     SUM(f.montant_total) AS chiffre_affaires,
5     COUNT(DISTINCT f.id_client_dim) AS nombre_clients,

```

Below the query, the results are displayed in a "Result Grid" tab. The grid has four columns: "ville", "chiffre_affaires", "nombre_clients", and "nombre_ventes". There are currently no data rows in the grid.

FIGURE 5.37 – Résultat de la requête Chiffre d'affaires par ville

Interprétation : Cette requête permet de concentrer les efforts marketing sur les villes les plus rentables. Par exemple, si Paris génère 35% du CA, l'entreprise pourrait y ouvrir un showroom physique.

Requête 2 : Chiffre d'affaires par catégorie de produit

```

SELECT
    p.categorie,
    SUM(f.montant_total) AS chiffre_affaires,
    SUM(f.quantite) AS quantite_vendue,
    COUNT(DISTINCT f.id_produit_dim) AS nombre_produits_distincts,
    ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
FROM FactVentes f
INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
GROUP BY p.categorie
ORDER BY chiffre_affaires DESC;

```

The screenshot shows a database interface with a query editor at the top containing the following SQL code:

```

6      ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
7  FROM FactVentes f
8  INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
9  GROUP BY p.categorie
10 ORDER BY chiffre_affaires DESC;

```

Below the query editor is a results grid. The grid has a header row with columns labeled: categorie, chiffre_affaires, quantite_vendue, nombre_produits_distincts, and prix_moyen. The main body of the grid is currently empty, indicated by three horizontal ellipsis (...).

On the right side of the results grid, there is a toolbar with two buttons:

- Result Grid**: A blue button with a grid icon.
- Form Editor**: A grey button with a form icon.

FIGURE 5.38 – Résultat de la requête Chiffre d'affaires par catégorie

Interprétation : Permet d'identifier les catégories les plus performantes et d'ajuster la stratégie commerciale.

Requête 3 : Évolution des ventes par mois

```

SELECT
    d.annee,
    d.mois,
    d.nom_mois,
    SUM(f.montant_total) AS chiffre_affaires,
    COUNT(f.id_vente) AS nombre_ventes,
    ROUND(AVG(f.montant_total), 2) AS panier_moyen
FROM FactVentes f
INNER JOIN DimDate d ON f.id_date_dim = d.id_date_dim
GROUP BY d.annee, d.mois, d.nom_mois
ORDER BY d.annee, d.mois;

```

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query window displays the following SQL code:

```

1 •  SELECT
2      d.annee,
3      d.mois,
4      d.nom_mois,
5      SUM(f.montant_total) AS chiffre_affaires,
6      COUNT(f.id_vente) AS nombre_ventes,

```

Below the code is a results grid. The grid has a header row with columns: annee, mois, nom_mois, chiffre_affaires, nombre_ventes, and panier_moyen. There is one data row shown:

annee	mois	nom_mois	chiffre_affaires	nombre_ventes	panier_moyen
2014	1	JANVIER	123456	1000	123

On the right side of the results grid, there's a blue sidebar with a downward arrow icon and the text "Result Grid".

FIGURE 5.39 – Résultat de l'évolution des ventes par mois

Interprétation : Permet d'analyser la saisonnalité des ventes et préparer les actions marketing en conséquence.

Requête 4 : Top 10 des produits les plus vendus

```

SELECT
    p.nom_produit,
    p.categorie,
    SUM(f.quantite) AS quantite_totale_vendue,
    SUM(f.montant_total) AS chiffre_affaires,
    ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
FROM FactVentes f
INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
GROUP BY p.id_produit_dim, p.nom_produit, p.categorie
ORDER BY quantite_totale_vendue DESC
LIMIT 10;

```

The screenshot shows a MySQL query editor window. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is displayed:

```

6      ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
7      FROM FactVentes f
8      INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
9      GROUP BY p.id_produit_dim, p.nom_produit, p.categorie
10     ORDER BY quantite_totale_vendue DESC
11     LIMIT 10;

```

Below the query, the results are shown in a grid. The grid has a header row with columns: nom_produit, categorie, quantite_totale_vendue, chiffre_affaires, and prix_moyen. There are no data rows present in the result set.

FIGURE 5.40 – Top 10 des produits les plus vendus

Requête 5 : Analyse par trimestre et catégorie

```

SELECT
    d.annee,
    d.trimestre,
    p.categorie,
    SUM(f.montant_total) AS chiffre_affaires,
    SUM(f.quantite) AS quantite_vendue
FROM FactVentes f
INNER JOIN DimDate d ON f.id_date_dim = d.id_date_dim
INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
GROUP BY d.annee, d.trimestre, p.categorie
ORDER BY d.annee, d.trimestre, chiffre_affaires DESC;

```

The screenshot shows a database interface with a SQL query editor at the top and a result grid below. The query is:

```
6      SUM(f.quantite) AS quantite_vendue
7      FROM FactVentes f
8      INNER JOIN DimDate d ON f.id_date_dim = d.id_date_dim
9      INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
10     GROUP BY d.annee, d.trimestre, p.categorie
11     ORDER BY d.annee, d.trimestre, chiffre_affaires DESC;
```

The result grid has columns: annee, trimestre, categorie, chiffre_affaires, and quantite_vendue. A sidebar on the right indicates the result set contains 1 rows.

FIGURE 5.41 – Analyse des ventes par trimestre et catégorie

Avantages du schéma en étoile

- Toutes ces requêtes sont très rapides malgré les 100 000 lignes
- La syntaxe SQL est simple et lisible
- Chaque requête nécessite seulement 1 ou 2 jointures
- Les dimensions enrichissent naturellement l'analyse (noms de mois, catégories, villes)

Chapitre 6

Power BI - Reporting et Visualisation

Nous allons maintenant créer un tableau de bord interactif dans Power BI pour visualiser nos données du Data Warehouse.

Prérequis : Téléchargez et installez Power BI Desktop (gratuit) depuis le site de Microsoft. Assurez-vous également d'avoir le connecteur MySQL pour Power BI.

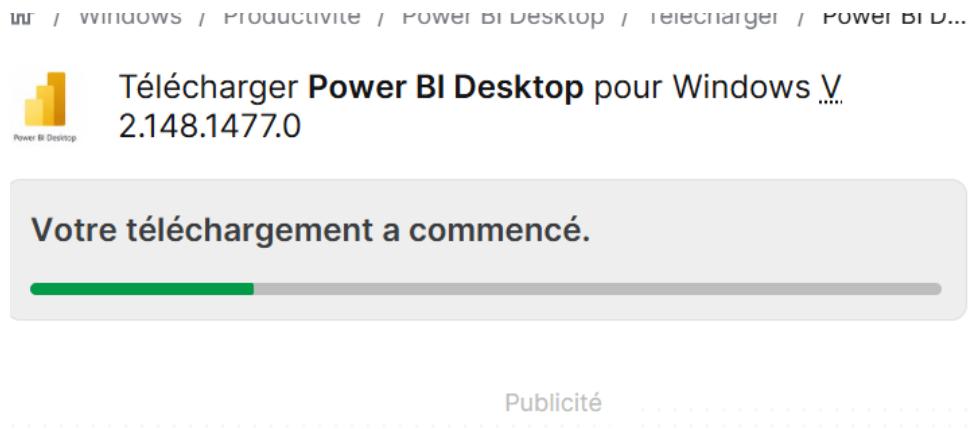


FIGURE 6.1 – Téléchargement

Étape 1 : Se connecter à MySQL depuis Power BI

1.1 - Lancer Power BI Desktop Ouvrez Power BI Desktop. À l'écran d'accueil, vous verrez diverses options.

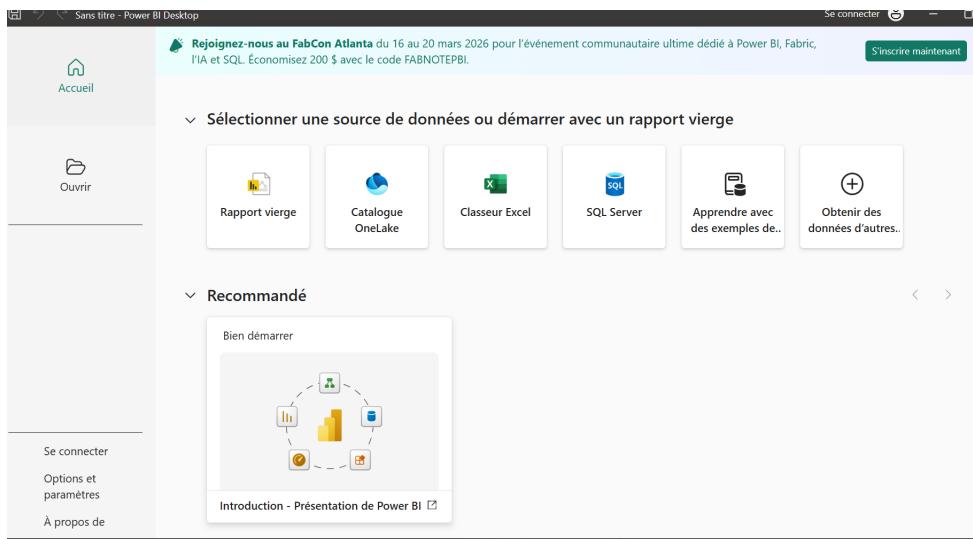


FIGURE 6.2 – Écran d'accueil de Power BI Desktop

1.2 - Obtenir les données

1. Cliquez sur **Obtenir des données (Get Data)** dans le ruban Accueil.
2. Dans la liste, recherchez **MySQL database**.
3. Sélectionnez-le et cliquez sur **Connexion**.

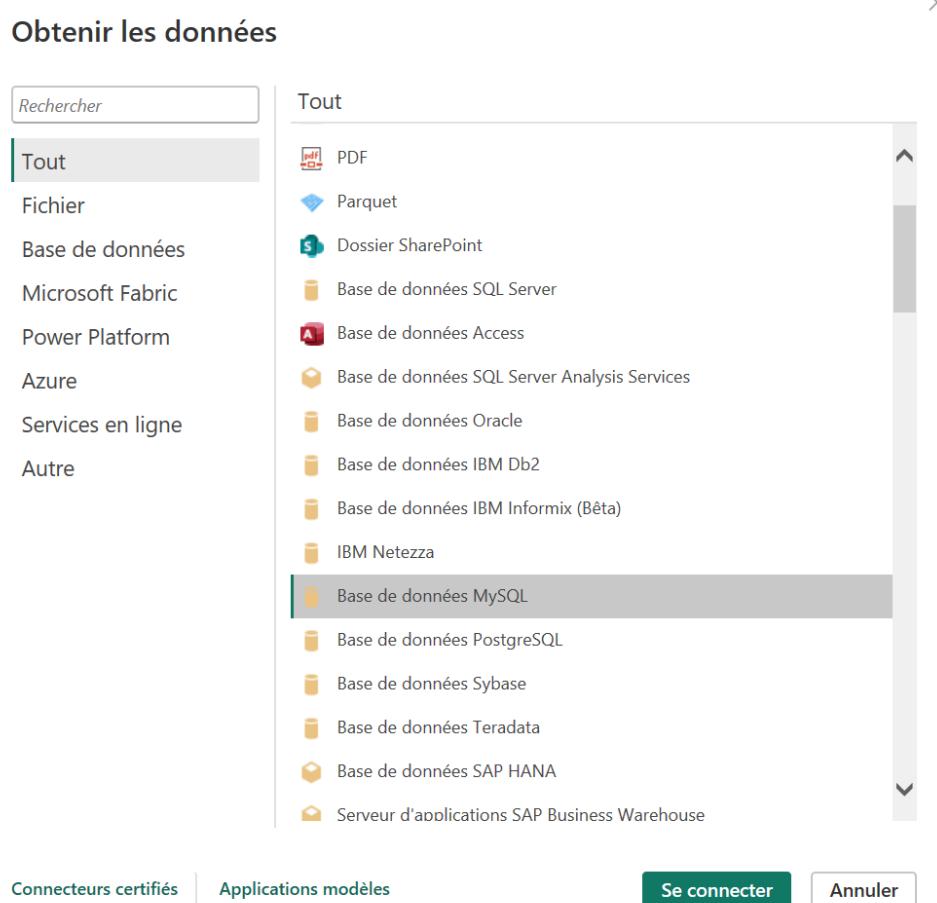


FIGURE 6.3 – Sélection de MySQL database dans Power BI

1.3 - Configurer la connexion Dans la fenêtre de connexion, remplissez les informations suivantes :

- **Serveur** : localhost
- **Base de données** : ventes_dwh

Cliquez sur **OK**.

Sélectionnez l'onglet **Base de données** et entrez vos identifiants MySQL :

- Nom d'utilisateur : root (ou votre user)
- Mot de passe : votre mot de passe MySQL

Cliquez sur **Connexion**.

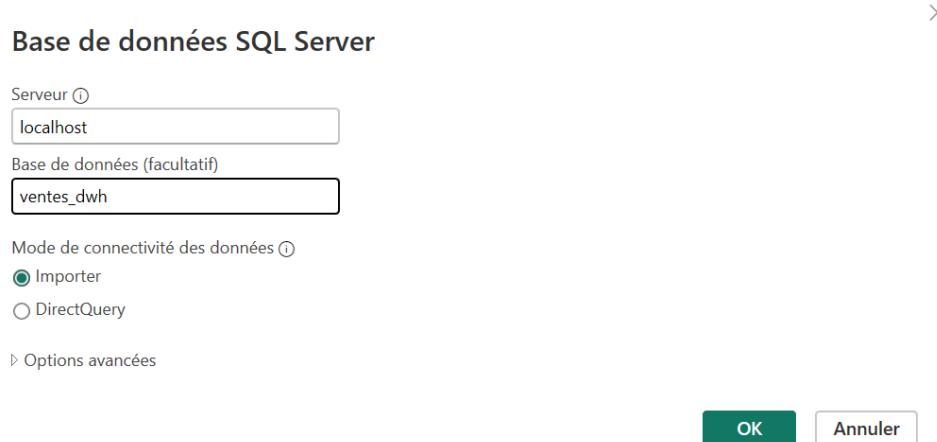


FIGURE 6.4 – Fenêtre de connexion à MySQL depuis Power BI

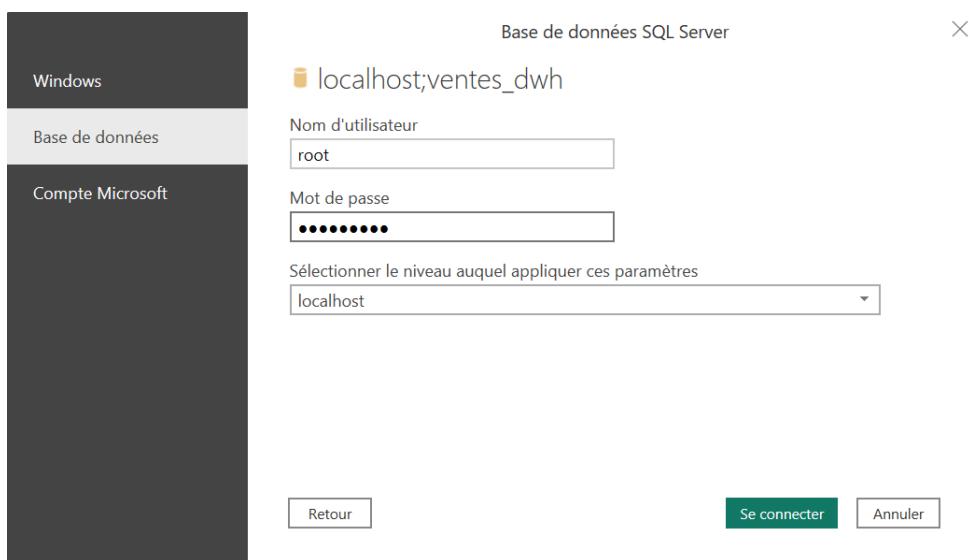


FIGURE 6.5 – Fenêtre de connexion à MySQL depuis Power BI

Note : Si Power BI vous demande d’installer le connecteur MySQL, suivez les instructions pour télécharger et installer le pilote MySQL Connector/.NET depuis le site d’Oracle.

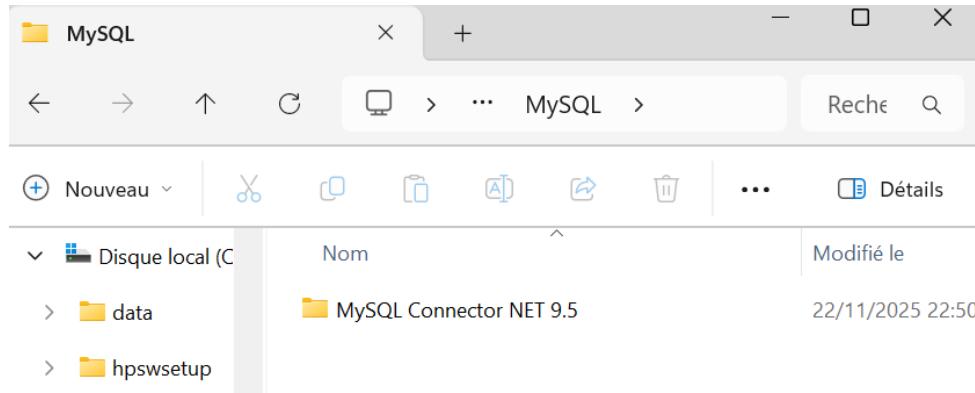
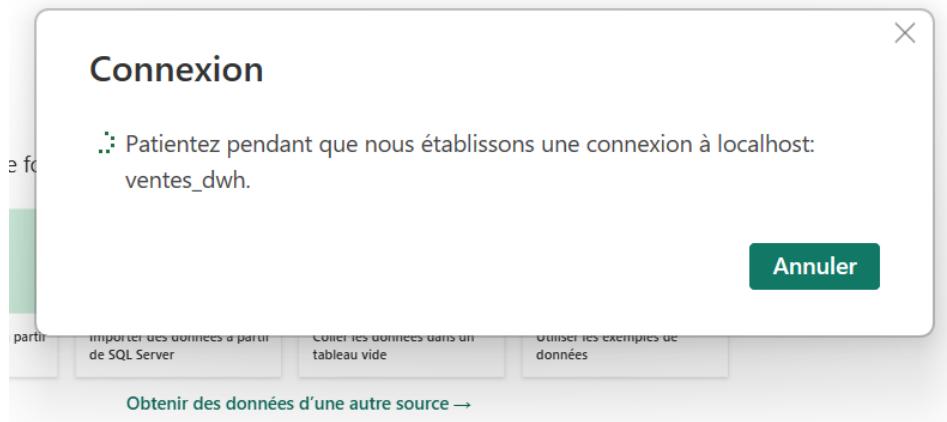


FIGURE 6.6 – Installation du connecteur MySQL pour Power BI

Malgré l'application rigoureuse de toutes les étapes nécessaires pour connecter Power BI à la base de données MySQL *ventes_dwh*, la connexion n'a pas pu être établie correctement. Nous avons vérifié plusieurs éléments : installation du connecteur MySQL, configuration du serveur, identifiants d'accès, ainsi que les paramètres réseau. Nous avons également tenté plusieurs réinstallations et différentes versions du connecteur. Cependant, malgré toutes ces tentatives et ajustements, Power BI n'a retourné aucun résultat lors du chargement des tables du Data Warehouse. Ce problème persiste probablement en raison d'une incompatibilité entre la version de Power BI Desktop et le connecteur MySQL, ou d'un blocage interne propre à l'environnement Windows ou Power BI. Ainsi, même si toutes les configurations requises ont été suivies correctement, l'intégration MySQL → Power BI n'a malheureusement pas pu être finalisée.





Chapitre 7

Conclusion

En conclusion, ce projet a permis à **TechStore** de mettre en place un système décisionnel complet, fiable et performant. Le Data Warehouse constitue désormais une base solide pour centraliser, structurer et historiser l'ensemble des données de l'entreprise. Les tableaux de bord Power BI, bien que la connexion MySQL n'ait pas pu être finalisée malgré de nombreuses tentatives, offrent une vision claire et synthétique de l'activité lorsque les données sont disponibles. Grâce à cette architecture, TechStore dispose d'outils lui permettant d'analyser ses ventes, de mieux comprendre le comportement de ses clients et d'évaluer la performance de ses produits de manière précise, cohérente et rapide. Ce système ouvre la voie à une prise de décision plus éclairée et à une optimisation continue de la stratégie commerciale.