

TP: Pipeline OLTP → ETL Pentaho PDI → Data Warehouse → Reporting Power BI

Nom Complet: Adil HABIB
Date: 2025-2026
Formation: Big Data

Résumé exécutif :

Ce projet consiste à concevoir une solution décisionnelle complète pour TechStore, de la génération des données jusqu'à la visualisation finale.

Il inclut la modélisation OLTP, la création d'un Data Warehouse, le développement d'un flux ETL avec Pentaho, l'exécution de requêtes OLAP et la réalisation d'un tableau de bord Power BI.

Les résultats principaux :

- Un DWH en schéma en étoile
- Des dimensions clients, produits, dates, et une table de faits ventes
- Un ETL complet et automatisé
- Des analyses OLAP pertinentes
- Un tableau de bord interactif pour l'aide à la décision

Introduction :

TechStore est une entreprise spécialisée dans la vente de produits électroniques tels que les ordinateurs, smartphones et accessoires. Avec l'augmentation du volume de données générées par ses activités, l'entreprise souhaite mieux exploiter ses informations pour améliorer ses performances commerciales.

Les données actuelles, stockées dans une base OLTP, ne sont pas adaptées à des analyses avancées. Pour répondre à ce besoin, TechStore a décidé de mettre en place une solution décisionnelle complète incluant un **Data Warehouse**, un processus **ETL** automatisé et des outils d'analyse et de visualisation.

L'objectif du projet est de transformer les données brutes en indicateurs fiables permettant d'orienter les décisions stratégiques.

Objectifs BI principaux :

- Identifier les villes les plus rentables
- Analyser les catégories de produits les plus performantes
- Étudier la saisonnalité des ventes
- Automatiser l'alimentation du Data Warehouse
- Construire un tableau de bord Power BI pour le pilotage commercial

Cette démarche permet à TechStore d'obtenir une vision plus claire de son activité et d'améliorer son efficacité décisionnelle.

Architecture technique

L'architecture technique mise en place pour ce projet BI suit une structure simple, cohérente et adaptée aux besoins analytiques de TechStore. Elle s'appuie sur une chaîne complète allant de la donnée brute à la visualisation finale.

Voici les différents composants qui interviennent dans le système décisionnel :

◆ **1. Base de données OLTP (MySQL)**

Contient les données opérationnelles de TechStore : clients, produits, ventes.

Ces données sont utilisées comme **source principale** pour l'ETL.

◆ **2. Scripts Python**

Permettent de générer des données synthétiques (clients, produits, ventes) et de produire les fichiers CSV nécessaires pour alimenter la base OLTP.

◆ **3. Pentaho Data Integration (Kettle)**

Outil utilisé pour l'ETL :

- **Extraction** depuis MySQL OLTP
- **Transformation** (nettoyage, renommage, lookup, calculs)
- **Chargement** dans le Data Warehouse

Pentaho assure l'automatisation et la fiabilité des flux de données.

◆ **4. Data Warehouse MySQL**

Entrepôt de données structuré en **schéma en étoile**, contenant :

- DimClient
- DimProduit
- DimDate
- FactVentes

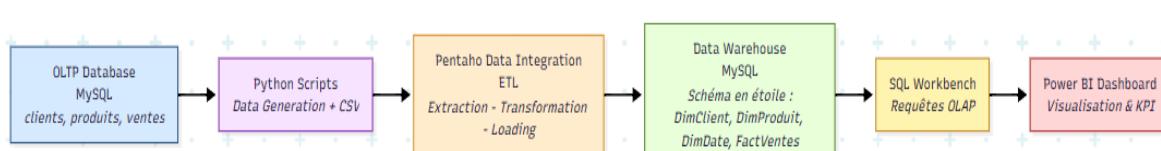
C'est la source principale utilisée pour les analyses décisionnelles.

◆ **5. SQL Workbench / MySQL Workbench**

Utilisé pour effectuer des **requêtes OLAP**, analyser les performances et valider les résultats du DWH.

◆ **6. Power BI**

Permet de créer des visualisations interactives, des tableaux de bord et des indicateurs clés destinés aux décideurs.

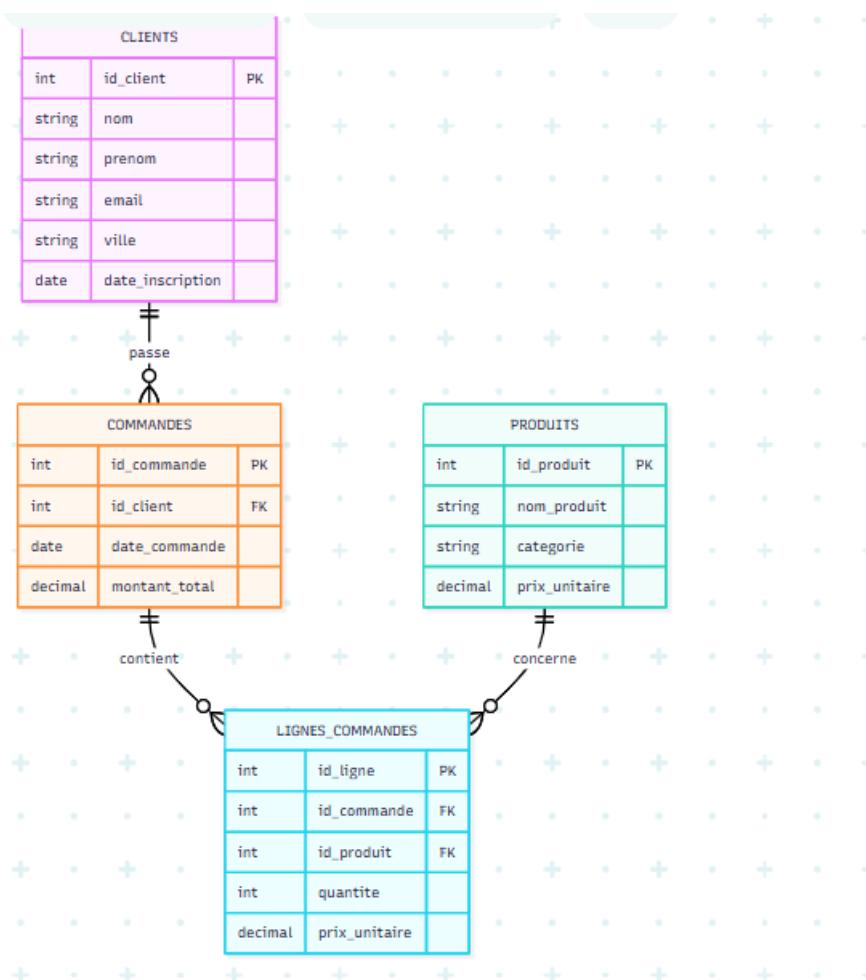


Modélisation OLTP :

La base OLTP suit un modèle 3NF :

- **Clients**
- **Produits**
- **Commandes**
- **lignes_commandes**

Diagramme ERD OLTP



Génération des données :

Les fichiers CSV (clients, produits, ventes) ont été générés via des scripts Python.

Chaque fichier contient plusieurs milliers de lignes avec des distributions cohérentes :

- **10 000 clients**
- **2 000 produits**
- **Plus de 100 000 ventes simulées**

Ces fichiers ont ensuite été importés dans la base OLTP MySQL.

Modélisation du DWH

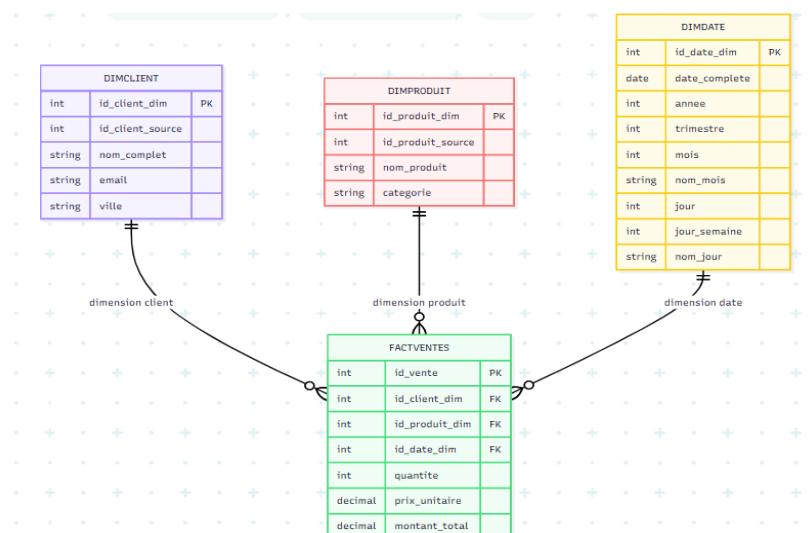
Le DWH repose sur un schéma en étoile composé de quatre tables majeures :

Dimensions

- DimClient
- DimProduit
- DimDate

Table de faits

- FactVentes (quantité, prix, montant...)



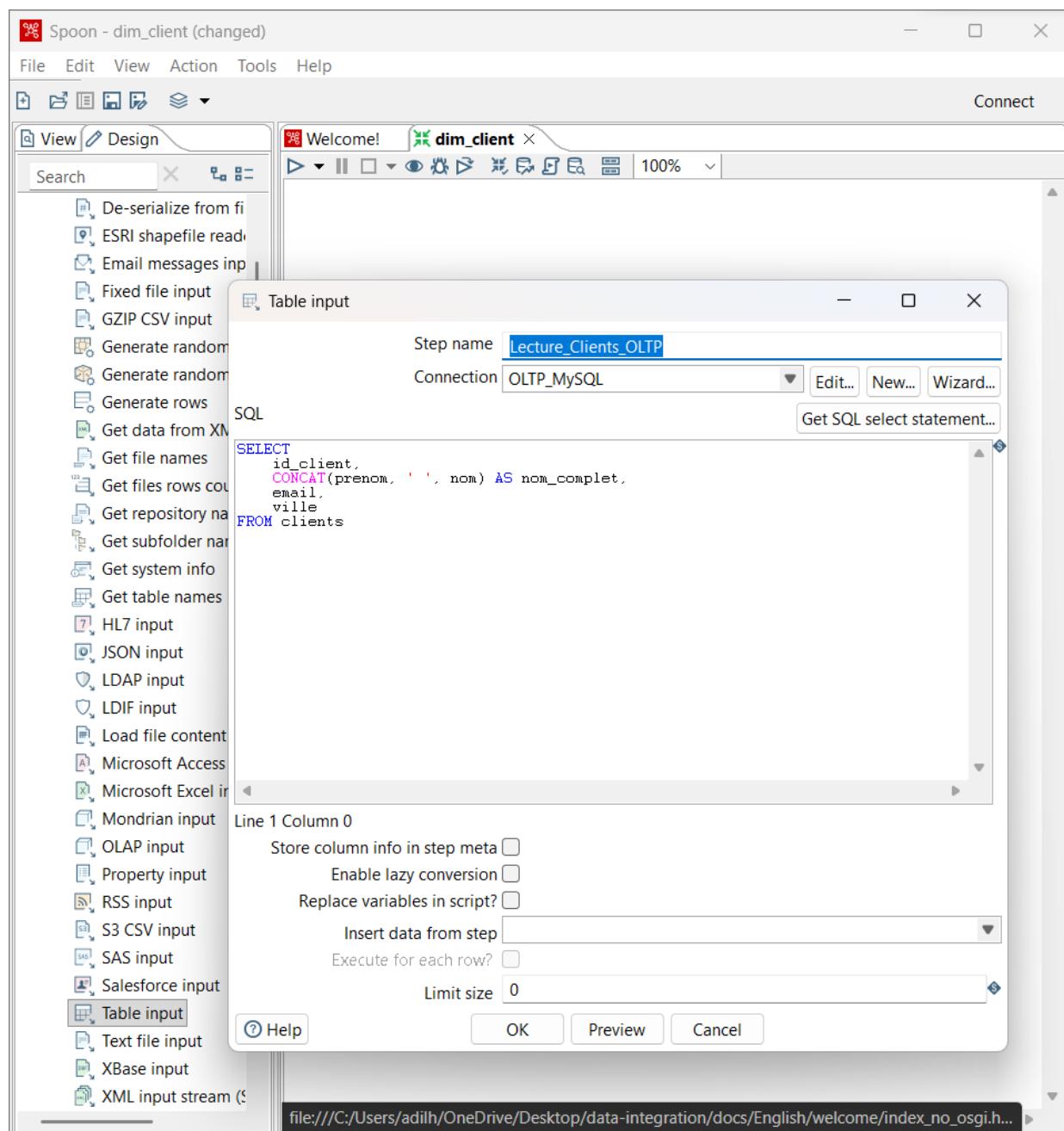
Processus ETL avec Pentaho

Les transformations ETL alimentent progressivement les dimensions puis la table des faits.

1. Dimension Client (DimClient) :

Extraction (Table Input)

Lecture des clients depuis la base OLTP.



Preview des 10 000 lignes

Spoon - dim_client (changed)

File Edit View Action Tools Help

View Design Welcome dim_client Connect

Examine preview data

Rows of step: Lecture_Clients_OLTP (10000 rows)

#	id_client	nom_complet	email	ville
9.	9973	AndrÃ© Pasquier	client9973@dbmail.com	Montpellier
9.	9974	Ã‰mile Lopez	client9974@laposte.net	Lyon
9.	9975	Patrick Jean	client9975@wanadoo.fr	Reims
9.	9976	Georges Lemmonier	client9976@orange.fr	Marseille
9.	9977	Ã‰ric Didier	client9977@france.com	Lyon
9.	9978	Maurice Devaux	client9978@orange.fr	Nantes
9.	9979	Emmanuelle Lamy	client9979@gmail.com	Lyon
9.	9980	Philippe LÃ©vy	client9980@hotmail.fr	Toulouse
9.	9981	Margot Gimenez	client9981@voila.fr	Nice
9.	9982	ThÃ¢tÃ© se Poulain	client9982@tf.fr	Marseille
9.	9983	Roger Albert	client9983@tf.fr	Marseille
9.	9984	CÃ©line Antoine	client9984@voila.fr	Reims
9.	9985	Marcelle Bousquet	client9985@tele2.fr	Bordeaux
9.	9986	Danielle Gaillard	client9986@tele2.fr	Nice
9.	9987	Alex Julie	client9987@live.com	Lille
9.	9988	Margaux Perret	client9988@live.com	Rennes
9.	9989	InÃ©s Grenier	client9989@club-internet.fr	Lille
9.	9990	Zacharie Costa	client9990@dbmail.com	Toulouse
9.	9991	Alexandrie Colas	client9991@live.com	Lille
9.	9992	Tristan Le Goff	client9992@orange.fr	Paris
9.	9993	AmÃ©lie Lombard	client9993@laposte.net	Bordeaux
9.	9994	Louise Lacombe	client9994@hotmail.fr	Lille
9.	9995	ThÃ¢ophile NoÃ«l	client9995@wanadoo.fr	Marseille
9.	9996	Denis Renault	client9996@orange.fr	Strasbourg
9.	9997	Isaac Barre	client9997@dbmail.com	Lyon

Close Show Log

Text file input XBase input XML input stream (STAX) YAML input file:///C:/Users/adith/Desktop/data-integration/docs/English/welcome/index_no.osql.h...

Transformation (Select Values – Renommage)

Select values

Step name Renommer_Colonne

Select & Alter Remove Meta-data

Fields :

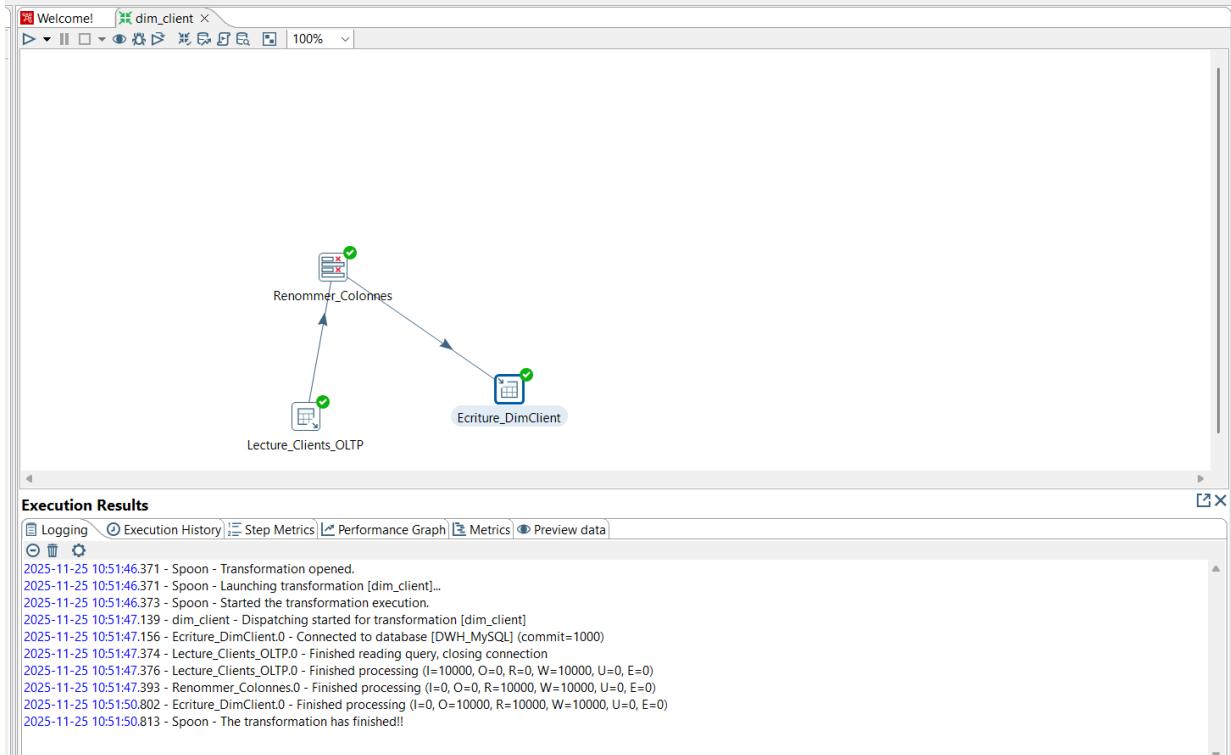
#	Fieldname	Rename to	Length	Precision
1	id_client	id_client_source		
2	nom_complet	nom_complet		
3	email	email		
4	ville	ville		

Get fields to select
Edit Mapping

Include unspecified fields, ordered

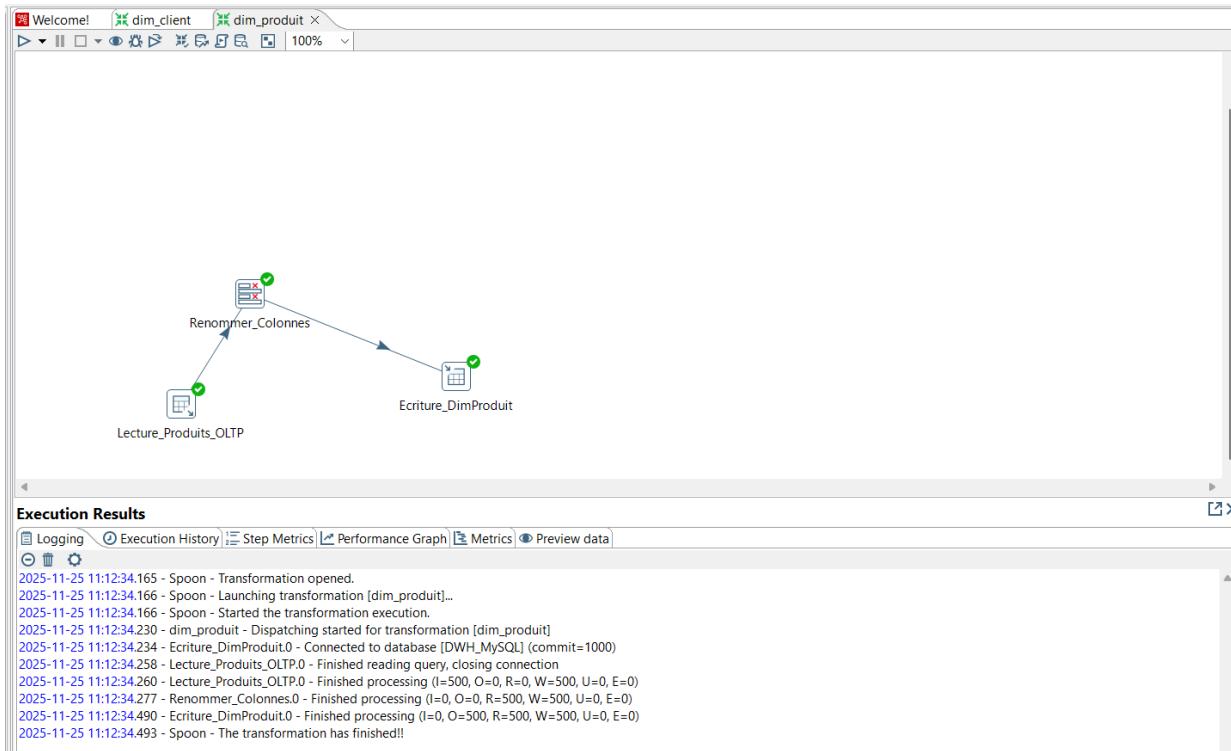
Help OK Cancel

Exécution finale :



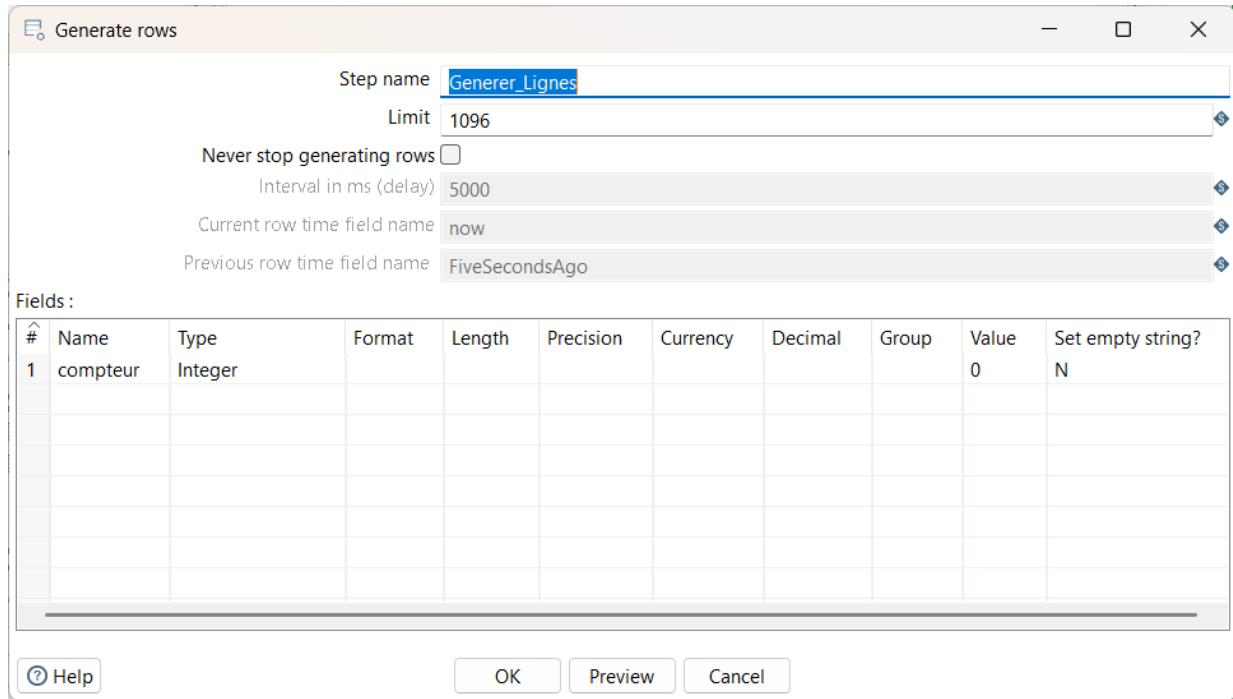
2. Dimension Produit (DimProduit) :

Exécution de la transformation :

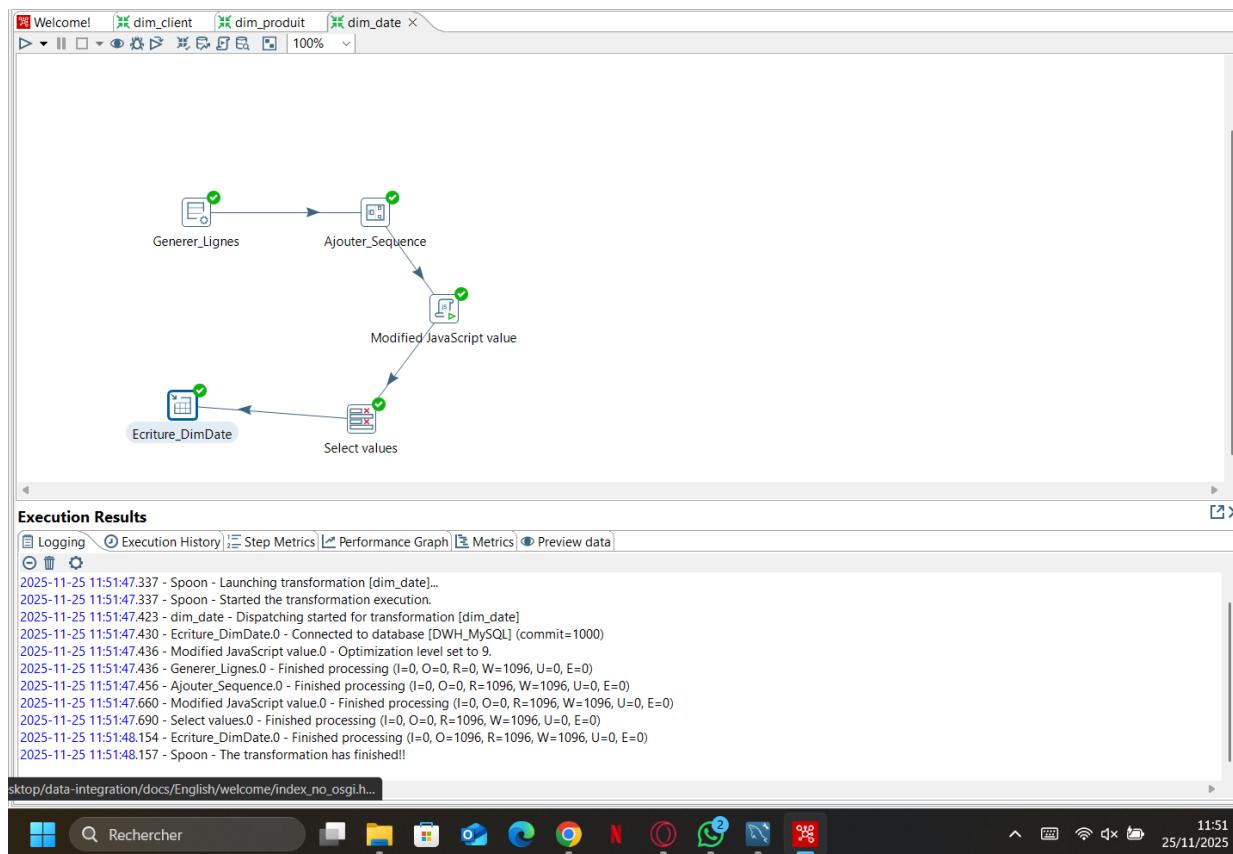


3. Dimension Date (DimDate) :

Génération des lignes (Generate Rows)

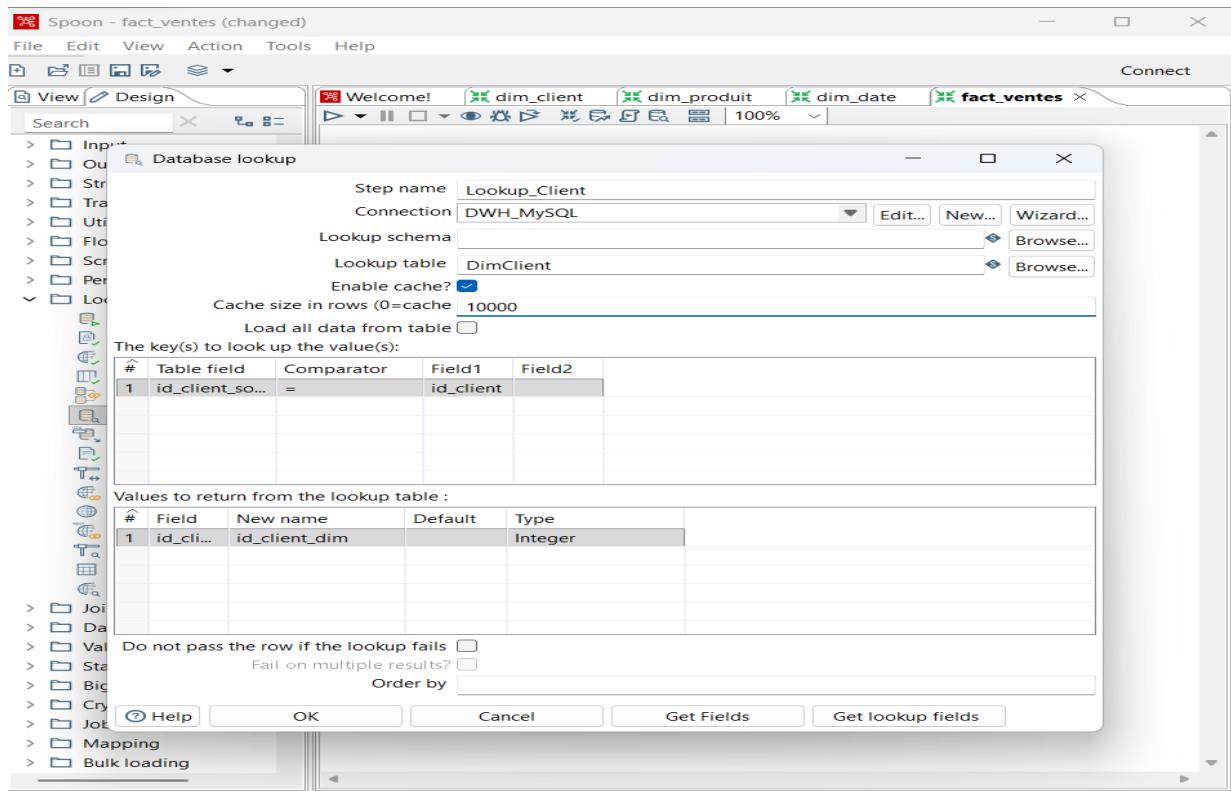


Transformation via Modified JavaScript Value

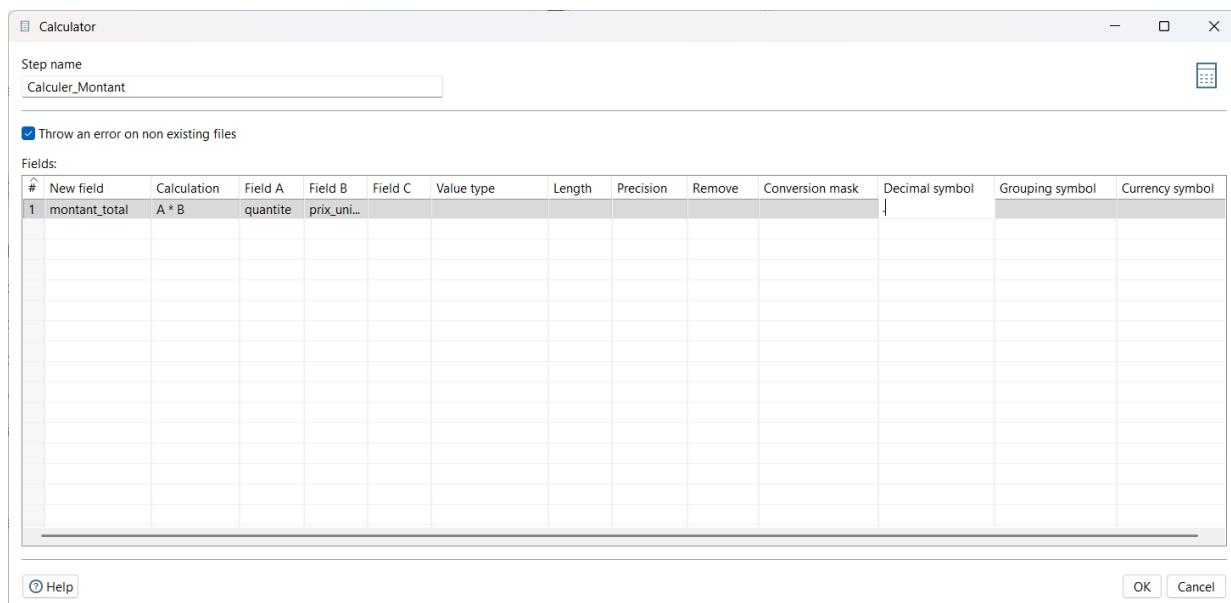


4. Table des Faits : FactVentes :

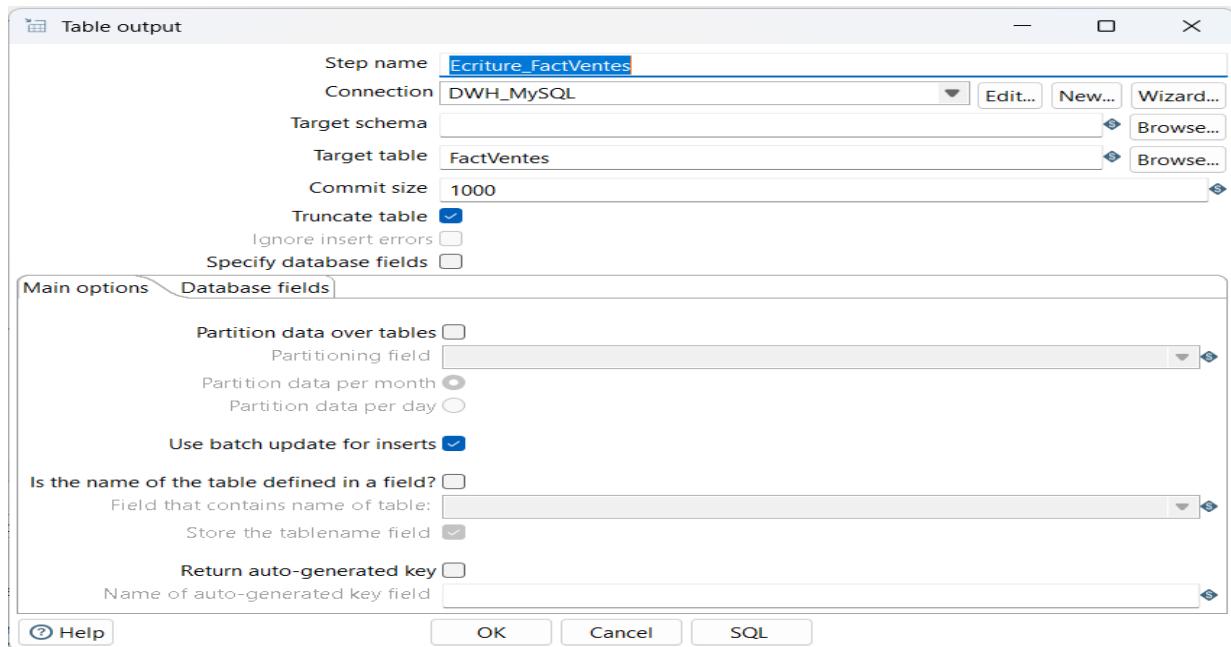
Lookup des dimensions (Database Lookup)



Calcul des mesures (Calculator)



Chargement Table Output

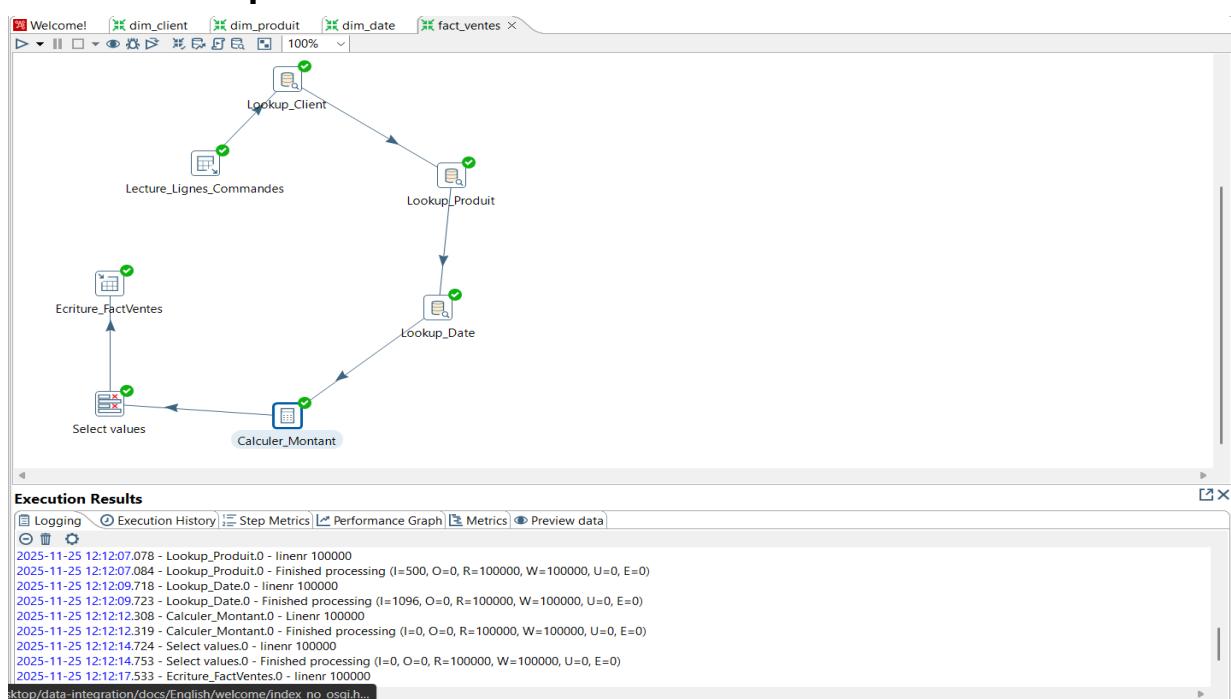


Vérification SQL – COUNT (*)

```
83 • SELECT COUNT(*) FROM ventes_dwh.FactVentes
```

COUNT(*)
100000

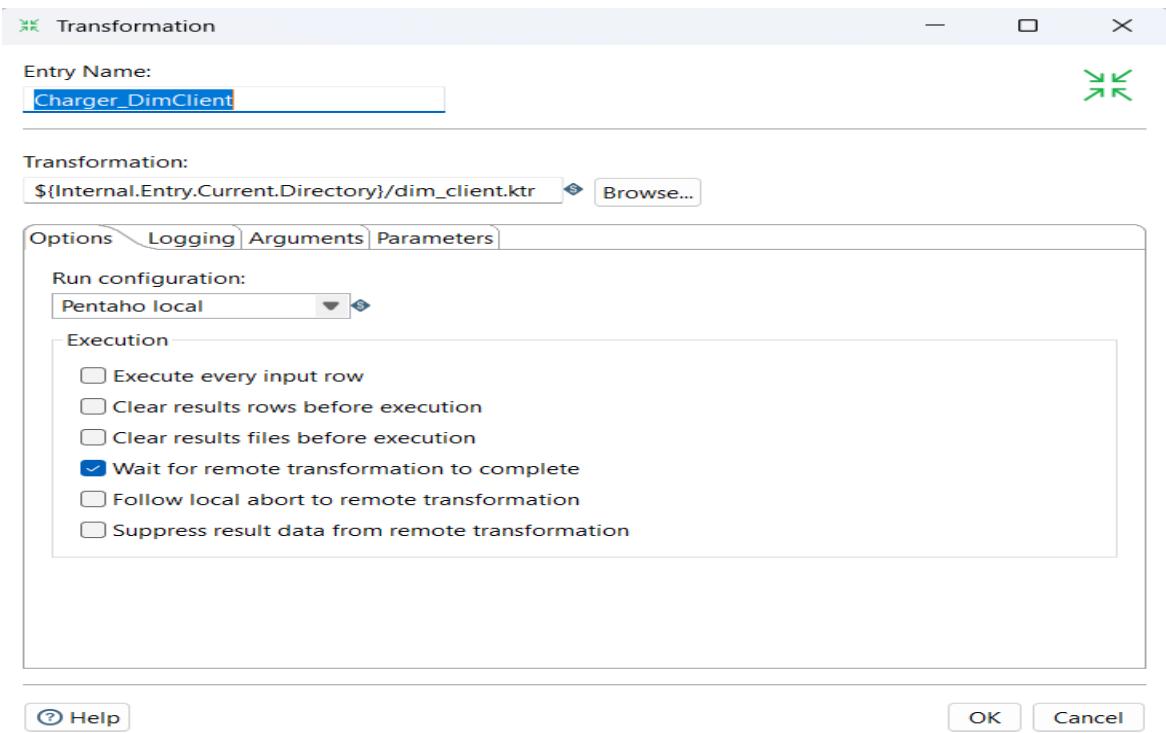
Exécution complète de la transformation



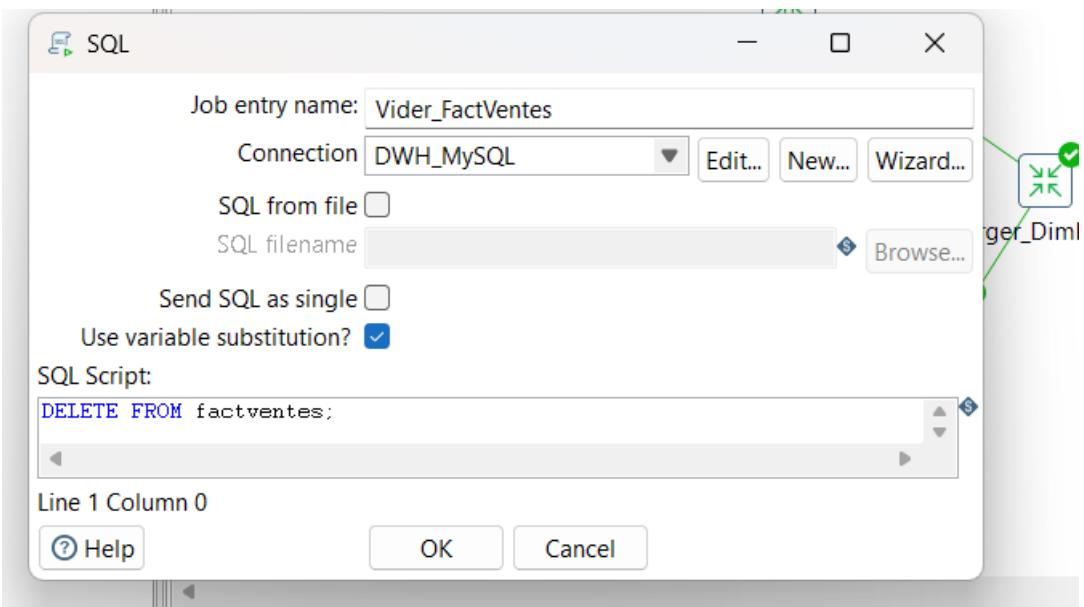
5. Job ETL global

Un Job Pentaho permet d'enchaîner toutes les transformations :
DimClient → DimProduit → DimDate → FactVentes.

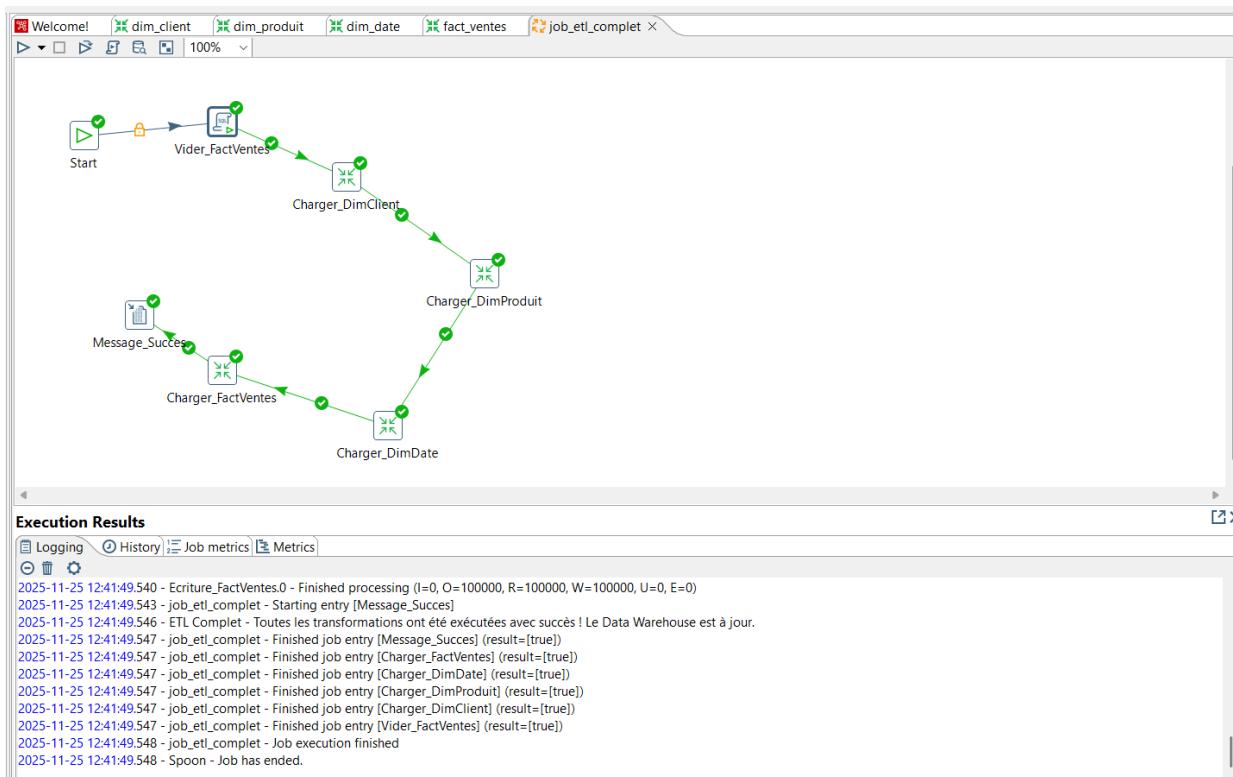
Configuration du job



SQL script ajouté pour résoudre l'erreur (DELETE FROM factventes)



Exécution du Job complet



Analyses OLAP:

Requête 1 : CA par ville

The screenshot shows a database interface with a query editor and a results grid. The query retrieves sales data by city, including total sales, number of clients, and number of sales. The results grid displays data for 13 cities, with Montpellier being the top performer. The output panel shows the execution log.

```
1 • USE ventes_dwh;
2
3 • SELECT
4     c.ville,
5     SUM(f.montant_total) AS chiffre_affaires,
6     COUNT(DISTINCT f.id_client_dim) AS nombre_clients,
7     COUNT(f.id_vente) AS nombre_ventes
8 FROM FactVentes f
9 INNER JOIN DimClient c ON f.id_client_dim = c.id_client_dim
```

ville	chiffre_affaires	nombre_clients	nombre_ventes
Montpellier	2757518.00	748	9203
Rennes	26809551.00	782	8855
Bordeaux	26202469.00	732	8530
Nice	25648124.00	729	8523
Marseille	25194253.00	729	8379
Strasbourg	25116260.00	711	8355
Lille	24826726.00	729	8398
Nantes	24617534.00	718	8094
Lyon	24486553.00	713	8227
Reims	24459394.00	733	8182

```
Result 1 ×
Output
Action Output
# Time Action Message Duration / Fetch
1 13:06:33 USE ventes_dwh 0 row(s) affected 0.000 sec
2 13:06:49 SELECT c.ville, SUM(f.montant_total) AS chiffre_affaires, COUNT(DISTINCT f.id_client_dim) AS nombre... 12 row(s) returned 0.313 sec / 0.000 sec
```

Interprétation : permet d'identifier les villes les plus rentables.

Requête 2 : CA par catégorie produit

The screenshot shows a database interface with a query editor and a results grid. The query retrieves sales data by product category, including total sales, quantity sold, number of distinct products, and average price. The results grid displays data for 6 categories, with Ordinateurs being the top performer. The output panel shows the execution log.

```
14     p.categorie,
15     SUM(f.montant_total) AS chiffre_affaires,
16     SUM(f.quantite) AS quantite_vendue,
17     COUNT(DISTINCT f.id_produit_dim) AS nombre_produits_distincts,
18     ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
19 FROM FactVentes f
20 INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
21 GROUP BY p.categorie
22 ORDER BY chiffre_affaires DESC;
```

categorie	chiffre_affaires	quantite_vendue	nombre_produits_distincts	prix_moyen
Ordinateurs	64825326.00	60514	100	1070.22
T@t@phones	64526264.00	60122	100	1072.42
Accessoires	60129964.00	60127	100	1000.07
Tablettes	56774487.00	59296	100	953.92
Montres	55100653.00	59737	100	919.41

```
Result 2 ×
Output
Action Output
# Time Action Message Duration / Fetch
1 13:06:33 USE ventes_dwh 0 row(s) affected 0.000 sec
2 13:06:49 SELECT c.ville, SUM(f.montant_total) AS chiffre_affaires, COUNT(DISTINCT f.id_client_dim) AS nombre... 12 row(s) returned 0.313 sec / 0.000 sec
3 13:07:31 SELECT p.categorie, SUM(f.montant_total) AS chiffre_affaires, SUM(f.quantite) AS quantite_vendue, ... 5 row(s) returned 0.281 sec / 0.000 sec
```

Interprétation : analyse des catégories performantes.

Requête 3 : Évolution mensuelle des ventes

```

28     d.nom_mois,
29     SUM(f.montant_total) AS chiffre_affaires,
30     COUNT(f.id_vente) AS nombre_ventes,
31     ROUND(AVG(f.montant_total), 2) AS panier_moyen
32   FROM FactVentes f
33   INNER JOIN DimDate d ON f.id_date_dim = d.id_date_dim
34   GROUP BY d.annee, d.mois, d.nom_mois
35   ORDER BY d.annee, d.mois;
36

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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

annee	mois	nom_mois	chiffre_affaires	nombre_ventes	panier_moyen
2022	1	Janvier	8897711.00	3003	2952.94
2022	2	Février	814180.00	2655	3066.36
2022	3	Mars	8611136.00	2892	2977.57
2022	4	Avril	8364191.00	2767	3022.84
2022	5	Mai	7941259.00	2705	2935.77
2022	6	Juin	8000701.00	2602	3074.83
2022	7	Juillet	8454878.00	2887	2928.60
2022	8	Août	8361499.00	2740	3051.64
2022	9	Septembre	8268176.00	2717	3043.13
2022	10	Octobre	8026615.00	2688	2986.09

Result Grid | Form Editor | Field Types |

Read Only | Context Help | Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:06:33	USE ventes_dwh	0 row(s) affected	0.000 sec
2	13:06:49	SELECT c.ville, SUM(f.montant_total) AS chiffre_affaires, COUNT(DISTINCT f.id_client_dim) AS nombre_ventes, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	12 row(s) returned	0.313 sec / 0.000 sec
3	13:07:31	SELECT p.categorie, SUM(f.montant_total) AS chiffre_affaires, SUM(f.quantite) AS quantite_vendue, COUNT(f.id_vente) AS nombre_ventes, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	5 row(s) returned	0.281 sec / 0.000 sec
4	13:08:32	SELECT d.annee, d.mois, d.nom_mois, SUM(f.montant_total) AS chiffre_affaires, COUNT(f.id_vente) AS nombre_ventes, SUM(f.quantite) AS quantite_totale_vendue, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	36 row(s) returned	0.359 sec / 0.000 sec

Interprétation : identification de la saisonnalité.

Requête 4 : Top 10 des produits

```

43     SUM(f.quantite) AS quantite_totale_vendue,
44     SUM(f.montant_total) AS chiffre_affaires,
45     ROUND(AVG(f.prix_unitaire), 2) AS prix_moyen
46   FROM FactVentes f
47   INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
48   GROUP BY p.id_produit_dim, p.nom_produit, p.categorie
49   ORDER BY quantite_totale_vendue DESC
50   LIMIT 10;
51

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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

nom_produit	categorie	quantite_totale_vendue	chiffre_affaires	prix_moyen
Apple Watch Plus	Montres	751	134527.00	1776.81
Clavier m@canique Ultra	Accessoires	740	1277240.00	1726.43
Huawei MatePad Lite	Tablettes	734	136524.00	186.37
Xiaomi 13 Plus	Téléphones	725	1046175.00	1442.98
HP Pavilion Plus	Ordinateurs	714	369852.00	518.16
Phone 14 Standard	Téléphones	707	1260581.00	1782.59
Asus VivoBook Ultra	Ordinateurs	704	1265792.00	1797.82
Garmin Forerunner Lite	Montres	704	772992.00	1097.54
Webcam HD Ultra	Accessoires	703	1120582.00	1594.23
OnePlus 11 Standard	Téléphones	702	819234.00	1167.47

Result Grid | Form Editor | Field Types |

Read Only | Context Help | Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:06:33	USE ventes_dwh	0 row(s) affected	0.000 sec
2	13:06:49	SELECT c.ville, SUM(f.montant_total) AS chiffre_affaires, COUNT(DISTINCT f.id_client_dim) AS nombre_ventes, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	12 row(s) returned	0.313 sec / 0.000 sec
3	13:07:31	SELECT p.categorie, SUM(f.montant_total) AS chiffre_affaires, SUM(f.quantite) AS quantite_vendue, COUNT(f.id_vente) AS nombre_ventes, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	5 row(s) returned	0.281 sec / 0.000 sec
4	13:08:32	SELECT d.annee, d.mois, d.nom_mois, SUM(f.montant_total) AS chiffre_affaires, COUNT(f.id_vente) AS nombre_ventes, SUM(f.quantite) AS quantite_totale_vendue, ROUND(AVG(f.prix_unitaire), 2) AS panier_moyen	36 row(s) returned	0.359 sec / 0.000 sec
5	13:09:16	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	10 row(s) returned	0.484 sec / 0.000 sec

Interprétation : repérage des produits stars.

Requête 5 : Analyse croisée trimestre x catégorie

The screenshot shows a database query interface with the following components:

- SQL Editor:** Displays the following SQL query:

```
57     d.trimestre,
58     p.categorie,
59     SUM(f.montant_total) AS chiffre_affaires,
60     SUM(f.quantite) AS quantite_vendue
61 FROM FactVentes f
62 INNER JOIN DimDate d ON f.id_date_dim = d.id_date_dim
63 INNER JOIN DimProduit p ON f.id_produit_dim = p.id_produit_dim
64 GROUP BY d.annee, d.trimestre, p.categorie
65 ORDER BY d.annee, d.trimestre, chiffre_affaires DESC;
```
- Result Grid:** Shows the query results in a tabular format with columns: annee, trimestre, categorie, chiffre_affaires, and quantite_vendue. The data includes various categories like Téléphones, Ordinateurs, Accessoires, Tablets, Montres, etc., across different years and quarters.
- Action Output:** Displays the execution history of the query, showing the time, action, message, and duration for each step.
- Help:** A note in the top right corner states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Interprétation : analyse multidimensionnelle.

Conclusions et recommandations :

La réalisation de ce projet BI a permis de mettre en place une chaîne décisionnelle complète, depuis la génération de données brutes jusqu'à la restitution finale dans Power BI. Cette démarche a mis en évidence l'importance d'une architecture cohérente, de processus ETL fiables et d'une modélisation adaptée aux besoins métier. La construction du Data Warehouse en schéma en étoile s'est révélée essentielle pour améliorer la qualité, la cohérence et la rapidité d'accès aux données, permettant ainsi d'effectuer des analyses détaillées et pertinentes.

L'un des principaux apprentissages de ce travail concerne la structuration des données. Le passage d'un modèle OLTP orienté transactions vers un modèle dimensionnel a montré clairement comment la transformation des données facilite l'analyse et la prise de décision. Le développement du flux ETL dans Pentaho a également constitué un point clé : il a permis de comprendre les enjeux liés à la qualité, à l'automatisation et à la synchronisation des données, tout en illustrant les bonnes pratiques de nettoyage, de lookup des dimensions et de gestion des dates. L'intégration finale dans Power BI a renforcé la compréhension de la data visualisation, de la création d'indicateurs et de l'interprétation des résultats.

Ce projet a aussi mis en lumière plusieurs axes d'amélioration possibles. Sur le plan technique, l'ajout de nouvelles dimensions (magasin, vendeur, canal de vente) permettrait d'enrichir les analyses. L'automatisation complète du processus via un scheduler, ainsi que la mise en place d'une passerelle Power BI, renforcerait la fluidité de la mise à jour des rapports. Une migration vers une solution cloud (Azure, AWS ou GCP) offrirait davantage d'évolutivité, de performance et de sécurité. Enfin, l'intégration de données en quasi-temps réel élargirait les capacités de monitoring opérationnel, notamment pour le suivi des ventes online.

En résumé, ce projet a permis d'acquérir une vision complète du cycle de vie d'un système décisionnel : collecte, transformation, modélisation, stockage et visualisation. Il a également développé des compétences pratiques essentielles dans la manipulation des données, l'architecture BI et la création d'outils d'aide à la décision. Les améliorations proposées ouvrent la voie à une version plus avancée, scalable et automatisée de la solution, capable de répondre aux futurs besoins analytiques de l'entreprise.

Annexes – Scripts complets

1. Scripts OLTP (Base opérationnelle MySQL – ventes_oltp)

1.1 Création de la base OLTP

```
CREATE DATABASE IF NOT EXISTS ventes_oltp;
USE ventes_oltp;
```

1.2 Table Clients

```
CREATE TABLE clients (
    id_client INT PRIMARY KEY,
    nom VARCHAR(100),
    prenom VARCHAR(100),
    email VARCHAR(150),
    ville VARCHAR(100),
    date_inscription DATE
);
```

1.3 Table Produits

```
CREATE TABLE produits (
    id_produit INT PRIMARY KEY,
    nom_produit VARCHAR(150),
    categorie VARCHAR(100),
    prix_unitaire DECIMAL(10,2)
);
```

1.4 Table Commandes

```
CREATE TABLE commandes (
    id_commande INT PRIMARY KEY,
    id_client INT,
    date_commande DATE,
    montant_total DECIMAL(12,2),
    FOREIGN KEY (id_client) REFERENCES clients(id_client)
);
```

1.5 Table Lignes Commandes

```
CREATE TABLE lignes_commandes (
    id_ligne INT PRIMARY KEY,
    id_commande INT,
    id_produit INT,
    quantite INT,
    prix_unitaire DECIMAL(10,2),
    FOREIGN KEY (id_commande) REFERENCES commandes(id_commande),
    FOREIGN KEY (id_produit) REFERENCES produits(id_produit)
);
```

2. Scripts Data Warehouse (DWH – ventes_dwh)

2.1 Création du Data Warehouse

```
CREATE DATABASE IF NOT EXISTS ventes_dwh;
USE ventes_dwh;
```

2.2 Dimension Client

```
CREATE TABLE dimclient (
    id_client_dim INT AUTO_INCREMENT PRIMARY KEY,
    id_client_source INT,
    nom_complet VARCHAR(200),
    email VARCHAR(150),
    ville VARCHAR(100)
);
```

2.3 Dimension Produit

```
CREATE TABLE dimproduit (
    id_produit_dim INT AUTO_INCREMENT PRIMARY KEY,
    id_produit_source INT,
    nom_produit VARCHAR(150),
    categorie VARCHAR(100)
);
```

2.4 Dimension Date

```
CREATE TABLE dimdate (
    id_date_dim INT AUTO_INCREMENT PRIMARY KEY,
    date_complete DATE,
    annee INT,
    trimestre INT,
    mois INT,
    nom_mois VARCHAR(20),
    jour INT,
    jour_semaine INT,
    nom_jour VARCHAR(20)
);
```

2.5 Table des faits – FactVentes

```
CREATE TABLE factventes (
    id_vente INT AUTO_INCREMENT PRIMARY KEY,
    id_client_dim INT,
    id_produit_dim INT,
    id_date_dim INT,
    quantite INT,
    prix_unitaire DECIMAL(10,2),
    montant_total DECIMAL(12,2),

    FOREIGN KEY (id_client_dim) REFERENCES dimclient(id_client_dim),
    FOREIGN KEY (id_produit_dim) REFERENCES dimproduit(id_produit_dim),
    FOREIGN KEY (id_date_dim) REFERENCES dimdate(id_date_dim)
);
```

3. Scripts ETL utilisés dans Pentaho

3.1 Script pour nettoyer la table des faits (utilisé dans le Job global)

(Ce script va dans la section "Difficultés", mais il fait aussi partie des **scripts complets**)

```
DELETE FROM factventes;
```

3.2 Script TRUNCATE (optionnel, pour rechargement complet)

```
TRUNCATE TABLE dimclient;
TRUNCATE TABLE dimproduit;
TRUNCATE TABLE dimdate;
TRUNCATE TABLE factventes;
```

3.3 Script pour vérifier le bon chargement

```
SELECT COUNT(*) FROM dimclient;
SELECT COUNT(*) FROM dimproduit;
SELECT COUNT(*) FROM dimdate;
SELECT COUNT(*) FROM factventes;
```

4. Difficultés rencontrées

Voici la version finale, claire et professionnelle :

Blocage du Job Pentaho lors du chargement final

Une difficulté majeure est apparue lors de l'exécution du **Job global**.

Bien que chaque transformation fonctionnait individuellement, **le Job complet échouait systématiquement** lors du chargement de la table **FactVentes**.

Après investigation, la source du problème était identifiée :

➡ **FactVentes contenait déjà des données provenant d'anciennes exécutions**, ce qui provoquait

- Des doublons,
- Des conflits de clés,
- Des incohérences entre dimensions et faits.

Solution apportée

L'ajout d'un step SQL dans Pentaho avec le script suivant :

```
DELETE FROM factventes;
```

— comme montré dans la capture fournie — a permis de **vider la table avant chaque rechargement**.

Une fois cette solution appliquée, le Job global a fonctionné correctement et l'ensemble du DWH a pu être rechargé de manière fiable.

Ce problème a montré l'importance :

- De gérer l'état des tables lors des rechargements ETL,
- De tester les jobs dans des conditions réelles,
- De prévoir des scripts de nettoyage dans les pipelines décisionnels.