

# Data-Driven pour la croissance d'E-Commerce

PRÉPARÉ PAR: BENTALEB SAAD



# TABLE DES MATIÈRES

## INTRODUCTION

CONTEXTE DU PROJET  
CONDUIT DU PROJET  
CADRE TECHNIQUE  
FICHE DU RGPD

## INTEGRATION

EXPLORATION  
EXTRACTION  
TRANSFORMATION  
CHARGEMENT  
DATA WAREHOUSE  
TEST UNITAIRE  
DATA MARTS  
OPTIMISATION

## VISUALISATION / ANALYSE

ANALYSE DES VENTES  
ANALYSE DES INVENTAIRES

## FINALISATION

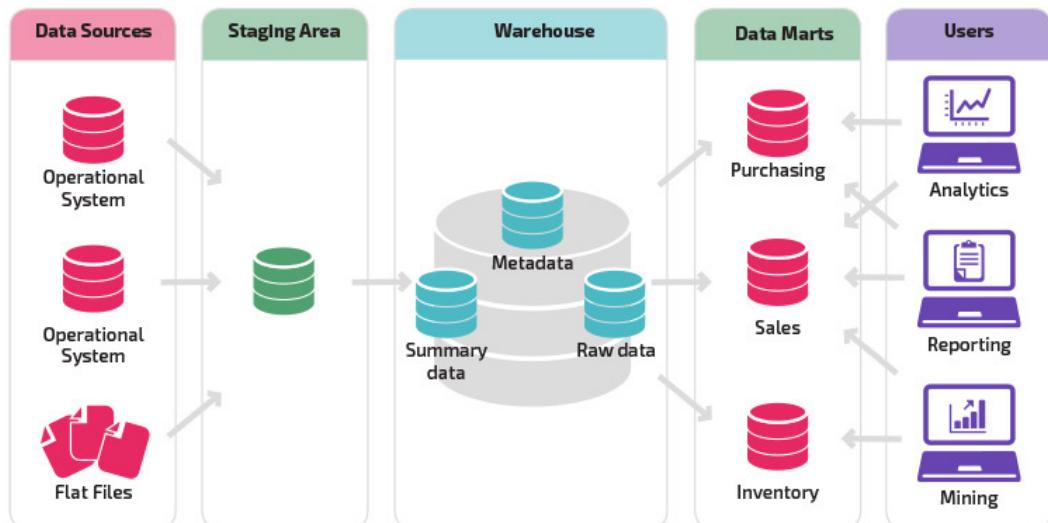
CONCLUSION



# I - INTRODUCTION

## 1 - CONTEXTE DU PROJET

La mise en place d'un entrepôt de données pour la société X, une plateforme e-commerce en croissance. L'objectif est d'optimiser les opérations commerciales et d'analyser le comportement des clients, tout en assurant la conformité au RGPD. Le projet comprend la création d'un entrepôt de données avec Talend, la construction de data marts pour un analyse ciblée, la gestion des accès basée sur les rôles, et la visualisation des données pour une meilleure prise de décision.



Aperçu du projet

## 2 - CONDUIT DU PROJET

Tout projet réussi exige une planification rigoureuse, mais également la flexibilité nécessaire pour ajuster le plan en fonction des circonstances imprévues. Dans notre cas, nous avons opté pour la planification à l'aide du bien connu "Diagramme de Gantt". Voici un aperçu de notre démarche, avec une version détaillée en pièce jointe pour référence.

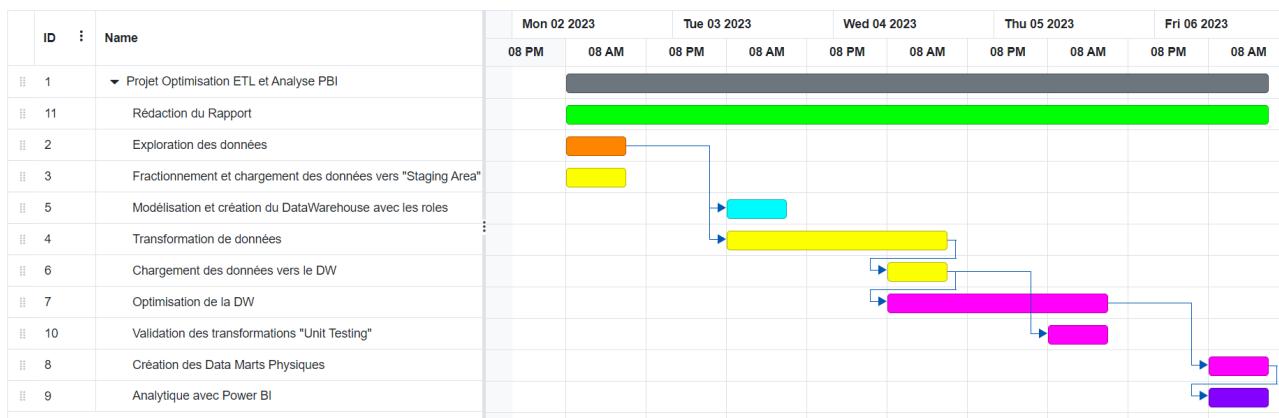


Diagramme de Gantt

## 3 - CADRE TECHNIQUE

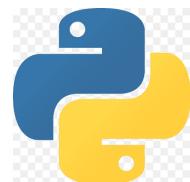
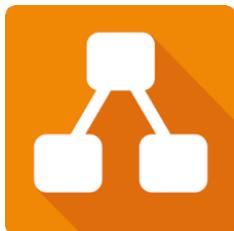
Dans notre projet, nos principales exigences technologiques étaient :

- Un outil puissant d'intégration de données : Talend open studio.
- Une base de données fiable pour l'entreposage de données et les data marts : Microsoft SQL Server.
- Un outil de visualisation de données riche et performant : Microsoft PowerBI.



Principales technologies

D'un autre côté, nous avons utilisé de nombreuses autres technologies pour atteindre nos objectifs dans ce projet, voici une liste de certaines de ces autres technologies :



Autre technologies

## 3 - RESPECT DU RGPD

En tant qu'ingénieurs de données, il est de la plus haute importance pour nous de garantir que notre manipulation des données ne porte pas atteinte au droit à la vie privée des individus. C'est pourquoi nous nous conformons strictement au Règlement Général sur la Protection des Données (RGPD). Vous trouverez ci-joint un document exhaustif détaillant notre approche et notre respect du RGPD dans l'utilisation des données.

# **II - INTEGRATION DES DONNÉES**

---

## **1 - INTRODUCTION**

Dans cette section centrale du projet, nos objectifs étaient les suivants:

- Explorer les données et détecter les anomalies.
- Charger les données de diverses sources dans une zone de transit sécurisée tout en respectant le RGPD.
- Appliquer les transformations nécessaires aux données mises en scène.
- Modéliser et créer un datawarehouse pour recevoir les données.
- Tester la validation de l'ETL à l'aide de tests unitaires SQL, puis essayer d'implémenter certaines techniques d'amélioration des performances.
- Enfin, construire les datamarts pour une analyse ciblée.

## **2 - EXPLORATION DES DONNÉES**

Étant donné que la taille des données n'était pas particulièrement volumineuse, nous avons choisi d'explorer les données de deux manières avant de commencer le processus d'intégration. Nous avons utilisé la puissance de Python pour cette tâche. Vous trouverez ci-joint un notebook Jupyter détaillé qui présente en profondeur l'exploration de chaque colonne de données, mettant en évidence les anomalies et les relations entre les colonnes.

En parallèle, nous avons également effectué des vérifications manuelles pour valider nos découvertes, puis nous avons généralisé ces résultats en utilisant Python pour confirmer les hypothèses que nous avions formulées lors de l'exploration manuelle. Afin de maintenir une organisation efficace, nous avons consigné chaque résultat de la phase d'exploration dans un tableau. Ce tableau contient à la fois une description des anomalies identifiées et des solutions suggérées pour y remédier. Vous trouverez ci-dessous la partie de ce tableau qui résume les problèmes détectés.

- - Personal Informations
- - Sales analysis
- - Inventories analysis

ProductPrice	CustomerName	CustomerEmail	CustomerAddress	CustomerPhone
<ul style="list-style-type: none"> <li>- InvalidPrice</li> <li>- Not same as Amount/Quantity</li> <li>- Count using Amount/Quantity</li> </ul>	<ul style="list-style-type: none"> <li>- Duplicates</li> </ul>	<ul style="list-style-type: none"> <li>- Nulls</li> <li>- Duplicates, More than 1 for a name</li> </ul>	<ul style="list-style-type: none"> <li>- Duplicates, More than 1 for a name</li> <li>- Only logical is state codes.</li> </ul>	<ul style="list-style-type: none"> <li>- Patterns : ~00000000000000000000000000000000</li> <li>- Extract user state based on code.</li> <li>- additional : 'x' + numbers</li> </ul>
Date	ProductName	ProductCategory	SupplierLocation	SupplierContact
<ul style="list-style-type: none"> <li>- Patterns : yyyy-mm-dd</li> <li>- Unify dd-mm-yyyy</li> </ul>	<ul style="list-style-type: none"> <li>- NonExistentProduct</li> <li>- Predicted using Price and subcategory.</li> </ul>	<ul style="list-style-type: none"> <li>- InvalidCategory</li> <li>- Predicted using Price and subcategory.</li> </ul>	<ul style="list-style-type: none"> <li>- Nonsense</li> </ul>	<ul style="list-style-type: none"> <li>- Nulls</li> <li>- additional : 'x' + numbers</li> <li>- Patterns : ~00000000000000000000000000000000</li> </ul>
CustomerName	CustomerEmail	CustomerAddress	CustomerPhone	CustomerSegment

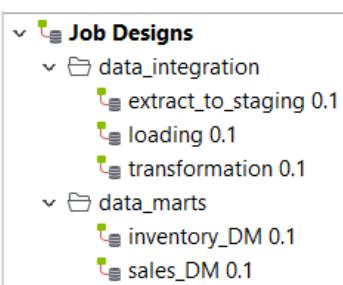
Anomalies trouvé après l'exploration

Date	ProductName	ProductCategory	ProductSubCategory	ProductPrice	CustomerName	CustomerEmail	CustomerAddress	CustomerPhone	CustomerSegment	SupplierName
<ul style="list-style-type: none"> <li>- Patterns : yyyy-mm-dd</li> <li>- Unify dd-mm-yyyy</li> </ul>	<ul style="list-style-type: none"> <li>- NonExistentProduct</li> <li>- Predicted using Price and subcategory.</li> </ul>	<ul style="list-style-type: none"> <li>- InvalidCategory</li> <li>- Predicted using Price and subcategory.</li> </ul>	✓	<ul style="list-style-type: none"> <li>- InvalidPrice</li> <li>- Not same as Amount/Quantity</li> <li>- Count using Amount/Quantity</li> </ul>	<ul style="list-style-type: none"> <li>- Duplicates</li> </ul>	<ul style="list-style-type: none"> <li>- Nulls</li> <li>- Duplicates, More than 1 for a name</li> </ul>	<ul style="list-style-type: none"> <li>- Duplicates, More than 1 for a name</li> <li>- Only logical is state codes.</li> </ul>	<ul style="list-style-type: none"> <li>- Patterns : ~00000000000000000000000000000000</li> <li>- Extract user state based on code.</li> <li>- additional : 'x' + numbers</li> </ul>	✓	✓
SupplierLocation	SupplierContact	SupplierName	ShippingMethod	QuantitySold	TotalAmount	DiscountAmount	NetAmount	StockReceived	StockSold	StockOnHand
X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓

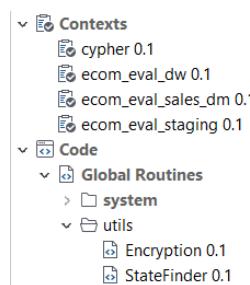
Resultats de l'exploration

## 3 - EXTRACTION DES DONNÉES

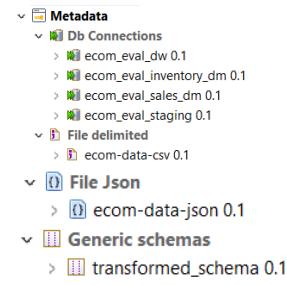
Avant de plonger dans le job d'extraction avec Talend, voici tout d'abord la structure des fichiers de mon projet Talend, dans laquelle le job de chargement représente le job complet qui fait appel à tous les autres jobs "Le job principal".



Structure des Jobs

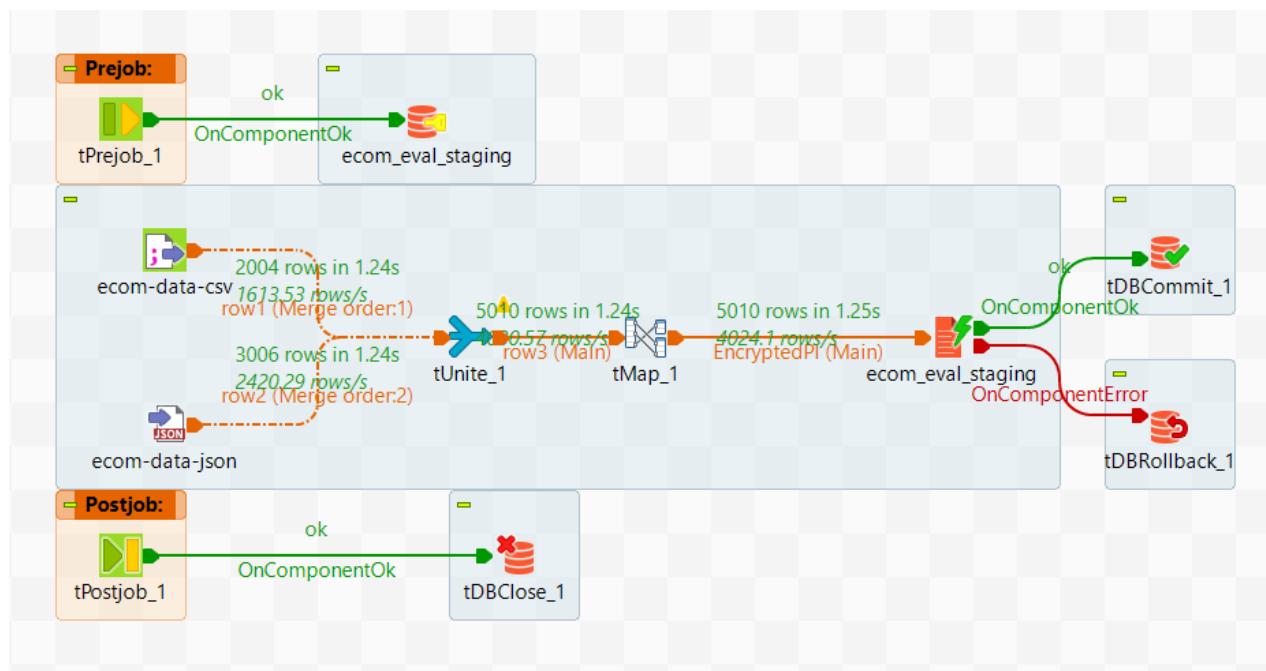


Contexte et routines

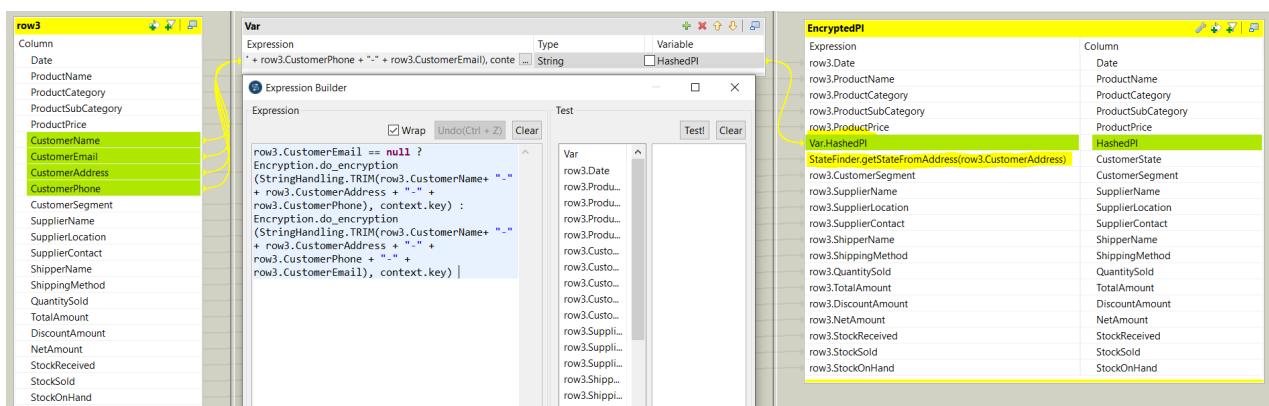


Metadata

Pour l'extraction des données, nous avons deux sources : un fichier CSV et un fichier JSON. Nous avons créé des métadonnées pour chaque fichier pour définir des schémas réutilisables et nous assurer de respecter le RGPD même dans la zone de staging, où il est crucial de ne pas stocker d'informations personnelles identifiables. Nous avons effectué des transformations telles que la suppression de colonnes inutiles, le cryptage des données personnelles et l'extraction de l'État à partir de l'adresse pour une géo-analyse des utilisateurs.



Job d'extraction



Transformations a l'extraction

```

package routines;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Base64;

public class Encryption {

    public static String do_encryption(String text, String passwd) throws Exception {
        String password = passwd;
        byte[] passwordBytes = password.getBytes(StandardCharsets.UTF_8);
        SecretKeySpec secretKeySpec = new SecretKeySpec(passwordBytes, "AES");

        // Create a cipher for encryption
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);

        //Code generated according to input schema and output schema
        return Base64.getEncoder().encodeToString(cipher.doFinal(text.getBytes(StandardCharsets.UTF_8)));
    }

    public static String do_decryption(String text, String passwd) throws Exception {
        String password = passwd;
        byte[] passwordBytes = password.getBytes(StandardCharsets.UTF_8);
        SecretKeySpec secretKeySpec = new SecretKeySpec(passwordBytes, "AES");
        // Create a cipher for encryption
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);

        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(text));
        //Code generated according to input schema and output schema
        String decryptedMessage = new String(decryptedBytes, StandardCharsets.UTF_8);
        return decryptedMessage;
    }
}

```

## Encryptage

```

package routines;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class StateFinder {

    private static final Map<String, String> stateCodeToNameMap = new HashMap<>();

    static {}

    public static String getStateFromAddress(String address) {
        if (address != null) {
            Pattern apoDpoFpoPattern = Pattern.compile("(?i)(APO|DPO|FPO)");
            Matcher apoDpoFpoMatcher = apoDpoFpoPattern.matcher(address);

            if (apoDpoFpoMatcher.find()) {
                return "Armed Forces";
            }

            Pattern pattern = Pattern.compile("[A-Z]{2}");
            Matcher matcher = pattern.matcher(address);

            if (matcher.find()) {
                String stateCode = matcher.group();
                String stateName = stateCodeToNameMap.get(stateCode);
                if (stateName != null) {
                    return stateName;
                }
            }
        }
        return "Unknown state";
    }
}

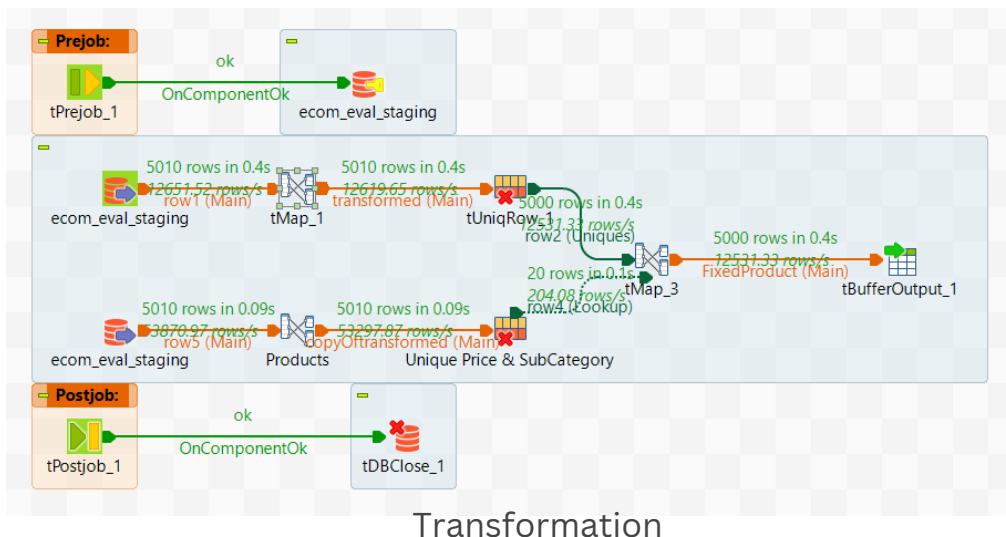
```

## Extraction de l'état

# 4 - TRANSFORMATION DES DONNÉES

Après les transformations appliquées en phase d'extraction, voici les modifications effectuées sur les colonnes suivantes :

- **Date:** Le format a été uniformisé en "jj-mm-aaaa".
- **ProductName & ProductCategory:** Les valeurs manquantes ont été estimées en fonction du prix et de la sous-catégorie des produits.
- **ProductPrice:** Ce champ a été calculé en utilisant la formule TotalAmount / QuantitySold.



Transformation

This screenshot shows the Expression Builder dialog for a transformation step. On the left, the 'row1' context is listed with a 'Date' column selected. In the center, the 'Var' section defines a variable 'Date' with the expression `.TO_DATE(row1.Date, "MM-dd-yyyy")`. On the right, the 'transformed' context is shown with a 'Date' column, indicating the resulting transformed data structure.

Date & ProductPrice

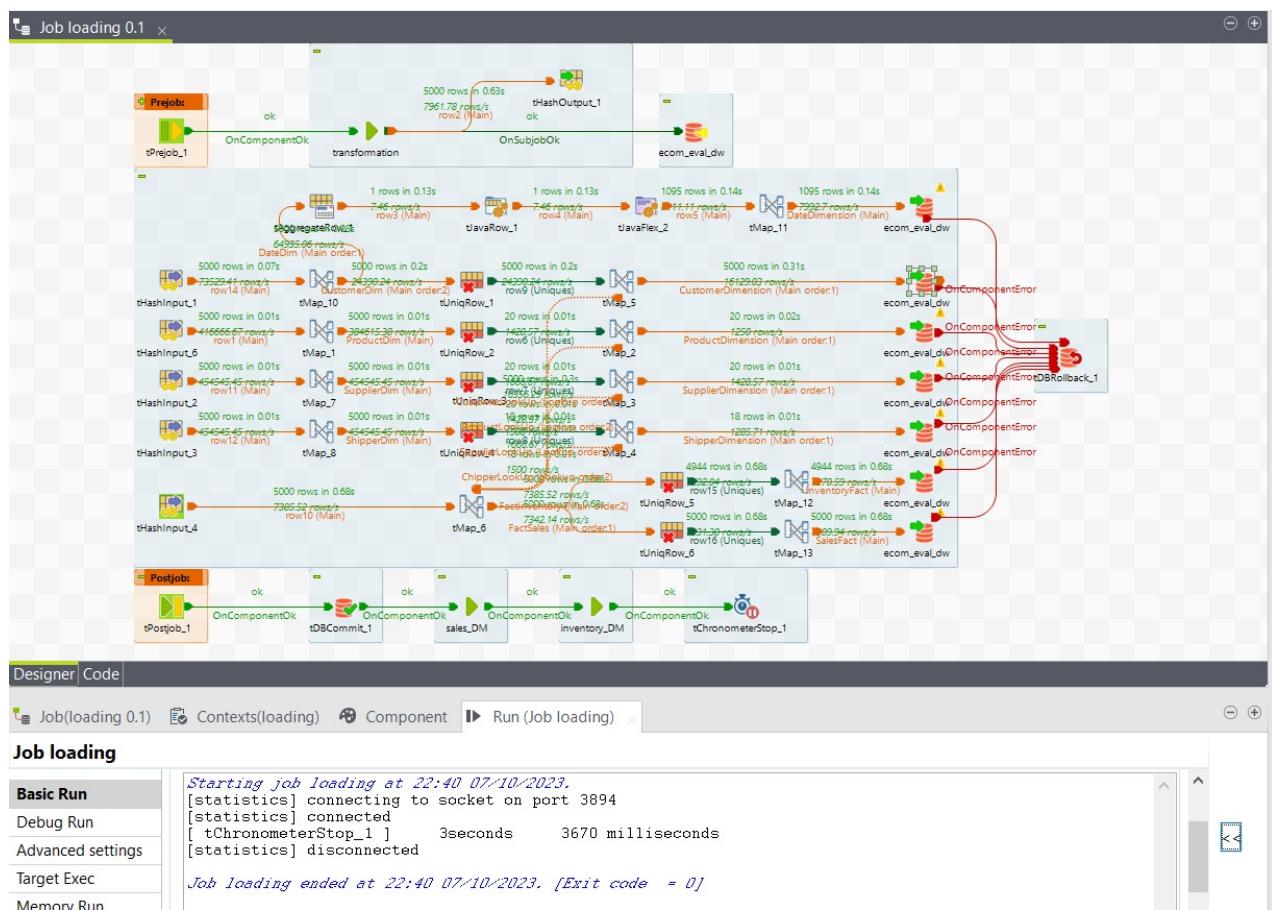
This screenshot shows the Expression Builder dialog for a transformation step. On the left, the 'row4' context is listed with properties like 'Lookup Model' set to 'Load once' and 'Match Model' set to 'Unique match'. In the center, the 'Var' section is empty. On the right, the 'FixedProduct' section defines variables for ProductName, ProductCategory, and ProductSubCategory based on row2 values.

ProductName & ProductCategory

# 5 - CHARGEMENT DES DONNÉES

Les données transformées ont été acheminées vers un tBufferOutput, puis récupérées lors de la phase de chargement. Ensuite, nous les avons stockées dans un tHashInput pour les rendre disponibles pour la suite du processus. Pour l'intégration dans le datawarehouse, nous avons utilisé plusieurs tHashOutput pour extraire les données transformées et les charger dans la base de données. Le processus a débuté par le chargement des dimensions, suivi par celui des tables de faits.

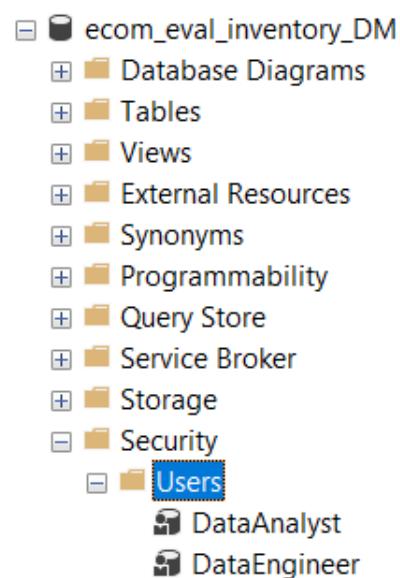
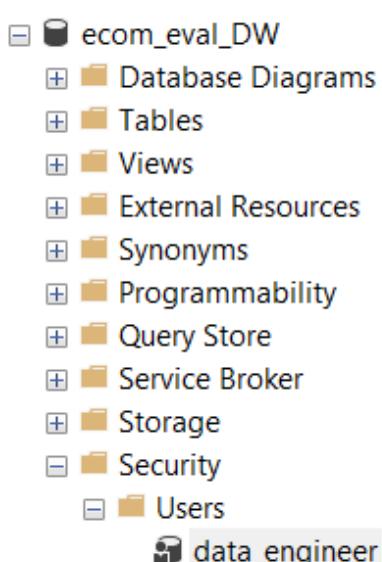
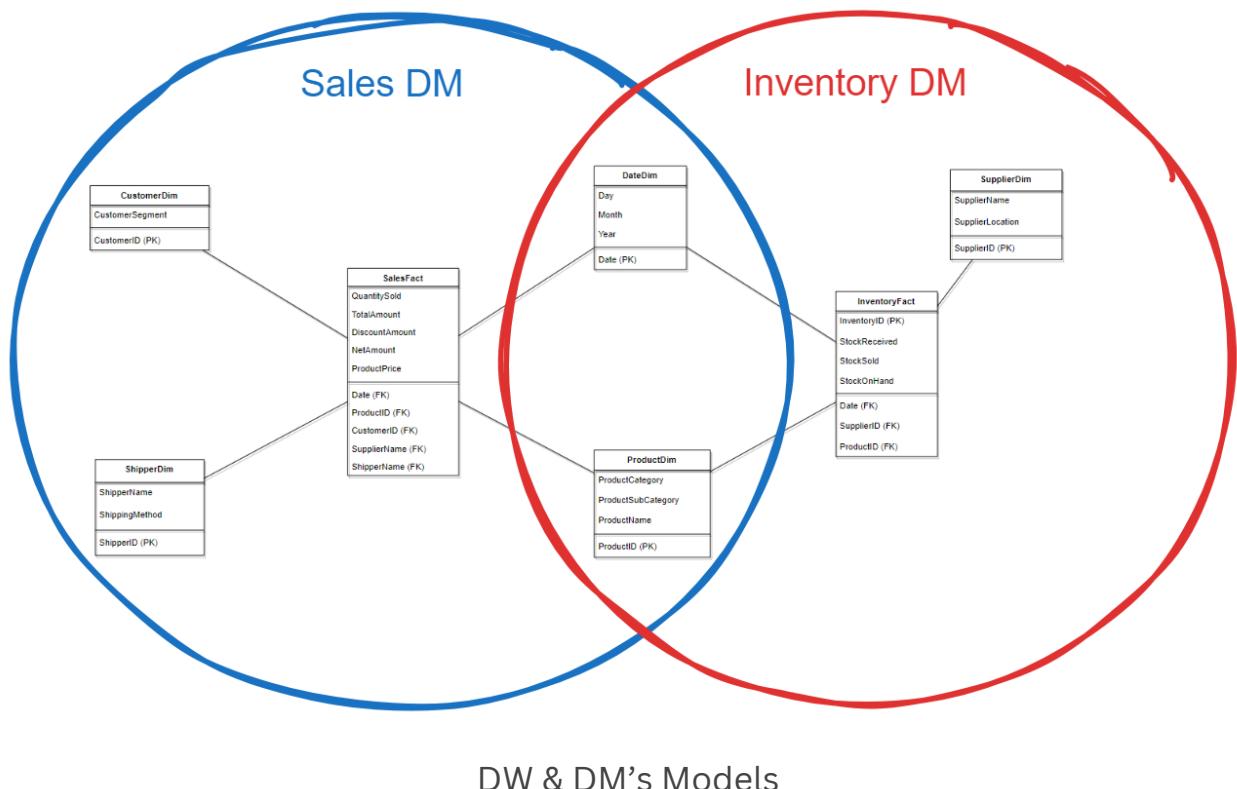
Dans le cadre de ce chargement, nous avons accordé une attention particulière à la gestion des mises à jour des dimensions en utilisant la méthode SCD (Slowly Changing Dimension). La plupart des tables ont suivi la stratégie SCD de type 1, ce qui signifie que les changements sont simplement écrasés par les nouvelles données, sans historisation. Cependant, dans des cas spécifiques, comme la table des dates, nous avons opté pour la stratégie SCD de type 0, où aucune mise à jour ou historisation n'est nécessaire car les données sont immuables.



Chargement ( Job total )

# 6 - DATA WAREHOUSE

Nous avons opté pour un modèle de schéma de constellation de faits (Galaxy) pour notre entrepôt de données. Contrairement à la normalisation, ce modèle ne divise pas excessivement les données en tables distinctes, privilégiant ainsi la vitesse d'accès aux données plutôt que la qualité de la structure. Cette décision a été motivée par le fait que nous ne traitons pas de Big Data, ce qui signifie que le stockage n'est pas un problème majeur pour nous.



Gestion d'accès

# 7 - TEST UNITAIRE

Afin de garantir que les transformations ont été correctement appliquées, nous avons mis en place des tests unitaires pour les colonnes impactées par ces transformations. Pour ce faire, nous avons utilisé tSQLt en conjonction avec le plugin DBFORGE pour SQL Server. Ci-dessous, vous trouverez les tests que nous avons réalisés ainsi que leurs résultats :

1. Test de format de Date : Nous avons vérifié que la colonne "Date" respecte le format "jj-mm-aaaa". Résultat : Les données passent le test avec succès.
2. Test de valeurs manquantes pour ProductName & ProductCategory : Nous avons confirmé que les valeurs manquantes de ces colonnes ont été correctement estimées en fonction du prix et de la sous-catégorie des produits. Résultat : Les valeurs manquantes ont été correctement remplies.
3. Test de calcul de ProductPrice : Nous avons vérifié que le calcul de la colonne "ProductPrice" en utilisant TotalAmount / QuantitySold est précis. Résultat : Le calcul est correctement effectué pour toutes les lignes.

Test Results		
Status	Test Name	Class Name
<span style="color: green;">✓ Succeeded</span>	ValidDateFormat	Date
<span style="color: green;">✓ Succeeded</span>	ValidNameAndCategory	Product
<span style="color: green;">✓ Succeeded</span>	ValidPrice	Product

Resultat des tests

```

ALTER PROCEDURE Date.[test DateFormat]
AS
BEGIN
    -- Act: Query the table for dates with an incorrect format
    DECLARE @incorrectFormatDates INT;
    SELECT @incorrectFormatDates = COUNT(*)
    FROM [ecom_eval_DW].[dbo].[DateDim]
    WHERE TRY_CAST([date] AS DATETIME) IS NULL
    OR FORMAT([date], 'yyyy-MM-dd') != [date];

    -- Assert: Verify that there are no dates with an incorrect format
    EXEC tSQLt.AssertEquals 0, @incorrectFormatDates;

END;

```

### Test de validité de la date

```

ALTER PROCEDURE Product.[test ValidNameAndCategory]
AS
BEGIN
    -- Act: Query the table for invalid product names and categories
    DECLARE @invalidProductNames INT;
    DECLARE @invalidProductCategories INT;

    SELECT @invalidProductNames = COUNT(*)
    FROM [ecom_eval_DW].[dbo].[ProductDim]
    WHERE [ProductName] = 'NonExistentProduct';

    SELECT @invalidProductCategories = COUNT(*)
    FROM [ecom_eval_DW].[dbo].[ProductDim]
    WHERE [ProductCategory] = 'InvalidCategory';

    -- Assert: Verify that there are no occurrences of invalid product names or categories
    EXEC tSQLt.AssertEquals 0, @invalidProductNames;
    EXEC tSQLt.AssertEquals 0, @invalidProductCategories;

END;

```

### Test de validité des ProductName & ProductCategory

```

ALTER PROCEDURE Product.[test ValidPrice]
AS
BEGIN
    -- Act: Query the table for discrepancies in ProductPrice calculation
    DECLARE @incorrectCalculations INT;

    SELECT @incorrectCalculations = COUNT(*)
    FROM [ecom_eval_DW].[dbo].[FactSales]
    WHERE ROUND([TotalAmount] / [QuantitySold], 0) != [ProductPrice];

    -- Assert: Verify that there are no discrepancies in ProductPrice calculation
    EXEC tSQLt.AssertEquals 0, @incorrectCalculations;

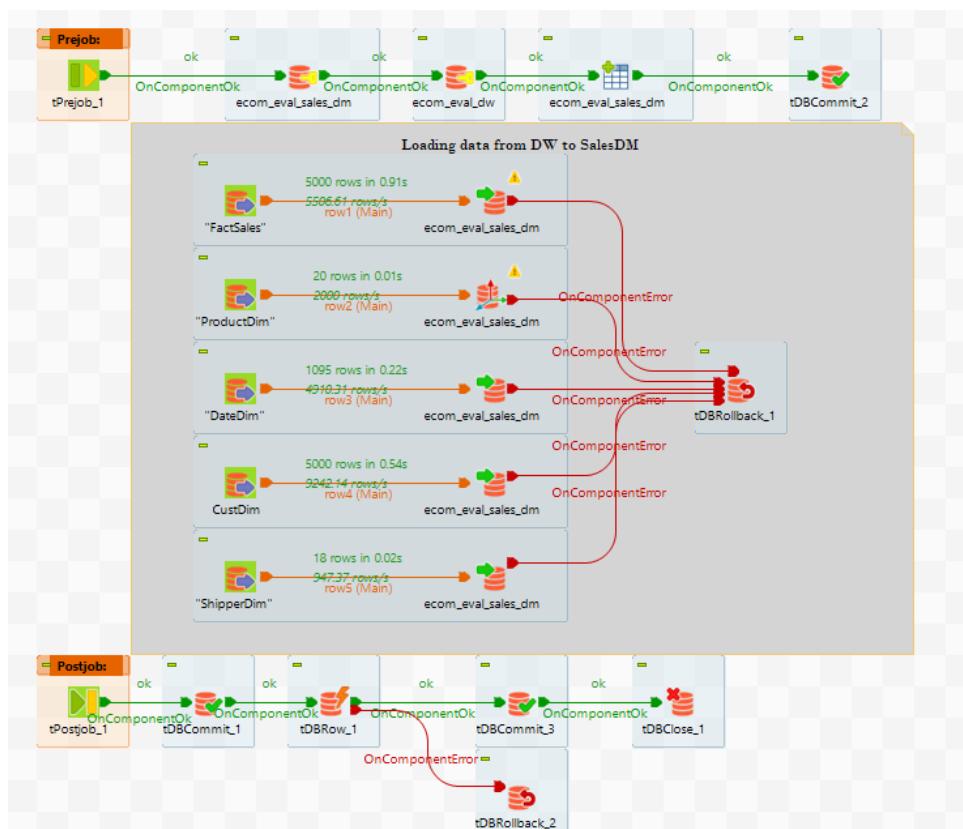
END;

```

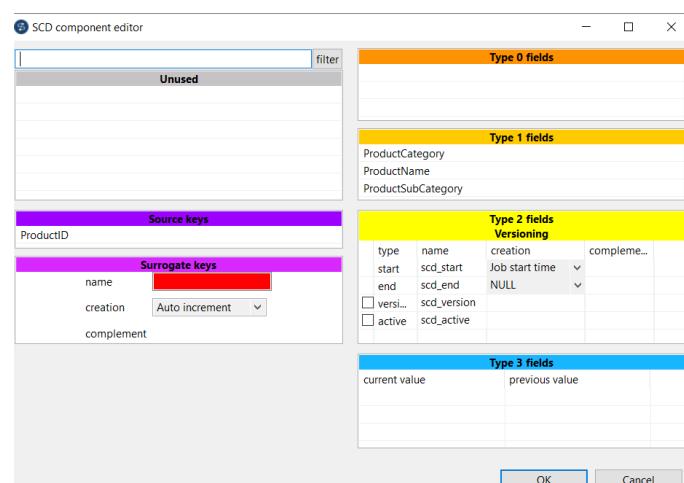
### Test de validité des ProductPrice

# 8 - DATAMARTS

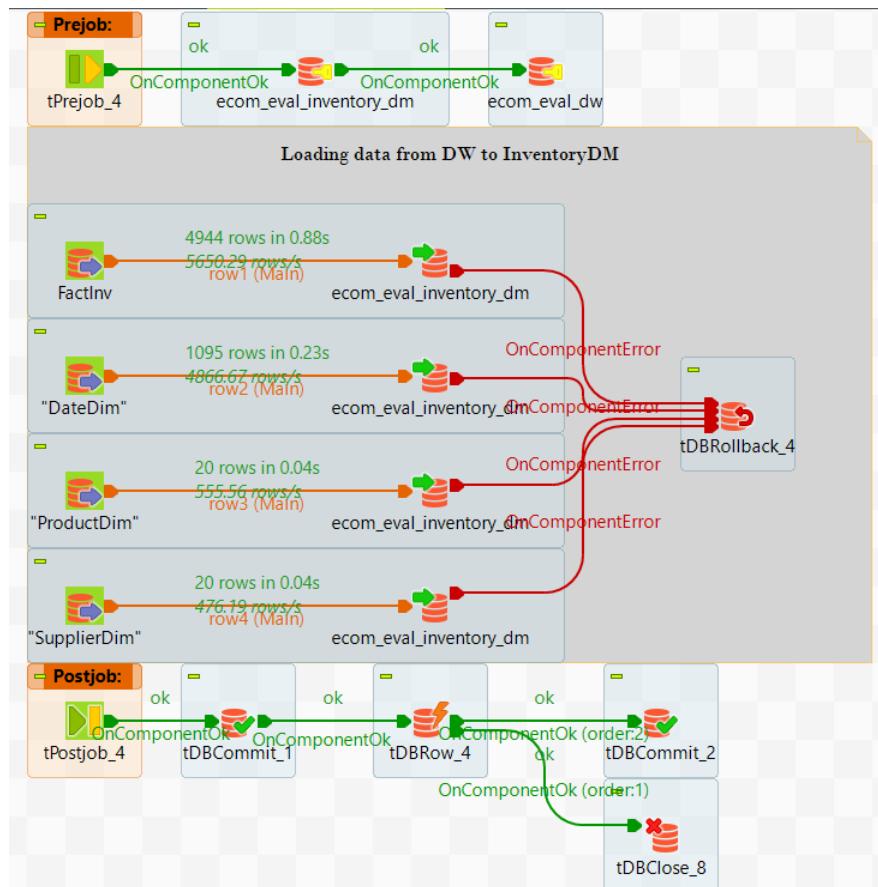
Pour les datamarts, le processus était simple : récupération des tables spécifiées depuis l'entrepôt de données et chargement dans les datamarts en utilisant la connexion du rôle DataEngineer, autorisant l'écriture et la mise à jour. Nous avons respecté les principes du SCD (Slowly Changing Dimension) en appliquant la stratégie SCD de type 1 pour la plupart des tables, tandis que certaines, comme la table des dates, ont suivi la stratégie SCD de type 0.



Sales DM



Composant du SCD



## Inventory DM

**ecom\_eval\_inventory\_dm(tDBOutput\_1)(Microsoft SQL Server)**

**Basic settings**

- Database: Microsoft SQL Server
- Table: "FactInventory"
- Action on table: Create table if does not exist
- Action on data: Update or insert
- Schema: Built-In

## SCD Type 1 utilisant tdbOutput

**ecom\_eval\_inventory\_dm(tDBOutput\_2)(Microsoft SQL Server)**

**Basic settings**

- Database: Microsoft SQL Server
- Table: "DateDim"
- Action on table: Create table if does not exist
- Action on data: Insert if not exist
- Schema: Built-In

## SCD Type 0 utilisant tdbOutput

# 9 - OPTIMISATION

Pour optimiser les temps de requête, nous avons ciblé l'indexation, car elle représente un moyen efficace d'améliorer les performances des requêtes, en particulier sur les datamarts qui seront soumis à des analyses et à des requêtes quotidiennes. Voici nos choix d'indexation pour chaque datamart :

Datamart Inventaire (Tableau FactInventory) :

1. INDEX1 sur [ProductID] : Cet index vise à améliorer les performances lors du filtrage ou de la jonction de la table FactInventory en fonction de ProductID, ce qui est probablement un critère de filtrage courant.
2. INDEX2 sur [Date] : Si les requêtes basées sur la date sont fréquentes, cet index peut accélérer les opérations de filtrage ou d'agrégation liées à cette colonne.

```
-- Index on ProductID
CREATE INDEX IX_FactInventory_ProductID ON [ecom_eval_inventory_DM].[dbo].[FactInventory] ([ProductID]);

-- Index on Date
CREATE INDEX IX_FactInventory_Date ON [ecom_eval_inventory_DM].[dbo].[FactInventory] ([Date]);
```

%  
Messages  
Commands completed successfully.  
Completion time: 2023-10-08T14:39:04.6708798+01:00

## Inventory Fact Indexes

Datamart Ventes (Tableau FactSales) :

1. INDEX1 sur [ProductID] : Comme pour la table FactInventory, un index sur ProductID peut améliorer les performances des requêtes impliquant cette colonne.
2. INDEX2 sur [Date] : Si les requêtes liées à la date sont courantes, un index sur la colonne [Date] peut contribuer à accélérer ces opérations.
3. INDEX3 sur [CustomerID] : Si vous interrogez fréquemment les données de ventes par client, un index sur CustomerID peut être bénéfique.
4. INDEX4 sur [ShipperID] : Si vous effectuez régulièrement des requêtes impliquant des expéditeurs, un index sur ShipperID peut améliorer les performances de ces requêtes.

```
-- Index on ProductID
CREATE INDEX IX_FactSales_ProductID ON [ecom_eval_DW].[dbo].[FactSales] ([ProductID]);

-- Index on Date
CREATE INDEX IX_FactSales_Date ON [ecom_eval_DW].[dbo].[FactSales] ([Date]);

-- Index on CustomerID
CREATE INDEX IX_FactSales_CustomerID ON [ecom_eval_DW].[dbo].[FactSales] ([CustomerID]);

-- Index on ShipperID
CREATE INDEX IX_FactSales_ShipperID ON [ecom_eval_DW].[dbo].[FactSales] ([ShipperID]);
```

## Sales Fact Indexes

La deuxième méthode d'optimisation que nous avons mise en œuvre est le partitionnement des tables en fonction de l'année. Ainsi, chaque année, les données sont stockées dans une partition distincte, ce qui contribue à améliorer la vitesse de traitement des requêtes. Voici le processus de partitionnement que nous avons suivi, ainsi que le nombre de lignes dans chaque partition après partitionnement

```
-- Create a partition function for years 2021, 2022, and 2023
CREATE PARTITION FUNCTION SalesDatePartitionFunction (DATE)
AS RANGE RIGHT FOR VALUES ('2021-01-01', '2022-01-01', '2023-01-01');
```

### Fonction de partitionnement

```
-- Create Partition file groups
ALTER DATABASE ecom_eval_sales_DM ADD FILEGROUP [FG_sales_Archive];
ALTER DATABASE ecom_eval_sales_DM ADD FILEGROUP [FG_sales_2021];
ALTER DATABASE ecom_eval_sales_DM ADD FILEGROUP [FG_sales_2022];
ALTER DATABASE ecom_eval_sales_DM ADD FILEGROUP [FG_sales_2023];

-- Create Files for file groups
ALTER DATABASE ecom_eval_sales_DM ADD FILE
(NAME = N'Ventes_Archive',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Ventes_Archive.ndf', SIZE = 2048KB) TO FILEGROUP [FG_sales_Archive];

ALTER DATABASE ecom_eval_sales_DM ADD FILE
(NAME = N'Ventes_2021',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Ventes_2021.ndf', SIZE = 2048KB) TO FILEGROUP [FG_sales_2021];

ALTER DATABASE ecom_eval_sales_DM ADD FILE
(NAME = N'Ventes_2022',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Ventes_2022.ndf', SIZE = 2048KB) TO FILEGROUP [FG_sales_2022];

ALTER DATABASE ecom_eval_sales_DM ADD FILE
(NAME = N'Ventes_2023',
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\Ventes_2023.ndf', SIZE = 2048KB) TO FILEGROUP [FG_sales_2023];
```

### Creation des fileGroups et des files

```
-- Create a partition scheme
CREATE PARTITION SCHEME SalesPartitionScheme
AS PARTITION SalesDatePartitionFunction
TO ([FG_sales_Archive], [FG_sales_2021], [FG_sales_2022], [FG_sales_2023]);

-- Create the partitioned fact table directly from the source data
CREATE TABLE SalesFactPartitioned
(
    SalesID INT,
    Date DATE,
    ProductID INT,
    CustomerID INT,
    ShipperID INT,
    QuantitySold INT,
    TotalAmount DECIMAL(10, 2),
    DiscountAmount DECIMAL(10, 2),
    NetAmount DECIMAL(10, 2),
    PRIMARY KEY (SalesID, Date)
)
ON SalesPartitionScheme(Date);
```

### Creation du schéma ainsi que la table partitionnée

```
-- Insert data into the partitioned fact table
INSERT INTO SalesFactPartitioned (SalesID, Date, ProductID, CustomerID, ShipperID, QuantitySold, TotalAmount, DiscountAmount, NetAmount)
SELECT SalesID, Date, ProductID, CustomerID, ShipperID, QuantitySold, TotalAmount, DiscountAmount, NetAmount
FROM FactSales;
```

### insertion des données au partitions

```
SELECT
    p.partition_number AS partition_number,
    f.name AS file_group,
    p.rows AS row_count
FROM sys.partitions p
JOIN sys.destination_data_spaces dds ON p.partition_number = dds.destination_id
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id
WHERE OBJECT_NAME(OBJECT_ID) = 'SalesFactPartitioned'
order by partition_number;
```

Results    Messages

	partition_number	file_group	row_count
1	1	FG_sales_Archive	0
2	2	FG_sales_2021	637
3	3	FG_sales_2022	2440
4	4	FG_sales_2023	1923

Lignes par partition

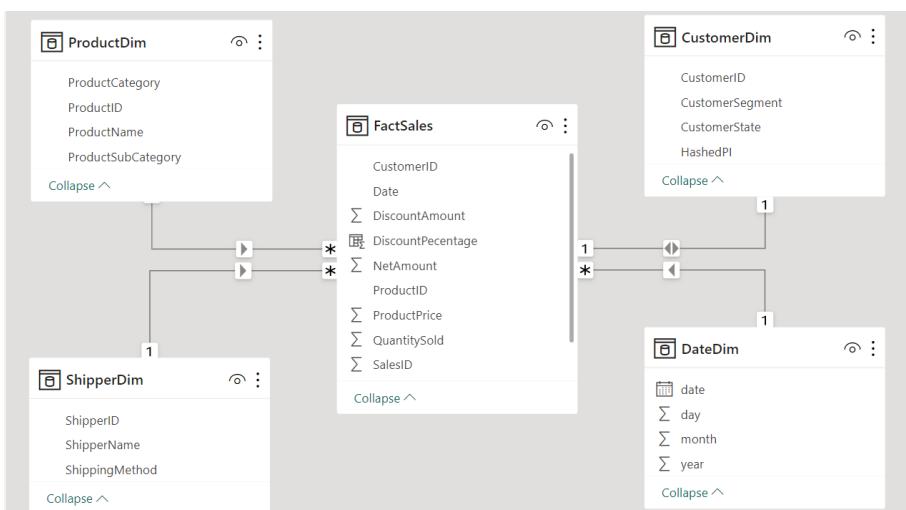
# III - VISUALISATION

## 1 - INTRODUCTION

En conclusion de ce projet, nous avons achevé la phase finale en créant des visualisations ciblées des ventes et des stocks. L'objectif principal était d'extraire des informations clés et des indicateurs essentiels à partir des données. Ces visualisations nous ont permis de fournir à la société X les outils nécessaires pour prendre des décisions stratégiques éclairées dans le domaine du commerce électronique. Cette étape de visualisation a transformé les données brutes en un tableau de bord interactif et informatif, fournissant un aperçu essentiel de la performance commerciale, des tendances et du comportement des clients.

## 2 - ANALYSE DES VENTES

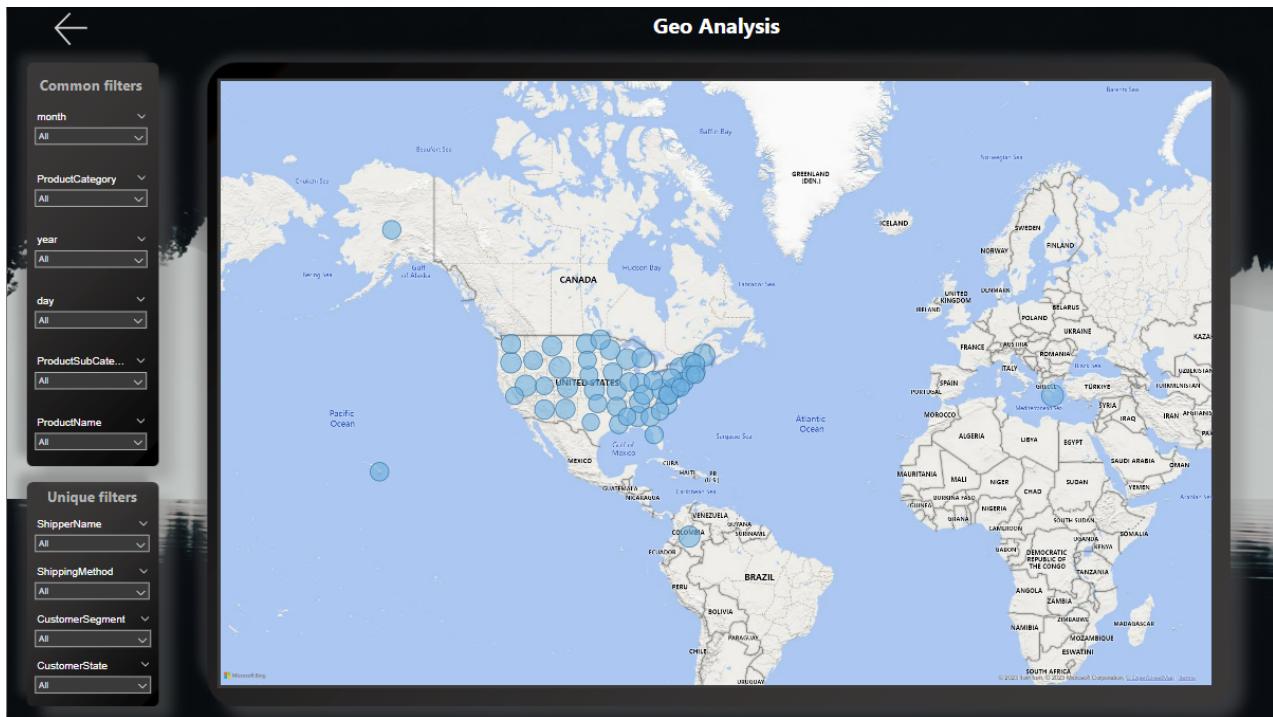
Le tableau de bord des ventes est connecté au datamart des ventes via le rôle DataAnalyst, qui a des droits de lecture sur le datamart. Nous avons choisi une connexion de chargement de données en raison de la taille modérée des données et du fait qu'elles ne nécessitent pas de mises à jour fréquentes. Le tableau de bord offre une page principale répondant aux besoins essentiels de l'entreprise, notamment l'analyse des tendances des ventes, des produits, de la segmentation des clients, des réductions et de la performance des transporteurs. De plus, une page distincte est dédiée à la visualisation de la répartition géographique des clients, ce qui peut fournir des informations cruciales sur la distribution géographique des clients. Dans l'ensemble, ce tableau de bord permet aux analystes de données de tirer des insights importants pour l'entreprise, contribuant ainsi à des décisions éclairées dans le secteur du commerce électronique.



Chargement des données du Sales DM



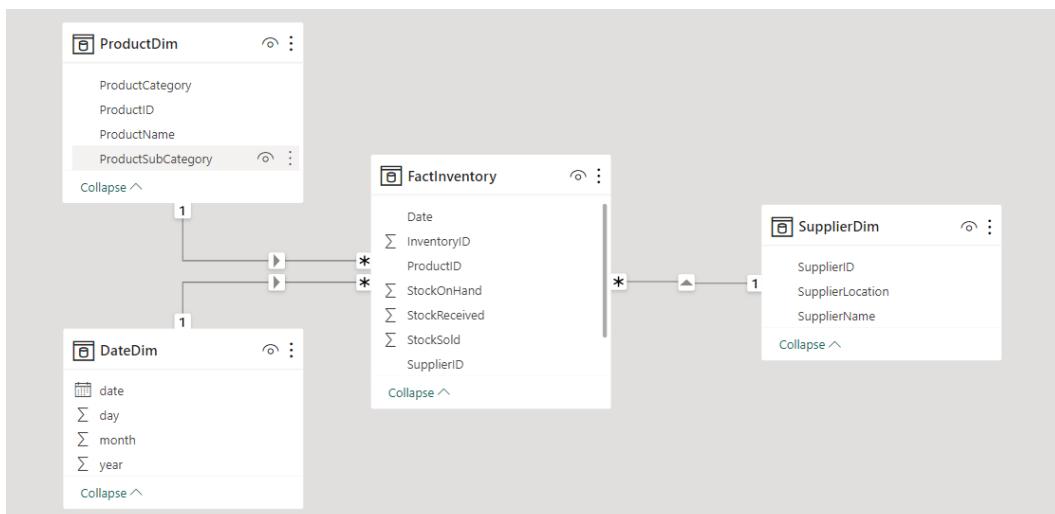
## Sales Dashboard



## Analyse géographique

# 3 - ANALYSE DES INVENTAIRES

Pour le deuxième tableau de bord dédié à l'analyse des stocks et à la disponibilité des produits, nous avons créé une page unique pour répondre aux besoins spécifiques de l'entreprise. Les données ont été chargées de la même manière que pour le tableau de bord des ventes, en utilisant le rôle DataAnalyst. Ce tableau de bord a été conçu pour satisfaire les exigences de l'entreprise en matière d'analytique du Data Mart de l'Inventaire, notamment en surveillant le niveau d'inventaire, en analysant la disponibilité des stocks, en évaluant la performance des fournisseurs et en prévoyant la demande de produits. Cette page unique offre une vue complète des données d'inventaire et permet aux analystes de données de prendre des décisions informées concernant la gestion des stocks et l'approvisionnement en produits.



Chargement des données du Inventory DM



Inventory dashboard

# IV - CONCLUSION

---

Ce projet de gestion de données pour la société X dans le secteur du commerce électronique a été mené avec succès, en mettant l'accent sur la qualité des données, la conformité au RGPD et l'optimisation des performances. Nous avons débuté par l'exploration et la préparation des données, en veillant à respecter la vie privée des individus et en éliminant les données superflues. La création d'un entrepôt de données basé sur le modèle de schéma de constellation de faits a permis d'accélérer l'accès aux données sans sacrifier la qualité. Les transformations, tests unitaires et stratégies SCD ont garantie la précision des données à chaque étape.

En matière d'optimisation, nous avons choisi des techniques telles que l'indexation et le partitionnement, adaptées à nos besoins spécifiques, tout en mettant en place des tableaux de bord interactifs pour l'analyse des ventes et des stocks. En fin de compte, ce projet a permis à la société X d'exploiter pleinement ses données, de prendre des décisions éclairées et de prospérer dans le secteur du commerce électronique grâce à une gestion de données robuste et à des outils d'analyse puissants.