YouCode
POWERED BY SIMPLON

Mise en Place et Optimisation d'un Entrepôt de Données et Analyse avec Power BI

Préparé par:
Mouflla Faissal

PLAN

Marketing Strategist Marketing Analyst Marketing Analyst

INTRODUCTION

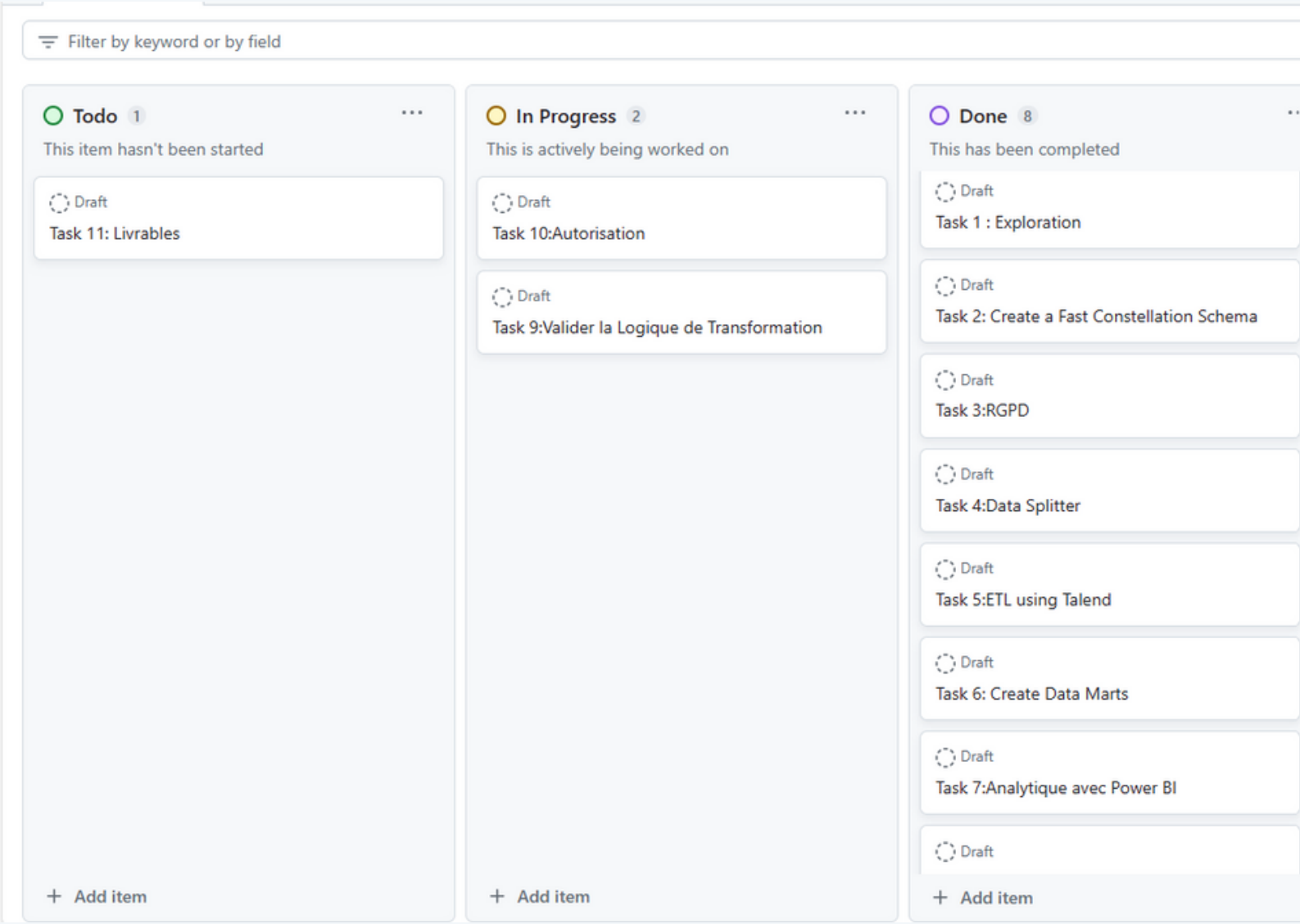
Ce rapport documente un projet essentiel axé sur la mise en place d'un entrepôt de données efficace et l'analyse des informations générées. Le projet se concentre sur l'amélioration de la gestion des données d'une plateforme e-commerce, en utilisant des technologies avancées telles que Talend pour l'ETL, SQL Server pour le stockage des données, et Power BI pour l'analyse. L'objectif principal est de créer une solution robuste de bout en bout, de l'extraction des données brutes à la génération d'insights précieux pour l'entreprise.

Le rapport décompose les différentes phases du projet, allant de l'extraction des données à leur transformation et chargement, en passant par la création de schémas de constellation et de data marts physiques. L'analyse des données est ensuite réalisée à travers diverses perspectives, y compris les tendances de vente, l'analyse des produits, la segmentation des clients, et bien plus encore, tout cela avec l'objectif de mieux comprendre les activités de l'entreprise.

En outre, ce rapport met en lumière la mise en œuvre de mesures de sécurité et de conformité au RGPD pour garantir la protection des données sensibles. Il examine également les méthodes d'optimisation de l'entrepôt de données, telles que l'indexation et le partitionnement, pour améliorer les performances.

Le projet a été mené en tant que réponse aux défis posés par la gestion des données d'une entreprise e-commerce moderne. Les détails de chaque étape, des scripts utilisés aux tests de validation, sont présentés pour une compréhension complète de l'ensemble du processus. Ce rapport illustre comment la mise en place de solutions basées sur les données peut contribuer à améliorer les opérations et à générer des informations exploitables.

PLANIFICATION



EXPLORATION DES DONNÉES AVEC PYTHON

La phase d'exploration des données avec Python a joué un rôle crucial dans la préparation des données pour notre entrepôt de données. Grâce à cette étape, nous avons pu plonger dans le cœur des informations contenues dans nos ensembles de données brutes, acquérant ainsi une compréhension approfondie de nos données. En utilisant des bibliothèques puissantes telles que Pandas, nous avons exploré les différentes colonnes de nos ensembles de données, identifiant les attributs clés et les informations critiques qui façonneront notre entrepôt de données.

En fin de compte, l'exploration des données avec Python a jeté les bases pour les étapes ultérieures de notre projet, notamment la modélisation, l'ETL avec Talend, la création de data marts, l'analytique avec Power BI, l'optimisation, la validation de la logique de transformation et l'attribution des autorisations. Cela montre l'importance cruciale de cette phase préliminaire pour le succès global de notre projet d'entrepôt de données.

APPLIQUER LES POLITIQUES RGPD SUR LES DONNÉES SENSIBLES

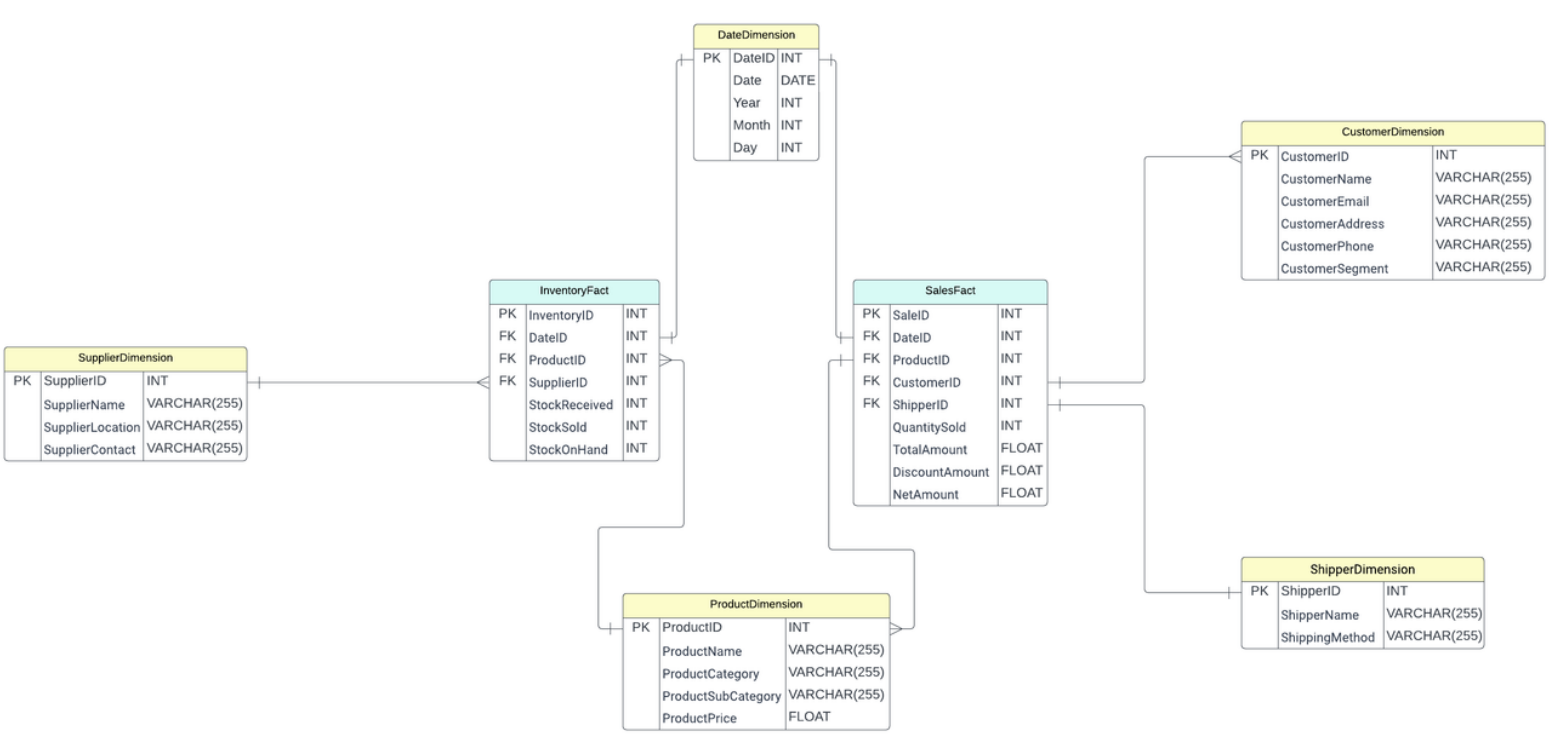
Après avoir minutieusement exploré nos données à l'aide de Python, nous avons identifié les colonnes contenant des informations sensibles qui requièrent une protection conformément aux réglementations RGPD. Dans notre contexte, les données sensibles incluent des informations personnelles telles que le nom des clients (CustomerName), leur adresse e-mail (CustomerEmail), adresse (CustomerAddress), numéro de téléphone (CustomerPhone), ainsi que la localisation de nos fournisseurs (SupplierLocation).

Pour garantir la confidentialité de ces données et se conformer aux directives RGPD, nous avons opté pour l'approche de chiffrement. Le chiffrement garantit que même en cas d'accès non autorisé aux données, elles restent inintelligibles. Ainsi, nos clients et fournisseurs peuvent être assurés que leurs informations personnelles restent protégées tout au long de leur parcours au sein de notre entrepôt de données.

Il est à noter que l'application de politiques RGPD n'est pas seulement une exigence légale, mais elle reflète également notre engagement envers la protection des données et le respect de la vie privée de nos clients et partenaires commerciaux.

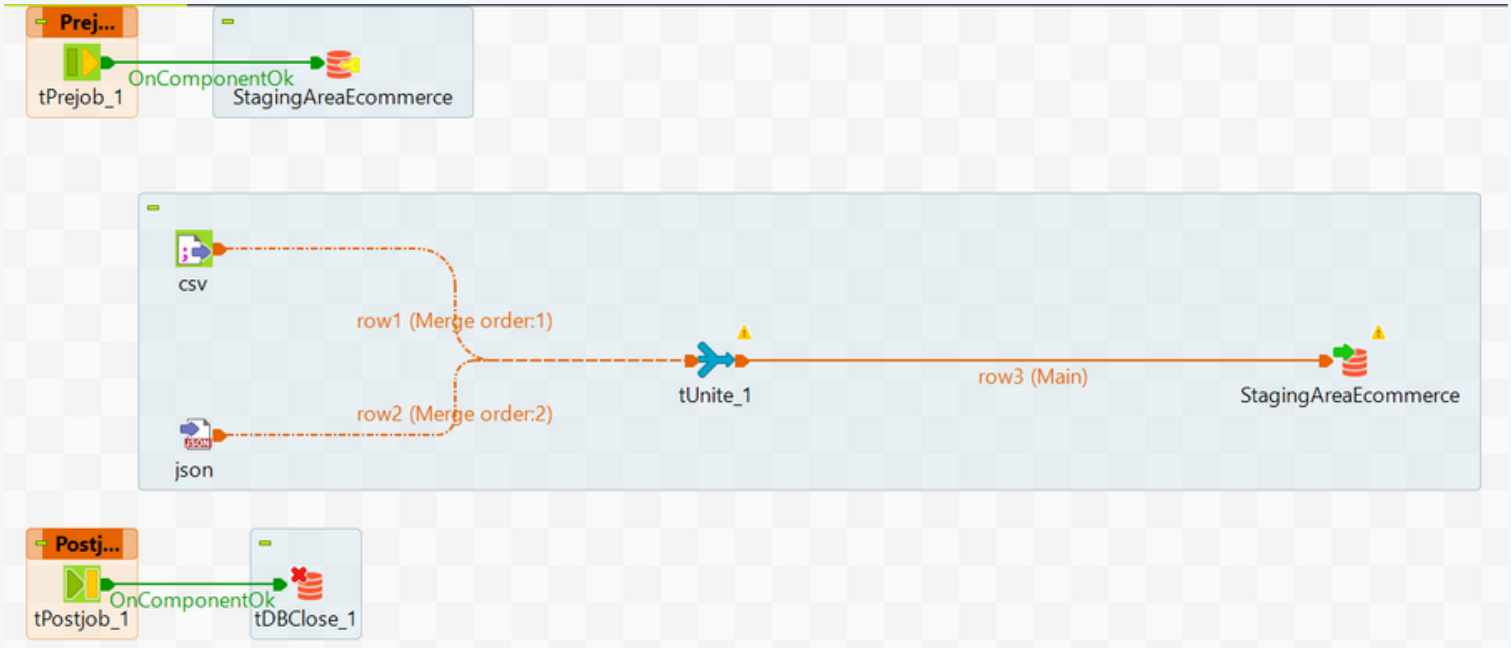
MODÉLISATION

La phase de modélisation est cruciale pour la mise en place d'un entrepôt de données efficace. Pour cette étape, nous avons conçu un schéma de base de données qui prend en compte la structure de nos données. Nous avons opté pour un modèle basé sur une constellation rapide, également connu sous le nom de modèle en étoile (star schema). Cette approche a été choisie en raison de ses avantages significatifs en matière de performances et de simplicité d'interrogation. Dans un schéma en étoile, les données sont organisées autour d'une table centrale de faits (dans notre cas, la table SalesFact), à laquelle sont liées plusieurs tablesde dimensions (SupplierDimension, ProductDimension,ShipperDimension,DateDimension,CustomerDimension) pour capturer les détails associés. Cette structure simplifie considérablement les requêtes, accélérant ainsi le processus d'analyse. De plus, le modèle en étoile facilite la navigation et la compréhension des données, ce qui est essentiel pour notre analyse avec Power BI et l'optimisation ultérieure.



ETL EN UTILISANT TALEND

Extraction des Données : Un job nommé "Extract" a été créé pour importer à la fois des fichiers JSON et CSV. Ces fichiers ont été initialement générés en divisant un fichier CSV en fichiers JSON et CSV à l'aide d'un script de fractionnement de données. Le job "Extract" a ensuite combiné ces fichiers à l'aide du composant TUnite et a inséré les données fusionnées dans une table appelée "StAEcommerce" dans la base de données "StagingAreaEcommerce".



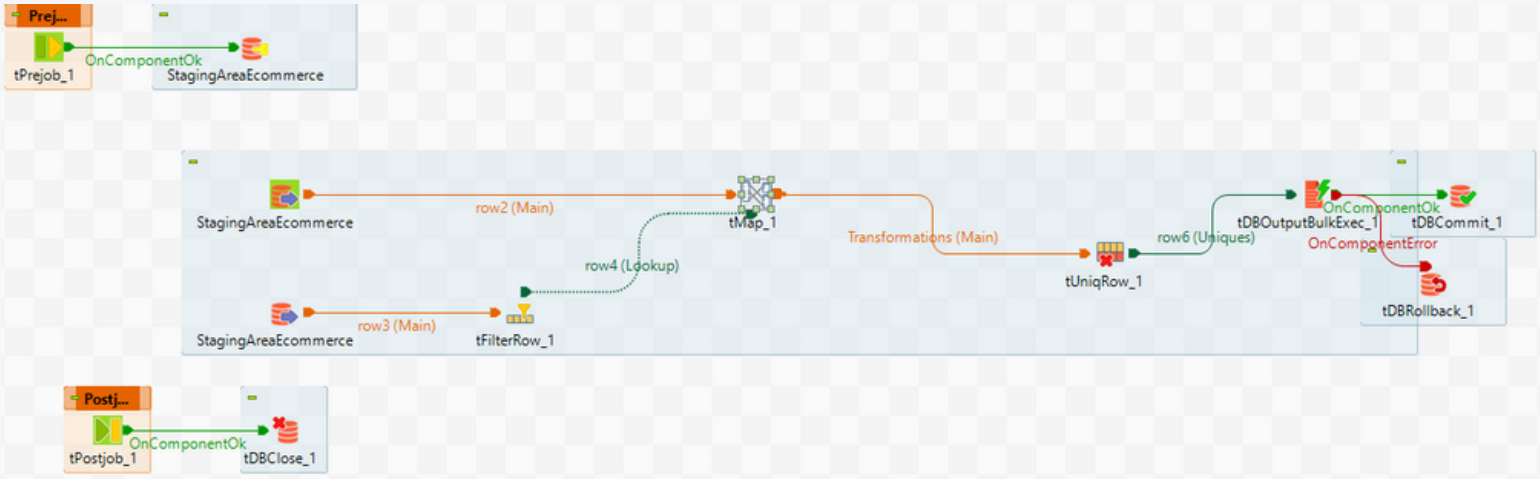
Transformation des Données : Le job "Transform" a été conçu pour traiter les données chargées dans la table "StAEcommerce". Diverses transformations ont été appliquées à l'aide du composant TMap pour garantir la qualité et la cohérence des données. Certaines des transformations comprenaient :

-Conversion des colonnes de chaînes en lettres minuscules pour assurer la cohérence.

-Normalisation des formats de date dans la colonne "Date".

-Vérifications conditionnelles pour les colonnes telles que "ProductName", "ProductCategory" et "ProductPrice" afin de gérer les cas exceptionnels et maintenir la qualité des données.

-Chiffrement des colonnes sensibles conformément aux exigences du RGPD.



Transformation des Données : Le job "Transform" a été conçu pour traiter les données chargées dans la table "StAEcommerce". Diverses transformations ont été appliquées à l'aide du composant TMap pour garantir la qualité et la cohérence des données. Certaines des transformations comprenaient :

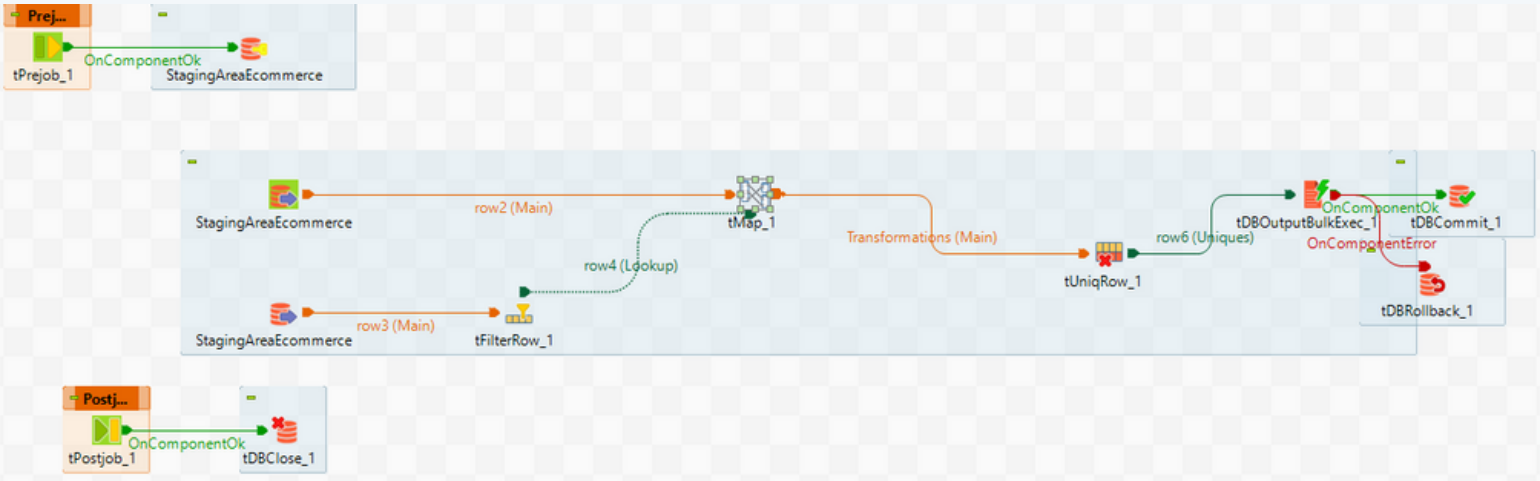
-Conversion des colonnes de chaînes en lettres minuscules pour assurer la cohérence.

-Normalisation des formats de date dans la colonne "Date".

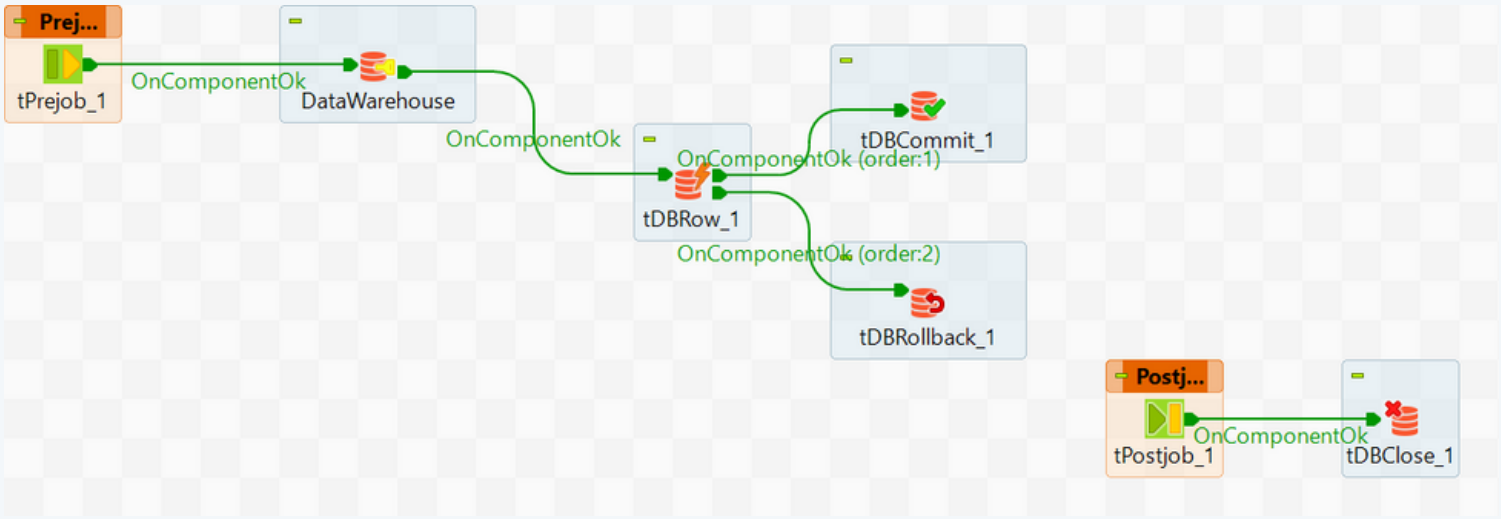
-Vérifications conditionnelles pour les colonnes telles que "ProductName", "ProductCategory" et "ProductPrice" afin de gérer les cas exceptionnels et maintenir la qualité des données.

-Chiffrement des colonnes sensibles conformément aux exigences du RGPD.

Déduplication : Pour supprimer les enregistrements en double, le composant TUniq a été utilisé, garantissant que seules les données uniques et propres étaient conservées. Les données nettoyées résultantes ont été chargées dans une table nommée "transformed_data" dans la base de données "StagingAreaEcommerce".



Suppression des Clés Étrangères : Un job dédié, "DeleteForeignKeys", a été créé en utilisant TDBRow pour supprimer les clés étrangères de l'entrepôt de données. Cette étape a permis de garantir que les données demeuraient propres et que les relations étaient correctement gérées.



Database Microsoft SQL Server Appliquer

Utiliser une connexion existante Liste des composants tDBConnection_1 - DataWarehouse

Schéma Built-in Modifier le schéma

Nom de la table "" Activer les insertions Identity

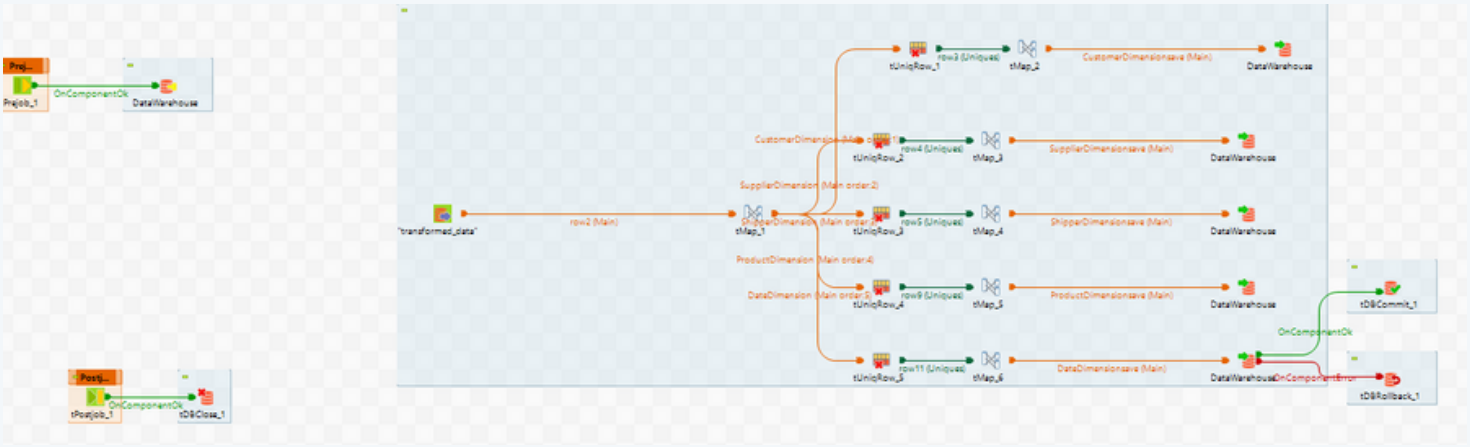
Type de requête Built-in Guess Query

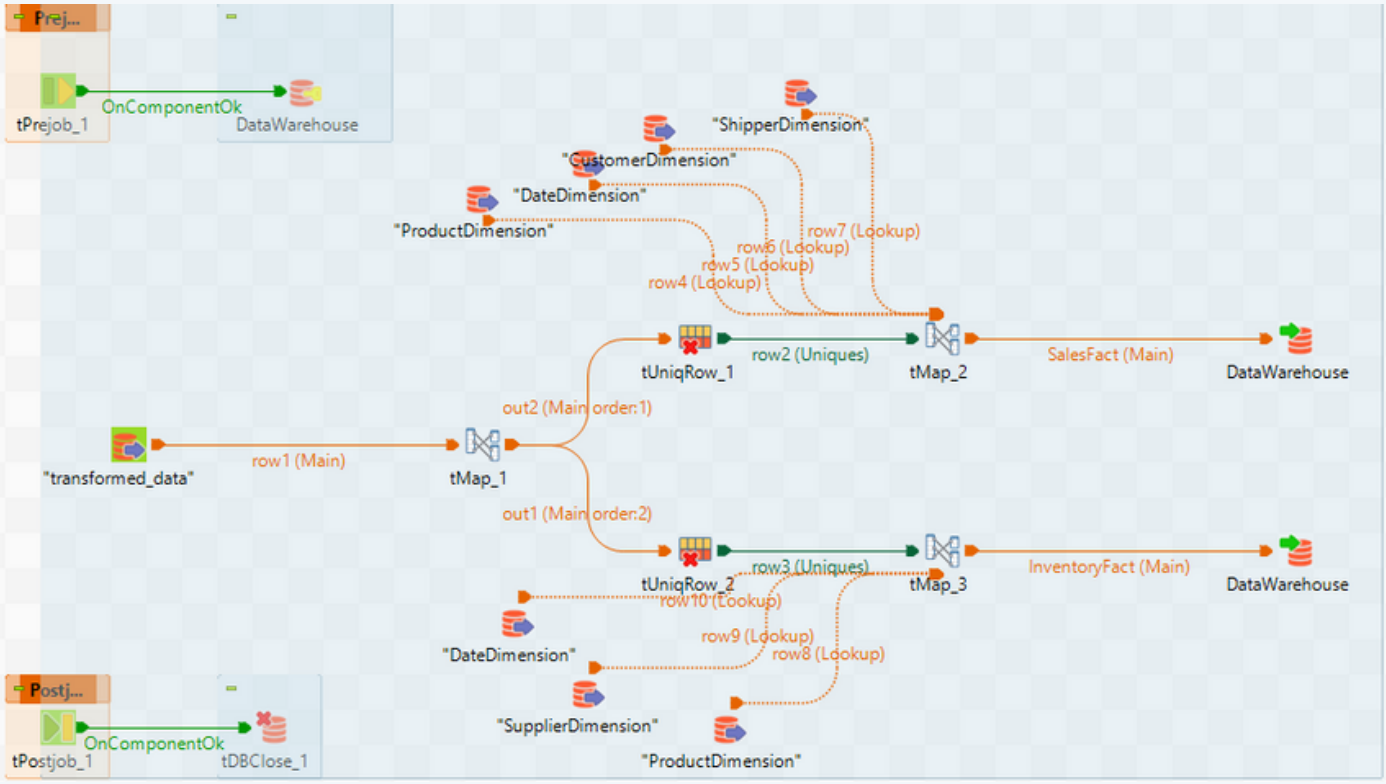
Requête

```
"

-- Drop foreign keys from SalesFact
ALTER TABLE SalesFact
DROP CONSTRAINT IF EXISTS fk_SalesFact_CustomerDimension;

ALTER TABLE SalesFact
DROP CONSTRAINT IF EXISTS fk_SalesFact_ShipperDimension;

ALTER TABLE SalesFact
DROP CONSTRAINT IF EXISTS fk_SalesFact_DateDimension;

ALTER TABLE SalesFact
DROP CONSTRAINT IF EXISTS fk_SalesFact_ProductDimension;

-- Drop foreign keys from InventoryFact
ALTER TABLE InventoryFact
DROP CONSTRAINT IF EXISTS fk_InventoryFact_ProductDimension;

ALTER TABLE InventoryFact
DROP CONSTRAINT IF EXISTS fk_InventoryFact_DateDimension;
```

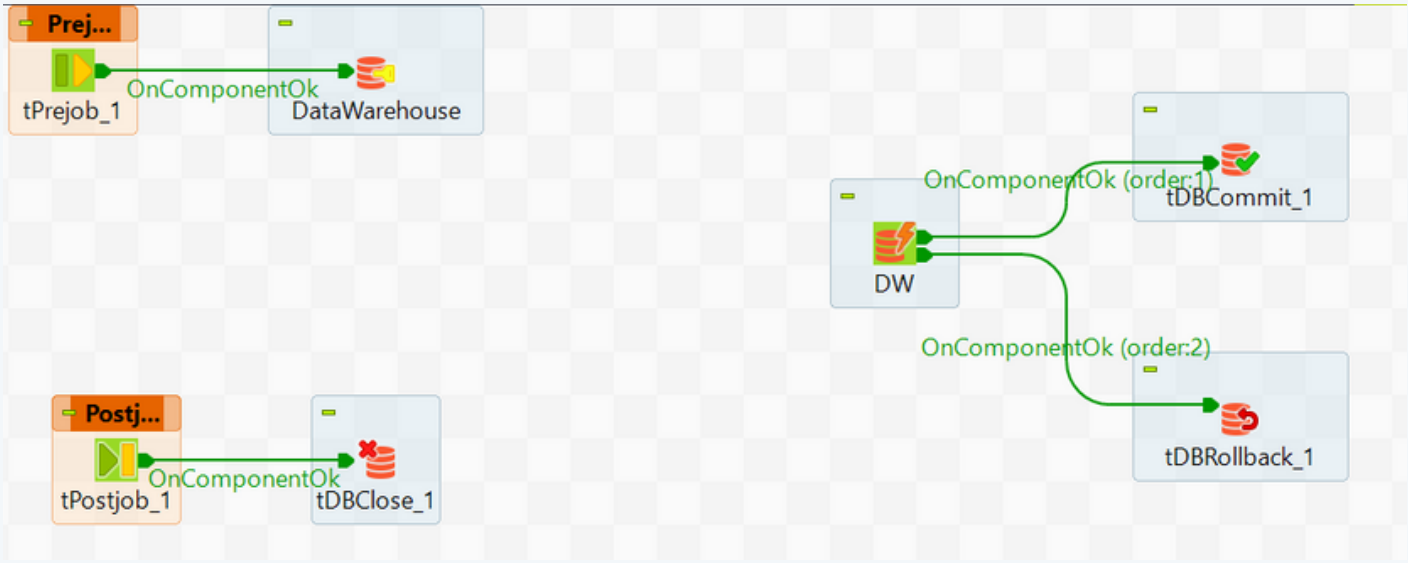
Données Dimensionnelles : Le job "Dimensions" a divisé les "transformed_data", conservé les enregistrements uniques, généré des identifiants uniques et les a insérés dans les tables de dimension respectives de la base de données "Datawarehouse".

Tables de Faits : Un job "Facts" a été développé pour importer les tables de dimension et les insérer dans les tables de faits de l'entrepôt de données. Cette étape était cruciale pour permettre l'analyse multidimensionnelle des données.





Ajout des Clés Étrangères : À l'aide d'un autre job basé sur TDBRow appelé "AddForeignKeys", les clés étrangères ont été réintroduites dans l'entrepôt de données pour restaurer les relations entre différentes tables, garantissant que les données demeuraient liées pour l'analyse.



Database Microsoft SQL Server Appliquer

Utiliser une connexion existante Liste des composants tDBConnection_1 - DataWarehouse

Schéma Built-in Modifier le schéma

Nom de la table "" Activer les insertions Identity

Type de requête Built-in Guess Query

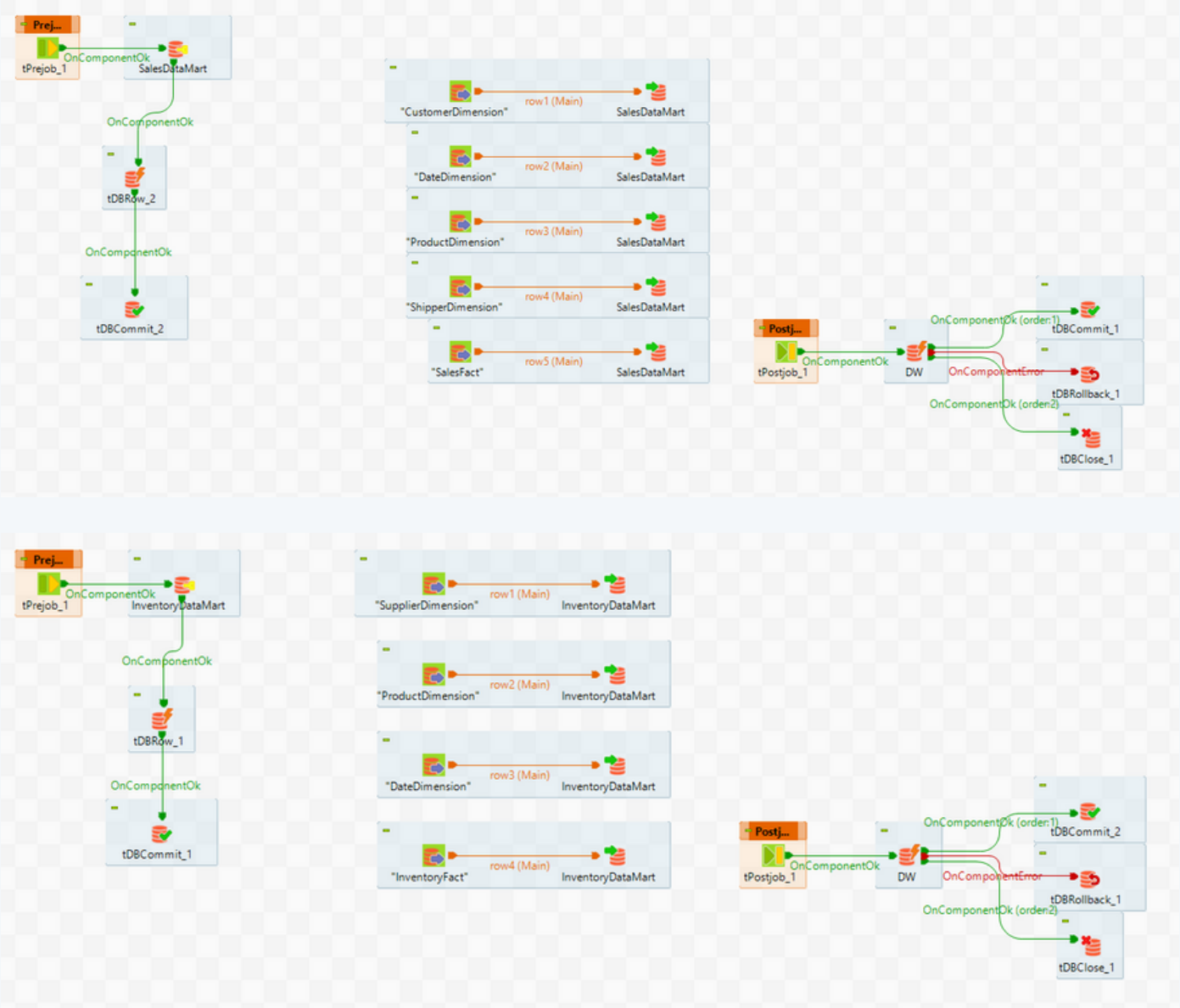
Requête

```
"

ALTER TABLE SalesFact
ADD CONSTRAINT fk_SalesFact_CustomerDimension
FOREIGN KEY (CustomerID)
REFERENCES CustomerDimension(CustomerID)
ON DELETE CASCADE;

ALTER TABLE SalesFact
ADD CONSTRAINT fk_SalesFact_ShipperDimension
FOREIGN KEY (ShipperID)
REFERENCES ShipperDimension(ShipperID)
ON DELETE CASCADE;

ALTER TABLE SalesFact
ADD CONSTRAINT fk_SalesFact_DateDimension
FOREIGN KEY (DateID)
REFERENCES DateDimension(DateID)
ON DELETE CASCADE;

ALTER TABLE SalesFact
```

DATA MARTS PHYSIQUES

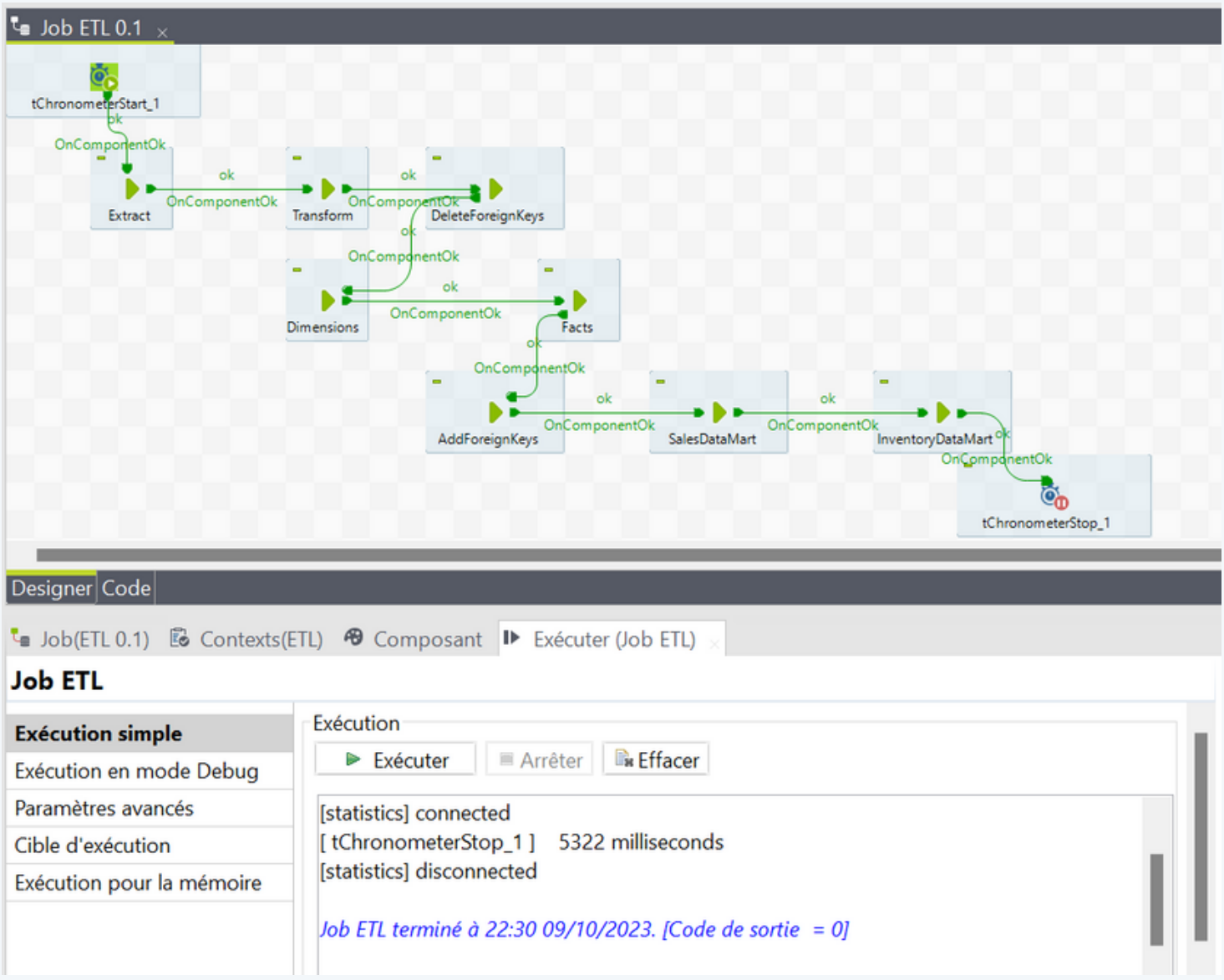
Création de Data Marts :

-Un job nommé "InSalesDataMart" a été créé pour importer les tables de dimension nécessaires et les insérer dans le data mart spécifique aux données de vente.
-De même, un job nommé "InInventoryDataMart" a été développé pour importer les tables de dimension requises et les insérer dans le data mart conçu pour les données d'inventaire.



Job ETL Global : Tous ces jobs individuels ont été intégrés dans un job final appelé "ETL". Ce job principal orchestre l'ensemble du processus ETL, garantissant le déplacement efficace et ordonné des données de la source à la destination, tout en gérant diverses transformations, validations et insertions en cours de route.

Le résultat est un entrepôt de données bien structuré, propre et organisé, avec des data marts associés, désormais prêts pour une analyse approfondie. Ce processus ETL complet constitue la base des capacités de gestion et d'analyse des données de ce projet.



ANALYTIQUE AVEC POWER BI

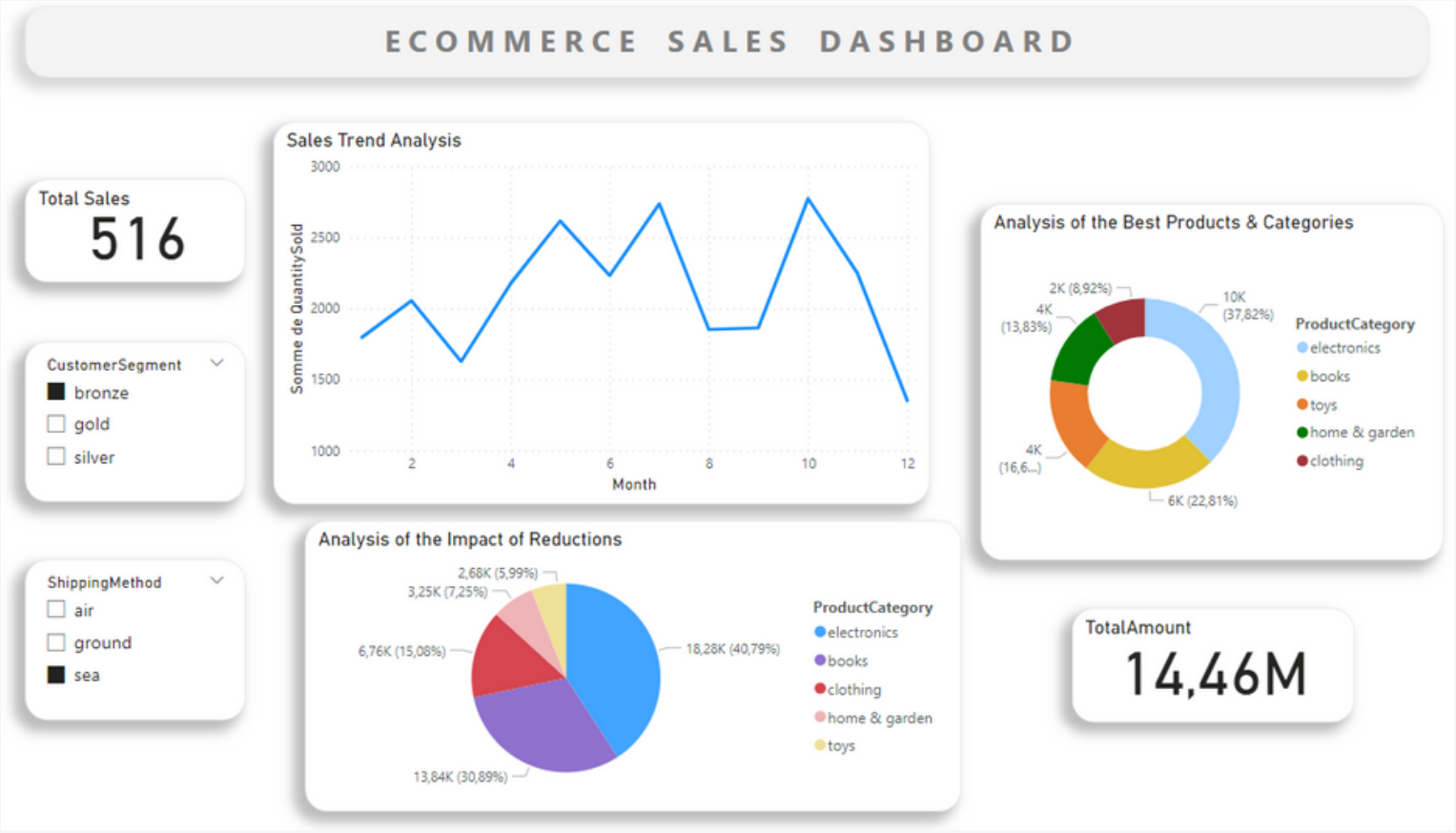
Dans la phase d'analytique avec Power BI, une série d'analyses détaillées a été menée pour exploiter pleinement les données stockées dans les data marts. Cela inclut une analyse approfondie du Data Mart des Ventes, où plusieurs aspects clés ont été explorés. Parmi ces analyses, nous avons réalisé une évaluation des tendances des ventes pour identifier les variations saisonnières ou les pics de demande. De plus, une analyse des meilleurs produits et catégories a été effectuée pour déterminer quels produits et catégories génèrent les revenus les plus importants. La segmentation des clients a également été un élément central de l'analyse, permettant de mieux comprendre les différents profils de clients et d'adapter les stratégies de marketing en conséquence. En outre, une évaluation de l'impact des réductions a été entreprise pour comprendre comment les promotions et les remises influencent les ventes.

De plus, une analyse de la performance des transporteurs a été réalisée pour évaluer l'efficacité des partenaires logistiques dans la livraison des produits aux clients. Cela permet d'identifier les domaines nécessitant des améliorations.

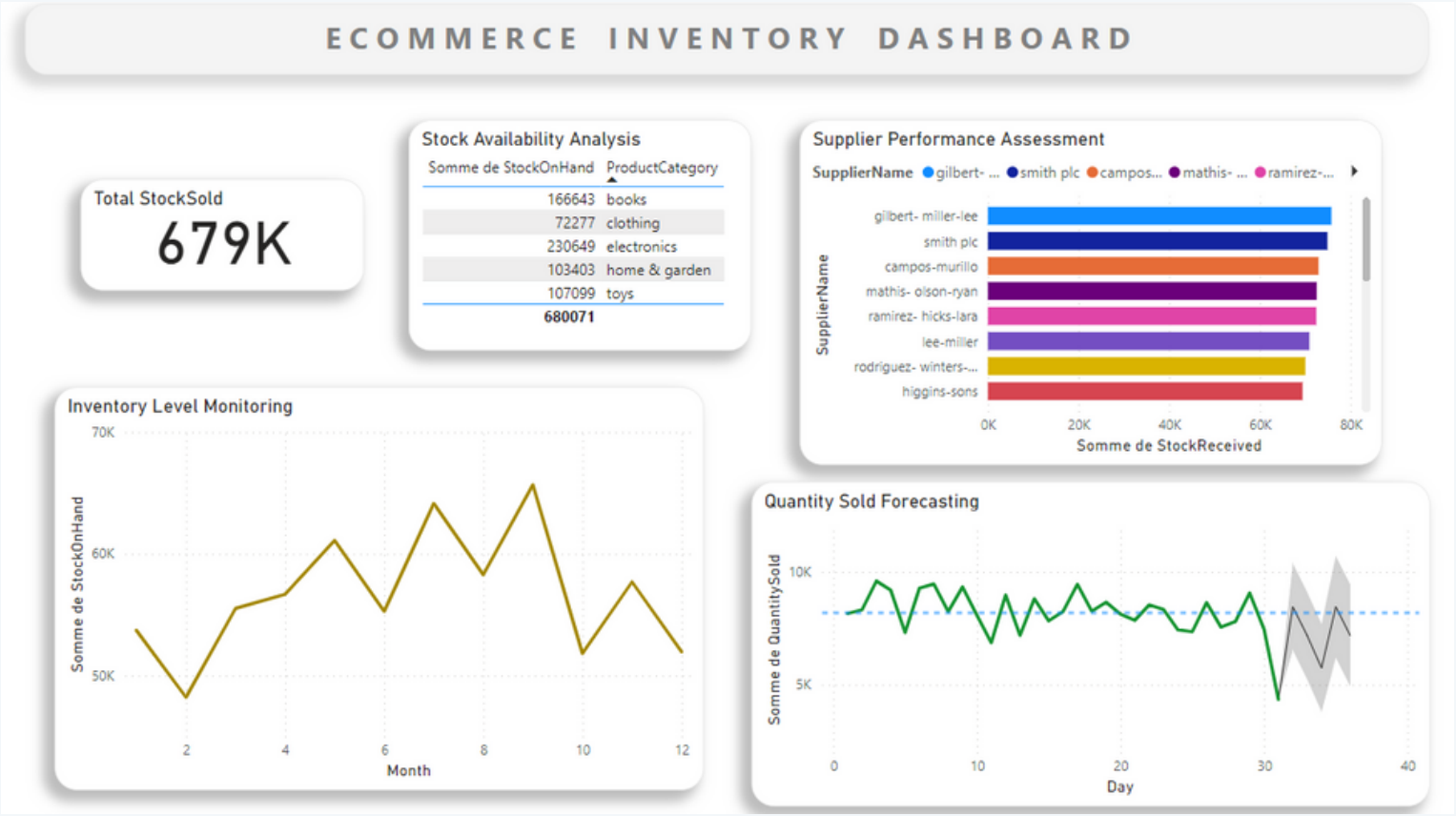
En parallèle, l'analyse du Data Mart de l'Inventaire a inclus des analyses visant à surveiller en temps réel les niveaux d'inventaire, à évaluer la disponibilité des stocks et à anticiper la demande future de produits. Cela garantit une gestion efficace des stocks et permet de mieux répondre à la demande des clients.

Power BI a été l'outil essentiel pour créer des visuels pertinents, tels que des graphiques en ligne, des graphiques en barres, des camemberts, des nuages de points, des tableaux et des cartes, pour représenter visuellement ces analyses complexes. Ces insights obtenus grâce à l'analytique sont essentiels pour prendre des décisions éclairées et améliorer les opérations de l'entreprise.

Sales Dashboard :



Inventory Dashboard :



OPTIMISATION

Dans cette étape d'optimisation, nous avons mis en place des stratégies visant à améliorer les performances et l'efficacité de notre entrepôt de données. Pour ce faire, nous avons utilisé des techniques d'indexation et de partitionnement.

Tout d'abord, nous avons créé des index non cluster sur plusieurs colonnes fréquemment interrogées, notamment sur les tables de dimensions, telles que DateDimension, ProductDimension, et SupplierDimension. Ces index permettent d'accélérer les requêtes et d'optimiser la recherche de données. En outre, nous avons mis en place une stratégie de partitionnement en fonction de la date, en utilisant la fonction de partitionnement SalesDatePartitionFunction. Cette stratégie nous a permis de diviser notre table de faits, SalesFact, en partitions basées sur la date. Les données ont été réparties dans différentes partitions, telles que celles pour l'année 2021, 2022 et 2023, ce qui facilite la gestion et l'accès aux données historiques.

L'ajout de fichiers de groupe de fichiers, tels que FG_sales_Archive, FG_sales_2021, FG_sales_2022, et FG_sales_2023, a été essentiel pour stocker les partitions. Chaque partition a été associée à un groupe de fichiers spécifique pour une meilleure gestion de l'espace de stockage.

De plus, nous avons optimisé les requêtes en créant des vues telles que SalesFactWithDateBase, qui joignent les données de la table de faits avec la table de dimension DateDimension. Ensuite, nous avons créé une nouvelle table de faits partitionnée, SalesFactPartitioned, qui a inclus la colonne SalesDate dans la clé primaire. Cette table optimisée nous permet d'accéder plus rapidement aux données en fonction de la date.

En fin de compte, ces mesures d'optimisation ont amélioré les performances de notre entrepôt de données, rendant l'accès aux informations plus rapide et plus efficace, ce qui est essentiel pour prendre des décisions éclairées.

```sql
use WareahouseEcommerce

-- Create non-clustered indexes on columns
CREATE NONCLUSTERED INDEX IX_Date ON DateDimension (Date);
CREATE NONCLUSTERED INDEX IX_ProductName ON ProductDimension (ProductName);
CREATE NONCLUSTERED INDEX IX_ProductSubCategorie ON ProductDimension (ProductSubCategory);
CREATE NONCLUSTERED INDEX IX_ProductCategorie ON ProductDimension (ProductCategory);
CREATE NONCLUSTERED INDEX IX_SupplierName ON SupplierDimension (SupplierName);
CREATE NONCLUSTERED INDEX IX_SupplierLocation ON SupplierDimension (SupplierLocation);
```

```sql
use WareahouseEcommerce

CREATE PARTITION FUNCTION SalesDatePartitionFunction (DATE)
AS RANGE LEFT FOR VALUES ('2021-01-01', '2022-01-01', '2023-01-01');
ALTER DATABASE WareahouseEcommerce ADD FILEGROUP [FG_sales_Archive]
GO
ALTER DATABASE WareahouseEcommerce ADD FILEGROUP [FG_sales_2021]
GO
ALTER DATABASE WareahouseEcommerce ADD FILEGROUP [FG_sales_2022]
GO
ALTER DATABASE WareahouseEcommerce ADD FILEGROUP [FG_sales_2023]
GO
-----------------------------------
ALTER DATABASE WareahouseEcommerce 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 WareahouseEcommerce 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 WareahouseEcommerce 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 WareahouseEcommerce 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];
```

```sql
CREATE PARTITION SCHEME SalesPartitionScheme
AS PARTITION SalesDatePartitionFunction
TO ([Primary], [FG_sales_2021], [FG_sales_2022], [FG_sales_2023]);

-----------------------------


CREATE VIEW SalesFactWithDateBase
AS
SELECT
    sf.SaleID,
    sf.DateID,
    sf.ProductID,
    sf.CustomerID,
    sf.ShipperID,
    sf.QuantitySold,
    sf.TotalAmount,
    sf.DiscountAmount,
    sf.NetAmount,
    dd.Date AS SalesDate
FROM
    SalesFact sf
JOIN
    DateDimension dd ON sf.DateID = dd.DateID;
```

```sql
-- Create a new partitioned fact table
CREATE TABLE SalesFactPartitioned
(
    SalesID INT,
    DateID INT,
    ProductID INT,
    CustomerID INT,
    ShipperID INT,
    QuantitySold INT,
    TotalAmount DECIMAL(10, 2),
    DiscountAmount DECIMAL(10, 2),
    NetAmount DECIMAL(10, 2),
    SalesDate DATE,
    PRIMARY KEY (SalesID, SalesDate)  -- Include SalesDate in the primary key
)
ON SalesPartitionScheme (SalesDate);



INSERT INTO SalesFactPartitioned (SalesID, DateID, ProductID, CustomerID, ShipperID, QuantitySold, NetAmount, TotalAmount, DiscountAmount, SalesDat
SELECT SaleID, DateID, ProductID, CustomerID, ShipperID, QuantitySold, NetAmount, TotalAmount, DiscountAmount, SalesDate
FROM SalesFactWithDateBase;
```

VALIDER LA LOGIQUE DE TRANSFORMATION

Dans cette phase de validation de la logique de transformation, nous avons effectué plusieurs requêtes et un test pour nous assurer que les données transformées et stockées dans notre entrepôt de données sont correctes et cohérentes.

Tout d'abord, nous avons exécuté une requête pour sélectionner des données spécifiques de la table SalesFactPartitioned. Cette requête nous permet de récupérer des informations sur les ventes qui ont eu lieu entre janvier 2022 et janvier 2023. Cela nous permet de valider que les données stockées dans notre entrepôt correspondent à la plage de dates que nous attendions.

Ensuite, nous avons réalisé une autre requête pour obtenir des données agrégées à partir de la table SalesFactPartitioned. Cette requête nous permet de calculer le total des quantités vendues et le chiffre d'affaires total pour les produits de la catégorie "Électronique" au cours des années 2022 et 2023. Cette agrégation nous permet de vérifier si les calculs sont corrects et cohérents avec nos attentes.

Enfin, nous avons mis en place un test de procédure stockée appelé "CategoryAndNameTest" pour vérifier la qualité de nos données. Ce test vérifie s'il existe des enregistrements dans la table ProductDimension avec des noms de produits non valides (comme 'NonExistentProduct') ou des catégories de produits non valides (comme 'InvalidCategory'). Le test s'assure que ces cas d'invalidité ne sont pas présents dans nos données, ce qui renforce la qualité de notre entrepôt de données.

Ces requêtes et ce test sont essentiels pour s'assurer que la logique de transformation que nous avons appliquée est correcte et que les données sont conformes à nos attentes, garantissant ainsi la fiabilité de notre entrepôt de données pour les futures analyses et rapports.

Optimisation-Part...-1US3GU3J\Youcode)*

```sql
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;
```

79 %

Results | Messages

	partition_number	file_group	row_count
1	1	PRIMARY	0
2	2	FG_sales_2021	646
3	3	FG_sales_2022	2436
4	4	FG_sales_2023	1918

MAX: 1 MIN: 1 AVG: 1 SUM: 1 COUNT: 1 DISTINCT: 1

Optimisation-Part...-1US3GU3J\Youcode)*

```sql
-- Select aggregated data from the SalesFactPartitioned table
SELECT
    dd.Year,
    pd.ProductCategory,
    SUM(sf.QuantitySold) AS TotalQuantitySold,
    SUM(sf.TotalAmount) AS TotalSalesAmount
FROM
    SalesFactPartitioned sf
JOIN
    DateDimension dd ON sf.DateID = dd.DateID
JOIN
    ProductDimension pd ON sf.ProductID = pd.ProductID
WHERE
    dd.Year IN (2022, 2023)
    AND pd.ProductCategory = 'Electronics'
GROUP BY
    dd.Year, pd.ProductCategory;
```

79 %

Results | Messages

	Year	ProductCategory	TotalQuantitySold	TotalSalesAmount
1	2022	electronics	43448	26810269.99
2	2023	electronics	33141	21067208.59

SQLQuery1.sql - L...1US3GU3J\Youcode)* | Test List Manager | UnitTesting.sql -...P-1US3GU3J\Youcode)

```sql
-- Comments here are associated with the test.
-- For test case examples, see: http://tsqlt.org/user-guide/tsqlt-tutorial/
ALTER PROCEDURE Product.[test CategoryAndNameTest]
AS
BEGIN

    -- Act: Query the table for invalid product names and categories
    DECLARE @invalidProductNames INT;
    DECLARE @invalidProductCategories INT;

    SELECT @invalidProductNames = COUNT(*)
    FROM [WareahouseEcommerce].[dbo].[ProductDimension] pd
    WHERE [ProductName] = 'NonExistentProduct';

    SELECT @invalidProductCategories = COUNT(*)
    FROM [WareahouseEcommerce].[dbo].[ProductDimension] pd
    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;
```

78 %

Connected. (1/1) | LAPTOP-1US3GU3J\SQLEXPRESS ... | LAPTOP-1US3GU3J\Youcod... | WareahouseEcommerce | 00:00:00 | 0 rows

Test Results

Status	Test Name	Class Name	Error Message	Execution Time
Succeeded	CategoryAndNameTest	Product		0 ms

AUTORISATION

Dans la phase d'autorisation, nous avons établi des règles de sécurité pour garantir que seuls les utilisateurs autorisés ont accès aux données de l'entrepôt. Nous avons créé des logins et des utilisateurs, associé ces utilisateurs à des rôles pertinents, et attribué des autorisations spécifiques. Par exemple, le Data Engineer a des autorisations pour gérer les données de vente, tandis que le Data Analyst a des autorisations limitées à l'analyse des données. Ces mesures de sécurité garantissent que seules les personnes appropriées ont accès aux données de l'entrepôt, tout en maintenant l'intégrité des données.

```
-- Use the target database
USE WareahouseEcommerce;


-- Create server-level logins with passwords
CREATE LOGIN DataEngineerLogin WITH PASSWORD = 'DataEngineer2004';
CREATE LOGIN DataAnalystLogin WITH PASSWORD = 'DataAnalyst2004';


-- Create database users
CREATE USER DataEngineerUser FOR LOGIN DataEngineerLogin;
CREATE USER DataAnalystUser FOR LOGIN DataAnalystLogin;

-- Create database roles for Data Engineer and Data Analyst
CREATE ROLE DataEngineerRole;
CREATE ROLE DataAnalystRole;

-- Add users to their respective roles
ALTER ROLE DataEngineerRole ADD MEMBER DataEngineerUser;
ALTER ROLE DataAnalystRole ADD MEMBER DataAnalystUser;
```

```
-- Grant permissions to DataEngineerRole
GRANT SELECT ON SalesFact TO DataEngineerRole;
GRANT INSERT ON SalesFact TO DataEngineerRole;
GRANT UPDATE ON SalesFact TO DataEngineerRole;
GRANT DELETE ON SalesFact TO DataEngineerRole;

GRANT SELECT ON SupplierDimension TO DataEngineerRole;
GRANT SELECT ON ProductDimension TO DataEngineerRole;
GRANT SELECT ON ShipperDimension TO DataEngineerRole;
GRANT SELECT ON DateDimension TO DataEngineerRole;



-- Grant permissions to DataAnalystRole
GRANT SELECT ON SalesFact TO DataAnalystRole;
GRANT SELECT ON SupplierDimension TO DataAnalystRole;
GRANT SELECT ON ProductDimension TO DataAnalystRole;
GRANT SELECT ON ShipperDimension TO DataAnalystRole;
GRANT SELECT ON DateDimension TO DataAnalystRole;
```

CONCLUSION

En conclusion, ce projet a permis la mise en place et l'optimisation d'un entrepôt de données pour une plateforme e-commerce, en utilisant des outils tels que Talend, SQL Server, et Power BI. Nous avons suivi une approche méthodique en commençant par l'exploration des données avec Python pour comprendre la structure des données, puis en appliquant des politiques RGPD pour protéger les données sensibles. La modélisation des données a été réalisée en utilisant un schéma de constellation, ce qui a facilité la gestion des tables de dimensions et de faits.

Le processus ETL avec Talend a été optimisé pour garantir l'efficacité et la qualité des données. Nous avons créé des Data Marts physiques pour les ventes et l'inventaire, permettant des analyses approfondies avec Power BI. Les analyses ont inclus la tendance des ventes, l'analyse des produits, la segmentation des clients, et bien d'autres.

L'optimisation de l'entrepôt de données a été réalisée grâce à l'indexation et au partitionnement, améliorant les performances des requêtes. La validation de la logique de transformation a été effectuée avec succès, garantissant la qualité des données transformées.

Enfin, des mesures de sécurité et d'autorisation ont été mises en place pour protéger les données, en attribuant des rôles et en définissant des autorisations spécifiques. Ce projet a permis de répondre aux besoins de gestion des données de la plateforme e-commerce de manière efficace et sécurisée, offrant des informations précieuses pour prendre des décisions éclairées.