



Rapport de Projet :

Business Intelligence avec Power BI

Création d'un ETL et d'un entrepôt de données

Réalisé par : MALEK Chakib Walid

Année universitaire 2025-2026

Table des matières

| | | |
|----------|--|-----------|
| 1 | Introduction à la Business Intelligence | 6 |
| 1.1 | Définition | 6 |
| 1.2 | Composantes d'une solution BI | 6 |
| 1.3 | Objectifs du projet | 7 |
| 2 | Présentation de Power BI | 8 |
| 2.1 | Qu'est-ce que Power BI? | 8 |
| 2.1.1 | Composantes de Power BI | 8 |
| 2.2 | Fonctionnalités principales | 8 |
| 2.2.1 | Power Query - L'outil ETL | 8 |
| 2.2.2 | Modélisation des données | 8 |
| 2.2.3 | Visualisations | 8 |
| 2.3 | Avantages de Power BI | 9 |
| 3 | Installation de Power BI Desktop | 10 |
| 3.1 | Prérequis | 10 |
| 3.2 | Téléchargement et Installation | 10 |
| 3.2.1 | Téléchargement | 10 |
| 3.2.2 | Installation | 10 |
| 4 | Exploration de l'outil Power BI | 11 |
| 4.1 | Les trois vues principales | 11 |
| 4.1.1 | Vue Rapport | 11 |
| 4.1.2 | Vue Données | 11 |
| 4.1.3 | Vue Modèle | 11 |
| 4.2 | Les panneaux principaux | 11 |
| 4.2.1 | Panneau Champs | 11 |
| 4.2.2 | Panneau Visualisations | 12 |
| 4.2.3 | Panneau Filtres | 12 |
| 4.3 | Power Query Editor | 12 |
| 4.3.1 | Accès à Power Query | 12 |
| 4.3.2 | Fonctionnalités principales | 12 |
| 4.4 | DAX - Langage de calcul | 12 |
| 4.4.1 | Qu'est-ce que DAX? | 12 |
| 4.4.2 | Types de calculs | 12 |
| 4.4.3 | Exemples de mesures DAX | 12 |
| 4.5 | Visualisations dans Power BI | 13 |
| 4.5.1 | Visuels natifs de Power BI | 13 |
| 4.5.2 | Visualisations avec Python | 13 |

| | |
|--|-----------|
| 5 Comparaison entre Talend et Power BI | 14 |
| 5.1 Positionnement des outils | 14 |
| 5.2 Principales différences | 14 |
| 5.2.1 Interface | 14 |
| 5.2.2 Comparaison synthétique | 15 |
| 5.3 Choix pour ce projet | 15 |
| 6 Sélection des données | 16 |
| 6.1 Présentation de la base Northwind | 16 |
| 6.1.1 Description | 16 |
| 6.1.2 Schéma de la base | 16 |
| 6.2 Tables sélectionnées depuis SQL Server | 17 |
| 6.2.1 Table Customers | 17 |
| 6.2.2 Table Employees | 17 |
| 6.2.3 Table EmployeeTerritories | 17 |
| 6.2.4 Table Territories | 17 |
| 6.2.5 Table Orders | 18 |
| 6.3 Tables sélectionnées depuis Microsoft Access | 18 |
| 6.3.1 Nécessité de l'export vers Excel | 18 |
| 6.3.2 Tables exportées | 19 |
| 6.4 Nettoyage des fichiers Excel | 19 |
| 6.4.1 Colonnes à supprimer | 19 |
| 6.4.2 Standardisation des données | 20 |
| 6.5 Récapitulatif des sources | 20 |
| 7 Déploiement des tables dans Power BI | 21 |
| 7.1 Introduction | 21 |
| 7.2 Connexion à SQL Server | 21 |
| 7.2.1 Procédure de connexion | 21 |
| 7.2.2 Sélection des tables SQL Server | 22 |
| 7.3 Importation des fichiers Excel | 22 |
| 7.3.1 Procédure d'importation | 22 |
| 7.4 Vérification du chargement | 23 |
| 7.4.1 Panneau Champs | 23 |
| 7.4.2 Vérification des données | 23 |
| 7.5 Résumé du déploiement | 24 |
| 8 Création et remplissage des dimensions | 25 |
| 8.1 Introduction | 25 |
| 8.2 Dimension Employee (DimEmployee) | 25 |
| 8.2.1 Objectif | 25 |
| 8.2.2 Accès à Power Query | 25 |
| 8.2.3 Script Power Query M | 25 |
| 8.2.4 Explication des étapes | 29 |
| 8.2.5 Structure finale de DimEmployee | 30 |
| 8.2.6 Chargement de la dimension | 30 |
| 8.3 Dimension Client (DimClient) | 31 |
| 8.3.1 Objectif | 31 |
| 8.3.2 Script Power Query M | 31 |

| | | |
|-----------|---|-----------|
| 8.3.3 | Explication des étapes | 33 |
| 8.3.4 | Structure finale de DimClient | 34 |
| 8.3.5 | Chargement de la dimension | 34 |
| 8.4 | Dimension Temps (DimTemps) | 34 |
| 8.4.1 | Objectif | 34 |
| 8.4.2 | Script Power Query M | 35 |
| 8.4.3 | Explication des étapes | 37 |
| 8.4.4 | Structure finale de DimTemps | 37 |
| 8.4.5 | Chargement de la dimension | 37 |
| 8.5 | Récapitulatif des dimensions | 38 |
| 8.6 | Vérification finale | 38 |
| 9 | Création de la table de faits et de l'entrepôt | 39 |
| 9.1 | Introduction | 39 |
| 9.2 | Table de faits : TF_Commande | 39 |
| 9.2.1 | Objectif | 39 |
| 9.2.2 | Métriques calculées | 39 |
| 9.2.3 | Script Power Query M | 39 |
| 9.2.4 | Explication détaillée des étapes | 43 |
| 9.2.5 | Structure finale de TF_Commande | 45 |
| 9.2.6 | Chargement de la table de faits | 45 |
| 9.3 | Création du modèle en étoile | 46 |
| 9.3.1 | Établissement des relations | 46 |
| 9.3.2 | Vérification du modèle | 46 |
| 9.4 | Conclusion | 47 |
| 10 | Visualisations et graphiques | 48 |
| 10.1 | Introduction | 48 |
| 10.2 | Visualisations Python | 48 |
| 10.2.1 | Configuration Python dans Power BI | 48 |
| 10.2.2 | Visuel Python 1 : Volume des commandes par mois | 48 |
| 10.2.3 | Visuel Python 2 : Taux de livraison par mois | 51 |
| 10.2.4 | Visuel Python 3 : Top 10 des clients | 54 |
| 10.2.5 | Visuel Python 4 : Top 10 des employés | 57 |
| 10.3 | Visualisations Power BI natives | 61 |
| 10.3.1 | Mesure DAX utilisée | 61 |
| 10.3.2 | Visuel Power BI 1 : Top clients avec répartition | 61 |
| 10.3.3 | Visuel Power BI 2 : Top employés avec répartition | 63 |
| 10.4 | Comparaison Python vs Power BI natif | 64 |
| 10.4.1 | Définitions clés observées | 64 |
| 10.4.2 | Avantage de l'interactivité Power BI | 65 |
| 10.4.3 | Conclusion sur les approches | 65 |
| 10.5 | Conclusion du chapitre | 65 |
| 11 | Conclusion | 66 |
| 11.1 | Objectifs atteints | 66 |
| 11.2 | Apports techniques et compétences développées | 66 |
| 11.2.1 | Power Query et le langage M | 66 |
| 11.2.2 | Modélisation dimensionnelle | 66 |

| | |
|---|----|
| 11.2.3 DAX (Data Analysis Expressions) | 67 |
| 11.2.4 Visualisation hybride | 67 |
| 11.3 Comparaison des approches de visualisation | 67 |
| 11.4 Avantages de Power BI | 67 |
| 11.5 Perspectives et améliorations futures | 67 |
| 11.5.1 Enrichissement du modèle | 67 |
| 11.5.2 Calculs avancés | 68 |
| 11.5.3 Optimisation des performances | 68 |
| 11.5.4 Automatisation | 68 |
| 11.5.5 Publication et partage | 68 |
| 11.5.6 Sécurité | 68 |
| 11.6 Conclusion finale | 68 |

Table des figures

| | | |
|------|---|----|
| 1.1 | Architecture d'une solution Business Intelligence | 6 |
| 5.1 | Interface Talend | 14 |
| 5.2 | Power Query dans Power BI | 15 |
| 6.1 | Schéma de la base de données Northwind | 16 |
| 6.2 | Export des tables depuis Access | 18 |
| 6.3 | Nettoyage des colonnes dans Excel AVANT | 19 |
| 6.4 | Nettoyage des colonnes dans Excel APRES | 20 |
| 7.1 | Connexion à SQL Server | 21 |
| 7.2 | Sélection des tables depuis SQL Server | 22 |
| 7.3 | Importation des fichiers Excel dans Power BI | 23 |
| 7.4 | Vérification des données chargées | 24 |
| 8.1 | Dimension DimEmployee créée | 30 |
| 8.2 | Dimension DimClient créée | 34 |
| 8.3 | Dimension DimTemps créée | 38 |
| 9.1 | Table de faits TF_Commande créée | 45 |
| 9.2 | Modèle en étoile complet | 46 |
| 10.1 | Volume des commandes par mois | 51 |
| 10.2 | Taux de livraison par mois | 54 |
| 10.3 | Top 10 des clients | 57 |
| 10.4 | Top 10 des employés | 61 |
| 10.5 | Top clients avec répartition - Power BI natif | 63 |
| 10.6 | Top employés avec répartition - Power BI natif | 64 |

Chapitre 1

Introduction à la Business Intelligence

1.1 Définition

La Business Intelligence (BI) est l'ensemble des technologies et processus permettant de transformer les données brutes en informations exploitables pour la prise de décision. Elle combine la collecte, l'analyse et la visualisation de données pour aider les organisations à mieux comprendre leur activité et à optimiser leurs performances.

1.2 Composantes d'une solution BI

Une solution BI complète comprend quatre éléments principaux :

- **Sources de données** : Bases de données, fichiers Excel, APIs, etc.
- **ETL (Extract, Transform, Load)** : Processus d'extraction, transformation et chargement des données
- **Entrepôt de données (Data Warehouse)** : Stockage centralisé des données pour l'analyse
- **Visualisation** : Tableaux de bord et rapports interactifs

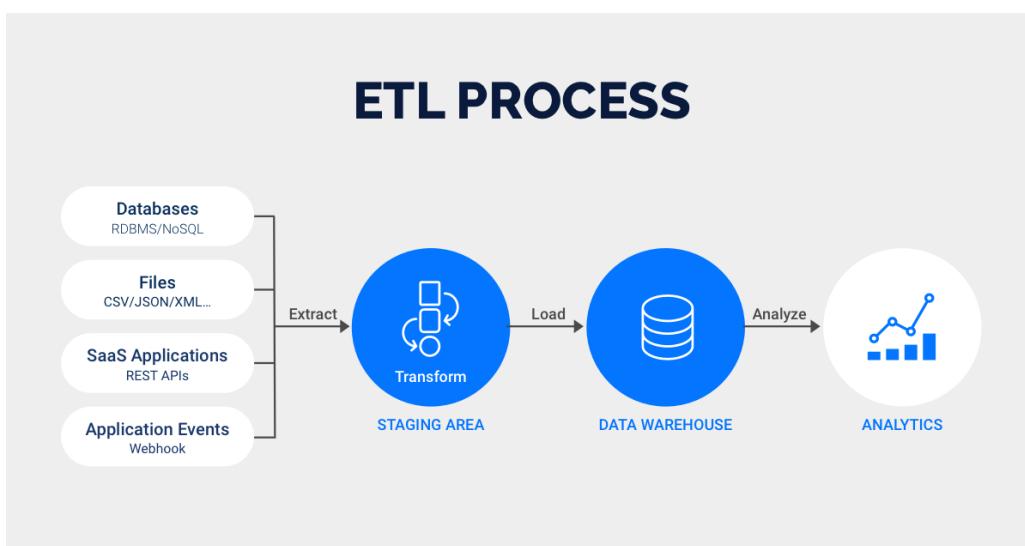


FIGURE 1.1 – Architecture d'une solution Business Intelligence

1.3 Objectifs du projet

Ce projet vise à :

- Mettre en place un processus ETL avec Power BI
- Créer un entrepôt de données à partir de deux bases : SQL Server et Microsoft Access
- Concevoir des visualisations pour l'analyse des données

Chapitre 2

Présentation de Power BI

2.1 Qu'est-ce que Power BI ?

Power BI est une plateforme de Business Intelligence développée par Microsoft. C'est un outil complet qui permet de se connecter à diverses sources de données, de les transformer, de les modéliser et de créer des visualisations interactives.

2.1.1 Composantes de Power BI

- **Power BI Desktop** : Application Windows gratuite pour créer des rapports
- **Power BI Service** : Plateforme cloud pour partager et collaborer
- **Power BI Mobile** : Applications pour consulter les rapports sur mobile

2.2 Fonctionnalités principales

2.2.1 Power Query - L'outil ETL

Power Query est l'éditeur de transformation des données intégré à Power BI. Il permet de :

- Se connecter à plus de 150 sources de données
- Nettoyer et transformer les données
- Fusionner et combiner des tables
- Automatiser les étapes de transformation

2.2.2 Modélisation des données

Power BI permet de créer un modèle relationnel entre les tables avec :

- Création de relations (1 :N, N :1)
- Définition de hiérarchies
- Calculs avec DAX (Data Analysis Expressions)

2.2.3 Visualisations

Large choix de visuels pour créer des tableaux de bord interactifs :

- Graphiques (barres, lignes, secteurs, cartes)
- Tableaux et matrices

- Cartes géographiques
- Indicateurs KPI
- Visuels personnalisés (Python, R)

2.3 Avantages de Power BI

- Interface intuitive et facile à prendre en main
- Version Desktop gratuite
- Intégration native avec l'écosystème Microsoft
- Mises à jour mensuelles
- Grande communauté d'utilisateurs
- Performances élevées pour traiter de gros volumes de données

Chapitre 3

Installation de Power BI Desktop

3.1 Prérequis

- Windows 10 ou ultérieur (64 bits)
- 2 Go de RAM minimum (8 Go recommandés)
- 2 Go d'espace disque disponible

3.2 Téléchargement et Installation

3.2.1 Téléchargement

Aller sur <https://powerbi.microsoft.com> et cliquer sur "Télécharger gratuitement".

3.2.2 Installation

1. Exécuter le fichier téléchargé
2. Accepter les conditions d'utilisation
3. Suivre l'assistant d'installation
4. Lancer Power BI Desktop

Power BI Desktop est maintenant installé et prêt à être utilisé !

Chapitre 4

Exploration de l'outil Power BI

4.1 Les trois vues principales

Power BI Desktop offre trois vues pour travailler avec les données :

4.1.1 Vue Rapport

C'est la vue principale pour créer des visualisations. Elle permet de :

- Créer et organiser des visuels (graphiques, tableaux, cartes)
- Appliquer des filtres
- Configurer l'interactivité entre les visuels

4.1.2 Vue Données

Affiche les données sous forme de tableaux. Utile pour :

- Vérifier les données importées
- Créer des colonnes calculées
- Inspecter la qualité des données

4.1.3 Vue Modèle

Permet de gérer les relations entre les tables :

- Visualiser le schéma du modèle de données
- Créer et modifier les relations entre tables
- Définir les cardinalités (1 :N, N :1)

4.2 Les panneaux principaux

4.2.1 Panneau Champs

Situé à droite, il affiche toutes les tables et colonnes du modèle de données. On peut :

- Glisser-déposer les champs vers les visuels
- Créer des mesures et colonnes calculées
- Organiser les champs par table

4.2.2 Panneau Visualisations

Contient :

- La galerie de visuels disponibles (graphiques, tableaux, cartes)
- Les propriétés du visuel sélectionné (axes, valeurs, légendes)
- Les options de formatage (couleurs, titres, étiquettes)

4.2.3 Panneau Filtres

Permet d'appliquer des filtres à trois niveaux :

- **Filtre visuel** : S'applique uniquement au visuel sélectionné
- **Filtre de page** : S'applique à tous les visuels de la page
- **Filtre de rapport** : S'applique à toutes les pages du rapport

4.3 Power Query Editor

4.3.1 Accès à Power Query

Accessible via : **Accueil** → **Transformer les données**

Power Query est l'outil ETL de Power BI qui permet de préparer et nettoyer les données avant de les charger dans le modèle.

4.3.2 Fonctionnalités principales

- **Connexion aux sources** : SQL Server, Excel, Access, CSV, Web, etc.
- **Transformations** : Filtrer, trier, grouper, fusionner, pivoter
- **Nettoyage** : Supprimer doublons, remplacer valeurs, gérer les erreurs
- **Étapes appliquées** : Historique de toutes les transformations

4.4 DAX - Langage de calcul

4.4.1 Qu'est-ce que DAX ?

DAX (Data Analysis Expressions) est le langage de formules de Power BI pour créer des calculs personnalisés.

4.4.2 Types de calculs

- **Mesures** : Calculs dynamiques évalués selon le contexte
- **Colonnes calculées** : Calculs ligne par ligne dans une table
- **Tables calculées** : Création de nouvelles tables

4.4.3 Exemples de mesures DAX

```

1 // Somme des ventes
2 Total Ventes = SUM(Orders[TotalPrice])
3
4 // Nombre de commandes

```

```
5 Nb Commandes = COUNTROWS(Orders)
6
7 // Moyenne des ventes
8 Moyenne Ventes = AVERAGE(Orders[TotalPrice])
```

Listing 4.1 – Exemples de mesures DAX simples

4.5 Visualisations dans Power BI

4.5.1 Visuels natifs de Power BI

Power BI propose une large gamme de visualisations natives :

- Graphiques en barres, colonnes, lignes
- Graphiques en secteurs et en anneau
- Tableaux et matrices
- Cartes géographiques
- Jauges et indicateurs KPI
- Graphiques combinés

4.5.2 Visualisations avec Python

Power BI permet d'intégrer des scripts Python pour créer des visualisations personnalisées. Les visuels Python permettent de créer des graphiques statistiques avancés et des visualisations personnalisées non disponibles nativement dans Power BI.

Chapitre 5

Comparaison entre Talend et Power BI

5.1 Positionnement des outils

- **Talend** : Plateforme ETL spécialisée dans l'intégration de données complexes
- **Power BI** : Solution BI complète incluant ETL (Power Query), modélisation et visualisation

5.2 Principales différences

5.2.1 Interface

Talend : Interface graphique par composants avec flux de données visuels. Génère du code Java.

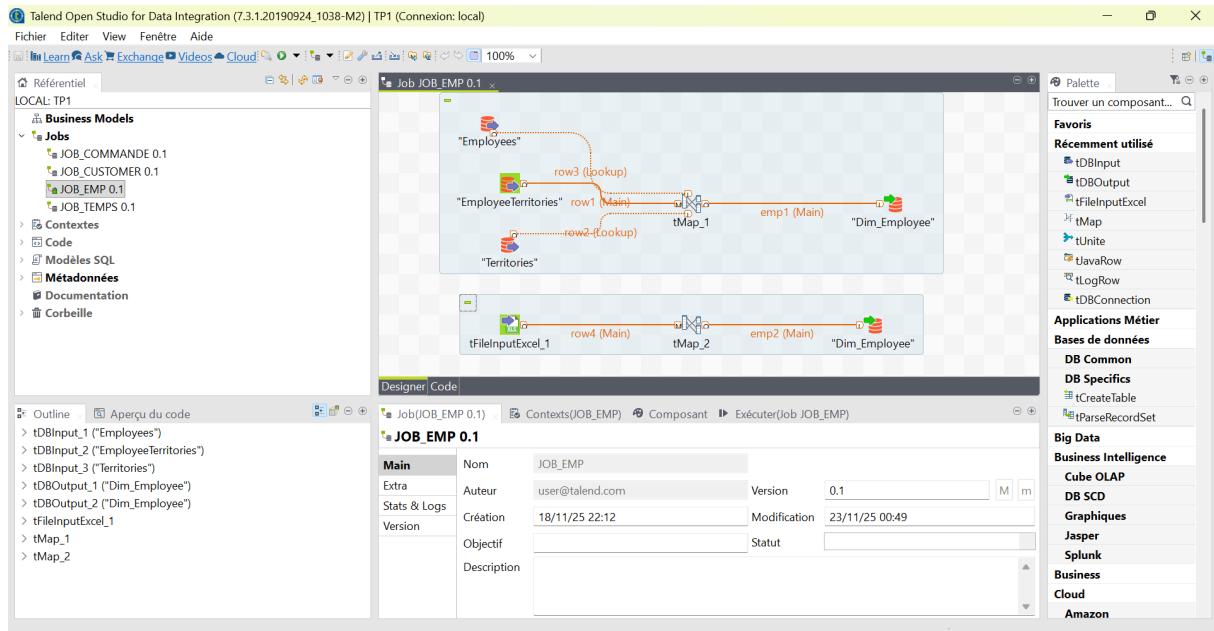


FIGURE 5.1 – Interface Talend

Power BI : Interface moderne et intuitive. Power Query avec transformations par étapes.

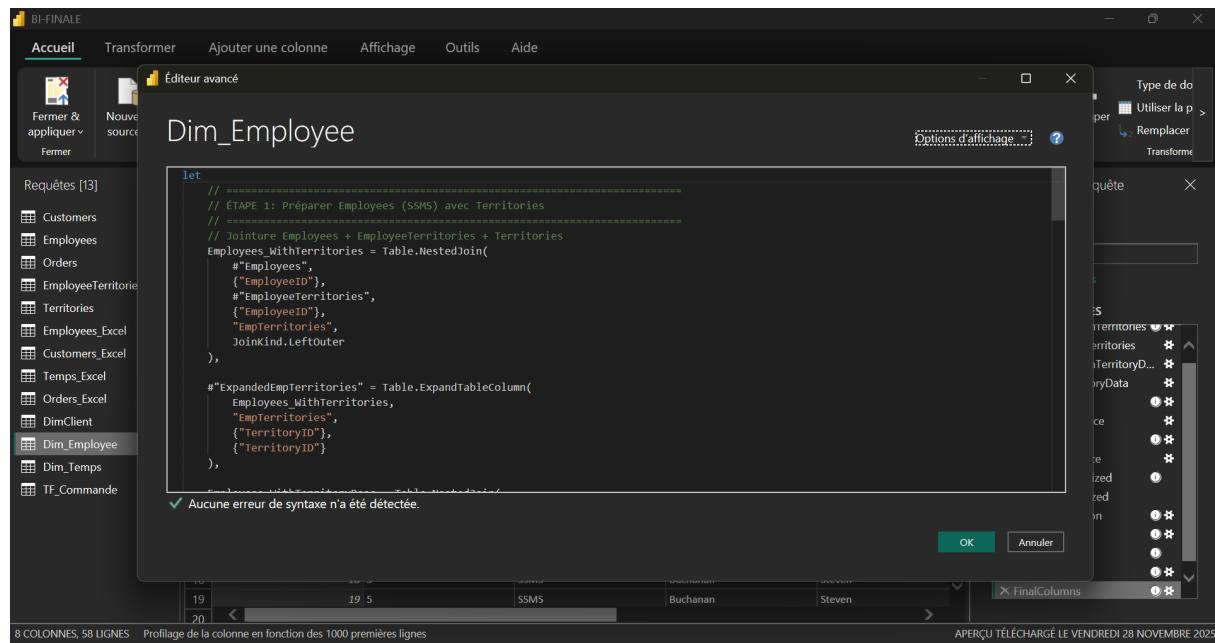


FIGURE 5.2 – Power Query dans Power BI

5.2.2 Comparaison synthétique

| Critère | Talend | Power BI |
|---------------|----------------|-------------------|
| Focus | ETL complexe | BI complète |
| Utilisateurs | Data engineers | Analystes métiers |
| Connecteurs | 900+ | 150+ |
| Volumes | Big Data | Millions lignes |
| Visualisation | Non intégrée | Complète |
| Facilité | Moyenne | Facile |

TABLE 5.1 – Comparaison Talend vs Power BI

5.3 Choix pour ce projet

Nous avons choisi **Power BI** pour ce projet car :

- Solution complète (ETL + visualisation)
- Adapté aux volumes de données (base Northwind)
- Facilité et rapidité de développement
- Excellente intégration avec SQL Server et Access

Chapitre 6

Sélection des données

6.1 Présentation de la base Northwind

6.1.1 Description

Northwind est une base de données exemple de Microsoft représentant une entreprise fictive d'import-export de produits alimentaires. Elle contient des informations complètes sur :

- Les clients et leurs commandes
- Les employés et leurs territoires de vente
- Les produits et les fournisseurs
- Les détails des transactions commerciales

6.1.2 Schéma de la base

La base Northwind comprend plusieurs tables reliées entre elles par des clés étrangères, formant un modèle relationnel complet.

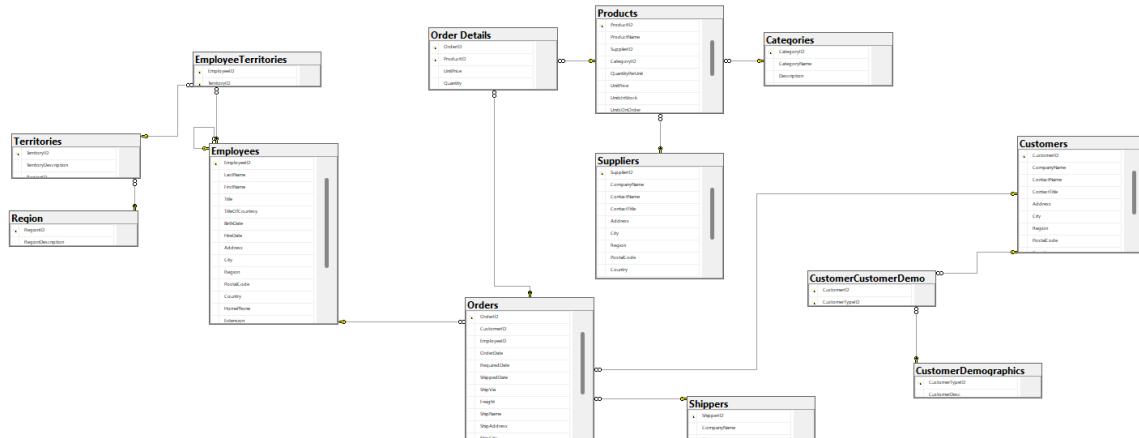


FIGURE 6.1 – Schéma de la base de données Northwind

6.2 Tables sélectionnées depuis SQL Server

Pour notre entrepôt de données, nous avons sélectionné cinq tables depuis SQL Server :

6.2.1 Table Customers

Description : Informations sur les clients de l'entreprise.

- CustomerID : Identifiant unique du client
- CompanyName : Nom de l'entreprise
- ContactName : Nom du contact
- ContactTitle : Fonction du contact
- Address : Adresse de l'entreprise
- City : Ville
- Region : Région
- PostalCode : Code postal
- Country : Pays
- Phone : Numéro de téléphone
- Fax : Fax

Utilisation : Dimension Client pour analyser les ventes par client et par région.

6.2.2 Table Employees

Description : Informations sur les employés.

- EmployeeID : Identifiant unique de l'employé
- LastName : Nom de famille
- FirstName : Prénom
- City : Ville
- Region : Date d'embauche
- BirthDate : Date de naissance

Utilisation : Dimension Employé pour analyser les performances commerciales par vendeur.

6.2.3 Table EmployeeTerritories

Description : Table de liaison entre employés et territoires.

- EmployeeID : Référence à l'employé
- TerritoryID : Référence au territoire

Utilisation : Permet d'associer les employés à leurs zones géographiques de vente.

6.2.4 Table Territories

Description : Territoires de vente de l'entreprise.

- TerritoryID : Identifiant du territoire
- TerritoryDescription : Nom du territoire
- RegionID : Référence à la région

6.2.5 Table Orders

Description : Commandes passées par les clients.

- OrderID : Identifiant unique de la commande
- CustomerID : Référence au client
- EmployeeID : Référence à l'employé
- OrderDate : Date de la commande
- ShippedDate : Date d'expédition

Utilisation : Base pour la table de faits de l'entrepôt.

6.3 Tables sélectionnées depuis Microsoft Access

La deuxième source de données provient d'une base Microsoft Access contenant des informations complémentaires.

6.3.1 Nécessité de l'export vers Excel

Microsoft Access ne se connecte pas aussi facilement que SQL Server à Power BI. La solution recommandée est d'exporter les tables vers Excel avant l'import.

Procédure d'export

1. Ouvrir la base Access Northwind
2. Pour chaque table à exporter :
 - Sélectionner la table
 - Données externes → Excel
 - Choisir le format et l'emplacement de sauvegarde
 - Exporter

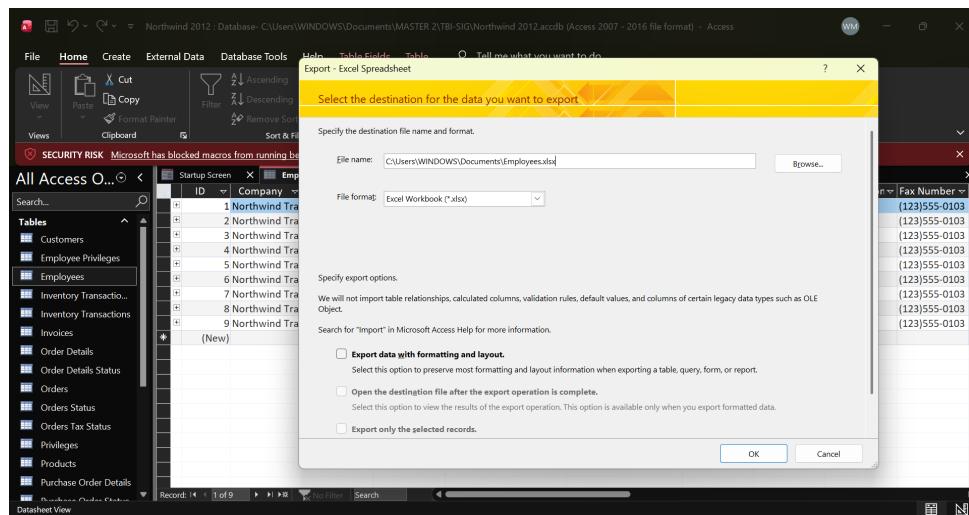


FIGURE 6.2 – Export des tables depuis Access

6.3.2 Tables exportées

Customers (Access)

- Fichier : Customers.xlsx
- Contient des données clients complémentaires ou historiques

Employees (Access)

- Fichier : Employees.xlsx
- Informations additionnelles sur les employés

Orders (Access)

- Fichier : Orders.xlsx
- Commandes additionnelles ou données historiques

6.4 Nettoyage des fichiers Excel

6.4.1 Colonnes à supprimer

Avant l'import dans Power BI, il est nécessaire de nettoyer les fichiers Excel exportés pour ne conserver que les colonnes pertinentes.

Types de colonnes à éliminer

- Colonnes vides ou sans données
- Colonnes système (OLE Object, Notes non structurées)
- Colonnes redondantes avec les données SQL Server
- Colonnes non pertinentes pour l'analyse (photos, signatures, etc.)

| ID | Company | Last Name | First Name | E-mail | Adr | Job Title | Business FHome Phc Mobile Ph Fax Numb Address | City | State/Prov ZIP/Postal Country/R | Web Page Notes | Attachments |
|----|-----------|-----------------|------------|-----------|-------------|-------------------------|---|------|---------------------------------|----------------|---|
| 1 | Northwind | Freehafer | Nancy | nancy@nc | Sales Rep | (123)555-((123)555-0102 | (123)555-123 1st Av Seattle | WA | 99999 | USA | #http://northwindtraders.com# |
| 2 | Northwind | Cencini | Andrew | andrew@ | Vice Presic | (123)555-((123)555-0102 | (123)555-123 2nd A Bellevue | WA | 99999 | USA | http://nor Joined the company as a sales rep |
| 3 | Northwind | Kotsas | Jan | jan@nortl | Sales Repr | (123)555-((123)555-0102 | (123)555-123 3rd Av Redmond | WA | 99999 | USA | http://nor Was hired as a sales associate and |
| 4 | Northwind | Sergienko | Mariya | mariya@n | Sales Repr | (123)555-((123)555-0102 | (123)555-123 4th Av Kirkland | WA | 99999 | USA | http://northwindtraders.com#http://northw |
| 5 | Northwind | Thorpe | Steven | steven@n | Sales Man | (123)555-((123)555-0102 | (123)555-123 5th Av Seattle | WA | 99999 | USA | http://nor Joined the company as a sales rep |
| 6 | Northwind | Nepper | Michael | michael@ | Sales Repr | (123)555-((123)555-0102 | (123)555-123 6th Av Redmond | WA | 99999 | USA | http://nor Fluent in Japanese and can read a |
| 7 | Northwind | Zare | Robert | robert@n | Sales Repr | (123)555-((123)555-0102 | (123)555-123 7th Av Seattle | WA | 99999 | USA | http://northwindtraders.com#http://northw |
| 8 | Northwind | Giussani | Laura | laura@n | Sales Coor | (123)555-((123)555-0102 | (123)555-123 8th Av Redmond | WA | 99999 | USA | http://nor Reads and writes French. |
| 9 | Northwind | Hellung-La Anne | | anne@n | Sales Repr | (123)555-((123)555-0102 | (123)555-123 9th Av Seattle | WA | 99999 | USA | http://nor Fluent in French and German. |
| 10 | | | | | | | | | | | |
| 11 | | | | | | | | | | | |
| 12 | | | | | | | | | | | |
| 13 | | | | | | | | | | | |
| 14 | | | | | | | | | | | |
| 15 | | | | | | | | | | | |
| 16 | | | | | | | | | | | |
| 17 | | | | | | | | | | | |
| 18 | | | | | | | | | | | |
| 19 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |

FIGURE 6.3 – Nettoyage des colonnes dans Excel AVANT

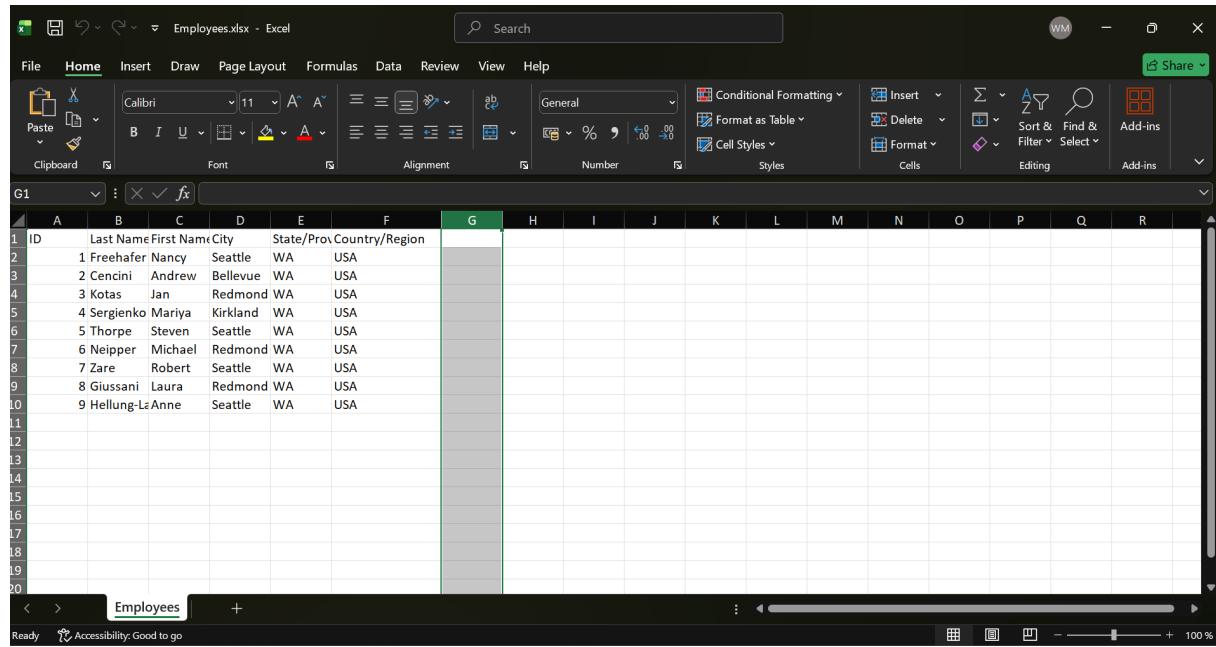


FIGURE 6.4 – Nettoyage des colonnes dans Excel APRES

6.4.2 Standardisation des données

Lors du nettoyage, s'assurer de :

- Uniformiser les noms de colonnes (pas d'espaces, caractères spéciaux)
- Vérifier les types de données (dates, nombres, texte)
- Supprimer les lignes vides
- Gérer les valeurs nulles

6.5 Récapitulatif des sources

| Source | Table | Utilisation |
|----------------|---------------------|----------------------------|
| SQL Server | Customers | Dimension Client |
| | Employees | Dimension Employé |
| | EmployeeTerritories | Liaison Employé-Territoire |
| | Territories | Données sur le territoire |
| | Orders | Table de faits |
| Access (Excel) | Customers | Données complémentaires |
| | Employees | Données complémentaires |
| | Orders | Données historiques |

TABLE 6.1 – Récapitulatif des sources de données

Les données sont maintenant prêtes à être importées dans Power BI pour le processus ETL et la création de l'entrepôt de données.

Chapitre 7

Déploiement des tables dans Power BI

7.1 Introduction

Le déploiement consiste à charger les données depuis les différentes sources dans Power BI pour pouvoir les transformer et les modéliser.

7.2 Connexion à SQL Server

7.2.1 Procédure de connexion

Pour importer les tables depuis SQL Server :

1. Ouvrir Power BI Desktop
2. Cliquer sur **Accueil** → **Obtenir des données** → **SQL Server**
3. Dans la fenêtre de connexion, renseigner :
 - **Serveur** : Nom ou adresse IP du serveur SQL Server
 - **Base de données** : Northwind
4. Cliquer sur **OK**

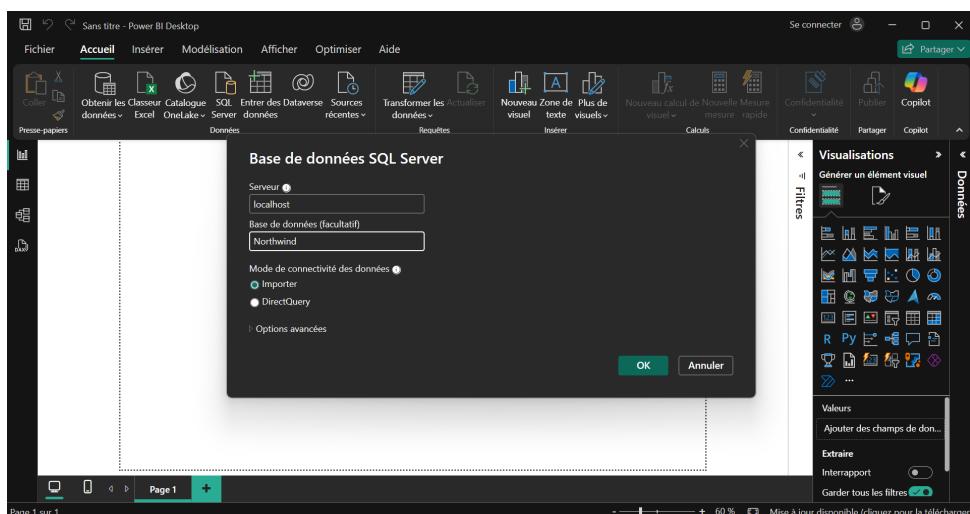


FIGURE 7.1 – Connexion à SQL Server

7.2.2 Sélection des tables SQL Server

Dans la fenêtre de navigation :

1. Cocher les tables nécessaires :
 - Customers
 - Employees
 - EmployeeTerritories
 - Territories
 - Orders
2. Cliquer sur **Charger** pour importer directement les tables

| TerritoryID | TerritoryDescription | RegionID | Emp |
|-------------|----------------------|----------|-----|
| 01581 | Westboro | 1 | T |
| 01730 | Bedford | 1 | T |
| 01833 | Georgetown | 1 | T |
| 02116 | Boston | 1 | T |
| 02139 | Cambridge | 1 | T |
| 02184 | Braintree | 1 | T |
| 02903 | Providence | 1 | T |
| 03049 | Hollis | 3 | T |
| 03801 | Portsmouth | 3 | T |
| 06897 | Wilton | 1 | T |
| 07960 | Morristown | 1 | T |
| 08837 | Edison | 1 | T |
| 10019 | New York | 1 | T |
| 10038 | New York | 1 | T |
| 11747 | Melville | 1 | T |
| 14450 | Fairport | 3 | T |
| 19428 | Philadelphia | 3 | T |
| 19713 | Neward | 1 | T |
| 20852 | Rockville | 1 | T |
| 27403 | Greensboro | 1 | T |
| 27511 | Cary | 1 | T |

FIGURE 7.2 – Sélection des tables depuis SQL Server

7.3 Importation des fichiers Excel

7.3.1 Procédure d'importation

Les tables Access ayant été préalablement exportées vers Excel, nous les importons maintenant :

1. Cliquer sur **Accueil** → **Obtenir des données** → **Excel**
2. Naviguer vers l'emplacement des fichiers Excel
3. Sélectionner le premier fichier : **Customers_Access.xlsx**
4. Dans la fenêtre de navigation, sélectionner la feuille contenant les données
5. Cliquer sur **Charger**
6. Répéter l'opération pour :
 - **Employees_Access.xlsx**
 - **Orders_Access.xlsx**

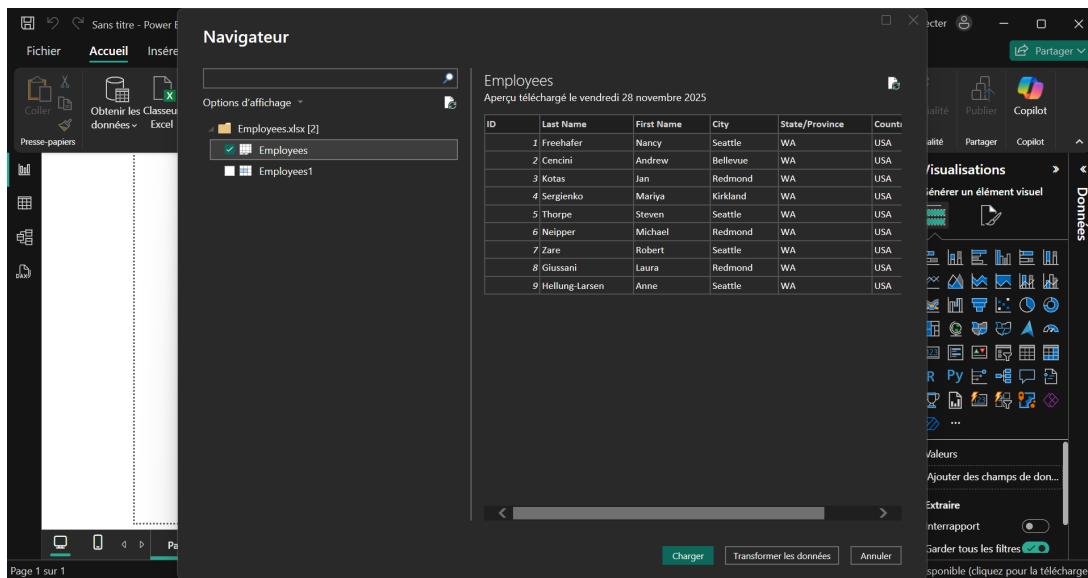


FIGURE 7.3 – Importation des fichiers Excel dans Power BI

7.4 Vérification du chargement

7.4.1 Panneau Champs

Après le chargement de toutes les sources, vérifier dans le panneau Champs (à droite) que toutes les tables sont présentes :

Tables SQL Server :

- Customers
- Employees
- EmployeeTerritories
- Territories
- Orders

Tables Excel (Access) :

- Customers_Excel (ou Customers_Access)
- Employees_Excel (ou Employees_Access)
- Temps_Excel (ou Orders_Access pour les dates)

7.4.2 Vérification des données

Pour chaque table, vérifier rapidement :

1. Aller dans la **Vue Données** (icône de tableau à gauche)
2. Sélectionner une table dans le panneau Champs
3. Vérifier :
 - Le nombre de lignes chargées
 - Les colonnes présentes
 - L'absence d'erreurs dans les données

The screenshot shows the Power BI Data Editor interface. At the top, there's a ribbon with 'Fichier', 'Accueil', 'Aide', and 'Outils de table'. The 'Outils de table' tab is selected. Below the ribbon, there are several buttons: 'Nom' (Customers), 'Gérer les relations' (Relations), 'Nouvelle mesure' (New Measure), 'Nouvelle colonne' (New Column), 'Nouvelle table' (New Table), 'Calculs' (Calculus), and 'Calendriers' (Calendars). The main area displays the 'Customers' table with 91 rows of data. The columns are: CustomerID, CompanyName, ContactName, ContactTitle, Address, City, and Region. The 'Region' column has a dropdown arrow indicating it's a dimension. To the right of the table is a 'Données' pane containing a tree view of data sources: Customers, Customers_Excel, Dim_Employe, Dim_Temps, DimClient, Employees, Employees_Excel, EmployeeTerritories, Orders, Orders_Excel, Temps_Excel, Territories, and TF_Commande. A search bar at the top of the pane says 'Rechercher'. At the bottom of the editor, it says 'Table : Customers (91 lignes)' and 'Mise à jour disponible (cliquez pour la télécharger)'.

FIGURE 7.4 – Vérification des données chargées

7.5 Résumé du déploiement

À ce stade, toutes les tables sources sont chargées dans Power BI :

- 5 tables depuis SQL Server
- 3 fichiers Excel (données Access)
- Total : 8 sources de données prêtes pour transformation

Les données sont maintenant disponibles pour la phase de transformation et la création des dimensions dans Power Query Editor.

Chapitre 8

Création et remplissage des dimensions

8.1 Introduction

Les dimensions sont créées dans Power Query Editor en fusionnant et transformant les données des différentes sources. Notre entrepôt suivra un modèle en étoile avec trois dimensions principales.

8.2 Dimension Employee (DimEmployee)

8.2.1 Objectif

Créer une dimension Employee unique en fusionnant les données de SQL Server (Employees, EmployeeTerritories, Territories) et du fichier Excel (Employees_Access), avec ajout d'une clé de substitution.

8.2.2 Accès à Power Query

Pour créer cette dimension :

1. Cliquer sur **Accueil** → **Transformer les données**
2. L'éditeur Power Query s'ouvre
3. Créer une nouvelle requête vide : **Nouvelle source** → **Requête vide**
4. Nommer la requête : **DimEmployee**

8.2.3 Script Power Query M

Dans l'éditeur avancé de la requête DimEmployee, saisir le code suivant :

```
1 let
2     // =====
3     // ETAPE 1: Preparer Employees (SSMS) avec Territories
4     // =====
5     // Jointure Employees + EmployeeTerritories
6     Employees_WithTerritories = Table.NestedJoin(
7         #"Employees",
8         {"EmployeeID"},
9         #"EmployeeTerritories",
```

```
10     {"EmployeeID"},  
11     "EmpTerritories",  
12     JoinKind.LeftOuter  
13 ),  
14  
15 // Développer la colonne TerritoryID  
16 #"ExpandedEmpTerritories" = Table.ExpandTableColumn(  
17     Employees_WithTerritories,  
18     "EmpTerritories",  
19     {"TerritoryID"},  
20     {"TerritoryID"}  
21 ),  
22  
23 // Jointure avec Territories  
24 Employees_WithTerritoryDesc = Table.NestedJoin(  
25     #"ExpandedEmpTerritories",  
26     {"TerritoryID"},  
27     #"Territories",  
28     {"TerritoryID"},  
29     "TerritoryData",  
30     JoinKind.LeftOuter  
31 ),  
32  
33 // Développer TerritoryDescription  
34 #"ExpandedTerritoryData" = Table.ExpandTableColumn(  
35     Employees_WithTerritoryDesc,  
36     "TerritoryData",  
37     {"TerritoryDescription"},  
38     {"TerritoryDescription"}  
39 ),  
40  
41 // Sélection colonnes SSMS  
42 SSMS_Prepared = Table.SelectColumns(  
43     #"ExpandedTerritoryData",  
44     {  
45         "EmployeeID",  
46         "LastName",  
47         "FirstName",  
48         "Region",  
49         "TerritoryID",  
50         "TerritoryDescription"  
51     }  
52 ),  
53  
54 // Ajout colonne Source  
55 #"SSMS_WithSource" = Table.AddColumn(  
56     SSMS_Prepared,  
57     "Source",  
58     each "SSMS"  
59 ),  
60
```

```
61 // =====
62 // ETAPE 2: Preparer Employees_Excel
63 // =====
64 Excel_Prepared = Table.SelectColumns(
65     #"Employees_Excel",
66     {"ID", "Last\u00c3\u00b9Name", "First\u00c3\u00b9Name", "City", "State/Province"})
67 ),
68
69 #"Excel_WithSource" = Table.AddColumn(
70     Excel_Prepared,
71     "Source",
72     each "EXCEL"
73 ),
74
75 // =====
76 // ETAPE 3: Standardiser les noms de colonnes
77 // =====
78 SSMS_Standardized = Table.RenameColumns(
79     #"SSMS_WithSource",
80     {
81         {"EmployeeID", "id_employee_prod"}, 
82         {"LastName", "Nom"}, 
83         {"FirstName", "Prenom"}, 
84         {"TerritoryID", "Territory"}, 
85         {"TerritoryDescription", "TerritoryDesc"}, 
86         {"Region", "Region"}, 
87         {"Source", "source_prod"}
88     }
89 ),
90
91 Excel_Standardized = Table.RenameColumns(
92     #"Excel_WithSource",
93     {
94         {"ID", "id_employee_prod"}, 
95         {"Last\u00c3\u00b9Name", "Nom"}, 
96         {"First\u00c3\u00b9Name", "Prenom"}, 
97         {"City", "Territory"}, 
98         {"State/Province", "TerritoryDesc"}, 
99         {"Source", "source_prod"}
100    }
101 ),
102
103 // Ajouter Region vide pour Excel
104 #"Excel_WithRegion" = Table.AddColumn(
105     Excel_Standardized,
106     "Region",
107     each null,
108     type text
109 ),
110
111 // =====
```

```
112 // ETAPE 4: Combiner les deux sources
113 // =====
114 CombinedData = Table.Combine({
115     SSMS_Standardized,
116     #"Excel_WithRegion"
117 },
118
119 // =====
120 // ETAPE 5: Convertir id_employee_prod en texte
121 // =====
122 #"TypeCorrige" = Table.TransformColumnTypes(
123     CombinedData,
124     {"id_employee_prod", type text}
125 ),
126
127 // =====
128 // ETAPE 6: Ajouter l'ID_sequentiel_unique
129 // =====
130 // #"AddedIndex"=Table.AddIndexColumn(
131 // #"TypeCorrige",
132 // "id_seqEmployee",
133 // 1,
134 // 1,
135 // Int64.Type
136 // ),
137
138 // =====
139 // ETAPE 7: Selection ordre final des colonnes
140 // =====
141 // #"FinalColumns"=Table.SelectColumns(
142 // #"AddedIndex",
143 //{
144     "id_seqEmployee",
145     "id_employee_prod",
146     "source_prod",
147     "Nom",
148     "Prenom",
149     "Territory",
150     "TerritoryDesc",
151     "Region"
152 }
153 )
154 in
155 // #"FinalColumns"
```

Listing 8.1 – Script complet de création de DimEmployee

8.2.4 Explication des étapes

Étape 1 : Enrichissement des données SSMS

Cette étape réalise trois jointures successives :

1. **Employees + EmployeeTerritories** : Associe chaque employé à ses territoires
2. **Résultat + Territories** : Récupère la description des territoires
3. **Sélection des colonnes** : Garde uniquement les champs pertinents
4. **Ajout de la source** : Marque les données comme provenant de "SSMS"

Étape 2 : Préparation des données Excel

- Sélection des colonnes du fichier Employees_Excel
- Ajout d'une colonne "Source" avec la valeur "EXCEL"

Étape 3 : Standardisation

Renommage des colonnes pour avoir la même structure :

- EmployeeID → id_employee_prod
- LastName → Nom
- FirstName → Prenom
- Etc.

Ajout de la colonne Region (valeur null) pour Excel afin d'harmoniser les structures.

Étape 4 : Combinaison

Fusion verticale des deux sources avec Table.Combine :

- Les lignes de SSMS
- Les lignes d'Excel
- = Une seule table unifiée

Étape 5 : Conversion des types

Conversion de id_employee_prod en texte pour uniformiser les identifiants (certains sont numériques, d'autres textuels).

Étape 6 : Clé de substitution

Ajout d'un identifiant unique séquentiel id_seqEmployee qui commence à 1 et s'incrémente de 1 pour chaque ligne.

Étape 7 : Sélection finale

Organisation des colonnes dans l'ordre souhaité pour la dimension.

8.2.5 Structure finale de DimEmployee

| Colonne | Type | Description |
|------------------|--------|--|
| id_seqEmployee | Entier | Clé de substitution unique (surrogate key) |
| id_employee_prod | Texte | Identifiant employé d'origine (business key) |
| source_prod | Texte | Source des données : SSMS ou EXCEL |
| Nom | Texte | Nom de famille de l'employé |
| Prenom | Texte | Prénom de l'employé |
| Territory | Texte | Identifiant du territoire |
| TerritoryDesc | Texte | Description du territoire de vente |
| Region | Texte | Région géographique (null pour Excel) |

TABLE 8.1 – Structure de la dimension DimEmployee

8.2.6 Chargement de la dimension

Une fois le script validé :

1. Cliquer sur **Fermer et appliquer** dans Power Query
2. La dimension DimEmployee est chargée dans le modèle
3. Vérifier dans la **Vue Données** que la table est présente

The screenshot shows the Power BI desktop application. The top navigation bar includes 'Fichier', 'Accueil', 'Aide', and 'Outils de table'. The 'Outils de table' tab is selected. Below the tabs, there are four buttons: 'Gérer les relations' (Manage relationships), 'Nouvelle Mesure' (New measure), 'Nouvelle colonne' (New column), and 'Calculs' (Calculus). To the right of these buttons are three more buttons: 'Marquer comme table de dates' (Mark as date table), 'Calendriers' (Calendars), and 'Partager' (Share). The main area displays the 'Dim_Employee' table with 58 rows. The columns are: id_seqEmployee, id_employee_prod, source_prod, Nom, Prenom, Territory, TerritoryDesc, and Region. The data shows various employees from different territories and regions. On the right side, the 'Données' pane is open, showing a hierarchical list of tables and measures available in the model, including 'Customers', 'Dim_Employee' (which is currently selected), 'Dim_Temps', 'DimClient', 'Employees', and others. At the bottom left, it says 'Table : Dim_Employee (58 lignes)'. At the bottom right, it says 'Mise à jour disponible (cliquez pour la télécharger)'.

FIGURE 8.1 – Dimension DimEmployee créée

La dimension Employee est maintenant prête. Nous allons créer les autres dimensions selon le même principe.

8.3 Dimension Client (DimClient)

8.3.1 Objectif

Créer une dimension Client unique en fusionnant les données de SQL Server (Customers) et du fichier Excel (Customers_Access), avec ajout d'une clé de substitution.

8.3.2 Script Power Query M

Créer une nouvelle requête nommée **DimClient** et saisir le code suivant :

```

1 let
2     // =====
3     // ETAPE 1: Preparer Customers (SQL Server)
4     // =====
5     Customers_SSMS_Prepared = Table.SelectColumns(
6         #"Customers",
7         {"CustomerID", "CompanyName", "City"})
8     ),
9
10    #"SSMS_WithSource" = Table.AddColumn(
11        Customers_SSMS_Prepared,
12        "Source",
13        each "SSMS")
14    ),
15
16    // =====
17    // ETAPE 2: Preparer Customers_Excel
18    // =====
19    Customers_Excel_Prepared = Table.SelectColumns(
20        #"Customers_Excel",
21        {"ID", "Company", "City"})
22    ),
23
24    #"Excel_WithSource" = Table.AddColumn(
25        Customers_Excel_Prepared,
26        "Source",
27        each "EXCEL")
28    ),
29
30    // =====
31    // ETAPE 3: Standardiser les noms de colonnes
32    // =====
33    SSMS_Standardized = Table.RenameColumns(
34        #"SSMS_WithSource",
35        {
36            {"CustomerID", "id_client_prod"},
37            {"CompanyName", "CompanyName"},
38            {"City", "City"},
39            {"Source", "source_prod"}
40        })

```

```
41 ),
42
43     Excel_Standardized = Table.RenameColumns(
44         #"Excel_WithSource",
45         {
46             {"ID", "id_client_prod"},  

47             {"Company", "CompanyName"},  

48             {"City", "City"},  

49             {"Source", "source_prod"}
50         }
51 ),
52
53 // =====
54 // ETAPE 4: Combiner les deux sources
55 // =====
56 CombinedData = Table.Combine({
57     SSMS_Standardized,
58     Excel_Standardized
59}),
60
61 // =====
62 // ETAPE 5: Convertir id_client_prod en texte
63 // =====
64 #"TypeCorrige" = Table.TransformColumnTypes(
65     CombinedData,
66     {{"id_client_prod", type text}}
67),
68
69 // =====
70 // ETAPE 6: Ajouter l'ID séquentiel unique
71 // =====
72 #"AddedIndex" = Table.AddIndexColumn(
73     #"TypeCorrige",
74     "id_seqClient",
75     1,
76     1,
77     Int64.Type
78),
79
80 // =====
81 // ETAPE 7: Sélectionner l'ordre final des colonnes
82 // =====
83 #"FinalColumns" = Table.SelectColumns(
84     #"AddedIndex",
85     {
86         "id_seqClient",
87         "id_client_prod",
88         "source_prod",
89         "CompanyName",
90         "City"
91     })
```

```
92       )
93   in
94       #"FinalColumns"
```

Listing 8.2 – Script complet de création de DimClient

8.3.3 Explication des étapes

Le processus de création de DimClient suit la même logique que DimEmployee :

Étape 1 : Préparation des données SQL Server

- Sélection des colonnes : CustomerID, CompanyName, City
- Ajout d'une colonne "Source" avec la valeur "SSMS"

Étape 2 : Préparation des données Excel

- Sélection des colonnes : ID, Company, City
- Ajout d'une colonne "Source" avec la valeur "EXCEL"

Étape 3 : Standardisation des colonnes

Renommage pour uniformiser :

- CustomerID / ID → id_client_prod
- CompanyName / Company → CompanyName
- City reste City
- Source → source_prod

Étape 4 : Combinaison

Fusion verticale des deux sources avec Table.Combine.

Étape 5 : Conversion des types

Conversion de id_client_prod en texte pour uniformité.

Étape 6 : Clé de substitution

Ajout de id_seqClient : identifiant unique séquentiel commençant à 1.

Étape 7 : Sélection finale

Organisation des colonnes dans l'ordre souhaité.

8.3.4 Structure finale de DimClient

| Colonne | Type | Description |
|----------------|--------|------------------------------------|
| id_seqClient | Entier | Clé de substitution unique |
| id_client_prod | Texte | Identifiant client d'origine |
| source_prod | Texte | Source des données : SSMS ou EXCEL |
| CompanyName | Texte | Nom de l'entreprise cliente |
| City | Texte | Ville du client |

TABLE 8.2 – Structure de la dimension DimClient

8.3.5 Chargement de la dimension

Après validation du script :

1. Cliquer sur **Fermer et appliquer**
2. Vérifier dans la Vue Données que DimClient est chargée

| id_seqClient | id_client_prod | source_prod | CompanyName | City |
|--------------|----------------|------------------------------------|--------------|------|
| 1 ALFKI | SSMS | Alfreds Futterkiste | Berlin | |
| 2 ANATR | SSMS | Ana Trujillo Emparedados y helados | México D.F. | |
| 3 ANTON | SSMS | Antonio Moreno Taquería | México D.F. | |
| 4 AROUT | SSMS | Around the Horn | London | |
| 5 BERGS | SSMS | Berglunds snabbköp | Luleå | |
| 6 BLAUS | SSMS | Blauer See Delikatessen | Mannheim | |
| 7 BLONP | SSMS | Blondes Dds p re et fils | Strasbourg | |
| 8 BOLID | SSMS | B lido Comidas preparadas | Madrid | |
| 9 BONAP | SSMS | Bon app' | Marseille | |
| 10 BOTTM | SSMS | Bottom-Dollar Markets | Tsawassen | |
| 11 BSBEV | SSMS | B's Beverages | London | |
| 12 CACTU | SSMS | Cactus Comidas para llevar | Buenos Aires | |
| 13 CENTC | SSMS | Centro comercial Moctezuma | México D.F. | |
| 14 CHOPS | SSMS | Chop-suey Chinese | Bern | |
| 15 COMMI | SSMS | Com'río Mineiro | Sao Paulo | |
| 16 CONSH | SSMS | Consolidated Holdings | London | |
| 17 DRACD | SSMS | Drachenblut Delikatessen | Aachen | |
| 18 DUMON | SSMS | Du monde entier | Nantes | |
| 19 EASTC | SSMS | Eastern Connection | London | |
| 20 ERNSH | SSMS | Ernst Handel | Graz | |

FIGURE 8.2 – Dimension DimClient créée

La dimension Client est maintenant prête. Il reste à créer la dimension Temps.

8.4 Dimension Temps (DimTemps)

8.4.1 Objectif

Créer une dimension temporelle à partir de toutes les dates de commandes provenant de SQL Server (Orders) et du fichier Excel (Temps_Excel), avec extraction de l'année et du format mois/année.

8.4.2 Script Power Query M

Créer une nouvelle requête nommée **DimTemps** et saisir le code suivant :

```

1 let
2     // =====
3     // ETAPE 1: Extraire les dates des commandes
4     // =====
5     // Dates depuis Temps_Excel
6     Dates_TempsExcel = Table.SelectColumns(
7         #"Temps_Excel",
8         {"Order\u00d7Date"})
9     ),
10
11    #"Renomme_TempsExcel" = Table.RenameColumns(
12        Dates_TempsExcel,
13        {"Order\u00d7Date", "DateComplete"})
14    ),
15
16    // Dates depuis Orders (SQL Server)
17    Dates_Orders = Table.SelectColumns(
18        #"Orders",
19        {"OrderDate"})
20    ),
21
22    #"Renomme_Orders" = Table.RenameColumns(
23        Dates_Orders,
24        {"OrderDate", "DateComplete"})
25    ),
26
27    // =====
28    // ETAPE 2: Combiner toutes les dates
29    // =====
30    ToutesDates = Table.Combine({
31        #"Renomme_TempsExcel",
32        #"Renomme_Orders"
33    }),
34
35    // Filtrer les dates nulles ou vides
36    #"DatesFiltrees" = Table.SelectRows(
37        ToutesDates,
38        each [DateComplete] <> null and [DateComplete] <> ""
39    ),
40
41    // Conversion en type date
42    #"TypeDate" = Table.TransformColumnTypes(
43        #"DatesFiltrees",
44        {"DateComplete", type date})
45    ),
46
47    // =====
48    // ETAPE 3: Ajouter colonnes calculees

```

```

49 // =====
50 // Ajout de l'annee
51 #"AjouteAnnee" = Table.AddColumn(
52     TypeDate",
53     "annee",
54     each Date.Year([DateComplete]),
55     Int64.Type
56 ),
57
58 // Ajout du format mois/annee (MM/YYYY)
59 #"AjouteMoisAnnee" = Table.AddColumn(
60     "AjouteAnnee",
61     "mois_annee",
62     each Text.PadStart(
63         Text.From(Date.Month([DateComplete])), 2,
64         "0"
65     ) & "/" & Text.From(Date.Year([DateComplete])),
66     type text
67 ),
68
69
70 // =====
71 // ETAPE 4: Ajouter l'ID sequentiel
72 // =====
73 #"AjouteIndex" = Table.AddIndexColumn(
74     "AjouteMoisAnnee",
75     "id_temps",
76     1,
77     1,
78     Int64.Type
79 ),
80
81 // =====
82 // ETAPE 5: Selection finale (sans DateComplete)
83 // =====
84 #"SansDateComplete" = Table.SelectColumns(
85     "AjouteIndex",
86     {
87         "id_temps",
88         "annee",
89         "mois_annee"
90     }
91 )
92 in
93     # "SansDateComplete"

```

Listing 8.3 – Script complet de création de DimTemps

8.4.3 Explication des étapes

Étape 1 : Extraction des dates

Extraction des dates depuis deux sources :

- **Temps_Excel** : Colonne "Order Date"
- **Orders (SQL Server)** : Colonne "OrderDate"
- Les deux colonnes sont renommées en "DateComplete" pour uniformiser

Étape 2 : Combinaison et nettoyage

- Fusion de toutes les dates avec Table.Combine
- Filtrage des valeurs nulles ou vides
- Conversion en type date

Étape 3 : Colonnes calculées

Ajout de deux colonnes dérivées :

- **annee** : Extraction de l'année avec Date.Year()
- **mois_annee** : Format MM/YYYY
 - Text.PadStart : Force le mois sur 2 chiffres (01, 02, ..., 12)
 - Exemple : 07/1996, 12/1997

Étape 4 : Clé de substitution

Ajout de **id_temps** : identifiant unique séquentiel.

Étape 5 : Sélection finale

Conservation uniquement des colonnes nécessaires :

- **id_temps** : Clé primaire
- **annee** : Pour analyses annuelles
- **mois_annee** : Pour analyses mensuelles

La colonne **DateComplete** est supprimée car non nécessaire dans la dimension finale.

8.4.4 Structure finale de DimTemps

| Colonne | Type | Description |
|-------------------|--------|--|
| id_temps | Entier | Clé de substitution unique |
| annee | Entier | Année de la commande (ex : 1996, 1997) |
| mois_annee | Texte | Format MM/YYYY (ex : 07/1996, 12/1997) |

TABLE 8.3 – Structure de la dimension DimTemps

8.4.5 Chargement de la dimension

Après validation du script :

1. Cliquer sur **Fermer et appliquer**
2. Vérifier dans la Vue Données que DimTemps est chargée

Table : Dim_Temps (878 lignes)

Mise à jour disponible (cliquez pour la télécharger)

FIGURE 8.3 – Dimension DimTemps créée

8.5 Récapitulatif des dimensions

Les trois dimensions de notre entrepôt sont maintenant créées :

| Dimension | Clé primaire | Attributs principaux |
|-------------|----------------|--|
| DimEmployee | id_seqEmployee | Nom, Prénom, Territory, Region, Source |
| DimClient | id_seqClient | CompanyName, City, Source |
| DimTemps | id_temps | Année, Mois/Année |

TABLE 8.4 – Récapitulatif des trois dimensions

8.6 Vérification finale

Dans Power BI Desktop, vérifier que les trois dimensions sont bien chargées :

1. Aller dans la **Vue Données**
2. Vérifier chaque dimension :
 - DimEmployee : Nombre de lignes, présence de la clé id_seqEmployee
 - DimClient : Nombre de lignes, présence de la clé id_seqClient
 - DimTemps : Nombre de lignes, présence de la clé id_temps
3. Vérifier qu'il n'y a pas d'erreurs dans les données

Les dimensions sont maintenant prêtes pour la création de la table de faits et l'établissement des relations dans le modèle en étoile.

Chapitre 9

Création de la table de faits et de l'entrepôt

9.1 Introduction

La table de faits est l'élément central du modèle en étoile. Elle contient les mesures quantitatives (faits) et les clés étrangères vers les dimensions.

9.2 Table de faits : TF_Commande

9.2.1 Objectif

Créer une table de faits centralisant les commandes provenant de SQL Server et Excel, avec des métriques sur les livraisons, et liée aux trois dimensions via leurs clés de substitution.

9.2.2 Métriques calculées

La table de faits contient deux mesures principales :

- **nbr_commande_livrees** : Nombre de commandes livrées (ShippedDate non null)
- **nbr_commande_non_livrees** : Nombre de commandes non livrées (ShippedDate null)

9.2.3 Script Power Query M

Créer une nouvelle requête nommée **TF_Commande** et saisir le code suivant :

```
1 let
2     // =====
3     // ETAPE 1: Preparer les donnees avec flags de livraison
4     // =====
5     // Donnees SQL Server
6     Orders_SSMS = Table.SelectColumns(
7         #"Orders",
8         {"OrderID", "CustomerID", "EmployeeID", "OrderDate", "ShippedDate"}
```

```

9   ),
10
11  #"SSMS_WithFlags" = Table.AddColumn(
12    Orders_SSMS,
13    "nbr_commande_livrees",
14    each if [ShippedDate] <> null then 1 else 0
15  ),
16
17  #"SSMS_WithFlags2" = Table.AddColumn(
18    #"SSMS_WithFlags",
19    "nbr_commande_non_livrees",
20    each if [ShippedDate] = null then 1 else 0
21  ),
22
23  #"SSMS_WithSource" = Table.AddColumn(
24    #"SSMS_WithFlags2",
25    "source_prod",
26    each "SSMS"
27  ),
28
29 // Donnees Excel
30 Orders_Excel_Prepared = Table.SelectColumns(
31   #"Orders_Excel",
32   {"Order\u00d7ID", "Customer\u00d7ID", "Employee\u00d7ID", "Order\u00d7Date", "Shipped\u00d7Date"}
33 ),
34
35  #"Excel_WithFlags" = Table.AddColumn(
36    Orders_Excel_Prepared,
37    "nbr_commande_livrees",
38    each if [Shipped Date] <> null then 1 else 0
39  ),
40
41  #"Excel_WithFlags2" = Table.AddColumn(
42    #"Excel_WithFlags",
43    "nbr_commande_non_livrees",
44    each if [Shipped Date] = null then 1 else 0
45  ),
46
47  #"Excel_WithSource" = Table.AddColumn(
48    #"Excel_WithFlags2",
49    "source_prod",
50    each "EXCEL"
51  ),
52
53 // Standardiser les noms de colonnes
54 SSMS_Standardized = Table.RenameColumns(
55   {"SSMS_WithSource",
56   {
57     {"OrderID", "OrderID"},

58     {"CustomerID", "CustomerID"},


```

```

59         {"EmployeeID", "EmployeeID"},  

60         {"OrderDate", "OrderDate"},  

61         {"ShippedDate", "ShippedDate"}  

62     }  

63 ),  

64  

65 Excel_Standardized = Table.RenameColumns(  

66     #"Excel_WithSource",  

67     {  

68         {"Order_ID", "OrderID"},  

69         {"Customer_ID", "CustomerID"},  

70         {"Employee_ID", "EmployeeID"},  

71         {"Order_Date", "OrderDate"},  

72         {"Shipped_Date", "ShippedDate"}  

73     }  

74 ),  

75  

76 // Combiner les deux sources  

77 CombinedData = Table.Combine({  

78     SSMS_Standardized,  

79     Excel_Standardized  

80 }),  

81  

82 // =====  

83 // ETAPE 2: Ajouter id_temps sequentiel  

84 // =====  

85 #"TypeCorrigé" = Table.TransformColumnTypes(  

86     CombinedData,  

87     {  

88         {"CustomerID", type text},  

89         {"EmployeeID", type text},  

90         {"OrderID", type text},  

91         {"OrderDate", type date}  

92     }  

93 ),  

94  

95 // Ajouter id_temps (1 par commande)  

96 #"AjouteIdTemps" = Table.AddIndexColumn(  

97     #"TypeCorrigé",  

98     "id_temps",  

99     1,  

100    1,  

101    Int64.Type  

102 ),  

103  

104 // =====  

105 // ETAPE 3: Jointures avec DimClient et DimEmployee  

106 // =====  

107 // Jointure avec DimClient  

108 #"JoinDimClient" = Table.NestedJoin(  

109     #"AjouteIdTemps",  


```

```

110     {"CustomerID", "source_prod"},  

111     # "DimClient",  

112     {"id_client_prod", "source_prod"},  

113     "ClientData",  

114     JoinKind.LeftOuter  

115 ),  

116  

117 # "ExpandedClient" = Table.ExpandTableColumn(  

118     # "JoinDimClient",  

119     "ClientData",  

120     {"id_seqClient"},  

121     {"id_seqClient"}  

122 ),  

123  

124 // Dedoublonner DimEmployee avant jointure  

125 Dim_Employee_Unique = Table.Group(  

126     # "Dim_Employee",  

127     {"id_employee_prod", "source_prod"},  

128     {  

129         {"id_seqEmployee", each List.Min([id_seqEmployee]),  

130             Int64.Type}  

131     }  

132 ),  

133  

134 // Jointure avec DimEmployee  

135 # "JoinDimEmployee" = Table.NestedJoin(  

136     # "ExpandedClient",  

137     {"EmployeeID", "source_prod"},  

138     Dim_Employee_Unique,  

139     {"id_employee_prod", "source_prod"},  

140     "EmployeeData",  

141     JoinKind.LeftOuter  

142 ),  

143  

144 # "ExpandedEmployee" = Table.ExpandTableColumn(  

145     # "JoinDimEmployee",  

146     "EmployeeData",  

147     {"id_seqEmployee"},  

148     {"id_seqEmployee"}  

149 ),  

150 // ======  

151 // ETAPE 4: Agregation par commande  

152 // ======  

153 # "AgreegeParCommande" = Table.Group(  

154     # "ExpandedEmployee",  

155     {  

156         "OrderID",  

157         "id_temps",  

158         "id_seqEmployee",  

159         "id_seqClient",  

160     }

```

```

160         "source_prod"
161     } ,
162     {
163         {"nbr_commande_livrees",
164          each List.Sum([nbr_commande_livrees]),
165          type number},
166         {"nbr_commande_non_livrees",
167          each List.Sum([nbr_commande_non_livrees]),
168          type number}
169     }
170   ),
171
172 // =====
173 // ETAPE 5: Nettoyage final
174 // =====
175 #"ColonnesNettoyees" = Table.RemoveColumns(
176     #"AgreggeParCommande",
177     {"OrderID", "source_prod"})
178   ,
179
180 #"AddedIndex" = Table.AddIndexColumn(
181     #"ColonnesNettoyees",
182     "id_seq_fait",
183     1,
184     1,
185     Int64.Type
186   ,
187
188 #"FinalColumns" = Table.SelectColumns(
189     #"AddedIndex",
190     {
191         "id_seq_fait",
192         "id_temps",
193         "id_seqEmployee",
194         "id_seqClient",
195         "nbr_commande_livrees",
196         "nbr_commande_non_livrees"
197     }
198   )
199
200 in
#"FinalColumns"

```

Listing 9.1 – Script complet de création de TF_Commande

9.2.4 Explication détaillée des étapes

Étape 1 : Préparation des données sources

Calcul des flags de livraison :

- nbr_commande_livrees :
 - Si ShippedDate n'est pas null → valeur = 1

- Sinon → valeur = 0
- **nbr_commande_non_livrees :**
 - Si ShippedDate est null → valeur = 1
 - Sinon → valeur = 0

Ces flags permettront d'agréger facilement le nombre de commandes livrées/non livrées.

Ajout de la source :

- "SSMS" pour les données SQL Server
- "EXCEL" pour les données du fichier Excel

Standardisation et combinaison :

- Renommage des colonnes pour uniformiser
- Fusion avec Table.Combine

Étape 2 : Ajout de id_temps

Ajout d'un identifiant temporel séquentiel :

- **id_temps** : commence à 1 et s'incrémente de 1
- Chaque commande reçoit un id_temps unique
- Permet de lier la table de faits avec DimTemps

Étape 3 : Jointures avec les dimensions

Jointure avec DimClient :

- Critère : CustomerID + source_prod
- Récupération de **id_seqClient**
- Type : LeftOuter (garde toutes les commandes)

Dédoublonnage de DimEmployee :

- Problème : DimEmployee peut contenir des doublons (un employé avec plusieurs territoires)
- Solution : Grouper par id_employee_prod + source_prod
- Prendre le List.Min([id_seqEmployee]) pour garder un seul id

Jointure avec DimEmployee :

- Critère : EmployeeID + source_prod
- Récupération de **id_seqEmployee**
- Type : LeftOuter

Étape 4 : Agrégation par commande

Regroupement par :

- OrderID (identifiant de commande)
- **id_temps**
- **id_seqEmployee**
- **id_seqClient**
- **source_prod**

Calcul des sommes :

- Sum(nbr_commande_livrees)
- Sum(nbr_commande_non_livrees)

Étape 5 : Nettoyage final

- Suppression des colonnes techniques : OrderID, source_prod
- Ajout de id_seq_fait : clé primaire de la table de faits
- Sélection des colonnes finales dans l'ordre

9.2.5 Structure finale de TF_Commande

| Colonne | Type | Description |
|--------------------------|--------|-----------------------------------|
| id_seq_fait | Entier | Clé primaire de la table de faits |
| id_temps | Entier | Clé étrangère vers DimTemps |
| id_seqEmployee | Entier | Clé étrangère vers DimEmployee |
| id_seqClient | Entier | Clé étrangère vers DimClient |
| nbr_commande_livrees | Nombre | Nombre de commandes livrées |
| nbr_commande_non_livrees | Nombre | Nombre de commandes non livrées |

TABLE 9.1 – Structure de la table de faits TF_Commande

9.2.6 Chargement de la table de faits

Après validation du script :

1. Cliquer sur **Fermer et appliquer**
2. Vérifier dans la Vue Données que TF_Commande est chargée
3. Vérifier le nombre de lignes (environ 878 lignes)

The screenshot shows the Power BI desktop application window. The top menu bar includes 'Fichier', 'Accueil', 'Aide', and 'Outils de table'. The 'Outils de table' tab is currently selected. Below the menu is a toolbar with icons for 'Nom' (Name), 'Gérer les relations' (Manage relationships), 'Nouvelle mesure' (New measure), 'Nouvelle rapide' (New quick), 'Nouvelle colonne' (New column), 'Nouvelle table' (New table), 'Calculs' (Calculations), and 'Calendriers' (Calendars). The main area displays a table named 'TF_Commande' with the following data:

| id_seq_fait | id_temps | id_seqEmployee | id_seqClient | nbr_commande_livrees | nbr_commande_non_livrees |
|-------------|----------|----------------|--------------|----------------------|--------------------------|
| 13 | 3 | 14 | 34 | 1 | 0 |
| 14 | 5 | 14 | 76 | 1 | 0 |
| 15 | 55 | 14 | 76 | 1 | 0 |
| 16 | 113 | 14 | 7 | 1 | 0 |
| 21 | 79 | 14 | 8 | 1 | 0 |
| 26 | 10 | 14 | 35 | 1 | 0 |
| 32 | 12 | 14 | 13 | 1 | 0 |
| 33 | 13 | 14 | 56 | 1 | 0 |
| 34 | 14 | 14 | 61 | 1 | 0 |
| 38 | 47 | 14 | 65 | 1 | 0 |
| 45 | 116 | 14 | 17 | 1 | 0 |
| 52 | 20 | 14 | 25 | 1 | 0 |
| 53 | 90 | 14 | 25 | 1 | 0 |
| 54 | 95 | 14 | 25 | 1 | 0 |
| 56 | 100 | 14 | 21 | 1 | 0 |
| 58 | 97 | 14 | 89 | 1 | 0 |
| 60 | 82 | 14 | 75 | 1 | 0 |
| 70 | 81 | 14 | 28 | 1 | 0 |
| 80 | 37 | 14 | 44 | 1 | 0 |
| 81 | 96 | 14 | 44 | 1 | 0 |

At the bottom left, it says 'Table : TF_Commande (878 lignes)'. At the bottom right, it says 'Mise à jour disponible (cliquez pour la télécharger)'.

FIGURE 9.1 – Table de faits TF_Commande créée

9.3 Crédation du modèle en étoile

9.3.1 Établissement des relations

Dans la Vue Modèle, créer les relations entre la table de faits et les dimensions :

Procédure de création des relations

1. Aller dans la Vue Modèle (icône en bas à gauche)
2. Pour chaque relation :
 - Glisser-déposer la clé de la table de faits vers la clé de la dimension
 - Vérifier la cardinalité (N :1)
 - Vérifier la direction du filtre croisé (Simple)

Relations à créer

| Depuis (Fait) | Vers (Dimension) | Cardinalité | Filtre |
|----------------|------------------|-------------|--------|
| id_temps | DimTemps | 1 :1 | Simple |
| id_seqEmployee | DimEmployee | N :1 | Simple |
| id_seqClient | DimClient | N :1 | Simple |

TABLE 9.2 – Relations du modèle en étoile

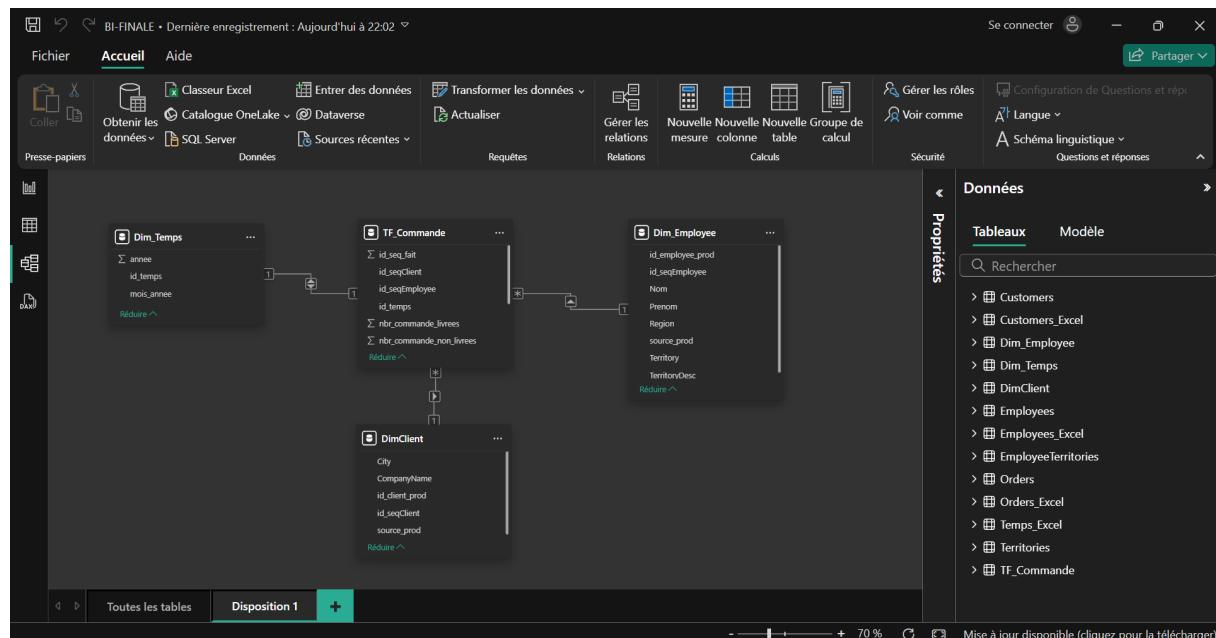


FIGURE 9.2 – Modèle en étoile complet

9.3.2 Vérification du modèle

Vérifier que :

- Toutes les relations sont actives (trait continu)
- Les cardinalités sont correctes (N :1)

- Les filtres croisés fonctionnent
- Aucun message d'erreur n'apparaît

9.4 Conclusion

L'entrepôt de données est maintenant complet :

- **3 dimensions** : DimEmployee, DimClient, DimTemps
- **1 table de faits** : TF_Commande
- **Modèle en étoile** : Relations établies
- **Données intégrées** : SQL Server + Excel (Access)

L'entrepôt est prêt pour la création de mesures DAX et de visualisations.

Chapitre 10

Visualisations et graphiques

10.1 Introduction

Ce chapitre présente les visualisations créées pour analyser les données de l'entrepôt. Nous avons développé 6 visualisations : 4 avec Python pour des analyses statistiques avancées, et 2 avec Power BI natif pour bénéficier de l'interactivité.

10.2 Visualisations Python

Les visualisations Python permettent de créer des graphiques statistiques personnalisés avec les bibliothèques matplotlib et pandas.

10.2.1 Configuration Python dans Power BI

Avant de créer les visuels Python :

1. Installer Python avec les bibliothèques : pandas, matplotlib
2. Dans Power BI : Fichier → Options et paramètres → Options Python
3. Sélectionner le répertoire d'installation de Python

10.2.2 Visuel Python 1 : Volume des commandes par mois

Objectif

Afficher l'évolution mensuelle du volume de commandes en distinguant celles livrées des non livrées pour identifier les tendances et problèmes de livraison.

Données utilisées

Colonnes : id_temps, nbr_commande_livrees, nbr_commande_non_livrees, mois_annee

Code Python

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Vérification des colonnes
```

```

5    colonnes_requises = ['id_temps', 'nbr_commande_livrees',
6                           'nbr_commande_non_livrees', 'mois_annee']
7    colonnes_manquantes = [col for col in colonnes_requises
8                             if col not in dataset.columns]
9
10   if colonnes_manquantes:
11       fig, ax = plt.subplots(figsize=(10, 2))
12       ax.text(0.5, 0.5,
13               f"Glissez les colonnes :\n{', '.join(colonnes_manquantes
14                                         )}"),
15               ha='center', va='center', fontsize=12, color='red',
16               bbox=dict(boxstyle="round, pad=0.5",
17                         facecolor="yellow", alpha=0.7))
18       ax.axis('off')
19       plt.show()
20   else:
21       # Aggregation par mois
22       commandes_par_mois = dataset.groupby('mois_annee').agg({
23           'nbr_commande_livrees': 'sum',
24           'nbr_commande_non_livrees': 'sum'
25       }).reset_index()
26
27       commandes_par_mois['total_commandes'] = (
28           commandes_par_mois['nbr_commande_livrees'] +
29           commandes_par_mois['nbr_commande_non_livrees'])
30
31       # Trier par date
32       try:
33           commandes_par_mois['date_sort'] = pd.to_datetime(
34               commandes_par_mois['mois_annee'] + '/01',
35               format='%m/%Y/%d'
36           )
37           commandes_par_mois = commandes_par_mois.sort_values(
38               'date_sort')
39       except:
40           commandes_par_mois = commandes_par_mois.sort_values(
41               'mois_annee')
42
43       # Creation du graphique
44       fig, ax = plt.subplots(figsize=(14, 7))
45       x = range(len(commandes_par_mois))
46       mois_labels = commandes_par_mois['mois_annee'].tolist()
47
48       # Barres empilees
49       ax.bar(x, commandes_par_mois['nbr_commande_livrees'],
50              label='Commandes Livrees', color='green', alpha=0.7)
51       ax.bar(x, commandes_par_mois['nbr_commande_non_livrees'],
52              bottom=commandes_par_mois['nbr_commande_livrees'],
53              label='Commandes Non Livrees', color='red', alpha=0.7)

```

```

53     ax.set_xlabel('Mois', fontsize=12)
54     ax.set_ylabel('Nombre de Commandes', fontsize=12)
55     ax.set_title('VOLUME DES COMMANDES PAR MOIS',
56                   fontsize=14, fontweight='bold')
57     ax.set_xticks(x)
58     ax.set_xticklabels(mois_labels, rotation=45, ha='right')
59     ax.legend()
60     ax.grid(True, alpha=0.3)
61
62     # Totaux au-dessus des barres
63     for i, total in enumerate(commandes_par_mois['total_commandes'],
64                               []):
64         ax.text(i, total + (total*0.01), f'{total:,}',
65                 ha='center', va='bottom', fontsize=9)
66
67     # Statistiques
68     total_periode = commandes_par_mois['total_commandes'].sum()
69     stats_text = f"Total période: {total_periode:,} commandes"
70     plt.figtext(0.02, 0.02, stats_text, fontsize=10,
71                 bbox=dict(boxstyle="round", pad=0.5",
72                            facecolor="lightgray", alpha=0.8))
73
74     plt.tight_layout()
75     plt.show()

```

Listing 10.1 – Volume des commandes par mois

Explication

- **Vérification** : Contrôle de la présence des colonnes nécessaires
- **Agrégation** : Groupement par mois avec somme des commandes
- **Tri** : Organisation chronologique des mois
- **Graphique** : Barres empilées (vert = livrées, rouge = non livrées)
- **Annotations** : Totaux affichés sur chaque barre


```
18     ax.axis('off')
19     plt.show()
20 else:
21     # Agregation par mois
22     commandes_par_mois = dataset.groupby('mois_annee').agg({
23         'nbr_commande_livrees': 'sum',
24         'nbr_commande_non_livrees': 'sum'
25     }).reset_index()
26
27     commandes_par_mois['total_commandes'] = (
28         commandes_par_mois['nbr_commande_livrees'] +
29         commandes_par_mois['nbr_commande_non_livrees']
30     )
31
32     commandes_par_mois['taux_livraison'] = (
33         commandes_par_mois['nbr_commande_livrees'] /
34         commandes_par_mois['total_commandes'] * 100
35     ).round(1)
36
37     # Trier par date
38 try:
39     commandes_par_mois['date_sort'] = pd.to_datetime(
40         commandes_par_mois['mois_annee'] + '/01',
41         format='%m/%Y/%d'
42     )
43     commandes_par_mois = commandes_par_mois.sort_values(
44         'date_sort')
45 except:
46     commandes_par_mois = commandes_par_mois.sort_values(
47         'mois_annee')
48
49     # Creation du graphique
50     fig, ax = plt.subplots(figsize=(14, 7))
51     x = range(len(commandes_par_mois))
52     mois_labels = commandes_par_mois['mois_annee'].tolist()
53     taux = commandes_par_mois['taux_livraison'].tolist()
54
55     # Ligne avec points
56     ax.plot(x, taux, marker='o', linewidth=2, color='blue',
57             markersize=8, label='Taux de Livraison')
58
59     # Zone remplie sous la ligne
60     ax.fill_between(x, taux, alpha=0.2, color='blue')
61
62     # Ligne d'objectif
63     ax.axhline(y=80, color='red', linestyle='--',
64                 linewidth=2, alpha=0.7, label='Objectif 80%')
65
66     # Mise en forme
67     ax.set_xlabel('Mois', fontsize=12)
68     ax.set_ylabel('Taux de Livraison (%)', fontsize=12)
```

```

67     ax.set_title('TAUX DE LIVRAISON PAR MOIS',
68                 fontsize=14, fontweight='bold')
69     ax.set_xticks(x)
70     ax.set_xticklabels(mois_labels, rotation=45, ha='right')
71     ax.set ylim([0, 105])
72     ax.grid(True, alpha=0.3, linestyle='--')
73     ax.legend(loc='upper right')

74
75     # Valeurs sur les points
76     for i, (taux_val, mois) in enumerate(zip(taux, mois_labels)):
77         ax.text(i, taux_val + 2, f'{taux_val}%',
78                 ha='center', va='bottom', fontsize=9, fontweight='bold',
79                 bbox=dict(boxstyle="round", pad=0.2,
80                           facecolor="white", alpha=0.8))

81
82     # Indicateurs de performance
83     taux_moyen = commandes_par_mois['taux_livraison'].mean()
84     au_dessus_objectif = (commandes_par_mois['taux_livraison'] >=
85                             80).sum()
86     total_mois = len(commandes_par_mois)

87     stats_text = f"""
88     - Taux moyen: {taux_moyen:.1f}%
89     - Mois >= 80%: {au_dessus_objectif}/{total_mois}
90     ({au_dessus_objectif/total_mois*100:.0f}%)"
91     - Meilleur: {commandes_par_mois.loc[commandes_par_mois['
92         taux_livraison'].idxmax(), 'mois_annee']}
93     ({commandes_par_mois['taux_livraison'].max():.1f}%)"
94     - Pire: {commandes_par_mois.loc[commandes_par_mois['taux_livraison
95         ].idxmin(), 'mois_annee']}
96     ({commandes_par_mois['taux_livraison'].min():.1f}%)"""

97     plt.figtext(0.02, 0.02, stats_text, fontsize=10,
98                 bbox=dict(boxstyle="round", pad=0.5",
99                           facecolor="lightgray", alpha=0.8))

100
101     # Coloration de fond par performance
102     for i in x:
103         if taux[i] >= 90:
104             ax.axvspan(i-0.4, i+0.4, alpha=0.1, color='green')
105         elif taux[i] >= 80:
106             ax.axvspan(i-0.4, i+0.4, alpha=0.1, color='lightgreen')
107         elif taux[i] >= 70:
108             ax.axvspan(i-0.4, i+0.4, alpha=0.1, color='orange')
109         else:
110             ax.axvspan(i-0.4, i+0.4, alpha=0.1, color='red')

111     plt.tight_layout()
112     plt.show()

```

Listing 10.2 – Taux de livraison par mois

Explication

- **Calcul du taux** : $(\text{Commandes livrées} / \text{Total}) \times 100$
- **Ligne d'objectif** : Barre horizontale rouge à 80%
- **Zones colorées** : Fond coloré selon performance (vert 90%, orange 70-80%, rouge <70%)
- **Statistiques** : Taux moyen, meilleur/pire mois, atteinte objectif
- **Annotations** : Valeurs de taux sur chaque point

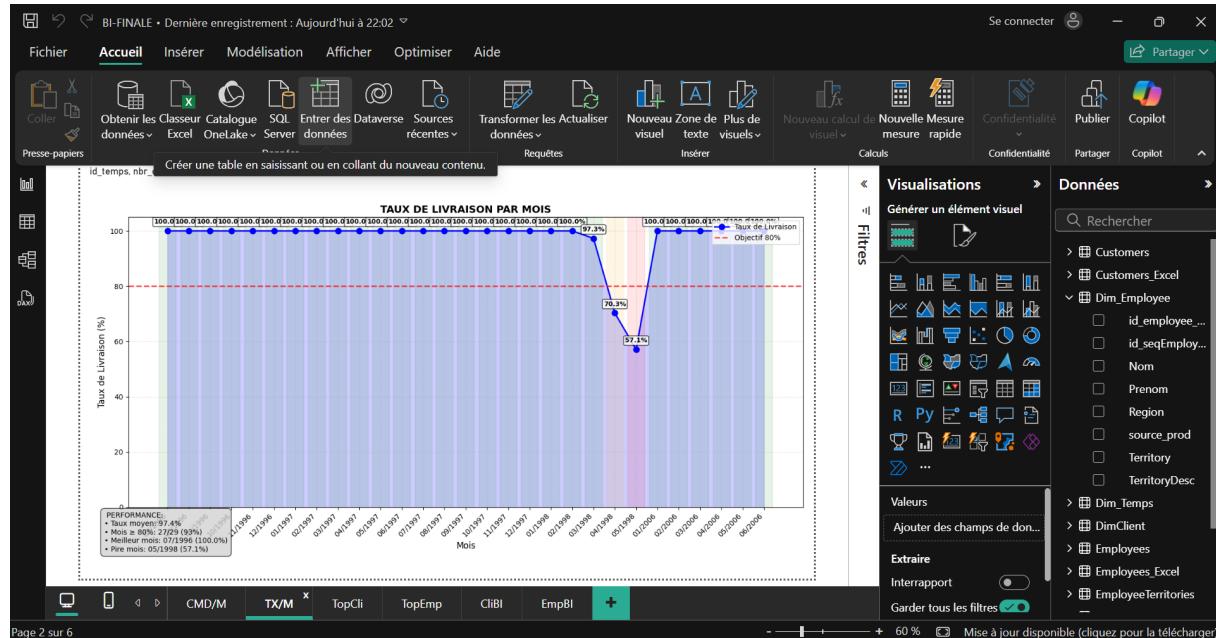


FIGURE 10.2 – Taux de livraison par mois

10.2.4 Visuel Python 3 : Top 10 des clients

Objectif

Identifier les 10 clients ayant passé le plus de commandes pour orienter les stratégies commerciales et évaluer la concentration de l'activité.

Données utilisées

Colonnes : CompanyName (DimClient), Nombre Total Commandes (mesure calculée)

Mesure DAX requise

```

1 Nombre Total Commandes =
2     SUM(TF_Commande[nbr_commande_livrees]) +
3     SUM(TF_Commande[nbr_commande_non_livrees])

```

Listing 10.3 – Mesure pour le nombre total de commandes

Code Python

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 print("====TOP 10 CLIENTS PAR NOMBRE DE COMMANDES===")
6
7 # Vérifier les données
8 print(f"    Données recues:{len(dataset):,} lignes")
9 print(f"    Colonnes:{list(dataset.columns)}")
10
11 # Vérifier la mesure
12 if 'NombreTotalCommandes' not in dataset.columns:
13     print("    Glissez la mesure avec CompanyName")
14 else:
15     # Calculs
16     total_general = dataset['NombreTotalCommandes'].sum()
17
18     # Top 10
19     top_10 = dataset.nlargest(10, 'NombreTotalCommandes').copy()
20     top_10_display = top_10.sort_values('NombreTotalCommandes',
21                                         ascending=True)
22
23     top10_total = top_10['NombreTotalCommandes'].sum()
24     pourcentage = (top10_total / total_general * 100)
25
26     print(f"\nSTATISTIQUES:")
27     print(f"    Clients totaux:{len(dataset)}")
28     print(f"    Commandes totales:{int(total_general):,}")
29     print(f"    Top 10:{pourcentage:.1f}%")
30
31     # Graphique
32     fig, ax = plt.subplots(figsize=(14, 8))
33
34     # Barres horizontales
35     y_pos = np.arange(len(top_10_display))
36     colors = plt.cm.Blues(np.linspace(0.4, 0.9, len(top_10_display)))
37
38     bars = ax.barh(y_pos, top_10_display['NombreTotalCommandes'],
39                     color=colors, edgecolor='darkblue', height=0.7)
40
41     # Ajouter valeurs
42     for i, (bar, total) in enumerate(zip(bars,
43                                           top_10_display['NombreTotalCommandes'])):
44         pourcentage_client = (total / top10_total * 100)
45
46         # Texte à droite
47         ax.text(bar.get_width() + bar.get_width() * 0.01,
```

```

48         bar.get_y() + bar.get_height()/2,
49         f"{{int(total)}},\n{{pourcentage_client:.1f}}%",
50         va='center', fontsize=10, fontweight='bold')
51
52     # Texte interieur pour grandes barres
53     if bar.get_width() > top_10_display['Nombre_Total_Commandes']
54         ].max() * 0.3:
55         ax.text(bar.get_width() * 0.02,
56                 bar.get_y() + bar.get_height()/2,
57                 top_10_display.iloc[i]['CompanyName'],
58                 va='center', fontsize=9,
59                 color='white', fontweight='bold')
60
61     # Configuration
62     ax.set_yticks(y_pos)
63     ax.set_yticklabels(top_10_display['CompanyName'], fontsize=11)
64     ax.set_xlabel('Nombre_de_Commandes', fontsize=12, fontweight='bold')
65     ax.set_title(f' TOP 10 CLIENTS - {{int(top10_total)} COMMANDES
66 {{pourcentage:.1f}}% du total)', fontsize=16, fontweight='bold', pad=20)
67
68     # Ligne moyenne
69     moyenne = top_10['Nombre_Total_Commandes'].mean()
70     ax.axvline(x=moyenne, color='red', linestyle='--', linewidth=2,
71                 alpha=0.7, label=f'Moyenne: {{int(moyenne)} cmd}')
72
73     # Grille et style
74     ax.grid(True, axis='x', alpha=0.3, linestyle=':')
75     ax.spines['top'].set_visible(False)
76     ax.spines['right'].set_visible(False)
77     ax.legend(loc='lower_right')
78
79     # Statistiques
80     stats_text = f"""
81     - Clients: {{len(dataset)}}
82     - Total commandes: {{int(total_general)}}
83     - Moyenne/client: {{dataset['Nombre_Total_Commandes'].mean():.1f}}
84     - Meilleur: {{int(top_10['Nombre_Total_Commandes'].max())}}
85     - Top 10: {{int(top10_total)}}, {{pourcentage:.1f}}%
86 """
87
88     plt.figtext(0.02, 0.02, stats_text, fontsize=10,
89                 bbox=dict(boxstyle="round", pad=0.5,
90                           facecolor="lightblue", alpha=0.8))
91
92     plt.tight_layout(rect=[0, 0.1, 1, 0.95])
93
94     # Afficher resultats
95     print("\nTOP 10 CLIENTS:")
96     for i, row in top_10.iterrows():

```

```

96     client_pct = (row['NombreTotalCommandes'] / total_general
97         * 100)
98     print(f"{row['CompanyName']}:{int(row['NombreTotal'
99         Commandes'])}:{client_pct:.1f}%)")
100    plt.show()

```

Listing 10.4 – Top 10 clients

Explication

- **Tri et sélection** : Top 10 clients par nombre de commandes
- **Barres dégradées** : Couleur bleue avec gradient selon le classement
- **Annotations doubles** : Valeurs absolues et pourcentages
- **Ligne moyenne** : Référence pour comparer les clients
- **Statistiques** : Concentration du top 10 dans le total

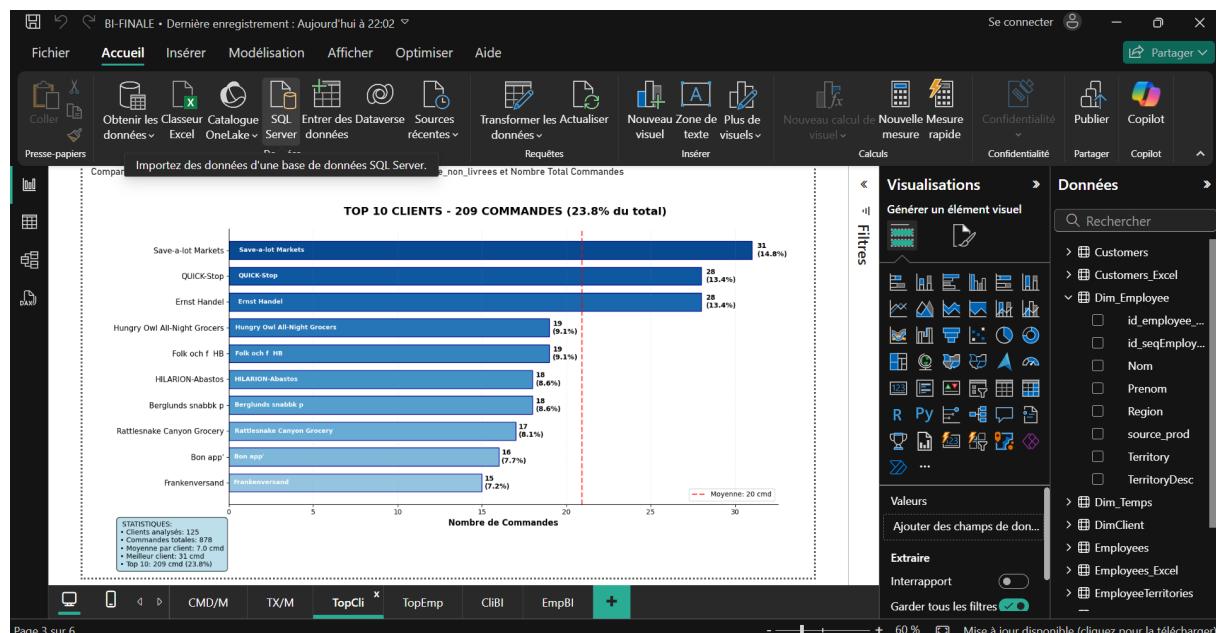


FIGURE 10.3 – Top 10 des clients

10.2.5 Visuel Python 4 : Top 10 des employés

Objectif

Identifier les 10 employés ayant traité le plus de commandes pour évaluer les performances individuelles et la répartition de la charge de travail.

Données utilisées

Colonnes : Nom, Prenom (DimEmployee), Nombre Total Commandes (mesure calculée)

Code Python

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 print("====TOP 10 EMPLOYES PAR NOMBRE DE COMMANDES====")
6
7 # Vérifier les données
8 print(f"Nombre d'Employés : {len(dataset)} lignes")
9 print(f"Colonnes : {list(dataset.columns)}")
10
11 if 'NombreTotalCommandes' not in dataset.columns:
12     print("Glissez 'NombreTotalCommandes' dans Power BI")
13 else:
14     # Vérifier colonnes nom/prenom
15     has_names = 'Nom' in dataset.columns and 'Prenom' in dataset.
16         columns
17
18     if has_names:
19         dataset['Employe'] = dataset['Prenom'] + ' ' + dataset['Nom']
20             ]
21         label_col = 'Employe'
22     elif 'Nom' in dataset.columns:
23         label_col = 'Nom'
24     elif 'id_seqEmployee' in dataset.columns:
25         dataset['Employe'] = 'Emp.' + dataset['id_seqEmployee'].
26             astype(str)
27         label_col = 'Employe'
28     else:
29         non_measure_cols = [col for col in dataset.columns
30                         if col != 'NombreTotalCommandes']
31         label_col = non_measure_cols[0] if non_measure_cols else 'index'
32
33 # Calculs
34 total_general = dataset['NombreTotalCommandes'].sum()
35
36 # Top 10
37 top_10 = dataset.nlargest(10, 'NombreTotalCommandes').copy()
38 top_10_display = top_10.sort_values('NombreTotalCommandes',
39                                     ascending=True)
40
41 top10_total = top_10['NombreTotalCommandes'].sum()
42 pourcentage = (top10_total / total_general * 100)
43
44 print(f"\nSTATISTIQUES:")
45 print(f"    Nombre d'Employés : {len(dataset)}")
46 print(f"    Total commandes : {int(total_general)}")
47 print(f"    Top 10 : {pourcentage:.1f}%")
```

```
46 # Graphique
47 fig, ax = plt.subplots(figsize=(14, 8))
48
49 # Barres horizontales
50 y_pos = np.arange(len(top_10_display))
51 colors = plt.cm.Greens(np.linspace(0.4, 0.9, len(top_10_display)))
52
53 bars = ax.bars(y_pos, top_10_display['Nombre de Commandes'],
54                  color=colors, edgecolor='darkgreen', height=0.7)
55
56 # Valeurs
57 for i, (bar, total) in enumerate(zip(bars,
58                                   top_10_display['Nombre de Commandes'])):
59     pourcentage_emp = (total / top10_total * 100)
60
61     # Texte droite
62     ax.text(bar.get_width() + bar.get_width() * 0.01,
63             bar.get_y() + bar.get_height()/2,
64             f'{int(total)}\n{pourcentage_emp:.1f}%',
65             va='center', fontsize=10, fontweight='bold')
66
67     # Nom interieur si place
68     if bar.get_width() > top_10_display['Nombre de Commandes'].max() * 0.3:
69         label = top_10_display.iloc[i][label_col]
70         if len(str(label)) > 20:
71             label = str(label)[:17] + "..."
72         ax.text(bar.get_width() * 0.02,
73                 bar.get_y() + bar.get_height()/2,
74                 label, va='center', fontsize=9,
75                 color='white', fontweight='bold')
76
77 # Configuration
78 ax.set_yticks(y_pos)
79
80 # Labels Y limites
81 y_labels = []
82 for i in range(len(top_10_display)):
83     label = str(top_10_display.iloc[i][label_col])
84     if len(label) > 25:
85         label = label[:22] + "..."
86     y_labels.append(label)
87
88 ax.set_yticklabels(y_labels, fontsize=11)
89 ax.set_xlabel('Nombre de Commandes', fontsize=12, fontweight='bold')
90 ax.set_title(f'TOP 10 EMPLOYES - {int(top10_total)}, COMMANDES
91           ({pourcentage:.1f}% du total)',
```

```

92         fontsize=16, fontweight='bold', pad=20)
93
94     # Ligne moyenne
95     moyenne = top_10['Nombre_Total_Commandes'].mean()
96     ax.axvline(x=moyenne, color='red', linestyle='--', linewidth=2,
97                  alpha=0.7, label=f'Moyenne:{int(moyenne)}')
98
99     # Grille et style
100    ax.grid(True, axis='x', alpha=0.3, linestyle=':')
101    ax.spines['top'].set_visible(False)
102    ax.spines['right'].set_visible(False)
103    ax.legend(loc='lower_right')
104
105    # Statistiques
106    stats_text = f"""
107        STATISTIQUES EMPLOYES:
108        {len(dataset)}
109        Total:{int(total_general)}
110        Moyenne/employe:{dataset['Nombre_Total_Commandes'].mean():.1f}
111        Meilleur:{int(top_10['Nombre_Total_Commandes'].max())}
112        Top10:{int(top10_total)} ({pourcentage:.1f}%)"""
113
114    plt.figtext(0.02, 0.02, stats_text, fontsize=10,
115                bbox=dict(boxstyle="round", pad=0.5",
116                           facecolor="lightgreen", alpha=0.8))
117
118    plt.tight_layout(rect=[0, 0.1, 1, 0.95])
119
120    # Afficher
121    print("\nTOP 10 EMPLOYES:")
122    for i, row in top_10.iterrows():
123        emp_pct = (row['Nombre_Total_Commandes'] / total_general *
124                    100)
125        emp_label = row[label_col]
126        print(f'{emp_label}:{int(row["Nombre_Total_Commandes"])} '
127              f'({emp_pct:.1f}%)')
128
129    plt.show()

```

Listing 10.5 – Top 10 employés

Explication

- **Gestion des noms** : Création automatique du nom complet (Prénom + Nom)
- **Barres vertes** : Gradient de couleur selon le classement
- **Même structure** : Identique au visuel clients pour cohérence
- **Ligne moyenne** : Permet d'identifier les sur-performeurs
- **Statistiques** : Concentration des commandes sur le top 10

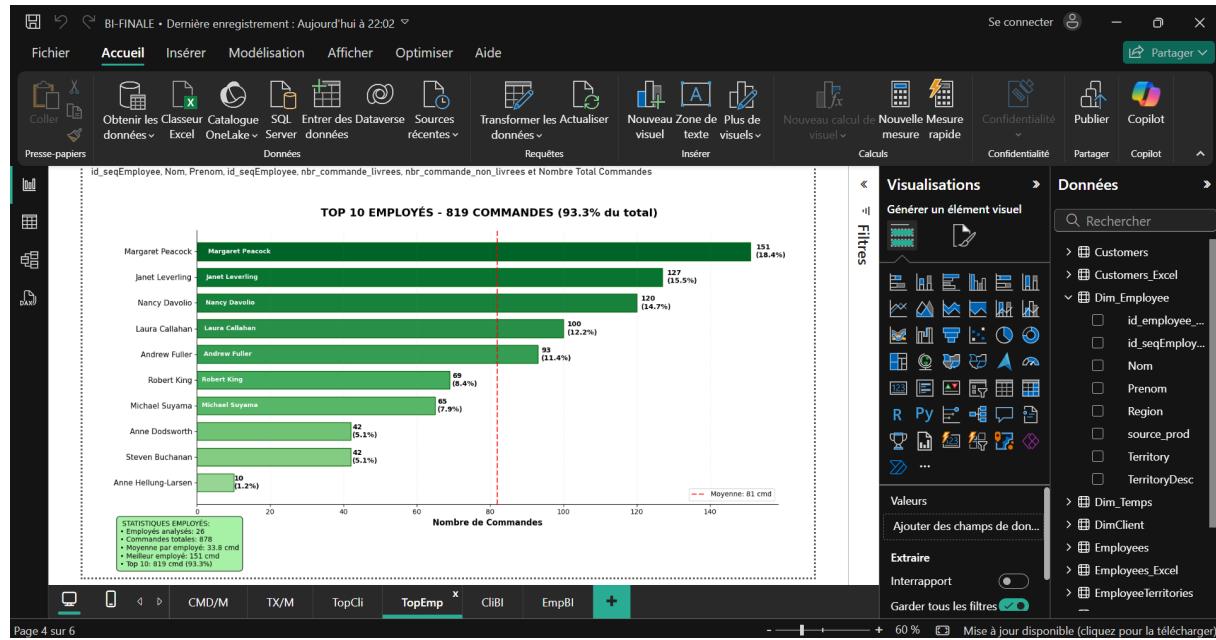


FIGURE 10.4 – Top 10 des employés

10.3 Visualisations Power BI natives

Les visualisations Power BI natives offrent l'avantage majeur de l'interactivité : les filtres croisés permettent de sélectionner un élément dans un visuel et voir automatiquement tous les autres visuels se mettre à jour.

10.3.1 Mesure DAX utilisée

Pour les visuels Power BI, nous avons créé une mesure DAX :

```

1 Nombre Total Commandes =
2     SUM(TF_Commande[nbr_commande_livrees]) +
3     SUM(TF_Commande[nbr_commande_non_livrees])

```

Listing 10.6 – Mesure Nombre Total Commandes

10.3.2 Visuel Power BI 1 : Top clients avec répartition

Objectif

Afficher le classement des meilleurs clients et leur répartition globale avec interaction dynamique entre les deux visuels.

Composition

Ce visuel combine deux graphiques :

- **Graphique en barres** : Classement des clients par nombre de commandes
- **Graphique en secteurs (camembert)** : Répartition proportionnelle des commandes par client

Création du graphique en barres

1. Dans le panneau Visualisations, sélectionner **Graphique à barres groupées**
2. Configuration :
 - **Axe Y** : CompanyName (depuis DimClient)
 - **Axe X** : Nombre Total Commandes (mesure)
 - **Tri** : Décroissant par valeur
3. Formatage :
 - Couleur des barres : Bleu
 - Afficher les étiquettes de données
 - Titre : "Top Clients"
4. Le graphique affiche automatiquement les clients dans l'ordre décroissant

Création du graphique en secteurs

1. Dans le panneau Visualisations, sélectionner **Graphique en secteurs**
2. Configuration :
 - **Légende** : CompanyName
 - **Valeurs** : Nombre Total Commandes
3. Formatage :
 - Afficher les pourcentages
 - Afficher les étiquettes
 - Titre : "Répartition des commandes"

Interactivité

L'avantage principal de Power BI : **les filtres croisés automatiques**

- **Action** : Cliquer sur un client dans le graphique en barres
- **Résultat** : Le secteur correspondant dans le camembert se met en évidence
- **Effet** : Les autres secteurs deviennent semi-transparent
- **Utilité** : Visualisation immédiate de la part du client sélectionné

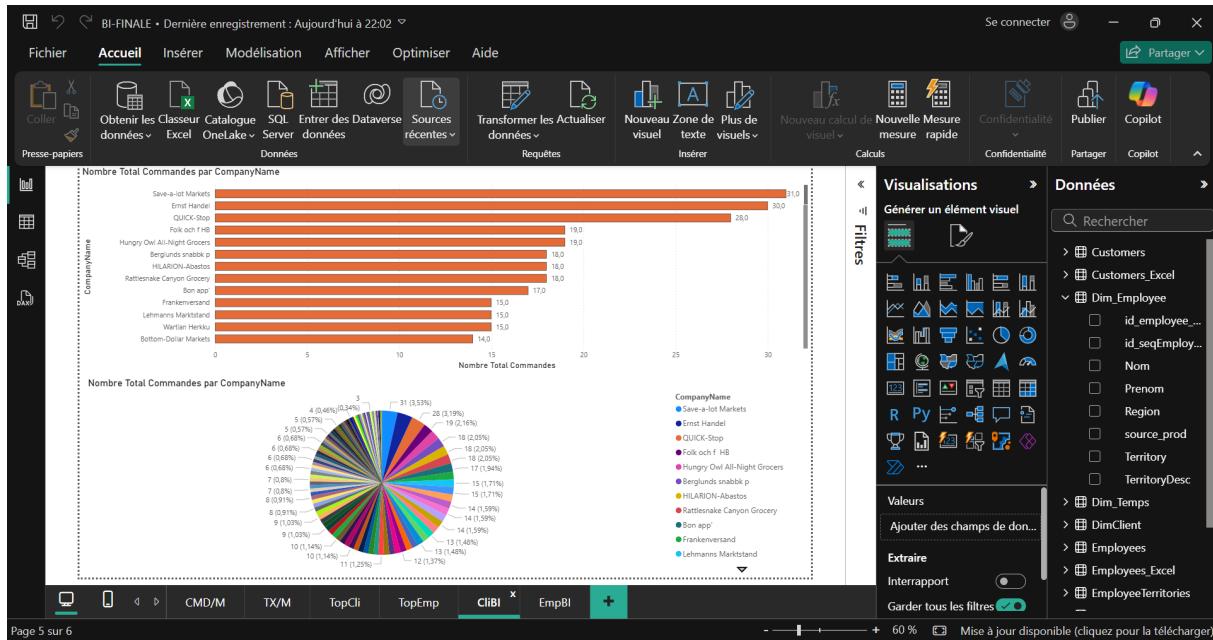


FIGURE 10.5 – Top clients avec répartition - Power BI natif

10.3.3 Visuel Power BI 2 : Top employés avec répartition

Objectif

Afficher le classement des meilleurs employés et leur répartition avec la même interactivité que pour les clients.

Composition

Même principe que pour les clients :

- **Graphique en barres** : Top employés par commandes traitées
- **Graphique en secteurs** : Répartition des commandes par employé

Création du graphique en barres

1. Sélectionner **Graphique à barres groupées**
2. Configuration :
 - Axe Y : Nom et Prénom (depuis DimEmployee)
 - Axe X : Nombre Total Commandes
 - Tri : Décroissant par valeur
3. Formatage :
 - Couleur : Vert
 - Étiquettes de données activées
 - Titre : "Top Employés"
4. Le graphique affiche les employés dans l'ordre décroissant

Création du graphique en secteurs

1. Sélectionner **Graphique en secteurs**

2. Configuration :
 - **Légende** : Nom complet employé (Prénom + Nom)
 - **Valeurs** : Nombre Total Commandes
3. Formatage identique au camembert clients

Interactivité

Fonctionnement identique aux visuels clients :

- Clic sur un employé dans les barres
- Mise en évidence automatique dans le camembert
- Visualisation de sa contribution au total

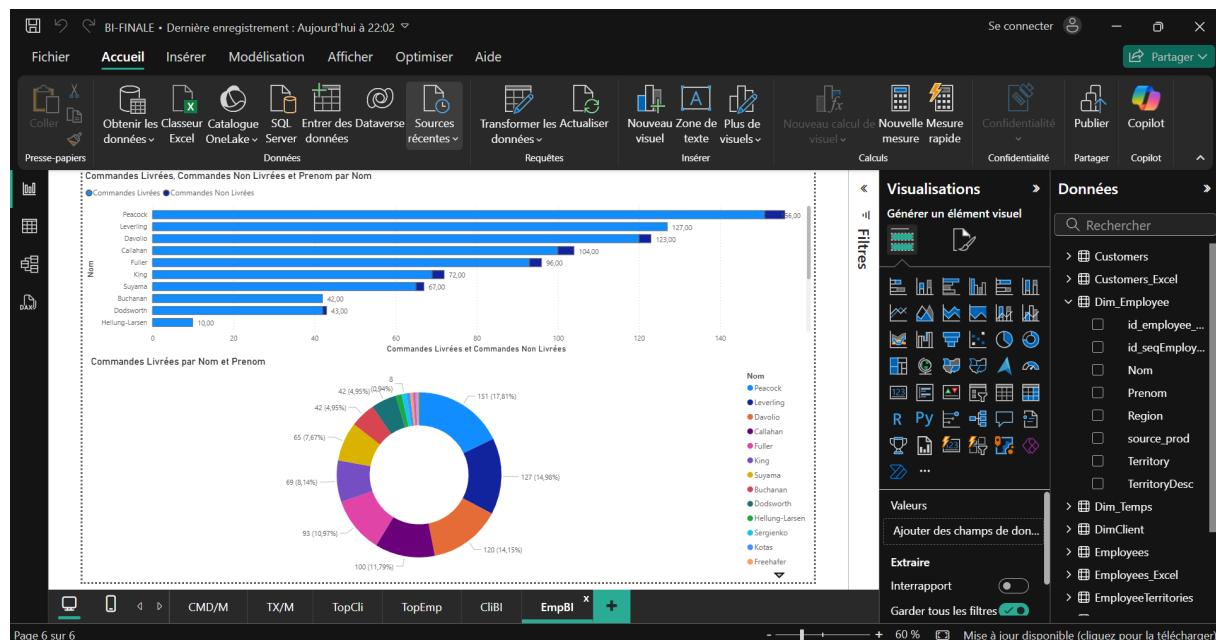


FIGURE 10.6 – Top employés avec répartition - Power BI natif

10.4 Comparaison Python vs Power BI natif

10.4.1 Différences clés observées

| Critère | Python | Power BI natif |
|------------------|--------------------------|---------------------------------|
| Interactivité | Statique (image fixe) | Dynamique avec filtres croisés |
| Personnalisation | Totale via code | Limitée aux options disponibles |
| Création | Code Python nécessaire | Glisser-déposer (no-code) |
| Performance | Plus lent (rendu Python) | Rapide et optimisé |
| Statistiques | Affichage avancé intégré | Nécessite mesures DAX |
| Publication | Desktop uniquement | Compatible Power BI Service |

TABLE 10.1 – Comparaison des deux approches

10.4.2 Avantage de l'interactivité Power BI

L'exemple le plus parlant de l'interactivité :

1. **Situation** : Un utilisateur veut analyser un client spécifique
2. **Avec Python** : L'utilisateur doit lire le graphique statiquement
3. **Avec Power BI** :
 - Clic sur le client dans le graphique en barres
 - → Le camembert met en évidence ce client automatiquement
 - → Tous les autres visuels de la page se filtrent sur ce client
 - → Les KPI se mettent à jour instantanément

Cette interactivité transforme un rapport statique en outil d'exploration dynamique des données.

10.4.3 Conclusion sur les approches

Quand utiliser Python :

- Analyses statistiques avancées
- Graphiques personnalisés spécifiques
- Rapports statiques avec calculs complexes
- Besoins de bibliothèques spécialisées (seaborn, plotly)

Quand utiliser Power BI natif :

- Dashboards interactifs pour exploration
- Besoins de filtres croisés entre visuels
- Publication sur Power BI Service
- Rapidité de création (no-code)
- Performance optimale

Notre approche : Combiner les deux pour maximiser les avantages : Python pour les analyses détaillées, Power BI natif pour l'exploration interactive.

10.5 Conclusion du chapitre

Les visualisations créées permettent une analyse complète des données de l'entrepôt :

- Les 4 visuels Python offrent des analyses statistiques détaillées avec annotations et statistiques avancées
- Les 2 visuels Power BI natifs permettent une exploration interactive et dynamique
- La combinaison des deux approches maximise les capacités d'analyse

Le choix de l'outil dépend du besoin : analyses statiques approfondies avec Python, ou exploration interactive avec Power BI natif.

Chapitre 11

Conclusion

Ce projet de Business Intelligence avec Power BI nous a permis de mettre en œuvre l'ensemble du cycle de vie d'une solution décisionnelle, depuis l'extraction des données jusqu'à leur visualisation interactive.

11.1 Objectifs atteints

Nous avons réussi à créer un entrepôt de données complet en modèle en étoile, intégrant des données hétérogènes provenant de deux sources distinctes : SQL Server et Microsoft Access. Le processus ETL développé avec Power Query a permis de :

- **Extraire** efficacement les données depuis les bases Northwind
- **Transformer** ces données en créant trois dimensions (Employee, Client, Temps) avec des clés de substitution
- **Charger** une table de faits (TF_Commande) contenant les métriques essentielles sur les commandes

L'architecture en étoile mise en place facilite les requêtes analytiques et garantit la cohérence des analyses à travers l'utilisation de clés de substitution qui permettent de tracer l'origine des données et de gérer l'évolution dimensionnelle.

11.2 Apports techniques et compétences développées

Ce projet nous a permis de maîtriser plusieurs aspects techniques :

11.2.1 Power Query et le langage M

La création de scripts complexes pour fusionner, transformer et nettoyer les données a développé notre compréhension approfondie du processus ETL et des bonnes pratiques de préparation des données.

11.2.2 Modélisation dimensionnelle

La conception d'un modèle en étoile avec ses dimensions et sa table de faits nous a familiarisés avec les concepts fondamentaux du datawarehousing et l'importance d'une architecture bien pensée.

11.2.3 DAX (Data Analysis Expressions)

La création de mesures calculées a enrichi nos compétences en analyse de données et en formulation de KPI pertinents.

11.2.4 Visualisation hybride

L'utilisation combinée de Python (matplotlib, pandas) et des visuels natifs de Power BI nous a permis de comprendre les forces et limites de chaque approche. Python offre une personnalisation avancée et des analyses statistiques poussées, tandis que Power BI natif excelle dans l'interactivité et la facilité d'utilisation.

11.3 Comparaison des approches de visualisation

L'un des enseignements majeurs de ce projet concerne le choix entre visualisations Python et Power BI natif :

- **Python** s'impose pour les analyses statistiques avancées, les graphiques hautement personnalisés et les rapports statiques nécessitant des calculs complexes
- **Power BI natif** brille par son interactivité, ses filtres croisés automatiques et sa rapidité d'exécution, idéal pour des dashboards exploratoires

Notre approche hybride, combinant 4 visualisations Python et 2 visualisations Power BI natives, maximise les avantages des deux mondes et offre une solution d'analyse complète et flexible.

11.4 Avantages de Power BI

Ce projet a confirmé les nombreux atouts de Power BI comme plateforme BI complète :

- **Solution intégrée** : ETL, modélisation et visualisation dans un seul outil
- **Accessibilité** : Interface intuitive facilitant l'adoption par les utilisateurs métiers
- **Performance** : Moteur d'analyse rapide capable de traiter efficacement de gros volumes
- **Écosystème Microsoft** : Intégration native avec SQL Server, Excel et l'ensemble des outils Microsoft
- **Évolutivité** : De la version Desktop gratuite au Service cloud pour le partage et la collaboration

11.5 Perspectives et améliorations futures

Plusieurs axes d'amélioration pourraient enrichir ce projet :

11.5.1 Enrichissement du modèle

Ajout de dimensions supplémentaires (Produits, Fournisseurs, Catégories) pour des analyses plus granulaires et l'intégration de dimensions temporelles plus détaillées (jour, semaine, trimestre).

11.5.2 Calculs avancés

Développement de mesures DAX plus complexes comme les calculs de croissance, les comparaisons temporelles (année N vs N-1), les moyennes mobiles ou les analyses de cohortes.

11.5.3 Optimisation des performances

Mise en place de tables agrégées pour accélérer les requêtes sur de gros volumes, et optimisation des relations et cardinalités.

11.5.4 Automatisation

Planification des actualisations automatiques des données et mise en place d'alertes sur les KPI critiques.

11.5.5 Publication et partage

Déploiement sur Power BI Service pour permettre l'accès aux rapports depuis n'importe où et la collaboration en temps réel entre utilisateurs.

11.5.6 Sécurité

Implémentation de la sécurité au niveau des lignes (RLS) pour contrôler l'accès aux données selon les profils utilisateurs.

11.6 Conclusion finale

Ce projet démontre qu'avec Power BI, il est possible de mettre en place rapidement et efficacement une solution de Business Intelligence complète et professionnelle. La combinaison d'un ETL robuste avec Power Query, d'un modèle dimensionnel bien conçu et de visualisations pertinentes permet de transformer des données brutes en informations actionables pour la prise de décision.

Au-delà des compétences techniques acquises, ce projet illustre l'importance croissante de la Business Intelligence dans le contexte actuel où les données constituent un actif stratégique majeur pour les organisations. La capacité à extraire, modéliser et visualiser efficacement les données devient une compétence essentielle pour tout professionnel évoluant dans l'environnement data-driven d'aujourd'hui.

Les connaissances et méthodologies développées durant ce projet constituent une base solide pour aborder des problématiques BI plus complexes et pour contribuer efficacement à la transformation digitale des entreprises. Power BI s'est révélé être un outil puissant, accessible et évolutif, parfaitement adapté aux besoins actuels et futurs de l'analyse décisionnelle.

*Ce projet a été réalisé dans le cadre de la formation en Business Intelligence,
année universitaire 2025-2026.*