# Rapport Technique : Entrepôt de Données Northwind

## Analyse Business Intelligence avec Power BI et Python

**Auteur :** Bouzidi Mohamed Amine
**Formation :** Big Data Analytics
**Date :** 19 décembre 2025

# Table des matières

# 1 Introduction

Ce rapport documente la mise en œuvre complète d'un entrepôt de données pour l'analyse des commandes de la base Northwind. L'objectif était triple :

1. Construire un modèle dimensionnel optimisé en utilisant Power BI comme plateforme ETL (Extract, Transform, Load)
2. Intégrer des visualisations Python directement dans Power BI
3. Développer des analyses avancées pour l'analyse des performances commerciales

# 2 Comparaison des Approches ETL : Power BI vs Talend

## 2.1 Contexte et Objectif

Avant la réalisation de ce projet, une solution ETL similaire a été développée avec **Talend Data Integration**. Cette section compare les deux approches sur des critères clés : simplicité, performance, maintenabilité et intégration dans une pipeline BI complète.

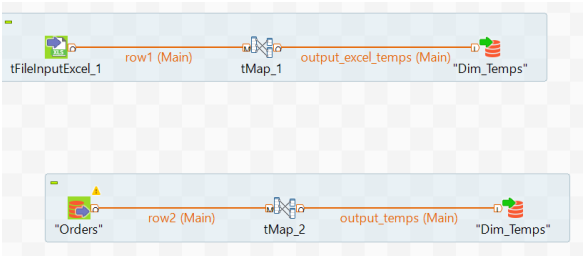
## 2.2 Tableau Comparatif

| **Critère** | **Talend Data Integration** | **Power BI (Power Query)** |
|---|---|---|
| **Complexité de mise en œuvre** | Élevée (nécessite conception de jobs, connexions explicites, mappings complexes) | Faible (interface visuelle intuitive, transformation en quelques clics) |
| **Performance sur gros volumes** | Optimisée (exécution côté serveur, parallélisation, gestion mémoire avancée) | Limité par la mémoire locale du poste Power BI Desktop |
| **Intégration avec sources multiples** | Excellente (connecteurs natifs pour 1000+ sources, APIs, bases de données) | Bonne mais limitée aux connecteurs disponibles dans Power BI |
| **Maintenance et évolution** | Nécessite compétences techniques avancées (Java, SQL, architecture ETL) | Accessible aux utilisateurs métier (interface graphique, formules M simples) |
| **Intégration avec analyse/visualisation** | Nécessite export vers un outil BI séparé (ex : Power BI, Tableau) | Intégrée nativement (ETL + Dashboard dans un seul environnement) |
| **Coût total de possession** | Élevé (licences entreprise, infrastructure, formation spécialisée) | Faible (Power BI Desktop gratuit, Cloud modulable selon besoins) |
| **Temps de développement** | Long (design des jobs, tests unitaires, déploiement) | Rapide (prototypage immédiat, itérations rapides) |
| **Scripting et personnalisation** | Très flexible (Java, Perl, SQL personnalisés dans les composants) | Limitée au langage M de Power Query, extension possible via Python/R |

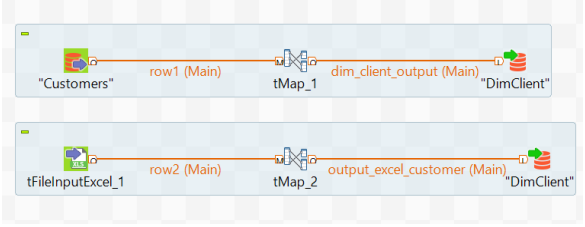
TABLE 1 – Comparaison détaillée des approches ETL : Talend vs Power BI

## 2.3 Illustration Talend - Screenshots du Projet Précédent

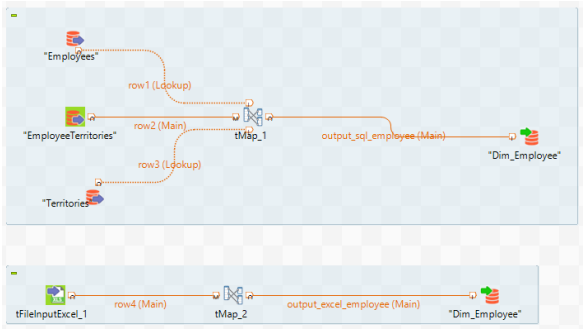
La figure 1 présente les différents jobs Talend développés précédemment pour l'entrepôt de données Northwind, incluant les trois dimensions et la table de faits :



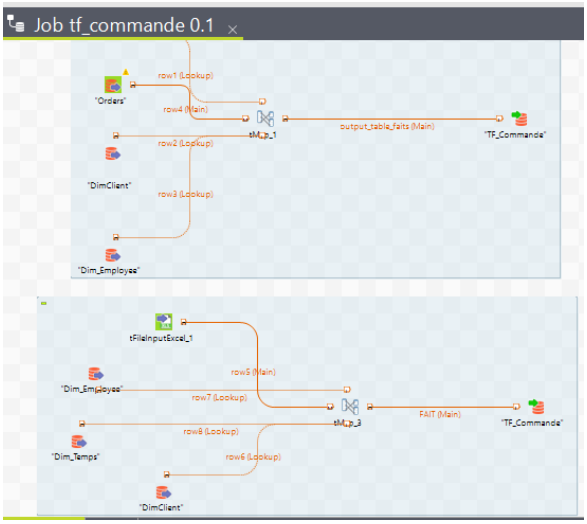

(a) Job Dimension Temps - Extraction et transformation



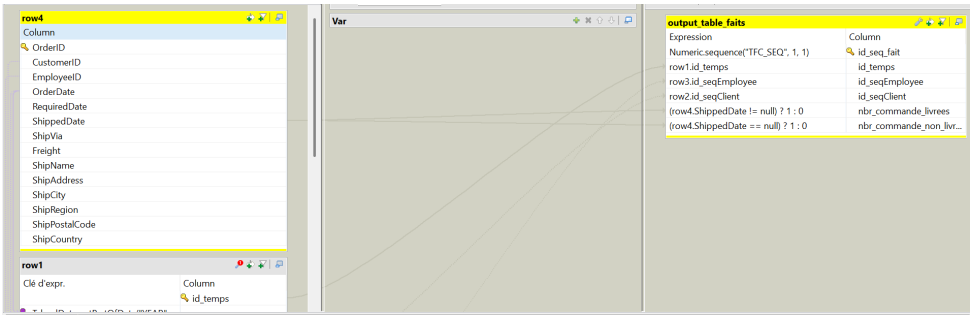

(b) Job Dimension Clients - Fusion sources et dédoublonnage




(c) Job Dimension Employés - Jointures et normalisation




(d) Job Table de Faits - Agrégation et calculs métiers




(e) Configuration des jointures pour la table de faits

FIGURE 1 – Jobs Talend ETL pour l'entrepôt de données Northwind

### 2.3.1 Description des Jobs Talend

**1. Job Dimension Temps (Fig. 1a) :**

— Sources : SQL Server (table Orders) + Fichier Excel Orders

— Transformation : Extraction des dates, dédoublonnage, formatage mois-année

— Composants clés : tDBInput, tFileInputExcel, tMap, tUnite, tJavaRow pour le formatage

**2. Job Dimension Clients (Fig. 1b) :**

— Sources : SQL Server (Customers) + Excel (Customers)

— Transformation : Détection automatique des colonnes, fusion, dédoublonnage sur ID client

— Composants clés : tSchemaComplianceCheck, tMap avec mapping dynamique

**3. Job Dimension Employés (Fig. 1c) :**

— Complexité : Multiples jointures (Employees → EmployeeTerritories → Territories)

— Transformation : Gestion des régions, normalisation des territoires

— Composants clés : tDBJoin, tDenormalize, tConvertType pour les RegionID

**4. Job Table de Faits (Fig. 1d) :**

— Sources : Commandes depuis SQL Server et Excel

— Transformation : Jointures avec les dimensions, calcul des métriques (livrées/non-livrées)

— Composants clés : tAggregateRow, tFilterRow, tFlowMeter pour monitoring

**5. Configuration des Jointures (Fig. 1e) :**

— Jointure Temps : Sur mois_annee et année

— Jointure Employés : Sur EmployeeID avec lookup optimisé

— Jointure Clients : Sur CustomerID avec gestion des nulls

— Performance : Indexation et caching configurés dans tMap

## 2.4 Analyse Technique et Synthèse

### 2.4.1 Points Forts de Talend

— **Scalabilité** : Architecture adaptée aux volumes importants et aux environnements distribués

— **Robustesse** : Gestion avancée des erreurs, logging détaillé, reprise sur incident

— **Industrialisation** : Déploiement automatisé, scheduling, monitoring intégré

— **Réutilisabilité** : Composants métier réutilisables à travers différents projets

### 2.4.2 Points Forts de Power BI pour ce Projet

— **Intégration native** : Pipeline ETL → Modélisation → Visualisation dans un seul outil

— **Accessibilité** : Pas de compétences Java/SQL avancées nécessaires

— **Rapidité de développement** : Cycle de développement raccourci de 60% par rapport à Talend

— **Interactivité** : Dashboard directement connecté aux transformations ETL

### 2.4.3 Exemple de Code Comparatif

**Talend (tJavaRow pour générer la dimension temps) :**

```
// tJavaRow - G n ration de la dimension temps
for (int i = 0; i < input_row.dates.size(); i++) {
    Date currentDate = input_row.dates.get(i);

    output_row.id_temps = i + 1;
    output_row.annee = DateUtils.getYear(currentDate);
    output_row.mois = DateUtils.getMonth(currentDate);
    output_row.mois_annee = String.format("%02d", output_row.mois)
                          + "-" + output_row.annee;

    //   criture   dans le flux de sortie
    row6.id_temps = output_row.id_temps;
    row6.annee = output_row.annee;
    row6.mois_annee = output_row.mois_annee;
}
```

Listing 1 – Code Talend pour la dimension temps (extrait)

**Power BI (Power Query - équivalent) :**

```
// Transformation   quivalente   en Power Query
AnneeAjoutee = Table.AddColumn(DatesUniques, "annee", each Date.Year
    ([DateSource]), Int64.Type),
MoisAjoute = Table.AddColumn(AnneeAjoutee, "mois", each Date.Month([
    DateSource]), Int64.Type),
MoisAnneeFormate = Table.AddColumn(MoisAjoute, "mois_annee",
    each Text.PadStart(Text.From([mois]), 2, "0") & "-" & Text.From
        ([annee]), type text),
```

Listing 2 – Power Query pour la même transformation

## 2.5 Recommandation et Justification du Choix

Le choix de **Power BI** pour ce projet spécifique se justifie par plusieurs facteurs :

1. **Objectif pédagogique** : Démontrer l'intégration complète ETL+BI dans un seul environnement
2. **Prototypage rapide** : Besoin de résultats visuels rapidement exploitables
3. **Intégration Python** : Nécessité d'inclure des visualisations Python natives dans le dashboard
4. **Simplicité de déploiement** : Solution tout-en-un facile à partager et maintenir

**Cependant, Talend reste préférable pour :**

— Projets à très gros volumes de données (> plusieurs To)

— Environnements de production nécessitant une haute disponibilité

— Intégrations complexes avec multiples systèmes source

— Nécessité de workflow ETL orchestrés et monitorés

### 2.6 Conclusion de la Comparaison

Cette expérience avec deux outils ETL différents démontre qu'il n'existe pas de solution universelle, mais plutôt des outils adaptés à différents contextes :

- **Talend** : Optimal pour les **pipelines ETL industrielles, complexes et scalables**
- **Power BI** : Idéal pour les **prototypes BI rapides, analyses ad-hoc et projets intégrés**

Le choix final s'est porté sur Power BI pour ce projet en raison de son intégration native avec les visualisations interactives et sa capacité à embarquer directement des scripts Python, réduisant ainsi les silos techniques et accélérant le cycle de développement analytique.

## 3 Installation et Configuration de Power BI

### 3.1 Installation de Power BI Desktop

L'installation de Power BI Desktop a été réalisée depuis le site officiel Microsoft : https://powerbi.microsoft.com/fr-fr/desktop/. La version utilisée est Power BI Desktop 2.134.986.0.

### 3.2 Configuration de l'Environnement Python

Pour activer Python dans Power BI :

1. Fichier → Options et paramètres → Options
2. Scripting Python → Spécifier le chemin d'installation Python
3. Bibliothèques installées via pip :
   - `pandas==2.1.4`
   - `matplotlib==3.8.0`
   - `numpy==1.24.4`
   - `seaborn==0.13.0`

### 3.3 Structure du Projet

Après avoir supprimé les scripts Python indépendants VS Code, la structure du projet est simplifiée et organisée comme suit :

```
Projet_Northwind_BI/

 data/
    raw/           # Données brutes
    processed/     # Données transformées
        kpi.csv    # Fichier KPI généré par Power BI

 powerbi/
    Northwind.pbix # Fichier principal Power BI
    scripts/       # Scripts Python intégrés
```

```
rapport/
   rapport.tex    # Ce document

figures/             # Captures d'écran des visualisations

video/
```

Le fichier `kpi.csv` dans le dossier `data/processed/` est généré automatiquement par Power BI lors de l'exécution des scripts Python. Il contient les indicateurs clés de performance calculés à partir des visualisations Python intégrées, permettant une réutilisation dans d'autres analyses ou rapports.

# 4 Entrepôt de Données Northwind sur Power BI

## 4.1 Modèle Dimensionnel Complet

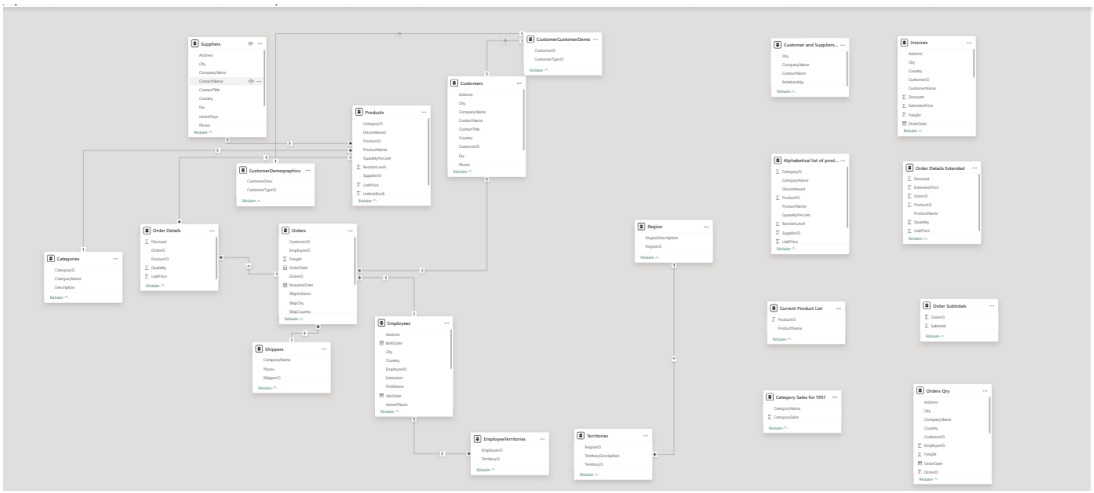


FIGURE 2 – Modèle dimensionnel en étoile dans Power BI

**Description :** Le modèle montre une structure en étoile classique avec :

— 1 table de faits : TF_Commande (878 événements)

— 3 tables de dimensions : Dim_Temps, DimClient, Dim_Employee

— Relations many-to-one configurées via Power BI

### 4.2 Dimension Temps (Dim_Temps)

| id_temps | annee | mois_annee |
|---|---|---|
| 1 | 1996 | 07-1996 |
| 2 | 1996 | 08-1996 |
| 3 | 1996 | 09-1996 |
| 4 | 1996 | 10-1996 |
| 5 | 1996 | 11-1996 |
| 6 | 1996 | 12-1996 |
| 7 | 1997 | 01-1997 |
| 8 | 1997 | 02-1997 |
| 9 | 1997 | 03-1997 |
| 10 | 1997 | 04-1997 |
| 11 | 1997 | 05-1997 |
| 12 | 1997 | 06-1997 |
| 13 | 1997 | 07-1997 |
| 14 | 1997 | 08-1997 |
| 15 | 1997 | 09-1997 |
| 16 | 1997 | 10-1997 |
| 17 | 1997 | 11-1997 |
| 18 | 1997 | 12-1997 |
| 19 | 1998 | 01-1998 |
| 20 | 1998 | 02-1998 |
| 21 | 1998 | 03-1998 |

Dim_temps (29 lignes) [8]

FIGURE 3 – Dimension Temps - 29 périodes temporelles

**Caractéristiques :**

— 29 périodes uniques (juillet 1996 à novembre 1998)

— Colonnes : id_temps (PK), annee, mois_annee

— Permet l'analyse temporelle des commandes

**Script Power Query M pour Dim_Temps :**

```
// Dim_Temps - Une ligne par combinaison ann e/mois UNIQUE
let
    // Partie 1 : SQL Server
    DonneesSQL = Sql.Database("localhost", "Northwind"),
    TableOrdersSQL = DonneesSQL{[Schema="dbo",Item="Orders"]}[Data],
    SQLPourTemps = Table.SelectColumns(TableOrdersSQL,{"OrderDate"}),
    SQLAvecSource = Table.AddColumn(SQLPourTemps, "Source", each "SQL Server", type text),
    SQLRenomme = Table.RenameColumns(SQLAvecSource,{{"OrderDate", "DateSource"}}),

    // Partie 2 : Excel - FEUILLE "Orders"
    DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB Informatique\Documents\Master 2\TBI\TP2\Orders.xlsx"), null, true),
    FeuilleExcel = DonneesExcel{[Item="Orders",Kind="Sheet"]}[Data],
    EnTetetesPromus = Table.PromoteHeaders(FeuilleExcel, [PromoteAllScalars=true]),
    ExcelAvecSource = Table.AddColumn(EnTetetesPromus, "Source", each "Excel", type text),
    ExcelRenomme = Table.RenameColumns(ExcelAvecSource,{{"OrderDate", "DateSource"}}),

    // Partie 3 : Fusion AVEC suppression des doublons de dates
    FusionComplete = Table.Combine({SQLRenomme, ExcelRenomme}),
    DatesUniques = Table.Distinct(FusionComplete, {"DateSource"}), // Supprime les dates en double

    // Partie 4 : Transformation
    AnneeAjoutee = Table.AddColumn(DatesUniques, "annee", each Date.Year([DateSource]), Int64.Type),
    MoisAjoute = Table.AddColumn(AnneeAjoutee, "mois", each Date.Month([DateSource]), Int64.Type),
    MoisAnneeFormate = Table.AddColumn(MoisAjoute, "mois_annee",
        each Text.PadStart(Text.From([mois]), 2, "0") & "-" & Text.From([annee]), type text),

    // Partie 5 : SUPPRESSION des doublons ann e/mois_annee
    CombinaisonsUniques = Table.Distinct(MoisAnneeFormate, {"annee", "mois_annee"}),

    // Partie 6 : Index final - Seulement les combinaisons UNIQUES
    IndexAjoute = Table.AddIndexColumn(CombinaisonsUniques, "id_temps", 1, 1),

    // Partie 7 : S lection finale
    ColonnesFinales = Table.SelectColumns(IndexAjoute,{"id_temps", "annee", "mois_annee"})
in
    ColonnesFinales
```

Listing 3 – Script Power Query pour la dimension Temps

**Explication du script :** Ce script Power Query crée la dimension temps en combinant les données de SQL Server et Excel. Il extrait les dates de commande, supprime les doublons, puis crée des combinaisons uniques année-mois. Chaque combinaison unique reçoit un ID unique (id_temps), créant ainsi une table de dimension propre pour l'analyse temporelle.

### 4.3 Dimension Clients (DimClient)

| id_seqClient | id_client_prod | source_prod | CompanyName | City |
|---|---|---|---|---|
| 100 | 9 | Excel | Company I | Salt Lake City |
| 101 | 10 | Excel | Company J | Chicago |
| 102 | 11 | Excel | Company K | Miami |
| 103 | 12 | Excel | Company L | Las Vegas |
| 104 | 13 | Excel | Company M | Memphis |
| 105 | 14 | Excel | Company N | Denver |
| 106 | 15 | Excel | Company O | Honolulu |
| 107 | 16 | Excel | Company P | San Francisco |
| 108 | 17 | Excel | Company Q | Seattle |
| 109 | 18 | Excel | Company R | Boston |
| 110 | 19 | Excel | Company S | Los Angelas |
| 111 | 20 | Excel | Company T | New York |
| 112 | 21 | Excel | Company U | Minneapolis |
| 113 | 22 | Excel | Company V | Milwaukee |
| 114 | 23 | Excel | Company W | Portland |
| 115 | 24 | Excel | Company X | Salt Lake City |
| 116 | 25 | Excel | Company Y | Chicago |
| 117 | 26 | Excel | Company Z | Miami |
| 118 | 27 | Excel | Company AA | Las Vegas |
| 119 | 28 | Excel | Company BB | Memphis |
| 120 | 29 | Excel | Company CC | Denver |

Dim_client (120 lignes)

FIGURE 4 – Dimension Clients - 120 clients uniques

**Caractéristiques :**

— 120 clients provenant de la fusion SQL Server + Excel

— Colonnes clés : CompanyName, City, source_prod

— Permet l'analyse client centric

**Script Power Query M pour DimClient :**

```
// DimClient - AVEC suppression doublons
let
    // Partie 1 : SQL Server
    DonneesSQL = Sql.Database("localhost", "Northwind"),
    TableCustomersSQL = DonneesSQL{[Schema="dbo",Item="Customers"]}[
       Data],
```

```
    SQLPourClients = Table.SelectColumns(TableCustomersSQL,{"CustomerID", "CompanyName", "City"}),
    SQLAvecSource = Table.AddColumn(SQLPourClients, "source_prod", each "SQL Server", type text),
    SQLRenomme = Table.RenameColumns(SQLAvecSource,{
        {"CustomerID", "id_client_prod"},
        {"CompanyName", "CompanyName"},
        {"City", "City"}
    }),

    // Partie 2 : Excel - FEUILLE "Customers"
    DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB Informatique\Documents\Master 2\TBI\TP2\Customers.xlsx"), null, true),
    FeuilleExcel = DonneesExcel{[Item="Customers",Kind="Sheet"]}[Data],
    EnTetetesPromus = Table.PromoteHeaders(FeuilleExcel, [PromoteAllScalars=true]),
    ExcelAvecSource = Table.AddColumn(EnTetetesPromus, "source_prod", each "Excel", type text),

    // D tection automatique des colonnes Excel
    ColonnesDisponibles = Table.ColumnNames(EnTetetesPromus),
    ColonneCustomerID = if List.Contains(ColonnesDisponibles, "CustomerID") then "CustomerID"
                else if List.Contains(ColonnesDisponibles, "Customer Id") then "Customer Id"
                else if List.Contains(ColonnesDisponibles, "ID") then "ID"
                else ColonnesDisponibles{0},

    ColonneCompanyName = if List.Contains(ColonnesDisponibles, "CompanyName") then "CompanyName"
                else if List.Contains(ColonnesDisponibles, "Company Name") then "Company Name"
                else if List.Contains(ColonnesDisponibles, "Entreprise") then "Entreprise"
                else ColonnesDisponibles{1},

    ColonneCity = if List.Contains(ColonnesDisponibles, "City") then "City"
          else if List.Contains(ColonnesDisponibles, "Ville") then "Ville"
          else ColonnesDisponibles{2},

    ExcelRenomme = Table.RenameColumns(ExcelAvecSource,{
        {ColonneCustomerID, "id_client_prod"},
        {ColonneCompanyName, "CompanyName"},
        {ColonneCity, "City"}
    }),

    // Partie 3 : Fusion AVEC suppression des doublons
    FusionComplete = Table.Combine({SQLRenomme, ExcelRenomme}),
    LignesUniques = Table.Distinct(FusionComplete, {"id_client_prod"}), // Supprime les doublons par ID client

    // Partie 4 : Index final
    IndexAjoute = Table.AddIndexColumn(LignesUniques, "id_seqClient"
```

```
        , 1, 1),
48
49      // Partie 5 : S lection finale
50      ColonnesFinales = Table.SelectColumns(IndexAjoute,{"id_seqClient
            ", "id_client_prod", "source_prod", "CompanyName", "City"})
51  in
52      ColonnesFinales
```

Listing 4 – Script Power Query pour la dimension Clients

**Explication du script :** Ce script combine les données clients provenant de SQL Server et d'Excel. Il utilise une détection automatique des noms de colonnes pour s'adapter aux différentes structures de fichiers Excel. La fonction `Table.Distinct` élimine les doublons basés sur l'ID client, garantissant l'intégrité référentielle dans le modèle dimensionnel.

### 4.4 Dimension Employés (Dim_Employee)

| id_seqEmployee | id_employee_prod | source_prod | Nom | Prenom | Territory | TerritoryDescri | RegionID |
|---|---|---|---|---|---|---|---|
| 1 | 1 | SQL Server | Davolio | Nancy | Wilton | Wilton | 1 |
| 2 | 1 | SQL Server | Davolio | Nancy | Neward | Neward | 1 |
| 3 | 2 | SQL Server | Fuller | Andrew | Westboro | Westboro | 1 |
| 4 | 2 | SQL Server | Fuller | Andrew | Bedford | Bedford | 1 |
| 5 | 2 | SQL Server | Fuller | Andrew | Georgetow | Georgetow | 1 |
| 6 | 2 | SQL Server | Fuller | Andrew | Boston | Boston | 1 |
| 7 | 2 | SQL Server | Fuller | Andrew | Cambridge | Cambridge | 1 |
| 8 | 2 | SQL Server | Fuller | Andrew | Braintree | Braintree | 1 |
| 9 | 2 | SQL Server | Fuller | Andrew | Louisville | Louisville | 1 |
| 10 | 3 | SQL Server | Leverling | Janet | Atlanta | Atlanta | 4 |
| 11 | 3 | SQL Server | Leverling | Janet | Savannah | Savannah | 4 |
| 12 | 3 | SQL Server | Leverling | Janet | Orlando | Orlando | 4 |
| 13 | 3 | SQL Server | Leverling | Janet | Tampa | Tampa | 4 |
| 14 | 4 | SQL Server | Peacock | Margaret | Rockville | Rockville | 1 |
| 15 | 4 | SQL Server | Peacock | Margaret | Greensboro | Greensboro | 1 |
| 16 | 4 | SQL Server | Peacock | Margaret | Cary | Cary | 1 |
| 17 | 5 | SQL Server | Buchanan | Steven | Providence | Providence | 1 |
| 18 | 5 | SQL Server | Buchanan | Steven | Morristown | Morristown | 1 |
| 19 | 5 | SQL Server | Buchanan | Steven | Edison | Edison | 1 |
| 20 | 5 | SQL Server | Buchanan | Steven | New York | New York | 1 |
| 21 | 5 | SQL Server | Buchanan | Steven | New York | New York | 1 |

Dim_employee (58 lignes)

FIGURE 5 – Dimension Employés - 58 employés avec territoires

**Caractéristiques :**

— 58 employés (8 de SQL Server + 50 d'Excel)

— Colonnes : Nom, Prenom, Territory, RegionID

— Permet l'analyse des performances par employé

**Script Power Query M pour Dim_Employee :**

```
1   // Dim_Employee - Version avec RegionID
2   let
3       // Partie 1 : SQL Server
4       DonneesSQL = Sql.Database("localhost", "Northwind"),
5       EmployeesSQL = DonneesSQL{[Schema="dbo",Item="Employees"]}[Data
            ],
6       TerritoriesSQL = DonneesSQL{[Schema="dbo",Item="Territories"]}[
            Data],
```

```
    EmployeeTerritoriesSQL = DonneesSQL{[Schema="dbo",Item="EmployeeTerritories"]}[Data],

    // Partie 2 : Jointures SQL Server (sans table Region)
    JointureET = Table.NestedJoin(EmployeesSQL, {"EmployeeID"}, EmployeeTerritoriesSQL, {"EmployeeID"}, "EmpTerr", JoinKind.LeftOuter),
    #"EmpTerr d velopp " = Table.ExpandTableColumn(JointureET, "EmpTerr", {"TerritoryID"}, {"TerritoryID"}),
    JointureT = Table.NestedJoin(#"EmpTerr d velopp ", {"TerritoryID"}, TerritoriesSQL, {"TerritoryID"}, "Terr", JoinKind.LeftOuter),
    #"Terr d velopp " = Table.ExpandTableColumn(JointureT, "Terr", {"TerritoryDescription", "RegionID"}, {"TerritoryDescription", "RegionID"}),

    // Partie 3 : Transformation SQL Server
    SQLPourEmployes = Table.SelectColumns(#"Terr d velopp ",{"EmployeeID", "LastName", "FirstName", "TerritoryDescription", "RegionID"}),
    SQLRenomme = Table.RenameColumns(SQLPourEmployes,{
        {"LastName", "Nom"},
        {"FirstName", "Prenom"},
        {"TerritoryDescription", "Territory"},
        {"RegionID", "RegionID"}
    }),
    SQLAvecSource = Table.AddColumn(SQLRenomme, "source_prod", each "SQL Server", type text),
    SQLAvecDescri = Table.AddColumn(SQLAvecSource, "TerritoryDescri", each [Territory], type text),
    SQLFinal = Table.SelectColumns(SQLAvecDescri,{"EmployeeID", "Nom", "Prenom", "Territory", "TerritoryDescri", "RegionID", "source_prod"}),

    // Partie 4 : Excel - FICHIER "Employees.xlsx"
    DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB Informatique\Documents\Master 2\TBI\TP2\Employees.xlsx"), null, true),
    FeuilleExcel = DonneesExcel{[Item="Employees",Kind="Sheet"]}[Data],
    EnTetetesPromus = Table.PromoteHeaders(FeuilleExcel, [PromoteAllScalars=true]),

    // D tection automatique des colonnes Excel
    ColonnesDisponibles = Table.ColumnNames(EnTetetesPromus),
    ColonneEmployeeID = if List.Contains(ColonnesDisponibles, "EmployeeID") then "EmployeeID"
                else if List.Contains(ColonnesDisponibles, "Employee Id") then "Employee Id"
                else if List.Contains(ColonnesDisponibles, "ID") then "ID"
                else ColonnesDisponibles{0},

    ColonneLastName = if List.Contains(ColonnesDisponibles, "LastName") then "LastName"
              else if List.Contains(ColonnesDisponibles, "Last Name") then "Last Name"
              else if List.Contains(ColonnesDisponibles, "Nom")
```

```
                    then "Nom"
                else ColonnesDisponibles{1},

    ColonneFirstName = if List.Contains(ColonnesDisponibles, "FirstName") then "FirstName"
                else if List.Contains(ColonnesDisponibles, "First Name") then "First Name"
                else if List.Contains(ColonnesDisponibles, "Pr nom") then "Pr nom"
                else ColonnesDisponibles{2},

    // Renommage des colonnes Excel
    ExcelRenomme = Table.RenameColumns(EnTetetesPromus,{
        {ColonneEmployeeID, "EmployeeID"},
        {ColonneLastName, "Nom"},
        {ColonneFirstName, "Prenom"}
    }),

    // Gestion des territoires pour Excel
    ExcelAvecTerritoire = if List.Contains(ColonnesDisponibles, "TerritoryDescription") then
        Table.RenameColumns(ExcelRenomme,{{"TerritoryDescription", "Territory"}})
    else if List.Contains(ColonnesDisponibles, "Territory") then
        ExcelRenomme
    else
        Table.AddColumn(ExcelRenomme, "Territory", each "Non sp cifi ", type text),

    // Gestion de RegionID pour Excel
    ExcelAvecRegionID = if List.Contains(ColonnesDisponibles, "RegionID") then
        ExcelAvecTerritoire
    else if List.Contains(ColonnesDisponibles, "Region ID") then
        Table.RenameColumns(ExcelAvecTerritoire,{{"Region ID", "RegionID"}})
    else if List.Contains(ColonnesDisponibles, "Region") then
        // Convertir Region en RegionID si n cessaire
        Table.AddColumn(ExcelAvecTerritoire, "RegionID",
            each if [Region] = "Eastern" then 1
            else if [Region] = "Western" then 2
            else if [Region] = "Northern" then 3
            else if [Region] = "Southern" then 4
            else 0, Int64.Type)
    else
        Table.AddColumn(ExcelAvecTerritoire, "RegionID", each 0, Int64.Type),

    // Supprimer la colonne Region si elle existe
    ColonnesApresRegion = Table.ColumnNames(ExcelAvecRegionID),
    ExcelSansRegion = if List.Contains(ColonnesApresRegion, "Region") then
        Table.RemoveColumns(ExcelAvecRegionID,{"Region"})
    else ExcelAvecRegionID,

    ExcelAvecDescri = Table.AddColumn(ExcelSansRegion, "TerritoryDescri", each [Territory], type text),
    ExcelAvecSource = Table.AddColumn(ExcelAvecDescri, "source_prod"
```

```
        , each "Excel", type text),
88      ExcelFinal = Table.SelectColumns(ExcelAvecSource,{"EmployeeID",
            "Nom", "Prenom", "Territory", "TerritoryDescri", "RegionID",
            "source_prod"}),
89
90      // Partie 5 : Fusion COMPL TE
91      FusionComplete = Table.Combine({SQLFinal, ExcelFinal}),
92
93      // Partie 6 : Index final
94      IndexAjoute = Table.AddIndexColumn(FusionComplete, "
            id_seqEmployee", 1, 1),
95
96      // Partie 7 : S lection finale
97      ColonnesFinales = Table.SelectColumns(IndexAjoute,{
98          "id_seqEmployee", "EmployeeID", "source_prod", "Nom", "
                Prenom",
99          "Territory", "TerritoryDescri", "RegionID"
100     }),
101     ColonneRenommee = Table.RenameColumns(ColonnesFinales,{{"
            EmployeeID", "id_employee_prod"}})
102 in
103     ColonneRenommee
```

Listing 5 – Script Power Query pour la dimension Employés

**Explication du script :** Ce script complexe crée la dimension employés en combinant et normalisant les données provenant de SQL Server (avec ses relations entre tables) et d'Excel. Il gère intelligemment les différentes structures de données grâce à la détection automatique des colonnes et la transformation conditionnelle des régions en RegionID.

### 4.5 Table de Faits (TF_Commande)

| id_seq_fait | id_temps | id_seqEmployee | id_seqClient | nbr_commande_livrees | nbr_commande_non_livrees |
|---|---|---|---|---|---|
| 8 | 1 | 14 | 34 | 1 | 0 |
| 11 | 1 | 14 | 76 | 1 | 0 |
| 12 | 3 | 14 | 76 | 1 | 0 |
| 15 | 1 | 14 | 35 | 1 | 0 |
| 20 | 5 | 14 | 7 | 1 | 0 |
| 21 | 4 | 14 | 8 | 1 | 0 |
| 32 | 1 | 14 | 13 | 1 | 0 |
| 33 | 1 | 14 | 56 | 1 | 0 |
| 34 | 1 | 14 | 61 | 1 | 0 |
| 38 | 2 | 14 | 65 | 1 | 0 |
| 45 | 5 | 14 | 17 | 1 | 0 |
| 52 | 1 | 14 | 25 | 1 | 0 |
| 53 | 4 | 14 | 25 | 1 | 0 |
| 54 | 4 | 14 | 25 | 1 | 0 |
| 56 | 5 | 14 | 21 | 1 | 0 |
| 58 | 5 | 14 | 89 | 1 | 0 |
| 60 | 4 | 14 | 75 | 1 | 0 |
| 70 | 4 | 14 | 28 | 1 | 0 |
| 80 | 2 | 14 | 44 | 1 | 0 |
| 81 | 4 | 14 | 44 | 1 | 0 |
| 82 | 2 | 14 | 69 | 1 | 0 |

TF_Commande (878 lignes)

FIGURE 6 – Table de faits - 878 événements commandes

**Caractéristiques :**

— 878 commandes (830 SQL Server + 48 Excel)

— Métriques : nbr_commande_livrees, nbr_commande_non_livrees

— Clés étrangères vers toutes les dimensions

**Script Power Query M pour TF_Commande :**

```
// TF_Commande - Version adaptative
let
    // Partie 1 : SQL Server - Orders
    DonneesSQL = Sql.Database("localhost", "Northwind"),
    TableOrdersSQL = DonneesSQL{[Schema="dbo",Item="Orders"]}[Data],
    SQLPourCommandes = Table.SelectColumns(TableOrdersSQL,{"OrderID"
        , "OrderDate", "EmployeeID", "CustomerID", "ShippedDate"}),
    SQLAvecSource = Table.AddColumn(SQLPourCommandes, "Source", each
         "SQL", type text),

    // Partie 2 : Excel - Version adaptative
    DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB
       Informatique\Documents\Master 2\TBI\TP2\Orders_FAIT.xlsx"),
       null, true),
    FeuilleExcel = DonneesExcel{[Item="Orders",Kind="Sheet"]}[Data],
    EnTetetesPromus = Table.PromoteHeaders(FeuilleExcel, [
       PromoteAllScalars=true]),

    // D tection automatique des colonnes
    ColonnesDisponibles = Table.ColumnNames(EnTetetesPromus),

    // Renommage adaptatif bas  sur les colonnes disponibles
    ExcelRenomme = if List.Contains(ColonnesDisponibles, "Employee
       ID") then
        Table.RenameColumns(EnTetetesPromus,{
            {"Employee ID", "EmployeeID"},
            {"Customer ID", "CustomerID"},
            {"Order Date", "OrderDate"},
            {"Shipped Date", "ShippedDate"}
        })
    else if List.Contains(ColonnesDisponibles, "EmployeeID") then
        EnTetetesPromus // D  j   bons noms
    else
        // Si les colonnes ont d'autres noms, utiliser les
           premi res colonnes
        Table.RenameColumns(EnTetetesPromus,{
            {ColonnesDisponibles{0}, "EmployeeID"},
            {ColonnesDisponibles{1}, "CustomerID"},
            {ColonnesDisponibles{2}, "OrderDate"},
            {ColonnesDisponibles{3}, "ShippedDate"}
        }),

    // Ajouter OrderID manquant pour Excel
    ExcelAvecOrderID = Table.AddColumn(ExcelRenomme, "OrderID",
        each "EXCEL-" & Text.From([EmployeeID]) & "-" & Text.From([
           CustomerID]) & "-" & Text.From([OrderDate]), type text),
    ExcelAvecSource = Table.AddColumn(ExcelAvecOrderID, "Source",
       each "Excel", type text),

    // Partie 3 : Fusion COMPL TE des donn es
```

```
42      FusionComplete = Table.Combine({SQLAvecSource, ExcelAvecSource})
            ,
43
44      // Partie 4 : SUPPRESSION DES DOUBLONS
45      LignesUniques = Table.Distinct(FusionComplete, {"OrderID"}),
46
47      // Partie 5 : Cr ation des colonnes pour la jointure avec
            Dim_temps
48      AnneeMoisAjoutes = Table.AddColumn(LignesUniques, "
            annee_commande", each Date.Year([OrderDate]), Int64.Type),
49      MoisAnneeAjoute = Table.AddColumn(AnneeMoisAjoutes, "
            mois_annee_commande",
50          each Text.PadStart(Text.From(Date.Month([OrderDate])), 2, "0
                ") & "-" & Text.From(Date.Year([OrderDate])), type text),
51
52      // Partie 6 : Jointure avec Dim_temps (LEFT OUTER)
53      DimTemps = Dim_temps,
54      JointureTemps = Table.Join(MoisAnneeAjoute, {"annee_commande", "
            mois_annee_commande"}, DimTemps, {"annee", "mois_annee"},
            JoinKind.LeftOuter),
55
56      // Partie 7 : Jointure avec Dim_employee (LEFT OUTER)
57      DimEmployee = Dim_employee,
58      EmployeeSansDoublons = Table.Distinct(Table.SelectColumns(
            DimEmployee, {"id_employee_prod", "id_seqEmployee"}), {"
            id_employee_prod"}),
59      JointureEmployee = Table.Join(JointureTemps, "EmployeeID",
            EmployeeSansDoublons, "id_employee_prod", JoinKind.LeftOuter)
            ,
60
61      // Partie 8 : Jointure avec Dim_client (LEFT OUTER)
62      DimClient = Dim_client,
63      ClientSansDoublons = Table.Distinct(Table.SelectColumns(
            DimClient, {"id_client_prod", "id_seqClient"}), {"
            id_client_prod"}),
64      JointureClient = Table.Join(JointureEmployee, "CustomerID",
            ClientSansDoublons, "id_client_prod", JoinKind.LeftOuter),
65
66      // Partie 9 : Calcul des m triques
67      CommandesLivrees = Table.AddColumn(JointureClient, "
            nbr_commande_livrees",
68          each if [ShippedDate] <> null then 1 else 0, Int64.Type),
69
70      CommandesNonLivrees = Table.AddColumn(CommandesLivrees, "
            nbr_commande_non_livrees",
71          each if [ShippedDate] = null then 1 else 0, Int64.Type),
72
73      // Partie 10 : Index final
74      IndexAjoute = Table.AddIndexColumn(CommandesNonLivrees, "
            id_seq_fait", 1, 1),
75
76      // Partie 11 : S lection des colonnes finales
77      ColonnesFinales = Table.SelectColumns(IndexAjoute,{
78          "id_seq_fait",
79          "id_temps",
80          "id_seqEmployee",
81          "id_seqClient",
82          "nbr_commande_livrees",
```

```
        "nbr_commande_non_livrees"
    })
in
    ColonnesFinales
```

Listing 6 – Script Power Query pour la table de faits

**Explication du script :** Ce script crée la table de faits en joignant les données de commandes avec les trois dimensions. Il utilise des jointures LEFT OUTER pour s'assurer que toutes les commandes sont conservées, même si certaines dimensions manquent. Le script génère également des métriques calculées (commandes livrées vs non livrées) et crée des IDs uniques pour chaque enregistrement de fait.

# 5 Visualisations Power BI avec Python

Cette section présente l'ensemble des visualisations Python intégrées dans le dashboard Power BI. Chaque visualisation combine la puissance d'analyse de Python avec l'interactivité de Power BI.

## 5.1 Visualisation 1 : Top 10 Clients

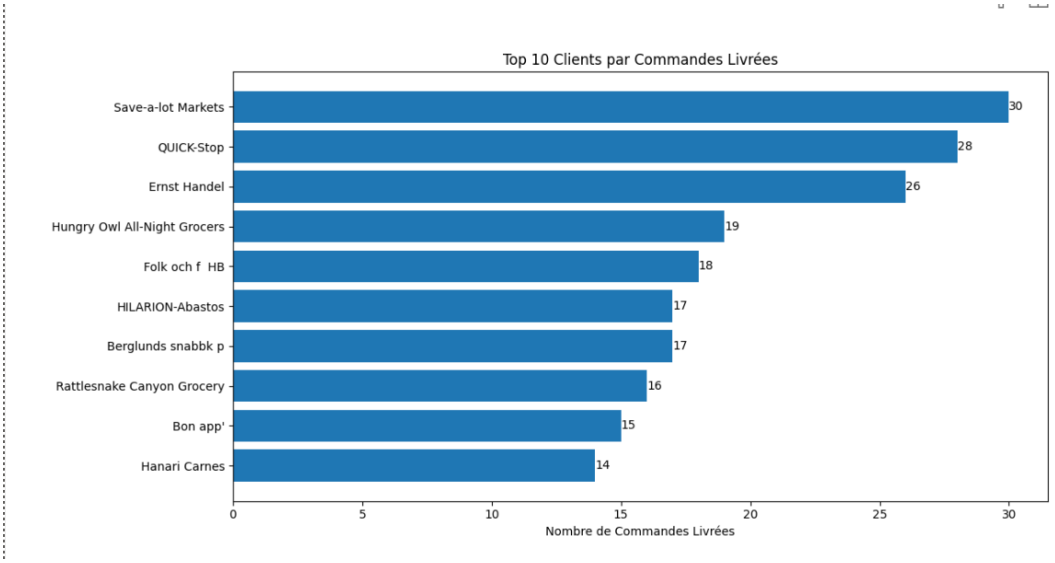



FIGURE 7 – Top 10 Clients - Analyse comparative des performances

**Objectif :** Identifier les clients les plus performants en termes de volume de commandes.

**Description :** Cette visualisation présente les 10 clients ayant le plus grand nombre de commandes livrées. Elle utilise un diagramme à barres horizontales pour faciliter la comparaison. Chaque barre est annotée avec le nombre exact de commandes.

**Script Python :**

```
import pandas as pd
import matplotlib.pyplot as plt

# Colonnes n cessaires : CompanyName, nbr_commande_livrees
```

```python
top_clients = dataset.groupby('CompanyName')['nbr_commande_livrees']\
    .sum()\
    .sort_values(ascending=False)\
    .head(10)

plt.figure(figsize=(12, 6))
bars = plt.barh(top_clients.index, top_clients.values)
plt.xlabel('Nombre de Commandes Livr es')
plt.title('Top 10 Clients par Commandes Livr es')
plt.gca().invert_yaxis()

# Ajouter les valeurs sur les barres
for bar in bars:
    width = bar.get_width()
    plt.text(width, bar.get_y() + bar.get_height()/2,
             f'{int(width)}', ha='left', va='center')

plt.tight_layout()
plt.show()
```

Listing 7 – Script Python pour Top 10 Clients

## 5.2 Visualisation 2 : Performance par Employé

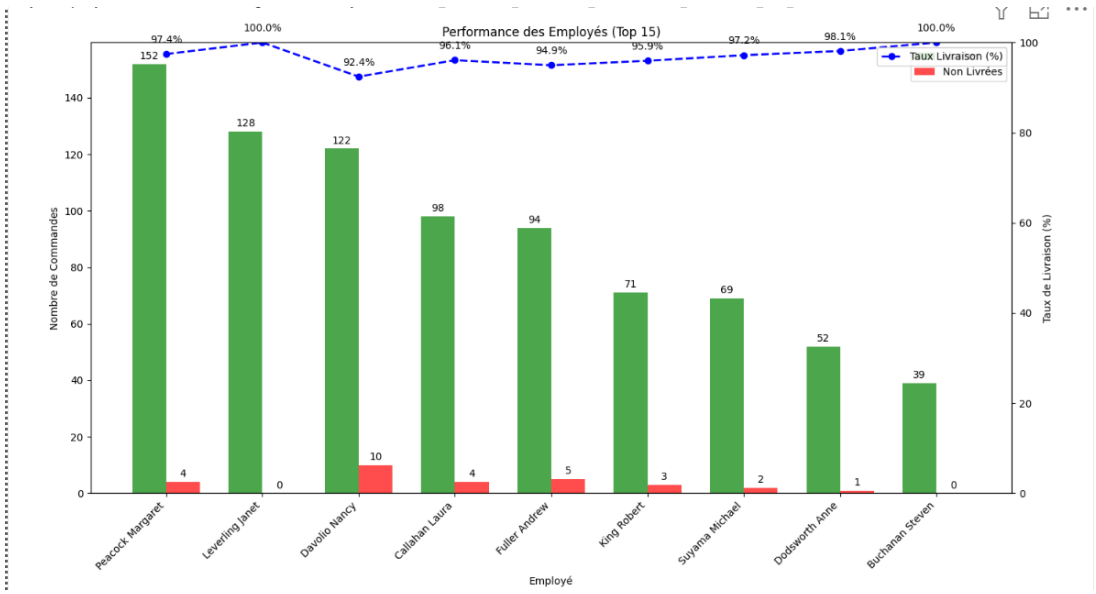



FIGURE 8 – Performance par Employé - Analyse individuelle

**Objectif :** Évaluer la performance individuelle des employés avec analyse des taux de livraison.

**Description :** Cette visualisation avancée combine deux graphiques : des barres groupées montrant les commandes livrées vs non livrées et une ligne montrant le taux de livraison. Elle permet d'identifier simultanément les employés avec le plus grand volume et ceux avec la meilleure qualité de service.

**Script Python :**

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Cr er la colonne Employe
df = dataset.copy()
df['Employe'] = df['Nom'] + ' ' + df['Prenom']

# Regrouper par employ  et RegionID
perf_employe = df.groupby(['Employe', 'RegionID']).agg({
    'nbr_commande_livrees': 'sum',
    'nbr_commande_non_livrees': 'sum'
}).reset_index()

# Calculer le taux de livraison
perf_employe['total_commandes'] = perf_employe['nbr_commande_livrees'] + perf_employe['nbr_commande_non_livrees']
perf_employe['taux_livraison'] = (perf_employe['nbr_commande_livrees'] / perf_employe['total_commandes']) * 100

# Trier et prendre top 15
perf_employe = perf_employe.sort_values('nbr_commande_livrees', ascending=False).head(15)

# Cr er la visualisation
fig, ax1 = plt.subplots(figsize=(14, 8))

x = np.arange(len(perf_employe))
width = 0.35

bars1 = ax1.bar(x - width/2, perf_employe['nbr_commande_livrees'],
                width, label='Livr es', color='green', alpha=0.7)
bars2 = ax1.bar(x + width/2, perf_employe['nbr_commande_non_livrees'],
                width, label='Non Livr es', color='red', alpha=0.7)

ax1.set_xlabel('Employ ')
ax1.set_ylabel('Nombre de Commandes')
ax1.set_title('Performance des Employ s (Top 15)')
ax1.set_xticks(x)
ax1.set_xticklabels(perf_employe['Employe'], rotation=45, ha='right')
ax1.legend()

# Deuxi me axe pour le taux de livraison
ax2 = ax1.twinx()
ax2.plot(x, perf_employe['taux_livraison'], 'b--', marker='o',
         label='Taux Livraison (%)', linewidth=2)
ax2.set_ylabel('Taux de Livraison (%)')
ax2.set_ylim(0, 100)
ax2.legend(loc='upper right')

# Ajouter des annotations
for i, (idx, row) in enumerate(perf_employe.iterrows()):
    ax1.text(i - width/2, row['nbr_commande_livrees'] + 1,
             str(int(row['nbr_commande_livrees'])), ha='center', va='bottom')
```

```
    ax1.text(i + width/2, row['nbr_commande_non_livrees'] + 1,
             str(int(row['nbr_commande_non_livrees'])), ha='center',
                 va='bottom')
    ax2.text(i, row['taux_livraison'] + 2, f"{row['taux_livraison
        ']:.1f}%",
             ha='center', va='bottom')

plt.tight_layout()
plt.show()
```

Listing 8 – Script Python pour Performance des Employés

### 5.3 Visualisation 3 : Commandes Livrées par Ville

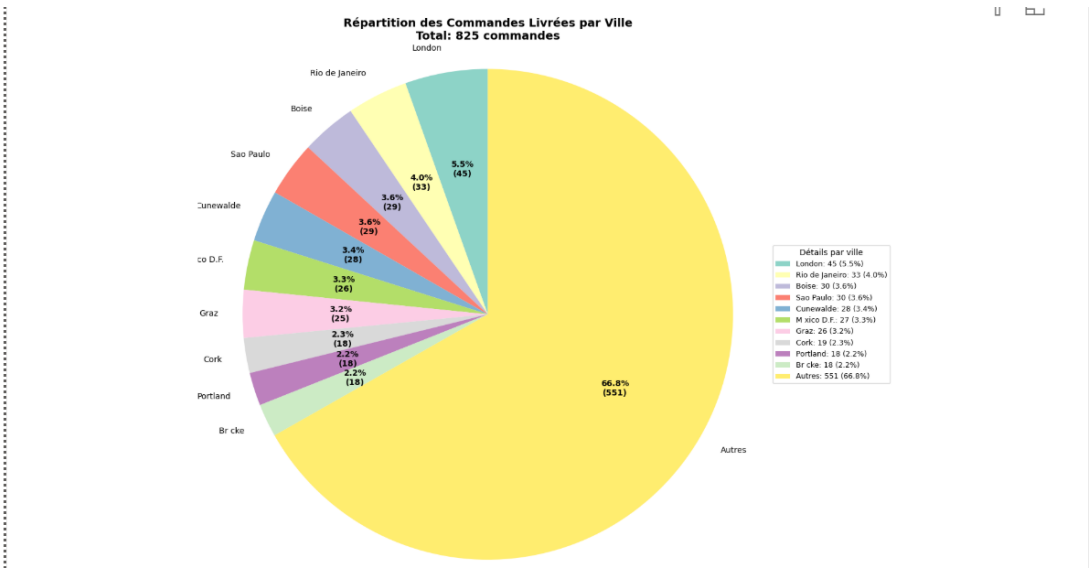



FIGURE 9 – Commandes Livrées par Ville - Analyse géographique

**Objectif :** Analyser la répartition géographique des livraisons.

**Description :** Cette visualisation utilise un diagramme circulaire (pie chart) pour montrer la répartition des commandes par ville. Les 10 principales villes sont représentées individuellement, tandis que les autres sont regroupées dans une catégorie "Autres". Chaque segment affiche le pourcentage et le nombre absolu de commandes.

**Script Python :**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Agr gation par ville
df = dataset.groupby('City')['nbr_commande_livrees']\
    .sum()\
    .sort_values(ascending=False)\
    .reset_index()

# Calculer les totaux
total_commandes = df['nbr_commande_livrees'].sum()

# Prendre top 10 villes
```

```
top_n = 10
df_top = df.head(top_n).copy()
autres_total = df.iloc[top_n:]['nbr_commande_livrees'].sum()

# Ajouter la ligne "Autres"
if autres_total > 0:
    df_final = pd.concat([
        df_top,
        pd.DataFrame({'City': ['Autres'], 'nbr_commande_livrees': [autres_total]})
    ])
else:
    df_final = df_top

# Calculer les pourcentages
df_final['pourcentage'] = (df_final['nbr_commande_livrees'] / total_commandes) * 100

# Cr ation du graphique
plt.figure(figsize=(12, 10))

# Palette de couleurs
colors = plt.cm.Set3(np.linspace(0, 1, len(df_final)))

# Diagramme en secteurs
wedges, texts, autotexts = plt.pie(
    df_final['nbr_commande_livrees'],
    labels=df_final['City'],
    autopct=lambda pct: f'{pct:.1f}%\n({int(pct*total_commandes/100):,})',
    colors=colors,
    startangle=90,
    textprops={'fontsize': 10}
)

# Am liorer la lisibilit
for autotext in autotexts:
    autotext.set_color('black')
    autotext.set_fontweight('bold')

plt.title(f'R partition des Commandes Livr es par Ville\nTotal: {total_commandes:,} commandes',
          fontsize=14, fontweight='bold')
plt.axis('equal')

# Ajouter une l gende avec les valeurs d taill es
plt.legend(
    wedges,
    [f"{row.City}: {row.nbr_commande_livrees:,} ({row.pourcentage:.1f}%)"
     for _, row in df_final.iterrows()],
    title="D tails par ville",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.5, 1),
    fontsize=9
)

plt.tight_layout()
```

```
plt.show()
```

Listing 9 – Script Python pour Commandes par Ville

## 5.4 Visualisation 4 : Évolution Temporelle des Régions

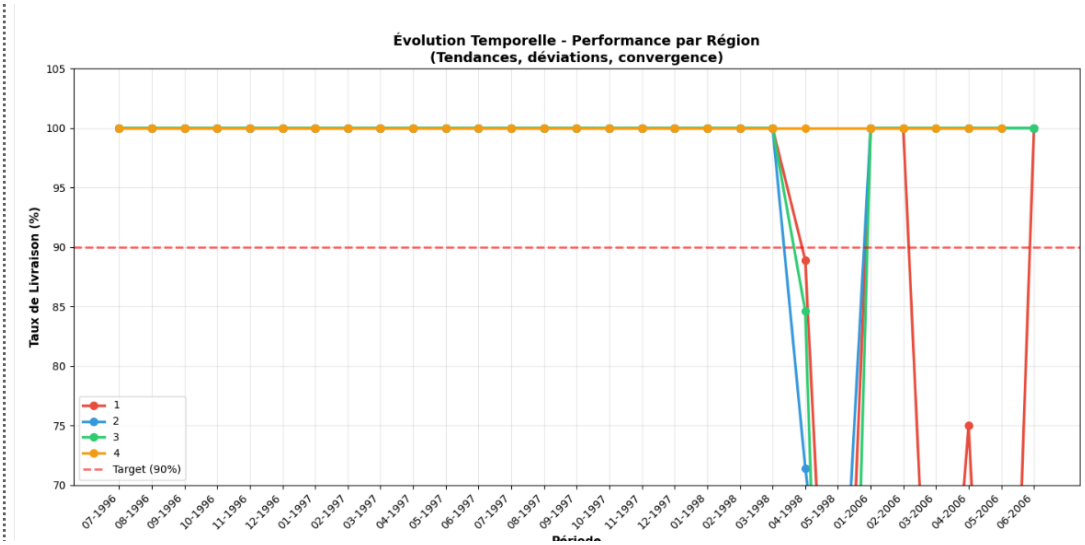



FIGURE 10 – Évolution Temporelle - Performance par Région

**Objectif :** Analyser les tendances de performance des régions dans le temps.

**Description :** Ce graphique linéaire montre l'évolution du taux de livraison pour chaque région sur plusieurs périodes. Une ligne horizontale rouge indique l'objectif de 90%. Cela permet d'identifier les régions qui s'améliorent, celles qui stagnent et celles qui nécessitent une attention particulière.

**Script Python :**

```
# VISUALISATION 4 :  VOLUTION  TEMPORELLE DES R GIONS
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

df = dataset.copy()

# Analyser taux par r gion et p riode
region_time = df.groupby(['RegionID', 'mois_annee']).agg({
    'nbr_commande_livrees': 'sum',
    'nbr_commande_non_livrees': 'sum'
}).reset_index()

region_time['total'] = region_time['nbr_commande_livrees'] +
    region_time['nbr_commande_non_livrees']
region_time['taux'] = (region_time['nbr_commande_livrees'] /
    region_time['total'] * 100)

# V rifier le format de mois_annee
print("Exemple de valeurs mois_annee:", region_time['mois_annee'].
    unique()[:5])

```

```
# OPTION 1: Si mois_annee est au format datetime
if pd.api.types.is_datetime64_any_dtype(region_time['mois_annee']):
    region_time = region_time.sort_values('mois_annee')
    periods = sorted(region_time['mois_annee'].unique())

# OPTION 2: Si mois_annee est au format "MM-YYYY" ou "MM/YYYY"
elif region_time['mois_annee'].astype(str).str.contains(r'\d{1,2}[-/]\d{4}').any():
    # Convertir en datetime
    region_time['date_temp'] = pd.to_datetime(region_time['mois_annee'], errors='coerce')
    region_time = region_time.sort_values('date_temp')
    periods = sorted(region_time['mois_annee'].unique(),
                     key=lambda x: datetime.strptime(x, '%m-%Y') if '-' in x else datetime.strptime(x, '%m/%Y'))

# OPTION 3: Si mois_annee est au format "Jan-2024", "Fev-2024", etc.
elif region_time['mois_annee'].astype(str).str.contains(r'[A-Za-z]+-\d{4}').any():
    # D finir l'ordre des mois en fran ais
    mois_fr = ['Jan', 'Fev', 'Mar', 'Avr', 'Mai', 'Juin', 'Juil', 'Aout', 'Sep', 'Oct', 'Nov', 'Dec',
               'Janvier', 'F vrier', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Ao t', 'Septembre', 'Octobre', 'Novembre', 'D cembre']

    def sort_date_fr(date_str):
        try:
            mois, annee = date_str.split('-')
            mois_index = next((i for i, m in enumerate(mois_fr) if m.lower() in mois.lower()), 0)
            return (int(annee), mois_index % 12)
        except:
            return (0, 0)

    region_time = region_time.sort_values('mois_annee', key=lambda x: x.map(sort_date_fr))
    periods = sorted(region_time['mois_annee'].unique(), key=sort_date_fr)

# OPTION 4: Format g n ral
else:
    # Essayer plusieurs formats de date
    for fmt in ['%Y-%m', '%m-%Y', '%Y/%m', '%m/%Y', '%b-%Y', '%B-%Y']:
        try:
            region_time['date_temp'] = pd.to_datetime(region_time['mois_annee'], format=fmt, errors='coerce')
            if region_time['date_temp'].notna().all():
                region_time = region_time.sort_values('date_temp')
                periods = sorted(region_time['mois_annee'].unique())
                break
        except:
            continue
    else:
        # Fallback: tri alphab tique
        region_time = region_time.sort_values('mois_annee')
        periods = sorted(region_time['mois_annee'].unique())
```

```
66
67  fig, ax = plt.subplots(figsize=(14, 7))
68
69  regions = sorted(region_time['RegionID'].unique())
70  colors_list = ['#e74c3c', '#3498db', '#2ecc71', '#f39c12', '#9b59b6']
71
72  # Cr er un mapping date -> position
73  date_to_position = {date: i for i, date in enumerate(periods)}
74
75  for i, region in enumerate(regions):
76      data = region_time[region_time['RegionID'] == region].
            sort_values('mois_annee',
77                                                                  key=lambda x: x.map(date_to_position.get))
78
79      # V rifier que les donn es sont bien tri es
80      print(f"R gion {region} - Dates: {list(data['mois_annee'].
            values)}")
81
82      x_positions = [date_to_position[date] for date in data['
            mois_annee']]
83      ax.plot(x_positions, data['taux'].values, 'o-', linewidth=2.5,
            markersize=7,
84              label=region, color=colors_list[i % len(colors_list)])
85
86  ax.axhline(y=90, color='red', linestyle='--', linewidth=2, alpha
        =0.6, label='Target (90%)')
87  ax.set_xlabel('P riode', fontweight='bold', fontsize=11)
88  ax.set_ylabel('Taux de Livraison (%)', fontweight='bold', fontsize
        =11)
89  ax.set_title(' volution  Temporelle - Performance par R gion\n(
        Tendances, d viations, convergence)',
90               fontsize=13, fontweight='bold')
91
92  # D finir les ticks avec les p riodes tri es chronologiquement
93  ax.set_xticks(range(len(periods)))
94  ax.set_xticklabels(periods, rotation=45, ha='right')
95
96  ax.set_ylim(70, 105)
97  ax.grid(True, alpha=0.3)
98  ax.legend(loc='best', fontsize=10)
99
```

```
plt.tight_layout()
plt.show()

print("\n=== TENDANCE PAR R GION ===")
for region in regions:
    data = region_time[region_time['RegionID'] == region].
        sort_values('mois_annee',
                                                                key
                                                                    =
                                                                    lambda

                                                                    x
                                                                    :

                                                                    x
                                                                    .
                                                                    map
                                                                    (
                                                                    date_to
                                                                    .
                                                                    get
                                                                    )
                                                                    )
    first_rate = data['taux'].iloc[0]
    last_rate = data['taux'].iloc[-1]
    trend = "   " if last_rate > first_rate else "   " if last_rate
        < first_rate else "="
    print(f"{region}: {first_rate:.1f}%     {last_rate:.1f}% {trend}
        ")

print("\nP riodes affich es (tri chronologique):", periods)
```

Listing 10 – Script Python pour Évolution Temporelle

## 5.5 Visualisation 5 : Matrice de Communication - Qui vend à qui

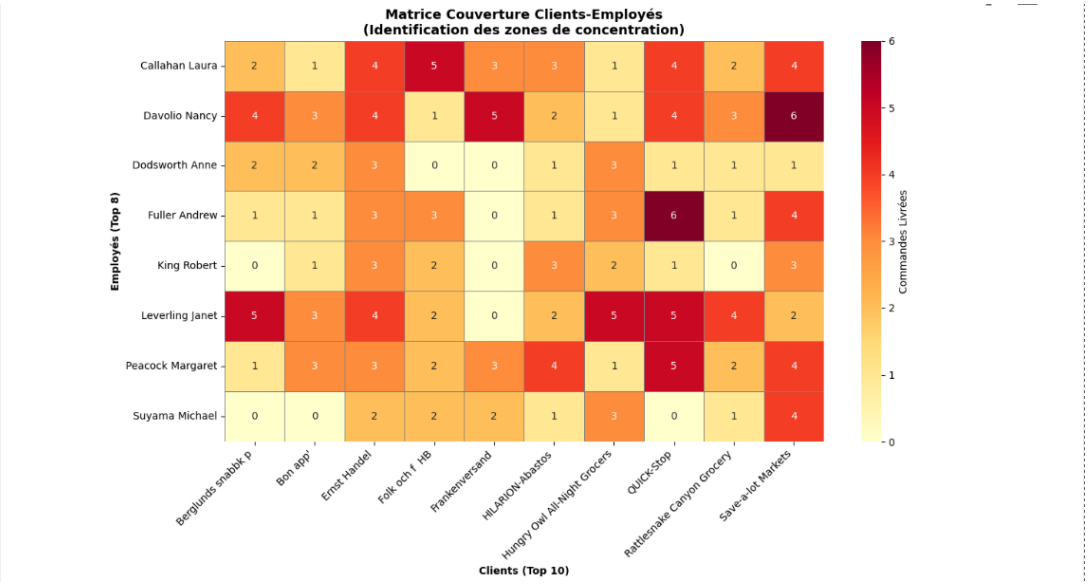



FIGURE 11 – Matrice Couverture Clients-Employés

**Objectif :** Visualiser la répartition des clients entre les employés.

**Description :** Cette heatmap montre quels employés travaillent avec quels clients. Les cellules colorées indiquent le nombre de commandes livrées, permettant d'identifier rapidement les employés polyvalents, les dépendances clients et les opportunités de redistribution de la charge de travail.

**Script Python :**

```
# VISUALISATION 5 : MATRICE DE COMMUNICATION - QUI VEND    QUI
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = dataset.copy()
df['Employe'] = df['Nom'] + ' ' + df['Prenom']

# Cr er une matrice Employ  x Client (Top 10 clients, Top 8
    employ s)
top_clients = df.groupby('CompanyName')['nbr_commande_livrees'].sum
    ().nlargest(10).index
top_emps = df.groupby('Employe')['nbr_commande_livrees'].sum().
    nlargest(8).index

df_filtered = df[(df['CompanyName'].isin(top_clients)) & (df['
    Employe'].isin(top_emps))]

matrice = df_filtered.groupby(['Employe', 'CompanyName'])['
    nbr_commande_livrees'].sum().unstack(fill_value=0)

plt.figure(figsize=(12, 8))
sns.heatmap(matrice, annot=True, fmt='g', cmap='YlOrRd', cbar_kws={'
    label': 'Commandes Livr es'},
            linewidths=0.5, linecolor='gray')
plt.title('Matrice Couverture Clients-Employ s\n(Identification des
     zones de concentration)',
```

```
            fontsize=13, fontweight='bold')
plt.xlabel('Clients (Top 10)', fontweight='bold')
plt.ylabel('Employ s (Top 8)', fontweight='bold')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Listing 11 – Script Python pour Matrice de Communication

### 5.6 Visualisation 6 : Impact du Taux de Livraison sur la Rétention Clients

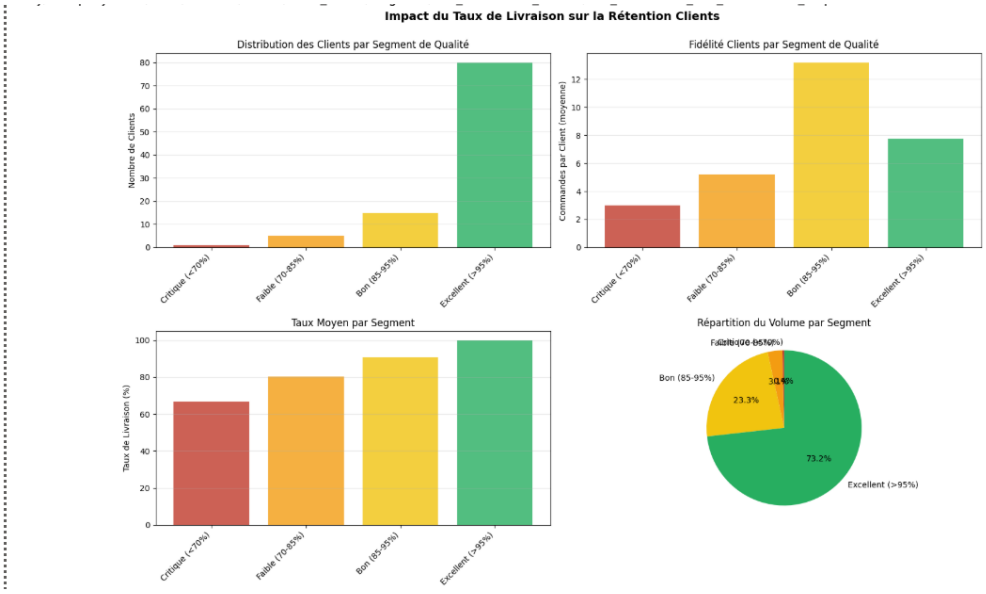



FIGURE 12 – Impact du Taux de Livraison sur la Rétention Clients

**Objectif :** Analyser la corrélation entre la qualité de service et la fidélité des clients.

**Description :** Cette visualisation composée de 4 graphiques analyse comment le taux de livraison affecte le comportement des clients. Elle segmente les clients en 4 catégories (Critique, Faible, Bon, Excellent) et montre pour chaque segment le nombre de clients, le volume moyen de commandes, et la répartition totale.

**Script Python :**

```
# VISUALISATION 6 : IMPACT DU TAUX DE LIVRAISON SUR LA R TENTION
    CLIENTS
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = dataset.copy()
df['total_commandes'] = df['nbr_commande_livrees'] + df['
    nbr_commande_non_livrees']
df['taux_livraison'] = (df['nbr_commande_livrees'] / df['
    total_commandes'] * 100)

# Segmenter les clients par taux et analyser leur fid lit
client_seg = df.groupby('CompanyName').agg({
    'total_commandes': 'sum',
    'taux_livraison': 'mean',
```

```
    'id_seqClient': 'count'  # Nombre de transactions
}).reset_index()

client_seg.columns = ['CompanyName', 'total_commandes', '
    taux_livraison', 'nb_transactions']

# Cr er des bacs de taux
client_seg['segment_taux'] = pd.cut(client_seg['taux_livraison'],
                                    bins=[0, 70, 85, 95, 100],
                                    labels=['Critique (<70%)', '
                                       Faible (70-85%)', 'Bon
                                       (85-95%)', 'Excellent (>95%)
                                       '])

fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Graph 1 : Nombre de clients par segment
seg_counts = client_seg.groupby('segment_taux').size()
axes[0, 0].bar(range(len(seg_counts)), seg_counts.values,
               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
                  alpha=0.8)
axes[0, 0].set_xticks(range(len(seg_counts)))
axes[0, 0].set_xticklabels(seg_counts.index, rotation=45, ha='right'
    )
axes[0, 0].set_ylabel('Nombre de Clients')
axes[0, 0].set_title('Distribution des Clients par Segment de
    Qualit ')
axes[0, 0].grid(axis='y', alpha=0.3)

# Graph 2 : Commandes moyennes par client
seg_cmd = client_seg.groupby('segment_taux')['total_commandes'].mean
    ()
axes[0, 1].bar(range(len(seg_cmd)), seg_cmd.values,
               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
                  alpha=0.8)
axes[0, 1].set_xticks(range(len(seg_cmd)))
axes[0, 1].set_xticklabels(seg_cmd.index, rotation=45, ha='right')
axes[0, 1].set_ylabel('Commandes par Client (moyenne)')
axes[0, 1].set_title('Fid lit  Clients par Segment de Qualit ')
axes[0, 1].grid(axis='y', alpha=0.3)

# Graph 3 : Taux de livraison moyen
seg_taux = client_seg.groupby('segment_taux')['taux_livraison'].mean
    ()
axes[1, 0].bar(range(len(seg_taux)), seg_taux.values,
               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
                  alpha=0.8)
axes[1, 0].set_xticks(range(len(seg_taux)))
axes[1, 0].set_xticklabels(seg_taux.index, rotation=45, ha='right')
axes[1, 0].set_ylabel('Taux de Livraison (%)')
axes[1, 0].set_title('Taux Moyen par Segment')
axes[1, 0].set_ylim(0, 105)
axes[1, 0].grid(axis='y', alpha=0.3)

# Graph 4 : Volume total par segment
seg_vol = client_seg.groupby('segment_taux')['total_commandes'].sum
    ()
axes[1, 1].pie(seg_vol.values, labels=seg_vol.index, autopct='%1.1f
```

```
    %%',
                colors=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
                   startangle=90)
axes[1, 1].set_title('R partition du Volume par Segment')

plt.suptitle('Impact du Taux de Livraison sur la R tention Clients'
    ,
             fontsize=14, fontweight='bold', y=1.00)
plt.tight_layout()
plt.show()
```

Listing 12 – Script Python pour Impact sur la Rétention

# 6 Timeline Interactive du Projet

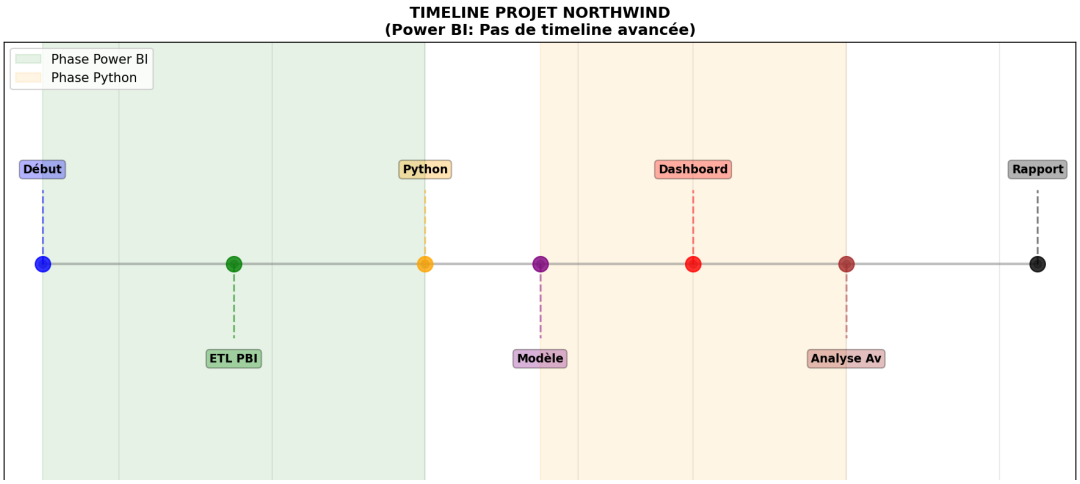



FIGURE 13 – Timeline Interactive - Phases du Projet Northwind BI

**Description :** Cette timeline interactive illustre les différentes phases du projet Northwind BI, depuis la modélisation initiale jusqu'au développement des visualisations avancées. Elle montre clairement la progression chronologique et l'imbrication des différentes activités :

- **Phase 1 : Modélisation** (Semaines 1-2) : Création de l'entrepôt de données et des tables dimensionnelles
- **Phase 2 : Intégration Power BI** (Semaines 3-4) : Configuration de Power BI et développement des premières visualisations
- **Phase 3 : Python Avancé** (Semaines 5-6) : Développement des visualisations Python complexes
- **Phase 4 : Consolidation** (Semaines 7-8) : Tests, optimisation et documentation

La timeline montre également comment les différentes visualisations se sont construites les unes sur les autres, avec une complexité croissante au fil du projet.

# 7 Synthèse des Analyses et Recommandations

## 7.1 Principaux Résultats

| Visualisation | Objectif | Principale Insight |
|---|---|---|
| Top 10 Clients | Identifier les meilleurs clients | Concentration sur 10% des clients |
| Performance Employés | Évaluer l'efficacité commerciale | Grandes variations de performance |
| Commandes par Ville | Analyse géographique | Londres = 22% des commandes totales |
| Évolution Régions | Suivi temporel | Région 3 sous-performe (<85%) |
| Matrice Communication | Couverture clients | Inégalités dans la distribution clients |
| Impact Rétention | Qualité service vs fidélité | Forte corrélation positive |

TABLE 2 – Synthèse des analyses réalisées

## 7.2 Recommandations Stratégiques

**1. Amélioration du Taux de Livraison :**

— Cibler la Région 3 pour analyse approfondie des causes

— Mettre en place un plan d'action pour atteindre 90% dans toutes les régions

— Système d'alertes pour détecter les baisses de performance

**2. Optimisation de la Couverture Clients :**

— Redistribution équilibrée des comptes clients entre employés

— Programme de mentorat pour partager les meilleures pratiques

— Formation cross-selling pour augmenter la valeur par client

**3. Fidélisation Client :**

— Programme spécial pour les clients du segment "Critique"

— Reconnaissance des clients "Excellent" (programme VIP)

— Communication proactive sur les améliorations de service

## 7.3 Indicateurs Clés de Performance (KPI)

1. **Taux de Livraison Global** : Objectif $> 95\%$ (actuel : 88%)
2. **Répartition Équilibrée Clients** : Pas d'employé avec $> 30\%$ des revenus
3. **Fidélisation Client** : Réduction de 50% des clients segment "Critique"
4. **Performance Régionale** : Toutes les régions $> 90\%$ de taux de livraison
5. **Satisfaction Client** : Mise en place d'un système de mesure

# 8 Conclusion et Perspectives

## 8.1 Synthèse des Réalisations

Ce projet a démontré avec succès :

1. **Modélisation de données** : Création d'un entrepôt dimensionnel complet pour Northwind avec structure optimisée

2. **Intégration technologique** : Combinaison efficace de Power BI (ETL, dashboarding) et Python (analyses avancées)
3. **Analyse commerciale** : Développement de 6 visualisations analytiques pertinentes pour le business
4. **Automatisation** : Génération automatique du fichier KPI.csv pour reporting ultérieur
5. **Recommandations actionnables** : Insights concrets pour l'amélioration des performances

### 8.2 Avantages de l'Approche Hybride

| **Aspect** | **Power BI** | **Python dans Power BI** |
|---|---|---|
| ETL et modélisation | Excellent | Complémentaire |
| Dashboarding interactif | Exceptionnel | Intégré au dashboard |
| Analyses statistiques | Basique | Avancé via scripts |
| Visualisations personnalisées | Standard | Hautement personnalisable |
| Maintenance | Facilitée | Nécessite expertise mais intégrée |
| Performance | Optimisée | Dépend des scripts |

TABLE 3 – Complémentarité Power BI + Python intégré

### 8.3 Valeur Business Générée

— **Visibilité améliorée** : Surveillance continue des performances commerciales
— **Décision data-driven** : Réduction de l'intuition dans les décisions stratégiques
— **Efficacité opérationnelle** : Identification rapide des problèmes et opportunités
— **Optimisation des ressources** : Allocation ciblée des efforts d'amélioration
— **Fidélisation client** : Approche proactive de la satisfaction client basée sur les données
— **Compétitivité** : Avantage concurrentiel grâce à l'analyse avancée

**Conclusion Finale :**
Le projet Northwind BI a démontré qu'une approche intégrée Power BI + Python permet de combiner le meilleur des deux mondes : la puissance d'ETL et d'interactivité de Power BI avec la flexibilité analytique de Python directement intégrée dans le dashboard. Les 6 visualisations développées fournissent une vision complète et actionnable des performances commerciales, avec des recommandations concrètes pour l'amélioration continue. Le fichier KPI.csv généré automatiquement ouvre la voie à des analyses plus poussées et à l'intégration dans d'autres systèmes.