

Rapport Technique : Entrepôt de Données Northwind

Analyse Business Intelligence avec Power BI et Python

Auteur : Bouzidi Mohamed Amine

Formation : Big Data Analytics

Date : 16 décembre 2025

Table des matières

1	Introduction	2
2	Comparaison des Approches ETL : Power BI vs Talend	2
2.1	Contexte et Objectif	2
2.2	Tableau Comparatif	2
2.3	Illustration Talend - Screenshots du Projet Précédent	3
2.3.1	Description des Jobs Talend	3
2.4	Analyse Technique et Synthèse	4
2.4.1	Points Forts de Talend	4
2.4.2	Points Forts de Power BI pour ce Projet	4
2.4.3	Exemple de Code Comparatif	5
2.5	Recommandation et Justification du Choix	5
2.6	Conclusion de la Comparaison	6
3	Installation et Configuration de Power BI	6
3.1	Installation de Power BI Desktop	6
3.2	Configuration de l'Environnement Python	6
3.3	Structure du Projet	6
4	Entrepôt de Données Northwind sur Power BI	7
4.1	Modèle Dimensionnel Complet	7
4.2	Dimension Temps (Dim_Temps)	8
4.3	Dimension Clients (DimClient)	10
4.4	Dimension Employés (Dim_Employee)	12
4.5	Table de Faits (TF_Commande)	15
5	Visualisations Power BI avec Python	18
5.1	Visualisation 1 : Top 10 Clients	18
5.2	Visualisation 2 : Performance par Employé	19
5.3	Visualisation 3 : Commandes Livrées par Ville	21
5.4	Visualisation 4 : Évolution Temporelle des Régions	23
5.5	Visualisation 5 : Matrice de Communication - Qui vend à qui	24
5.6	Visualisation 6 : Impact du Taux de Livraison sur la Rétention Clients	26
6	Timeline Interactive du Projet	28
7	Synthèse des Analyses et Recommandations	29
7.1	Principaux Résultats	29
7.2	Recommandations Stratégiques	29
7.3	Indicateurs Clés de Performance (KPI)	29
8	Conclusion et Perspectives	29
8.1	Synthèse des Réalisations	29
8.2	Avantages de l'Approche Hybride	30
8.3	Valeur Business Générée	30

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.

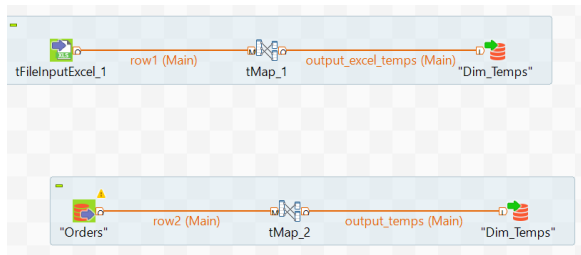
2.2 Tableau Comparatif

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

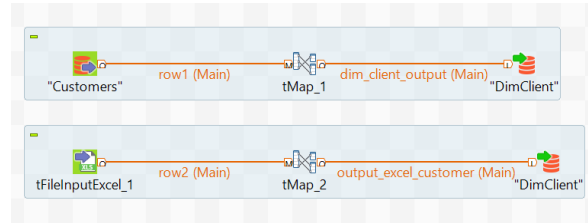
TABLE 1 – Comparaison détaillée des approches ETL : Talend vs Power BI

2.3 Illustration Talend - Screenshots du Projet Précédent

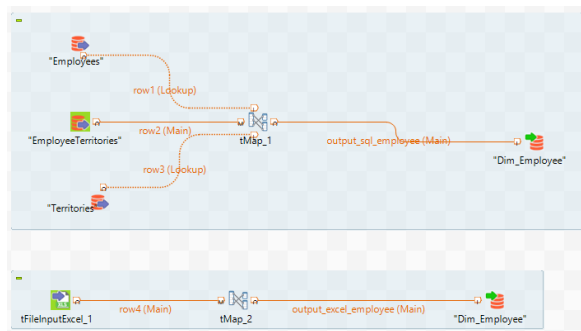
La figure 1 présente les différents jobs Talend développés précédemment pour l'entrepôt de données Northwind, incluant les trois dimensions et la table de faits :



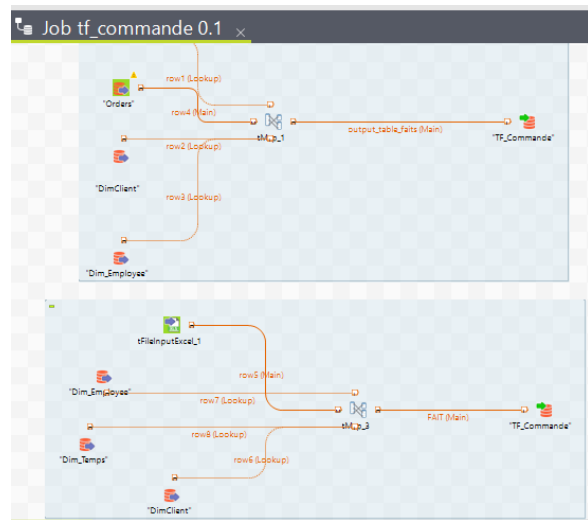
(a) Job Dimension Temps - Extraction et transformation



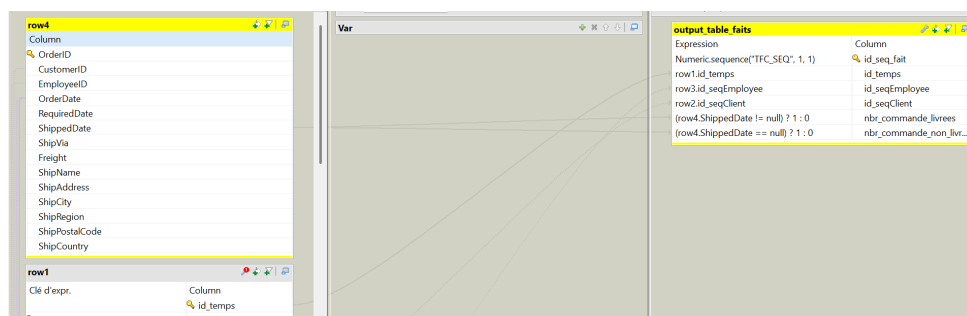
(b) Job Dimension Clients - Fusion sources et dédoublement



(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, dédoublement, 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) :

```
1 // tJavaRow - G n r a t i o n d e l a d i m e n s i o n t e m p s
2 for (int i = 0; i < input_row.dates.size(); i++) {
3     Date currentDate = input_row.dates.get(i);
4
5     output_row.id_temps = i + 1;
6     output_row.annee = DateUtils.getYear(currentDate);
7     output_row.mois = DateUtils.getMonth(currentDate);
8     output_row.mois_annee = String.format("%02d", output_row.mois)
9                             + "-" + output_row.annee;
10
11     // criture dans le flux de sortie
12     row6.id_temps = output_row.id_temps;
13     row6.annee = output_row.annee;
14     row6.mois_annee = output_row.mois_annee;
15 }
```

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

Power BI (Power Query - équivalent) :

```
1 // Transformation quivalente en Power Query
2 AnneeAjoutee = Table.AddColumn(DatesUniques, "annee", each Date.Year
3 ([DateSource]), Int64.Type),
4 MoisAjoute = Table.AddColumn(AnneeAjoutee, "mois", each Date.Month([
5 DateSource]), Int64.Type),
6 MoisAnneeFormate = Table.AddColumn(MoisAjoute, "mois_annee",
7     each Text.PadStart(Text.From([mois]), 2, "0") & "-" & Text.From
8 ([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

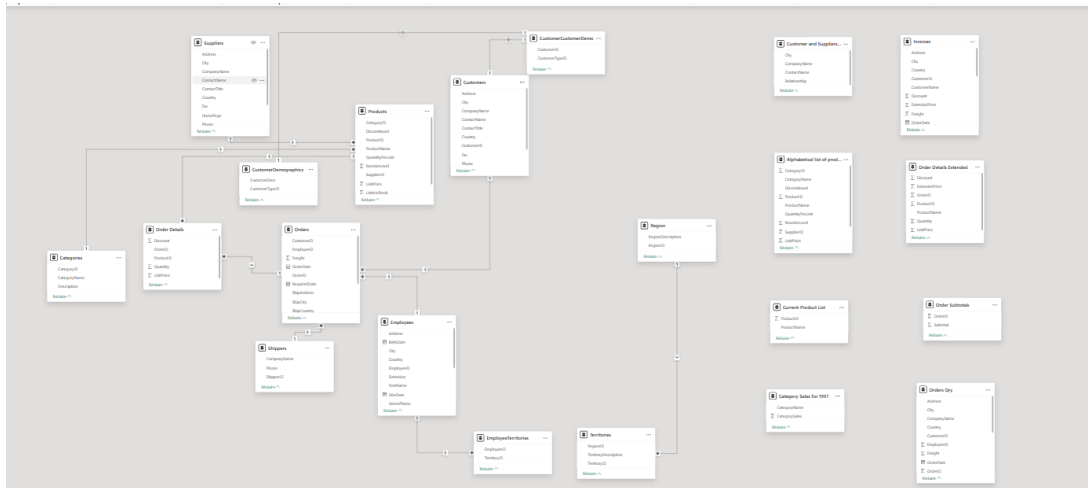


FIGURE 2 – Modèle dimensionnel en étoile dans Power BI

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 :

```
1 // Dim_Temps - Une ligne par combinaison ann e/mois UNIQUE
2 let
3     // Partie 1 : SQL Server
4     DonneesSQL = Sql.Database("localhost", "Northwind"),
5     TableOrdersSQL = DonneesSQL[Schema="dbo",Item="Orders"][Data],
6     SQLPourTemps = Table.SelectColumns(TableOrdersSQL,{"OrderDate"})
7
8     ,
9     SQLAvecSource = Table.AddColumn(SQLPourTemps, "Source", each "
10         SQL Server", type text),
11     SQLRenomme = Table.RenameColumns(SQLAvecSource,{"OrderDate", "
12         DateSource"})),
13
14     // Partie 2 : Excel - FEUILLE "Orders"
15     DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB
16         Informatique\Documents\Master 2\TBI\TP2\Orders.xlsx"), null,
17         true),
18     FeuilleExcel = DonneesExcel[Item="Orders",Kind="Sheet"][Data],
19     EnTetesPromus = Table.PromoteHeaders(FeuilleExcel, [
20         PromoteAllScalars=true]),
21     ExcelAvecSource = Table.AddColumn(EnTetesPromus, "Source",
22         each "Excel", type text),
23     ExcelRenomme = Table.RenameColumns(ExcelAvecSource,{"OrderDate"
24         , "DateSource"})),
25
26     // Partie 3 : Fusion AVEC suppression des doublons de dates
27     FusionComplete = Table.Combine({SQLRenomme, ExcelRenomme}),
28     DatesUniques = Table.Distinct(FusionComplete, {"DateSource"}),
29     // Supprime les dates en double
30
31     // Partie 4 : Transformation
32     AnneeAjoutee = Table.AddColumn(DatesUniques, "annee", each Date.
33         Year([DateSource]), Int64.Type),
34     MoisAjoute = Table.AddColumn(AnneeAjoutee, "mois", each Date.
35         Month([DateSource]), Int64.Type),
36     MoisAnneeFormate = Table.AddColumn(MoisAjoute, "mois_annee",
37         each Text.PadStart(Text.From([mois]), 2, "0") & "-" & Text.
38         From([annee]), type text),
39
40     // Partie 5 : SUPPRESSION des doublons ann e/mois_annee
41     CombinaisonsUniques = Table.Distinct(MoisAnneeFormate, {"annee",
42         "mois_annee"}),
43
44     // Partie 6 : Index final - Seulement les combinaisons UNIQUES
45     IndexAjoute = Table.AddIndexColumn(CombinaisonsUniques, "
46         id_temps", 1, 1),
47
48     // Partie 7 : S lection finale
49     ColonnesFinales = Table.SelectColumns(IndexAjoute,{"id_temps", "
50         annee", "mois_annee"})
51 in
52     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 :

```

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

```

```

6      SQLPourClients = Table.SelectColumns(TableCustomersSQL,{
7          CustomerID", "CompanyName", "City"}),
8      SQLAvecSource = Table.AddColumn(SQLPourClients, "source_prod",
9          each "SQL Server", type text),
10     SQLRenomme = Table.RenameColumns(SQLAvecSource,{
11         {"CustomerID", "id_client_prod"},
12         {"CompanyName", "CompanyName"},
13         {"City", "City"}
14     }),
15
16     // Partie 2 : Excel - FEUILLE "Customers"
17     DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB
18         Informatique\Documents\Master 2\TBI\TP2\Customers.xlsx"),
19         null, true),
20     FeuilleExcel = DonneesExcel[[Item="Customers",Kind="Sheet"]][
21         Data],
22     EnTetesPromus = Table.PromoteHeaders(FeuilleExcel, [
23         PromoteAllScalars=true]),
24     ExcelAvecSource = Table.AddColumn(EnTetesPromus, "source_prod"
25         , each "Excel", type text),
26
27     // D tection automatique des colonnes Excel
28     ColonnesDisponibles = Table.ColumnNames(EnTetesPromus),
29     ColonneCustomerID = if List.Contains(ColonnesDisponibles, "
30         CustomerID") then "CustomerID"
31         else if List.Contains(ColonnesDisponibles, "
32         Customer Id") then "Customer Id"
33         else if List.Contains(ColonnesDisponibles, "ID")
34         then "ID"
35         else ColonnesDisponibles{0},
36
37     ColonneCompanyName = if List.Contains(ColonnesDisponibles, "
38         CompanyName") then "CompanyName"
39         else if List.Contains(ColonnesDisponibles, "
40         Company Name") then "Company Name"
41         else if List.Contains(ColonnesDisponibles, "
42         Entreprise") then "Entreprise"
43         else ColonnesDisponibles{1},
44
45     ColonneCity = if List.Contains(ColonnesDisponibles, "City") then
46         "City"
47         else if List.Contains(ColonnesDisponibles, "Ville")
48         then "Ville"
49         else ColonnesDisponibles{2},
50
51     ExcelRenomme = Table.RenameColumns(ExcelAvecSource,{
52         {ColonneCustomerID, "id_client_prod"},
53         {ColonneCompanyName, "CompanyName"},
54         {ColonneCity, "City"}
55     }),
56
57     // Partie 3 : Fusion AVEC suppression des doublons
58     FusionComplete = Table.Combine({SQLRenomme, ExcelRenomme}),
59     LignesUniques = Table.Distinct(FusionComplete, {"id_client_prod"
60         }), // Supprime les doublons par ID client
61
62     // Partie 4 : Index final
63     IndexAjoute = Table.AddIndexColumn(LignesUniques, "id_seqClient"

```

```

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

```

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     ],
7     TerritoriesSQL = DonneesSQL{[Schema="dbo",Item="Territories"]}[
8     Data],
9
10

```

```

7 EmployeeTerritoriesSQL = DonneesSQL{[Schema="dbo",Item="
  EmployeeTerritories"]}[Data],
8
9 // Partie 2 : Jointures SQL Server (sans table Region)
10 JointureET = Table.NestedJoin(EmployeesSQL, {"EmployeeID"},
  EmployeeTerritoriesSQL, {"EmployeeID"}, "EmpTerr", JoinKind.
  LeftOuter),
11 #"EmpTerr d velopp " = Table.ExpandTableColumn(JointureET, "
  EmpTerr", {"TerritoryID"}, {"TerritoryID"}),
12 JointureT = Table.NestedJoin(#"EmpTerr d velopp ", {"
  TerritoryID"}, TerritoriesSQL, {"TerritoryID"}, "Terr",
  JoinKind.LeftOuter),
13 #"Terr d velopp " = Table.ExpandTableColumn(JointureT, "Terr",
  {"TerritoryDescription", "RegionID"}, {"TerritoryDescription
  ", "RegionID"}),
14
15 // Partie 3 : Transformation SQL Server
16 SQLPourEmployes = Table.SelectColumns(#"Terr d velopp ",{"
  EmployeeID", "LastName", "FirstName", "TerritoryDescription",
  "RegionID"}),
17 SQLRenomme = Table.RenameColumns(SQLPourEmployes,{
18 {"LastName", "Nom"},
19 {"FirstName", "Prenom"},
20 {"TerritoryDescription", "Territory"},
21 {"RegionID", "RegionID"}
22 },
23 SQLAvecSource = Table.AddColumn(SQLRenomme, "source_prod", each
  "SQL Server", type text),
24 SQLAvecDescri = Table.AddColumn(SQLAvecSource, "TerritoryDescri"
  , each [Territory], type text),
25 SQLFinal = Table.SelectColumns(SQLAvecDescri,{"EmployeeID", "Nom
  ", "Prenom", "Territory", "TerritoryDescri", "RegionID", "
  source_prod"}),
26
27 // Partie 4 : Excel - FICHER "Employees.xlsx"
28 DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB
  Informatique\Documents\Master 2\TBI\TP2\Employees.xlsx"),
  null, true),
29 FeuilleExcel = DonneesExcel{[Item="Employees",Kind="Sheet"]}[
  Data],
30 EnTetesPromus = Table.PromoteHeaders(FeuilleExcel, [
  PromoteAllScalars=true]),
31
32 // Dtection automatique des colonnes Excel
33 ColonnesDisponibles = Table.ColumnNames(EnTetesPromus),
34 ColonneEmployeeID = if List.Contains(ColonnesDisponibles, "
  EmployeeID") then "EmployeeID"
35 else if List.Contains(ColonnesDisponibles, "
  Employee Id") then "Employee Id"
36 else if List.Contains(ColonnesDisponibles, "ID")
  then "ID"
37 else ColonnesDisponibles{0},
38
39 ColonneLastName = if List.Contains(ColonnesDisponibles, "
  LastName") then "LastName"
40 else if List.Contains(ColonnesDisponibles, "Last
  Name") then "Last Name"
41 else if List.Contains(ColonnesDisponibles, "Nom")

```

```

42         then "Nom"
43         else ColonnesDisponibles{1},
44
45 ColonneFirstName = if List.Contains(ColonnesDisponibles, "
46     FirstName") then "FirstName"
47     else if List.Contains(ColonnesDisponibles, "First
48     Name") then "First Name"
49     else if List.Contains(ColonnesDisponibles, "
50     Pr nom") then "Pr nom"
51     else ColonnesDisponibles{2},
52
53 // Renommage des colonnes Excel
54 ExcelRenomme = Table.RenameColumns(EnTetetesPromus,{
55     {ColonneEmployeeID, "EmployeeID"},
56     {ColonneLastName, "Nom"},
57     {ColonneFirstName, "Prenom"}
58 },
59
60 // Gestion des territoires pour Excel
61 ExcelAvecTerritoire = if List.Contains(ColonnesDisponibles, "
62     TerritoryDescription") then
63     Table.RenameColumns(ExcelRenomme,{{"TerritoryDescription", "
64     Territory"}})
65 else if List.Contains(ColonnesDisponibles, "Territory") then
66     ExcelRenomme
67 else
68     Table.AddColumn(ExcelRenomme, "Territory", each "Non
69     sp cifi ", type text),
70
71 // Gestion de RegionID pour Excel
72 ExcelAvecRegionID = if List.Contains(ColonnesDisponibles, "
73     RegionID") then
74     ExcelAvecTerritoire
75 else if List.Contains(ColonnesDisponibles, "Region ID") then
76     Table.RenameColumns(ExcelAvecTerritoire,{{"Region ID", "
77     RegionID"}})
78 else if List.Contains(ColonnesDisponibles, "Region") then
79     // Convertir Region en RegionID si n cessaire
80     Table.AddColumn(ExcelAvecTerritoire, "RegionID",
81         each if [Region] = "Eastern" then 1
82         else if [Region] = "Western" then 2
83         else if [Region] = "Northern" then 3
84         else if [Region] = "Southern" then 4
85         else 0, Int64.Type)
86 else
87     Table.AddColumn(ExcelAvecTerritoire, "RegionID", each 0,
88         Int64.Type),
89
90 // Supprimer la colonne Region si elle existe
91 ColonneApresRegion = Table.ColumnNames(ExcelAvecRegionID),
92 ExcelSansRegion = if List.Contains(ColonneApresRegion, "Region"
93 ) then
94     Table.RemoveColumns(ExcelAvecRegionID,{"Region"})
95 else ExcelAvecRegionID,
96
97 ExcelAvecDescri = Table.AddColumn(ExcelSansRegion, "
98     TerritoryDescri", each [Territory], type text),
99 ExcelAvecSource = Table.AddColumn(ExcelAvecDescri, "source_prod"

```

```

88      , each "Excel", type text),
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 :

```
1 // TF_Commande - Version adaptative
2 let
3     // Partie 1 : SQL Server - Orders
4     DonneesSQL = Sql.Database("localhost", "Northwind"),
5     TableOrdersSQL = DonneesSQL{[Schema="dbo",Item="Orders"]}[Data],
6     SQLPourCommandes = Table.SelectColumns(TableOrdersSQL,{"OrderID",
7         "OrderDate", "EmployeeID", "CustomerID", "ShippedDate"}),
8     SQLAvecSource = Table.AddColumn(SQLPourCommandes, "Source", each
9         "SQL", type text),
10
11     // Partie 2 : Excel - Version adaptative
12     DonneesExcel = Excel.Workbook(File.Contents("C:\Users\GAB
13         Informatique\Documents\Master 2\TBI\TP2\Orders_FAIT.xlsx"),
14         null, true),
15     FeuilleExcel = DonneesExcel{[Item="Orders",Kind="Sheet"]}[Data],
16     EnTetesPromus = Table.PromoteHeaders(FeuilleExcel, [
17         PromoteAllScalars=true]),
18
19     // Détection automatique des colonnes
20     ColonnesDisponibles = Table.ColumnNames(EnTetesPromus),
21
22     // Renommage adaptatif bas sur les colonnes disponibles
23     ExcelRenomme = if List.Contains(ColonnesDisponibles, "Employee
24         ID") then
25         Table.RenameColumns(EnTetesPromus,{
26             {"Employee ID", "EmployeeID"},
27             {"Customer ID", "CustomerID"},
28             {"Order Date", "OrderDate"},
29             {"Shipped Date", "ShippedDate"}}
30     )
31 else if List.Contains(ColonnesDisponibles, "EmployeeID") then
32     EnTetesPromus // Déjà bons noms
33 else
34     // Si les colonnes ont d'autres noms, utiliser les
35     premières colonnes
36     Table.RenameColumns(EnTetesPromus,{
37         {ColonnesDisponibles{0}, "EmployeeID"},
38         {ColonnesDisponibles{1}, "CustomerID"},
39         {ColonnesDisponibles{2}, "OrderDate"},
40         {ColonnesDisponibles{3}, "ShippedDate"}}
41     ),
42
43     // Ajouter OrderID manquant pour Excel
44     ExcelAvecOrderID = Table.AddColumn(ExcelRenomme, "OrderID",
45         each "EXCEL-" & Text.From([EmployeeID]) & "-" & Text.From([
46             CustomerID]) & "-" & Text.From([OrderDate]), type text),
47     ExcelAvecSource = Table.AddColumn(ExcelAvecOrderID, "Source",
48         each "Excel", type text),
49
50     // Partie 3 : Fusion COMPLÈTE des données
```

```

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
48 Dim_temps
49 AnneeMoisAjoutes = Table.AddColumn(LignesUniques, "
50     annee_commande", each Date.Year([OrderDate]), Int64.Type),
51 MoisAnneeAjoute = Table.AddColumn(AnneeMoisAjoutes, "
52     mois_annee_commande",
53     each Text.PadStart(Text.From(Date.Month([OrderDate])), 2, "0
54         ") & "-" & Text.From(Date.Year([OrderDate])), type text),
55
56 // Partie 6 : Jointure avec Dim_temps (LEFT OUTER)
57 DimTemps = Dim_temps,
58 JointureTemps = Table.Join(MoisAnneeAjoute, {"annee_commande", "
59     mois_annee_commande"}, DimTemps, {"annee", "mois_annee"},
60     JoinKind.LeftOuter),
61
62 // Partie 7 : Jointure avec Dim_employee (LEFT OUTER)
63 DimEmployee = Dim_employee,
64 EmployeeSansDoublons = Table.Distinct(Table.SelectColumns(
65     DimEmployee, {"id_employee_prod", "id_seqEmployee"}), {"
66     id_employee_prod"}),
67 JointureEmployee = Table.Join(JointureTemps, "EmployeeID",
68     EmployeeSansDoublons, "id_employee_prod", JoinKind.LeftOuter)
69
70 // Partie 8 : Jointure avec Dim_client (LEFT OUTER)
71 DimClient = Dim_client,
72 ClientSansDoublons = Table.Distinct(Table.SelectColumns(
73     DimClient, {"id_client_prod", "id_seqClient"}), {"
74     id_client_prod"}),
75 JointureClient = Table.Join(JointureEmployee, "CustomerID",
76     ClientSansDoublons, "id_client_prod", JoinKind.LeftOuter),
77
78 // Partie 9 : Calcul des m triques
79 CommandesLivrees = Table.AddColumn(JointureClient, "
80     nbr_commande_livrees",
81     each if [ShippedDate] <> null then 1 else 0, Int64.Type),
82
83 CommandesNonLivrees = Table.AddColumn(CommandesLivrees, "
84     nbr_commande_non_livrees",
85     each if [ShippedDate] = null then 1 else 0, Int64.Type),
86
87 // Partie 10 : Index final
88 IndexAjoute = Table.AddIndexColumn(CommandesNonLivrees, "
89     id_seq_fait", 1, 1),
90
91 // Partie 11 : S lection des colonnes finales
92 ColonnesFinales = Table.SelectColumns(IndexAjoute, {
93     "id_seq_fait",
94     "id_temps",
95     "id_seqEmployee",
96     "id_seqClient",
97     "nbr_commande_livrees",

```

```

83         "nbr_commande_non_livrees"
84     })
85     in
86     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

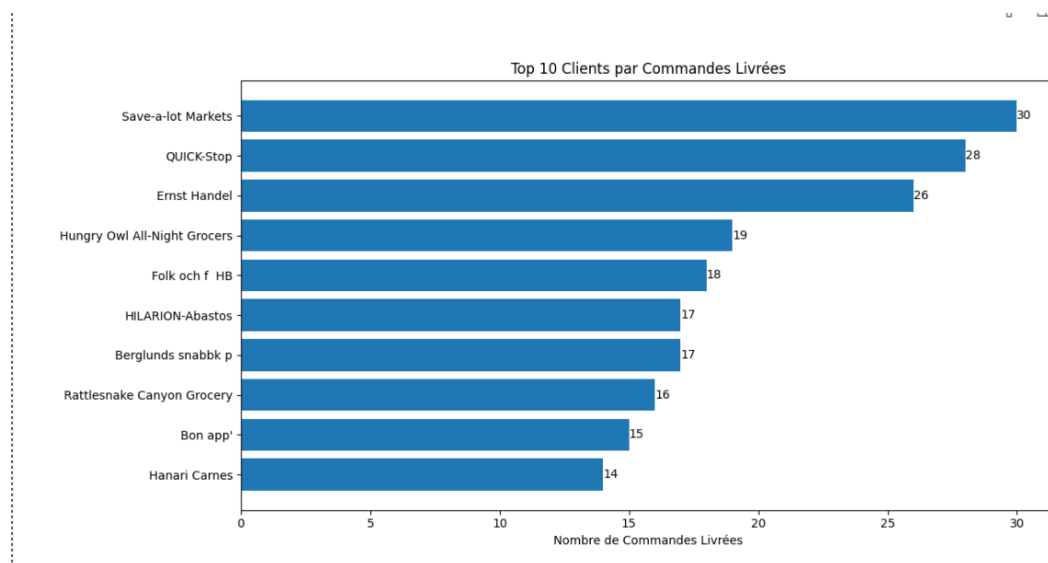


FIGURE 7 – Top 10 Clients - Analyse comparative des performances

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 :

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Colonnes nécessaires : CompanyName, nbr_commande_livrees

```

```

5 top_clients = dataset.groupby('CompanyName')['nbr_commande_livrees',
6   ]\
7   .sum()\
8   .sort_values(ascending=False)\
9   .head(10)
10
11 plt.figure(figsize=(12, 6))
12 bars = plt.barh(top_clients.index, top_clients.values)
13 plt.xlabel('Nombre de Commandes Livrées')
14 plt.title('Top 10 Clients par Commandes Livrées')
15 plt.gca().invert_yaxis()
16
17 # Ajouter les valeurs sur les barres
18 for bar in bars:
19     width = bar.get_width()
20     plt.text(width, bar.get_y() + bar.get_height()/2,
21             f'{int(width)}', ha='left', va='center')
22
23 plt.tight_layout()
24 plt.show()

```

Listing 7 – Script Python pour Top 10 Clients

5.2 Visualisation 2 : Performance par Employé

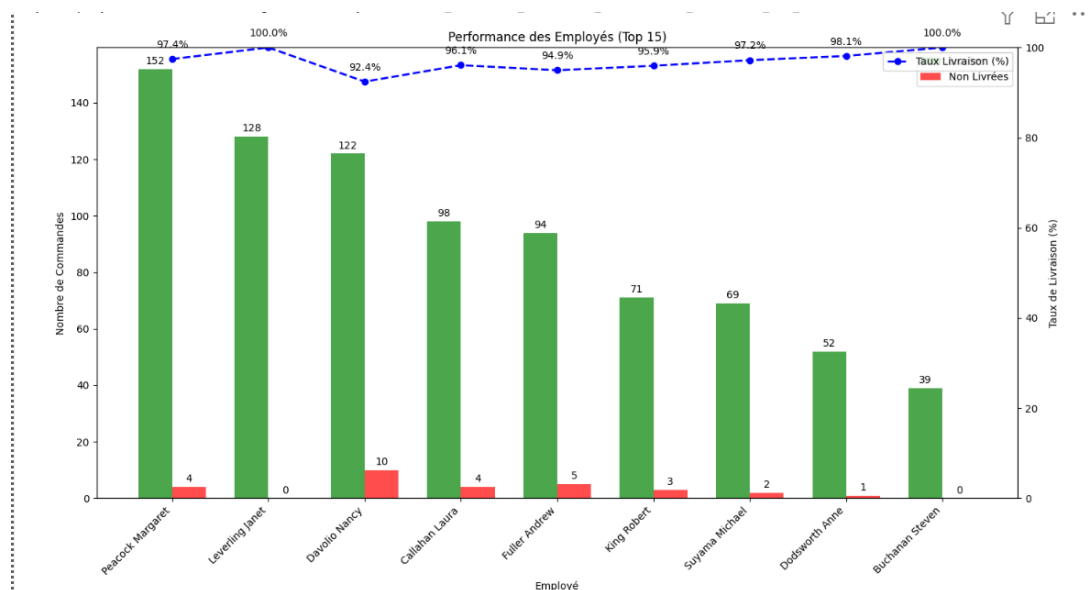


FIGURE 8 – Performance par Employé - Analyse individuelle

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 :

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 # Cr er la colonne Employe
6 df = dataset.copy()
7 df['Employe'] = df['Nom'] + ' ' + df['Prenom']
8
9 # Regrouper par employe et RegionID
10 perf_employe = df.groupby(['Employe', 'RegionID']).agg({
11     'nbr_commande_livrees': 'sum',
12     'nbr_commande_non_livrees': 'sum'
13 }).reset_index()
14
15 # Calculer le taux de livraison
16 perf_employe['total_commandes'] = perf_employe['nbr_commande_livrees'] + perf_employe['nbr_commande_non_livrees']
17 perf_employe['taux_livraison'] = (perf_employe['nbr_commande_livrees'] / perf_employe['total_commandes']) * 100
18
19 # Trier et prendre top 15
20 perf_employe = perf_employe.sort_values('nbr_commande_livrees',
21     ascending=False).head(15)
22
23 # Cr er la visualisation
24 fig, ax1 = plt.subplots(figsize=(14, 8))
25
26 x = np.arange(len(perf_employe))
27 width = 0.35
28
29 bars1 = ax1.bar(x - width/2, perf_employe['nbr_commande_livrees'],
30     width, label='Livr es', color='green', alpha=0.7)
31 bars2 = ax1.bar(x + width/2, perf_employe['nbr_commande_non_livrees'],
32     width, label='Non Livr es', color='red', alpha=0.7)
33
34 ax1.set_xlabel('Employ ')
35 ax1.set_ylabel('Nombre de Commandes')
36 ax1.set_title('Performance des Employ s (Top 15)')
37 ax1.set_xticks(x)
38 ax1.set_xticklabels(perf_employe['Employe'], rotation=45, ha='right',
39     )
40 ax1.legend()
41
42 # Deuxi me axe pour le taux de livraison
43 ax2 = ax1.twinx()
44 ax2.plot(x, perf_employe['taux_livraison'], 'b--', marker='o',
45     label='Taux Livraison (%)', linewidth=2)
46 ax2.set_ylabel('Taux de Livraison (%)')
47 ax2.set_ylim(0, 100)
48 ax2.legend(loc='upper right')
49
50 # Ajouter des annotations
51 for i, (idx, row) in enumerate(perf_employe.iterrows()):
52     ax1.text(i - width/2, row['nbr_commande_livrees'] + 1,
53         str(int(row['nbr_commande_livrees'])), ha='center', va=
54             'bottom')

```

```

52     ax1.text(i + width/2, row['nbr_commande_non_livrees'] + 1,
53             str(int(row['nbr_commande_non_livrees'])), ha='center',
54             va='bottom')
55     ax2.text(i, row['taux_livraison'] + 2, f"{row['taux_livraison']:.1f}%",
56             ha='center', va='bottom')
57 plt.tight_layout()
58 plt.show()

```

Listing 8 – Script Python pour Performance des Employés

5.3 Visualisation 3 : Commandes Livrées par Ville

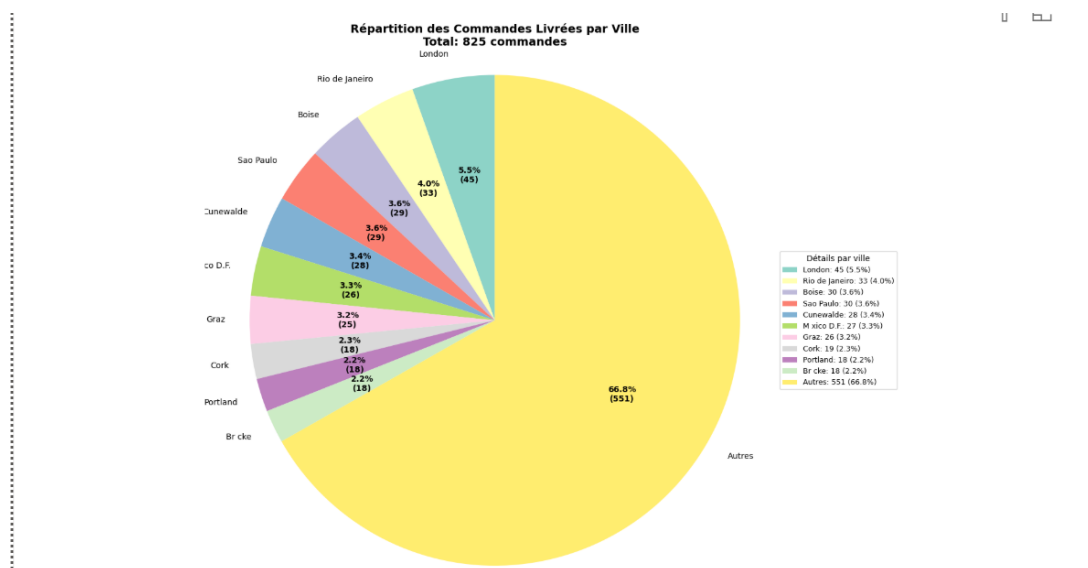


FIGURE 9 – Commandes Livrées par Ville - Analyse géographique

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 :

```

1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # Agrégation par ville
6  df = dataset.groupby('City')['nbr_commande_livrees']\
7      .sum()\
8      .sort_values(ascending=False)\
9      .reset_index()
10
11 # Calculer les totaux
12 total_commandes = df['nbr_commande_livrees'].sum()
13
14 # Prendre top 10 villes

```

```

15 top_n = 10
16 df_top = df.head(top_n).copy()
17 autres_total = df.iloc[top_n:]['nbr_commande_livrees'].sum()
18
19 # Ajouter la ligne "Autres"
20 if autres_total > 0:
21     df_final = pd.concat([
22         df_top,
23         pd.DataFrame({'City': ['Autres'], 'nbr_commande_livrees': [
24             autres_total]})
25     ])
26 else:
27     df_final = df_top
28
29 # Calculer les pourcentages
30 df_final['pourcentage'] = (df_final['nbr_commande_livrees'] /
31     total_commandes) * 100
32
33 # Cr ation du graphique
34 plt.figure(figsize=(12, 10))
35
36 # Palette de couleurs
37 colors = plt.cm.Set3(np.linspace(0, 1, len(df_final)))
38
39 # Diagramme en secteurs
40 wedges, texts, autotexts = plt.pie(
41     df_final['nbr_commande_livrees'],
42     labels=df_final['City'],
43     autopct=lambda pct: f'{pct:.1f}%\n({int(pct*total_commandes/100)
44         :.1f})',
45     colors=colors,
46     startangle=90,
47     textprops={'fontsize': 10}
48 )
49
50 # Am liorer la lisibilit
51 for autotext in autotexts:
52     autotext.set_color('black')
53     autotext.set_fontweight('bold')
54
55 plt.title(f'R partition des Commandes Livr es par Ville\nTotal: {
56     total_commandes:,} commandes',
57     fontsize=14, fontweight='bold')
58 plt.axis('equal')
59
60 # Ajouter une l gende avec les valeurs d taill es
61 plt.legend(
62     wedges,
63     [f"{row.City}: {row.nbr_commande_livrees:,} ({row.pourcentage:.1
64         f}%)"
65         for _, row in df_final.iterrows()],
66     title="D tails par ville",
67     loc="center left",
68     bbox_to_anchor=(1, 0, 0.5, 1),
69     fontsize=9
70 )
71
72 plt.tight_layout()

```

```
plt.show()
```

Listing 9 – Script Python pour Commandes par Ville

5.4 Visualisation 4 : Évolution Temporelle des Régions

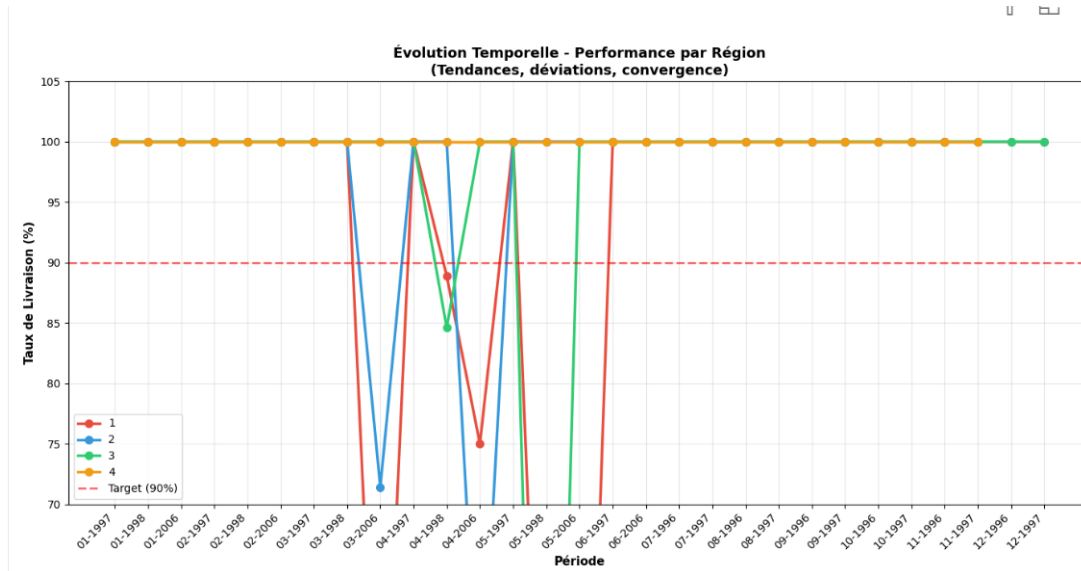


FIGURE 10 – Évolution Temporelle - Performance par Région

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 :

```
1 # VISUALISATION 4 : ÉVOLUTION TEMPORELLE DES RÉGIONS
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 df = dataset.copy()
6
7 # Analyser taux par région et période
8 region_time = df.groupby(['RegionID', 'mois_annee']).agg({
9     'nbr_commande_livrees': 'sum',
10    'nbr_commande_non_livrees': 'sum'
11 }).reset_index()
12
13 region_time['total'] = region_time['nbr_commande_livrees'] +
14     region_time['nbr_commande_non_livrees']
15 region_time['taux'] = (region_time['nbr_commande_livrees'] /
16     region_time['total'] * 100)
17
18 fig, ax = plt.subplots(figsize=(14, 7))
19 regions = sorted(region_time['RegionID'].unique())
```



```

19 colors_list = ['#e74c3c', '#3498db', '#2ecc71', '#f39c12', '#9b59b6',
20 ]
21 for i, region in enumerate(regions):
22     data = region_time[region_time['RegionID'] == region].
23         sort_values('mois_annee')
24     ax.plot(range(len(data)), data['taux'].values, 'o-', linewidth
25             =2.5, markersize=7,
26             label=region, color=colors_list[i % len(colors_list)])
27
28 ax.axhline(y=90, color='red', linestyle='--', linewidth=2, alpha
29           =0.6, label='Target (90%)')
30 ax.set_xlabel('P riode', fontweight='bold', fontsize=11)
31 ax.set_ylabel('Taux de Livraison (%)', fontweight='bold', fontsize
32              =11)
33 ax.set_title('volution Temporelle - Performance par R gion\n(
34             Tendances, d viations, convergence)',
35             fontsize=13, fontweight='bold')
36
37 # R cup rer les labels de mois uniques
38 periods = region_time['mois_annee'].unique()
39 ax.set_xticks(range(len(periods)))
40 ax.set_xticklabels(periods, rotation=45, ha='right')
41
42 ax.set_ylim(70, 105)
43 ax.grid(True, alpha=0.3)
44 ax.legend(loc='best', fontsize=10)
45
46 plt.tight_layout()
47 plt.show()

```

Listing 10 – Script Python pour Évolution Temporelle

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

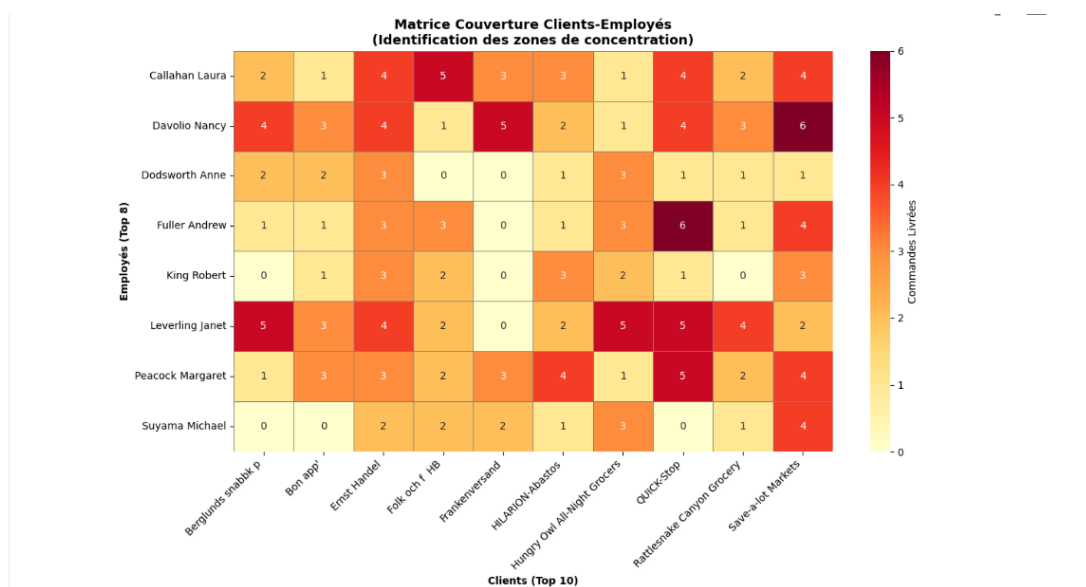


FIGURE 11 – Matrice Couverture Clients-Employés

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 :

```
1 # VISUALISATION 5 : MATRICE DE COMMUNICATION - QUI VEND      QUI
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 df = dataset.copy()
7 df['Employe'] = df['Nom'] + ' ' + df['Prenom']
8
9 # Cr e r une matrice Employ e x Client (Top 10 clients, Top 8
   employ es)
10 top_clients = df.groupby('CompanyName')['nbr_commande_livrees'].sum
   ().nlargest(10).index
11 top_emps = df.groupby('Employe')['nbr_commande_livrees'].sum().
   nlargest(8).index
12
13 df_filtered = df[(df['CompanyName'].isin(top_clients)) & (df['
   Employe'].isin(top_emps))]
14
15 matrice = df_filtered.groupby(['Employe', 'CompanyName'])['
   nbr_commande_livrees'].sum().unstack(fill_value=0)
16
17 plt.figure(figsize=(12, 8))
18 sns.heatmap(matrice, annot=True, fmt='g', cmap='YlOrRd', cbar_kws={'
   label': 'Commandes Livr es'},
19               linewidths=0.5, linecolor='gray')
20 plt.title('Matrice Couverture Clients-Employ es\n(Identification des
   zones de concentration)',
21           fontsize=13, fontweight='bold')
22 plt.xlabel('Clients (Top 10)', fontweight='bold')
23 plt.ylabel('Employ es (Top 8)', fontweight='bold')
24 plt.xticks(rotation=45, ha='right')
25 plt.tight_layout()
26 plt.show()
```

Listing 11 – Script Python pour Matrice de Communication

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

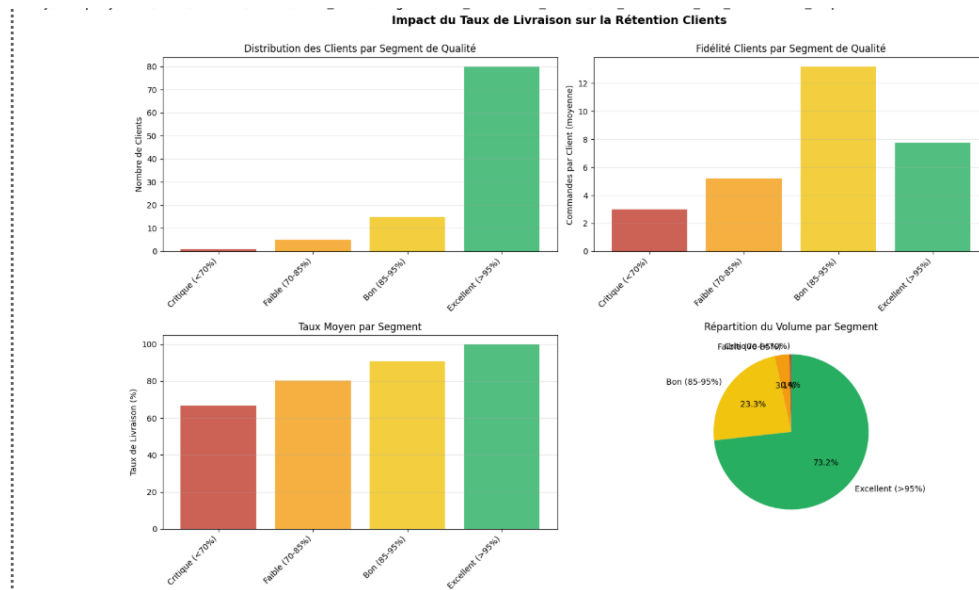


FIGURE 12 – Impact du Taux de Livraison sur la Rétention Clients

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 :

```
1 # VISUALISATION 6 : IMPACT DU TAUX DE LIVRAISON SUR LA R TENTION
  CLIENTS
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 df = dataset.copy()
7 df['total_commandes'] = df['nbr_commande_livrees'] + df['
  nbr_commande_non_livrees']
8 df['taux_livraison'] = (df['nbr_commande_livrees'] / df['
  total_commandes'] * 100)
9
10 # Segmenter les clients par taux et analyser leur fid lit
11 client_seg = df.groupby('CompanyName').agg({
12     'total_commandes': 'sum',
13     'taux_livraison': 'mean',
14     'id_seqClient': 'count' # Nombre de transactions
15 }).reset_index()
16
17 client_seg.columns = ['CompanyName', 'total_commandes', '
  taux_livraison', 'nb_transactions']
18
19 # Cr er des bacs de taux
20 client_seg['segment_taux'] = pd.cut(client_seg['taux_livraison'],
21     bins=[0, 70, 85, 95, 100],
```

```

22         labels=['Critique (<70%)', '
23             Faible (70-85%)', 'Bon
24             (85-95%)', 'Excellent (>95%)
25             '])
26
27 fig, axes = plt.subplots(2, 2, figsize=(15, 10))
28
29 # Graph 1 : Nombre de clients par segment
30 seg_counts = client_seg.groupby('segment_taux').size()
31 axes[0, 0].bar(range(len(seg_counts)), seg_counts.values,
32               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
33               alpha=0.8)
34 axes[0, 0].set_xticks(range(len(seg_counts)))
35 axes[0, 0].set_xticklabels(seg_counts.index, rotation=45, ha='right'
36                             )
37 axes[0, 0].set_ylabel('Nombre de Clients')
38 axes[0, 0].set_title('Distribution des Clients par Segment de
39                       Qualit ')
40 axes[0, 0].grid(axis='y', alpha=0.3)
41
42 # Graph 2 : Commandes moyennes par client
43 seg_cmd = client_seg.groupby('segment_taux')['total_commandes'].mean
44 ()
45 axes[0, 1].bar(range(len(seg_cmd)), seg_cmd.values,
46               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
47               alpha=0.8)
48 axes[0, 1].set_xticks(range(len(seg_cmd)))
49 axes[0, 1].set_xticklabels(seg_cmd.index, rotation=45, ha='right')
50 axes[0, 1].set_ylabel('Commandes par Client (moyenne)')
51 axes[0, 1].set_title('Fid lit Clients par Segment de Qualit ')
52 axes[0, 1].grid(axis='y', alpha=0.3)
53
54 # Graph 3 : Taux de livraison moyen
55 seg_taux = client_seg.groupby('segment_taux')['taux_livraison'].mean
56 ()
57 axes[1, 0].bar(range(len(seg_taux)), seg_taux.values,
58               color=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
59               alpha=0.8)
60 axes[1, 0].set_xticks(range(len(seg_taux)))
61 axes[1, 0].set_xticklabels(seg_taux.index, rotation=45, ha='right')
62 axes[1, 0].set_ylabel('Taux de Livraison (%)')
63 axes[1, 0].set_title('Taux Moyen par Segment')
64 axes[1, 0].set_ylim(0, 105)
65 axes[1, 0].grid(axis='y', alpha=0.3)
66
67 # Graph 4 : Volume total par segment
68 seg_vol = client_seg.groupby('segment_taux')['total_commandes'].sum
69 ()
70 axes[1, 1].pie(seg_vol.values, labels=seg_vol.index, autopct='%1.1f
71                %%',
72                colors=['#c0392b', '#f39c12', '#f1c40f', '#27ae60'],
73                startangle=90)
74 axes[1, 1].set_title('R partition du Volume par Segment')
75
76 plt.suptitle('Impact du Taux de Livraison sur la R tention Clients'
77             ,
78             fontsize=14, fontweight='bold', y=1.00)
79 plt.tight_layout()

```

6 Timeline Interactive du Projet

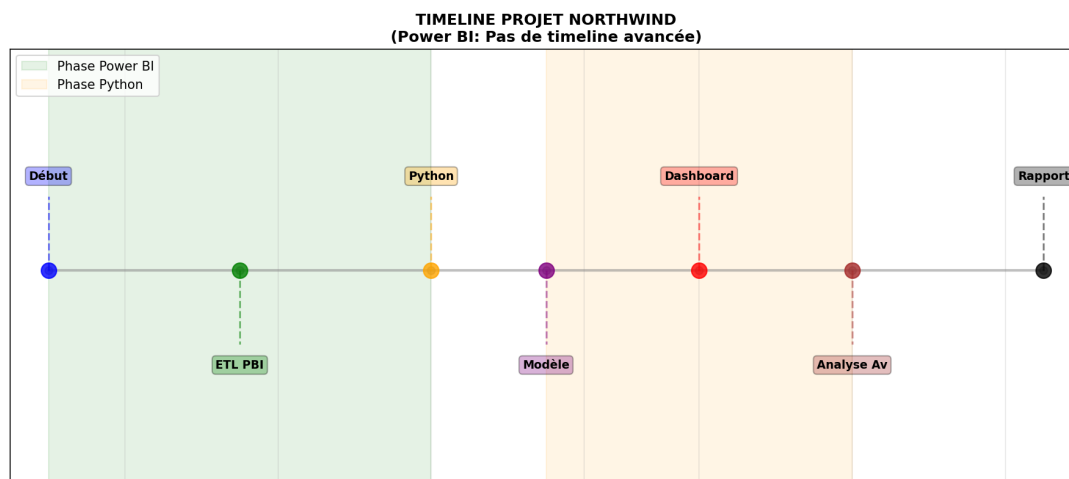


FIGURE 13 – Timeline Interactive - Phases du Projet Northwind BI

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.