Rapport de Projet TP de module
TBI-SIG

Section : ***M2 BIGDATA***
Nom : ***AMIMER***
Prénom : ***Abderrahmane Mohamed Elmahdi***
Matricule : ***202031049121***

Introduction:

Le but de ce projet est de concevoir un tableau de bord complète qui regroupe l'ensemble des données ***Northwind***, issues à la fois de la base de données 'Northwind' sous SQL SERVER, et le fichier Access associé. L'établissement de ce tableau de bord nécessite une opération ETL réalisée dans **Power Bi** et une visualisation des données sur **Power Bi** et utilisant **des bibliothèques python**.

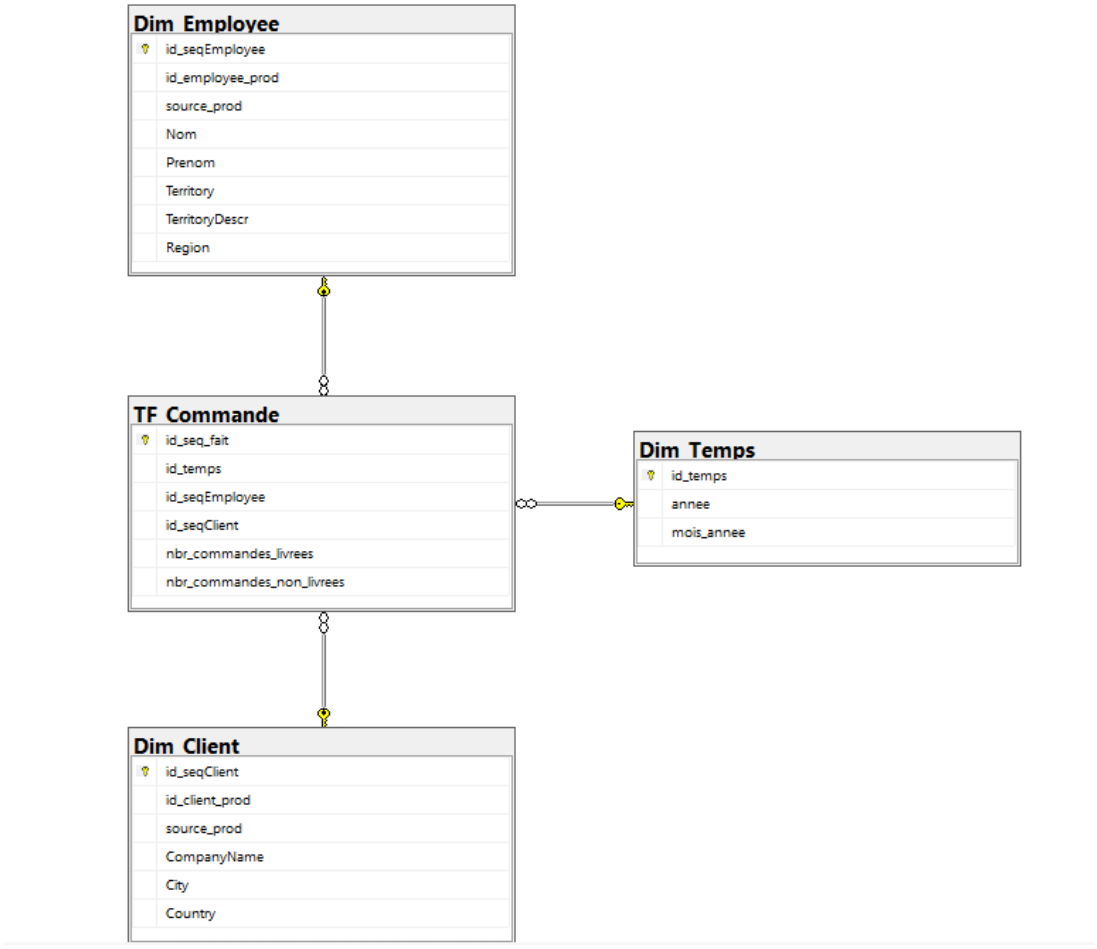
1- *Processus ETL :*

Dans notre cas, le processus ETL consiste à se connecter à la base de données SQL Server afin d'en extraire les données nécessaires pour remplir les dimensions et la table de fait de notre entrepôt de données, qui sont les tables :

- **Orders** (pour la dimension Temps et la table de fait)
- **Employees, Employee Territories, Territory, Region** (pour la dimension Employee)
- **Customers** (pour la dimension Client)

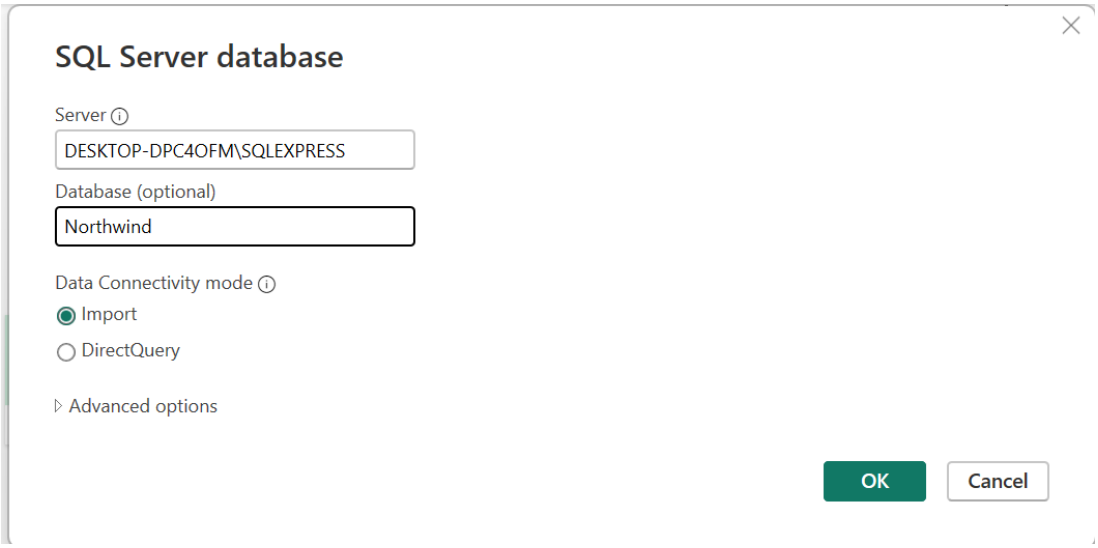
Tout en convertissant au préalable les trois tables requises du fichier Access en fichiers CSV pour pouvoir les intégrer (**Orders.csv** , **Employee.csv**, **Customer.csv**). Ces différentes sources sont ensuite extraites, transformées et chargées dans l'entrepôt de données, afin de préparer un ensemble cohérent et exploitable pour la construction du tableau de bord.

- La structure de l'entrepôt de données :

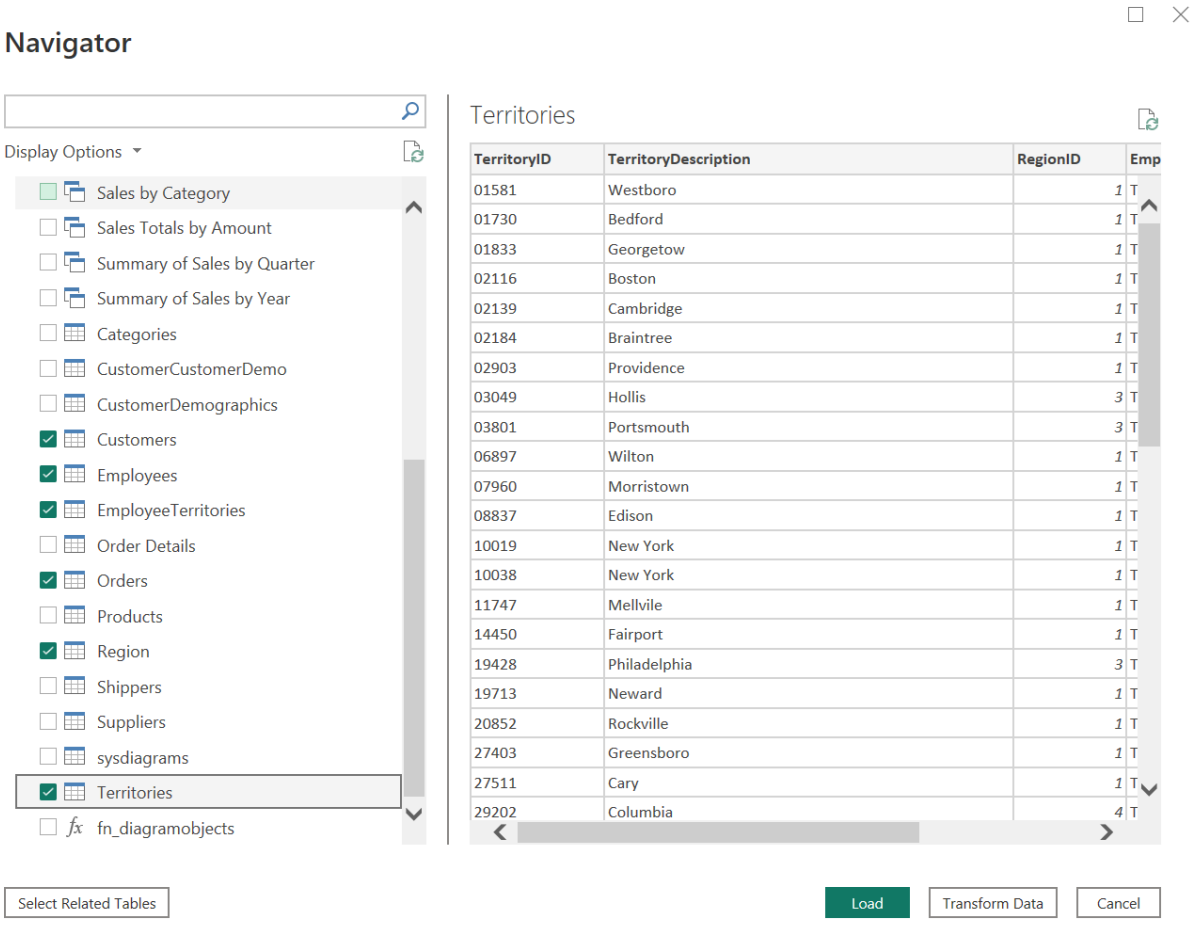


1.1. Extraction des données :

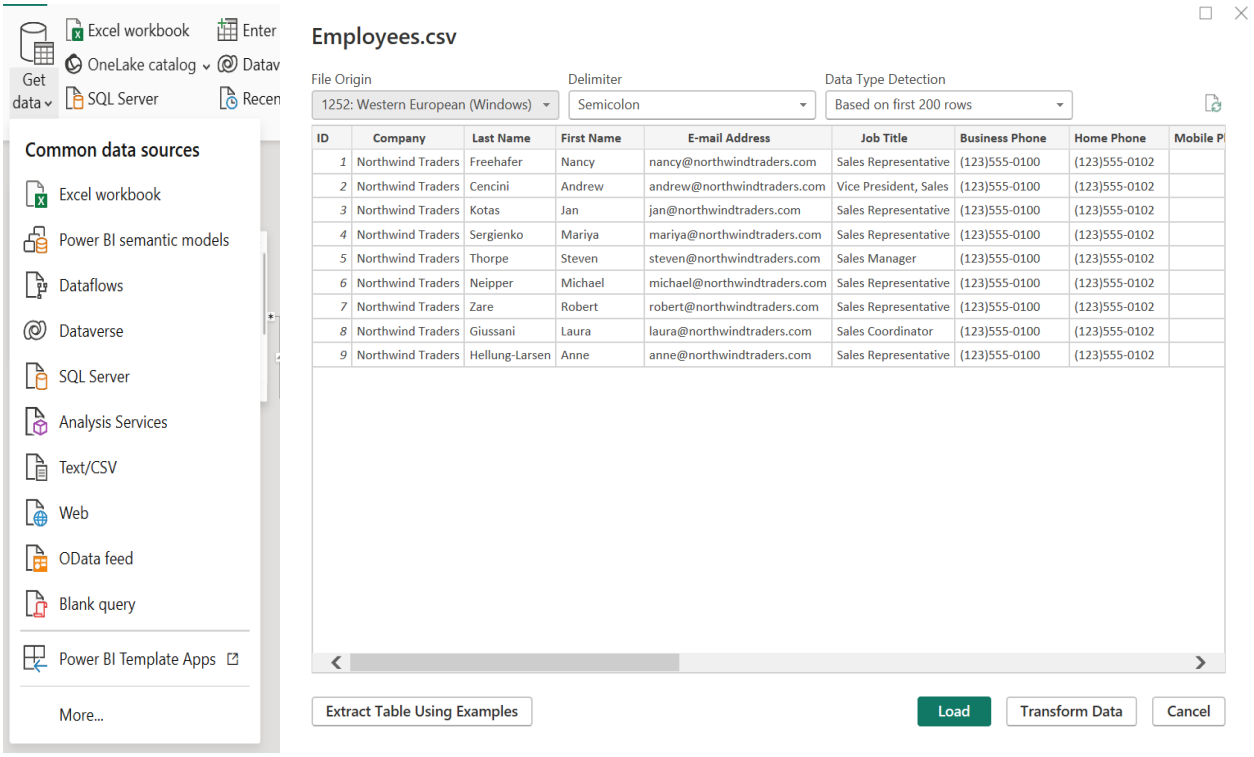
- La connexion de Power Bi avec la base de données **Northwind** dans SQL SERVER :



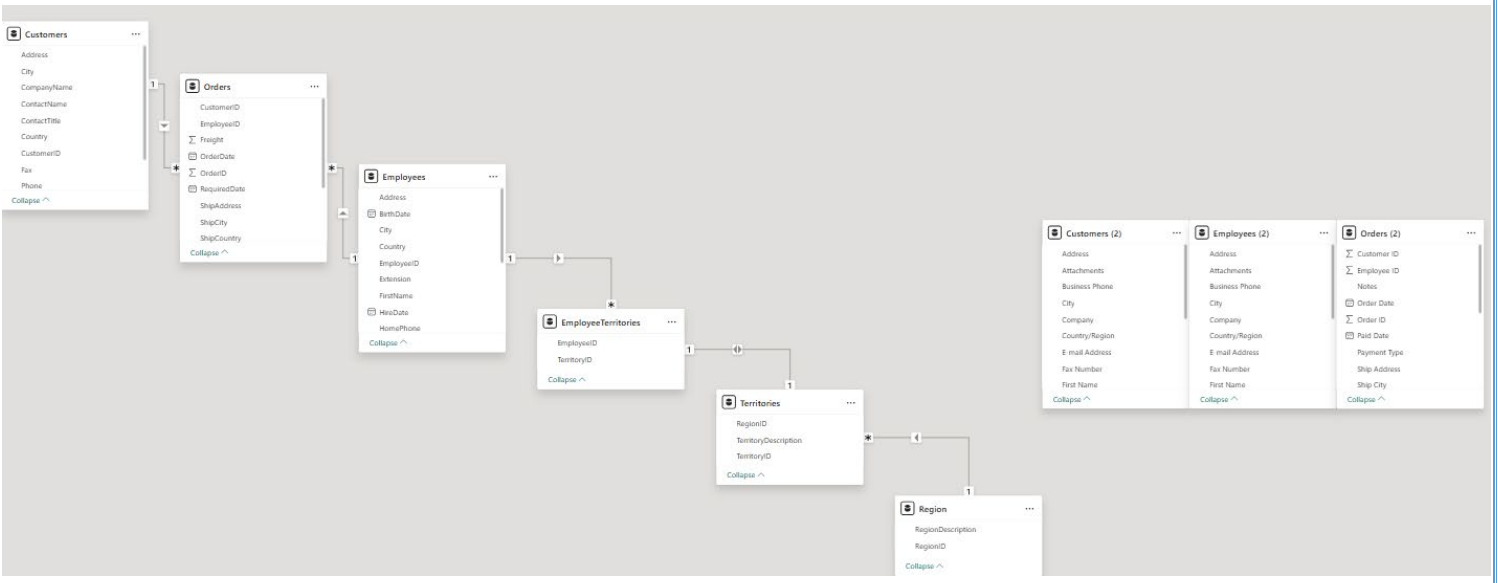
- Et on extraire les tables nécessaires depuis la base de données :



- Et on extraire les données des fichiers .csv par choisit l'option (Text/CSV) de menu **Get data :**

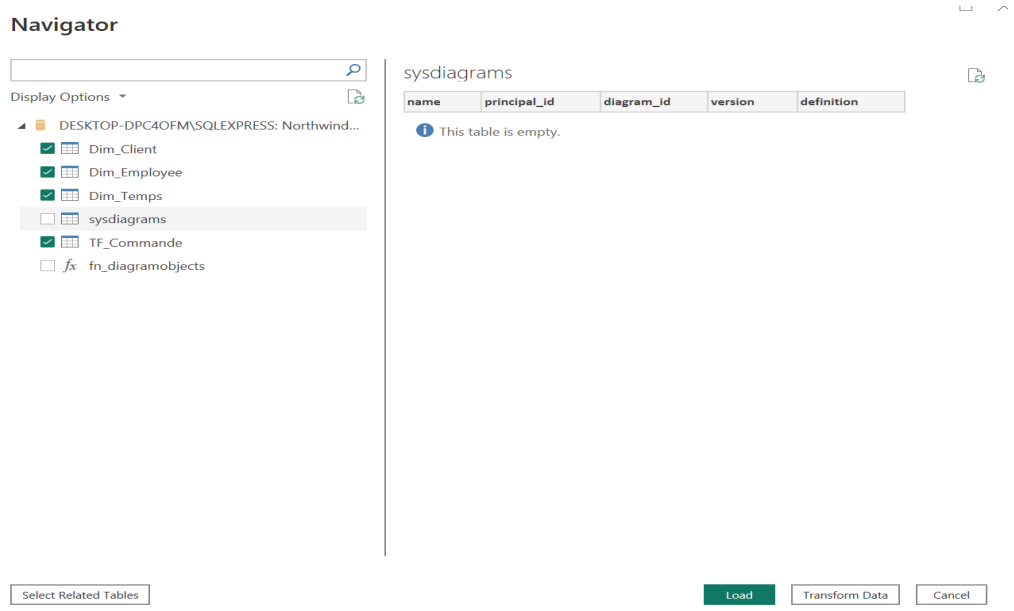


- Et on fait ça pour tous les autres fichiers .csv, la relation entre les tableaux sera etre détécté automatiquement comme suit (tables de SQL SERVER à droite et fichiers csv à gauche):



- On fait aussi une extraction des dimensions et table de fait de l'entrepôt de données **vide** (créé dans SQL SERVER) afin d'obtenir la structure nécessaire et de pouvoir les renseigner directement.

Même si ce processus ne remplit pas le véritable entrepôt de données sous SQL Server, il permet néanmoins d'en reproduire la structure pour faciliter le travail et l'organisation des données :



1.2. Transformation des données :

Transformation de données pour le remplissage des dimensions et de table de fait:

1.2.1. Dimension Client :

- On supprime les colonnes non nécessaires de table customer.csv et conserve uniquement celles dont nous avons besoin et modifiant le type de id_client_produit de *'Integer'* à *'Text'*, et ajoutant l'attribut source_produit, le résultat :

```
= Table.RemoveColumns(#"Added Custom",{"Job Title", "Last Name", "First Name", "E-mail Address", "Business Phone", "Home Phone", "Mobile
```

	id_client_prod	Company	City	Country/Region	source_prod
1	1	Company A	Seattle	USA	Access
2	2	Company B	Boston	USA	Access
3	3	Company C	Los Angelas	USA	Access
4	4	Company D	New York	USA	Access
5	5	Company E	Minneapolis	USA	Access
6	6	Company F	Milwaukee	USA	Access
7	7	Company G	Boise	USA	Access
8	8	Company H	Portland	USA	Access
9	9	Company I	Salt Lake City	USA	Access
10	10	Company J	Chicago	USA	Access
11	11	Company K	Miami	USA	Access
12	12	Company L	Las Vegas	USA	Access
13	13	Company M	Memphis	USA	Access
14	14	Company N	Denver	USA	Access
15	15	Company O	Honolulu	USA	Access
16	16	Company P	San Francisco	USA	Access
17	17	Company Q	Seattle	USA	Access
18	18	Company R	Boston	USA	Access
19	19	Company S	Los Angelas	USA	Access
20	20	Company T	New York	USA	Access
21	21	Company U	Minneapolis	USA	Access

- On fait le même chose pour la table Customer de la base de données de SQL SERVER :

	id_client_prod	CompanyName	City	Country	source_prod
1	ALFKI	Alfreds Futterkiste	Berlin	Germany	SQL_SERVER
2	ANATR	Ana Trujillo Emparedados y helados	M xico D.F.	Mexico	SQL_SERVER
3	ANTON	Antonio Moreno Taquer a	M xico D.F.	Mexico	SQL_SERVER
4	AROUT	Around the Horn	London	UK	SQL_SERVER
5	BERGS	Berglunds snabbk p	Lule	Sweden	SQL_SERVER
6	BLAUS	Blauer See Delikatessen	Mannheim	Germany	SQL_SERVER
7	BLONP	Blondesddsl p re et fils	Strasbourg	France	SQL_SERVER
8	BOLID	B lido Comidas preparadas	Madrid	Spain	SQL_SERVER
9	BONAP	Bon app'	Marseille	France	SQL_SERVER
10	BOTTM	Bottom-Dollar Markets	Tsawassen	Canada	SQL_SERVER
11	BSBEV	B's Beverages	London	UK	SQL_SERVER
12	CACTU	Cactus Comidas para llevar	Buenos Aires	Argentina	SQL_SERVER
13	CENTC	Centro comercial Moctezuma	M xico D.F.	Mexico	SQL_SERVER
14	CHOPS	Chop-suey Chinese	Bern	Switzerland	SQL_SERVER
15	COMMI	Com rcio Mineiro	Sao Paulo	Brazil	SQL_SERVER
16	CONSH	Consolidated Holdings	London	UK	SQL_SERVER
17	DRACD	Drachenblut Delikatessen	Aachen	Germany	SQL_SERVER
18	DUMON	Du monde entier	Nantes	France	SQL_SERVER
19	EASTC	Eastern Connection	London	UK	SQL_SERVER
20	ERNSH	Ernst Handel	Graz	Austria	SQL_SERVER
21	FAMIA	Familia Arquibaldo	Sao Paulo	Brazil	SQL_SERVER

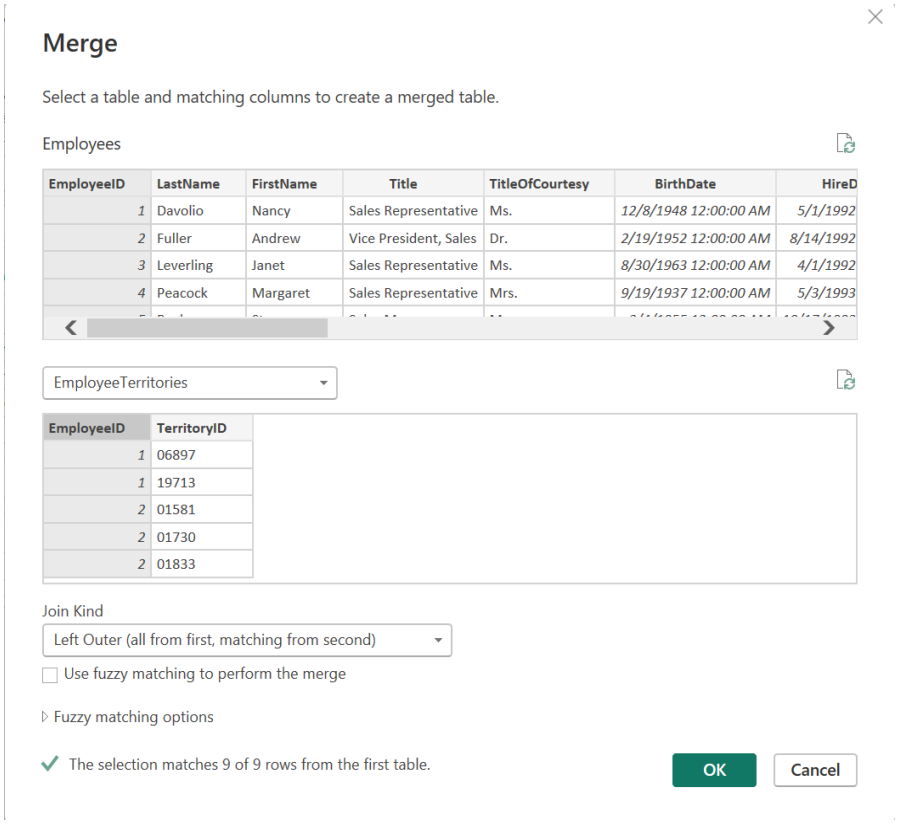
- On utilise la fonction **Combine** pour concater les tables dans la table *'Dim_Client'* et on ajoute l'attribut id_seq_client qui est un index sur la dimension :

```
= Table.RemoveColumns(#"Changed Type1",{"CustomerCustomerDemo", "Orders"})
```

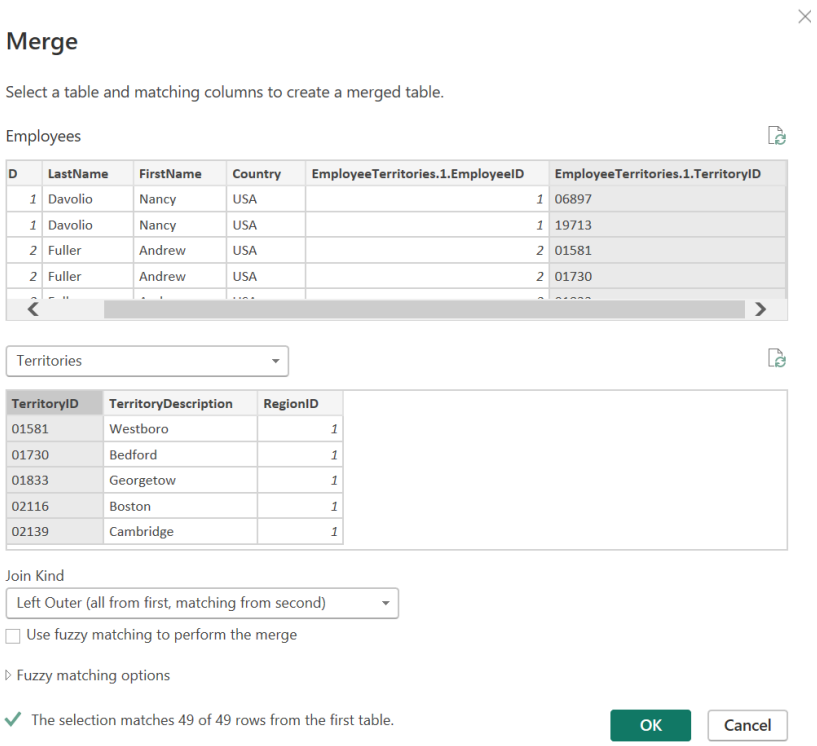
	id_seqClient	id_client_prod	source_prod	CompanyName	City	Country
16	16	16	Access	Company P	San Francisco	USA
17	17	17	Access	Company Q	Seattle	USA
18	18	18	Access	Company R	Boston	USA
19	19	19	Access	Company S	Los Angelas	USA
20	20	20	Access	Company T	New York	USA
21	21	21	Access	Company U	Minneapolis	USA
22	22	22	Access	Company V	Milwaukee	USA
23	23	23	Access	Company W	Portland	USA
24	24	24	Access	Company X	Salt Lake City	USA
25	25	25	Access	Company Y	Chicago	USA
26	26	26	Access	Company Z	Miami	USA
27	27	27	Access	Company AA	Las Vegas	USA
28	28	28	Access	Company BB	Memphis	USA
29	29	29	Access	Company CC	Denver	USA
30	30	ALFKI	SQL_SERVER	Alfreds Futterkiste	Berlin	Germany
31	31	ANATR	SQL_SERVER	Ana Trujillo Emparedados y helados	M xico D.F.	Mexico
32	32	ANTON	SQL_SERVER	Antonio Moreno Taquer a	M xico D.F.	Mexico
33	33	AROUT	SQL_SERVER	Around the Horn	London	UK
34	34	BERGS	SQL_SERVER	Berglunds snabbk p	Lule	Sweden
35	35	BLAUS	SQL_SERVER	Blauer See Delikatessen	Mannheim	Germany

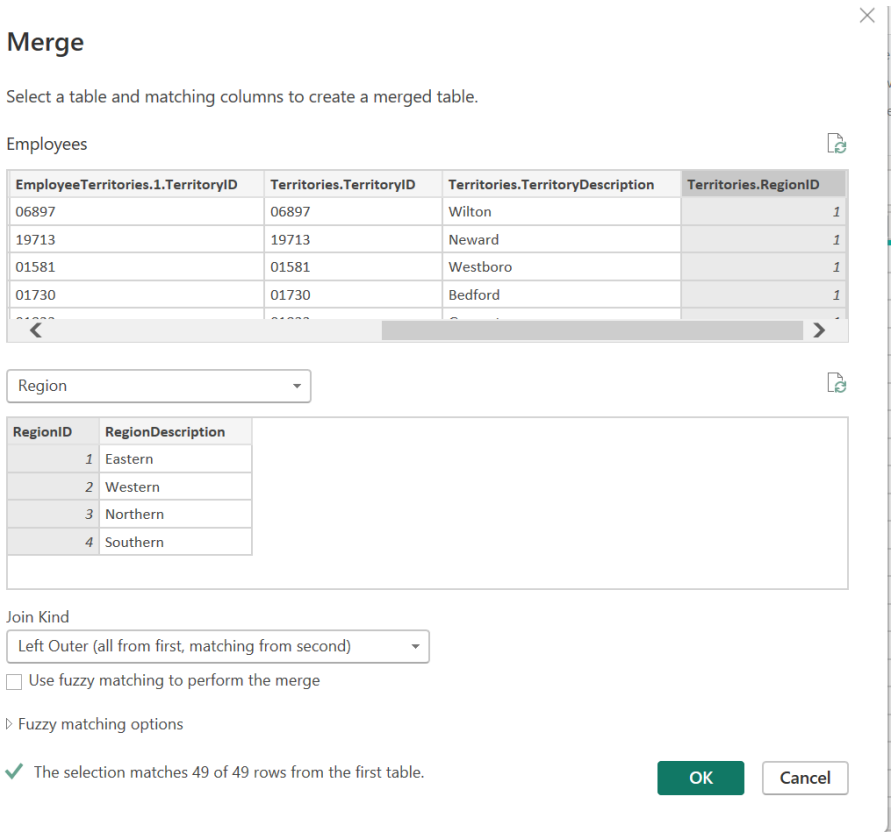
1.2.2. Dimension Employee :

- On utilise l'opération **Merge** pour fait une jointure sur la table *Employee* et *EmployeeTerritories* de la base de données SQL SERVER :



- Après on utilise **Merge** sur la table résultat *(Employee Join EmployeeTerritories)* avec la table *Territory* :





- On supprime les colonnes non nécessaires et conserve uniquement celles dont nous avons besoin et ajoutant l'attribut source_produit, le résultat :

```
= Table.AddColumn(#"Renamed Columns", "source_prod", each "SQL_SERVER")
```

	Prenom	Country	Territory	TerritoryDescr	Region	source_prod
1	Nancy	USA	06897	Wilton	Eastern	SQL_SERVER
2	Nancy	USA	19713	Neward	Eastern	SQL_SERVER
3	Andrew	USA	01581	Westboro	Eastern	SQL_SERVER
4	Andrew	USA	01730	Bedford	Eastern	SQL_SERVER
5	Andrew	USA	01833	Georgetow	Eastern	SQL_SERVER
6	Andrew	USA	02116	Boston	Eastern	SQL_SERVER
7	Andrew	USA	02139	Cambridge	Eastern	SQL_SERVER
8	Andrew	USA	02184	Braintree	Eastern	SQL_SERVER
9	Andrew	USA	40222	Louisville	Eastern	SQL_SERVER
10	Janet	USA	30346	Atlanta	Southern	SQL_SERVER
11	Janet	USA	31406	Savannah	Southern	SQL_SERVER
12	Janet	USA	32859	Orlando	Southern	SQL_SERVER
13	Janet	USA	33607	Tampa	Southern	SQL_SERVER
14	Margaret	USA	20852	Rockville	Eastern	SQL_SERVER
15	Margaret	USA	27403	Greensboro	Eastern	SQL_SERVER
16	Margaret	USA	27511	Cary	Eastern	SQL_SERVER
17	Steven	UK	02903	Providence	Eastern	SQL_SERVER
18	Steven	UK	07960	Morristown	Eastern	SQL_SERVER
19	Steven	UK	08837	Edison	Eastern	SQL_SERVER
20	Steven	UK	10019	New York	Eastern	SQL_SERVER

- Pour la table de fichier Employee.csv on remarque l'existence d'une colonne nommé *'Country/Region'* qui contient l'information de pays de l'employé :

Country/Region
USA
USA
USA
USA
USA
USA
USA
USA
USA

- Dans notre dimension **Dim_Employee** on est besoin de l'information '*Region'* qui est le pays dans Employee.csv et la région (eastern, western …) dans les tables de SQL SERVER, alors pour éviter la confusion on ajoute une colonne dans notre dimension **Dim_Employee** nommé *'Country'* qui aura l'information pays de l'employé et la colonne '*Region*' déjà existant aura l'information région, donc pour les données de Employee.csv l'attribut *Region* (dans la dimension **Dim_Employee**) sera être NULL :

Donc la dimension Employee après la concaténation des deux tables :

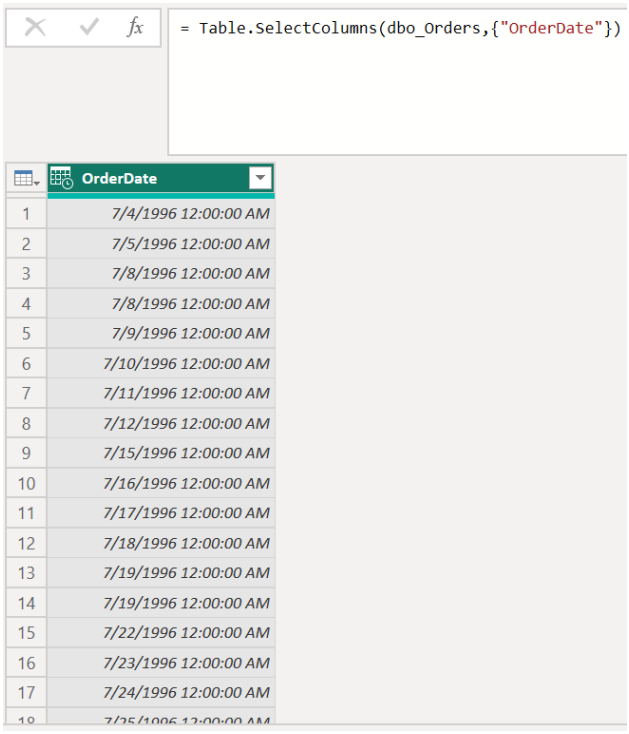
```
= Table.ReorderColumns(#"Renamed Columns1",{"id_seqEmployee", "id_employee_prod", "Nom", "Prenom", "Territory", "source_prod",
```

	Prenom	Territory	source_prod	TerritoryDescr	Country	Region
1	Nancy	*null*	Access	Seattle	USA	*null*
2	Andrew	*null*	Access	Bellevue	USA	*null*
3	Jan	*null*	Access	Redmond	USA	*null*
4	Mariya	*null*	Access	Kirkland	USA	*null*
5	Steven	*null*	Access	Seattle	USA	*null*
6	Michael	*null*	Access	Redmond	USA	*null*
7	Robert	*null*	Access	Seattle	USA	*null*
8	Laura	*null*	Access	Redmond	USA	*null*
9	Anne	*null*	Access	Seattle	USA	*null*
10	Nancy	06897	SQL_SERVER	Wilton	USA	Eastern
11	Nancy	19713	SQL_SERVER	Neward	USA	Eastern
12	Andrew	01581	SQL_SERVER	Westboro	USA	Eastern
13	Andrew	01730	SQL_SERVER	Bedford	USA	Eastern
14	Andrew	01833	SQL_SERVER	Georgetow	USA	Eastern
15	Andrew	02116	SQL_SERVER	Boston	USA	Eastern
16	Andrew	02139	SQL_SERVER	Cambridge	USA	Eastern
17	Andrew	02184	SQL_SERVER	Braintree	USA	Eastern
18	Andrew	40222	SQL_SERVER	Louisville	USA	Eastern
19	Janet	30346	SQL_SERVER	Atlanta	USA	Southern
20	Janet	31406	SQL_SERVER	Savannah	USA	Southern

(L'attribut *Territory* de dimension est NULL dans les données de Employee.csv à cause de l'absence de l'information *TerritoryID* dans les données Access).

1.2.3. Dimension Temps :

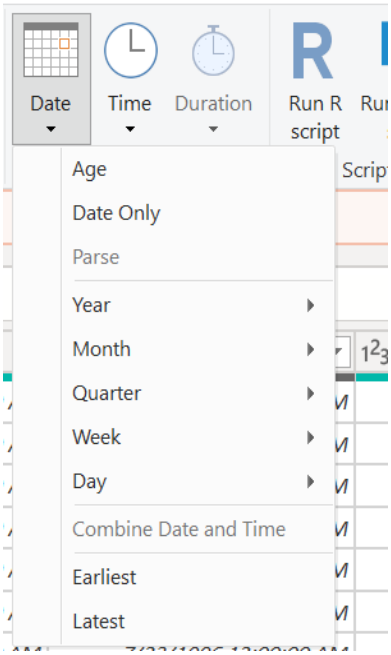
- On prend la colonne Order_date de table Orders de la base de données SQL SERVER dans une table individuellement :

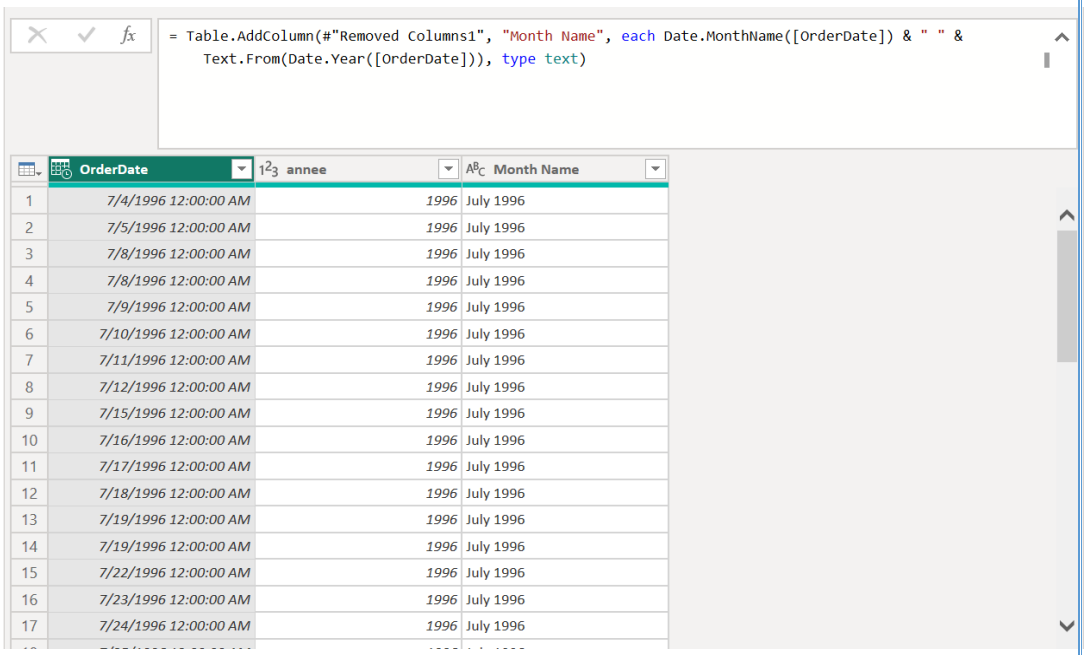


```
= Table.SelectColumns(dbo_Orders,{"OrderDate"})
```

	OrderDate
1	7/4/1996 12:00:00 AM
2	7/5/1996 12:00:00 AM
3	7/8/1996 12:00:00 AM
4	7/8/1996 12:00:00 AM
5	7/9/1996 12:00:00 AM
6	7/10/1996 12:00:00 AM
7	7/11/1996 12:00:00 AM
8	7/12/1996 12:00:00 AM
9	7/15/1996 12:00:00 AM
10	7/16/1996 12:00:00 AM
11	7/17/1996 12:00:00 AM
12	7/18/1996 12:00:00 AM
13	7/19/1996 12:00:00 AM
14	7/19/1996 12:00:00 AM
15	7/22/1996 12:00:00 AM
16	7/23/1996 12:00:00 AM
17	7/24/1996 12:00:00 AM

- Ajoutant un attribut Année et mois_année par les fonctions de l'opération ***Date*** qui permet de transformer les données des dates et les personnaliser:

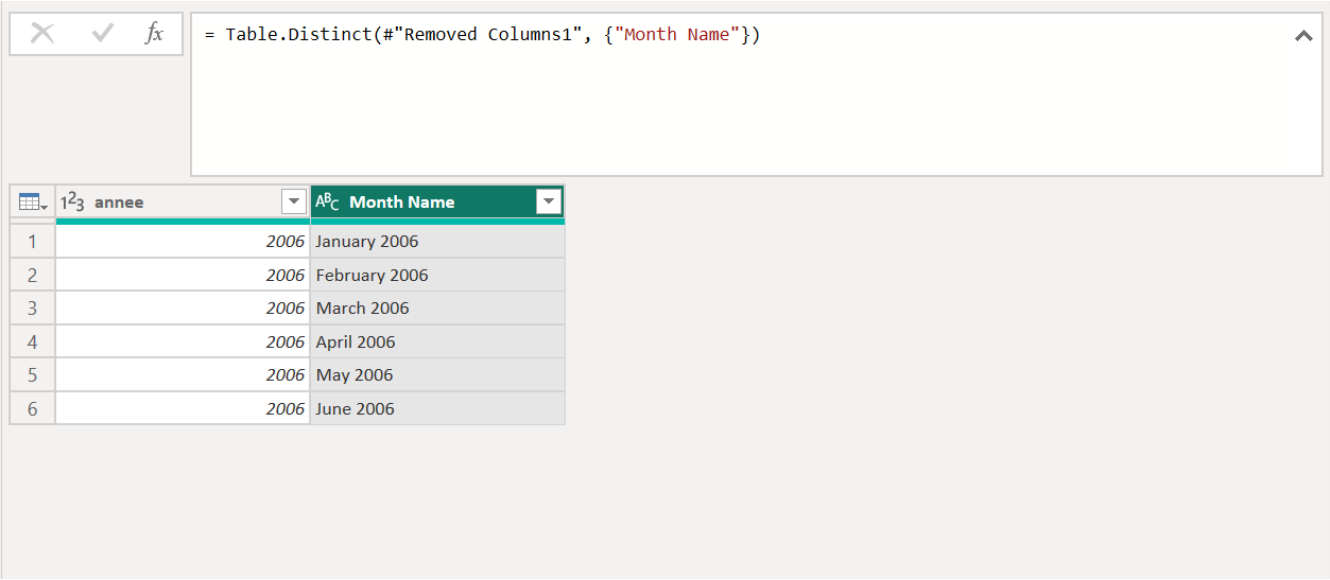




```
= Table.AddColumn(#"Removed Columns1", "Month Name", each Date.MonthName([OrderDate]) & " " &
    Text.From(Date.Year([OrderDate])), type text)
```

	OrderDate	annee	Month Name
1	7/4/1996 12:00:00 AM	1996	July 1996
2	7/5/1996 12:00:00 AM	1996	July 1996
3	7/8/1996 12:00:00 AM	1996	July 1996
4	7/8/1996 12:00:00 AM	1996	July 1996
5	7/9/1996 12:00:00 AM	1996	July 1996
6	7/10/1996 12:00:00 AM	1996	July 1996
7	7/11/1996 12:00:00 AM	1996	July 1996
8	7/12/1996 12:00:00 AM	1996	July 1996
9	7/15/1996 12:00:00 AM	1996	July 1996
10	7/16/1996 12:00:00 AM	1996	July 1996
11	7/17/1996 12:00:00 AM	1996	July 1996
12	7/18/1996 12:00:00 AM	1996	July 1996
13	7/19/1996 12:00:00 AM	1996	July 1996
14	7/19/1996 12:00:00 AM	1996	July 1996
15	7/22/1996 12:00:00 AM	1996	July 1996
16	7/23/1996 12:00:00 AM	1996	July 1996
17	7/24/1996 12:00:00 AM	1996	July 1996

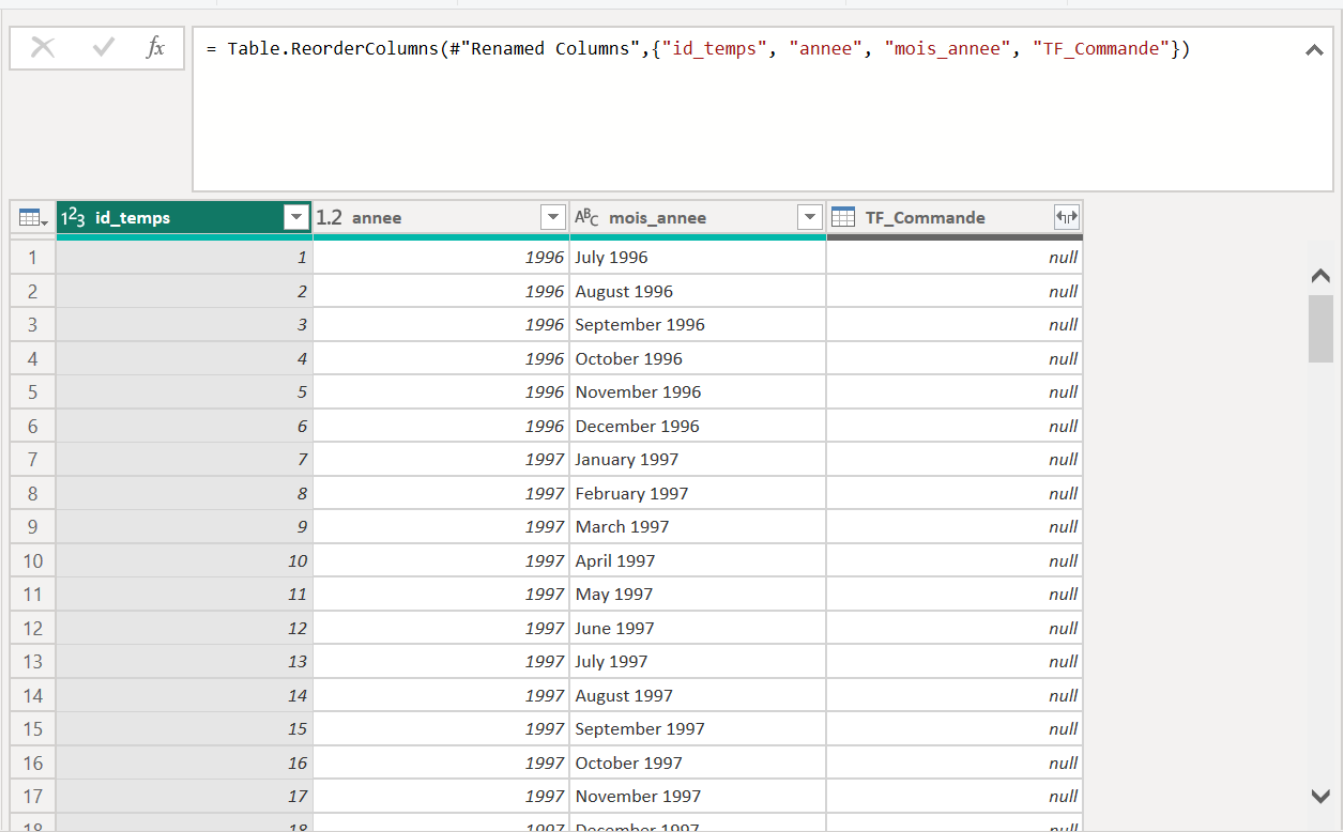
- On fait la même chose pour Orders.csv et on supprime les lignes dupliquant :



```
= Table.Distinct(#"Removed Columns1", {"Month Name"})
```

	annee	Month Name
1	2006	January 2006
2	2006	February 2006
3	2006	March 2006
4	2006	April 2006
5	2006	May 2006
6	2006	June 2006

- On concater les deux tables et supprimer les lignes dupliquant pour toutes :



```
= Table.ReorderColumns(#"Renamed Columns",{"id_temps", "annee", "mois_annee", "TF_Commande"})
```

	id_temps	annee	mois_annee	TF_Commande
1	1	1996	July 1996	null
2	2	1996	August 1996	null
3	3	1996	September 1996	null
4	4	1996	October 1996	null
5	5	1996	November 1996	null
6	6	1996	December 1996	null
7	7	1997	January 1997	null
8	8	1997	February 1997	null
9	9	1997	March 1997	null
10	10	1997	April 1997	null
11	11	1997	May 1997	null
12	12	1997	June 1997	null
13	13	1997	July 1997	null
14	14	1997	August 1997	null
15	15	1997	September 1997	null
16	16	1997	October 1997	null
17	17	1997	November 1997	null

- On veut créer un calendrier pour la dimension Temps, qui va contient tous les mois de plus ancienne disponible jusqu'à la plus récente, on peut faire ça exécutant ce script dans l'éditeur *Advanced Editor :*

```
let
    // SQL Database Source
    SourceSQL = Sql.Database("DESKTOP-DPC4OFM\SQLEXPRESS", "Northwind"),
    dbo_Orders = SourceSQL{[Schema="dbo",Item="Orders"]}[Data],
    #"SQL Removed Columns" = Table.SelectColumns(dbo_Orders,{"OrderDate"}),
    #"SQL Renamed" = Table.RenameColumns(#"SQL Removed Columns",{{"OrderDate", "Order Date"}}),

    // CSV Source
    SourceCSV = Csv.Document(File.Contents("path\Orders.csv"),[Delimiter=",", Columns=20, Encoding=65001, QuoteStyle=QuoteStyle.None]),
    #"CSV Promoted Headers" = Table.PromoteHeaders(SourceCSV, [PromoteAllScalars=true]),
    #"CSV Changed Type" = Table.TransformColumnTypes(#"CSV Promoted Headers",{{"Order Date", type datetime}}),
    #"CSV Removed Columns" = Table.SelectColumns(#"CSV Changed Type",{"Order Date"}),

    // Append both tables
    #"Appended Tables" = Table.Combine({#"SQL Renamed", #"CSV Removed Columns"}),

    // Get min and max dates from combined data
    MinDate = List.Min(#"Appended Tables"[Order Date]),
    MaxDate = List.Max(#"Appended Tables"[Order Date]),

    // Generate all months between min and max
    AllMonths = List.Generate(
        () => Date.StartOfMonth(MinDate),
        each _ <= MaxDate,
        each Date.AddMonths(_, 1)
    ),

    // Convert to table
    #"AllMonths Table" = Table.FromList(AllMonths, Splitter.SplitByNothing(), {"Order Date"}),
    #"Changed Type" = Table.TransformColumnTypes(#"AllMonths Table",{{"Order Date", type datetime}}),

    // Add year column
    #"Inserted Year" = Table.AddColumn(#"Changed Type", "annee", each Date.Year([Order Date]), Int64.Type),

    // Add month name column
```

```
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Year", "mois_annee", each
Date.MonthName([Order Date]) & " " & Text.From(Date.Year([Order Date])), type text),

    // Remove Order Date and get distinct
    #"Removed Columns" = Table.RemoveColumns(#"Inserted Month Name",{"Order Date"}),
    #"Removed Duplicates" = Table.Distinct(#"Removed Columns", {"mois_annee"})
in
    #"Removed Duplicates"
```

- Ce script va automatiquement inclus les deux tables ***Orders*** de SQL SERVER et Orders.csv, et prend la date la plus ancienne dans les deux tables et la plus récente, et supprimant tous les autres colonnes et affichant tous les années et mois entre la date max et la date min, la fenêtre de ***Advanced Editor*** :

- Et après l'exécution tous les mois et années qui n'existe pas dans les données des tables (de 1997 jusqu'à 2006) sera être dans notre dimension temps Pour un travail plus propre, mieux organisé et offrant une meilleure visualisation des données :

	id_temps	annee	mois_annee	TF_Commande
91	91	2004	January 2004	null
92	92	2004	February 2004	null
93	93	2004	March 2004	null
94	94	2004	April 2004	null
95	95	2004	May 2004	null
96	96	2004	June 2004	null
97	97	2004	July 2004	null
98	98	2004	August 2004	null
99	99	2004	September 2004	null
100	100	2004	October 2004	null
101	101	2004	November 2004	null
102	102	2004	December 2004	null
103	103	2005	January 2005	null
104	104	2005	February 2005	null
105	105	2005	March 2005	null
106	106	2005	April 2005	null
107	107	2005	May 2005	null
108	108	2005	June 2005	null
109	109	2005	July 2005	null
110	110	2005	August 2005	null
111	111	2005	September 2005	null

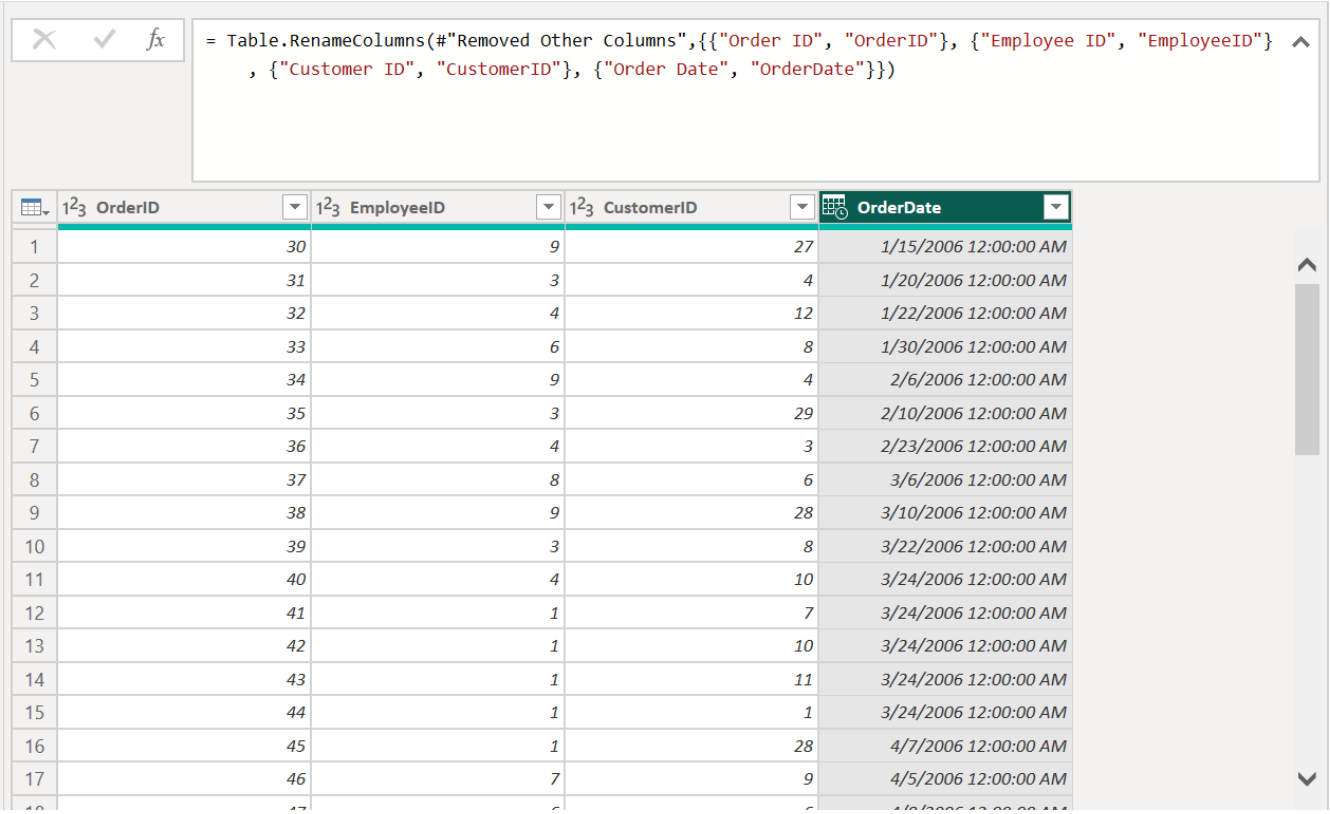
1.2.4. Table de fait :

- On prend la table Orders.csv et supprimer tous les colonnes non et conserve uniquement celles dont nous avons besoin :

```
= Table.RemoveColumns(dbo_Orders,{"RequiredDate", "ShippedDate", "ShipVia", "Freight", "ShipName",
    "ShipAddress", "ShipCity", "ShipRegion", "ShipPostalCode", "ShipCountry", "Customers",
    "Employees", "Order Details", "Shippers"})
```

	OrderID	CustomerID	EmployeeID	OrderDate
1	10248	VINET	5	7/4/1996 12:00:00 AM
2	10249	TOMSP	6	7/5/1996 12:00:00 AM
3	10250	HANAR	4	7/8/1996 12:00:00 AM
4	10251	VICTE	3	7/8/1996 12:00:00 AM
5	10252	SUPRD	4	7/9/1996 12:00:00 AM
6	10253	HANAR	3	7/10/1996 12:00:00 AM
7	10254	CHOPS	5	7/11/1996 12:00:00 AM
8	10255	RICSU	9	7/12/1996 12:00:00 AM
9	10256	WELLI	3	7/15/1996 12:00:00 AM
10	10257	HILAA	4	7/16/1996 12:00:00 AM
11	10258	ERNSH	1	7/17/1996 12:00:00 AM
12	10259	CENTC	4	7/18/1996 12:00:00 AM
13	10260	OTTIK	4	7/19/1996 12:00:00 AM
14	10261	QUEDE	4	7/19/1996 12:00:00 AM
15	10262	RATTC	8	7/22/1996 12:00:00 AM
16	10263	ERNSH	9	7/23/1996 12:00:00 AM
17	10264	FOLKO	6	7/24/1996 12:00:00 AM

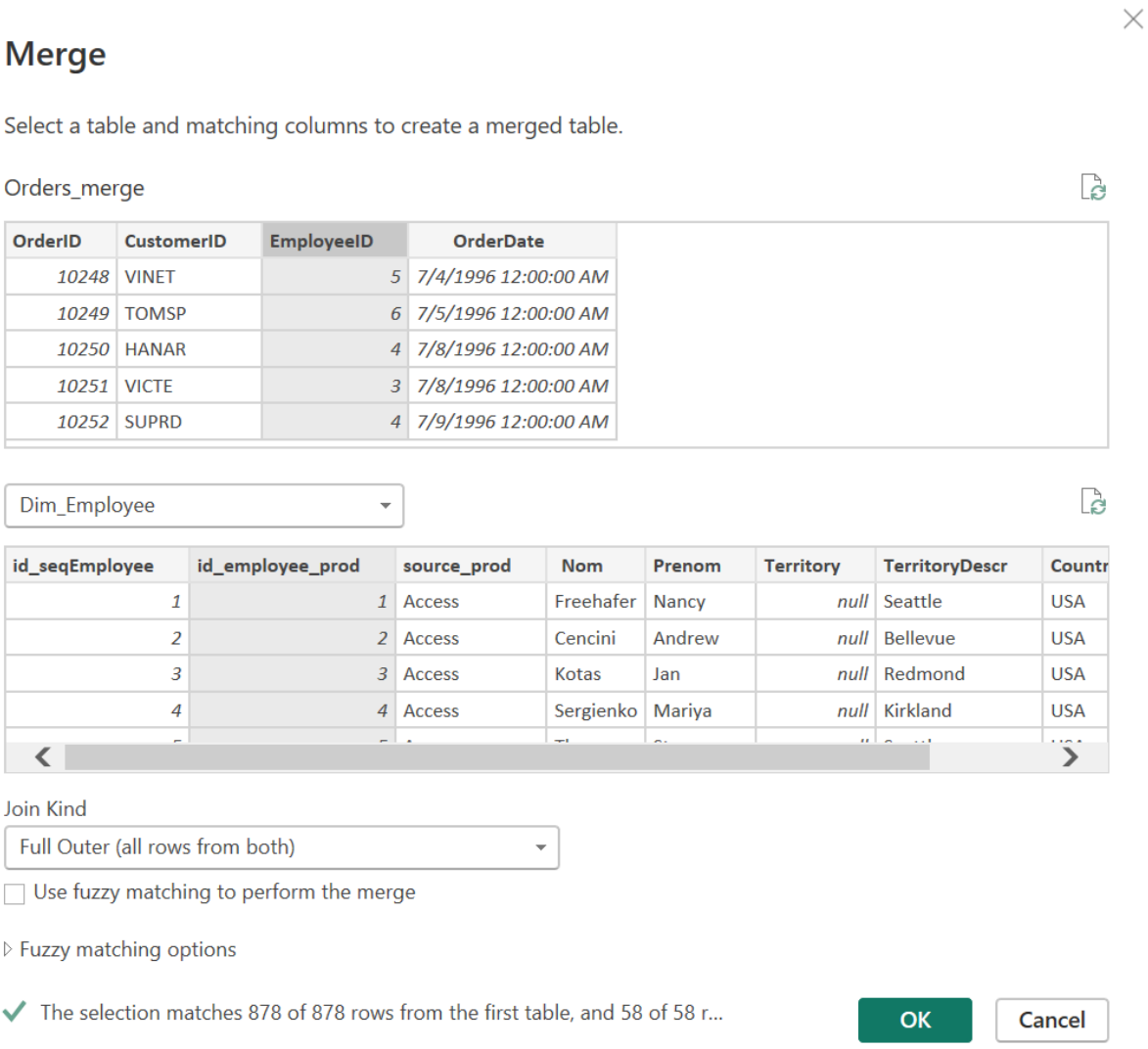
- On fait la même chose pour la table ***Orders*** de base de données SQL SERVER :



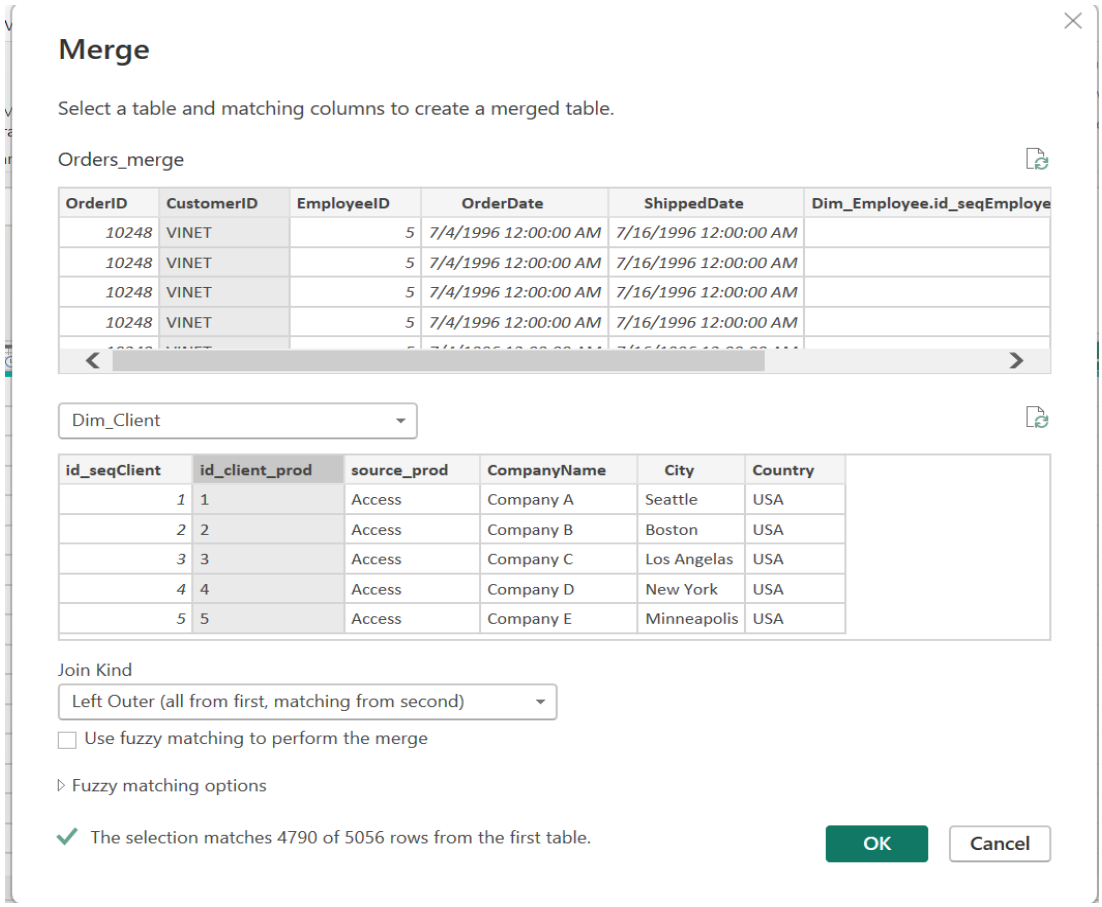
```
= Table.RenameColumns(#"Removed Other Columns",{{"Order ID", "OrderID"}, {"Employee ID", "EmployeeID"}
    , {"Customer ID", "CustomerID"}, {"Order Date", "OrderDate"}})
```

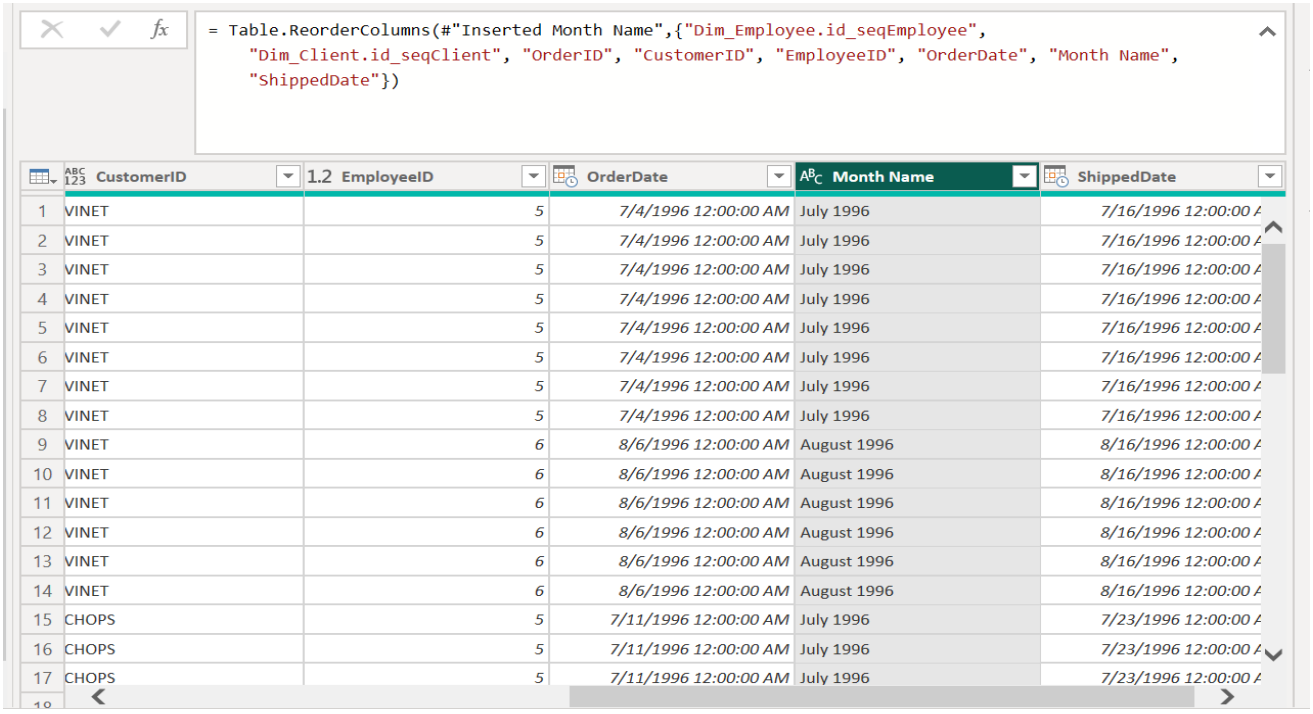
	OrderID	EmployeeID	CustomerID	OrderDate
1	30	9	27	1/15/2006 12:00:00 AM
2	31	3	4	1/20/2006 12:00:00 AM
3	32	4	12	1/22/2006 12:00:00 AM
4	33	6	8	1/30/2006 12:00:00 AM
5	34	9	4	2/6/2006 12:00:00 AM
6	35	3	29	2/10/2006 12:00:00 AM
7	36	4	3	2/23/2006 12:00:00 AM
8	37	8	6	3/6/2006 12:00:00 AM
9	38	9	28	3/10/2006 12:00:00 AM
10	39	3	8	3/22/2006 12:00:00 AM
11	40	4	10	3/24/2006 12:00:00 AM
12	41	1	7	3/24/2006 12:00:00 AM
13	42	1	10	3/24/2006 12:00:00 AM
14	43	1	11	3/24/2006 12:00:00 AM
15	44	1	1	3/24/2006 12:00:00 AM
16	45	1	28	4/7/2006 12:00:00 AM
17	46	7	9	4/5/2006 12:00:00 AM

- On concater les deux tables dans une table ***'Orders_merge'*** et après on utilise **Merge** pour appliquer une Jointure entre ***Orders_merge*** et ***Dim_Employee*** par l'attribut EmployeeID de ***Orders_merge*** et id_employee_prod dans ***Dim_Employee*** comme une clé étrangère:

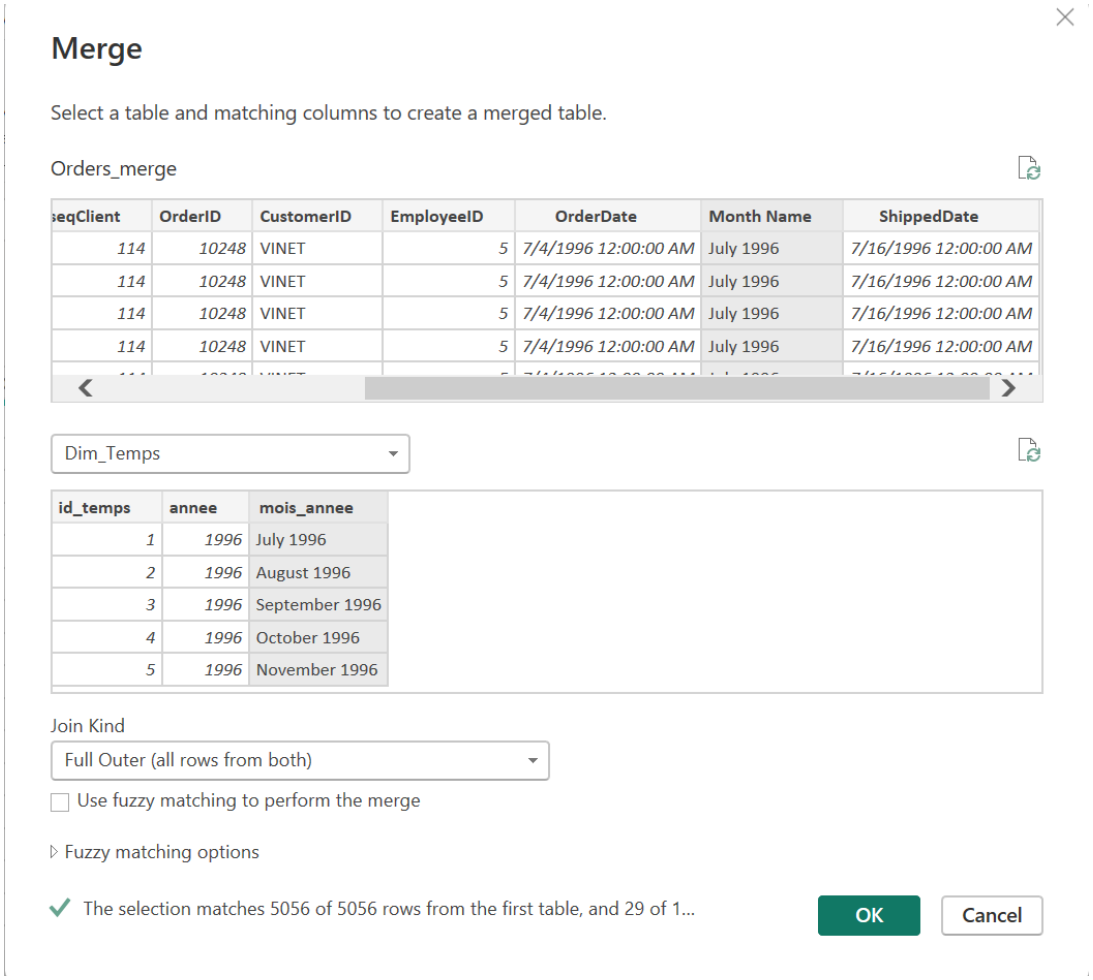


- On utilise une autre **Merge** sur le table résultat (***Orders_merge*** + ***Dim_Employe***) avec ***Dim_Client*** par l'attribut CustomerID de ***Orders_merge*** et id_client_prod dans ***Dim_Client*** comme une clé étrangère :


Merge
Select a table and matching columns to create a merged table.
Orders_merge
OrderID | CustomerID | EmployeeID | OrderDate | ShippedDate | Dim_Employee.id_seqEmploye
10248 | VINET | 5 | 7/4/1996 12:00:00 AM | 7/16/1996 12:00:00 AM
10248 | VINET | 5 | 7/4/1996 12:00:00 AM | 7/16/1996 12:00:00 AM
10248 | VINET | 5 | 7/4/1996 12:00:00 AM | 7/16/1996 12:00:00 AM
10248 | VINET | 5 | 7/4/1996 12:00:00 AM | 7/16/1996 12:00:00 AM
Dim_Client
id_seqClient | id_client_prod | source_prod | CompanyName | City | Country
1 | 1 | Access | Company A | Seattle | USA
2 | 2 | Access | Company B | Boston | USA
3 | 3 | Access | Company C | Los Angelas | USA
4 | 4 | Access | Company D | New York | USA
5 | 5 | Access | Company E | Minneapolis | USA
Join Kind
Left Outer (all from first, matching from second)
Use fuzzy matching to perform the merge
Fuzzy matching options
The selection matches 4790 of 5056 rows from the first table.
OK
Cancel


= Table.ReorderColumns(#"Inserted Month Name",{"Dim_Employee.id_seqEmployee",
"Dim_Client.id_seqClient", "OrderID", "CustomerID", "EmployeeID", "OrderDate", "Month Name",
"ShippedDate"})
CustomerID | EmployeeID | OrderDate | Month Name | ShippedDate
1 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
2 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
3 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
4 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
5 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
6 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
7 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
8 VINET 5 7/4/1996 12:00:00 AM July 1996 7/16/1996 12:00:00 A
9 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
10 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
11 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
12 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
13 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
14 VINET 6 8/6/1996 12:00:00 AM August 1996 8/16/1996 12:00:00 A
15 CHOPS 5 7/11/1996 12:00:00 AM July 1996 7/23/1996 12:00:00 A
16 CHOPS 5 7/11/1996 12:00:00 AM July 1996 7/23/1996 12:00:00 A
17 CHOPS 5 7/11/1996 12:00:00 AM July 1996 7/23/1996 12:00:00 A

- Et maintenant on fait une Jointure entre la table *Orders_merge* et *Dim_Temps* utilisant **Merge** par l'attribut **Month name** de *Orders_merge* et mois_annee dans *Dim_Temps* comme une clé étrangère, **et utilisant une jointure Full Join pour avoir tous les dates de notre calendrier même s'il n'existe aucun produit livré dans cette date :**



- On ajoute une nouvelle colonne *'Shipped'* qui a la valeur 1 si *'Shipped_Date'* n'est pas NULL et 0 sinon :

```
= Table.AddColumn(#"Removed Duplicates", "Shipped", each if [ShippedDate] = null then 0 else 1)
```

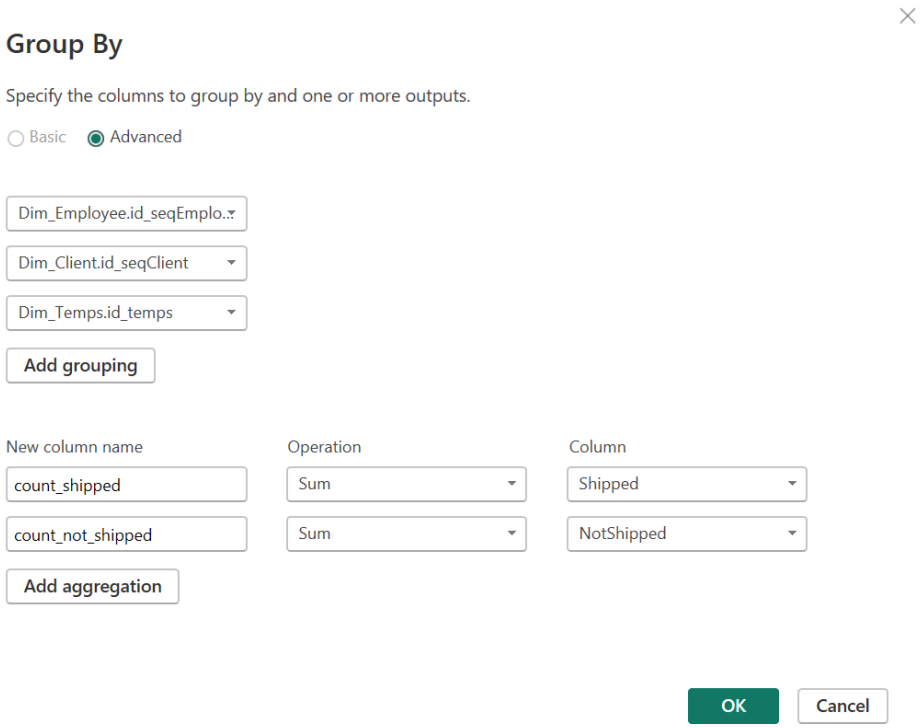
	m_Temps.mois_annee	OrderDate	Month Name	ShippedDate	Shipped
1	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
2	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
3	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
4	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
5	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
6	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
7	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
8	996	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1
9	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
10	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
11	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
12	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
13	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
14	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
15	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
16	996	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1
17	996	7/31/1996 12:00:00 AM	July 1996	8/9/1996 12:00:00 AM	1

- On ajoute une nouvelle colonne *'not_Shipped'* qui a la valeur 0 si *'Shipped_Date'* n'est pas NULL et 1 sinon :

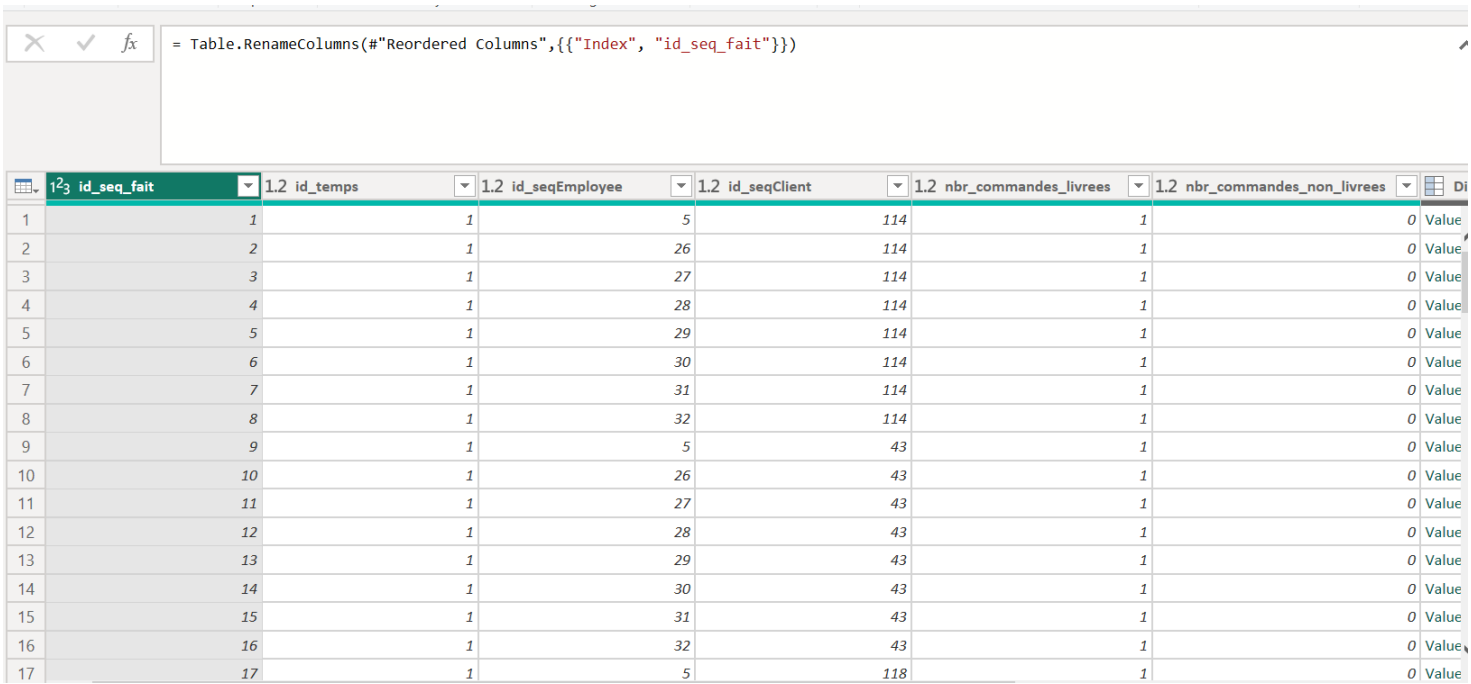
```
= Table.AddColumn(#"Added Custom1", "NotShipped", each if [ShippedDate] = null then 1 else 0)
```

	OrderDate	Month Name	ShippedDate	Shipped	NotShipped
1	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
2	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
3	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
4	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
5	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
6	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
7	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
8	7/4/1996 12:00:00 AM	July 1996	7/16/1996 12:00:00 AM	1	0
9	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
10	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
11	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
12	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
13	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
14	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
15	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
16	7/11/1996 12:00:00 AM	July 1996	7/23/1996 12:00:00 AM	1	0
17	7/31/1996 12:00:00 AM	July 1996	8/9/1996 12:00:00 AM	1	0

- Utilisant l'opération **Group By** on calcule la somme des commandes livrées et non livrées par calculant la somme de *'Shipped'* et *'notShipped'* pour chaque (*id_seq_employee, id_seq_client, id_temps*) unique :

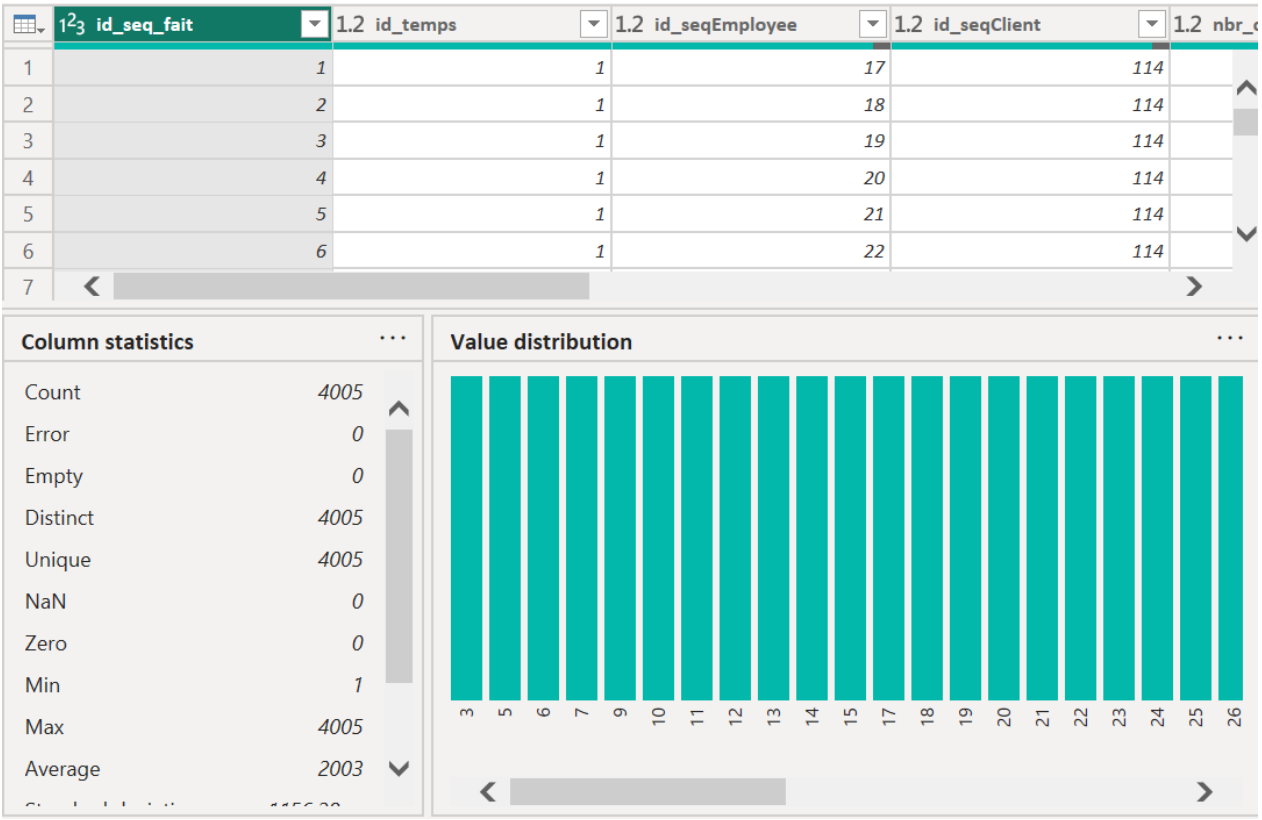


- Et on obtient le résultat après le changement des noms des colonnes et suppression des colonnes non nécessaires:



	id_seq_fait	id_temps	id_seqEmployee	id_seqClient	nbr_commandes_livrees	nbr_commandes_non_livrees	Di
1	1	1	5	114	1	0	Value
2	2	1	26	114	1	0	Value
3	3	1	27	114	1	0	Value
4	4	1	28	114	1	0	Value
5	5	1	29	114	1	0	Value
6	6	1	30	114	1	0	Value
7	7	1	31	114	1	0	Value
8	8	1	32	114	1	0	Value
9	9	1	5	43	1	0	Value
10	10	1	26	43	1	0	Value
11	11	1	27	43	1	0	Value
12	12	1	28	43	1	0	Value
13	13	1	29	43	1	0	Value
14	14	1	30	43	1	0	Value
15	15	1	31	43	1	0	Value
16	16	1	32	43	1	0	Value
17	17	1	5	118	1	0	Value

- Et le nombre des lignes de la table de fait est 4005 :



	id_seq_fait	id_temps	id_seqEmployee	id_seqClient	nbr_c
1	1	1	17	114	
2	2	1	18	114	
3	3	1	19	114	
4	4	1	20	114	
5	5	1	21	114	
6	6	1	22	114	
7					

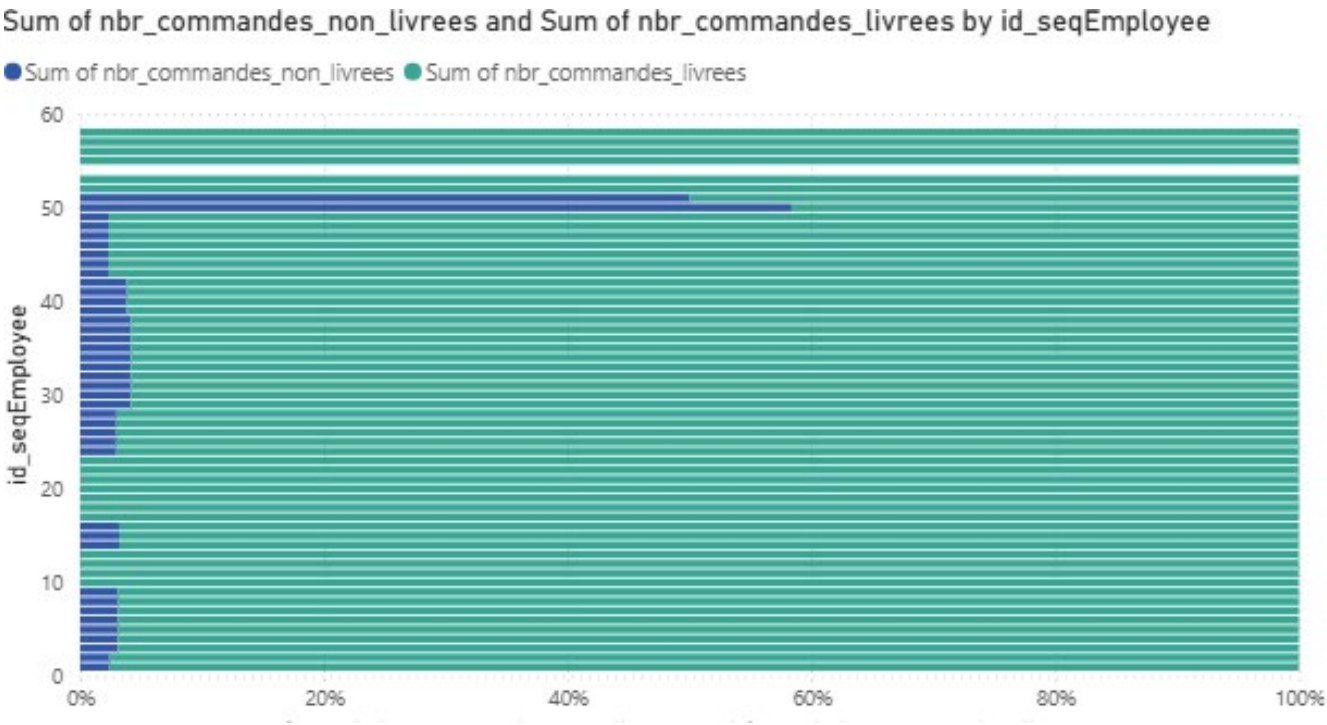
2- Visualisation des données :

2.1. Table de bords sur PowerBI :

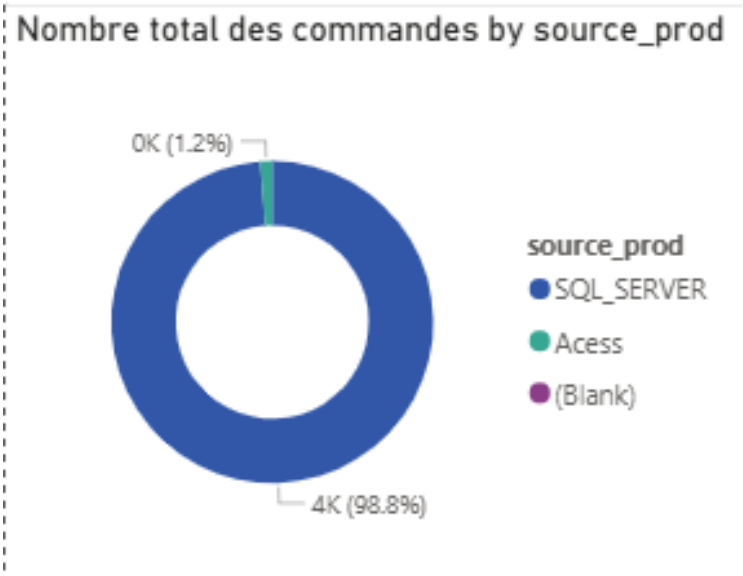
Afin d'obtenir un tableau de bord plus clair, plus professionnel et mieux structuré, on a divisé le tableau de bord en trois pages correspondant aux trois dimensions utilisées :

2.1.1. Table de bords pour la dimension Employé :

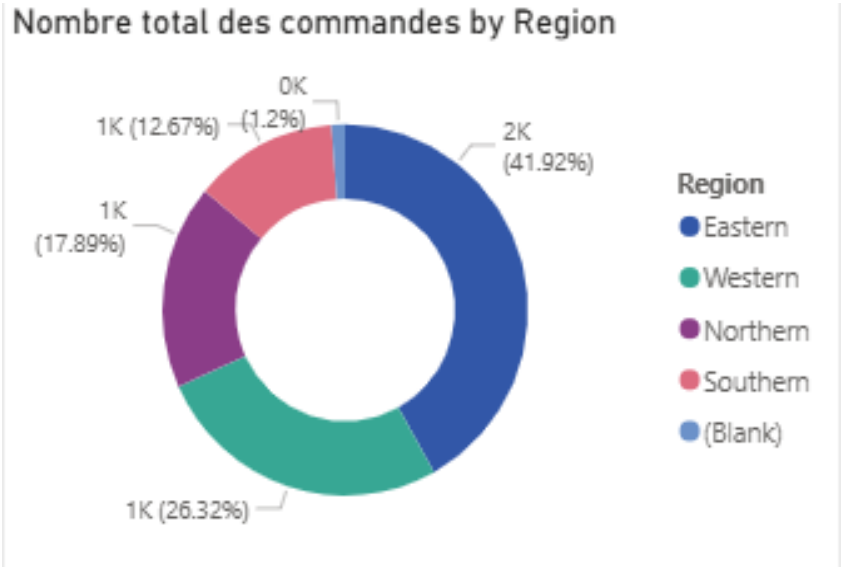
1- Un Diagramme en barres empilées à 100 % pour le nombre des commandes livrées et non livrées pour chaque employé, ce diagramme permettre de visualiser le pourcentage et le nombre des commandes livrées et non livrées pour chaque employé id dans la dimension :



2- Un diagramme en anneau pour le nombre total des commandes par la source des données pour visualiser la source la plus courante et la plus efficace :

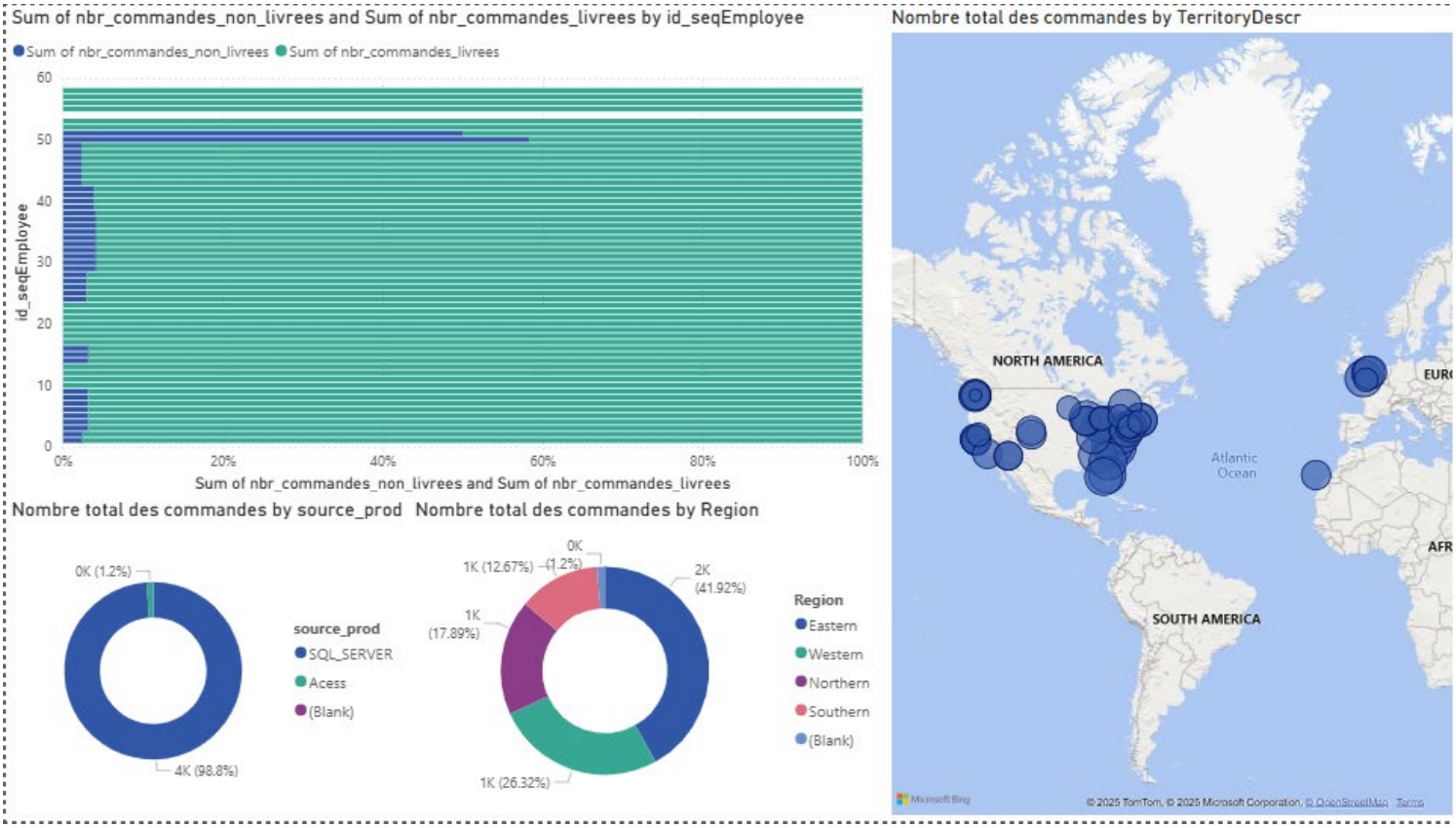


3- Un diagramme en anneau pour le nombre total des commandes par la région des employées, les régions sont identifié dans les données de la base de données SQL Server (Northern, Southern, Eastern, Western) mais pour les données de la source Access c'est NULL et donc (Blank) dans le diagramme :

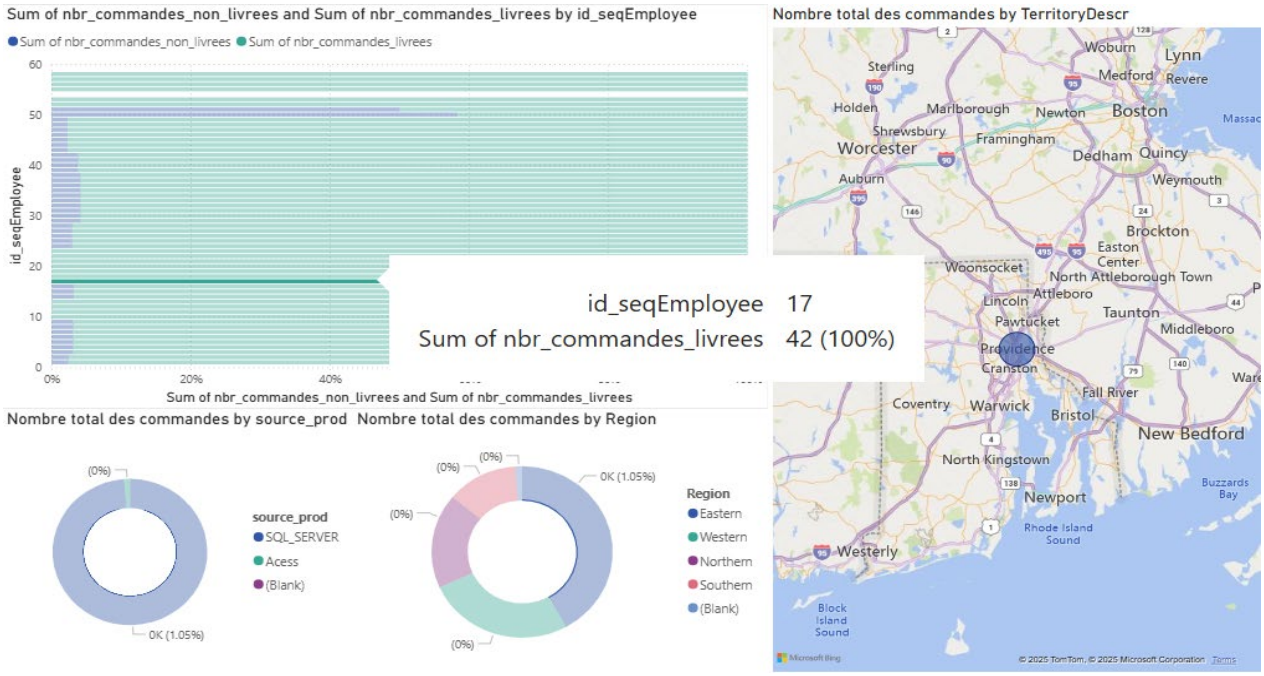


4- Une carte géographique à bulles présentant les villes (TerritoryDescreption) des employés en fonction du nombre total de commandes par les employés de chaque ville. Cette visualisation permet d'identifier les villes les plus performantes :

- Et alors la page :

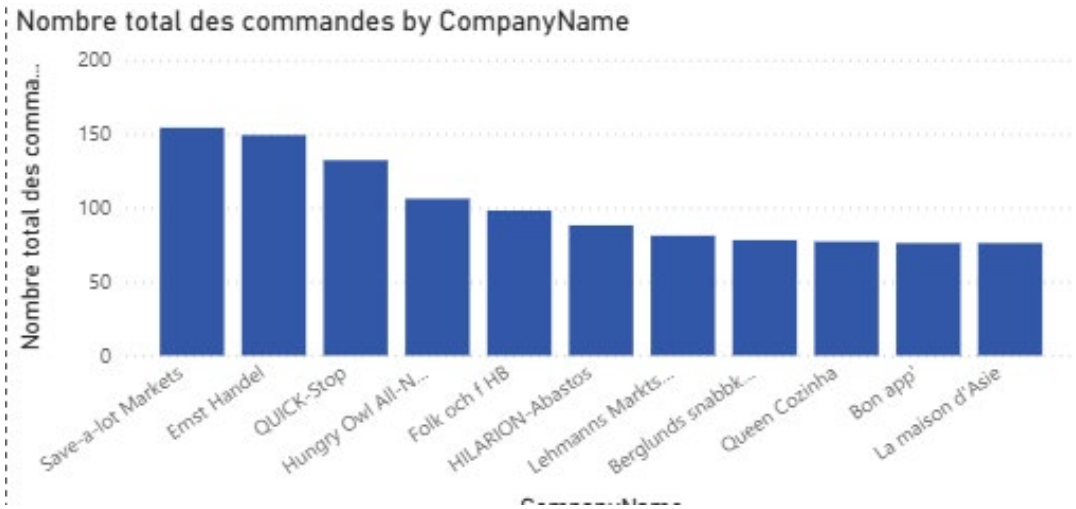


- Quand on sélectionne un employé sur l'un des graphiques entraîne automatiquement la mise à jour des autres visualisations avec les données correspondantes, exemple :

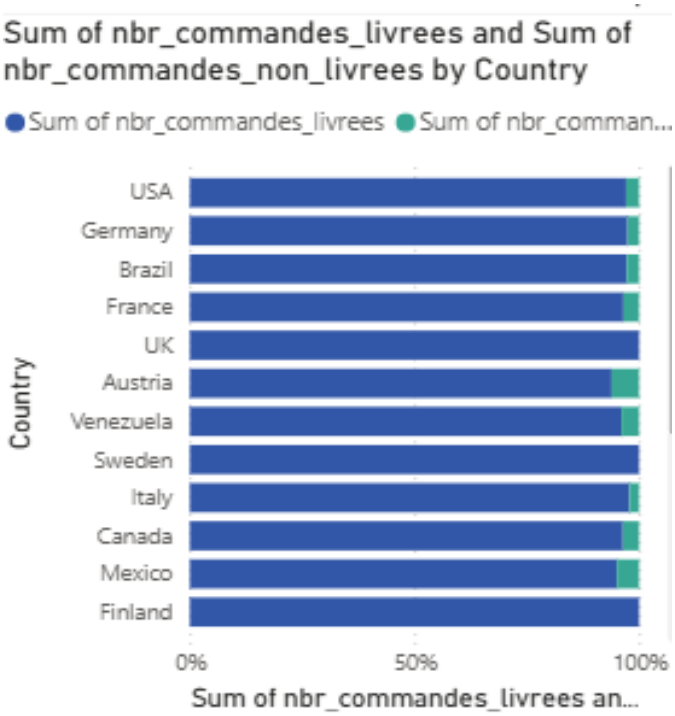


2.1.2. Table de bords pour la dimension Client :

1- Un diagramme en barre pour les top 10 client d'après les nombres des commandes (nom de compagnie par nombre des commandes) :

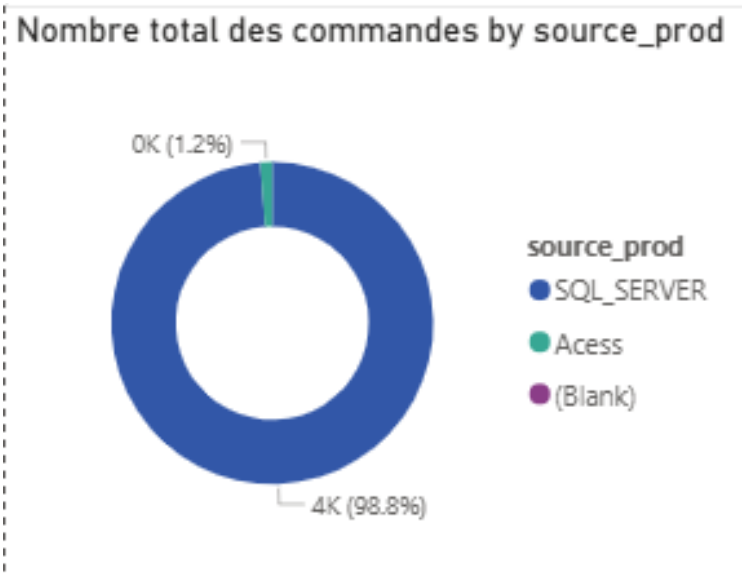


2- Un Diagramme en barres empilées à 100 % pour le nombre des commandes livrées et non livrées pour chaque pays des clients, Ce diagramme permet de repérer les pays où le volume de commandes livrées est le plus élevé et d'identifier les zones où des améliorations sont nécessaires pour réduire les commandes non livrées :

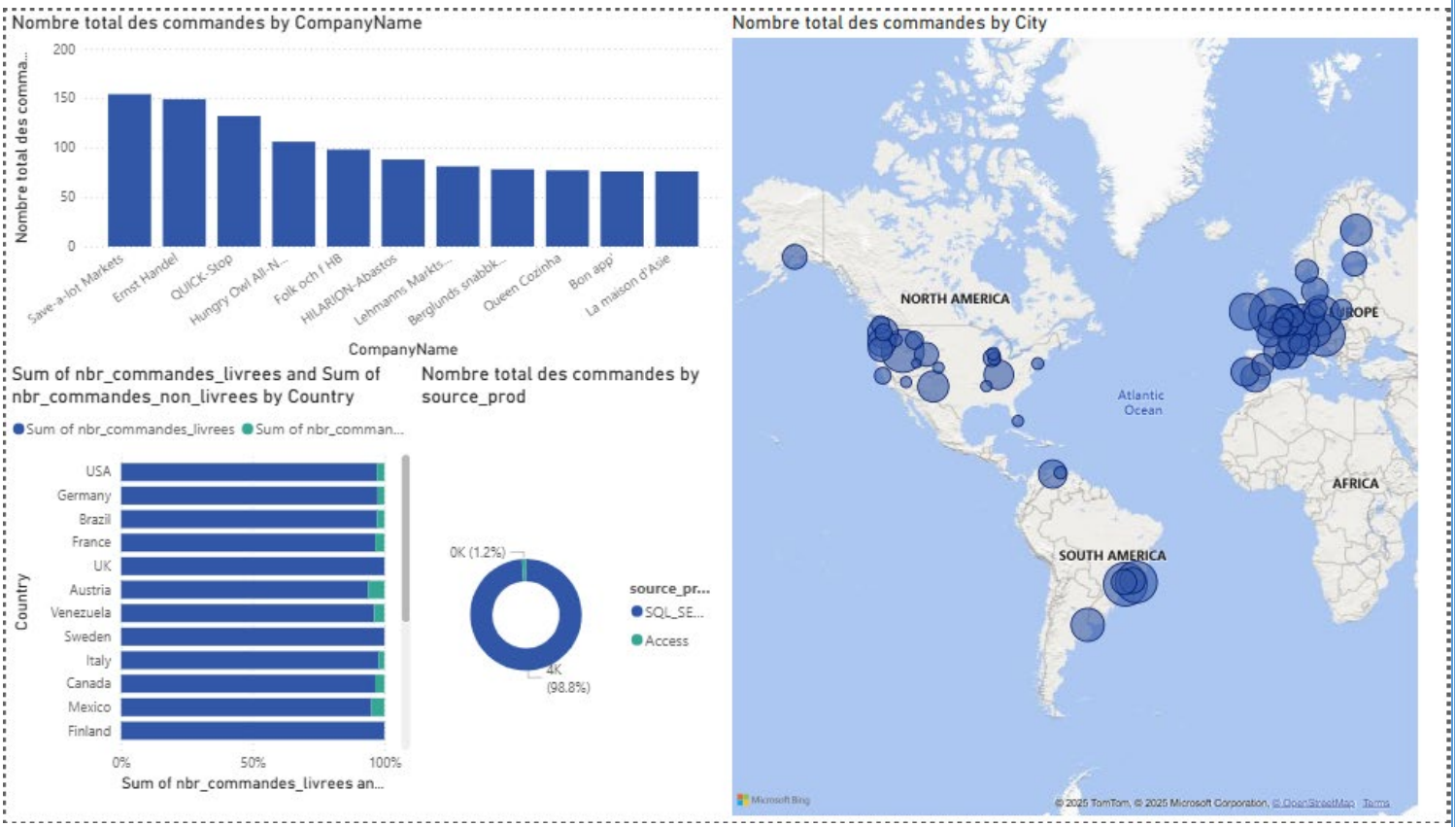


3- Une carte géographique à bulles présentant les villes des clients en fonction du nombre total de commandes de chaque ville. Cette visualisation permet d'identifier les villes les plus élevé :

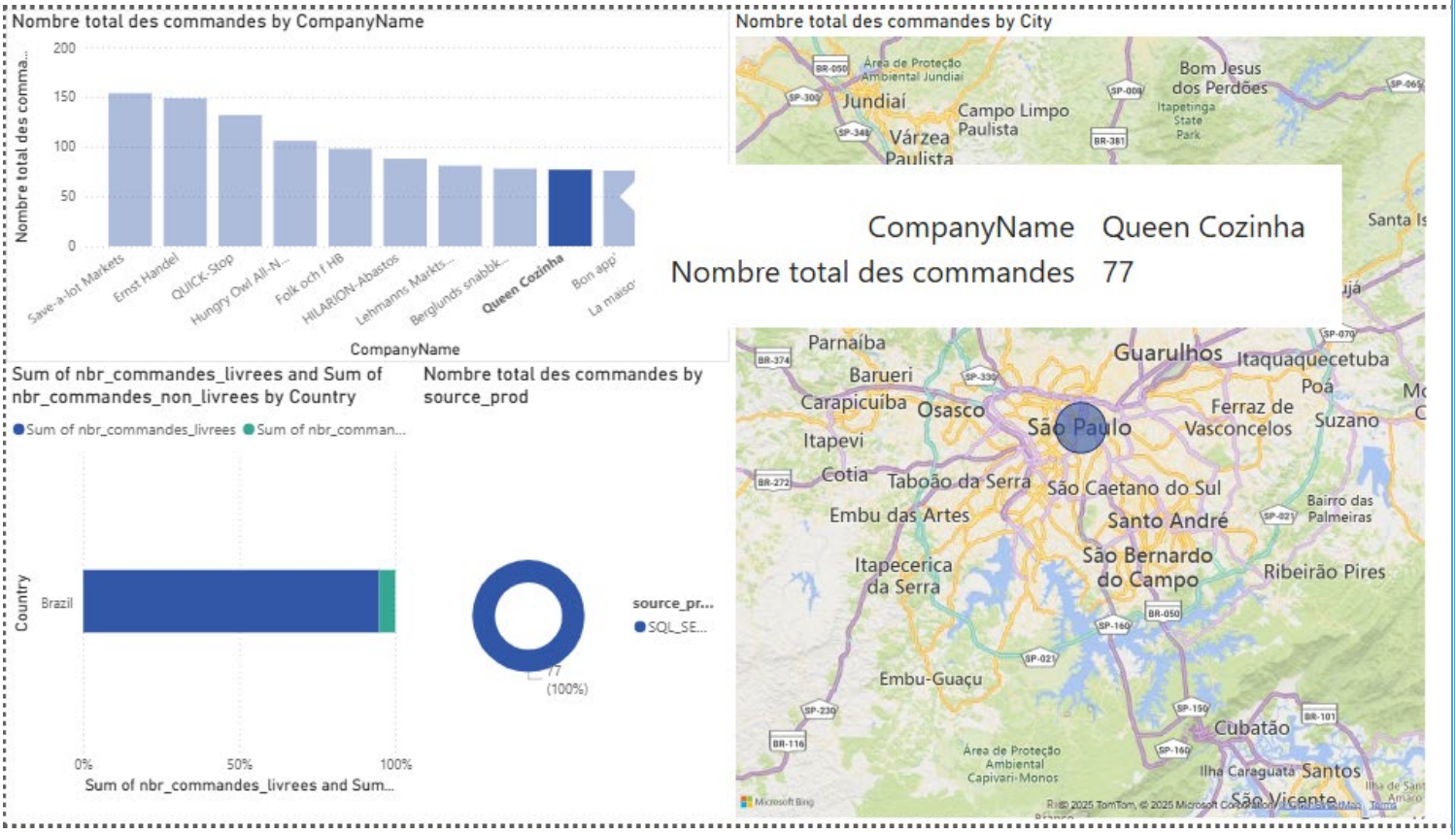
4- Un diagramme en anneau pour le nombre total des commandes par la source des données pour visualiser la source la plus courante et la plus efficace :



- Et alors la page complète :

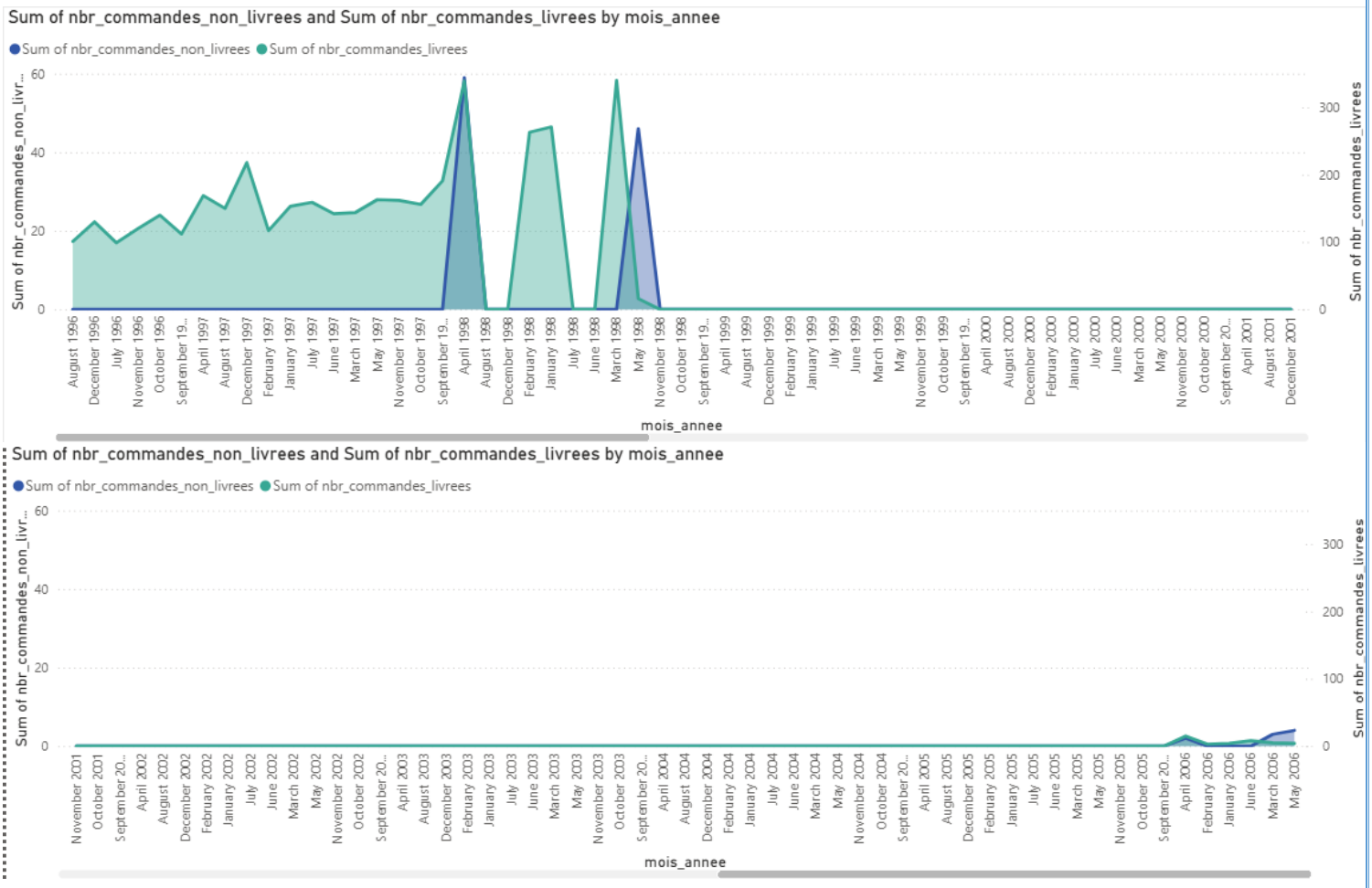


- Quand on sélectionne un élément sur l'un des graphiques entraîne automatiquement la mise à jour des autres visualisations avec les données correspondantes, exemple :

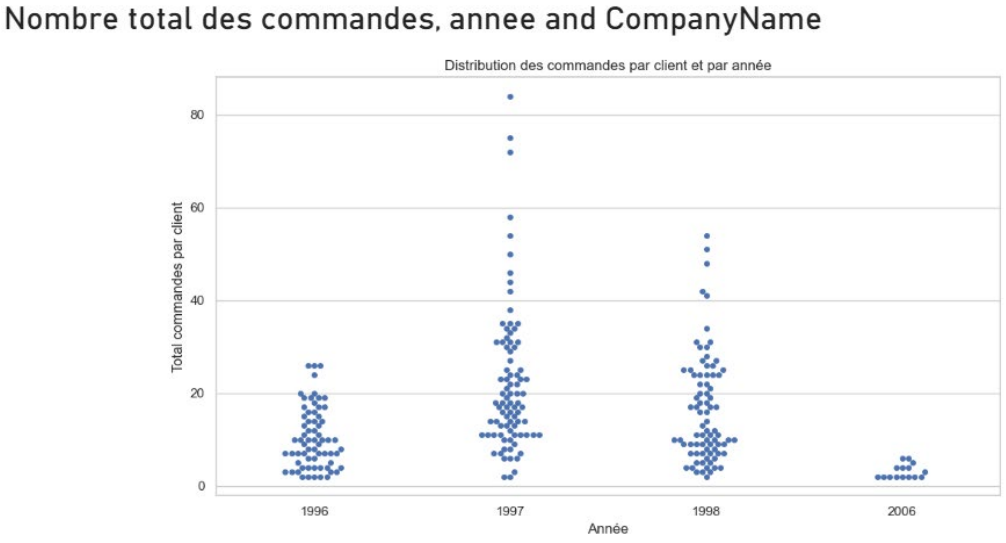


2.1.3. Table de bords pour la dimension Temps :

1- Un diagramme en aires pour les nombres des commandes livrées et non livrées pour chaque mois, Ce graphique permet de visualiser les périodes de pic et de faiblesse, d'identifier les phases de croissance rapide ou de déclin, et d'analyser l'évolution des indicateurs clés dans le temps :



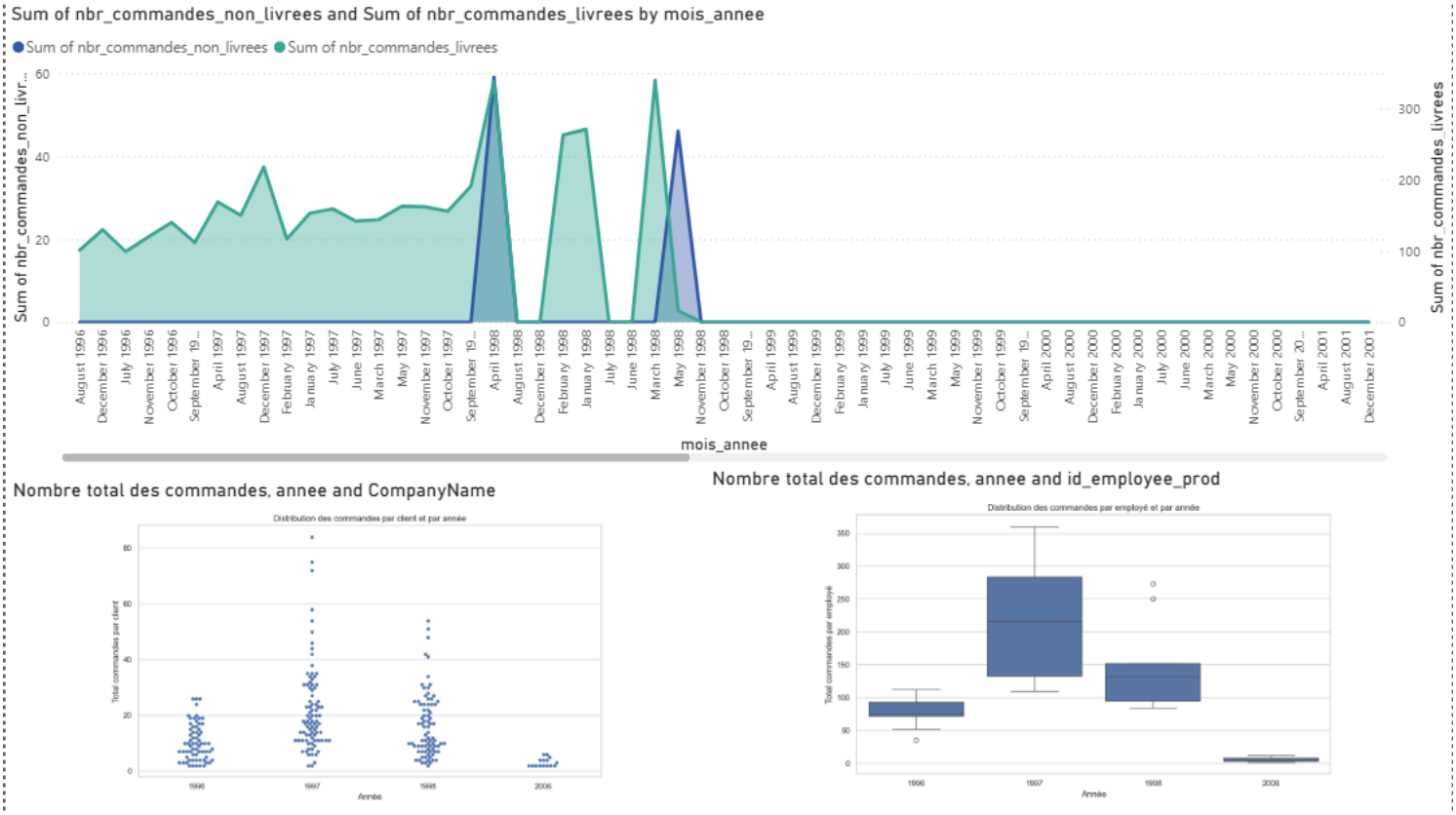
2- Un nuage de points répartis pour le nombre total des commandes par les clients pour chaque année qui permettre de visualise les valeurs extrêmes et les tendances générales chaque année (**créé par un script python**) :



3- Un diagramme en boîte permet de visualiser la distribution des commandes par employé pour chaque année, de repérer les valeurs extrêmes et d'analyser les variations de performance entre employés :

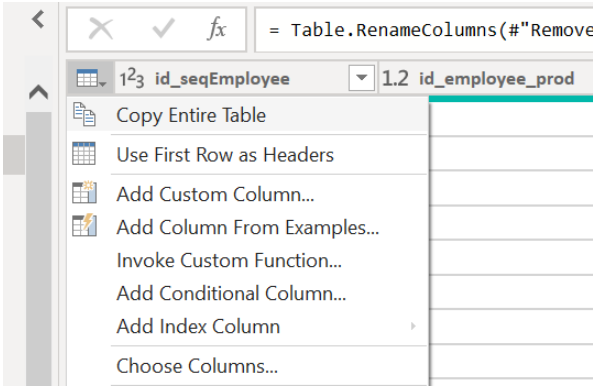
Nombre total des commandes, annee and id_employee_prod

- Et alors la page complète :



2.2. Représentations graphiques utilisant des bibliothèques python :

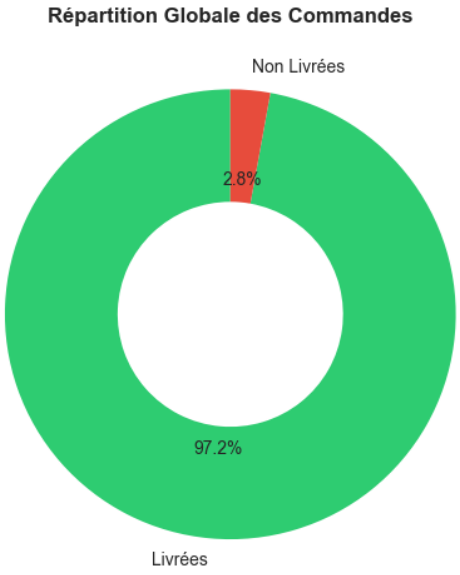
Avant de commencer l'analyse avec Python, les tables finales des dimensions et la table de fait de l'entrepôt de données ont été exportées en fichiers Excel. Ce format s'intègre parfaitement avec Python via Pandas (`pd.read_excel()`). Les données sont juste copiées de tables PowerBI à les fichiers Excel par l'option **'Copy Entire Table'** :



2.2.1. Analyse sur la dimension Employé :

- *Répartition Globale des Commandes (Donut Chart)*

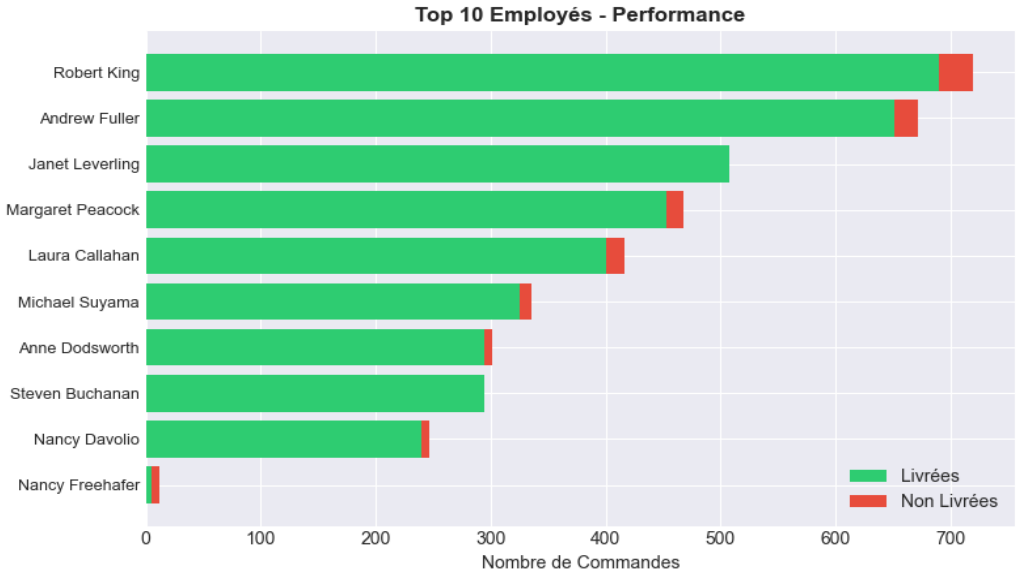
Ce graphique en anneau présente la proportion globale entre les commandes livrées et non livrées. Il permet d'avoir une vue d'ensemble immédiate du taux de réussite des livraisons dans l'ensemble de l'entreprise. Les couleurs vert et rouge facilitent l'identification rapide des performances.



- *Top 10 Employés - Performance (Barres Horizontales)*

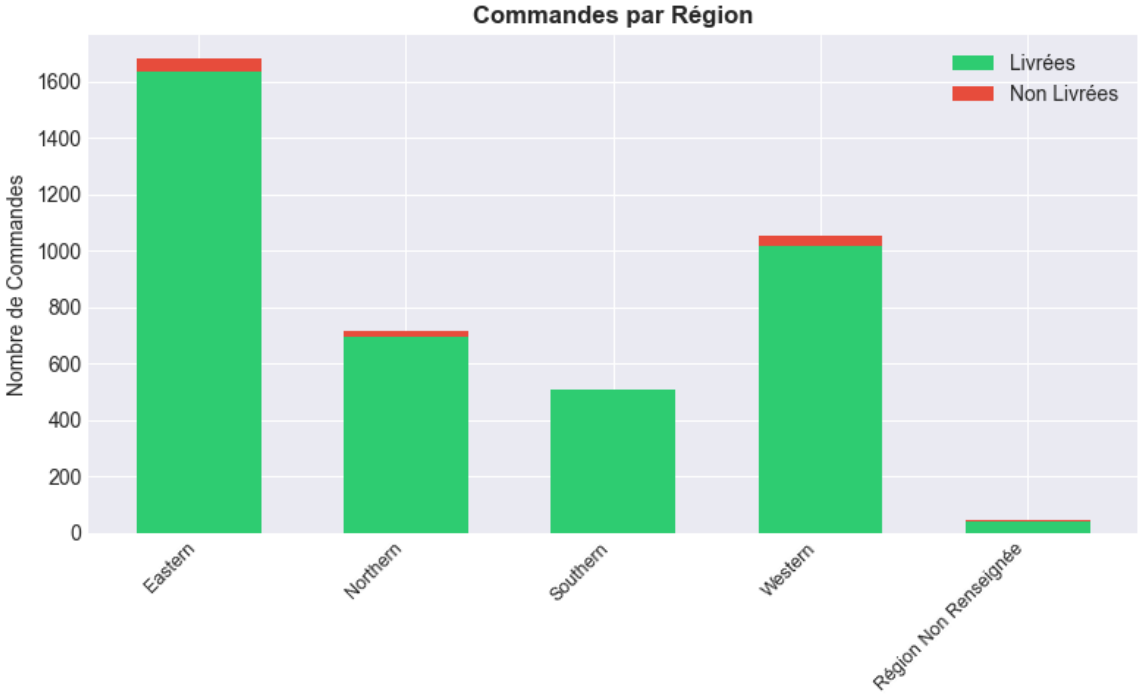
Ce graphique classe les 10 meilleurs employés selon leur volume total de commandes traitées. Les barres empilées montrent la répartition entre commandes livrées (vert) et non livrées (rouge) pour chaque

employé. Cela permet d'identifier non seulement les employés les plus productifs, mais aussi ceux qui maintiennent le meilleur taux de livraison.



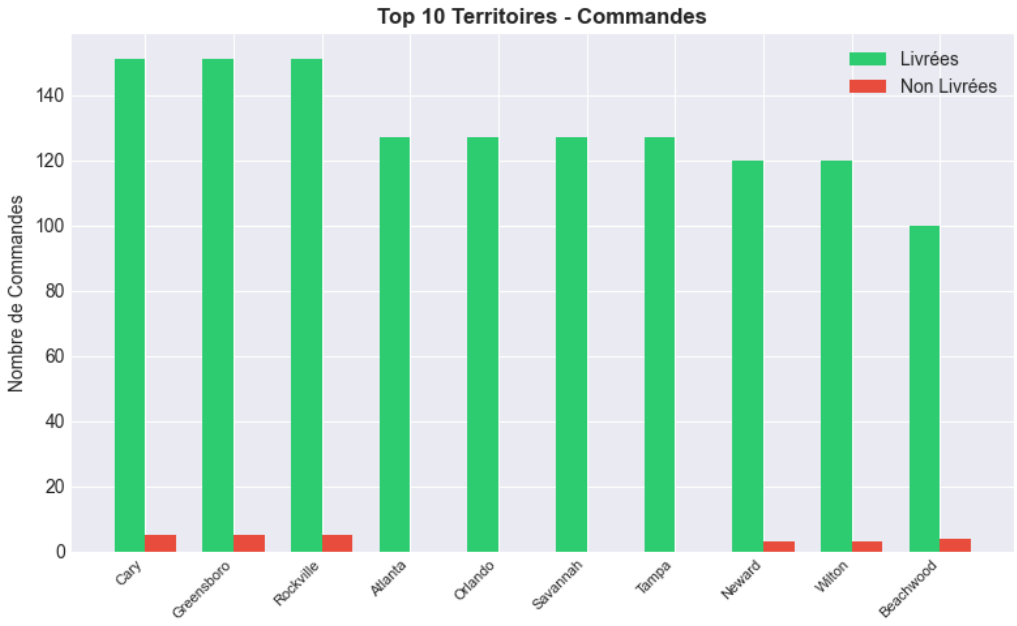
- *Commandes par Région (Barres Empilées)*

Cette visualisation présente le volume de commandes par région géographique. Les barres empilées permettent de comparer simultanément le volume total d'activité de chaque région et leur efficacité de livraison. On peut ainsi identifier les régions les plus actives et celles nécessitant des améliorations (les régions NULL sont de données de source Access)



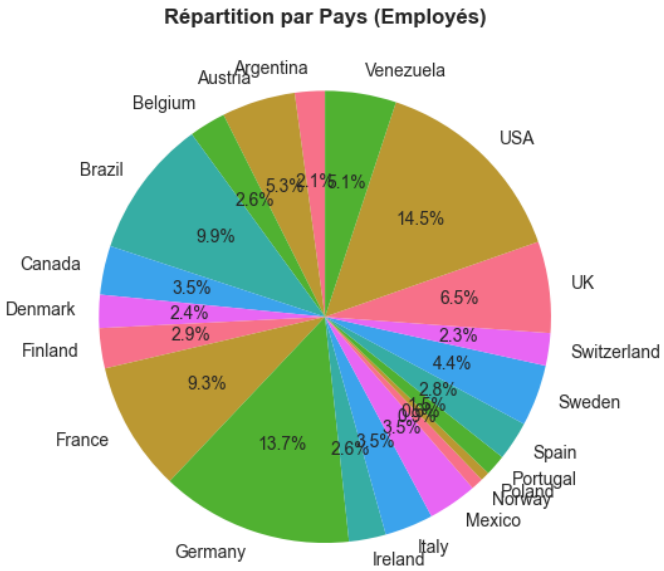
- *Top 10 Territoires - Commandes (Barres Groupées)*

Ce graphique compare les performances des 10 territoires les plus actifs. Les barres groupées (vert pour livrées, rouge pour non livrées) facilitent la comparaison directe entre les deux catégories pour chaque territoire. Cela aide à identifier les territoires performants et ceux nécessitant un support additionnel.



- *Répartition par Pays (Employés) (Camembert)*

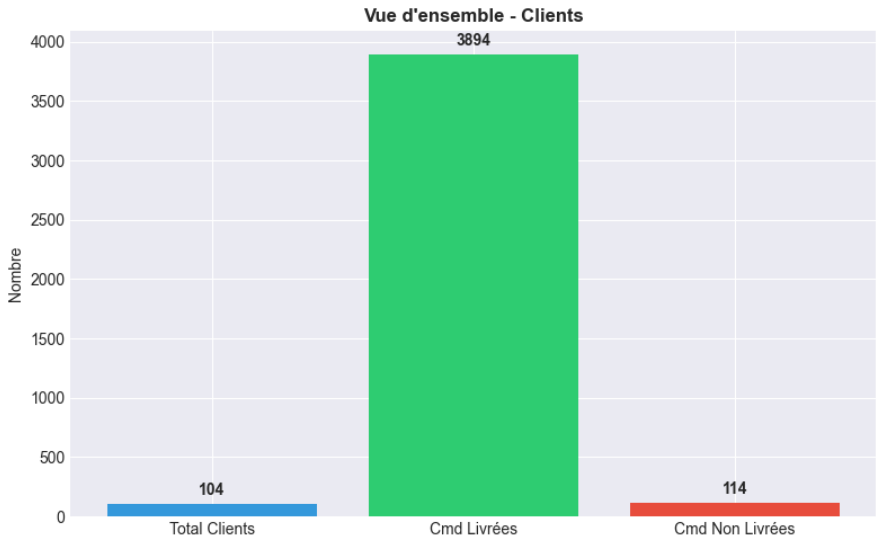
Ce diagramme circulaire illustre la distribution des commandes selon les pays où travaillent les employés. Il permet de visualiser rapidement la concentration géographique de l'activité et d'identifier les marchés principaux de l'entreprise.



2.2.2. Analyse sur la dimension Client :

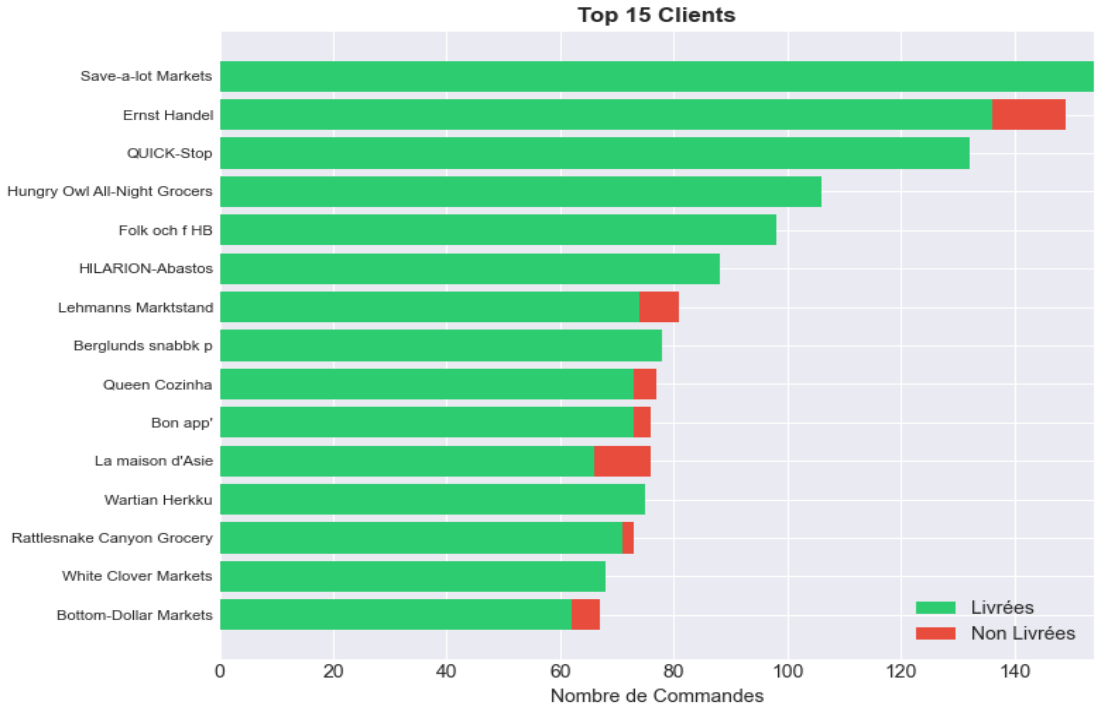
- *Vue d'ensemble - Clients (Barres Simples)*

Ce graphique présente trois indicateurs clés : le nombre total de clients actifs, le total des commandes livrées et celui des commandes non livrées. Cette vue synthétique permet d'évaluer rapidement la base client et le volume d'activité global.



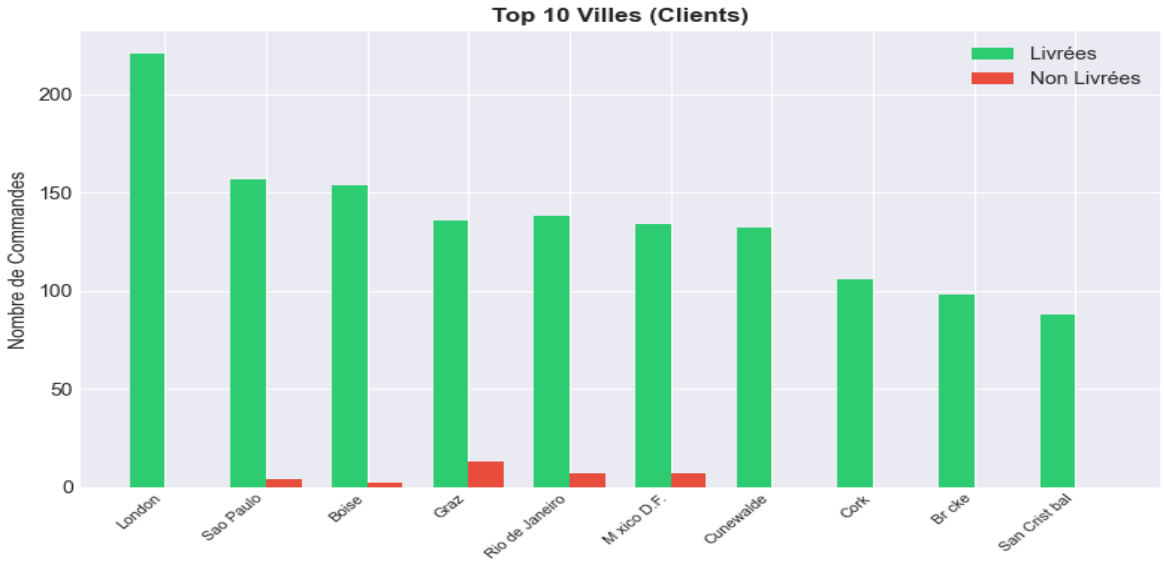
- *Top 15 Clients (Barres Horizontales Empilées)*

Cette visualisation identifie les 15 clients générant le plus de commandes. Les barres empilées révèlent non seulement le volume d'affaires de chaque client, mais aussi leur profil de livraison. C'est un outil précieux pour la gestion des comptes clés et l'identification des clients stratégiques.



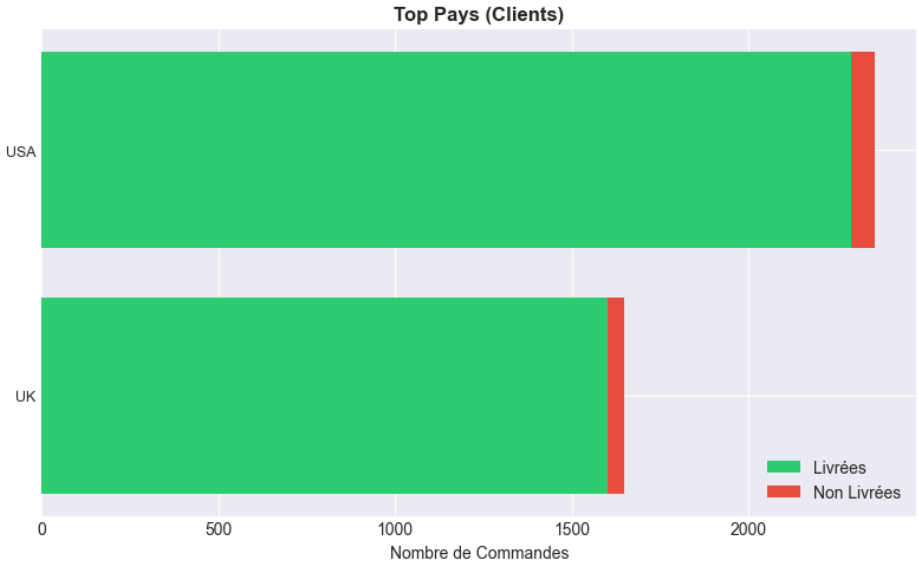
- *Top 10 Villes (Clients) (Barres Groupées)*

Ce graphique compare les performances des 10 villes ayant le plus de clients. La représentation en barres groupées permet d'analyser facilement le volume d'activité et le taux de réussite de livraison pour chaque ville, aidant à optimiser la logistique régionale.



- *Top Pays (Clients) (Barres Horizontales Empilées)*

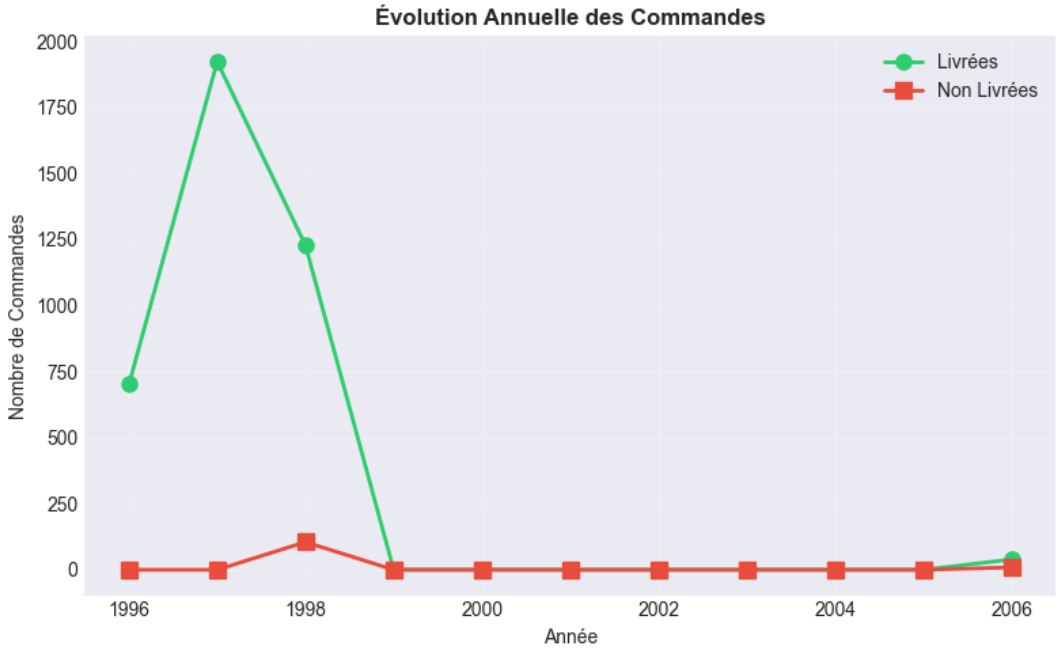
Cette analyse présente les pays générant le plus de commandes du côté des clients. Les barres empilées montrent la répartition livrées/non livrées pour chaque pays, permettant d'identifier les marchés les plus prometteurs et ceux nécessitant des améliorations logistiques.



2.2.3. Analyse sur la dimension Temps :

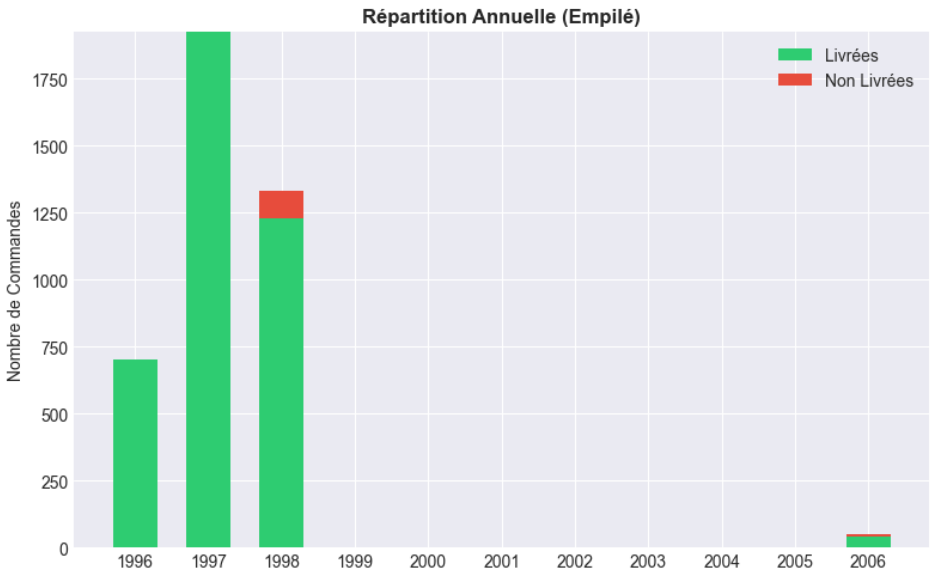
- *Évolution Annuelle des Commandes (Courbes)*

Ce graphique linéaire trace l'évolution des commandes livrées et non livrées sur plusieurs années. Les courbes avec marqueurs facilitent l'identification des tendances à long terme, des périodes de croissance ou de déclin, et permettent d'anticiper les besoins futurs.



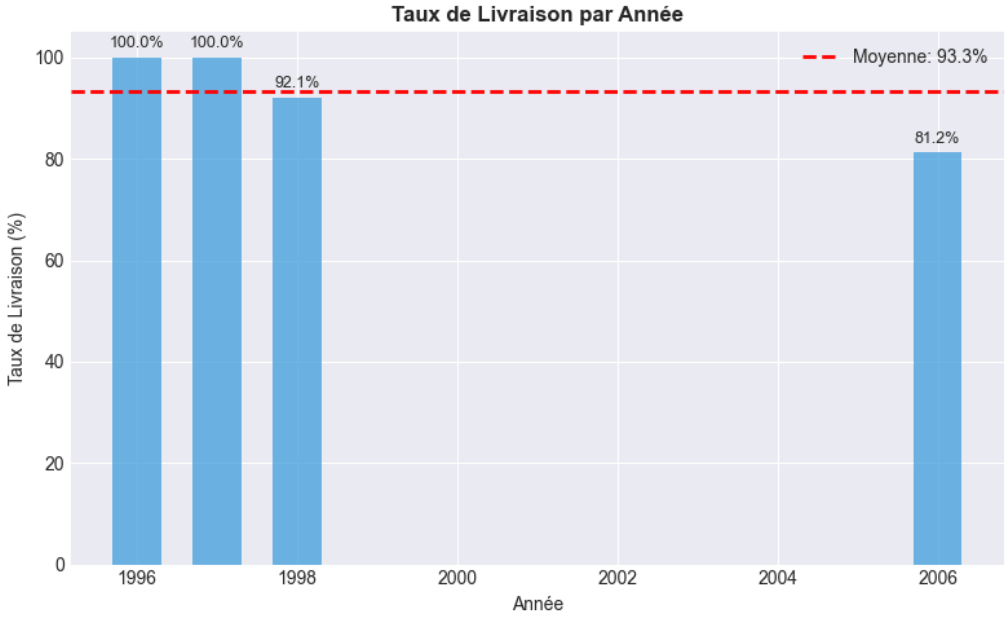
- *Répartition Annuelle (Empilé) (Barres Empilées)*

Cette visualisation présente le volume total de commandes par année sous forme de barres empilées. Elle permet de voir d'un coup d'œil la croissance globale de l'activité tout en maintenant la visibilité sur la proportion de commandes non livrées.



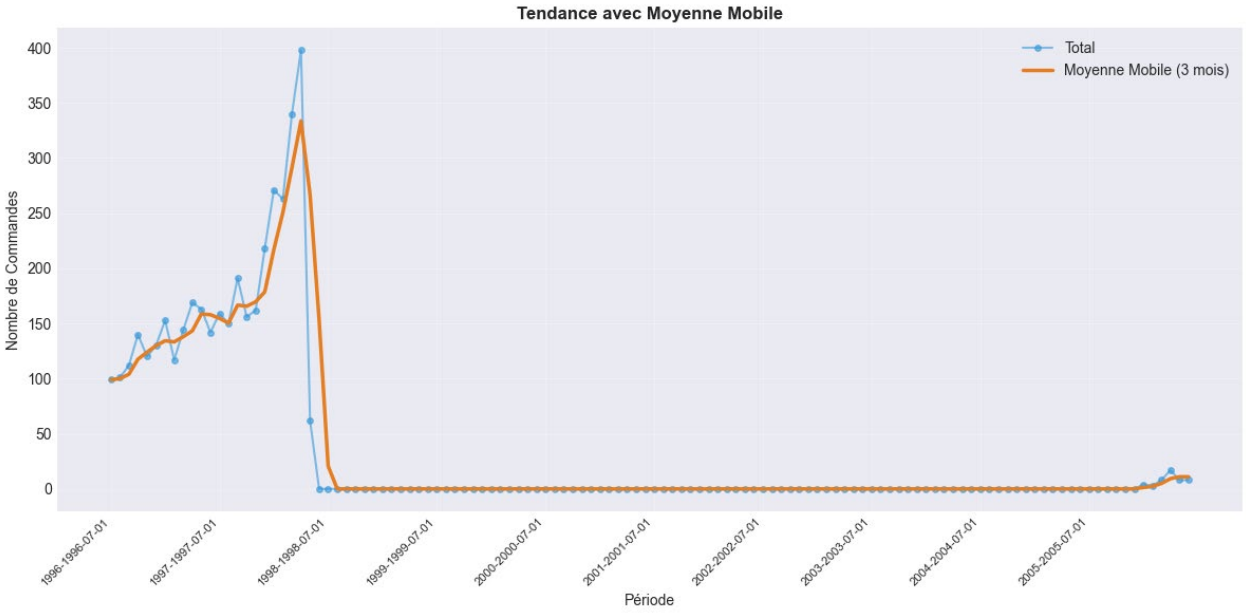
- *Taux de Livraison par Année (Barres avec Moyenne)*

Ce graphique critique affiche le pourcentage de commandes livrées avec succès pour chaque année. La ligne horizontale représente la moyenne globale, permettant d'identifier rapidement les années de sous-performance ou de surperformance. Les étiquettes de pourcentage facilitent la lecture précise.



- *Tendance avec Moyenne Mobile (Courbes)*

Cette analyse avancée combine les données mensuelles brutes avec une moyenne mobile sur 3 mois. La courbe lissée (orange) filtre les variations à court terme pour révéler la tendance de fond, facilitant la prise de décision stratégique et la détection de changements structurels dans l'activité.



3- Justification des choix techniques:

Pour la phase d'extraction, transformation et chargement (ETL) ainsi que la création des tableaux de bord interactifs, Power BI a été retenu pour sa capacité native à connecter diverses sources de données, sa puissance de modélisation avec Power Query, et son interface intuitive permettant aux utilisateurs métiers de créer et personnaliser leurs propres visualisations en temps réel, et la possibilité de l'interaction avec les graphes. Power BI offre également une excellente intégration avec l'écosystème Microsoft et facilite le partage sécurisé des rapports au sein de l'organisation.

En complément, Python avec les bibliothèques Pandas et Matplotlib a été choisi pour l'analyse exploratoire approfondie et la génération de visualisations personnalisées. Pandas excelle dans la manipulation et l'agrégation de données tabulaires, offrant des performances optimales pour le traitement de datasets volumineux et des fonctionnalités avancées de groupement et de transformation. Matplotlib, quant à lui, permet une personnalisation fine des graphiques (barres, courbes, camemberts, heatmaps) et s'intègre parfaitement dans un workflow d'analyse reproductible via des scripts Python. Cette approche programmatique garantit la traçabilité des analyses, facilite l'automatisation des rapports périodiques, et permet une flexibilité maximale pour répondre à des besoins analytiques spécifiques non couverts par les outils de BI standards.

L'utilisation combinée de ces technologies rend l'information accessible et actionnable pour tous les niveaux de décision, allant de l'exploration ad-hoc par les analystes à la consultation stratégique par les dirigeants.