A. Importance de la Vue de modèle et méthode

Je ne saurais trop insister sur l'importance de la Vue de modèle. Cette étape, qui intervient juste après la requête, est le cœur du fonctionnement de Power BI.

Si vous êtes connecté à un modèle sémantique, la Vue de modèle est déjà prête.

Si vous êtes connecté à une source ne contenant qu'une seule table (une grande table dans Excel par exemple), le rôle de la Vue de modèle est moins critique, et le travail de mise en place du modèle peut, dans une certaine mesure, être ignoré. Mais ceci est sujet à caution, et n'est pas considéré comme la meilleure pratique.

La Vue de modèle a plusieurs rôles essentiels :

- vérifier les relations entre les tables, et au besoin créer, modifier ou supprimer des relations ;
- ajouter, si elle n'existe pas déjà, une table du temps (un calendrier) ;
- améliorer la lisibilité du modèle de données, et sa facilité d'utilisation.

La mise en place du modèle est une séquence de tâches clairement définies, et ne prend en définitive que peu de temps. Mais le bon accomplissement de cette séquence est déterminant pour la suite de votre travail dans Power BI.

Voici la checklist précise de ces tâches :

- vérifier les relations entre les tables, et au besoin les créer,
- disposer les tables dans la vue,
- ajouter la table calendrier (création, marquage, relation avec les autres tables),
- créer des mesures explicites,
- organiser les colonnes.

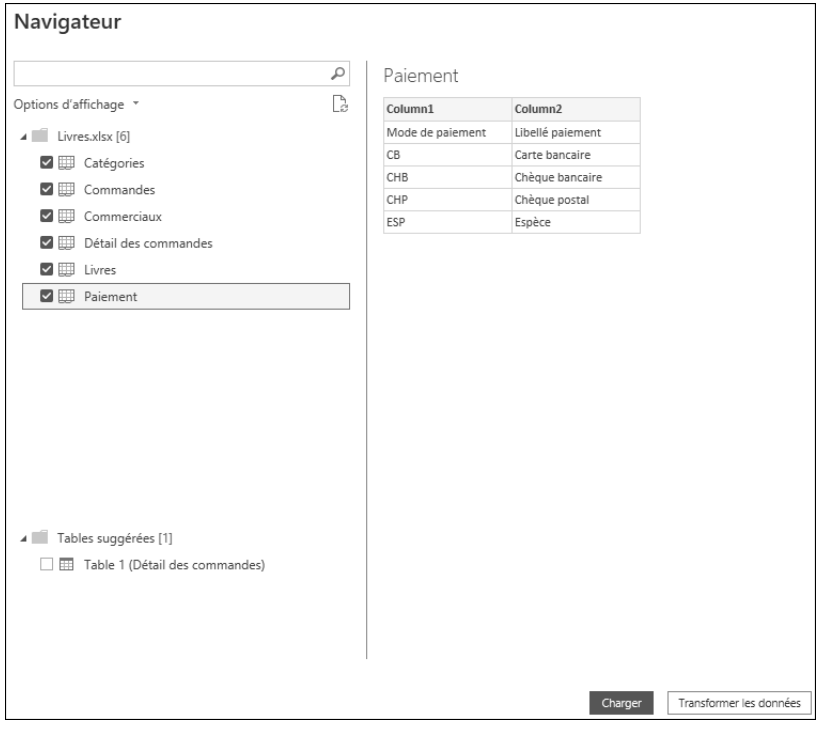
B. Mise en place de l'exemple

Nous allons créer un nouveau fichier Power BI pour illustrer tout ce chapitre. Vous trouverez le résultat de ce travail dans le fichier **Livres.pbix**, mais je vous recommande de refaire ce travail, qui reprend les points vus dans les chapitres précédents.

Les sept tables que nous allons importer, issues de deux sources différentes, représentent les ventes d'une librairie. Vous y trouverez des clients, des commandes, des commerciaux, et des livres. Bien qu'elles se présentent sous forme d'un fichier Excel, il s'agit à l'origine d'une vraie base de données, avec des codes et des identifiants.

Notez que les données relatives aux clients sont stockées dans un fichier à part (**Livres_clients.xlsx**).

- Ouvrez un nouveau fichier Power BI.
- Utilisez le connecteur Excel et sélectionnez le fichier **Livres.xlsx**.
- Sélectionnez toutes les tables (sauf la table suggérée au bas de l'écran) et chargez les données :



- Ajoutez une nouvelle source Excel (cliquez à nouveau sur le connecteur Excel) pour aller chercher cette fois-ci la table des clients (fichier **Livres_clients.xlsx**), et chargez les données.

  Vous remarquerez que les colonnes des tables **Clients**, **Catégorie** et **Paiement** ne sont pas correctement nommées : en effet, Power BI n'a pas identifié la ligne d'en-têtes.
- Cliquez sur **Transformer les données** pour ouvrir Power Query.
- Sélectionnez la table **Clients**.
- Cliquez sur le bouton **Utiliser la première ligne pour les en-têtes** de l'onglet **Accueil** - groupe **Transformer**.
- Répétez l'opération pour les deux autres tables.
- Puisque nous sommes dans Power Query, profitons-en pour :
  - Dans la table **Commandes**, passer le type des colonnes **Date de commande** et **Date de livraison**, de **Date/Heure** à **Date**. Power BI vous demande s'il faut remplacer l'actuelle modification du type : cliquez sur **Remplacer l'actuelle**.
  - Dans la table **Détail des commandes**, changer le type de la colonne **Prix de vente**, à **Nombre décimal fixe** (adapté aux montants).
  - Renommez cette table **Detail des commandes** (sans accent) : c'est une bonne pratique de supprimer les accents au début des noms de tables et de colonnes.
  - Dans la table **Livres**, changez le type de la colonne **Prix catalogue**, à **Nombre décimal fixe**.
- Cliquez sur **Fermer & appliquer**.
- Enregistrez le fichier **Livres.pbix**, et laissez le fichier ouvert.

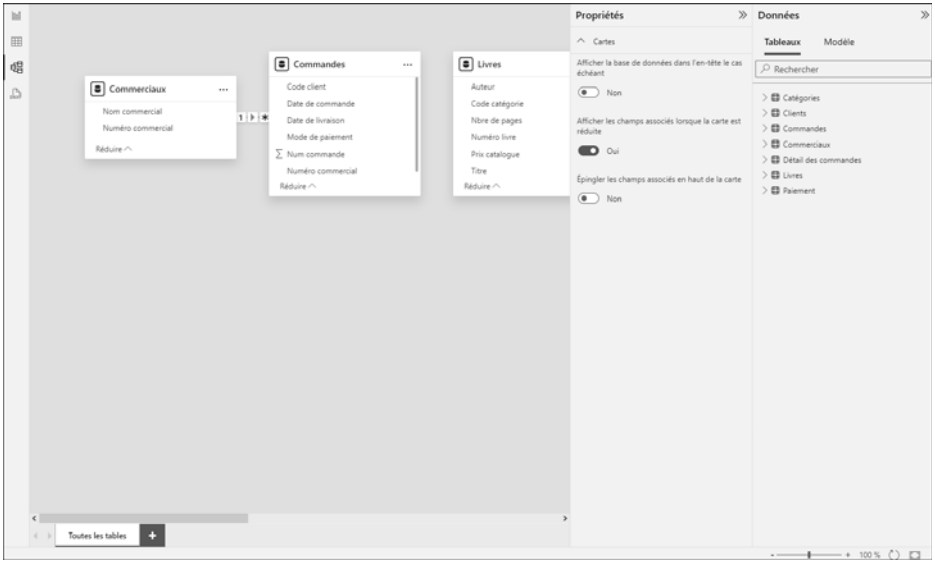
Vous voilà prêts !

Pour accéder à la Vue de modèle, cliquez sur le troisième bouton  dans la barre verticale à gauche :



Dans votre travail avec Power BI, vous passerez souvent d'une vue à une autre à l'aide de ces boutons :

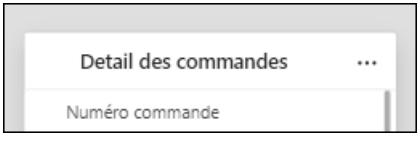
- Le premier, en haut de la barre, **Affichage du Rapport** , est la vue que vous utiliserez pour construire vos graphiques et votre tableau de bord.
- Le deuxième, **Affichage Table** , permet de voir les données brutes, directement dans les tables.
- Le troisième est la **Vue de modèle**  qui nous intéresse.
- Le quatrième, **Affichage des requêtes DAX** , est une addition récente, et offre un environnement très intéressant pour l'écriture de formules DAX. Cette vue intègre également Copilot.

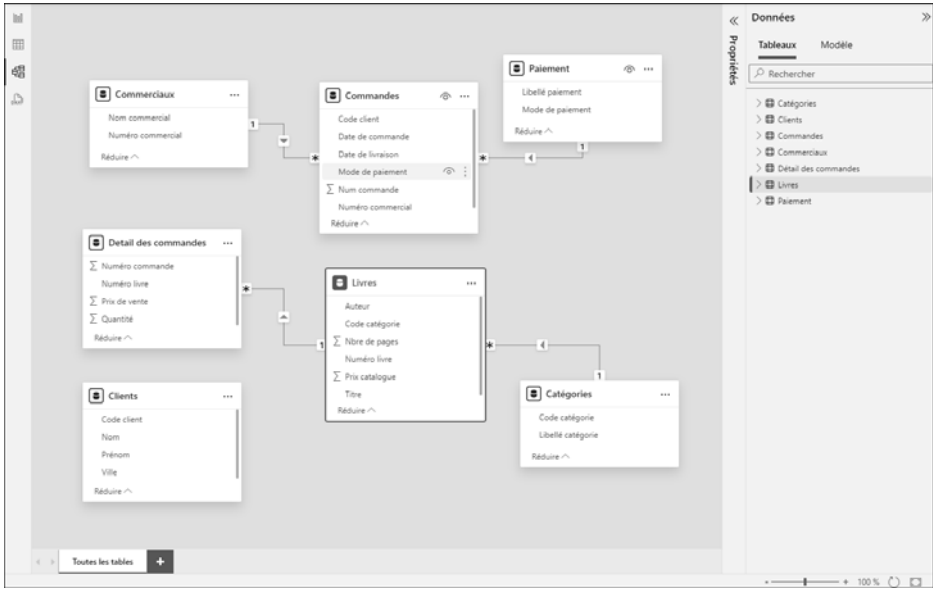


La Vue de modèle

À la droite de cette fenêtre se trouvent les volets **Propriétés** (que je vous suggère de replier à l'aide du double chevron >>, pour gagner de la place) et **Données**, où se trouvent les tables. Vous pouvez déplier les tables à l'aide du chevron, pour en afficher les colonnes.

- En bas à droite, utilisez le zoom pour voir, dans la partie centrale, l'ensemble des tables. Vous pouvez déplacer les tables dans cette vue centrale par cliquer-glisser (cliquez pour cela dans le cartouche en haut, où figure le nom de la table) :





La Vue de modèle réorganisée : certaines tables sont déjà reliées, d'autres non

Si votre vue ne ressemble pas exactement à celle-ci, ne vous inquiétez pas, c'est tout l'objet de ce chapitre.

C. Comprendre les relations

La bonne utilisation de cette fenêtre suppose de comprendre le rôle central des relations dans Power BI.

Pour le dire simplement, la relation a pour objectif de permettre la circulation de l'information entre les tables. Par exemple, un tableau avec les trois commerciaux et les quantités vendues (dans la table **Detail des commandes**), affiche en face de chaque commercial les quantités qui lui sont attribuées. Pour cela, l'information "Qui est le commercial" doit arriver à la table **Detail des commandes** : c'est à cela que servent les relations.

Nom commercial	Somme de Quantité
Kian Motebassem	572
Lénaïck Le Peutrec	572
Ronald Mittler	572
Total	**572**

En l'absence de relation, les chiffres sont manifestement faux

Nom commercial	Somme de Quantité
Kian Motebassem	190
Lénaïck Le Peutrec	197
Ronald Mittler	185
Total	**572**

Une fois la relation créée, les résultats sont corrects

La relation est donc un chemin, permettant la circulation entre un point A et un point B, passant par une ou plusieurs tables. Ce mécanisme, dans le langage de Power BI, se dit « propagation du filtre », et c'est le concept fondamental de Power BI.

Le chemin peut être à sens unique (c'est ce qu'indique la petite flèche sur le trait représentant la relation – par exemple, entre Catégorie et Livres), ou à double sens. Cette caractéristique est déterminée par les données présentes dans les colonnes de part et d'autre de la relation. C'est ce qu'on appelle la cardinalité, figurée par le chiffre 1 ou l'astérisque sur la relation :

- La relation la plus courante est dite relation 1-à-N (ou un à plusieurs – le N est représenté par un astérisque) : à une ligne de la table du côté 1 de la relation correspondent une ou plusieurs lignes dans la table du côté N (ou plus précisément, à une ligne du côté 1 peuvent correspondre zéro, une ou plusieurs lignes du côté N). La direction naturelle de cette relation (le sens de la circulation) est de 1 vers N.

  Dans notre modèle, c'est le cas de la relation entre la table **Catégories** et la table **Livres** : une catégorie peut regrouper zéro, un ou plusieurs livres.

- Une relation de N-à-1 revient exactement au même, et le sens de circulation naturel reste le même, de 1 vers N.
- Un deuxième type de relation, plus rare, peut advenir : la relation de 1-à-1, où une valeur de la première table à un équivalent et un seul dans la deuxième.

  Il n'y a pas d'exemple de ce cas dans notre modèle. Dans ce type de situation, Power BI crée une relation à deux sens de circulation. Attention, ce type de relation, appelé bidirectionnel, peut être délicat à utiliser.

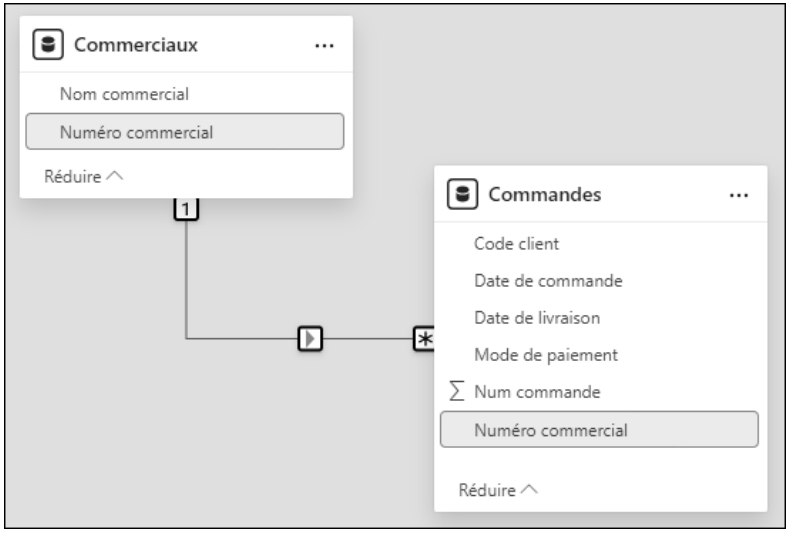
- Enfin, le troisième type de relation est plus complexe, et amène le plus généralement à revoir son modèle et à le transformer : la relation de N-à-N, où une ligne de la première table est liée à plusieurs lignes de la seconde, et inversement, une ligne de la seconde est liée à plusieurs lignes de la première.

  Là encore, pas d'exemple dans notre modèle.

La cardinalité de la relation est calculée par Power BI lors de sa création.

La relation que vous devez privilégier est la relation 1-à-N : c'est la plus performante, et la plus simple à utiliser. Pour plus de détails sur les relations 1-à-1 et N-à-N, complexes, je vous renvoie au deuxième livre que je consacre à Power BI.

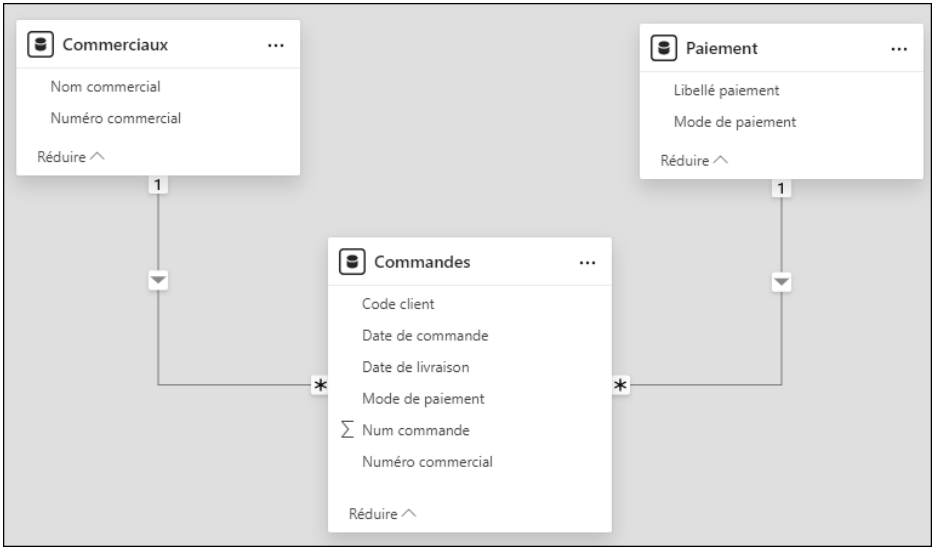
1. La relation de 1-à-N



Sur cette illustration figurent plusieurs informations importantes :

- La relation porte sur le champ **Numéro commercial** (notez qu'il n'est pas nécessaire que les deux champs liés portent le même nom).
- La table **Commerciaux** contient une liste de personnes identifiées de manière unique par le **Numéro commercial**. Un même numéro ne peut pas être utilisé plusieurs fois.
- En revanche, du côté des commandes, le même **Numéro commercial** peut se retrouver plusieurs fois.
- La relation est donc bien de type 1-à-N (1 à *).
- Enfin, le sens (la direction) de la relation va de la table **Commerciaux** à la table **Commandes**. Pour renforcer encore cette information, une flèche montre clairement la direction de la relation.

Ce dernier point est capital : en effet, dans Power BI, un filtre se propage automatiquement et en cascade le long d'une relation de 1-à-N : il passe de table en table et ne s'arrête que lorsqu'il rencontre une relation de sens inverse, N-à-1.



Dans cet exemple, un filtre posé sur la table **Commerciaux** va automatiquement se propager à la table **Commandes**, mais en revanche, il ne peut pas continuer vers la table **Paiement**.

La relation de 1-à-N est la plus courante, et la plus souhaitable, pour que le modèle fonctionne correctement. Elle ne pose aucun problème de mise en place et d'utilisation.

A. Introduction

L'intégralité de ce chapitre est consacré à l'idée de « visuel augmenté », à savoir l'imbrication des visuels et des formules DAX en vue d'obtenir une meilleure représentation de l'information : dynamique (une partie du résultat est liée aux choix de l'utilisateur), conditionnelle (certains points de données sont mis en avant) ou encore contextualisée (l'information est mise dans un contexte plus vaste, le temps par exemple).

C'est pourquoi nous commencerons par un rappel des notions qu'il est essentiel de maîtriser. Nous ne rentrerons cependant pas dans le détail. Pour cela vous pouvez consulter notre ouvrage *Power BI Desktop, Reporting et analyse de données au quotidien (3e édition)*.

B. Fondamentaux

1. Les mesures et les colonnes

Au-delà de l'aspect lié au stockage des données qui est optimisé dans le cas des mesures, la différence essentielle réside dans le fait que les mesures sont dynamiques, alors que les colonnes sont fixes.

Les colonnes sont en effet calculées lors de leur création initiale, puis à chaque actualisation du rapport. Un segment n'a donc aucun effet sur la valeur stockée dans une colonne. C'est exactement l'inverse pour les mesures : elles ne sont calculées qu'au moment où elles sont déposées dans un rapport, et sont donc impactées par un segment.

Prenez l'exemple du classement de produits : la fonction RANKX permet de classer en fonction d'une valeur (par exemple le montant) et peut être utilisée pour créer une colonne ou une mesure. La colonne va classer les produits en fonction de la quantité, mais de la quantité globale (c'est-à-dire, toutes années, toutes catégories de produits, tous types de clientèle). Et ce classement sera figé, et ne changera qu'au prochain chargement de données.

Calculé sous forme de mesure, le classement devient sensible aux segments présents sur le rapport : l'utilisateur pourra donc obtenir un classement en fonction d'une année, de l'année en cours, d'une catégorie de produits, ou encore d'un type de clientèle. A chaque changement, la mesure classement est recalculée.

L'inconvénient, c'est que l'écriture de la mesure est plus complexe que celui de la colonne.

2. Le contexte de filtre, le contexte de ligne

Il ne s'agit pas pour nous ici de reprendre intégralement ces notions, mais simplement de rappeler que le contexte de ligne est invoqué automatiquement lors de la création d'une colonne ainsi que lors de l'utilisation d'une fonction de type itérateur (en particulier les fonctions X – SUMX, etc.).

Le contexte de filtre, bien plus courant, est défini par les visuels (tables, graphiques, segments) présents sur le rapport (et parfois sur les autres rapports). Mais il peut aussi être manipulé par formule, et c'est précisément le rôle de la fonction CALCULATE.

Le contexte de ligne filtre la table sur laquelle il s'applique, et n'en retient qu'une ligne. L'utilisation des fonctions RELATED ou RELATEDTABLE permet cependant au contexte de ligne de se propager vers d'autres tables, dans le sens N vers 1 de la relation pour la première, dans le sens 1 à N pour la seconde.

Le contexte de filtre filtre le modèle dans son ensemble, selon le sens de propagation de 1 à N le long des relations. L'utilisation de la fonction CROSSFILTER, ou d'une relation bidirectionnelle, permet toutefois au filtre de se propager dans le sens N à 1.

Enfin, dans certains cas, le contexte de ligne est transformé en contexte de filtre, selon le principe de la transition de contexte que nous avons déjà bien abordé.

3. Aide-mémoire DAX et mise en forme

Afin d'assurer la lisibilité des formules, et notamment lorsqu'elles deviennent complexes, certaines règles de mises en forme peuvent être suivies :

- Les colonnes référencées le sont toujours en précisant le nom de la table d'abord, alors que les mesures sont simplement référencées entre crochets : cela aide en particulier à mieux « voir » le CALCULATE implicite.
- Ajoutez autant de commentaires que nécessaire, précédés d'un //.
- Facilitez la vue des parenthèses ouvrantes et fermantes.
- Allez à la ligne autant de fois que vous le voulez.

```
[Montant moyen facturé] =
   AVERAGEX (
       Date[Date],
       [Montant facturé]
   )
```

Notez aussi qu'une variante peut consister à décaler la virgule sur la ligne suivante, pour faciliter la mise en commentaire d'une ligne et par conséquent les tests :

```
[Mesure] =
   CALCULATE (
       expression
       , filtre1
       , filtre2
       , filtre3
   )
```

Écrite comme ça, la formule permet facilement de mettre une partie du code en commentaire (ici, le filtre 3) :

```
[Mesure] =
   CALCULATE (
       expression
       , filtre1
       , filtre2
//     ; filtre3
   )
```

Voici un tableau des formules clés du DAX, base sur laquelle nous allons pouvoir bâtir des formules plus complexes dans le reste de ce chapitre.

Attention, le modèle est un peu différent de celui que nous utilisons, puisque les tables **Commandes** et **Détails de commandes** d'une part, **Livres** et **Catégories** d'autre part sont ici restées distinctes. C'est un modèle entête-détail en flocon.

SUM	Pour faire la somme des valeurs dans une colonne, une fois le contexte de filtre appliqué
`SUM(Colonne)`	`quantité = SUM('Detail des commandes'[Quantité])`
SUMX	Pour effectuer un calcul sur chaque ligne de la table passée en paramètre
`SUMX(Table, Expression)`	`montant = SUMX( 'Detail des commandes', [quantité] * 'Detail des commandes'[Prix de vente] )`

RELATED	Argument de SUMX notamment, permet de voir les colonnes des tables situées « au-dessus » de celle qui est passée en paramètre

	`montant prix catalogue =`
	`SUMX(`
	`  'Detail des commandes',`
	`  [quantité] * RELATED('Livres'[Prix catalogue])`
	`)`

AVERAGEX	Pour calculer une moyenne au niveau de détail que vous souhaitez (VALUES)

`AVERAGEX(Table, Expression)`
Notez que VALUES retourne une table
Ce schéma fonctionne également avec MINX et MAXX
	`montant mensuel moyen =`
	`AVERAGEX(`
	`  VALUES(Datum[AAMM]),`
	`  [montant]`
	`)`

Compter avec COUNTROWS et DISTINCTCOUNT	
`COUNTROWS(Table)`	`cpt client = COUNTROWS(Clients)`
`DISTINCTCOUNT (Colonne)`	`nb de clients ayant passé des commandes =` `DISTINCTCOUNT(Commandes[Code client])`

	`% de clients ayant commandé / total client =`
	`DISTINCTCOUNT(Commandes[Code client]) /`
	`COUNTROWS(Clients)`

`FILTER(Table, Expression de filtre)`	`cpt clients nantais =`
	`COUNTROWS(`
	`  FILTER(`
	`  Clients,`
	`  Clients[Ville] = "Nantes"`
	`  )`
	`)`

	Un peu plus complexe…
	clients ayant passé plus de 2 commandes = COUNTROWS(FILTER(SUMMARIZE(Commandes, Commandes[Code client], "nb comm", COUNT(Commandes[Code client])), [nb comm] > 1))
	clients n'ayant pas commandé = COUNTROWS(EXCEPT(VALUES(Clients[Code client]), VALUES(Commandes[Code client])))
CALCULATE	Pour modifier un filtre (ajouter, supprimer, remplacer) CALCULATE est la fonction majeure du DAX
CALCULATE(expression, expression de filtre)	quantité Hachette = CALCULATE([quantité], Vendeurs[Nom vendeur] = "Hachette")
	Vendeurs[Nom vendeur] IN { "Hachette" , "Favreau" }
	quantité Hachette à Nantes = CALCULATE([quantité], Vendeurs[Nom vendeur] = "Hachette", Clients[Ville] = "Nantes")
	quantité annuelle = CALCULATE([quantité], ALL(Calendrier[Annee Mois]))

	`quantité tous livres =`
	`CALCULATE(`
	`  [quantité],`
	`//  ALL(Livres)`
	`//  ALL()`
	`  ALLSELECTED()`
	`)`

USERELATIONSHIP	Argument de CALCULATE, permet d'activer le temps d'un calcul une relation (physique) inactive

	`quantité livraison =`
	`CALCULATE(`
	`  [quantité],`
	`  USERELATIONSHIP(`
	`  Datum[Date],`
	`  'Detail des commandes'[Date de livraison]`
	`  )`
	`)`

CROSSFILTER	Argument de CALCULATE, permet d'inverser le temps d'un calcul le sens d'une relation

	`quantité livraison =`
	`CALCULATE(`
	`  [quantité],`
	`  CROSSFILTER (`
	`  Datum[Date],`
	`  'Detail des commandes'[Date de livraison],`
	`  Both`
	`  )`
	`)`

	`cpt catégories =`
	`CALCULATE(`
	`  DISTINCTCOUNT(Livres[Code catégorie]),`
	`  CROSSFILTER(`
	`  'Detail des commandes'[Numéro livre],`
	`  Livres[Numéro livre],`
	`  Both`
	`  )`
	`)`

IF ISINSCOPE	Faire un calcul seulement si.

	`% catégorie / suspense =`
	`IF(`
	`  ISINSCOPE(Livres[Signification]),`
	`  [quantité] / [quantité Suspense] -1,`
	`  "n/a"`
	`)`

Time Intelligence	Comparer deux périodes

C'est tout ce qu'il y a besoin de connaître !
Le dernier argument peut être YEAR, QUARTER, MONTH ou DAY
	`quantité M-3 (comparaison) =`
	`CALCULATE(`
	`  [quantité],`
	`  DATEADD(`
	`  Datum[Date],`
	`  -3, MONTH`
	`  )`
	`)`

Time Intelligence	Cumuler des résultats
Variantes : TOTALQTD, TOTALMTD	`TOTALYTD([quantité], Datum[Date])`
SUMMARIZECOLUMNS	Pour créer une table physique (très utile pour créer des tables d'agrégats)

	`agrégat ville année =`
	`SUMMARIZECOLUMNS(`
	`  Clients[Ville],`
	`  Datum[Année],`
	`  FILTER(`
	`  Clients,`
	`  Clients[Ville] in {"ANGERS", "NANTES"}`
	`  ),`
	`  "qte agr.", [quantité],`
	`  "montant agr.", [montant])`

4. Utilisation de la vue de requête DAX

La vue de requête DAX est particulièrement utile dans quatre situations :

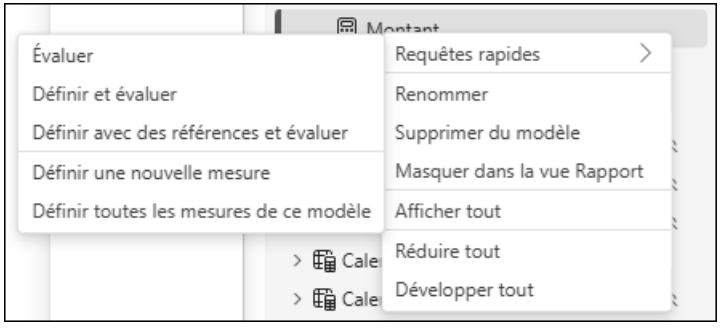
- prendre en main un rapport et en comprendre l'organisation des mesures ;
- prévisualiser le résultat d'une formule grâce à la fonction **EVALUATE** ou les 100 premières lignes d'une table ;
- faire des mises à jour groupées ;
- utiliser Copilot pour écrire des formules DAX.

a. Prendre en main un rapport

Vous prenez le relais sur le développement ou la maintenance d'un rapport, et vous avez besoin de faire un tour d'horizon des mesures qu'il contient. Jusque-là, il fallait pour cela recourir à des outils externes.

Dans la **Vue de requête DAX**, faites un clic droit sur une mesure et choisissez **Requêtes rapides** puis une des options suivantes :

- **évaluer** : pour calculer la mesure ;
- **définir et évaluer** : comme ci-dessus, avec l'affichage de la formule et la possibilité de la modifier et de mettre à jour le modèle ;
- **définir avec des références et évaluer** : définit la mesure et les mesures auxquelles elle fait appel (afficher les formules), et calcule le résultat ;
- **définir toutes les mesures de ce modèle** : pour afficher toutes les formules utilisées dans le rapport.



Sur l'exemple suivant, nous avons défini toutes les mesures du modèle, résultant en presque 250 lignes de DAX :



- Passez la souris sur une mesure pour afficher une fenêtre pop-up avec le code de la mesure citée.



Notez au passage le Code Lens : c'est l'indication en gris **Mettre à jour le modèle** : remplacer la mesure. Comme son intitulé l'indique, le Code Lens permet d'implémenter directement les modifications faites sur une mesure à partir de la **Vue de requête DAX**.

Le bouton **Mettre à jour le modèle avec les modifications (26)** en haut de page permet d'implémenter toutes les modifications et de mettre à jour toutes les mesures concernées.

b. Faire des mises à jour groupées

Que vous affichiez la formule d'une seule mesure, ou de toutes les mesures, la **Vue de requête DAX** offre un outil classique de recherche/remplacement, bien utile lorsque vous avez besoin d'effectuer une mise à jour groupée.

- Dans l'onglet **Accueil** cliquez sur le bouton **Remplacer** du groupe **Modification** ou utilisez le raccourci Ctrl H).

