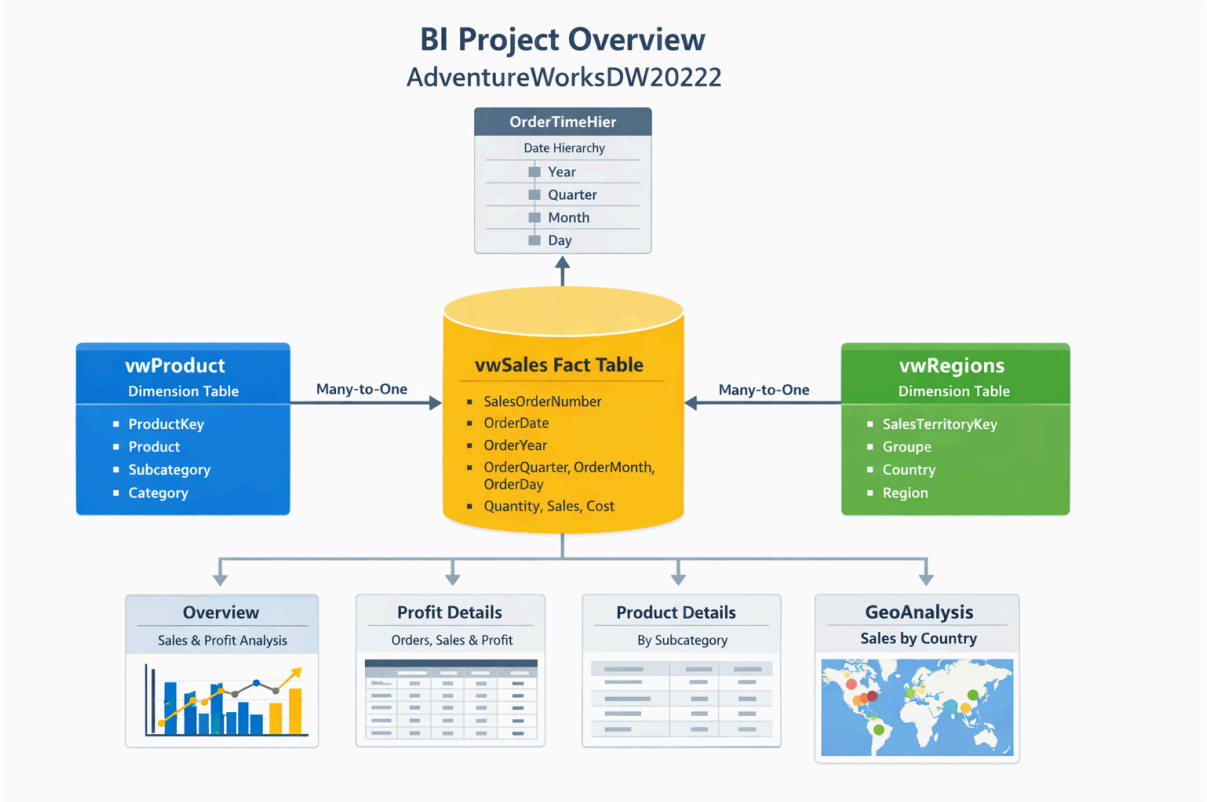

📊 Projet BI – Analyse des ventes et de la rentabilité

Cours pratique Power BI & SQL Server (cas réel)

🧩 Contexte du projet (Scénario)

La société **AdventureWorks**, une entreprise manufacturière mondiale fictive qui produit des équipements et accessoires de cyclisme, dispose d'un entrepôt de données (`AdventureWorksDW2022`) alimenté quotidiennement par ses systèmes opérationnels.

La direction constate que :

- le **chiffre d'affaires global progresse**
- mais la **rentabilité varie fortement** selon :
    - les périodes
    - les régions
    - les catégories de produits

La société décide donc de mettre en place un **reporting décisionnel Power BI**, connecté directement à SQL Server, afin de :

- suivre les ventes dans le temps
- analyser la profitabilité
- identifier les zones géographiques et les produits les plus performants

👉 Le rôle du **Data Analyst** est de **transformer les données brutes** en **indicateurs clairs, fiables et exploitables** par les équipes métiers.

🎯 Objectifs business du reporting

Le rapport Power BI doit permettre de répondre aux questions suivantes :

📈 Performance globale

- Comment évoluent les **ventes et la marge** dans le temps ?
- Existe-t-il des **saisonnalités** ou des périodes plus rentables ?

🌍 Analyse géographique

- Quels **groupes de régions** et **pays** génèrent le plus de ventes ?
- La rentabilité est-elle homogène selon les régions ?

📦 Analyse produits

- Quelles **catégories et sous-catégories** génèrent le plus de volume ?
- Les produits les plus vendus sont-ils aussi les plus rentables ?

💰 Rentabilité

- Quelle est la **marge réelle** (Profit Margin) selon :
  - le temps
  - les régions
  - les produits
- Identifier les segments à **faible marge** malgré un volume élevé

🧑‍💼 Rôle et responsabilités du Data Analyst

Le Data Analyst est responsable de :

- La **préparation des données** dans SQL Server
- La création de **vues propres et orientées BI**
- La modélisation des **mesures DAX**
- La conception d'un **rapport Power BI structuré, lisible et interactif**

Il doit garantir :

- la **cohérence des calculs**
- la **justesse des totaux**
- une **navigation fluide** pour l'utilisateur métier

🧠 Attentes finales de la direction

À la fin du projet, le rapport doit permettre :

✔ Une lecture rapide de la performance globale
✔ Une analyse fine par région, produit et période
✔ Des décisions basées sur la **marge**, pas seulement sur le volume
✔ Un modèle robuste, évolutif et maintenable

📌 **Dans la suite de ce cours**, nous allons :

1. Analyser les **vues SQL Server**
2. Transformer les données côté SQL
3. Construire un modèle Power BI propre
4. Créer les mesures DAX pas à pas

🛠️ Mise en place de l'environnement & préparation des données

Projet Power BI – AdventureWorksDW2022

1️⃣ Installation de l'environnement de travail

Avant de commencer l'analyse et la création du rapport Power BI, il est indispensable de disposer d'un **environnement technique stable** permettant de :

- stocker les données
- interroger la base
- transformer les données côté SQL Server

1.1 Installation de SQL Server Express

Pour ce projet, nous utilisons **SQL Server Express**, une version gratuite largement suffisante pour l'apprentissage et les projets BI.

Étapes générales :

- Télécharger **SQL Server Express**
- Choisir l'option **Basic** ou **Custom**
- Installer le moteur de base de données
- Vérifier que le service SQL Server est bien démarré

👉 La base **AdventureWorksDW2022** sera ensuite restaurée ou attachée au serveur.

1.2 Choix du client SQL

Deux options sont possibles :

- **SQL Server Management Studio (SSMS)**
- **Visual Studio Code (VSCode)** ✅ *(utilisé dans ce projet)*

Pourquoi VSCode ?

- plus léger

- plus rapide à lancer
- très utilisé dans les environnements modernes data / cloud

👉 Les apprenants sont libres d'utiliser SSMS s'ils le souhaitent.

1.3 Installation de VSCode et de l'extension SQL Server

Si vous choisissez **VSCode** :

1. Installer **Visual Studio Code**
2. Ouvrir VSCode
3. Aller dans **Extensions**
4. Installer l'extension officielle :
    - **SQL Server (mssql)** – Microsoft

Cette extension permet :

- de se connecter à SQL Server
- d'exécuter des requêtes SQL
- de gérer plusieurs connexions

1.4 Création d'un profil de connexion SQL Server

Dans VSCode :

1. Ouvrir la palette de commandes (`Ctrl + Shift + P`)
2. Rechercher **MS SQL: Add Connection**
3. Renseigner :
    - Server name (ex : `localhost\SQLEXPRESS`)
    - Authentication type
    - Database : `AdventureWorksDW2022`
4. Sauvegarder le profil de connexion

Une fois connecté, la base apparaît dans l'explorateur SQL.

📦 Installation de la base de données AdventureWorksDW2022

Pour ce projet, nous utilisons la base **AdventureWorksDW2022**, fournie sous forme d'un fichier de sauvegarde (`.bak`).
Cette base doit être **restaurée manuellement** dans SQL Server Express.

1️⃣ Vérifier le contenu du fichier de sauvegarde

Avant de restaurer la base, il est indispensable d'identifier :

- le nom logique des fichiers de données (`.mdf`)
- le nom logique des fichiers de log (`.ldf`)

```
RESTORE FILELISTONLY

FROM DISK = 'D:\Logiciels\AdventureWorksDW2022.bak';

GO
```

👉 Cette requête retourne notamment :

- `LogicalName` des fichiers
- leur type (`D` = Data, `L` = Log)

Ces noms seront utilisés dans l'étape suivante.

2️⃣ Restaurer la base AdventureWorksDW2022

Une fois les noms logiques identifiés, exécuter la restauration :

```
RESTORE DATABASE AdventureWorksDW2022

FROM DISK = 'D:\Logiciels\AdventureWorksDW2022.bak'

WITH

MOVE 'AdventureWorksDW2022'

TO 'C:\Program Files\Microsoft SQL Server\MSSQL17.SQLEXPRESS\MSSQL\DATA\AdventureWorksDW2022.mdf',

MOVE 'AdventureWorksDW2022_log'
```

TO 'C:\Program Files\Microsoft SQL Server\MSSQL17.SQLEXPRESS\MSSQL\DATA\AdventureWorksDW2022_log.ldf',

RECOVERY;

GO

📌 **Points importants** :

- Le chemin doit correspondre à votre instance SQL Server
- Adapter `MSSQL17.SQLEXPRESS` si nécessaire
- Les noms logiques (`AdventureWorksDW2022`, `AdventureWorksDW2022_log`) doivent correspondre au résultat de `RESTORE FILELISTONLY`

3️⃣ Vérifier que la base est bien installée

SELECT name

FROM sys.databases;

👉 La base **AdventureWorksDW2022** doit apparaître dans la liste.

Vous pouvez ensuite tester l'accès :

USE AdventureWorksDW2022;

GO

SELECT TOP 10 *

FROM dbo.FactResellerSales;

1.5 Vérification du bon fonctionnement (requêtes de test)

Exécuter quelques requêtes simples pour vérifier que tout fonctionne correctement.

```
SELECT GETDATE();


USE AdventureWorksDW2022;
GO


SELECT TOP 10 *
FROM dbo.FactResellerSales;
```

👉 Si les résultats s’affichent correctement, l’environnement est prêt.

2️⃣ Comprendre le modèle de données (avant toute transformation)

Avant de transformer les données, il est essentiel de comprendre **comment les tables sont liées entre elles**.

Dans ce projet, nous travaillons avec **trois grandes entités** :

- **Sales** (ventes)
- **Product** (produits)
- **Regions** (zones géographiques)

2.1 Table de faits vs tables de dimensions

📦 Table de faits

- Contient les **mesures quantitatives**
- Exemple : ventes, coûts, quantités
- Très volumineuse
- Exemple ici :
  - `FactResellerSales` → **Sales**

📚 Tables de dimensions

- Contiennent le **contexte descriptif**
- Peu volumineuses
- Exemple :
  - Produits
  - Catégories
  - Régions
  - Temps

2.2 Notions de clés primaires et étrangères

- **Clé primaire (Primary Key)** :
  - identifie une ligne de manière unique
  - ex : `ProductKey`
- **Clé étrangère (Foreign Key)** :
  - référence une clé primaire d'une autre table
  - ex : `FactResellerSales.ProductKey`

2.3 Types de relations

Dans un modèle BI classique :

- **One-to-Many (1 → n)**
  - Un produit → plusieurs ventes
  - Une région → plusieurs ventes
- **Many-to-One (n → 1)**
  - Plusieurs ventes → un produit
  - Plusieurs ventes → une région

👉 En Power BI :

- la table de faits est du côté *many*
- les dimensions du côté *one*

3️⃣ Nettoyage et transformation des données (SQL Server)

Nous allons maintenant préparer des **vues SQL propres**, orientées BI, afin de simplifier :

- le modèle Power BI
- les relations
- les calculs DAX

3.1 Dimension Produit – nettoyage de base

🎯 Objectif métier

Analyser uniquement les **produits réellement vendus**, en excluant :

- produits intermédiaires
- composants non commercialisés

🔍 Règle métier

👉 `FinishedGoodsFlag = 1`

-- DimProduct

SELECT

```
    ProductKey,
    EnglishProductName AS Product,
    StandardCost,
    Color,
    ProductSubcategoryKey
FROM DimProduct
WHERE FinishedGoodsFlag = 1;
```

3.2 Dimension Catégorie et Sous-catégorie

Ces tables apportent une **structure hiérarchique produit** indispensable au reporting.

```
-- DimProductCategory
SELECT
    ProductCategoryKey,
    EnglishProductCategoryName AS Category
FROM DimProductCategory;


-- DimProductSubcategory
SELECT
    ProductSubcategoryKey,
    EnglishProductSubcategoryName AS Subcategory,
    ProductCategoryKey
FROM DimProductSubcategory;
```

3.3 Création d'une dimension Produit unifiée

🎯 Objectif BI

Power BI fonctionne mieux avec :

- moins de tables
- des dimensions dénormalisées

👉 Nous regroupons :

- produit
- sous-catégorie
- catégorie
  dans **une seule table Product**

```
SELECT
    ProductKey,
    EnglishProductName AS Product,
    StandardCost,
    Color,
    dps.EnglishProductSubcategoryName AS Subcategory,
    dpc.EnglishProductCategoryName AS Category
FROM AdventureWorksDW2022.dbo.DimProduct dp
    LEFT JOIN DimProductSubcategory dps
        ON dp.ProductSubcategoryKey = dps.ProductSubcategoryKey
    LEFT JOIN DimProductCategory dpc
        ON dps.ProductCategoryKey = dpc.ProductCategoryKey
WHERE FinishedGoodsFlag = 1;
GO
```

3.4 Table de faits Sales – enrichissement temporel

🎯 Objectif métier

Permettre des analyses :

- par année
- par trimestre
- par mois
- par jour
- par jour de la semaine

Sans dépendre du moteur Power BI.

```
SELECT
    SalesOrderNumber,
    OrderDate,
    YEAR(OrderDate) AS OrderYear,
    DATEPART(QUARTER, OrderDate) AS OrderQuarter,
    MONTH(OrderDate) AS OrderMonth,
    DATENAME(MONTH, OrderDate) AS OrderMonthName,
    DAY(OrderDate) AS OrderDay,
    DATENAME(WEEKDAY, OrderDate) AS OrderWeekdayName,
    ProductKey,
    ResellerKey,
    EmployeeKey,
    SalesTerritoryKey,
    OrderQuantity AS Quantity,
    UnitPrice,
    SalesAmount AS Sales,
    TotalProductCost AS Cost
```

```
FROM AdventureWorksDW2022.dbo.FactResellerSales;

GO
```

Pourquoi enrichir côté SQL ?

- cohérence des calculs
- meilleures performances
- modèle Power BI plus simple
- hiérarchies prêtes à l'emploi

🧠 Résumé pédagogique

✔ Modèle en étoile (Star Schema)
✔ Table de faits : Sales
✔ Dimensions : Product, Regions, Time
✔ Données filtrées selon des règles métier
✔ SQL Server utilisé comme couche de préparation

🏗️ Création des Vues SQL pour Power BI

Dans un projet Power BI, il est **fortement recommandé** de créer des **vues SQL côté serveur** pour plusieurs raisons :

- **Simplicité** : Power BI importe directement des tables prêtes à l'emploi, sans avoir à transformer les colonnes à chaque chargement.
- **Performance** : les calculs et filtrages sont effectués côté SQL Server, ce qui réduit le volume et la complexité côté Power BI.
- **Maintenance** : les transformations sont centralisées dans la base, faciles à mettre à jour et réutilisables pour d'autres rapports.
- **Cohérence** : toutes les analyses utilisent les mêmes règles métier (ex : produits finis uniquement, territoires valides, enrichissements temporels).

Nous allons créer trois vues principales correspondant aux **dimensions et table de faits** utilisées dans le rapport :

- `vwSales` → Faits / ventes
- `vwProduct` → Dimension produit
- `vwRegions` → Dimension géographique

1️⃣ Vue des ventes : `vwSales`

✅ Script SQL

```
USE AdventureWorksDW2022;

GO


-- Supprimer la vue si elle existe déjà

IF OBJECT_ID('dbo.vwSales', 'V') IS NOT NULL

   DROP VIEW dbo.vwSales;

GO
```

```
-- Création de la vue Sales
CREATE VIEW dbo.vwSales AS
SELECT
    SalesOrderNumber,
    OrderDate,
    YEAR(OrderDate) AS OrderYear,
    DATEPART(QUARTER, OrderDate) AS OrderQuarter,
    MONTH(OrderDate) AS OrderMonth,
    DATENAME(MONTH, OrderDate) AS OrderMonthName,
    DAY(OrderDate) AS OrderDay,
    DATENAME(WEEKDAY, OrderDate) AS OrderWeekdayName,
    ProductKey,
    ResellerKey,
    EmployeeKey,
    SalesTerritoryKey,
    OrderQuantity AS Quantity,
    UnitPrice,
    SalesAmount AS Sales,
    TotalProductCost AS Cost
FROM dbo.FactResellerSales;
GO

-- Vérification de la vue
SELECT TOP 10 * FROM dbo.vwSales;
GO
```

Explications métier :

- Les colonnes temporelles (`OrderYear`, `OrderQuarter`, etc.) permettent la création de hiérarchies dans Power BI.
- Les colonnes `SalesAmount` et `TotalProductCost` sont préparées pour calculer `Profit` et `ProfitMargin`.
- Cette vue servira de **table de faits** dans le modèle Power BI.

2️⃣ Vue de la dimension produit : `vwProduct`

✅ Script SQL

```
-- Supprimer la vue si elle existe déjà
IF OBJECT_ID('dbo.vwProduct', 'V') IS NOT NULL
    DROP VIEW dbo.vwProduct;
GO

-- Création de la vue Product
CREATE VIEW dbo.vwProduct AS
SELECT
    dp.ProductKey,
    dp.EnglishProductName AS Product,
    dp.StandardCost,
    dp.Color,
    dps.EnglishProductSubcategoryName AS Subcategory,
    dpc.EnglishProductCategoryName AS Category
FROM dbo.DimProduct dp
LEFT JOIN dbo.DimProductSubcategory dps
```

```
    ON dp.ProductSubcategoryKey = dps.ProductSubcategoryKey
LEFT JOIN dbo.DimProductCategory dpc
    ON dps.ProductCategoryKey = dpc.ProductCategoryKey
WHERE dp.FinishedGoodsFlag = 1;
GO


-- Vérification de la vue
SELECT TOP 10 * FROM dbo.vwProduct;
GO
```

Explications métier :

- Seuls les **produits finis** sont inclus (`FinishedGoodsFlag = 1`).
- La jointure avec **Subcategory** et **Category** permet de créer des **hiérarchies produits** directement utilisables dans Power BI.
- Cette vue sera une **dimension** liée à la table de faits `vwSales` via `ProductKey`.

3️⃣ Vue de la dimension région : `vwRegions`

✅ Script SQL

```
-- Supprimer la vue si elle existe déjà
IF OBJECT_ID('dbo.vwRegions', 'V') IS NOT NULL
    DROP VIEW dbo.vwRegions;
GO


-- Création de la vue Regions
CREATE VIEW dbo.vwRegions AS
```

```
SELECT
    SalesTerritoryKey,
    SalesTerritoryRegion AS Region,
    SalesTerritoryCountry AS Country,
    SalesTerritoryGroup AS Groupe
FROM dbo.DimSalesTerritory
WHERE SalesTerritoryAlternateKey <> 0;
GO

-- Vérification de la vue
SELECT TOP 10 * FROM dbo.vwRegions;
GO
```

Explications métier :

- Les territoires invalides ou tests (`SalesTerritoryAlternateKey = 0`) sont exclus.
- Cette vue servira de **dimension géographique** pour filtrer ou segmenter les ventes dans Power BI.
- La hiérarchie `Groupe → Country → Region` peut être utilisée directement dans un slicer hiérarchique.

✅ Résumé

- Nous avons créé **3 vues SQL** : `vwSales`, `vwProduct`, `vwRegions`.
- Ces vues sont **prêtes à être utilisées en mode Import dans Power BI**.
- Les relations entre les tables seront basées sur les clés **ProductKey** et **SalesTerritoryKey**.
- La préparation côté SQL garantit :
    - des performances optimales
    - des calculs cohérents
    - une facilité de maintenance pour les rapports futurs.

🏗️ Modélisation dans Power BI

Après avoir préparé et testé nos vues SQL (`vwSales`, `vwProduct`, `vwRegions`), nous allons maintenant **charger les données dans Power BI** et construire un **modèle robuste et exploitable** pour le reporting.

1️⃣ Importer les vues SQL

Étapes :

1. **Get Data → SQL Server**
2. Sélectionner le serveur et la base `AdventureWorksDW2022`
3. Choisir le mode **Import** (pas DirectQuery pour ce projet)
4. Sélectionner les **3 vues** : `vwSales`, `vwProduct`, `vwRegions`
5. Cliquer sur **Load** directement dans le **Report View**

Explication métier :

- L'**Import** permet un accès rapide et fluide aux données
- Les transformations lourdes ont été faites côté SQL Server
- Les données sont prêtes pour la construction des **hiérarchies et mesures**

2️⃣ Vérification des relations automatiques

1. Aller dans **Model View**
2. Observer les relations que Power BI a créées automatiquement

Points à vérifier :

- `vwSales[ProductKey]` → `vwProduct[ProductKey]` (Many-to-One)
- `vwSales[SalesTerritoryKey]` → `vwRegions[SalesTerritoryKey]` (Many-to-One)

💡 **Business view** :
Ces relations permettent de lier les ventes à leur **produit** et à leur **région** pour analyser les **ventes et la marge par catégorie ou territoire**.

3️⃣ Masquer les colonnes clés inutiles

Pour simplifier le modèle et l'interface utilisateur :

1. Dans **Model View → Data Pane**
2. **Expand All**
3. Multi-sélection de toutes les colonnes **Key** dans chaque table
4. Clic droit → **Hide in report view** (`Is Hidden = Yes`)

💡 Pourquoi :

- Les utilisateurs finaux n'ont pas besoin de voir les clés techniques
- Le modèle reste **lisible et propre**
- Les relations fonctionnent toujours, les colonnes restent utilisées en interne

---

4️⃣ Catégorisation des colonnes

Pour les visuels géographiques :

1. Sélectionner `vwRegions[Country]`
2. **Column tools → Data Category → Country/Region**

💡 Explication :

- Permet à Power BI de **reconnaître les pays** pour les cartes et cartes choroplèthes
- Evite les erreurs de géocodage dans les visuels

5️⃣ Création de la hiérarchie temporelle

Dans la table `vwSales` :

1. Sélectionner les colonnes :
   - `OrderYear`, `OrderQuarter`, `OrderMonth`, `OrderDay`
2. Clic droit → **Create Hierarchy**
3. Renommer la hiérarchie `OrderTimeHier`
4. Réordonner les niveaux : Year → Quarter → Month → Day

💡 Explication Business :

- Permet le **drill-down** dans les visuels temporels
- Facilite l'analyse des ventes et marges par **année, trimestre, mois et jour**

6️⃣ Organisation du modèle : Display Folders

Pour améliorer la lisibilité :

1. Sélectionner toutes les **colonnes temporelles** dans `vwSales`
2. Dans **Properties Pane → Display Folder**, créer un dossier nommé `Time`
3. Glisser les colonnes dedans

💡 Avantage :

- Les colonnes sont regroupées logiquement
- L'interface Power BI reste **propre et intuitive**

7️⃣ Création des mesures Business

Dans **Report View** :

1. Clic droit sur `vwSales → New measure`

Mesures à créer :

Orders = DISTINCTCOUNT(vwSales[SalesOrderNumber])

- Compte le nombre de commandes
- Permet d'analyser la performance opérationnelle

Profit = SUM(vwSales[Sales]) - SUM(vwSales[Cost])

- Représente la **marge brute**
- Indicateur clé pour le contrôle de gestion

ProfitMargin = DIVIDE([Profit], SUM(vwSales[Sales]))

- Ratio de rentabilité (%)
- Utilisé pour comparer la **profitabilité entre régions, produits ou périodes**

💡 **Business rationale** :

- Ces mesures sont calculées dynamiquement selon le **contexte de filtre** (pays, catégorie, période)
- Elles permettent une **analyse décisionnelle fiable**

8️⃣ Construction des visuels – Page *Overview*

Line and Stacked Column Chart

- X-axis : `OrderTimeHier`
- Column Values : `Sales (Sum)`
- Line Values : `ProfitMargin`

Stacked Column Chart – Sales by Country & Category

- X-axis : `Country`
- Y-axis : `Sales (Sum)`
- Legend : `Category`

Stacked Bar Chart – Quantity by Category

- Y-axis : `Category`
- X-axis : `Quantity (Sum)`

Slicers

- `OrderYear`
- `RegionsHier` (hiérarchie Groupe → Country → Region)

💡 Business view : permet un **aperçu synthétique des ventes, marges et volumes par région et produit**

9️⃣ Construction des visuels – Page *Profit Details*

Slicer

- `RegionsHier`

Matrix

- Rows : `OrderTimeHier`
- Values : `Orders`, `Sales (Sum)`, `Cost (Sum)`, `Profit`, `ProfitMargin`

💡 Business view : analyse détaillée des ventes et de la rentabilité dans le temps, filtrable par région

---

🔟 Construction des visuels – Page *Product Details*

Slicer

- `Category`

Table

- Columns : `Subcategory`, `Quantity (Sum)`, `Sales (Sum)`, `ProfitMargin`

💡 Business view : identifier les **produits les plus vendus et rentables**, ou ceux à optimiser

---

✅ Résumé

- **Modèle propre et performant** grâce aux vues SQL
- **Hiérarchies temporelles** et **Display Folders** pour lisibilité
- **Mesures clés** (Orders, Profit, ProfitMargin) pour décisions métier
- **Slicers et visuels interactifs** sur les 3 pages répondant aux besoins business

---

📐 Cahier des charges fonctionnel du rapport

Le rapport Power BI doit être structuré en **trois pages complémentaires**.

🟦 Page 1 – *Overview*

🎯 Objectif

Fournir une **vue synthétique** des ventes et de la rentabilité avec possibilité de filtrer par **année** et **région**.

📊 Visuels attendus

1️⃣Sales and Profit by Time

Line and Stacked Column Chart

- Axe X : hiérarchie temporelle `OrderTimeHier`
- Colonnes : `Sales (Sum)`
- Courbe : `ProfitMargin`

👉 Permet d'analyser :

- l'évolution des ventes
- la dynamique de la marge dans le temps

2️⃣Sales by Country and Category

Stacked Column Chart

- Axe X : `Country`
- Valeur : `Sales (Sum)`
- Légende : `Category`

👉 Comparaison des ventes :

- entre pays
- par catégorie de produits

3️⃣Quantity by Category

Stacked Bar Chart

- Axe Y : `Category`
- Axe X : `Quantity (Sum)`

👉 Identification des catégories les plus vendues en volume.

🎛️ Filtres

- Slicer `OrderYear`
- Slicer hiérarchique `RegionsHier`

🟦 Page 2 – *Profit Details*

🎯 Objectif

Analyser la **rentabilité détaillée dans le temps**, avec une vision financière plus précise.

📊 Visuel principal

Matrix – Analyse temporelle

- Lignes : `OrderTimeHier`
- Valeurs :
  - `Orders`
  - `Sales (Sum)`
  - `Cost (Sum)`
  - `Profit`
  - `ProfitMargin`

📐 Mesures clés utilisées

- **Orders** : nombre de commandes distinctes
- **Profit** : différence entre ventes et coûts
- **ProfitMargin** : ratio de rentabilité

👉 Cette page est destinée :

- aux équipes financières
- au contrôle de gestion

🎛️ Filtre

- Slicer `RegionsHier`

🟦 Page 3 – *Product Details*

🎯 Objectif

Identifier les **produits performants** et ceux à **faible marge**.

📊 Visuel

Table – Analyse produit

- Colonnes :
  - `Subcategory`
  - `Quantity (Sum)`
  - `Sales (Sum)`
  - `Profit Margin`

👉 Permet de :

- comparer volume vs rentabilité
- identifier les produits à optimiser

🎛️ Filtre

- Slicer `Category`

🌐 Optionnelle : Publication sur Power BI Service

Power BI Service permet de **partager vos rapports** avec vos collègues et managers via le cloud, et d'assurer un suivi **en temps réel ou quasi temps réel** des indicateurs métier.

1️⃣ Préparer la publication

Étapes :

1. Vérifier que votre rapport Power BI Desktop est **sauvegardé**.
2. Créer un compte sur Power BI Service :
    - Il existe une **version d'essai gratuite** de 60 jours.
3. Dans Power BI Desktop :
    - Cliquer sur **Publish → My Workspace** (ou un espace de travail partagé)
4. Confirmer la publication
5. Une fois publié, le rapport est accessible depuis n'importe quel navigateur, tablette ou mobile

💡 **Business view** :

- Permet à plusieurs utilisateurs d'accéder **aux mêmes données et mesures**, garantissant **cohérence et standardisation** des analyses.

2️⃣ Exercice final : Carte géographique

Objectif :

Créer un visuel géographique pertinent pour analyser les **ventes et la rentabilité par pays**.

Étapes :

1. Dans **Power BI Desktop → Report View → New Page**
2. Insérer une **Map** ou **Filled Map** visual
3. Glisser la colonne `Country` (catégorisée en Country/Region) dans **Location**
4. Glisser `Sales (Sum)` ou `Profit (Sum)` dans **Size** ou **Color saturation**

5. Ajouter éventuellement `ProfitMargin` dans Tooltips pour enrichir la visualisation

💡 **Business rationale** :

- Permet de visualiser **rapidement les marchés les plus performants**
- Identification des régions à fort potentiel ou à faible marge
- Utile pour la **prise de décision stratégique**

3️⃣ Programmer le rafraîchissement automatique

Pourquoi ?

Si la base SQL Server est **mise à jour régulièrement**, le rapport doit pouvoir refléter **ces nouvelles données** sans intervention manuelle.

Étapes dans Power BI Service :

1. Dans Power BI Service, ouvrir le rapport publié
2. Aller dans **Datasets → Schedule Refresh**
3. Configurer :
    - Fréquence : quotidienne, hebdomadaire ou multiple fois par jour
    - Crédentiels de connexion à SQL Server
4. Activer le **refresh automatique**

💡 **Business view** :

- Les utilisateurs métier voient **toujours les données les plus récentes**
- Les décisions sont basées sur **des informations à jour**, améliorant **réactivité et performance**
- Réduit le risque d'erreur lié à des rapports obsolètes

4️⃣ Bénéfices pour l'entreprise et les utilisateurs

Bénéfice	Explication
Rapidité de décision	Les managers peuvent analyser les ventes et marges actualisées en temps réel
Cohérence	Tous les utilisateurs consultent le **même modèle et les mêmes mesures**

Collaboration	Les rapports peuvent être partagés, commentés et exportés
Gain de temps	Plus besoin de générer manuellement les rapports Power BI Desktop

---

✅ Résumé

- **Publication sur Power BI Service** rend le rapport accessible partout
- **Carte géographique** avec `Country` permet l'analyse visuelle des ventes par pays
- **Rafraîchissement automatique** garantit des données toujours à jour
- L'ensemble maximise la **valeur métier et stratégique** du projet BI

🎓 Conclusion du cours pratique Power BI – AdventureWorksDW2022

Ce cours vous a guidé étape par étape, de la **préparation des données** jusqu'à la **publication sur Power BI Service**, en suivant un projet réaliste d'entreprise.

1️⃣ Récapitulatif des étapes du projet

Étape	Actions principales	Résultat / Bénéfice
Mise en place de l'environnement	Installation SQL Server Express, VSCode / SSMS, extension SQL Server	Environnement prêt pour interroger la base
Installation de la base AdventureWorksDW2022	RESTORE de la base `.bak`	Base opérationnelle pour les vues et Power BI
Nettoyage et transformation des données	Création des vues `vwSales`, `vwProduct`, `vwRegions`	Données prêtes pour le modèle Power BI, règles métier appliquées
Import dans Power BI	Get Data → Import des vues SQL	Accès rapide et performant aux données
Modélisation	Relations, masquage des clés, hiérarchies temporelles, Display Folders, catégorisation des colonnes	Modèle propre, lisible et maintenable
Création des mesures DAX	Orders, Profit, ProfitMargin	Indicateurs clés métier et financiers

Construction des visuels	Overview, Profit Details, Product Details	Rapport interactif répondant aux besoins business
Publication (optionnelle)	Power BI Service, essai gratuit, partage	Accès cloud, collaboration et analyse en temps réel
Carte géographique	Map visual sur `Country`	Analyse visuelle des marchés par pays
Rafraîchissement automatique	Schedule Refresh dans Power BI Service	Données toujours à jour, décisions fiables

---

2️⃣ Bonnes pratiques à retenir

1. **Toujours centraliser les transformations côté SQL Server**
   - Meilleure performance et cohérence des mesures
2. **Masquer les clés techniques dans Power BI**
   - Les utilisateurs voient uniquement les champs utiles
3. **Créer des hiérarchies pour le temps et les dimensions**
   - Facilite le drill-down et la navigation dans les visuels
4. **Utiliser Display Folders**
   - Organise les colonnes par thème, améliore la lisibilité du modèle
5. **Créer des mesures plutôt que des colonnes calculées**
   - Les mesures sont dynamiques et respectent le contexte de filtre
6. **Catégoriser les colonnes pour les cartes et axes**
   - Ex : `Country` → Country/Region
7. **Publier sur Power BI Service avec rafraîchissement programmé**
   - Garantit que les rapports restent **fiables et à jour**

3️⃣ Checklist finale pour l'apprenant

Avant de partager votre rapport :

- Vérifier que toutes les vues SQL sont correctes et retournent les bons résultats
- Confirmer que les relations Power BI sont correctes (Many-to-One depuis Sales vers Product et Regions)
- Masquer toutes les clés inutiles dans Model View
- Créer la hiérarchie temporelle et organiser les Display Folders
- Créer les mesures Orders, Profit, ProfitMargin
- Construire les trois pages de rapport : Overview, Profit Details, Product Details
- Vérifier que les slicers fonctionnent correctement (OrderYear, RegionsHier, Category)
- Tester la carte géographique sur Country
- Publier sur Power BI Service et configurer le Schedule Refresh

---

4️⃣ Conclusion métier

En suivant ce projet :

- Vous êtes capable de **préparer, transformer et modéliser les données** côté SQL Server
- Vous savez construire un **modèle Power BI robuste et maintenable**
- Vous pouvez créer **des indicateurs pertinents et interactifs** pour l'entreprise
- Vous comprenez comment **diffuser des rapports fiables et à jour** sur Power BI Service

🎯 Résultat : un rapport BI professionnel, prêt pour **l'analyse stratégique et opérationnelle**, capable de répondre aux questions critiques de la direction sur **ventes, produits et rentabilité**.
