# Dax

# 📘 Guide d'Utilisation de DAX (Data Analysis Expressions)

## 📌 Introduction à DAX

**DAX (Data Analysis Expressions)** est un langage utilisé dans **Power BI, SQL Server Analysis Services (SSAS) et Excel Power Pivot** pour créer des calculs et des mesures avancées.

Il permet de **manipuler des données, créer des indicateurs clés de performance (KPIs)** et réaliser des analyses avancées.

---

## 🔹 Syntaxe de Base

DAX utilise une **syntaxe similaire à Excel** avec des fonctions avancées adaptées aux modèles de données relationnels.

- **Création d'une colonne calculée :**

```
Nom_Colonne = Table[Colonne1] + Table[Colonne2]
```

- **Création d'une mesure :**

```
Total_Ventes = SUM(Table[Ventes])
```

Les **colonnes calculées** sont **stockées dans le modèle de données**, tandis que les **mesures** sont calculées à la volée.

---

## 🔹 Fonctions DAX Essentielles

### 📊 1. Fonctions d’agrégation

- `SUM()` : Somme d'une colonne.

```
Total_Sales = SUM(Sales[Montant])
```

- `AVERAGE()` : Moyenne d'une colonne.

```
Avg_Sales = AVERAGE(Sales[Montant])
```

- `COUNT()` : Nombre d'éléments.

```
Total_Clients = COUNT(Customers[ClientID])
```

- `DISTINCTCOUNT()` : Nombre de valeurs uniques.

```
Unique_Clients = DISTINCTCOUNT(Sales[ClientID])
```

---

## 🔄 2. Fonctions conditionnelles

- `IF()` : Condition simple.

```
Remise = IF(Sales[Montant] > 1000, "Remisé", "Normal")
```

- `SWITCH()` : Alternative à plusieurs IF imbriqués.

```
Catégorie = SWITCH(Sales[Montant],
  0, "Gratuit",
  1, "Petit Montant",
  2, "Moyen Montant",
  "Autre")
```

---

## 📅 3. Fonctions temporelles (Time Intelligence)

Ces fonctions nécessitent une **table calendrier** bien structurée avec une colonne de dates.

- `TOTALYTD()` : Total cumulé depuis le début de l'année.

```
Ventes_YTD = TOTALYTD(SUM(Sales[Montant]), Dates[Date])
```

- `SAMEPERIODLASTYEAR()` : Valeur de la même période l'année précédente.

```
Ventes_LY = CALCULATE(SUM(Sales[Montant]), SAMEPERIODLASTYEAR(Dates[Date]))
```

- `DATEADD()` : Décalage temporel (ex: -1 mois, -1 an).

```
Ventes_Mois_Precedent = CALCULATE(SUM(Sales[Montant]), DATEADD(Dates[Date],
-1, MONTH))
```

---

## 🎯 4. Fonctions de filtrage et contexte

- `FILTER()` : Filtrer une table.

```
Ventes_Filtrees = FILTER(Sales, Sales[Montant] > 1000)
```

- `CALCULATE()` : Modifier le contexte de calcul.

```
Total_High_Sales = CALCULATE(SUM(Sales[Montant]), Sales[Montant] > 1000)
```

- `ALL()` : Supprimer les filtres.

```
Total_Ventes_Sans_Filtre = CALCULATE(SUM(Sales[Montant]), ALL(Sales))
```

---

## 🎨 Bonnes pratiques

✓ **Utiliser CALCULATE pour gérer le contexte des calculs.**
✓ **Créer une table calendrier** pour utiliser les fonctions temporelles.
✓ **Privilégier les mesures plutôt que les colonnes calculées** (meilleure performance).

✓ **Éviter les fonctions trop gourmandes (ex: SUMX sur de grosses tables).**
✓ **Utiliser `VAR` pour stocker des valeurs intermédiaires et optimiser les calculs.**

---

## 🚀 Exemple complet d'analyse avec DAX

```
Total_CA_Année_Precédente =
VAR Annee_Prec = MAX(Dates[Année]) - 1
RETURN
CALCULATE(SUM(Sales[Montant]), Dates[Année] = Annee_Prec)
```

✅ Cette mesure récupère le chiffre d'affaires de l'année précédente en utilisant une **variable (VAR)** pour stocker la valeur de l'année précédente et l'appliquer dans `CALCULATE()`.

---

## 🎯 Conclusion

DAX est un **puissant langage d'analyse** qui permet de **créer des mesures dynamiques, des agrégations avancées et d'analyser des données temporelles**. 🚀
En maîtrisant **les fonctions clés (SUM, CALCULATE, FILTER, TIME INTELLIGENCE)**, vous serez capable de **réaliser des dashboards efficaces et performants** dans Power BI.

---

💡 **Besoin d'un approfondissement ? Expérimentez avec DAX et testez vos formules dans Power BI ! 🎯**