

# Manuel Complet DAX (Data Analysis Expressions)

## Introduction

DAX est le langage de formules utilisé dans Power BI, Power Pivot et Analysis Services. Ce manuel exhaustif couvre toutes les fonctions et concepts nécessaires pour maîtriser DAX.

---

## Partie 1: Fondamentaux DAX

### 1.1 Syntaxe de Base

```
// Commentaire sur une ligne
/* Commentaire
   sur plusieurs lignes */

// Nommage des mesures
Nom Mesure = formule

// Référence aux colonnes
Table[Colonne]

// Référence aux mesures
[Nom Mesure]
```

### 1.2 Types de Données

Type	Description	Exemple
<b>Whole Number</b>	Entiers 64-bit	123, -456
<b>Decimal Number</b>	Virgule flottante	123.45
<b>Currency</b>	Monétaire (4 décimales)	\$123.4500
<b>Date/Time</b>	Date et heure	2024-01-15 14:30:00
<b>Boolean</b>	Vrai/Faux	TRUE, FALSE
<b>Text</b>	Chaîne de caractères	"Texte"

### 1.3 Opérateurs

```
// Arithmétiques
+, -, *, /, ^

// Comparaison
=, ==, <>, <, >, <=, >=

// Logiques
&&, ||, NOT, IN

// Concaténation
&

// Priorité: () pour grouper
```

---

## Partie 2: Colonnes Calculées vs Mesures

### 2.1 Colonne Calculée

```
// Créée dans une table, calculée ligne par ligne
// Stockée dans le modèle, consomme mémoire

Profit = Ventes[Revenu] - Ventes[Cout]

// Avec condition
Categorie Age =
IF(Clients[Age] < 30, "Jeune",
  IF(Clients[Age] < 50, "Adulte", "Senior"))

// Avec texte
Nom Complet = Clients[Prenom] & " " & Clients[Nom]
```

### 2.2 Mesure

```
// Calculée dynamiquement selon le contexte de filtre
// Pas de stockage, recalculée à chaque requête

Total Ventes = SUM(Ventes[Montant])

// Avec contexte
Ventes YTD = TOTALYTD(SUM(Ventes[Montant]), Calendrier[Date])
```

### 2.3 Quand Utiliser Quoi

Critère	Colonne Calculée	Mesure
Filtrage/Slicer	<input type="checkbox"/> Oui	<input type="checkbox"/> Non
Agrégation dynamique	<input type="checkbox"/> Non	<input type="checkbox"/> Oui
Relations	<input type="checkbox"/> Oui	<input type="checkbox"/> Non
Mémoire	Consomme	Pas de stockage
Contexte	Ligne	Filtre

## Partie 3: Fonctions d'Agrégation

### 3.1 Agrégation Simple

```
// Somme
Total = SUM(Table[Colonne])

// Moyenne
Moyenne = AVERAGE(Table[Colonne])

// Minimum / Maximum
Min = MIN(Table[Colonne])
Max = MAX(Table[Colonne])

// Comptage
```

```

Nb Lignes = COUNT(Table[Colonne])           // Ignore les vides
Nb Total = COUNTA(Table[Colonne])           // Compte non-vides
Nb Distinct = DISTINCTCOUNT(Table[Col])    // Valeurs uniques
Nb Lignes Table = COUNTROWS(Table)          // Toutes les lignes

// Sans vides
Nb Non Vides = COUNTBLANK(Table[Colonne])   // Compte les vides

```

### 3.2 Agrégation avec Itération (X-functions)

```

// SUMX - Somme avec expression ligne par ligne
Revenu Total = SUMX(
    Ventes,
    Ventes[Quantite] * Ventes[PrixUnitaire]
)

// AVERAGEX
Prix Moyen Pondere = AVERAGEX(
    Produits,
    Produits[Prix] * Produits[Volume]
)

// MINX / MAXX
Meilleur Marge = MAXX(
    Produits,
    Produits[Prix] - Produits[Cout]
)

// COUNTX
Nb Profitables = COUNTX(
    FILTER(Produits, Produits[Marge] > 0),
    Produits[ProduitID]
)

// CONCATENATEX - Concaténer des valeurs
Liste Produits = CONCATENATEX(
    Produits,
    Produits[Nom],
    ", ",
    Produits[Nom], ASC
)

```

---

## Partie 4: CALCULATE - La Fonction Centrale

### 4.1 Syntaxe

```

CALCULATE(
    <expression>,
    <filtre1>,
    <filtre2>,
    ...
)

```

## 4.2 Filtres Simples

```
// Filtre sur valeur
Ventes Premium = CALCULATE(
    SUM(Ventes[Montant]),
    Clients[Segment] = "Premium"
)

// Filtres multiples (ET logique)
Ventes Premium Paris = CALCULATE(
    SUM(Ventes[Montant]),
    Clients[Segment] = "Premium",
    Clients[Ville] = "Paris"
)

// OU logique avec || ou FILTER
Ventes Nord Sud = CALCULATE(
    SUM(Ventes[Montant]),
    Clients[Region] = "Nord" || Clients[Region] = "Sud"
)
```

## 4.3 Modificateurs de Contexte

```
// ALL - Supprime tous les filtres
Total Global = CALCULATE(
    SUM(Ventes[Montant]),
    ALL(Ventes)
)

// ALL sur colonnes spécifiques
Total Sans Filtre Date = CALCULATE(
    SUM(Ventes[Montant]),
    ALL(Calendrier[Annee], Calendrier[Mois])
)

// ALLEXCEPT - Garde certains filtres
Total Par Produit = CALCULATE(
    SUM(Ventes[Montant]),
    ALLEXCEPT(Ventes, Produits[Categorie])
)

// ALLSELECTED - Respecte les filtres du visual
% Selection = DIVIDE(
    SUM(Ventes[Montant]),
    CALCULATE(SUM(Ventes[Montant]), ALLSELECTED())
)

// REMOVEFILTERS - Supprime les filtres (synonyme de ALL)
Sans Filtres = CALCULATE(
    SUM(Ventes[Montant]),
    REMOVEFILTERS()
)

// KEEPFILTERS - Ajoute un filtre sans remplacer
```

```
Intersection = CALCULATE(
    SUM(Ventes[Montant]),
    KEEPFILTERS(Produits[Categorie] = "A")
)
```

#### 4.4 FILTER avec CALCULATE

```
// Filtre complexe
Gros Clients = CALCULATE(
    SUM(Ventes[Montant]),
    FILTER(
        Clients,
        Clients[ChiffreAffaires] > 100000
    )
)

// Filtre avec calcul
Top Vendeurs = CALCULATE(
    SUM(Ventes[Montant]),
    FILTER(
        ALL(Vendeurs),
        CALCULATE(SUM(Ventes[Montant])) > 50000
    )
)
```

---

## Partie 5: Time Intelligence

### 5.1 Table de Dates

```
// Création d'une table de dates
Calendrier =
ADDCOLUMNS(
    CALENDAR(DATE(2020,1,1), DATE(2025,12,31)),
    "Annee", YEAR([Date]),
    "Mois", MONTH([Date]),
    "NomMois", FORMAT([Date], "MMMM"),
    "MoisAnnee", FORMAT([Date], "MMM YYYY"),
    "Trimestre", "Q" & QUARTER([Date]),
    "Semaine", WEEKNUM([Date]),
    "JourSemaine", WEEKDAY([Date]),
    "NomJour", FORMAT([Date], "dddd"),
    "EstWeekend", IF(WEEKDAY([Date]) IN {1, 7}, TRUE, FALSE),
    "AnneeMois", YEAR([Date]) * 100 + MONTH([Date])
)
```

### 5.2 Fonctions To-Date

```
// Year-to-Date
YTD Ventes = TOTALYTD(
    SUM(Ventes[Montant]),
    Calendrier[Date]
)
```

```

// Quarter-to-Date
QTD Ventes = TOTALQTD(
    SUM(Ventes[Montant]),
    Calendrier[Date]
)

// Month-to-Date
MTD Ventes = TOTALMTD(
    SUM(Ventes[Montant]),
    Calendrier[Date]
)

// Année fiscale (fin juin)
YTD Fiscal = TOTALYTD(
    SUM(Ventes[Montant]),
    Calendrier[Date],
    "6/30"
)

```

### 5.3 Comparaisons Périodiques

```

// Même période année précédente
Ventes N-1 = CALCULATE(
    SUM(Ventes[Montant]),
    SAMEPERIODLASTYEAR(Calendrier[Date])
)

// Mois précédent
Ventes Mois Prec = CALCULATE(
    SUM(Ventes[Montant]),
    PREVIOUSMONTH(Calendrier[Date])
)

// Trimestre précédent
Ventes Trim Prec = CALCULATE(
    SUM(Ventes[Montant]),
    PREVIOUSQUARTER(Calendrier[Date])
)

// Année précédente
Ventes Annee Prec = CALCULATE(
    SUM(Ventes[Montant]),
    PREVIOUSYEAR(Calendrier[Date])
)

// Variation YoY
Var YoY =
VAR Actuel = SUM(Ventes[Montant])
VAR Precedent = CALCULATE(
    SUM(Ventes[Montant]),
    SAMEPERIODLASTYEAR(Calendrier[Date])
)
RETURN

```

```

DIVIDE(Actuel - Precedent, Precedent)

// Variation MoM
Var MoM =
VAR Actuel = SUM(Ventes[Montant])
VAR MoisPrec = CALCULATE(
    SUM(Ventes[Montant]),
    PREVIOUSMONTH(Calendrier[Date])
)
RETURN
DIVIDE(Actuel - MoisPrec, MoisPrec)

```

## 5.4 Dates Glissantes

```

// 12 derniers mois glissants
Ventes 12M = CALCULATE(
    SUM(Ventes[Montant]),
    DATESINPERIOD(
        Calendrier[Date],
        MAX(Calendrier[Date]),
        -12,
        MONTH
    )
)

// 30 derniers jours
Ventes 30J = CALCULATE(
    SUM(Ventes[Montant]),
    DATESINPERIOD(
        Calendrier[Date],
        MAX(Calendrier[Date]),
        -30,
        DAY
    )
)

// Entre deux dates
Ventes Periode = CALCULATE(
    SUM(Ventes[Montant]),
    DATESBETWEEN(
        Calendrier[Date],
        DATE(2024, 1, 1),
        DATE(2024, 6, 30)
    )
)

```

## 5.5 Moyennes Mobiles

```

// Moyenne mobile 3 mois
MM3 = AVERAGEX(
    DATESINPERIOD(
        Calendrier[Date],
        MAX(Calendrier[Date]),
        -3,

```

```

        MONTH
    ),
    CALCULATE(SUM(Ventes[Montant]))
)

// Moyenne mobile 7 jours
MM7J = AVERAGEX(
    DATESINPERIOD(
        Calendrier[Date],
        MAX(Calendrier[Date]),
        -7,
        DAY
    ),
    CALCULATE(SUM(Ventes[Montant]))
)

```

---

## Partie 6: Fonctions Logiques

### 6.1 IF et SWITCH

```

// IF simple
Statut = IF(Ventes[Montant] > 1000, "Gros", "Petit")

// IF imbriqué
Categorie = IF(
    Ventes[Montant] > 10000, "Premium",
    IF(Ventes[Montant] > 5000, "Standard", "Basic")
)

// SWITCH (plus lisible)
Categorie = SWITCH(
    TRUE(),
    Ventes[Montant] > 10000, "Premium",
    Ventes[Montant] > 5000, "Standard",
    "Basic"
)

// SWITCH sur valeur
Trimestre Nom = SWITCH(
    Calendrier[Trimestre],
    1, "Premier",
    2, "Deuxième",
    3, "Troisième",
    4, "Quatrième",
    "Inconnu"
)

```

### 6.2 Fonctions Logiques

```

// AND / OR
Condition = AND(A > 0, B > 0)
Condition = OR(A > 0, B > 0)

```



```

Condition = A > 0 && B > 0
Condition = A > 0 || B > 0

// NOT
Inverse = NOT(Condition)

// IFERROR - Gère les erreurs
Ratio Safe = IFERROR(DIVIDE(A, B), 0)

// COALESCE - Première valeur non vide
Valeur = COALESCE(Col1, Col2, Col3, "Défaut")

// ISBLANK
Test = IF(ISBLANK(Valeur), "Vide", "Non vide")

// TRUE / FALSE
Toujours Vrai = TRUE()

```

---

## Partie 7: Fonctions de Texte

```

// Concaténation
Nom Complet = [Prenom] & " " & [Nom]
Phrase = CONCATENATE("Bonjour ", [Nom])

// Extraction
Gauche = LEFT([Texte], 5)
Droite = RIGHT([Texte], 3)
Milieu = MID([Texte], 2, 4)

// Transformation
Majuscule = UPPER([Texte])
Minuscule = LOWER([Texte])
Proper = PROPER([Texte]) // Première lettre majuscule
Trim = TRIM([Texte]) // Supprime espaces

// Recherche
Position = SEARCH("mot", [Texte]) // Insensible à la casse
Position = FIND("mot", [Texte]) // Sensible à la casse
Contient = CONTAINSSTRING([Texte], "mot")

// Remplacement
Nouveau = SUBSTITUTE([Texte], "ancien", "nouveau")
Remplace = REPLACE([Texte], 1, 3, "xxx")

// Formatage
Texte = FORMAT([Date], "DD/MM/YYYY")
Texte = FORMAT([Montant], "#,##0.00 €")

```

---

## Partie 8: Fonctions de Table

### 8.1 FILTER

```
// Filtre une table
Gros Clients = FILTER(
    Clients,
    Clients[CA] > 100000
)

// Avec CALCULATE
Ventes Gros Clients = CALCULATE(
    SUM(Ventes[Montant]),
    FILTER(Clients, Clients[CA] > 100000)
)
```

### 8.2 ALL et Variantes

```
// ALL - Table sans filtres
Toutes Lignes = ALL(Table)

// ALL sur colonnes
Toutes Valeurs = ALL(Table[Colonne])

// ALLEXCEPT
Sauf Colonne = ALLEXCEPT(Table, Table[ColonneAGarder])

// ALLSELECTED
Selection Courante = ALLSELECTED(Table)
```

### 8.3 VALUES et DISTINCT

```
// VALUES - Valeurs distinctes avec filtres actuels
Nb Categories = COUNTROWS(VALUES(Produits[Categorie]))

// DISTINCT - Similaire mais ignore les lignes vides
Distinctes = DISTINCT(Table[Colonne])

// SELECTEDVALUE - Retourne la valeur si unique
Categorie Selectionnee = SELECTEDVALUE(Produits[Categorie], "Multiple")
```

### 8.4 SUMMARIZE et SUMMARIZECOLUMNS

```
// SUMMARIZE - Grouper et calculer
Resume = SUMMARIZE(
    Ventes,
    Ventes[Categorie],
    "Total", SUM(Ventes[Montant]),
    "Moyenne", AVERAGE(Ventes[Montant])
)

// SUMMARIZECOLUMNS - Plus performant
Resume = SUMMARIZECOLUMNS(
    Produits[Categorie],
```

```

    Calendrier[Annee],
    "Total", SUM(Ventes[Montant])
)

```

## 8.5 ADDCOLUMNS et SELECTCOLUMNS

```

// ADDCOLUMNS - Ajoute des colonnes
Enrichi = ADDCOLUMNS(
    Clients,
    "Total Ventes", CALCULATE(SUM(Ventes[Montant])),
    "Nb Commandes", CALCULATE(COUNTROWS(Ventes))
)

```

```

// SELECTCOLUMNS - Sélectionne et renomme
Selection = SELECTCOLUMNS(
    Clients,
    "ID", Clients[ClientID],
    "Nom Client", Clients[Nom]
)

```

## 8.6 TOPN et RANKING

```

// TOPN - N premières lignes
Top5 = TOPN(
    5,
    Clients,
    [Total Ventes],
    DESC
)

// Dans une mesure
Ventes Top 5 = SUMX(
    TOPN(5, VALUES(Clients[ClientID]), [Total Ventes], DESC),
    [Total Ventes]
)

// RANKX
Rang = RANKX(
    ALL(Clients),
    [Total Ventes],
    ,
    DESC,
    Dense
)

```

## 8.7 UNION, INTERSECT, EXCEPT

```

// UNION - Combine des tables
Combinee = UNION(Table1, Table2)

// INTERSECT - Lignes communes
Communes = INTERSECT(Table1, Table2)

```

```
// EXCEPT - Lignes dans la première mais pas dans la deuxième
Difference = EXCEPT(Table1, Table2)
```

---

## Partie 9: Variables (VAR/RETURN)

### 9.1 Syntaxe et Avantages

```
// Améliore lisibilité et performance
Mesure Complexe =
VAR TotalVentes = SUM(Ventes[Montant])
VAR TotalCout = SUM(Ventes[Cout])
VAR Marge = TotalVentes - TotalCout
RETURN
DIVIDE(Marge, TotalVentes)
```

### 9.2 Variables de Table

```
Top Clients =
VAR TopTable = TOPN(10, Clients, [Total Ventes], DESC)
VAR TotalTop = SUMX(TopTable, [Total Ventes])
VAR TotalGlobal = CALCULATE([Total Ventes], ALL(Clients))
RETURN
DIVIDE(TotalTop, TotalGlobal)
```

### 9.3 Debug avec Variables

```
// Pour déboguer, retourner différentes variables
Debug =
VAR A = SUM(Ventes[Montant])
VAR B = COUNTROWS(Ventes)
VAR C = DIVIDE(A, B)
RETURN
C // Changer pour voir A ou B
```

---

## Partie 10: Patterns Avancés

### 10.1 Pourcentage du Total

```
% Total = DIVIDE(
    SUM(Ventes[Montant]),
    CALCULATE(SUM(Ventes[Montant]), ALL(Ventes))
)

% Categorie = DIVIDE(
    SUM(Ventes[Montant]),
    CALCULATE(SUM(Ventes[Montant]), ALLEXCEPT(Ventes, Produits[Categorie]))
)

% Selection = DIVIDE(
    SUM(Ventes[Montant]),
```

```

    CALCULATE(SUM(Ventes[Montant]), ALLSELECTED())
)

```

## 10.2 Cumul Courant

```

Cumul = CALCULATE(
    SUM(Ventes[Montant]),
    FILTER(
        ALLSELECTED(Calendrier[Date]),
        Calendrier[Date] <= MAX(Calendrier[Date])
    )
)

```

## 10.3 Mesures Semi-Additives

```

// Solde de fin de période (pas une somme!)
Solde Fin = CALCULATE(
    SUM(Comptes[Solde]),
    LASTNONBLANK(Calendrier[Date], SUM(Comptes[Solde]))
)

// Moyenne des soldes
Solde Moyen = AVERAGEX(
    VALUES(Calendrier[Date]),
    CALCULATE(SUM(Comptes[Solde]))
)

```

## 10.4 ABC Analysis (Pareto)

```

Classe ABC =
VAR TotalGlobal = CALCULATE([Total Ventes], ALL(Produits))
VAR CumulPct =
    DIVIDE(
        CALCULATE(
            [Total Ventes],
            FILTER(
                ALL(Produits),
                [Total Ventes] >= CALCULATE([Total Ventes])
            )
        ),
        TotalGlobal
    )
RETURN
SWITCH(
    TRUE(),
    CumulPct <= 0.8, "A",
    CumulPct <= 0.95, "B",
    "C"
)

```

## 10.5 Cohortes

```

Mois Cohorte =
VAR PremierAchat = CALCULATE(

```

```

        MIN(Ventes[Date]),
        ALLEXCEPT(Ventes, Clients[ClientID])
    )
RETURN
FORMAT(PremierAchat, "YYYY-MM")

```

---

## Partie 11: Optimisation des Performances

### 11.1 Bonnes Pratiques

1. **Utiliser des variables** pour éviter les calculs répétés
2. **Éviter FILTER** quand un filtre simple suffit
3. **Préférer DISTINCTCOUNT** à COUNTROWS(DISTINCT())
4. **Utiliser SUMMARIZECOLUMNS** plutôt que SUMMARIZE
5. **Éviter les colonnes calculées** quand une mesure suffit

### 11.2 Patterns à Éviter

```

// ÉVITER: Double calcul
Mauvais = DIVIDE(
    CALCULATE(SUM(Ventes[Montant]), Catégorie = "A"),
    CALCULATE(SUM(Ventes[Montant]), ALL(Ventes))
)

// PRÉFÉRER: Variables
Bon =
VAR Numerateur = CALCULATE(SUM(Ventes[Montant]), Catégorie = "A")
VAR Denominateur = CALCULATE(SUM(Ventes[Montant]), ALL(Ventes))
RETURN DIVIDE(Numerateur, Denominateur)

```

---

## Questions d'Entretien DAX

1. **Différence entre CALCULATE et SUMX?** → CALCULATE modifie le contexte de filtre; SUMX itère sur chaque ligne
  2. **Qu'est-ce que le contexte de filtre vs contexte de ligne?** → Filtre: ensemble des filtres actifs; Ligne: ligne courante dans une itération
  3. **Comment calculer un YTD?** → TOTALYTD(SUM(), Calendrier[Date]) ou CALCULATE(SUM(), DATESYTD())
  4. **Différence entre ALL et ALLSELECTED?** → ALL supprime tous les filtres; ALLSELECTED respecte les filtres du rapport
  5. **Quand utiliser VAR?** → Pour la lisibilité et éviter de recalculer la même expression
- 

## Résumé des Fonctions Essentielles

Catégorie	Fonctions Clés
<b>Agrégation</b>	SUM, AVERAGE, COUNT, DISTINCTCOUNT
<b>Itération</b>	SUMX, AVERAGEX, COUNTX, MAXX, MINX
<b>Filtre</b>	CALCULATE, FILTER, ALL, ALLEXCEPT
<b>Time Intelligence</b>	TOTALYTD, SAMEPERIODLASTYEAR, DATESINPERIOD
<b>Logique</b>	IF, SWITCH, COALESCE, IFERROR
<b>Table</b>	SUMMARIZE, ADDCOLUMNS, TOPN, VALUES
<b>Texte</b>	CONCATENATE, FORMAT, UPPER, TRIM

---

**Rappel:** La maîtrise de DAX passe par la compréhension des contextes (filtre vs ligne) et de CALCULATE. Pratiquez régulièrement!