Test Power BI et DAX - Test 1

Sujet: Power BI, DAX et Python
Niveau: Intermédiaire
Nombre de questions: 25

Glossaire des Définitions Clés

Terme	Définition
DAX	Data Analysis Expressions - langage de formules pour Power BI, Power Pivot et Analysis Services
Mesure	Calcul dynamique évalué au moment de la requête, selon le contexte de filtre actif
Colonne calculée	Colonne dont les valeurs sont calculées à l'importation et stockées dans le modèle
Filter Context	Ensemble des filtres actifs (slicers, filtres visuels, filtres de page) qui déterminent quelles données sont incluses
Row Context	Contexte qui évalue une expression pour chaque ligne d'une table, créé par les colonnes calculées ou les itérateurs
CALCULATE	Fonction DAX la plus importante, qui évalue une expression dans un contexte de filtre modifié
Time Intelligence	Ensemble de fonctions DAX pour les calculs temporels (YTD, YoY, périodes précédentes)
ALL	Fonction qui supprime tous les filtres d'une table ou colonne
FILTER	Fonction qui retourne une table filtrée selon une condition
RLS	Row-Level Security - sécurité au niveau des lignes, restreint l'accès aux données par utilisateur

---

Questions et Réponses

Q1. Quels sont les trois composants principaux de Power BI Desktop?

R1. Power BI Desktop est structuré autour de trois composants distincts qui correspondent aux étapes du workflow d'analyse de données:

1. **Power Query (Éditeur M):** Composant ETL (Extract, Transform, Load) pour l'acquisition et la préparation des données
2. **Modèle de données:** Définition des relations entre tables, création des calculs DAX et gestion du schéma
3. **Visualisations:** Création des rapports interactifs, graphiques et tableaux de bord

---

Q2. Quelle est la différence entre une colonne calculée et une mesure en DAX?

R2. Cette distinction est fondamentale en DAX et détermine quand et comment un calcul est effectué. Le mauvais choix peut impacter significativement les performances et les résultats.

Aspect	Colonne Calculée	Mesure
Moment du calcul	À l'importation des données	À chaque interaction/requête
Stockage	Stockée en mémoire (augmente la taille)	Calculée dynamiquement
Contexte	Row context (valeur par ligne)	Filter context (agrégation)
Usage	Valeurs fixes, catégorisation	KPIs, totaux, pourcentages

Exemple colonne calculée:

```
DTI = Prets[Dette] / Prets[Revenu]
```

Exemple mesure:

```
Total Montant = SUM(Prets[Montant])
```

Q3. Expliquez le row context et le filter context.

R3. La compréhension des contextes est essentielle pour maîtriser DAX. Ces deux types de contextes déterminent comment les expressions sont évaluées et quelles données sont accessibles.

Row Context: - Référence à la ligne actuelle dans une itération - Créé automatiquement dans les colonnes calculées - Créé explicitement par CALCULATE() ou les fonctions itératives (SUMX, AVERAGEX)

Filter Context: - Ensemble des filtres actifs provenant des slicers, filtres visuels et filtres de page - Affecte les mesures en déterminant quelles lignes sont incluses - Peut être modifié par CALCULATE()

```
// Row context: calculé pour chaque ligne de la table
Colonne = Prets[Montant] * 0.1

// Filter context: agrège selon les filtres actifs dans le rapport
Mesure = SUM(Prets[Montant])
```

Q4. Comment utiliser CALCULATE() pour modifier le contexte de filtre?

R4. CALCULATE est la fonction la plus puissante et la plus importante de DAX. Elle permet d'évaluer une expression dans un contexte de filtre modifié, ce qui est essentiel pour créer des calculs conditionnels.

Syntaxe:

```
CALCULATE(<expression>, <filter1>, <filter2>, ...)
```

Exemples pratiques:

```
// Montant total des prêts du secteur agricole uniquement
Montant Agriculture =
CALCULATE(
    SUM(Prets[Montant]),
    Prets[Secteur] = "Agriculture"
)
```

```
// NPL tous secteurs (ignore le filtre secteur actuel du rapport)
NPL Total =
CALCULATE(
    [NPL Ratio],
    ALL(Prets[Secteur])
)
```

Q5. Quelle est la différence entre SUM et SUMX?

R5. SUM et SUMX sont deux fonctions d'agrégation, mais elles fonctionnent différemment. Comprendre cette différence est crucial pour les calculs impliquant plusieurs colonnes.

SUM: Agrège directement une colonne existante

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

SUMX: Itère sur chaque ligne et évalue une expression avant de sommer

```
Total = SUMX(Table, [Col1] * [Col2])
```

Règle pratique: Utiliser SUMX quand le calcul nécessite une opération entre colonnes ou une expression complexe pour chaque ligne.

Q6. Comment calculer un pourcentage du total en DAX?

R6. Le calcul du pourcentage du total est un pattern DAX fondamental. Il nécessite de comparer la valeur filtrée au total non filtré, ce qui requiert l'utilisation de ALL ou ALLSELECTED.

```
// Pourcentage du total global (ignore tous les filtres)
% du Total =
DIVIDE(
    SUM(Prets[Montant]),
    CALCULATE(
        SUM(Prets[Montant]),
        ALL(Prets)
    )
)

// Pourcentage du total sélectionné (respecte les slicers de page)
% du Total Sélectionné =
DIVIDE(
    SUM(Prets[Montant]),
    CALCULATE(
        SUM(Prets[Montant]),
        ALLSELECTED(Prets)
    )
)
```

Q7. Comment implémenter le calcul Year-to-Date (YTD)?

R7. Les calculs Year-to-Date sont essentiels pour le reporting financier. DAX offre des fonctions dédiées de Time Intelligence qui nécessitent une table de dates correctement configurée.

```
// Méthode 1: TOTALYTD (la plus simple)
Montant YTD =
```

```
TOTALYTD(
    SUM(Prets[Montant]),
    Calendrier[Date]
)

// Méthode 2: CALCULATE avec DATESYTD (plus flexible)
Montant YTD v2 =
CALCULATE(
    SUM(Prets[Montant]),
    DATESYTD(Calendrier[Date])
)
```

Prérequis: Une table de dates marquée comme “table de dates” dans Power BI.

Q8. Comment comparer avec la même période de l’année précédente?

R8. La comparaison Year-over-Year (YoY) est un indicateur clé de croissance. SAMEPERIODLASTYEAR est la fonction standard pour ce type de calcul en DAX.

```
// Valeur de l'année précédente
Montant N-1 =
CALCULATE(
    SUM(Prets[Montant]),
    SAMEPERIODLASTYEAR(Calendrier[Date])
)

// Variation YoY en valeur absolue et pourcentage
Var YoY =
VAR Actuel = SUM(Prets[Montant])
VAR Precedent = [Montant N-1]
RETURN
DIVIDE(Actuel - Precedent, Precedent)

// Formatage pour affichage
Var YoY % = FORMAT([Var YoY], "0.0%")
```

Q9. Comment créer une table de dates (calendrier)?

R9. Une table de dates complète est indispensable pour utiliser les fonctions de Time Intelligence. Elle doit couvrir toute la période de vos données sans interruption.

```
Calendrier =
ADDCOLUMNS(
    CALENDAR(DATE(2020,1,1), DATE(2025,12,31)),
    "Annee", YEAR([Date]),
    "Mois", MONTH([Date]),
    "NomMois", FORMAT([Date], "MMMM"),
    "Trimestre", "T" & QUARTER([Date]),
    "Semaine", WEEKNUM([Date]),
    "JourSemaine", WEEKDAY([Date]),
    "EstWeekend", IF(WEEKDAY([Date]) IN {1,7}, 1, 0)
)
```

Important: Après création, marquer cette table comme “Table de dates” dans les propriétés du modèle.

Q10. Comment utiliser les variables (VAR/RETURN)?

R10. Les variables DAX améliorent considérablement la lisibilité du code et les performances en stockant des valeurs intermédiaires qui ne sont calculées qu'une seule fois.

```
NPL Ratio =
VAR NPL = CALCULATE(
    SUM(Prets[Solde]),
    Prets[JoursRetard] > 90
)
VAR Total = SUM(Prets[Solde])
RETURN
DIVIDE(NPL, Total, 0)
```

Avantages des variables: - Code plus lisible et maintenable - Calcul unique (améliore les performances) - Facilite le débogage

---

Q11. Comment utiliser FILTER() vs les filtres directs dans CALCULATE()?

R11. CALCULATE accepte deux types de filtres: les filtres directs (simples) et FILTER (pour conditions complexes). Le choix impacte les performances.

```
// Filtre direct (simple) - RECOMMANDÉ pour les conditions simples
Montant Agri =
CALCULATE(
    SUM(Prets[Montant]),
    Prets[Secteur] = "Agriculture"
)

// FILTER - Pour conditions complexes impliquant des calculs
Montant Gros Prets =
CALCULATE(
    SUM(Prets[Montant]),
    FILTER(Prets, Prets[Montant] > 1000000)
)

// FILTER avec plusieurs conditions
Prets Risque =
CALCULATE(
    COUNTROWS(Prets),
    FILTER(
        Prets,
        Prets[Score] < 500 && Prets[Montant] > 500000
    )
)
```

Règle: Préférer les filtres directs quand possible (plus performants).

Q12. Comment calculer une moyenne mobile sur 3 mois?

R12. Les moyennes mobiles lissent les données pour révéler les tendances. DATESINPERIOD est la fonction clé pour définir la fenêtre temporelle.

```
// Moyenne mobile sur les 3 derniers mois
MM3 =
AVERAGEX(
    DATESINPERIOD(
        Calendrier[Date],
        LASTDATE(Calendrier[Date]),
        -3,
        MONTH
    ),
    [Montant Total Mensuel]
)

// Alternative avec CALCULATE
MM3 v2 =
CALCULATE(
    AVERAGE(Faits[Montant]),
    DATESINPERIOD(Calendrier[Date], MAX(Calendrier[Date]), -3, MONTH)
)
```

Q13. Comment créer un ranking en DAX?

R13. RANKX est la fonction pour créer des classements dynamiques. Elle est essentielle pour identifier les top/bottom performers dans les rapports.

```
// Ranking par montant (dans une table)
Rank Agence =
RANKX(
    ALL(Agences[Agence]),  -- Table sur laquelle classer
    [Total Montant],        -- Expression de classement
    ,                       -- Valeur à classer (optionnel)
    DESC,                   -- Ordre décroissant
    Dense                   -- Type de rang (pas de trous)
)

// Filtrer pour afficher seulement le Top 10
Top 10 =
IF(
    [Rank Agence] <= 10,
    [Total Montant],
    BLANK()
)
```

Q14. Comment gérer les valeurs vides (BLANK)?

R14. La gestion des valeurs BLANK est cruciale pour éviter les erreurs et garantir des calculs corrects. DAX offre plusieurs fonctions pour cela.

```
// Remplacer BLANK par 0 avec IF
Montant Safe =
IF(ISBLANK([Montant]), 0, [Montant])

// Avec COALESCE (retourne la première valeur non-BLANK)
Montant v2 = COALESCE([Montant], 0)
```

```
// Gérer les erreurs de division
Ratio Safe =
IFERROR(DIVIDE([A], [B]), 0)
```

Q15. Comment créer une mesure conditionnelle avec SWITCH?

R15. SWITCH est une alternative plus lisible à IF imbriqués pour les conditions multiples. Elle est idéale pour les catégorisations et les sélections dynamiques.

```
// Catégorisation basée sur le NPL Ratio
Catégorie Risque =
SWITCH(
    TRUE(),
    [NPL Ratio] < 0.02, "Faible",
    [NPL Ratio] < 0.05, "Modéré",
    [NPL Ratio] < 0.10, "Élevé",
    "Critique"
)

// Sélection dynamique de mesure (utile avec un paramètre)
Mesure Sélectionnée =
SWITCH(
    SELECTEDVALUE(Parametres[Mesure]),
    "NPL", [NPL Ratio],
    "Coverage", [Coverage Ratio],
    "ROE", [ROE],
    BLANK()
)
```

Q16. Comment utiliser SUMMARIZE et ADDCOLUMNS?

R16. SUMMARIZE et ADDCOLUMNS sont des fonctions de manipulation de tables. Elles sont utiles pour créer des tables virtuelles avec des agrégations ou des colonnes supplémentaires.

```
// SUMMARIZE: groupe et agrège (comme GROUP BY en SQL)
SommaireAgence =
SUMMARIZE(
    Prets,
    Agences[Agence],
    "Total", SUM(Prets[Montant]),
    "Nb Prets", COUNTROWS(Prets)
)

// ADDCOLUMNS: ajoute des colonnes calculées à une table existante
AgenceEnrichi =
ADDCOLUMNS(
    VALUES(Agences[Agence]),
    "Total", [Total Montant],
    "NPL", [NPL Ratio]
)
```

Q17. Comment intégrer Python dans Power BI?

R17. Power BI permet d'utiliser Python comme source de données et pour créer des visuels personnalisés. Cela étend considérablement les capacités analytiques.

Comme source de données: 1. Accueil > Obtenir des données > Script Python 2. Écrire le script:

```
import pandas as pd
import numpy as np

# Votre code de préparation
df = pd.read_csv('data.csv')
df['nouvelle_col'] = df['col1'] * 2

# Le DataFrame 'df' sera disponible dans Power BI
```

Comme visual Python: 1. Ajouter un visuel Python au canevas 2. Glisser les champs souhaités 3. Écrire le script de visualisation:

```
import matplotlib.pyplot as plt
import seaborn as sns

# 'dataset' contient automatiquement les données
sns.histplot(dataset['Montant'])
plt.title('Distribution des Montants')
plt.show()
```

Q18. Comment optimiser les performances DAX?

R18. L'optimisation des performances est essentielle pour les rapports avec de gros volumes de données. Voici les bonnes pratiques principales:

1. **Éviter les colonnes calculées** quand une mesure suffit (moins de stockage)
2. **Utiliser des variables** pour éviter les calculs répétés
3. **Préférer les filtres directs** à FILTER() (plus efficace)
4. **Réduire la cardinalité** des colonnes (moins de valeurs uniques)
5. **Éviter DISTINCTCOUNT** si possible (opération coûteuse)
6. **Utiliser SUMMARIZECOLUMNS** plutôt que SUMMARIZE

```
// Mauvais (FILTER sur toute la table pour condition simple)
Mauvais =
CALCULATE(SUM(T[V]), FILTER(ALL(T), T[A] = "X" && T[B] > 100))

// Bon (filtre direct + FILTER seulement si nécessaire)
Bon =
CALCULATE(SUM(T[V]), T[A] = "X", FILTER(ALL(T), T[B] > 100))
```

Q19. Comment créer des paramètres what-if?

R19. Les paramètres what-if permettent aux utilisateurs de tester différents scénarios en ajustant des valeurs via des slicers. Ils sont utiles pour les projections et analyses de sensibilité.

Création: 1. Modélisation > Nouveau paramètre 2. Configurer: Nom: Taux_Croissance, Min: 0, Max: 0.20, Incrément: 0.01, Default: 0.05

Utilisation dans une mesure:

```
Projection = [Montant Actuel] * (1 + [Taux_Croissance Value])
```

Q20. Comment créer une mesure de comparaison budget vs actual?

R20. La comparaison Budget vs Actual est un cas d'usage classique en reporting financier. Elle nécessite des mesures de variance et des indicateurs visuels.

```
// Variance en valeur absolue
Variance = [Actual] - [Budget]

// Variance en pourcentage
Variance % = DIVIDE([Variance], [Budget])

// Indicateur visuel de performance
Indicateur =
SWITCH(
    TRUE(),
    [Variance %] >= 0.05, "Au-dessus (+5%)",
    [Variance %] >= -0.05, "Dans les limites",
    "En dessous (-5%)"
)
```

Q21. Comment gérer les relations many-to-many?

R21. Les relations many-to-many sont complexes car une valeur peut correspondre à plusieurs enregistrements des deux côtés. Power BI et DAX offrent plusieurs solutions.

Options disponibles: 1. **Table pont (bridge table)** - Créer une table intermédiaire 2. **CROSSFILTER** - Activer le filtrage bidirectionnel 3. **TREATAS** - Créer une relation virtuelle

```
// Utilisation de TREATAS pour simuler une relation
Ventes Filtrées =
CALCULATE(
    SUM(Ventes[Montant]),
    TREATAS(VALUES(DimProduit[Categorie]), Ventes[Categorie])
)
```

Q22. Comment implémenter la sécurité au niveau des lignes (RLS)?

R22. La Row-Level Security (RLS) permet de restreindre l'accès aux données selon l'utilisateur connecté. C'est essentiel pour les rapports partagés avec des données sensibles.

Étapes: 1. Modélisation > Gérer les rôles 2. Créer un rôle avec une expression DAX:

```
// Filtre par agence de l'utilisateur
[Agence] = USERPRINCIPALNAME()

// Ou avec lookup dans une table d'utilisateurs
[Region] = LOOKUPVALUE(
    Utilisateurs[Region],
    Utilisateurs[Email],
    USERPRINCIPALNAME()
)
```

3. Tester avec "Afficher en tant que"
4. Publier et assigner les rôles dans le Service Power BI

Q23. Comment créer un tooltip personnalisé?

R23. Les tooltips personnalisés enrichissent l'expérience utilisateur en affichant des informations détaillées au survol d'un visuel. Ils sont créés comme des pages de rapport dédiées.

Étapes: 1. Créer une nouvelle page de rapport 2. Format > Page > Type = Tooltip 3. Définir une taille compacte (ex: 280x200 pixels) 4. Ajouter les visuels souhaités (KPIs, graphiques compacts) 5. Sur le visuel principal: Format > Tooltip > Page = votre page tooltip

---

Q24. Comment utiliser les bookmarks pour la navigation?

R24. Les bookmarks (signets) capturent l'état d'une page de rapport et permettent de créer une navigation interactive ou de basculer entre différentes vues.

Création: 1. Configurer la vue souhaitée (filtres, visibilité des visuels) 2. Vue > Signets > Ajouter 3. Nommer le bookmark de manière descriptive 4. Ajouter un bouton ou une image 5. Format > Action > Type = Signet > Sélectionner le bookmark

Cas d'usage: - Navigation entre pages ou sections - Basculer entre vue détail et vue résumé - Filtres prédéfinis (ex: "Vue Région Nord")

Q25. Créez une mesure DAX complète pour le NPL Ratio avec tendance.

R25. Cette question synthétise plusieurs concepts DAX pour créer un ensemble de mesures interconnectées permettant d'analyser le NPL Ratio avec sa tendance et un indicateur visuel.

```
// 1. NPL Ratio de base
NPL Ratio =
VAR NPL = CALCULATE(
    SUM(Prets[Solde]),
    Prets[JoursRetard] > 90
)
VAR Total = SUM(Prets[Solde])
RETURN
DIVIDE(NPL, Total, 0)

// 2. NPL Ratio du mois précédent
NPL Ratio PM =
CALCULATE(
    [NPL Ratio],
    PREVIOUSMONTH(Calendrier[Date])
)

// 3. Variation mensuelle
NPL Variation = [NPL Ratio] - [NPL Ratio PM]

// 4. Indicateur de tendance avec symboles
NPL Tendance =
VAR Variation = [NPL Variation]
RETURN
SWITCH(
    TRUE(),
    ISBLANK(Variation), "-",
    Variation > 0.005, "Hausse forte",
```

```
    Variation > 0, "Hausse legere",
    Variation > -0.005, "Baisse legere",
    "Baisse forte"
)

// 5. Code couleur pour formatage conditionnel
NPL Couleur =
SWITCH(
    TRUE(),
    [NPL Ratio] > 0.08, "#FF0000",  // Rouge - Critique
    [NPL Ratio] > 0.05, "#FFA500",  // Orange - Préoccupant
    [NPL Ratio] > 0.03, "#FFFF00",  // Jaune - Acceptable
    "#00FF00"                        // Vert - Bon
)
```

Scoring

Score	Niveau
0-10	À améliorer
11-17	Intermédiaire
18-22	Avancé
23-25	Expert