Test SQL pour Data Analyst - Test 2

Sujet: SQL Avancé pour Data Analyst
Niveau: Intermédiaire
Nombre de questions: 20

Questions et Réponses

Q1. Comment écrire une requête pour calculer le taux de conversion d'un funnel?

R1.

```sql
WITH funnel AS (
    SELECT
        COUNT(DISTINCT CASE WHEN etape = 'Demande' THEN client_id END) as demandes,
        COUNT(DISTINCT CASE WHEN etape = 'Eligible' THEN client_id END) as eligibles,
        COUNT(DISTINCT CASE WHEN etape = 'Approuve' THEN client_id END) as approuves,
        COUNT(DISTINCT CASE WHEN etape = 'Decaisse' THEN client_id END) as decaisses
    FROM demandes_prets
)
SELECT
    demandes,
    eligibles,
    ROUND(eligibles * 100.0 / demandes, 1) as taux_eligibilite,
    approuves,
    ROUND(approuves * 100.0 / eligibles, 1) as taux_approbation,
    decaisses,
    ROUND(decaisses * 100.0 / approuves, 1) as taux_decaissement,
    ROUND(decaisses * 100.0 / demandes, 1) as taux_conversion_global
FROM funnel;
```

Q2. Comment calculer les percentiles en SQL?

R2.

```sql
-- Avec PERCENTILE_CONT (interpolation)
SELECT
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY montant) as median,
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY montant) as q1,
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY montant) as q3,
    PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY montant) as p90
FROM prets;

-- Avec PERCENTILE_DISC (valeur discrète)
SELECT
    PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY montant) as median_disc
FROM prets;
```

Q3. Comment implémenter une analyse de panier (market basket)?

R3.

```sql
-- Produits souvent achetés ensemble
WITH paires AS (
    SELECT
        t1.produit as produit1,
        t2.produit as produit2,
        COUNT(DISTINCT t1.transaction_id) as nb_cooccurrences
    FROM transactions t1
    JOIN transactions t2
        ON t1.transaction_id = t2.transaction_id
        AND t1.produit < t2.produit
    GROUP BY t1.produit, t2.produit
)
SELECT *
FROM paires
ORDER BY nb_cooccurrences DESC
LIMIT 10;
```

Q4. Comment calculer le taux de croissance composé (CAGR)?

R4.

```sql
WITH bornes AS (
    SELECT
        MIN(annee) as annee_debut,
        MAX(annee) as annee_fin,
        FIRST_VALUE(montant) OVER (ORDER BY annee) as montant_debut,
        LAST_VALUE(montant) OVER (ORDER BY annee
            ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as montant_fin,
        COUNT(DISTINCT annee) - 1 as nb_periodes
    FROM ventes_annuelles
)
SELECT
    ROUND(
        (POWER(montant_fin::float / montant_debut, 1.0 / nb_periodes) - 1) * 100,
        2
    ) as cagr_pct
FROM bornes;
```

Q5. Comment identifier les clients à risque de churn?

R5.

```sql
WITH client_activite AS (
    SELECT
        client_id,
        MAX(date_tx) as derniere_tx,
        COUNT(*) as nb_tx_total,
        COUNT(*) FILTER (WHERE date_tx >= CURRENT_DATE - INTERVAL '30 days') as tx_30j,
        COUNT(*) FILTER (WHERE date_tx >= CURRENT_DATE - INTERVAL '90 days') as tx_90j
    FROM transactions
    GROUP BY client_id
)
SELECT
    client_id,
    derniere_tx,
    CURRENT_DATE - derniere_tx::date as jours_inactif,
    nb_tx_total,
    tx_30j,
    tx_90j,
    CASE
        WHEN jours_inactif > 60 THEN 'Risque Élevé'
        WHEN jours_inactif > 30 OR tx_30j = 0 THEN 'Risque Modéré'
        WHEN tx_30j < tx_90j / 3.0 THEN 'Déclin Activité'
        ELSE 'Actif'
    END as risque_churn
FROM client_activite;
```

Q6. Comment calculer une moyenne mobile exponentielle (EMA)?

R6.

```
-- Approximation avec CTE récursive
WITH RECURSIVE ema AS (
    -- Première valeur = valeur initiale
    SELECT
        date_tx,
        montant,
        montant as ema_value,
        1 as n
    FROM ventes
    ORDER BY date_tx
    LIMIT 1

    UNION ALL

    SELECT
        v.date_tx,
        v.montant,
        -- EMA =   × valeur + (1- ) × EMA_prev,   = 2/(n+1)
        (2.0 / (e.n + 1)) * v.montant + (1 - 2.0 / (e.n + 1)) * e.ema_value,
        e.n + 1
    FROM ventes v
    JOIN ema e ON v.date_tx = (SELECT MIN(date_tx) FROM ventes WHERE date_tx > e.date_tx)
)
SELECT * FROM ema;
```

Q7. Comment créer une requête pour l'analyse ABC (Pareto)?

R7.

```
WITH client_revenue AS (
    SELECT
        client_id,
        SUM(montant) as revenue_total
    FROM ventes
    GROUP BY client_id
),
classement AS (
```

```sql
    SELECT
        client_id,
        revenue_total,
        SUM(revenue_total) OVER (ORDER BY revenue_total DESC) as cumul,
        SUM(revenue_total) OVER () as total,
        ROW_NUMBER() OVER (ORDER BY revenue_total DESC) as rang,
        COUNT(*) OVER () as nb_clients
    FROM client_revenue
)
SELECT
    client_id,
    revenue_total,
    ROUND(cumul * 100.0 / total, 2) as pct_cumul,
    CASE
        WHEN cumul <= total * 0.8 THEN 'A'
        WHEN cumul <= total * 0.95 THEN 'B'
        ELSE 'C'
    END as classe_abc
FROM classement
ORDER BY revenue_total DESC;
```

Q8. Comment détecter les anomalies statistiques en SQL?

R8.

```sql
WITH stats AS (
    SELECT
        AVG(montant) as moyenne,
        STDDEV(montant) as ecart_type
    FROM transactions
)
SELECT
    t.*,
    (t.montant - s.moyenne) / s.ecart_type as z_score,
    CASE
        WHEN ABS((t.montant - s.moyenne) / s.ecart_type) > 3 THEN 'Anomalie'
        WHEN ABS((t.montant - s.moyenne) / s.ecart_type) > 2 THEN 'Suspect'
        ELSE 'Normal'
    END as statut
FROM transactions t
CROSS JOIN stats s
WHERE ABS((t.montant - s.moyenne) / s.ecart_type) > 2;
```

Q9. Comment calculer le time-to-event (survie)?

R9.

```sql
-- Temps jusqu'au premier défaut
SELECT
    client_id,
    date_premier_pret,
    date_premier_defaut,
    COALESCE(
        date_premier_defaut - date_premier_pret,
        CURRENT_DATE - date_premier_pret  -- Censuré si pas de défaut
    ) as temps_survie,
    CASE WHEN date_premier_defaut IS NOT NULL THEN 1 ELSE 0 END as evenement
FROM (
    SELECT
        client_id,
        MIN(date_octroi) as date_premier_pret,
        MIN(CASE WHEN statut = 'Defaut' THEN date_defaut END) as date_premier_defaut
    FROM prets
    GROUP BY client_id
) t;
```

Q10. Comment utiliser LATERAL JOIN?

R10.

```
-- Top 3 transactions par client
SELECT c.nom, t.*
FROM clients c
CROSS JOIN LATERAL (
    SELECT *
    FROM transactions
    WHERE client_id = c.client_id
    ORDER BY montant DESC
    LIMIT 3
) t;
```

Q11. Comment implémenter une recherche de séquences?

R11.

```
-- Clients avec 3 transactions consécutives > 100K
WITH sequenced AS (
    SELECT
        client_id,
        date_tx,
        montant,
        CASE WHEN montant > 100000 THEN 1 ELSE 0 END as flag,
        ROW_NUMBER() OVER (PARTITION BY client_id ORDER BY date_tx) -
        ROW_NUMBER() OVER (PARTITION BY client_id,
            CASE WHEN montant > 100000 THEN 1 ELSE 0 END ORDER BY date_tx) as grp
    FROM transactions
)
SELECT DISTINCT client_id
FROM sequenced
WHERE flag = 1
GROUP BY client_id, grp
HAVING COUNT(*) >= 3;
```

Q12. Comment calculer le ratio de concentration (HHI)?

R12.

```sql
-- Herfindahl-Hirschman Index par secteur
WITH sector_shares AS (
    SELECT
        secteur,
        SUM(montant) as total_secteur,
        SUM(montant) * 100.0 / SUM(SUM(montant)) OVER () as part_marche
    FROM prets
    GROUP BY secteur
)
SELECT
    SUM(POWER(part_marche, 2)) as hhi
FROM sector_shares;

-- HHI < 1500: Peu concentré
-- 1500-2500: Modérément concentré
-- > 2500: Très concentré
```

Q13. Comment créer un rapport de vieillissement (aging)?

R13.

```sql
SELECT
    CASE
        WHEN jours_retard = 0 THEN 'Current'
        WHEN jours_retard BETWEEN 1 AND 30 THEN '1-30 jours'
        WHEN jours_retard BETWEEN 31 AND 60 THEN '31-60 jours'
        WHEN jours_retard BETWEEN 61 AND 90 THEN '61-90 jours'
        ELSE '90+ jours'
    END as bucket,
    COUNT(*) as nb_prets,
    SUM(solde_restant) as encours,
    ROUND(SUM(solde_restant) * 100.0 / SUM(SUM(solde_restant)) OVER (), 2) as pct_total
FROM portefeuille_prets
GROUP BY
    CASE
        WHEN jours_retard = 0 THEN 'Current'
        WHEN jours_retard BETWEEN 1 AND 30 THEN '1-30 jours'
        WHEN jours_retard BETWEEN 31 AND 60 THEN '31-60 jours'
        WHEN jours_retard BETWEEN 61 AND 90 THEN '61-90 jours'
        ELSE '90+ jours'
    END
ORDER BY MIN(jours_retard);
```

Q14. Comment optimiser une requête avec EXPLAIN?

R14.

```sql
EXPLAIN ANALYZE
SELECT c.nom, SUM(t.montant)
FROM clients c
JOIN transactions t ON c.client_id = t.client_id
WHERE t.date_tx >= '2024-01-01'
GROUP BY c.nom;
```

```
-- Regarder:
-- - Seq Scan vs Index Scan
-- - Nested Loop vs Hash Join
-- - Actual time vs Estimated
-- - Rows returned vs estimated
```

Q15. Comment créer une vue matérialisée pour le reporting?

R15.

```
CREATE MATERIALIZED VIEW mv_kpi_mensuel AS
SELECT
    DATE_TRUNC('month', date_tx) as mois,
    COUNT(DISTINCT client_id) as clients_actifs,
    SUM(montant) as volume_total,
    COUNT(*) as nb_transactions,
    AVG(montant) as ticket_moyen
FROM transactions
GROUP BY DATE_TRUNC('month', date_tx);

-- Rafraîchir
REFRESH MATERIALIZED VIEW mv_kpi_mensuel;

-- Rafraîchir sans bloquer les lectures
REFRESH MATERIALIZED VIEW CONCURRENTLY mv_kpi_mensuel;
```

Q16-20. [Questions sur les procédures stockées, les triggers, et l'optimisation avancée...]

Scoring

Score	Niveau
0-8	À améliorer
9-13	Intermédiaire
14-17	Avancé
18-20	Expert