

AIA Bloc_02: stripe

Analyse des Uses Cases “métier” SQL et NoSQL Queries

- OLAP (Snowflake/ClickHouse) : pour les analyses structurées, agrégations, reporting et KPIs.
- NoSQL (MongoDB) : pour les données non structurées, comportement client, feedback, et explication ML.

1. Analyse de revenus

Le modèle OLAP permet de mesurer la performance financière en temps quasi réel.

```
SELECT
    d.full_date,
    dm.merchant_name,
    SUM(ft.amount) as total_revenue,
    SUM(ft.fee_amount) as total_fees
FROM fact_transaction ft
JOIN dim_date d ON ft.date_key = d.date_key
JOIN dim_merchant dm ON ft.merchant_key = dm.merchant_key
WHERE d.full_date >= DATEADD('day', -30, CURRENT_DATE())
    AND ft.status = 'succeeded'
GROUP BY d.full_date, dm.merchant_name;
```

1.1 Top 10 merchants par revenu

```
SELECT
    dm.merchant_name,
    dm.country,
```

```

SUM(ft.amount_usd) as total_revenue_usd,
COUNT(*) as transaction_count,
AVG(ft.amount_usd) as avg_transaction_usd,
COUNT(DISTINCT ft.customer_key) as unique_customers
FROM fact_transaction ft
INNER JOIN dim_merchant dm ON ft.merchant_key = dm.merchant_key AND
dm.is_current = TRUE
INNER JOIN dim_date d ON ft.date_key = d.date_key
WHERE d.year = YEAR(CURRENT_DATE())
      AND d.month = MONTH(CURRENT_DATE())
      AND ft.status = 'succeeded'
GROUP BY dm.merchant_name, dm.country
ORDER BY total_revenue_usd DESC
LIMIT 10;

```

2. Segmentation client

Grâce à la dimension `dim_customer` et aux métriques agrégées, on classe les clients par valeur et comportement.

Exemple : Analyse RFM (Récence, Fréquence, Valeur monétaire)

```

SELECT
    dc.customer_segment,
    COUNT(DISTINCT dc.customer_key) as customer_count,
    AVG(dc.lifetime_value) as avg_lifetime_value,
    SUM(dc.lifetime_transaction_count) as total_transactions,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY dc.lifetime_value) as
median_ltv,
    PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY dc.lifetime_value) as
p95_ltv
FROM dim_customer dc
WHERE dc.is_current = TRUE
GROUP BY dc.customer_segment
ORDER BY avg_lifetime_value DESC;

```

```

WITH rfm_scores AS (
    SELECT

```

```

    ft.customer_key,
    dc.customer_id,
    MAX(d.full_date) as last_transaction_date,
    DATEDIFF('day', MAX(d.full_date), CURRENT_DATE()) as recency_days,
    COUNT(*) as frequency,
    SUM(ft.amount_usd) as monetary_value,
    -- Scores RFM (1-5, 5 étant le meilleur)
    NTILE(5) OVER (ORDER BY DATEDIFF('day', MAX(d.full_date),
CURRENT_DATE()) DESC) as recency_score,
    NTILE(5) OVER (ORDER BY COUNT(*)) as frequency_score,
    NTILE(5) OVER (ORDER BY SUM(ft.amount_usd)) as monetary_score
FROM fact_transaction ft
INNER JOIN dim_customer dc ON ft.customer_key = dc.customer_key
INNER JOIN dim_date d ON ft.date_key = d.date_key
WHERE ft.status = 'succeeded'
    AND d.full_date >= DATEADD('year', -1, CURRENT_DATE())
GROUP BY ft.customer_key, dc.customer_id
)

```

```

SELECT
    customer_id,
    recency_days,
    frequency,
    ROUND(monetary_value, 2) as total_spend,
    recency_score,
    frequency_score,
    monetary_score,
    (recency_score + frequency_score + monetary_score) as rfm_total,
    CASE
        WHEN recency_score >= 4 AND frequency_score >= 4 THEN 'Champions'
        WHEN recency_score >= 3 AND frequency_score >= 3 THEN 'Loyal
Customers'
        WHEN recency_score >= 4 AND frequency_score <= 2 THEN 'Promising'
        WHEN recency_score <= 2 AND frequency_score >= 3 THEN 'At Risk'
        WHEN recency_score <= 2 AND frequency_score <= 2 THEN 'Hibernating'
        ELSE 'Regular'
    END as customer_segment
FROM rfm_scores
ORDER BY rfm_total DESC;

```

3. Détection et analyse de fraude

3.1 Transactions à haut risque nécessitant une revue

```
SELECT
    ft.transaction_id,
    d.full_date,
    t.full_time,
    dm.merchant_name,
    ft.amount,
    ft.currency_code,
    dg.country_name as customer_country,
    ffd.fraud_score,
    ffd.risk_level,
    ffd.requires_review,
    -- Indicateurs de risque
    ffd.velocity_check_failed,
    ffd.geolocation_mismatch,
    ffd.amount_anomaly
FROM fact_fraud_detection ffd
INNER JOIN fact_transaction ft ON ffd.transaction_key = ft.transaction_key
INNER JOIN dim_date d ON ffd.date_key = d.date_key
INNER JOIN dim_time t ON ffd.time_key = t.time_key
INNER JOIN dim_merchant dm ON ft.merchant_key = dm.merchant_key AND
dm.is_current = TRUE
INNER JOIN dim_geography dg ON ft.geography_key = dg.geography_key
WHERE ffd.risk_level IN ('high', 'very_high')
    AND ffd.reviewed_at IS NULL
    AND d.full_date >= DATEADD('day', -7, CURRENT_DATE())
ORDER BY ffd.fraud_score DESC, ft.amount DESC;
```

3.2 Taux de fraude par pays et méthode de paiement

```
SELECT
    dg.country_name,
    dpm.payment_type,
    COUNT(*) as total_transactions,
    SUM(CASE WHEN ffd.risk_level IN ('high', 'very_high') THEN 1 ELSE 0 END) as
high_risk_count,
    ROUND(
        SUM(CASE WHEN ffd.risk_level IN ('high', 'very_high') THEN 1 ELSE 0
END)::FLOAT
        / COUNT(*) * 100,
        2
    ) as fraud_rate_percent,
```

```

    AVG(ffd.fraud_score) as avg_fraud_score,
    SUM(CASE WHEN ft.status = 'failed' THEN ft.amount_usd ELSE 0 END) as
blocked_amount_usd
FROM fact_transaction ft
INNER JOIN fact_fraud_detection ffd ON ft.transaction_key = ffd.transaction_key
INNER JOIN dim_geography dg ON ft.geography_key = dg.geography_key
INNER JOIN dim_payment_method dpm ON ft.payment_method_key =
dpm.payment_method_key
INNER JOIN dim_date d ON ft.date_key = d.date_key
WHERE d.full_date >= DATEADD('day', -30, CURRENT_DATE())
GROUP BY dg.country_name, dpm.payment_type
HAVING COUNT(*) >= 100 -- Seuil minimum pour statistiques significatives
ORDER BY fraud_rate_percent DESC;

```

-- 3.3 Évolution du taux de fraude dans Le temps

```

SELECT
    d.full_date,
    COUNT(DISTINCT ft.transaction_id) as total_transactions,
    COUNT(DISTINCT CASE WHEN ffd.risk_level IN ('high', 'very_high') THEN
ft.transaction_id END) as high_risk_transactions,
    ROUND(
        COUNT(DISTINCT CASE WHEN ffd.risk_level IN ('high', 'very_high') THEN
ft.transaction_id END)::FLOAT
        / COUNT(DISTINCT ft.transaction_id) * 100,
        2
    ) as daily_fraud_rate,
    AVG(ffd.fraud_score) as avg_fraud_score,
    -- Moving average 7 jours
    AVG(
        COUNT(DISTINCT CASE WHEN ffd.risk_level IN ('high', 'very_high') THEN
ft.transaction_id END)::FLOAT
        / COUNT(DISTINCT ft.transaction_id) * 100
    ) OVER (ORDER BY d.full_date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as
ma_7d_fraud_rate
FROM fact_transaction ft
LEFT JOIN fact_fraud_detection ffd ON ft.transaction_key = ffd.transaction_key
INNER JOIN dim_date d ON ft.date_key = d.date_key
WHERE d.full_date >= DATEADD('day', -90, CURRENT_DATE())
GROUP BY d.full_date
ORDER BY d.full_date DESC;

```

> Permet de cibler les régions ou méthodes à haut risque.

4. Performance opérationnelle

Mesure de la qualité du service : réussite des paiements, latence, stabilité.

Exemple : Taux de réussite par marchand

```
ROUND(SUM(CASE WHEN ft.status = 'succeeded' THEN 1 ELSE 0 END)::FLOAT  
/ COUNT(*) * 100, 2) as success_rate
```

5. Rétention et churn

Analyse par cohortes pour mesurer la fidélité client sur le long terme.

Exemple : Taux de rétention mensuelle

```
-- Mois d'acquisition → comportement dans les mois suivants  
WITH customer_cohorts AS (  
    SELECT  
        ft.customer_key,  
        MIN(d.year || '-' || LPAD(d.month::VARCHAR, 2, '0')) as  
cohort_month  
    FROM fact_transaction ft  
    JOIN dim_date d ON ft.date_key = d.date_key  
    GROUP BY ft.customer_key  
)
```

6. Analyse géographique

Découpage par région, pays, zone réglementaire pour adapter les stratégies locales.

Exemple : Revenus par continent

```

SELECT
    dg.region,
    dg.continent,
    SUM(ft.amount_usd) as total_revenue_usd
FROM fact_transaction ft
JOIN dim_geography dg ON ft.geography_key = dg.geography_key
GROUP BY dg.region, dg.continent;

```

> Alignement des produits avec les marchés à fort potentiel.

7. Analyse comportementale & feedback (NoSQL)

MongoDB permet d'aller au-delà des transactions pour comprendre l'expérience client.

a. Parcours client avant achat

```

db.customer_interactions.aggregate([
  {
    $match: {
      customer_id: "cus_xyz789",
      timestamp: {
        $gte: new Date(new Date().setDate(new Date().getDate() -
7))
      }
    }
  },
  {
    $sort: { timestamp: 1 }
  },
  {
    $group: {
      _id: "$session_data.session_id",
      customer_id: { $first: "$customer_id" },
      session_start: { $min: "$timestamp" },
      session_end: { $max: "$timestamp" },
      interactions: { $push: {
        type: "$interaction_type",
        timestamp: "$timestamp",
        channel: "$channel"
      } }
    }
  }
])

```

```

    }},
    total_interactions: { $sum: 1 },
    converted: {
      $max: {
        $cond: [{ $eq: ["$interaction_type", "payment_attempt"]
}, 1, 0]
      }
    }
  },
},
{
  $project: {
    session_id: "$_id",
    session_duration_minutes: {
      $divide: [
        { $subtract: ["$session_end", "$session_start"] },
        60000
      ]
    },
    interactions: 1,
    total_interactions: 1,
    converted: 1
  }
}
]);

```

a.1 Analyse de sentiment par merchant

```

db.customer_feedback.aggregate([
  {
    $match: {
      feedback_type: { $in: ["nps", "review"] },
      timestamp: {
        $gte: new Date(new Date().setMonth(new
Date().getMonth() - 3))
      }
    }
  },
  {
    $group: {
      _id: "$merchant_id",

```

```

        total_feedback: { $sum: 1 },
        avg_rating: { $avg: "$rating" },
        positive_count: {
            $sum: {
                $cond: [
                    { $in: ["$sentiment", ["positive",
"very_positive"]] },
                    1,
                    0
                ]
            }
        },
        negative_count: {
            $sum: {
                $cond: [
                    { $in: ["$sentiment", ["negative",
"very_negative"]] },
                    1,
                    0
                ]
            }
        },
        recent_feedbacks: {
            $push: {
                rating: "$rating",
                sentiment: "$sentiment",
                text: "$text_content",
                timestamp: "$timestamp"
            }
        }
    },
    {
        $project: {
            merchant_id: "$_id",
            total_feedback: 1,
            avg_rating: { $round: ["$avg_rating", 2] },
            nps_score: {
                $subtract: [

```

```

        { $multiply: [{ $divide: ["$positive_count",
"$total_feedback"] }, 100] },
        { $multiply: [{ $divide: ["$negative_count",
"$total_feedback"] }, 100] }
      ]
    },
    recent_feedbacks: { $slice: ["$recent_feedbacks", -5] }
  },
  {
    $sort: { nps_score: -1 }
  }
]);

```

> Cartographie les frictions dans le tunnel de conversion.

b. Sentiment client par marchand

```

db.customer_feedback.aggregate([
  { $group: {
    _id: "$merchant_id",
    avg_rating: { $avg: "$rating" },
    nps_score: { ... }
  }}
]);

```

> Mesure de la satisfaction client et détection des problèmes qualité.

c. Explication des décisions ML

```

db.fraud_signals.aggregate([
  { $unwind: "$model_explainability.top_features" },
  { $group: {
    _id: "$feature_name",
    avg_importance: { $avg: "$importance" }
  }}
]);

```

```
])
```

> Transparence du modèle de fraude : quelles features déclenchent les alertes ?

Top Features contribuant au score de fraude

```
db.fraud_signals.aggregate([
  {
    $match: {
      risk_level: { $in: ["high", "very_high"] },
      timestamp: {
        $gte: new Date(new Date().setDate(new
Date().getDate() - 30))
      }
    }
  },
  {
    $unwind: "$model_explainability.top_features"
  },
  {
    $group: {
      _id: "$model_explainability.top_features.feature_name",
      avg_importance: { $avg:
"$model_explainability.top_features.importance" },
      avg_contribution: { $avg:
"$model_explainability.top_features.contribution" },
      frequency: { $sum: 1 }
    }
  },
  {
    $sort: { avg_importance: -1 }
  },
  {
    $limit: 10
  }
]);

// 7.4 Profil 360° client enrichi
```

```

db.customer_360.findOne(
  { customer_id: "cus_xyz789" },
  {
    profile: 1,
    transaction_summary: 1,
    segmentation: 1,
    "merchants_interacted": { $slice: -5 }
  }
);

```

Profil 360° client

```

db.customer_360.findOne(
  { customer_id: "cus_xyz789" },
  { profile: 1, transaction_summary: 1, segmentation: 1 }
)

```

> Vue unifiée pour le support, le marketing ou l'ingénierie produit.

ANALYSE DE FRAUDE (fraud_signals)

```

db.fraud_signals.aggregate([
  {
    $match: {
      timestamp: {
        $gte: new Date(new Date().setDate(new Date().getDate() -
1)) // Dernières 24h
      },
      risk_level: { $in: ["high", "very_high"] },
      "decision.action": { $in: ["review", "decline"] }
    }
  },
  {
    $lookup: {
      from: "transaction_logs",

```

```

        localField: "transaction_id",
        foreignField: "transaction_id",
        as: "transaction_details"
    }
},
{
    $unwind: "$transaction_details"
},
{
    $project: {
        transaction_id: 1,
        timestamp: 1,
        fraud_score: 1,
        risk_level: 1,
        merchant_id: "$transaction_details.merchant_id",
        amount: "$transaction_details.amount",
        currency: "$transaction_details.currency",
        country:
"$transaction_details.request_metadata.country_code",
        top_risk_indicators: {
            $slice: [
                {
                    $sortBy: {
                        input: "$risk_indicators",
                        sortBy: { confidence: -1 }
                    }
                },
                3 // Top 3 indicateurs
            ]
        },
        decision: 1
    }
},
{
    $sort: { fraud_score: -1, timestamp: -1 }
},
{

```

```
    $limit: 50
  }
});
```

Analyse patterns fraude par pays et heure

```
db.fraud_signals.aggregate([
  {
    $match: {
      timestamp: { $gte: new Date(new Date().setMonth(new
Date().getMonth() - 1)) },
      risk_level: { $in: ["high", "very_high"] }
    }
  },
  {
    $lookup: {
      from: "transaction_logs",
      localField: "transaction_id",
      foreignField: "transaction_id",
      as: "txn"
    }
  },
  {
    $unwind: "$txn"
  },
  {
    $group: {
      _id: {
        country: "$txn.request_metadata.country_code",
        hour: { $hour: "$timestamp" }
      },
      fraud_count: { $sum: 1 },
      avg_fraud_score: { $avg: "$fraud_score" },
      total_amount: { $sum: "$txn.amount" },
      unique_merchants: { $addToSet: "$txn.merchant_id" }
    }
  },
]);
```

```

{
  $project: {
    country: "$_id.country",
    hour: "$_id.hour",
    fraud_count: 1,
    avg_fraud_score: { $round: ["$avg_fraud_score", 3] },
    total_amount: { $round: ["$total_amount", 2] },
    merchant_count: { $size: "$unique_merchants" }
  },
  {
    $sort: { fraud_count: -1 }
  },
  {
    $limit: 20
  }
});

```

ANALYSE LOGS TRANSACTIONS (transaction_logs)

```

db.transaction_logs.aggregate([
  {
    $match: {
      timestamp: { $gte: new Date(new Date().setDate(new Date().getDate() - 7)) }
    },
    event_type: { $in: ["captured", "failed"] }
  },
  {
    $group: {
      _id: {
        merchant_id: "$merchant_id",
        country: "$request_metadata.country_code"
      },
      total_attempts: { $sum: 1 },
      successful: {
        $sum: { $cond: [{ $eq: ["$status", "succeeded"] }, 1, 0] }
      },
      failed: {
        $sum: { $cond: [{ $eq: ["$status", "failed"] }, 1, 0] }
      }
    }
  }
]);

```

```

    },
    avg_processing_time: { $avg: "$response_metadata.processing_time_ms" },
    total_amount: { $sum: "$amount" }
  }
},
{
  $project: {
    merchant_id: "$_id.merchant_id",
    country: "$_id.country",
    total_attempts: 1,
    successful: 1,
    failed: 1,
    success_rate: {
      $round: [
        { $multiply: [{ $divide: ["$successful", "$total_attempts"] }, 100]
      ],
      2
    },
    avg_processing_time: { $round: ["$avg_processing_time", 0] },
    total_amount: { $round: ["$total_amount", 2] }
  }
},
{
  $match: {
    total_attempts: { $gte: 50 } // Minimum 50 transactions pour stats
    significatives
  }
},
{
  $sort: { success_rate: 1, total_attempts: -1 } // Pires success rates
  first
},
{
  $limit: 20
}
]);

```

ANALYSE FEEDBACK CLIENT (customer_feedback)

```

db.customer_feedback.aggregate([
  {

```

```

    $match: {
      timestamp: { $gte: new Date(new Date().setMonth(new
Date().getMonth() - 3)) },
      feedback_type: { $in: ["nps", "review"] }
    }
  },
  {
    $group: {
      _id: {
        merchant_id: "$merchant_id",
        year_month: {
          $dateToString: { format: "%Y-%m", date: "$timestamp" }
        }
      },
      total_feedback: { $sum: 1 },
      avg_rating: { $avg: "$rating" },
      sentiment_breakdown: {
        $push: "$sentiment"
      },
      keywords: { $push: "$text_analysis.keywords" }
    }
  },
  {
    $addFields: {
      positive_count: {
        $size: {
          $filter: {
            input: "$sentiment_breakdown",
            as: "s",
            cond: { $in: ["$$s", ["positive", "very_positive"]] }
          }
        }
      },
      negative_count: {
        $size: {
          $filter: {
            input: "$sentiment_breakdown",

```

```

    as: "S",
    cond: { $in: ["$$s", ["negative", "very_negative"]] }
  }
}
},
{
  $project: {
    merchant_id: "$_id.merchant_id",
    year_month: "$_id.year_month",
    total_feedback: 1,
    avg_rating: { $round: ["$avg_rating", 2] },
    positive_rate: {
      $round: [
        { $multiply: [{ $divide: ["$positive_count",
"$total_feedback"] }, 100] },
        1
      ]
    },
    negative_rate: {
      $round: [
        { $multiply: [{ $divide: ["$negative_count",
"$total_feedback"] }, 100] },
        1
      ]
    },
    // NPS score (promoteurs - détracteurs)
    nps_score: {
      $round: [
        {
          $subtract: [
            { $multiply: [{ $divide: ["$positive_count",
"$total_feedback"] }, 100] },
            { $multiply: [{ $divide: ["$negative_count",
"$total_feedback"] }, 100] }
          ]
        }
      ]
    }
  }
}

```

```
    },  
    0  
  ]  
}  
},  
{  
  $sort: { year_month: -1, nps_score: -1 }  
}  
]);
```