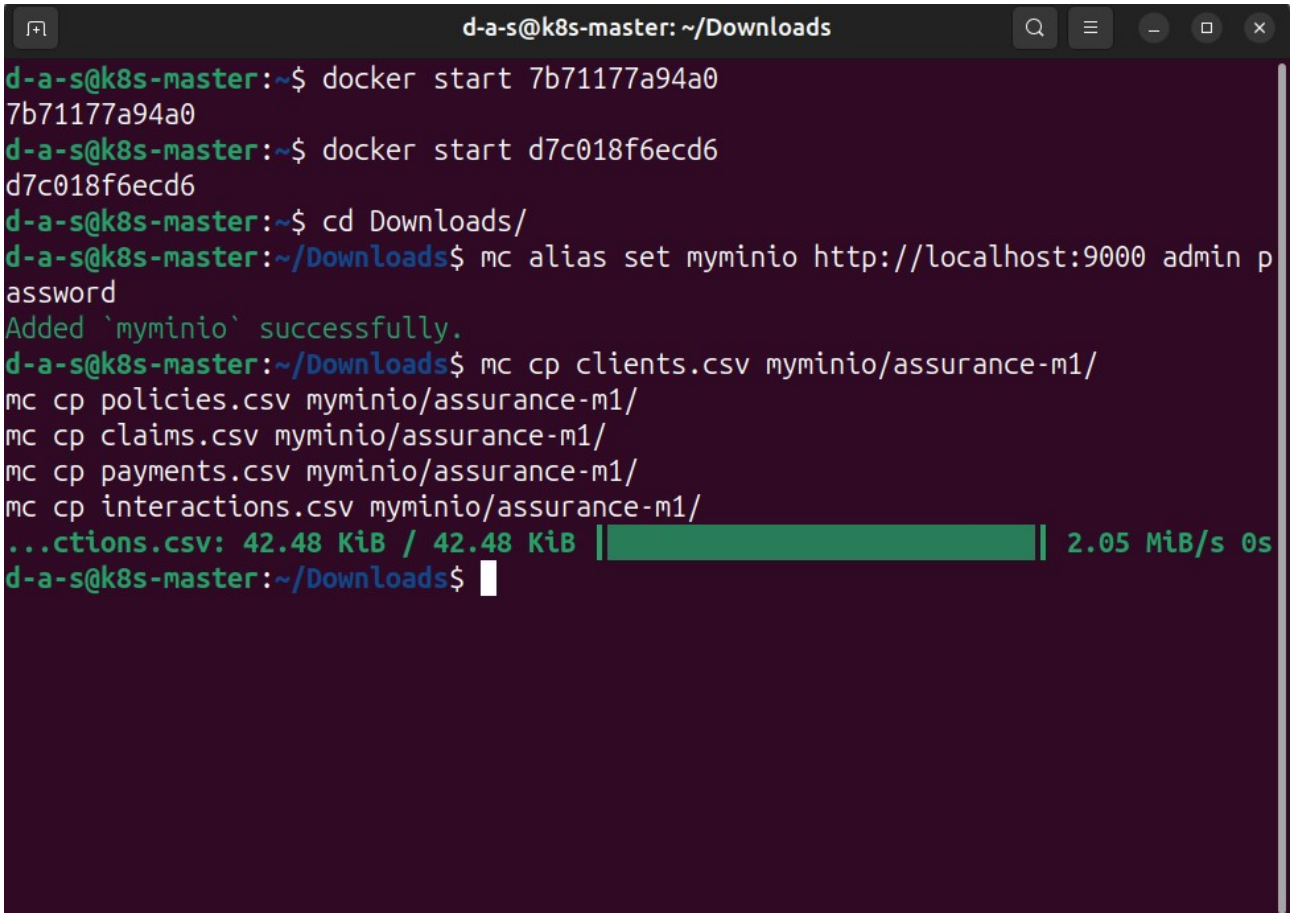
1. Ingestion des données dans MinIO

Création du bucket et upload des fichiers

bash

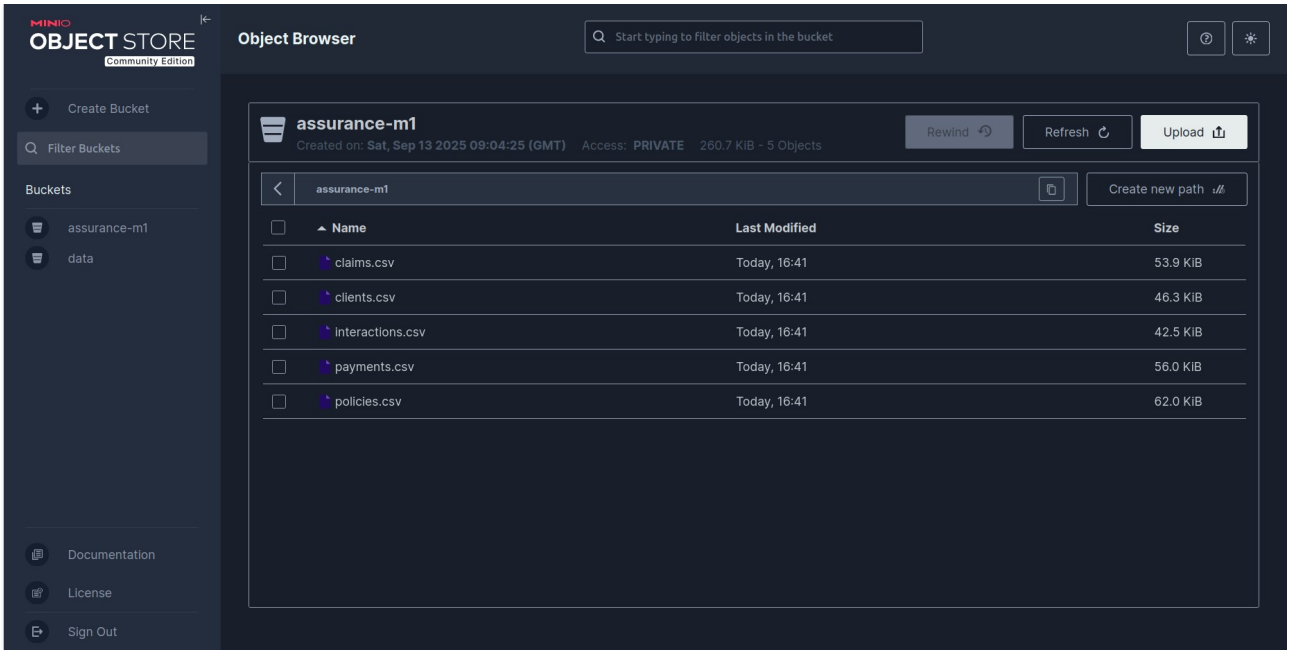
```
# Configuration de l'accès à MinIO
mc alias set myminio http://localhost:9000 admin password

# Création du bucket
mc mb myminio/assurance-ml

# Upload des fichiers CSV
mc cp clients.csv myminio/assurance-m1/
mc cp policies.csv myminio/assurance-m1/
mc cp claims.csv myminio/assurance-m1/
mc cp payments.csv myminio/assurance-m1/
mc cp interactions.csv myminio/assurance-m1/

# Vérification
mc ls myminio/assurance-m1/
```

Livrable : Captures d'écran montrant la création du bucket et l'upload des fichiers.



Upload des fichiers vers minio

Upload des fichiers vers minio



Verifications que les fichiers sont bien dans le bucket sur minio

2. Nettoyage et Transformation dans Dremio

Étape A - Couche Bronze

1. **Connexion à MinIO depuis Dremio**
    - Aller dans "Sources" → "Add Source"
    - Sélectionner "Amazon S3"
    - Configurer avec:
        - Name: `minio`
        - AWS Access Key: admin
        - AWS Access Secret: password

        Connection Properties:
        - `fs.s3a.endpoint` → Value: storage`:9000`
        - `fs.s3a.path.style.access` → `true`
        - `dremio.s3.compat` → `true`
2. **Promotion des datasets**
    - Naviguer vers le bucket `assurance-m1`
    - Pour chaque fichier CSV: clic droit → "Promote Dataset"
    - Vérifier les paramètres (séparateur ",", en-têtes, encodage UTF-8)

Étape B - Couche Silver (Nettoyage)

Scripts SQL pour chaque table:

sql

```
-- 1. Table Clients
SELECT CONVERT_TO_INTEGER(customer_id, 1, 1, 0) AS customer_id, name,
birth_date, email, phone, city
FROM (
SELECT
TRIM(customer_id) AS customer_id,
INITCAP(TRIM(name)) AS name,
CASE
WHEN birth_date LIKE '%/%' THEN TO_DATE(birth_date, 'DD/MM/YYYY')
WHEN birth_date LIKE '%-%' THEN TO_DATE(birth_date, 'YYYY-MM-DD')
ELSE CAST(birth_date AS DATE)
END AS birth_date,
LOWER(TRIM(email)) AS email,
REGEXP_REPLACE(TRIM(phone), '[^0-9+]', '') AS phone,
INITCAP(TRIM(city)) AS city
FROM minio."assurance-m1"."clients.csv"
WHERE customer_id IS NOT NULL
) nested_0


-- 2. Table Contrats (Policies)

SELECT CONVERT_TO_INTEGER(policy_id, 1, 1, 0) AS policy_id,
CONVERT_TO_INTEGER(CASE WHEN length(substr(customer_id, 1,
length(customer_id) - 2)) > 0 THEN substr(customer_id, 1,
length(customer_id) - 2) ELSE NULL END, 1, 1, 0) AS customer_id,
policy_type, annual_premium_xof, start_date, end_date, sales_channel,
status, original_currency
FROM (
SELECT
TRIM(policy_id) AS policy_id,
TRIM(customer_id) AS customer_id,
CASE
WHEN LOWER(TRIM(policy_type)) IN ('auto', 'car', 'vehicle') THEN 'Auto'
WHEN LOWER(TRIM(policy_type)) IN ('home', 'house', 'habitation') THEN
'Habitation'
WHEN LOWER(TRIM(policy_type)) IN ('health', 'sante', 'medical') THEN 'Santé'
WHEN LOWER(TRIM(policy_type)) IN ('life', 'vie') THEN 'Vie'
ELSE INITCAP(TRIM(policy_type))
END AS policy_type,
CASE
WHEN CAST(annual_premium AS DECIMAL(10,2)) <= 0 THEN NULL
WHEN UPPER(TRIM(currency)) = 'EUR' THEN CAST(annual_premium AS
DECIMAL(10,2)) * 655.957
WHEN UPPER(TRIM(currency)) = 'USD' THEN CAST(annual_premium AS
DECIMAL(10,2)) * 600
ELSE CAST(annual_premium AS DECIMAL(10,2))
```

```
END AS annual_premium_xof,
CASE
WHEN start_date LIKE '%/%' THEN TO_DATE(start_date, 'DD/MM/YYYY')
WHEN start_date LIKE '%-%' THEN TO_DATE(start_date, 'YYYY-MM-DD')
ELSE CAST(start_date AS DATE)
END AS start_date,
CASE
WHEN end_date LIKE '%/%' THEN TO_DATE(end_date, 'DD/MM/YYYY')
WHEN end_date LIKE '%-%' THEN TO_DATE(end_date, 'YYYY-MM-DD')
ELSE CAST(end_date AS DATE)
END AS end_date,
CASE
WHEN LOWER(TRIM(sales_channel)) IN ('agency', 'agence') THEN 'Agence'
WHEN LOWER(TRIM(sales_channel)) IN ('broker', 'courtier') THEN 'Courtier'
WHEN LOWER(TRIM(sales_channel)) IN ('online', 'web', 'internet') THEN 'En
ligne'
WHEN LOWER(TRIM(sales_channel)) IN ('phone', 'telephone') THEN 'Téléphone'
ELSE INITCAP(TRIM(sales_channel))
END AS sales_channel,
CASE
WHEN LOWER(TRIM(status)) IN ('active', 'actif') THEN 'Actif'
WHEN LOWER(TRIM(status)) IN ('cancelled', 'annule') THEN 'Annulé'
WHEN LOWER(TRIM(status)) IN ('expired', 'expire') THEN 'Expiré'
WHEN LOWER(TRIM(status)) IN ('pending', 'en_attente') THEN 'En attente'
ELSE INITCAP(TRIM(status))
END AS status,
UPPER(TRIM(currency)) AS original_currency
FROM minio."assurance-m1"."policies.csv"
WHERE policy_id IS NOT NULL AND customer_id IS NOT NULL
) nested_0


-- 3. Table Sinistres (Claims)

SELECT CONVERT_TO_INTEGER(claim_id, 1, 1, 0) AS claim_id,
CONVERT_TO_INTEGER(policy_id, 1, 1, 0) AS policy_id, estimated_amount_xof,
amount_paid_xof, claim_date, cause, claim_status, reported_by,
original_currency
FROM (
SELECT
TRIM(claim_id) AS claim_id,
TRIM(policy_id) AS policy_id,
CASE
WHEN estimated_amount IS NULL OR TRIM(estimated_amount) = '' THEN NULL
ELSE
CASE
WHEN CAST(estimated_amount AS DECIMAL(10,2)) <= 0 THEN NULL
WHEN UPPER(TRIM(currency)) = 'EUR' THEN CAST(estimated_amount AS
DECIMAL(10,2)) * 655.957
WHEN UPPER(TRIM(currency)) = 'USD' THEN CAST(estimated_amount AS
DECIMAL(10,2)) * 600
```

```
WHEN UPPER(TRIM(currency)) = 'XOF' THEN CAST(estimated_amount AS
DECIMAL(10,2))
ELSE CAST(estimated_amount AS DECIMAL(10,2))
END
END AS estimated_amount_xof,
CASE
WHEN amount_paid IS NULL OR TRIM(amount_paid) = '' THEN NULL
ELSE
CASE
WHEN CAST(amount_paid AS DECIMAL(10,2)) <= 0 THEN NULL
WHEN UPPER(TRIM(currency)) = 'EUR' THEN CAST(amount_paid AS DECIMAL(10,2)) *
655.957
WHEN UPPER(TRIM(currency)) = 'USD' THEN CAST(amount_paid AS DECIMAL(10,2)) *
600
WHEN UPPER(TRIM(currency)) = 'XOF' THEN CAST(amount_paid AS DECIMAL(10,2))
ELSE CAST(amount_paid AS DECIMAL(10,2))
END
END AS amount_paid_xof,
CASE
WHEN claim_date LIKE '%/%' THEN TO_DATE(claim_date, 'DD/MM/YYYY')
WHEN claim_date LIKE '%-%' THEN TO_DATE(claim_date, 'YYYY-MM-DD')
ELSE CAST(claim_date AS DATE)
END AS claim_date,
INITCAP(TRIM(cause)) AS cause,
CASE
WHEN LOWER(TRIM(claim_status)) IN ('open', 'ouvert', 'ouverte') THEN
'Ouvert'
WHEN LOWER(TRIM(claim_status)) IN ('closed', 'ferme', 'fermé') THEN 'Fermé'
WHEN LOWER(TRIM(claim_status)) IN ('rejected', 'rejete', 'rejeté') THEN
'Rejeté'
ELSE INITCAP(TRIM(claim_status))
END AS claim_status,
CASE
WHEN LOWER(TRIM(reported_by)) IN ('tiers') THEN 'Tiers'
WHEN LOWER(TRIM(reported_by)) IN ('assure', 'assuré') THEN 'Assuré'
ELSE INITCAP(TRIM(reported_by))
END AS reported_by,
UPPER(TRIM(currency)) AS original_currency
FROM minio."assurance-m1"."claims.csv"
WHERE claim_id IS NOT NULL AND TRIM(claim_id) != ''
AND policy_id IS NOT NULL AND TRIM(policy_id) != ''
) nested_0

-- 4. Table Paiements (Payments)

SELECT CONVERT_TO_INTEGER(payment_id, 1, 1, 0) AS payment_id, CASE WHEN
length(substr(policy_id, 1, 5)) > 0 THEN substr(policy_id, 1, 5) ELSE NULL
END AS policy_id, reference, amount_xof, payment_date, nested_0."method" AS
"method", original_currency
FROM (
```

```sql
SELECT
TRIM(payment_id) AS payment_id,
TRIM(policy_id) AS policy_id,
TRIM(reference) AS reference,
CASE
WHEN CAST(amount AS DECIMAL(10,2)) <= 0 THEN NULL
WHEN UPPER(TRIM(currency)) = 'EUR' THEN CAST(amount AS DECIMAL(10,2)) *
655.957
WHEN UPPER(TRIM(currency)) = 'USD' THEN CAST(amount AS DECIMAL(10,2)) * 600
ELSE CAST(amount AS DECIMAL(10,2))
END AS amount_xof,
CASE
WHEN payment_date LIKE '%/%' THEN TO_DATE(payment_date, 'DD/MM/YYYY')
WHEN payment_date LIKE '%-%' THEN TO_DATE(payment_date, 'YYYY-MM-DD')
ELSE CAST(payment_date AS DATE)
END AS payment_date,
CASE
WHEN LOWER(TRIM("method")) IN ('card', 'credit_card', 'carte') THEN 'Carte'
WHEN LOWER(TRIM("method")) IN ('transfer', 'virement') THEN 'Virement'
WHEN LOWER(TRIM("method")) IN ('check', 'cheque', 'chèque') THEN 'Chèque'
WHEN LOWER(TRIM("method")) IN ('cash', 'especes') THEN 'Espèces'
ELSE INITCAP(TRIM("method"))
END AS "method",
UPPER(TRIM(currency)) AS original_currency
FROM (
SELECT
payment_id,
policy_id,
reference,
amount,
currency,
"method",
payment_date,
ROW_NUMBER() OVER (PARTITION BY TRIM(payment_id) ORDER BY payment_date DESC)
as rn
FROM minio."assurance-m1"."payments.csv"
WHERE payment_id IS NOT NULL AND policy_id IS NOT NULL
)
WHERE rn = 1
) nested_0

-- 5. Table Interactions

SELECT CONVERT_TO_INTEGER(interaction_id, 1, 1, 0) AS interaction_id, CASE
WHEN length(substr(customer_id, 1, 3)) > 0 THEN substr(customer_id, 1, 3)
ELSE NULL END AS customer_id, CASE WHEN length(substr(policy_id, 1, 5)) > 0
THEN substr(policy_id, 1, 5) ELSE NULL END AS policy_id, agent, date_time,
duration_min, intent, satisfaction
FROM (
SELECT
```

```
-- Colonnes texte
TRIM("interaction_id") AS interaction_id,
TRIM("customer_id") AS customer_id,
TRIM("policy_id") AS policy_id,
INITCAP(TRIM("agent")) AS agent,
-- Gestion des dates (formats CSV : 'DD/MM/YYYY HH:MI' et 'YYYY-MM-DD
HH:MI:SS')
CASE
WHEN TRIM("date_time") LIKE '__/__/____ __:__' THEN
TO_TIMESTAMP(TRIM("date_time"), 'DD/MM/YYYY HH24:MI')
WHEN TRIM("date_time") LIKE '____-__-__ __:__:__' THEN
TO_TIMESTAMP(TRIM("date_time"), 'YYYY-MM-DD HH24:MI:SS')
ELSE NULL
END AS date_time,
-- Duration en minutes
CASE
WHEN REGEXP_LIKE(REPLACE(TRIM("duration_min"), ',', '.'), '^[0-9]+(\.[0-
9]+)?$') THEN
ROUND(GREATEST(0, CAST(REPLACE(TRIM("duration_min"), ',', '.') AS DOUBLE)))
ELSE NULL
END AS duration_min,
-- Intent texte
INITCAP(TRIM("intent")) AS intent,
-- Satisfaction entre 1 et 5
CASE
WHEN REGEXP_LIKE(REPLACE(TRIM("satisfaction"), ',', '.'), '^[0-9]+(\.[0-
9]+)?$') THEN
ROUND(LEAST(5, GREATEST(1, CAST(REPLACE(TRIM("satisfaction"), ',', '.') AS
DOUBLE))))
ELSE NULL
END AS satisfaction
FROM minio."assurance-m1"."interactions.csv"
WHERE "interaction_id" IS NOT NULL
AND "customer_id" IS NOT NULL
) nested_0
```

Étape C - Couche Gold (Normalisation)

1. Création des tables de dimensions

sql

```
-- Dimension dim_customers

SELECT
customer_id,
name,
birth_date,
email,
phone,
city,
```

```sql
FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365.25) AS age,
CASE
WHEN FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365.25) < 25 THEN 'Jeune
(<25)'
WHEN FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365.25) BETWEEN 25 AND 40
THEN 'Adulte (25-40)'
WHEN FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365.25) BETWEEN 41 AND 60
THEN 'Mûr (41-60)'
ELSE 'Senior (>60)'
END AS age_group
FROM Silver.clients


-- Dimension Contrats

-- Dimension Contrats (version corrigée avec critères de départage
supplémentaires)
SELECT
policy_id,
customer_id,
policy_type,
annual_premium_xof,
start_date,
end_date,
sales_channel,
status,
CASE
WHEN DATEDIFF(end_date, start_date) < 365 THEN 'Court terme'
WHEN DATEDIFF(end_date, start_date) BETWEEN 365 AND 1095 THEN 'Moyen terme'
ELSE 'Long terme'
END AS duration_category
FROM (
SELECT
*,
-- Ajouter plusieurs critères de tri pour garantir l'unicité
ROW_NUMBER() OVER (
PARTITION BY policy_id
ORDER BY start_date DESC, end_date DESC, annual_premium_xof DESC,
customer_id
) as version_rank
FROM Silver.policies
WHERE policy_id IS NOT NULL
)
WHERE version_rank = 1;



-- Identifier les policy_id qui ont toujours des doublons
--SELECT policy_id, COUNT(*) as nb_doublons
--FROM gold.dim_policies
--GROUP BY policy_id
```

```sql
--HAVING COUNT(*) > 1;

--SELECT *
--FROM gold.dim_policies
--WHERE policy_id = '10842';



-- Dimension Dates

SELECT
date_actual,
EXTRACT(YEAR FROM date_actual) AS "year",
EXTRACT(QUARTER FROM date_actual) AS "quarter",
EXTRACT(MONTH FROM date_actual) AS "month",
EXTRACT(DAY FROM date_actual) AS "day",
TO_CHAR(date_actual, 'Day') AS day_name,
TO_CHAR(date_actual, 'Month') AS month_name,
CASE
WHEN TO_CHAR(date_actual, 'Dy') IN ('Sat', 'Sun') THEN 'Weekend'
ELSE 'Weekday'
END AS day_type
FROM (
SELECT DISTINCT date_actual
FROM (
SELECT start_date AS date_actual FROM silver.policies
UNION SELECT end_date FROM silver.policies
UNION SELECT claim_date FROM silver.claims
UNION SELECT payment_date FROM silver.payments
UNION SELECT CAST(date_time AS DATE) FROM silver.interactions
)
WHERE date_actual IS NOT NULL
)


-- Dimension Causes de sinistres

SELECT
DISTINCT cause,
CASE
WHEN LOWER(cause) LIKE '%accident%' THEN 'Accident'
WHEN LOWER(cause) LIKE '%vol%' OR LOWER(cause) LIKE '%theft%' THEN 'Vol'
WHEN LOWER(cause) LIKE '%fire%' OR LOWER(cause) LIKE '%feu%' THEN 'Incendie'
WHEN LOWER(cause) LIKE '%water%' OR LOWER(cause) LIKE '%eau%' THEN 'Dégât
des eaux'
WHEN LOWER(cause) LIKE '%storm%' OR LOWER(cause) LIKE '%tempête%' THEN
'Tempête'
WHEN LOWER(cause) LIKE '%health%' OR LOWER(cause) LIKE '%medical%' THEN
'Problème de santé'
ELSE 'Autre'
END AS cause_category
FROM Silver.claims
```

```sql
WHERE cause IS NOT NULL
```

2. Création des tables de faits

sql

```sql
-- Faits Paiements

SELECT payment_id, CONVERT_TO_INTEGER(policy_id, 1, 1, 0) AS policy_id,
customer_id, amount_xof, payment_date, nested_0."method" AS "method",
policy_type, sales_channel
FROM (
-- Faits Paiements
SELECT
p.payment_id,
p.policy_id,
pol.customer_id,
p.amount_xof,
p.payment_date,
p."method",
pol.policy_type,
pol.sales_channel
FROM Silver.payments p
JOIN Silver.policies pol ON p.policy_id = pol.policy_id
) nested_0


-- Faits Sinistres

-- Faits Sinistres
SELECT
c.claim_id,
c.policy_id,
pol.customer_id,
c.estimated_amount_xof,
c.amount_paid_xof,
c.claim_date,
c.cause,
c.claim_status,
c.reported_by,
pol.policy_type,
pol.sales_channel,
CASE
WHEN c.amount_paid_xof > pol.annual_premium_xof * 2 THEN 'High'
WHEN c.amount_paid_xof > pol.annual_premium_xof THEN 'Medium'
ELSE 'Low'
END AS severity
FROM Silver.claims c
JOIN Silver.policies pol ON c.policy_id = pol.policy_id


-- Faits Interactions
```

```sql
SELECT interaction_id, CAST(SPLIT_PART(customer_id, '.', 1) AS INTEGER) AS
customer_id, policy_id, agent, date_time, duration_min, intent,
satisfaction, policy_type, sales_channel, intent_category
FROM (
-- Faits Interactions
SELECT
i.interaction_id,
i.customer_id,
i.policy_id,
i.agent,
i.date_time,
i.duration_min,
i.intent,
i.satisfaction,
pol.policy_type,
pol.sales_channel,
CASE
WHEN LOWER(i.intent) LIKE '%claim%' OR LOWER(i.intent) LIKE '%sinistre%'
THEN 'Sinistre'
WHEN LOWER(i.intent) LIKE '%complaint%' OR LOWER(i.intent) LIKE
'%reclamation%' THEN 'Réclamation'
WHEN LOWER(i.intent) LIKE '%renew%' OR LOWER(i.intent) LIKE '%renouvel%'
THEN 'Renouvellement'
WHEN LOWER(i.intent) LIKE '%info%' OR LOWER(i.intent) LIKE '%information%'
THEN 'Information'
WHEN LOWER(i.intent) LIKE '%payment%' OR LOWER(i.intent) LIKE '%paiement%'
THEN 'Paiement'
ELSE 'Autre'
END AS intent_category
FROM Silver.interactions i
LEFT JOIN Silver.policies pol ON i.policy_id = pol.policy_id
) nested_0
```

3. Vues analytiques KPIs

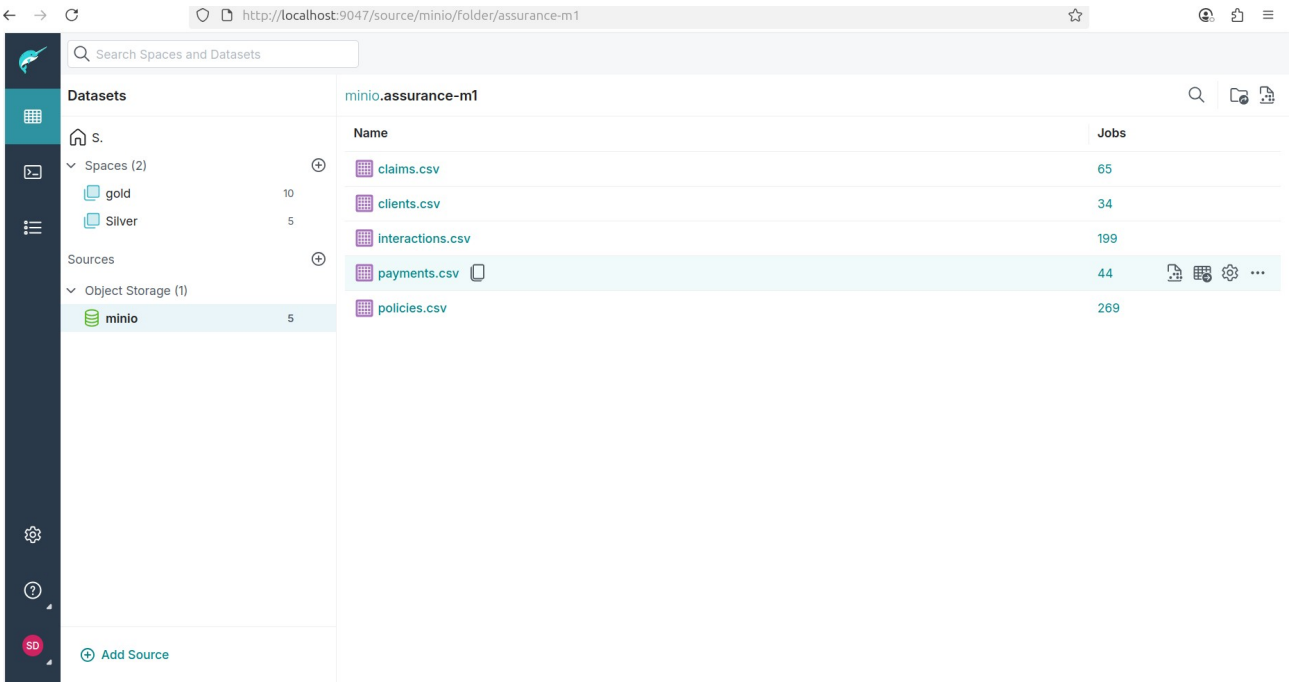
sql

```sql
-- Vue Loss Ratio par produit et canal

SELECT
pol.policy_type,
pol.sales_channel,
EXTRACT(YEAR FROM c.claim_date) AS claim_year,
EXTRACT(MONTH FROM c.claim_date) AS claim_month,
COUNT(DISTINCT c.claim_id) AS nb_claims,
SUM(pol.annual_premium_xof) AS total_premiums,
SUM(COALESCE(c.amount_paid_xof, 0)) AS total_claims_paid,
CASE
WHEN SUM(pol.annual_premium_xof) = 0 THEN 0
ELSE SUM(COALESCE(c.amount_paid_xof, 0)) / SUM(pol.annual_premium_xof)
END AS loss_ratio,
CASE
```

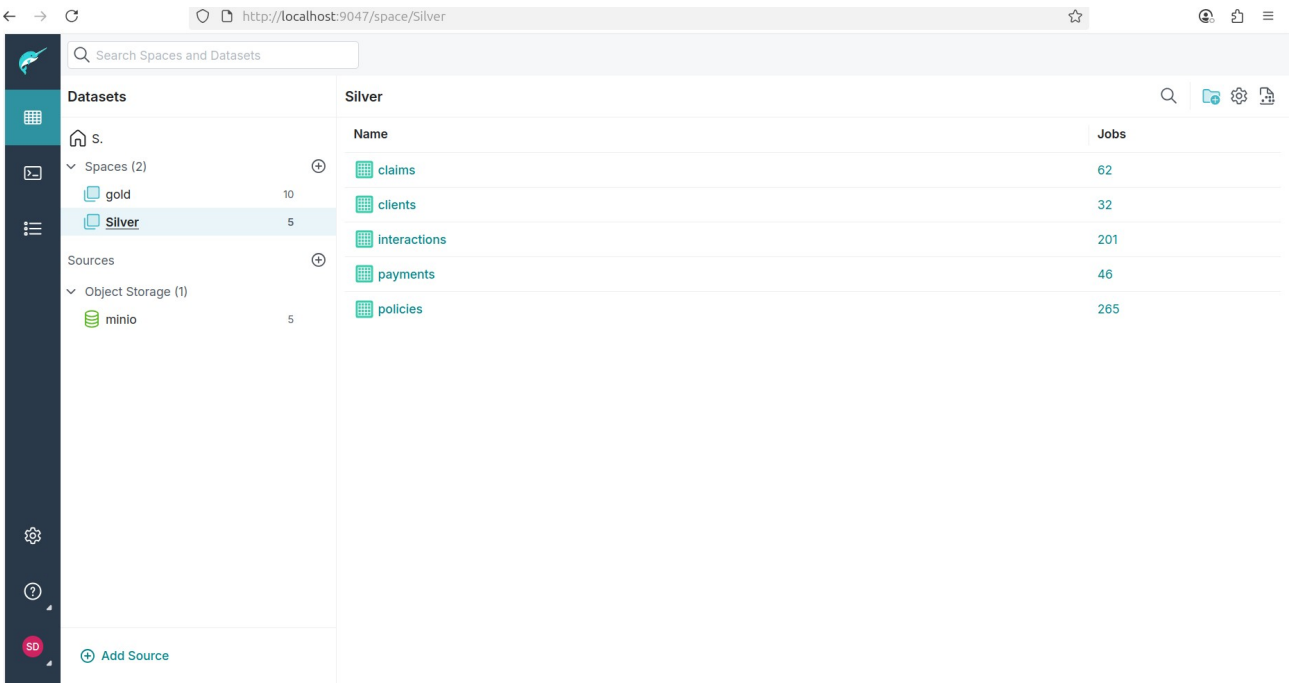
```sql
WHEN SUM(pol.annual_premium_xof) = 0 THEN 0
ELSE SUM(COALESCE(c.amount_paid_xof, 0)) / SUM(pol.annual_premium_xof)
END * 100 AS loss_ratio_percentage
FROM silver.policies pol
LEFT JOIN silver.claims c ON pol.policy_id = c.policy_id
WHERE pol.status = 'Actif'
GROUP BY pol.policy_type, pol.sales_channel, EXTRACT(YEAR FROM
c.claim_date), EXTRACT(MONTH FROM c.claim_date)


-- Vue Satisfaction client

SELECT
pol.policy_type,
pol.sales_channel,
i.intent_category,
COUNT(i.interaction_id) AS nb_interactions,
AVG(i.satisfaction) AS avg_satisfaction,
SUM(CASE WHEN i.satisfaction <= 2 THEN 1 ELSE 0 END) AS nb_low_satisfaction,
SUM(CASE WHEN i.satisfaction >= 4 THEN 1 ELSE 0 END) AS nb_high_satisfaction
FROM gold.fact_interactions i
JOIN silver.policies pol ON i.policy_id = pol.policy_id
WHERE i.satisfaction IS NOT NULL
GROUP BY pol.policy_type, pol.sales_channel, i.intent_category


-- Vue Rentabilité par client

SELECT
CAST(SPLIT_PART(c.customer_id, '.', 1) AS INTEGER) AS customer_id,
c.name,
c.city,
COUNT(DISTINCT CAST(SPLIT_PART(pol.policy_id, '.', 1) AS INTEGER)) AS
nb_policies,
SUM(pol.annual_premium_xof) AS total_premiums,
SUM(COALESCE(cl.amount_paid_xof, 0)) AS total_claims_paid,
SUM(pol.annual_premium_xof) - SUM(COALESCE(cl.amount_paid_xof, 0)) AS
profitability,
CASE
WHEN SUM(pol.annual_premium_xof) = 0 THEN 0
ELSE SUM(COALESCE(cl.amount_paid_xof, 0)) / SUM(pol.annual_premium_xof)
END AS loss_ratio,
AVG(i.satisfaction) AS avg_satisfaction
FROM silver.clients c
JOIN silver.policies pol
ON CAST(SPLIT_PART(c.customer_id, '.', 1) AS INTEGER) =
CAST(SPLIT_PART(pol.customer_id, '.', 1) AS INTEGER)
LEFT JOIN silver.claims cl
ON CAST(SPLIT_PART(pol.policy_id, '.', 1) AS INTEGER) =
CAST(SPLIT_PART(cl.policy_id, '.', 1) AS INTEGER)
LEFT JOIN silver.interactions i
```

```
ON CAST(SPLIT_PART(c.customer_id, '.', 1) AS INTEGER) =
CAST(SPLIT_PART(i.customer_id, '.', 1) AS INTEGER)
WHERE pol.status = 'Actif'
GROUP BY CAST(SPLIT_PART(c.customer_id, '.', 1) AS INTEGER), c.name, c.city
```

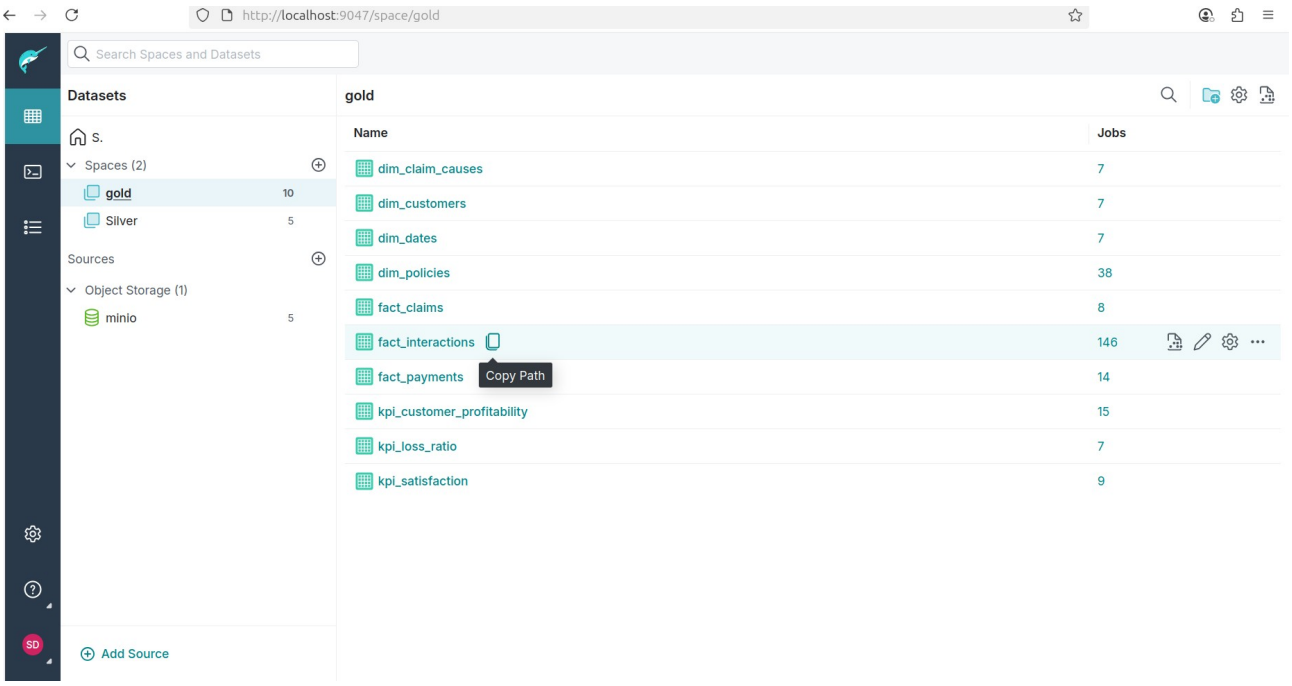
Livrable : Scripts SQL Dremio et captures d'écran des vues bronze/silver/gold.



Bronze



Silver

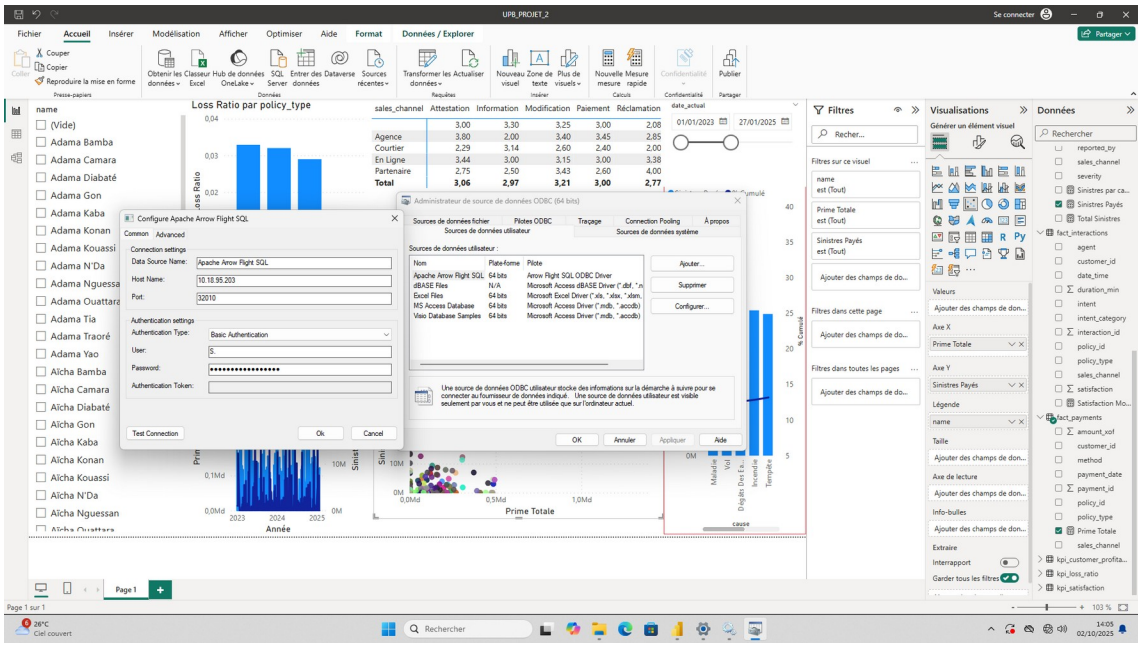


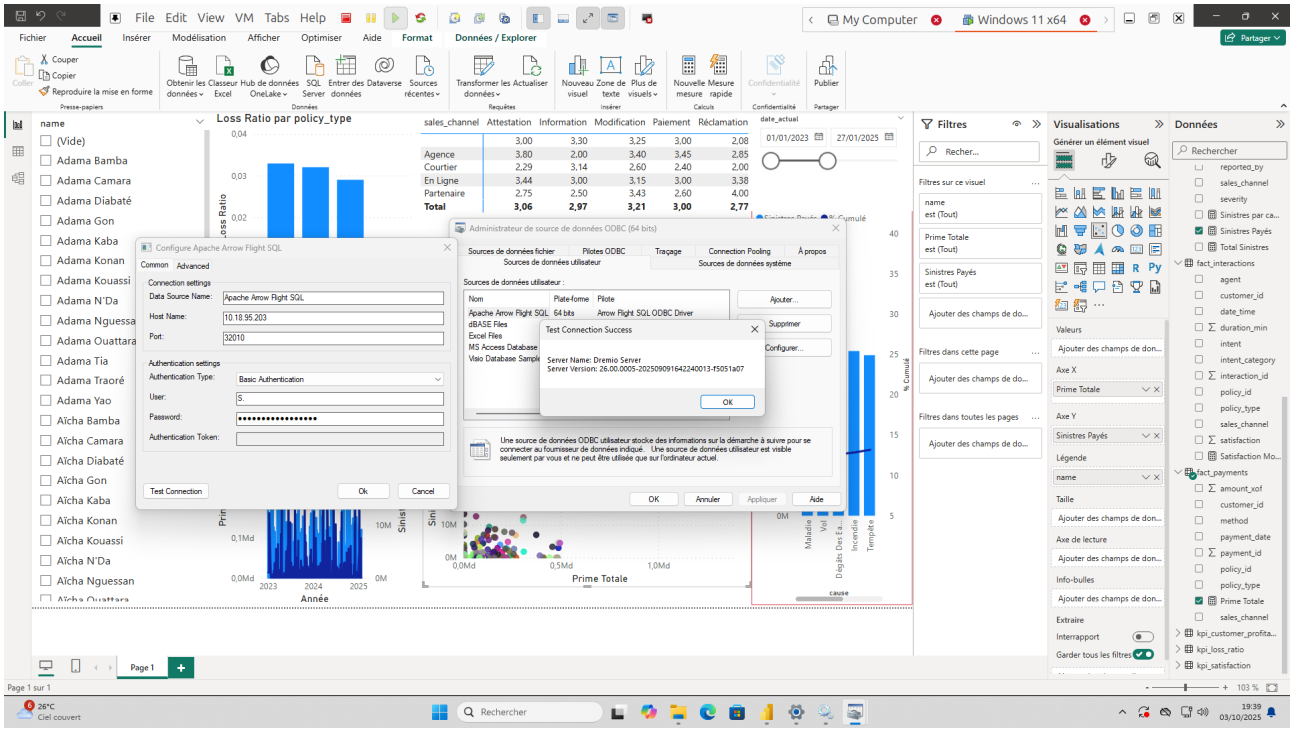
gold

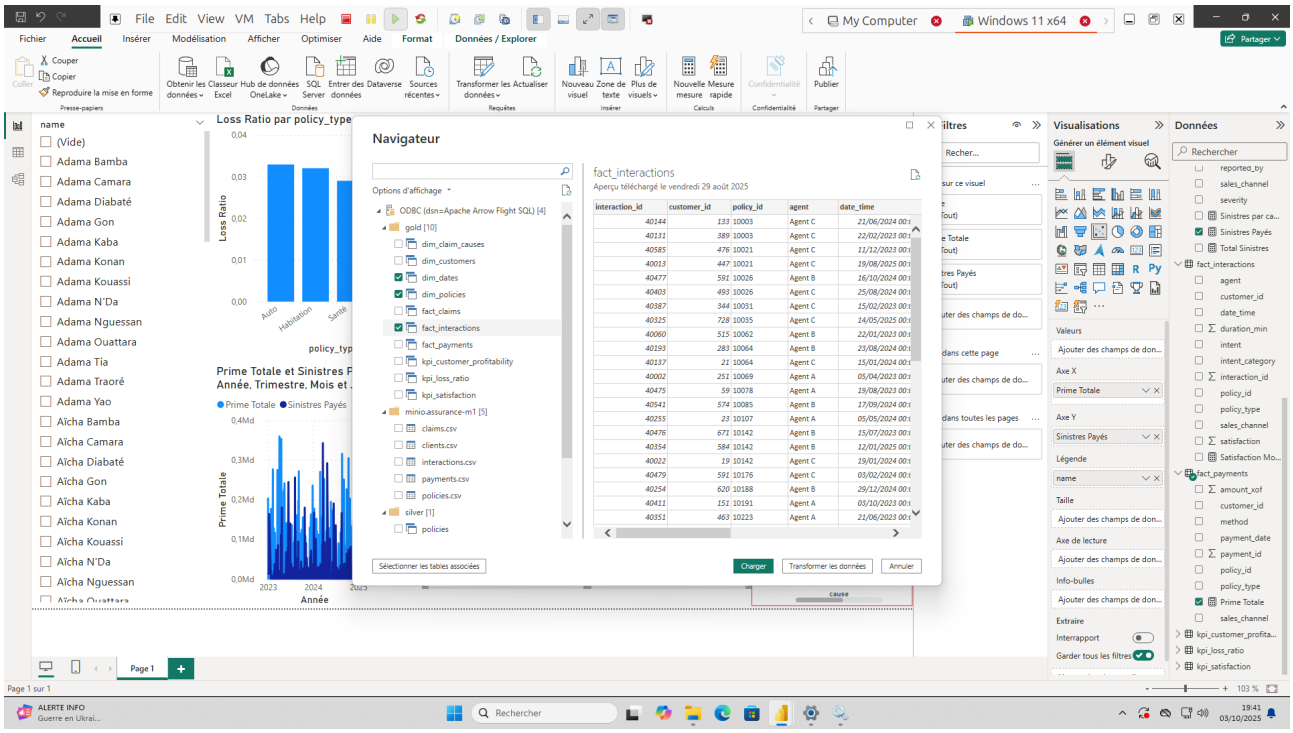
3. Modélisation et Analyse dans Power BI

Connexion à Dremio et import des vues Gold

1. **Dans Power BI :**
    - Sélectionner "Dremio" comme source de données
    - Se connecter avec les identifiants Dremio
    - Importer les vues gold (dim_customer, dim_policy, dim_date, fact_payments, fact_claims, fact_interactions)

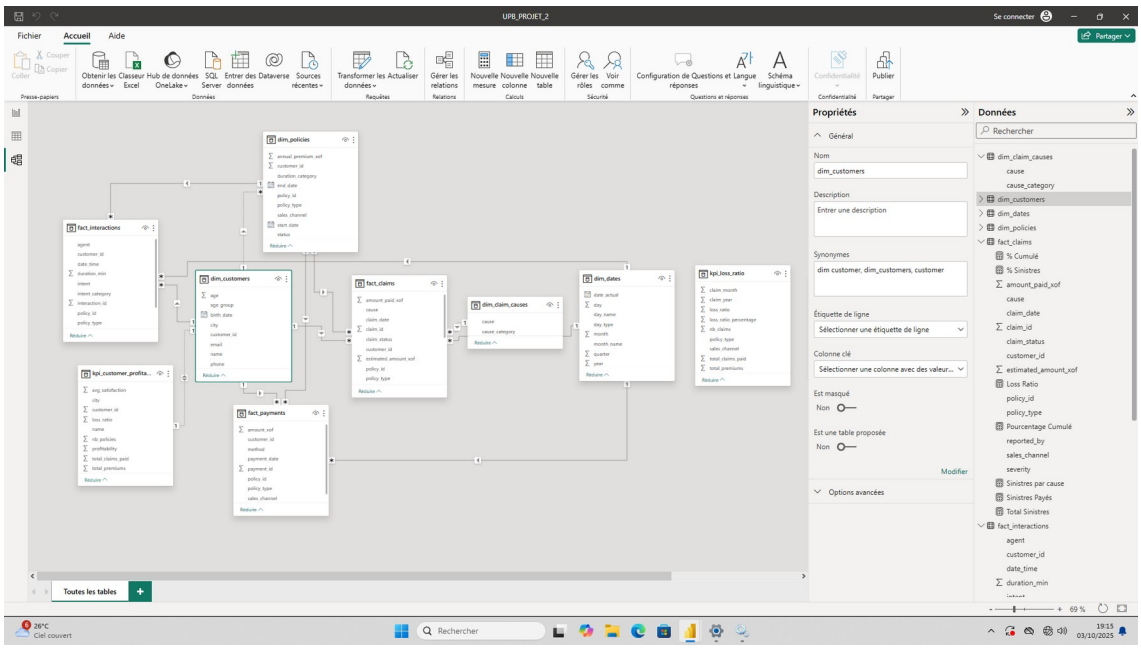




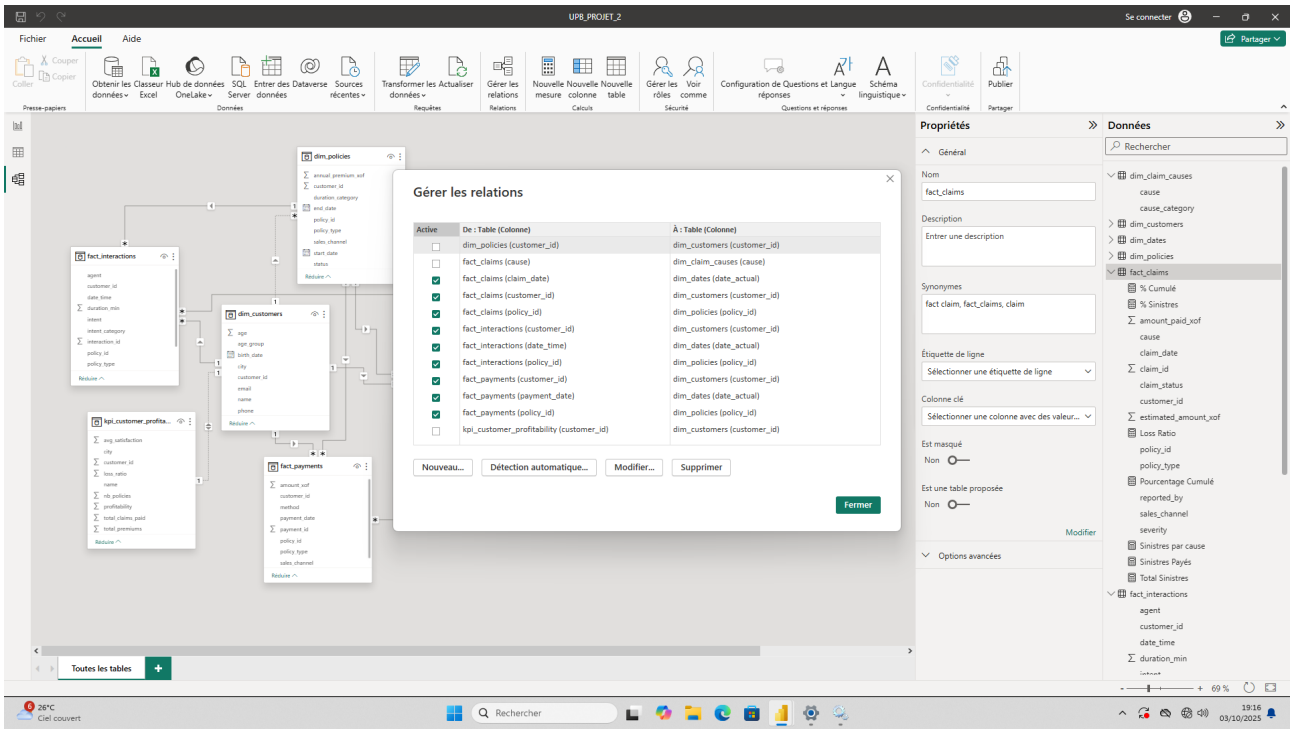


Modèle en étoile

- **Tables de dimensions :** dim_customer, dim_policy, dim_date
- **Tables de faits :** fact_payments, fact_claims, fact_interactions



Vue de modèle



Relation entre les tables

Mesures DAX

Dax

- % Cumulé =

VAR CurrentCause = MAX(fact_claims[cause])

RETURN

CALCULATE(

```
    SUMX(
        FILTER(
            ALL(fact_claims),
            [Sinistres par cause] >= CALCULATE([Sinistres par cause], fact_claims[cause] =
CurrentCause)
        ),
        [Sinistres par cause]
    )
) / [Total Sinistres]
```

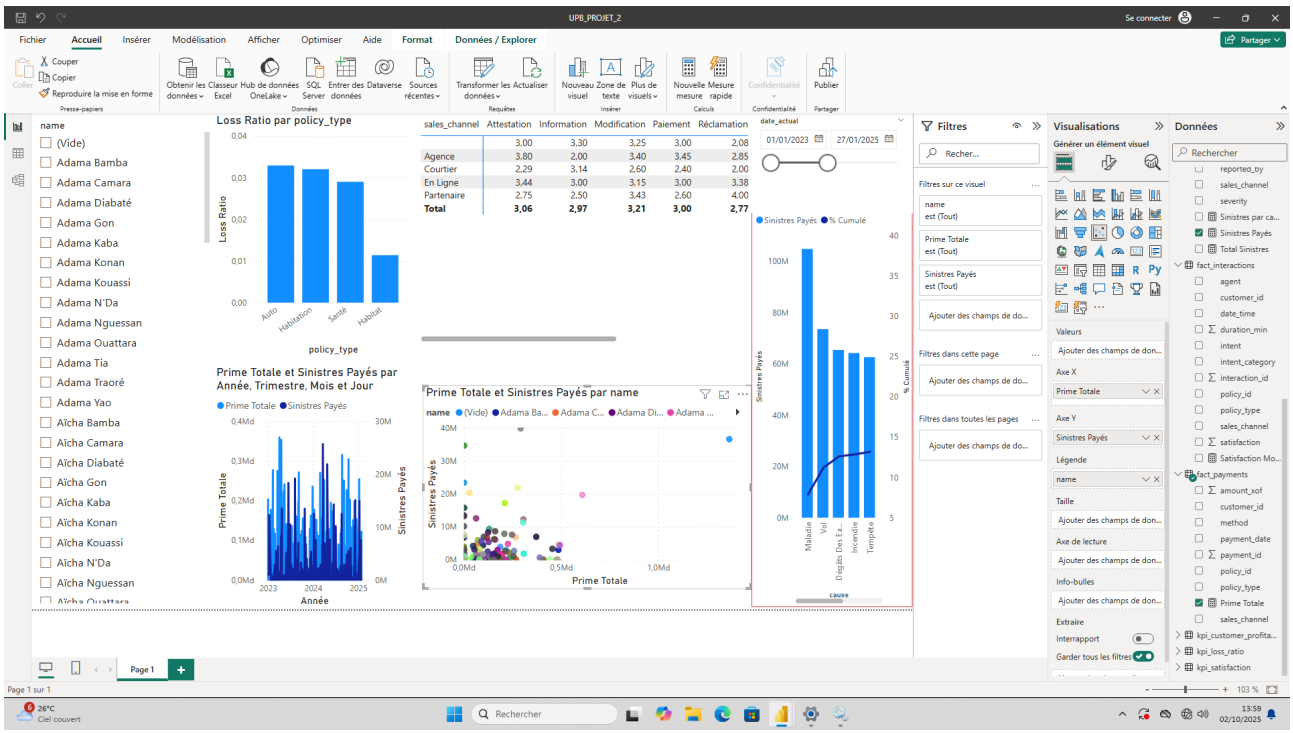
- Loss Ratio = DIVIDE([Sinistres Payés], [Prime Totale], 0)
- Pourcentage Cumulé =

```
VAR TotalSinistres = CALCULATE([Sinistres payés], ALL(fact_claims))
VAR Cumulative =
    SUMX(
        FILTER(
            ALLSELECTED(fact_claims),
            fact_claims[amount_paid_xof] >= fact_claims[amount_paid_xof]
        ),
        fact_claims[amount_paid_xof]
    )
RETURN
DIVIDE([Sinistres Payés], TotalSinistres,0)
```

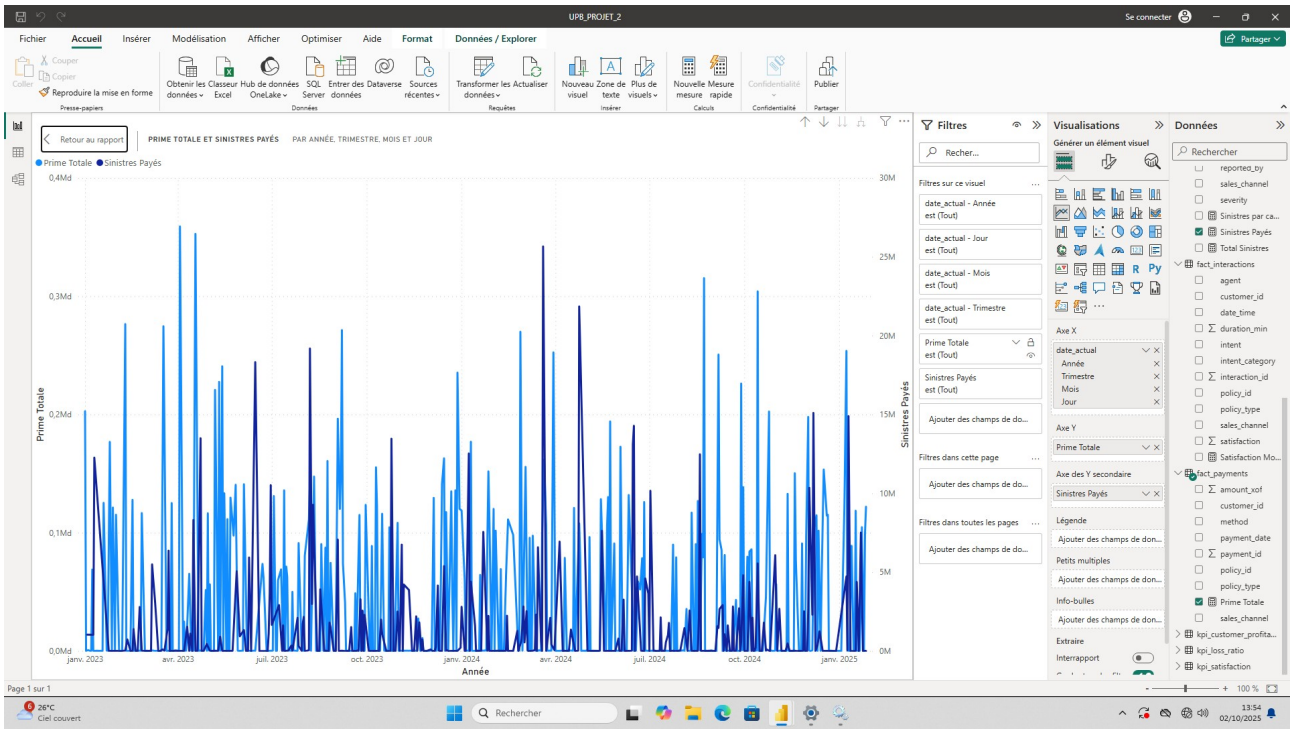
- % Sinistres = DIVIDE([Sinistres par cause], [Total Sinistres])

- Sinistres par cause = SUM(fact_claims[amount_paid_xof])
- Sinistres Payés = SUM(fact_claims[amount_paid_xof])
- Total Sinistres = SUM(fact_claims[amount_paid_xof])
- Satisfaction Moyenne = AVERAGE(fact_interactions[satisfaction])
- Prime Totale = SUM(fact_payments[amount_xof])

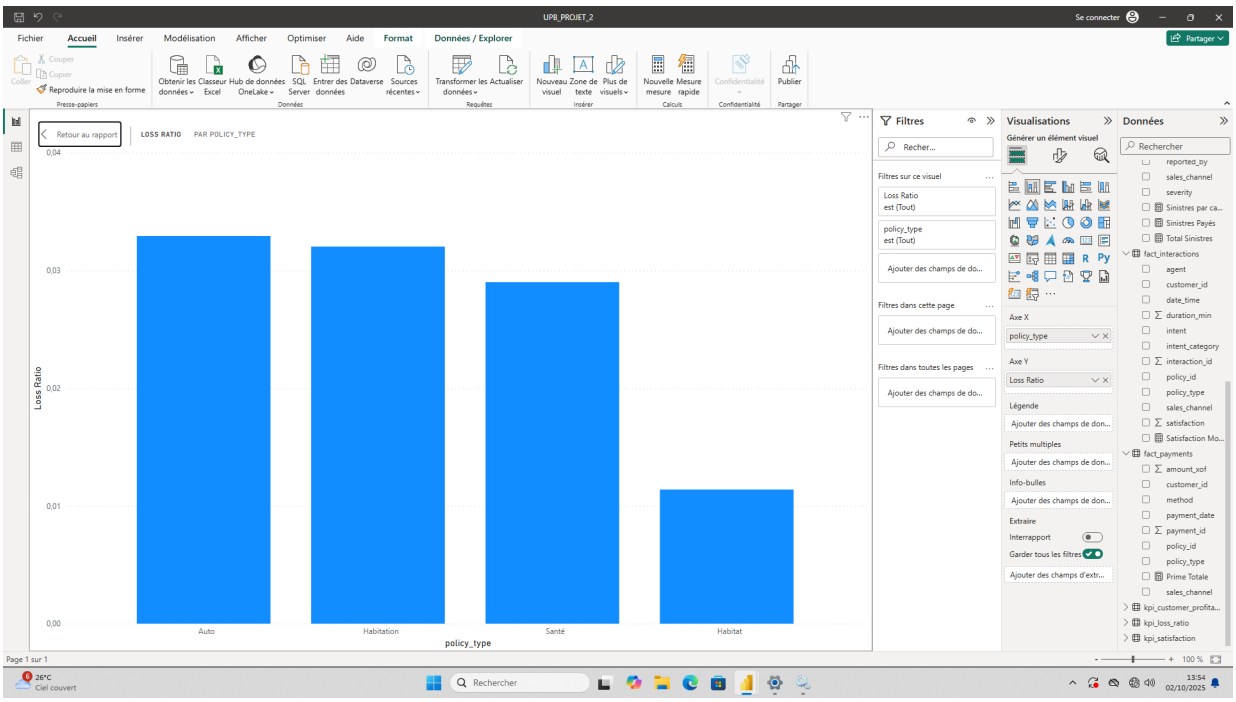
Visualisations



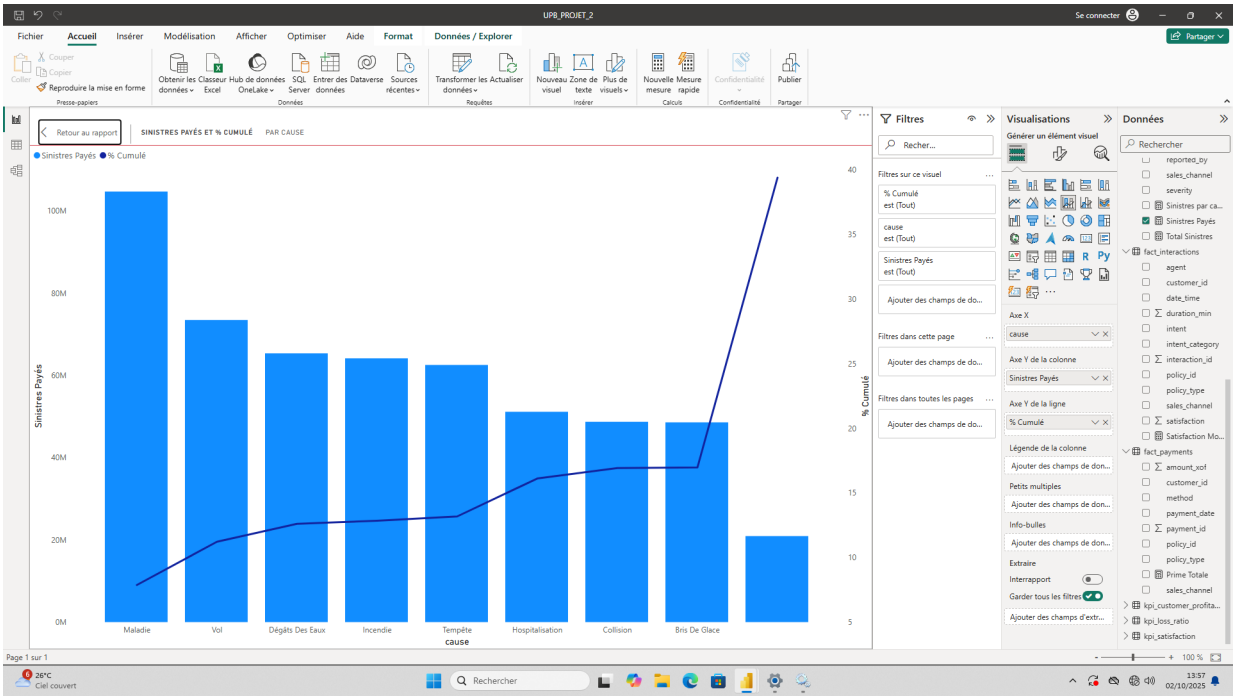
1. **Courbes temporelles :** Primes vs Sinistres par mois



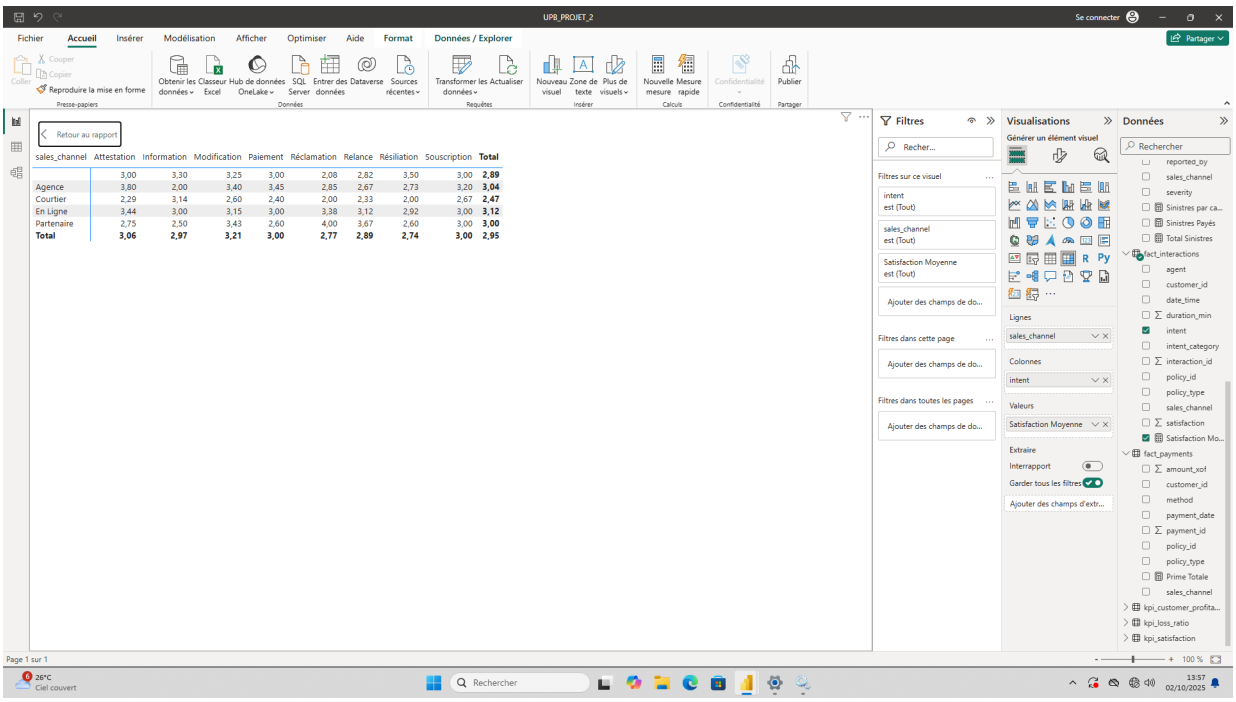
2. **Histogrammes :** Loss Ratio par produit et par canal



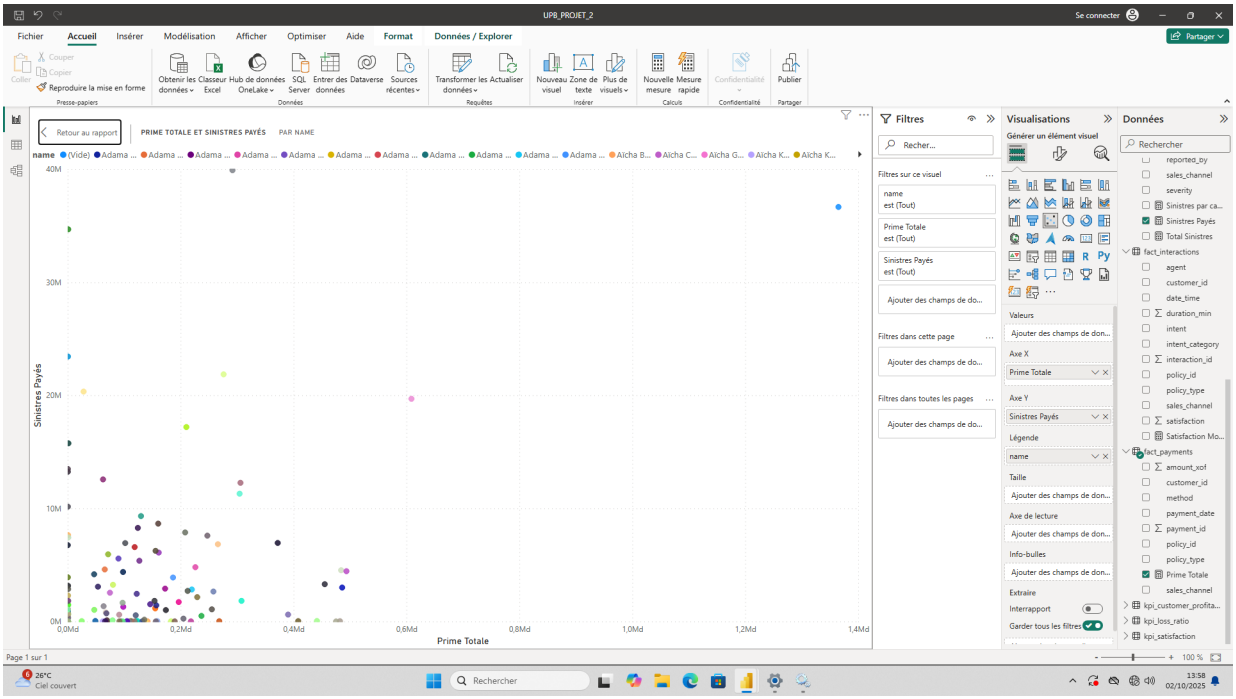
3. **Diagramme de Pareto :** Causes de sinistres les plus fréquentes



4. **Matrice :** Satisfaction par canal et type d'interaction



5. **Nuage de points :** Contrats "outliers" (primes basses, sinistres élevés)



Livrable : Fichier .pbix et captures d'écran des visualisations.

Insights et Recommandations

:

🎯 1. Analyse du Ratio de Sinistralité par Type de Police (Loss Ratio par `policy_type`)

- **Insights :**
  - Les polices **Auto**, **Habitation** et **Santé** ont un ratio de sinistralité similaire, aux alentours de **0,03 (3%)**.
  - La police **Habitat** a un ratio de sinistralité bien plus bas, autour de **0,01 (1%)**.
- **Recommandations :**
  - Mieux analyser la performance des produits **Habitat**, car le faible ratio pourrait signifier soit **peu de sinistres**, soit **des primes élevées** par rapport aux remboursements.
  - Revoir les politiques de souscription des produits **Auto**, **Santé**, et **Habitation**, s'ils génèrent un coût élevé (3%) en sinistres.

📈 2. Performance par Canal de Vente (`sales_channel`)

- **Insights :**
  - Le canal **"En Ligne"** montre un ratio de sinistralité le plus bas (moyenne autour de **2.08**).
  - Le canal **"Partenaire"** a un ratio élevé (**4.00** dans le cas de Réclamation).
  - Les canaux **Agence** et **Courtier** sont intermédiaires.
- **Recommandations :**
  - **Investir davantage dans les canaux digitaux ("En Ligne")**, car ils semblent plus rentables.
  - **Auditer les partenaires** : pourquoi le ratio est-il si élevé ? Mauvais ciblage, risque élevé ?
  - Optimiser ou **reformer les politiques de commissions** pour les courtiers/partenaires à forte sinistralité.

📊 3. Prime Totale vs Sinistres Payés par Nom (`name`)

- **Insights :**
  - Des individus ont des primes très élevées, mais aussi des sinistres payés élevés → potentiellement des **clients risqués mais rentables**.
  - Beaucoup d'individus ont des sinistres payés proches de zéro malgré des primes significatives.
- **Recommandations :**
  - Segmenter les clients pour identifier les **bons profils de rentabilité**.

- Mettre en place un **scoring de risque personnalisé** basé sur l'historique des paiements de sinistres et des primes versées.

📅 4. Évolution dans le temps (Graphique Année/Trimestre/Mois/Jour)

- **Insights :**
  - Des pics visibles fin 2024 et mi-2025 montrent des périodes de **hausse de sinistres**.
  - La tendance générale est relativement stable mais **cyclique**.
- **Recommandations :**
  - Analyser les **causes saisonnières ou conjoncturelles** de ces pics (météo, événements, lois, etc.).
  - Prévoir des **réserves** ou des **actions marketing ciblées** pendant les périodes à forte sinistralité.

📉 5. Sinistres Payés par Cause

- **Insights :**
  - Les causes comme **Maladie**, **Objets volés**, **Dégâts des eaux** ont les sinistres payés les plus élevés.
  - La courbe cumulée montre une **concentration importante** des sinistres sur peu de causes.
- **Recommandations :**
  - Prioriser les **actions de prévention** sur les causes les plus coûteuses (campagnes santé, sécurité, etc.).
  - Revoir les **conditions générales** pour ces causes (plafonds, exclusions, franchise...).

✅ Synthèse des Recommandations Stratégiques

Domaine	Recommandation
Type de police	Revoir produits avec ratio > 3%
Canal de vente	Promouvoir le canal en ligne
Clientèle	Segmentation par rentabilité & risque
Sinistralité saisonnière	Analyser pics pour actions préventives
Causes des sinistres	Réduire impact des causes les plus fréquentes

Conclusion

Cette solution complète permet de répondre à la problématique business en identifiant les segments à risque et en proposant des actions correctives pour améliorer la rentabilité. L'architecture en trois couches (bronze, silver, gold) assure la qualité des données et facilite l'analyse dans Power BI.