CREATE OR REPLACE TABLE silver\_orders (

order\_id STRING,

customer\_id STRING,

product\_id STRING,

quantity NUMBER(38,0),

order\_date TIMESTAMP\_TZ,

order\_notes STRING,

ingestion\_date DATE

);

CREATE OR REPLACE TABLE silver\_orders\_rejects (

raw\_record OBJECT,

reason STRING,

logged\_at TIMESTAMP\_TZ DEFAULT CURRENT\_TIMESTAMP()

);

DROP STREAM IF EXISTS ARBORE\_ORDERS\_strm;

stream cassé (sécurisation)

CREATE STREAM ARBORE\_ORDERS\_strm

ON TABLE ARBORE\_ORDERS;

CREATE OR REPLACE PROCEDURE sp\_load\_silver\_orders\_from\_parsed\_v1()

RETURNS STRING

LANGUAGE SQL

EXECUTE AS OWNER

AS

$$

BEGIN

CREATE OR REPLACE TEMP TABLE tmp\_parsed AS

SELECT

TRIM(order\_id) AS order\_id,

TRIM(customer\_id) AS customer\_id,

TRIM(product\_id) AS product\_id,

TRY\_TO\_NUMBER(TO\_VARCHAR(quantity)) AS quantity\_num,

COALESCE(

TRY\_TO\_TIMESTAMP\_TZ(order\_date, 'YYYY-MM-DD"T"HH24:MI:SS"Z"'),

TRY\_TO\_TIMESTAMP\_TZ(order\_date, 'YYYY-MM-DD"T"HH24:MI:SS.FF"Z"'),

TRY\_TO\_TIMESTAMP\_TZ(order\_date, 'YYYY-MM-DD'),

TRY\_TO\_TIMESTAMP\_TZ(order\_date)

) AS order\_date\_ts,

NULLIF(TRIM(order\_notes), '') AS order\_notes,

OBJECT\_CONSTRUCT(

'order\_id', order\_id,

'customer\_id', customer\_id,

'product\_id', product\_id,

'quantity', quantity,

'order\_date', order\_date,

'order\_notes', order\_notes

) AS raw\_full

FROM ARBORE\_ORDERS\_strm

WHERE METADATA$ACTION IN ('INSERT');

CREATE OR REPLACE TEMP TABLE tmp\_validated AS

SELECT

\*,

IFF(order\_id IS NULL, 'order\_id is NULL',

IFF(quantity\_num IS NULL, 'quantity not numeric',

IFF(quantity\_num < 0, 'quantity negative',

IFF(order\_date\_ts IS NULL, 'order\_date invalid',

NULL)

)

)

) AS reject\_reason

FROM tmp\_parsed;

INSERT INTO silver\_orders\_rejects (raw\_record, reason)

SELECT raw\_full, reject\_reason

FROM tmp\_validated

WHERE reject\_reason IS NOT NULL;

CREATE OR REPLACE TEMP TABLE tmp\_good AS

SELECT

order\_id,

customer\_id,

product\_id,

CAST(quantity\_num AS NUMBER(38,0)) AS quantity,

order\_date\_ts AS order\_date,

order\_notes,

CURRENT\_DATE() AS ingestion\_date

FROM tmp\_validated

WHERE reject\_reason IS NULL;

CREATE OR REPLACE TEMP TABLE tmp\_dedup AS

SELECT \*

FROM (

SELECT

g.\*,

ROW\_NUMBER() OVER (PARTITION BY order\_id

ORDER BY order\_date DESC NULLS LAST) AS rn

FROM tmp\_good g

)

WHERE rn = 1;

MERGE INTO silver\_orders AS tgt

USING tmp\_dedup AS src

ON tgt.order\_id = src.order\_id

WHEN MATCHED THEN

UPDATE SET

tgt.customer\_id = src.customer\_id,

tgt.product\_id = src.product\_id,

tgt.quantity = src.quantity,

tgt.order\_date = src.order\_date,

tgt.order\_notes = src.order\_notes,

tgt.ingestion\_date = src.ingestion\_date

WHEN NOT MATCHED THEN

INSERT (

order\_id, customer\_id, product\_id, quantity, order\_date,

order\_notes, ingestion\_date

)

VALUES (

src.order\_id, src.customer\_id, src.product\_id, src.quantity,

src.order\_date, src.order\_notes, src.ingestion\_date

);

RETURN 'OK';

END;

$$;

/\* =========================

4) (OPTIONNEL) TASK D’ORCHESTRATION

========================= \*/

-- Remarque : si tu veux un rafraîchissement automatique régulier, décommente ces lignes :

-- CREATE OR REPLACE TASK t\_load\_silver\_orders\_from\_parsed -- je (re)crée la task d’orchestration

-- SCHEDULE = 'USING CRON \*/5 \* \* \* \* Europe/Paris' -- je planifie toutes les 5 minutes (TZ Paris)

-- USER\_TASK\_MANAGED\_INITIAL\_WAREHOUSE\_SIZE = 'XSMALL' -- je laisse Snowflake gérer le warehouse

-- AS

-- CALL sp\_load\_silver\_orders\_from\_parsed\_v1(); -- j’appelle la procédure d’ETL Silver

--

-- ALTER TASK t\_load\_silver\_orders\_from\_parsed RESUME; -- j’active la task

CALL sp\_load\_silver\_orders\_from\_parsed\_v1();

SELECT \* FROM silver\_orders ORDER BY order\_date DESC LIMIT 20;

SELECT reason, COUNT(\*) FROM silver\_orders\_rejects -- je contrôle les rejets éventuels

GROUP BY 1 ORDER BY 2 DESC;