# SNOWFLAKE CASE STUDY

## QUERIES

Create database sales\_db;

create schema sales\_db.raw\_schema;

create file format sales\_db.raw\_schema.csv\_format

type = 'csv'

field\_optionally\_enclosed\_by = '"'

skip\_header = 1;

create stage sales\_db.raw\_schema.sales\_stage

file\_format = (format\_name = 'csv\_format');

CREATE OR REPLACE TABLE SALES\_DB.RAW\_SCHEMA.sales\_raw\_csv (

transaction\_id STRING,

region STRING,

country STRING,

product STRING,

customer STRING,

sales\_rep STRING,

transaction\_date STRING,

timestamp VARCHAR(30),

quantity STRING,

unit\_price STRING,

total\_amount NUMBER,

order\_status STRING,

payment\_method STRING,

product\_details VARIANT,

customer\_info VARIANT

);

COPY INTO SALES\_DB.RAW\_SCHEMA.sales\_raw\_csv

FROM @sales\_stage/sales\_data\_dirty.csv

FILE\_FORMAT = (FORMAT\_NAME = 'csv\_format')

ON\_ERROR = 'SKIP\_FILE\_3%';

select count(\*) as total\_txns, count(distinct transaction\_id) as distinct\_txns

from sales\_db.raw\_schema.sales\_raw\_csv;

select count(\*) as null\_count

from sales\_db.raw\_schema.sales\_raw\_csv

where transaction\_id is null or total\_amount is null;

select count(\*) from sales\_db.raw\_schema.sales\_raw\_csv

where total\_amount < 0;

select \* from sales\_db.raw\_schema.sales\_raw\_csv;

create schema clean\_schema;

CREATE OR REPLACE TABLE SALES\_DB.CLEAN\_SCHEMA.sales\_clean AS

SELECT

transaction\_id,

region,

country,

product,

customer,

sales\_rep,

-- Convert transaction\_date to DATE

TRY\_TO\_DATE(transaction\_date, 'DD-MM-YYYY') AS transaction\_date,

-- Convert timestamp to TIMESTAMP

TRY\_TO\_TIMESTAMP(timestamp, 'DD-MM-YYYY HH24:MI') AS transaction\_ts,

-- Replace negative quantity and unit\_price with NULL

CASE WHEN TRY\_TO\_NUMBER(quantity) >= 0 THEN TRY\_TO\_NUMBER(quantity) ELSE NULL END AS quantity,

CASE WHEN TRY\_TO\_NUMBER(unit\_price) >= 0 THEN TRY\_TO\_NUMBER(unit\_price) ELSE NULL END AS unit\_price,

-- Replace negative total\_amount with NULL

CASE WHEN TRY\_TO\_NUMBER(total\_amount) >= 0 THEN TRY\_TO\_NUMBER(total\_amount) ELSE NULL END AS total\_amount,

order\_status,

payment\_method,

-- Flattened product\_details JSON

product\_details:brand::STRING AS brand,

product\_details:category::STRING AS category,

TRY\_TO\_DOUBLE(product\_details:ratings::STRING) AS ratings,

product\_details:specs::STRING AS specs,

product\_details:subcategory::STRING AS subcategory,

-- Flattened customer\_info JSON

customer\_info:demographics::STRING AS demographics,

customer\_info:preferences::STRING AS preferences,

customer\_info:segment::STRING AS segment

FROM SALES\_DB.RAW\_SCHEMA.sales\_raw\_csv

WHERE transaction\_id IS NOT NULL

AND TRY\_TO\_TIMESTAMP(timestamp, 'DD-MM-YYYY HH24:MI') IS NOT NULL;

delete from sales\_db.clean\_schema.sales\_clean

where transaction\_id is null;

DELETE

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

WHERE total\_amount IS NULL

OR quantity IS NULL

OR unit\_price IS NULL

OR quantity = 0

OR unit\_price = 0;

select \* from sales\_db.clean\_schema.sales\_clean;

select \* from sales\_db.raw\_schema.sales\_raw\_csv;

select count(\*) as Negetive\_value from sales\_db.clean\_schema.sales\_clean

where total\_amount<0;

ALTER TABLE SALES\_DB.CLEAN\_SCHEMA.sales\_clean

ADD profit\_margin FLOAT;

ALTER TABLE SALES\_DB.CLEAN\_SCHEMA.sales\_clean

ADD sales\_quarter STRING;

UPDATE SALES\_DB.CLEAN\_SCHEMA.sales\_clean

SET profit\_margin = total\_amount - (quantity \* unit\_price);

UPDATE SALES\_DB.CLEAN\_SCHEMA.sales\_clean

SET sales\_quarter = 'Q' || CAST(EXTRACT(QUARTER FROM transaction\_date) AS STRING);

select count(\*) from sales\_db.clean\_schema.sales\_clean;

Create schema dwh\_schema;

drop schema dwh\_schema;

CREATE OR REPLACE TABLE SALES\_DB.CLEAN\_SCHEMA.DIM\_REGION AS

SELECT DISTINCT TRIM(region) as regions, country FROM SALES\_DB.CLEAN\_SCHEMA.SALES\_CLEAN;

-- DIM\_PRODUCT

CREATE OR REPLACE TABLE SALES\_DB.CLEAN\_SCHEMA.DIM\_PRODUCT AS

SELECT DISTINCT brand, category, ratings, subcategory FROM SALES\_DB.CLEAN\_SCHEMA.SALES\_CLEAN;

-- DIM\_CUSTOMER

CREATE OR REPLACE TABLE SALES\_DB.CLEAN\_SCHEMA.DIM\_CUSTOMER AS

SELECT DISTINCT customer, segment, demographics, preferences FROM SALES\_DB.CLEAN\_SCHEMA.SALES\_CLEAN;

CREATE OR REPLACE TABLE SALES\_DB.CLEAN\_SCHEMA.FACT\_SALES AS

SELECT

transaction\_id,

transaction\_date,

transaction\_ts,

customer,

product,

region,

country,

sales\_rep,

quantity,

unit\_price,

total\_amount,

profit\_margin,

sales\_quarter,

order\_status,

payment\_method,

ratings

FROM SALES\_DB.CLEAN\_SCHEMA.SALES\_CLEAN;

SELECT \* FROM SALES\_DB.CLEAN\_SCHEMA.FACT\_SALES;

SELECT \* FROM SALES\_DB.CLEAN\_SCHEMA.DIM\_CUSTOMER;

SELECT \* FROM SALES\_DB.CLEAN\_SCHEMA.DIM\_PRODUCT;

SELECT \* FROM SALES\_DB.CLEAN\_SCHEMA.DIM\_REGION;

SELECT

TRIM(region) AS cleaned\_region,

SUM(total\_amount) AS total\_sales

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

GROUP BY cleaned\_region

ORDER BY total\_sales DESC;

delete from sales\_db.clean\_schema.sales\_clean

where segment is null;

SELECT

segment,

COUNT(\*) AS total\_orders,

SUM(total\_amount) AS total\_sales,

AVG(total\_amount) AS avg\_order\_value

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

GROUP BY segment

ORDER BY total\_sales DESC;

delete from sales\_db.clean\_schema.sales\_clean

where brand is null;

SELECT

brand,

COUNT(\*) AS total\_orders,

SUM(total\_amount) AS total\_sales,

AVG(ratings) AS avg\_rating,

ROUND(

(AVG(ratings) \* COUNT(\*))/100) AS performance\_score

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

GROUP BY brand

ORDER BY performance\_score DESC;

SELECT regions

FROM (

SELECT TRIM(region) as regions, SUM(total\_amount) AS total\_sales

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

GROUP BY regions

ORDER BY total\_sales DESC

LIMIT 1

);

SELECT brand

FROM (

SELECT brand, ROUND(AVG(ratings) / NULLIF(SUM(total\_amount), 0), 4) AS ratio

FROM SALES\_DB.CLEAN\_SCHEMA.sales\_clean

GROUP BY brand

ORDER BY ratio DESC

LIMIT 1

);

## SCREEN SHOTS









