CASE STUDY : SNOWFLAKE

You should build a complete ETL pipeline in Snowflake Snowsight using a Sales dataset (CSV + Parquet). They will create stages, load raw data, clean and transform, build star-schema tables (fact + dimensions), and finally run reports for analytics.

Assignment Tasks

Task 1: Setup Database & Schema

1. Create a new database SALES_DB and schema RAW_SCHEMA.
2. Create file formats for CSV and Parquet data (csv_format, parquet_format).
3. Create a named stage sales_stage in RAW_SCHEMA.

```
create database sales_db;


create or replace schema raw_schema;

create schema clean_schema;

create schema star_schema;


create or replace file format csv_format
    type = 'csv'
    field_delimiter = ','
    skip_header = 1
    field_optionally_enclosed_by = '"'
    null_if = ('', 'null');


create stage sales_db.raw_schema.sales_stage;
```

Task 2: Load Raw Data

1. Create a staging table sales_raw with JSON columns for product_details and customer_info.
2. Create another staging table Raw_Parquet_table for Parquet data.
3. Load sales_data_1000.csv and sales_data_1000.parquet into the staging tables.

```
create or replace table sales_raw (
    transaction_id string,
```

```
    region string,
    country string,
    product string,
    customer string,
    sales_rep string,
    transaction_date varchar(400),
    timestamp varchar(400),
    quantity varchar(400),
    unit_price varchar(400),
    total_amount varchar(400),
    order_status string,
    payment_method string,
    product_details varchar(400),
    customer_info varchar(400)
);

copy into sales_db.raw_schema.sales_raw
from @sales_db.raw_schema.sales_stage
files = ('sales_data_dirty.csv')
file_format = csv_format
on_error = 'continue';

create or replace table raw_schema.sales_raw_refined as
select
    transaction_id,
    region,
    country,
    product,
    customer,
    sales_rep,
    try_to_date(transaction_date, 'dd-mm-yyyy') as transaction_date,
    timestamp,
```

```
    try_cast(quantity as integer) as quantity,
    try_cast(unit_price as float) as unit_price,
    try_cast(total_amount as float) as total_amount,
    order_status,
    payment_method,
    try_parse_json(product_details) as product_details,
    try_parse_json(customer_info) as customer_info
from raw_schema.sales_raw
where coalesce(transaction_id, region, country, product, customer,
sales_rep, transaction_date, order_status, payment_method, quantity,
unit_price, total_amount) is not null;
```

Task 3: Data Quality Checks

1. Write queries to count total vs distinct transactions, null values, and negative values.

```
select count(*) as total_transactions, count(distinct
transaction_id) as distinct_transactions
from sales_raw_refined;

select count(*) as null_values_count
from sales_raw_refined
where transaction_id is null or unit_price is null;

select count(*) as negative_values_count
from sales_raw_refined
where total_amount < 0;
```

Task 4: Flatten JSON Data

1. Extract fields from product_details and customer_info using Snowflake's JSON operators.

```
create or replace table clean_schema.sales_flattened as
select
```

```
    transaction_id,
    product_details:brand as brand,
    product_details:category as product_category,
    product_details:ratings as product_rating,
    product_details:specs as product_specs,
    product_details:subcategory as product_subcategory,
    customer_info:demographics as demographics,
    customer_info:preferences as preferences,
    customer_info:segment as segment
from raw_schema.sales_raw_refined;
```

Task 5: Clean & Transform

1. Create a cleaned table sales_clean in CLEAN_SCHEMA.
2. Apply transformations like TRY_TO_DATE, replacing negatives with NULL, removing missing values.

```
create or replace table clean_schema.sales_raw_cleaned as
select *
from raw_schema.sales_raw_refined;

delete from clean_schema.sales_raw_cleaned
where coalesce(transaction_id, region, country, product, customer,
sales_rep, transaction_date, order_status, payment_method, quantity,
unit_price, total_amount) is null;
```

Task 6: Feature Engineering

1. Add new columns profit_margin and sales_quarter to sales_clean.
2. Update table with calculated values.

```
alter table sales_db.clean_schema.sales_raw_cleaned add column
profit_margin number;
alter table sales_db.clean_schema.sales_raw_cleaned add column
sales_quarter varchar;
```

```
update sales_db.clean_schema.sales_raw_cleaned
set
    profit_margin = total_amount * 0.2,
    sales_quarter = case
        when month(transaction_date) in (1, 2, 3) then 'q1'
        when month(transaction_date) in (4, 5, 6) then 'q2'
        when month(transaction_date) in (7, 8, 9) then 'q3'
        else 'q4'
    end;
```

Task 7: Create Fact & Dimension Tables

1. Create dimension tables DIM_REGION, DIM_PRODUCT, DIM_CUSTOMER.
2. Create fact table FACT_SALES with transaction and sales metrics.

```
create or replace table sales_db.star_schema.dim_region as
select distinct region, country from clean_schema.sales_raw_cleaned;

create or replace table sales_db.star_schema.dim_product as
select distinct product, quantity, unit_price from
clean_schema.sales_raw_cleaned;

create or replace table sales_db.star_schema.dim_customer as
select distinct customer, customer_info, region, country,
order_status from clean_schema.sales_raw_cleaned;

create or replace table sales_db.star_schema.fact_sales as
select
    transaction_id,
    transaction_date,
    region,
    country,
    product_details,
    customer_info,
    sales_rep,
```

```
    quantity,
    unit_price,
    total_amount,
    profit_margin,
    sales_quarter,
    order_status,
    payment_method
from clean_schema.sales_raw_cleaned;
```

Task 8: Reporting

Run queries:
1. Sales by Region
2. Customer Segment Analysis
3. Product Brand Performance
4. Order Status Distribution

Question 14: Which region had the highest sales?
Question 15: Which product brand has the best rating-to-sales ratio?

```
-- Top region by sales
select region, country, sum(total_amount) as total_sales
from star_schema.fact_sales
group by region, country
order by total_sales desc;
```

	REGION	COUNTRY	TOTAL_SALES
1	Africa	South Africa	388645.84
2	North America	USA	348519.38
3	Europe	Spain	341294.39
4	Africa	Egypt	339998.25
5	South America	Chile	315199.28
6	North America	Mexico	280126.15
7	South America	Brazil	262877.54
8	Africa	Nigeria	256379.83
9	South America	Argentina	244956.3
10	Asia	Japan	235978.37
11	North America	Canada	232958.3

-- Customer segment performance

```
select
    split_part(customer_info, ':', 2) as customer_segment,
    sum(total_amount) as total_sales
from star_schema.fact_sales
group by customer_segment
order by total_sales desc;
```

	CUSTOMER_SEGMENT	TOTAL_SALES
1	"Young","preferences"	1601594.75
2	"Middle-aged","preferences"	1548322.42
3	"Senior","preferences"	1497330.65

-- Product brand performance

```
select
    split_part(product_details, ':', 2) as product_brand,
    sum(total_amount) as total_sales
from star_schema.fact_sales
group by product_brand
```

```
order by total_sales desc;
```

	PRODUCT_BRAND	TOTAL_SALES
1	"Nike","category"	498493.01
2	"Wayfair","category"	442798.55
3	"Home Depot","category"	430548.8
4	"Puma","category"	423963.24
5	"Adidas","category"	419151.18
6	"Reebok","category"	409443.56
7	"Under Armour","category"	337323.89
8	"Dell","category"	322265.23
9	"HP","category"	318416.02
10	"IKEA","category"	316866.43
11	"Samsung","category"	285943.75

-- Order status distribution

```
select
    order_status,
    count(*) as order_count
from star_schema.fact_sales
group by order_status
order by order_count desc;
```

	ORDER_STATUS	ORDER_COUNT
1	Completed	228
2	Returned	221
3	Pending	213
4	Cancelled	204
5	Shipped	5
6	Delivered	3

Question 14: Which region had the highest sales?

	REGION	COUNTRY	TOTAL_SALES
1	Africa	South Africa	388645.84

```
select    product_details:brand::string as product_brand,

regexp_substr(product_details:ratings::string, '[0-9]+(\.[0-9]+)?')
as extracted_ratings,

total_amount

from star_schema.fact_sales

limit 10;

select

    product_details:brand::string as product_brand,

    avg(cast(regexp_substr(product_details:ratings::string, '[0-
9]+(\.[0-9]+)?') as float) / total_amount) as rating_to_sales_ratio

from star_schema.fact_sales

where regexp_substr(product_details:ratings::string, '[0-9]+(\.[0-
9]+)?') is not null

group by product_brand

order by rating_to_sales_ratio desc;
```

	PRODUCT_BRAND	RATING_TO_SALES_RATIO
1	Adidas	0.002926540606
2	Reebok	0.00244055405
3	Home Depot	0.00208088128
4	Under Armour	0.002018575633
5	Samsung	0.002002757783
6	Decathlon	0.001923721211
7	Wayfair	0.001865868392
8	Nike	0.001861574979
9	HP	0.001778467488
10	Apple	0.001611586633
11	IKEA	0.001218314094