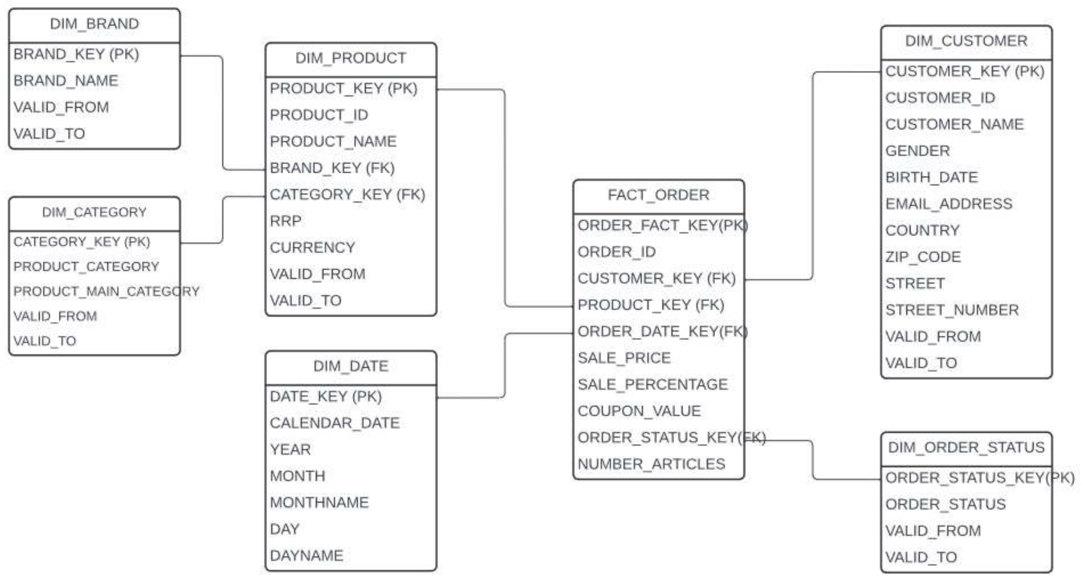
Logical Model



Explanation of steps

Note: This is the same description as in the .SQL file. Copying the description here as well in case the indentation in the .SQL file does not show clearly in the editor.

Step 1: Loading the 3 files into the Snowflake database

1. Uploaded the 3 files to AWS S3 to emulate the scenario which usually happens in a real world project

2. Created database, staging schema, table definitions in Snowflake

3. Created Storage Integration object, File Format object, External Stage object to facilitate bringing data from AWS S3 to Snowflake

4. Load the data from external stage into Snowflake tables using COPY command

Step 2: Create a DWH "mid layer" data model for the data

1. Created CORE schema in the database and created DIM_DATE generic date dimension table

2. Created Dimension Tables

    a. Tables have Surrogate Key as the Primary Key

    b. VALID_FROM and VALID_TO are used to track history in SCD2 implementation

c. Following tables are created

- DIM_CUSTOMER: Source data has multiple customers with the same name and email address, but different gender, birth date and address. This implies that they are different customers.
- DIM_BRAND
- DIM_CATEGORY
- DIM_PRODUCT: Has foreign key references to DIM_BRAND and DIM_CATEGORY
- DIM_ORDER_STATUS

3. Created FACT_ORDER fact table

a. Fact table is also using Surrogate Key as the Primary Key because of following reasons:

- ORDER_ID which is the business key of the OLTP system is not unique by itself
- Even the composite primary key comprising ORDER_ID, CUSTOMER_KEY, PRODUCT_KEY, ORDER_DATE_KEY is not unique
- There is no unique identifier (like line item number) that differentiates between multiple entries of the same product and same order.
- Measures like sale_price should not be part of a primary key

b. There are 2 rows in the order data where all attributes are identical suggesting that there are data entry issues

c. For some orders while ORDER_ID, CUSTOMER_ID, PRODUCT_ID, and ORDER_DATE are the same, the SALE_PRICE, SALE_PERCENTAGE, and COUPON_VALUE differ. This suggests that multiple discounts or promotions are applied separately or same product might have been sold in separate transactions under the same order. But there is no line item number or any other unique identifier that differentiates between multiple entries of the same product and same order.

d. We must talk to source system experts to understand the business processes and consider enhancing the source data model to include a unique line item identifier if possible.

e. Data will be appended to the FACT table daily

<u>Step 3: Loading the data into the mid layer tables</u>

1. Loading data into DIMENSION tables

   - MERGE Statement is used to INSERT OR UPDATE the data in the DIMENSION tables

   - VALID_FROM and VALID_TO are used to track history in the SCD2 implementation

   - VALID_TO = '9999-12-31 00:00:00.000' represents the ACTIVE record

2. Loading data into FACT table

   - Data in the FACT is appended with each load

   - MERGE is used to ensure that if load is executed again with the same source data then duplicates are not inserted

   - While loading the FACT table, lookup to fetch the dimension keys are made such that it only looks up at the ACTIVE record in the DIMENSION table

<u>Step 4: To ensure high data quality in the "mid layer" tables and stability of the load process</u>

1. Data Quality Checks in the Staging Layer

   - Null Value Checks: Key fields like customer_id, order_id, product_id should not have null or missing values

   - Duplicate Data Checks: There should be no duplicate rows in STAGING.STG_ORDER, STAGING.STG_CUSTOMER, STAGING.STG_PRODUCT tables

   - Data Type Validation: Data must be stored with the correct data types (eg. numbers in numeric fields, dates in date fields, check for invalid date formats)

   - Business Logic checks: Data must follow the business logic (eg. sale_price should be positive, sale_percentage should not exceed 100%). Values like sale_price, coupon_value and sale_percentage must fall within reasonable boundaries

2. Load process stability

   - Incremental Loads: Instead of loading all data every time, implement incremental loads using CDC

   - Error Logging: Build logic to capture and log any errors during the ETL process

   - Batch re-runnability: Incase the batch load fails midway, the load process must be re-runnable without getting duplicate data errors. Proper rollback and retry mechanism must be developed to handle failures

3. Data Validations in CORE layer

   - Data in DIM and FACT tables must not contain duplicates. Use MERGE or UPSERT operations to deduplicate data or UNIQUE constraints on the table columns

   - Foreign keys in FACT_ORDER correctly map to the corresponding dimension tables DIM_CUSTOMER,DIM_PRODUCT etc.

4. Batch monitoring

- Monitor the time taken by the batch and ensure that it meets the SLA
- Automated alerts in case of failures or long running jobs

Step 5: Designing the 3rd layer of the Datawarehouse

1. Created PUBLISH schema in the DWH

The tables in this schema will be modeled using a datamart design and will be optimized for analytics and reporting

2. Created table CUSTOMER_BRAND_FACT in PUBLISH schema

This table will store aggregated purchase information at the customer-brand level and will allow to quickly filter customers who purchased specific brands within a given time period (eg. last 24 months)

3. Suggestions for the CRM department

   a. By analyzing the CUSTOMER_BRAND_FACT table, we can identify customers who have consistently bought products from certain brands. These loyal customers are good candidates for targeted brand-related offers.

   b. By querying the last_purchase_date in the CUSTOMER_BRAND_FACT table, we can find customers with no recent purchases. We can offer exclusive discounts or limited-time deals to reactivate those customers.

   c. By looking at total_purchase_amount in the CUSTOMER_BRAND_FACT table, we can find high value customers. We can create campaigns that offer these customers early product releases, personalized recommendations or loyalty rewards.

   d. The purchase_count and total_purchase_amount for specific brands in the CUSTOMER_BRAND_FACT table can help to design campaigns that offer bundle deals, discounts on similar products or product launches from the same brand.