**1. Overview**

This documentation provides a structured explanation of the dbt models created to transform and analyze marketing and sales data. It includes details on model purpose, key columns, assumptions, and additional notes.

**2. Data Sources**

Schema.yml was configured globally to pull the raw tables from specified database, schema and tables from redshift.

The following tables are used as input data:

* **marketing\_events**: Contains data on marketing campaigns.
* **sales\_transactions**: Stores all sales transactions.
* **product\_catalog**: Holds product-related details.

The profile was named as ‘**test**’ which would house all the tables, test and macros.

**3. DBT Models Structure**

In order to follow a medallion data architecture, 3 staging tables were materialized as views under the staging folder.

**3.1. stg\_sales\_transactions (Staging Model)**

* **Purpose**: Standardizes and cleans sales transaction data before transformation.
* **Columns**:
  + transaction\_id: Unique identifier for each transaction.
  + product\_id: Product associated with the transaction.
  + user\_id: Customer making the purchase.
  + transaction\_timestamp: Timestamp of the transaction.
  + revenue: Revenue generated from the transaction.
  + cost: Cost of the transaction.
* **Assumptions**:
  + All transactions are recorded in UTC.
  + No duplicate transaction IDs exist.

**3.2. stg\_marketing\_events (Staging Model)**

* **Purpose**: Prepares marketing data for use in downstream transformations.
* **Columns**:
  + event\_id: Unique ID of the marketing event.
  + user\_id: Customer associated with the event.
  + event\_type: Type of marketing event.
  + event\_timestamp: Timestamp of the event.
  + channel: Marketing channel used.
  + campaign: Campaign name.
  + cost: Cost associated with the event.
* **Assumptions**:
  + Marketing events have unique event\_id.
  + Costs are recorded in the same currency.

**3.3. stg\_product\_catalog (Staging Model)**

* **Purpose**: Standardizes product catalog data.
* **Columns**:
  + product\_id: Unique identifier for the product.
  + product\_name: Name of the product.
  + category: Category to which the product belongs.

**3.4. fact\_sales\_performance (Mart Model)**

* **Purpose**: Aggregates sales and marketing data to derive key performance indicators.
* **Columns**:
  + transaction\_year: Year of the transaction.
  + transaction\_month: Month of the transaction.
  + category: Product category.
  + unique\_customers: Count of distinct customers making purchases.
  + total\_revenue: Total revenue generated.
  + total\_cost: Total cost incurred.
  + total\_marketing\_cost: Sum of marketing expenses in the same period.
  + total\_marketing\_events: Number of marketing events in the period.
  + revenue\_to\_cost\_ratio: Ratio of revenue to cost.
* **Assumptions**:
  + Sales transactions and marketing events are recorded in the same time zone.
  + Marketing events are linked to sales based on matching user\_id and the same period.

**4. Data Quality Checks**

* **Ensure no duplicate transaction IDs** in stg\_sales\_transactions.
* **Check for null values in key columns** (product\_id, user\_id, etc.).
* **Validate foreign key relationships** (e.g., product\_id in sales should exist in product\_catalog).
* **Ensure revenue and cost are non-negative**.

**5. DBT Macro**

* **Purpose**: A macro to dynamically calculate revenue\_to\_cost\_ratio to improve reusability.
* **Usage**: Can be applied to different fact models also handle division by zero error.

6. Documentation

Once the whole dbt project is setup and we run dbt debug to find out any issue, we run the following commands to create a data lineage and documentation using the in built dbt feature.

*dbt docs generate*

*dbt docs serve*

As the raw tables are not available to me the doc was not generated in my test env.

N.B Everything was setup in the dbt project in dev environment, for production we will need to add the production details in profiles.yml also to make the dbt model for efficient set up incremental materialization where applicable specially for the transaction table.