**Assignment 1-Transforming Data from RAW.JAFFLE\_SHOP.ORDERS in dbt Cloud with Snowflake**

**Objective:**

**Transforming Data from RAW.JAFFLE\_SHOP.ORDERS in dbt Cloud with Snowflake**

In this assignment, you'll perform the following steps:

1. **Create the orders table** in the raw.jaffle\_shop schema.
2. **Insert sample data** into the orders table.
3. **Transform the data** in dbt Cloud, including:
   * Filtering out cancelled orders.
   * Calculating the total value of each order (quantity \* unit\_price).
   * Adding a flag for high-value orders, based on the total value.

**Step 1: Table Creation in Snowflake**

Let's start by creating the orders table in the raw.jaffle\_shop schema.

**SQL Code to Create the Table**

CREATE TABLE IF NOT EXISTS raw.jaffle\_shop.hks\_orders (

order\_id INT, -- Unique Order ID

customer\_id INT, -- Customer ID who placed the order

order\_date DATE, -- Date when the order was placed

product\_id INT, -- ID of the product ordered

quantity INT, -- Quantity of the product ordered

unit\_price FLOAT, -- Unit price of the product

order\_status STRING -- Status of the order ('completed' or 'cancelled')

);

**Step 2: Insert Sample Data into the Table**

Now, let's insert some sample data into the orders table to work with. This data will have a mix of completed and cancelled orders.

**SQL Code to Insert Sample Data**

INSERT INTO raw.jaffle\_shop.hks\_orders (order\_id, customer\_id, order\_date, product\_id, quantity, unit\_price, order\_status)

VALUES

(1, 101, '2024-11-01', 2001, 2, 10.0, 'completed'),

(2, 102, '2024-11-02', 2002, 1, 15.0, 'cancelled'),

(3, 103, '2024-11-03', 2003, 3, 20.0, 'completed'),

(4, 104, '2024-11-04', 2001, 4, 10.0, 'completed'),

(5, 105, '2024-11-05', 2004, 2, 30.0, 'cancelled');

This sample data includes both completed and cancelled orders, with varying quantities and prices.

**Step 3: Create dbt Model for Transformation**

Next, we will create a dbt model that:

1. Filters out cancelled orders.
2. Adds a calculated field for the total order value (quantity \* unit\_price).
3. Flags high-value orders, where the total value is greater than 50.0.

**Create the dbt Model: clean\_orders.sql**

1. **Create a new SQL file** in the models directory called clean\_orders.sql.
2. **Write the transformation SQL**.

{{ config(materialized='table') }} -- Materialization type is 'table'

WITH filtered\_orders AS (

-- Step 1: Filter out cancelled orders

SELECT

order\_id,

customer\_id,

order\_date,

product\_id,

quantity,

unit\_price,

order\_status

FROM {{ source('raw', 'jaffle\_shop.orders') }} -- Reference the source table in dbt

WHERE order\_status != 'cancelled' -- Exclude cancelled orders

)

-- Step 2: Transform the data by calculating total order value and flagging high-value orders

SELECT

order\_id,

customer\_id,

order\_date,

product\_id,

quantity,

unit\_price,

order\_status,

quantity \* unit\_price AS total\_order\_value, -- Calculated field for total order value

CASE

WHEN quantity \* unit\_price > 50 THEN TRUE -- Flag for high-value orders

ELSE FALSE

END AS high\_value\_order -- Flag indicating whether the order is high-value

FROM filtered\_orders;

**Step 4: Verify the Transformed Data in Snowflake**

After the dbt model runs, query the transformed data in Snowflake to verify the result.

SELECT \* FROM analytics.clean\_orders;

**Expected Output in the clean\_orders Table**

| **order\_id** | **customer\_id** | **order\_date** | **product\_id** | **quantity** | **unit\_price** | **order\_status** | **total\_order\_value** | **high\_value\_order** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 101 | 2024-11-01 | 2001 | 2 | 10.0 | completed | 20.0 | FALSE |
| 3 | 103 | 2024-11-03 | 2003 | 3 | 20.0 | completed | 60.0 | TRUE |
| 4 | 104 | 2024-11-04 | 2001 | 4 | 10.0 | completed | 40.0 | FALSE |

**Run command in Snowflake:**

drop table raw.jaffle\_shop.hks\_orders;

**Explanation:**

* **Cancelled orders** (ID 2 and 5) are excluded from the transformed table.
* The **total\_order\_value** is calculated as quantity \* unit\_price.
* The **high\_value\_order** flag is set to TRUE for orders where the total\_order\_value is greater than 50.0.