**Assignment 2- Jinja Macro for Employee Table**

**Objective:**

In this exercise, you will:

1. Create an employee table.
2. Write a **Jinja macro** that calculates the average salary of employees in a specific department.
3. Use the macro in a dbt model to create a report showing the average salary by department.
4. Include the CREATE TABLE and INSERT INTO commands for inserting data into the employee table.

**Step 1: Create the Employee Table**

Let's start by creating the employee table in the database, which includes information about employees such as their employee\_id, department\_id, salary, and hire\_date.

**SQL to Create Employee Table**

-- Create the employee table in your Snowflake schema (e.g., jaffel\_shop)

CREATE OR REPLACE TABLE jaffel\_shop.employee (

employee\_id INT,

department\_id INT,

salary DECIMAL(18, 2),

hire\_date DATE

);

This table structure includes:

* employee\_id: The ID of the employee.
* department\_id: The department to which the employee belongs.
* salary: The salary of the employee.
* hire\_date: The date when the employee was hired.

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

Next, let's insert some sample data into the employee table to use in our exercise.

**SQL to Insert Sample Data**

-- Insert sample data into the employee table

INSERT INTO jaffel\_shop.employee (employee\_id, department\_id, salary, hire\_date)

VALUES

(1, 101, 60000.00, '2020-01-15'),

(2, 101, 65000.00, '2021-03-22'),

(3, 102, 75000.00, '2020-07-11'),

(4, 102, 80000.00, '2019-09-05'),

(5, 103, 55000.00, '2022-04-12'),

(6, 103, 50000.00, '2022-06-19'),

(7, 104, 90000.00, '2018-11-25'),

(8, 104, 95000.00, '2017-12-30');

This query inserts 8 records into the employee table, representing employees from different departments with varying salaries and hire dates.

**Step 3: Define the Jinja Macro to Calculate Average Salary**

Now, let's define the **Jinja macro** that will calculate the average salary for employees within a specific department.

**File Location: macros/calculate\_avg\_salary.sql**

-- macros/calculate\_avg\_salary.sql

{% macro calculate\_avg\_salary(department\_id) %}

SELECT

department\_id,

AVG(salary) AS avg\_salary

FROM jaffel\_shop.employee

WHERE department\_id = {{ department\_id }}

GROUP BY department\_id

{% endmacro %}

**Explanation:**

* **Macro Definition**: The macro calculate\_avg\_salary takes one argument, department\_id, which specifies the department for which we want to calculate the average salary.
* **Dynamic SQL**: The {{ department\_id }} is used to dynamically inject the department ID into the SQL query.
* **Aggregation**: The SQL calculates the average salary for employees in the specified department using the AVG(salary) function.

**Step 4: Use the Macro in a dbt Model**

Now, we will create a dbt model that uses the calculate\_avg\_salary macro to generate a report of the average salary for each department.

**File Location: models/avg\_salary\_by\_department.sql**

-- models/avg\_salary\_by\_department.sql

{{ config(

materialized = 'table'

) }}

-- Call the macro for each department (e.g., 101, 102, 103, 104)

WITH avg\_salary\_report AS (

{{ calculate\_avg\_salary(101) }}

UNION ALL

{{ calculate\_avg\_salary(102) }}

UNION ALL

{{ calculate\_avg\_salary(103) }}

UNION ALL

{{ calculate\_avg\_salary(104) }}

)

SELECT \*

FROM avg\_salary\_report;

**Explanation:**

1. **Model Configuration**: The model is configured with materialized = 'table', meaning the result will be stored as a table in the database.
2. **Calling the Macro**: The macro calculate\_avg\_salary(department\_id) is called for each department (101, 102, 103, and 104). The UNION ALL is used to combine the results for all departments.
3. **Result**: The result of the WITH clause is stored in a CTE called avg\_salary\_report, which is then selected in the final SELECT statement.

**Step 5: Run the dbt Model**

To run the dbt model, use the following command:

dbt run --select avg\_salary\_by\_department

This command will:

1. Execute the calculate\_avg\_salary macro for each department.
2. Combine the results using UNION ALL.
3. Insert the result into the avg\_salary\_by\_department table in the Snowflake database.

**Step 6: Verify the Results**

After running the dbt model, you can query the avg\_salary\_by\_department table in Snowflake to verify the results:

SELECT \*

FROM jaffel\_shop.avg\_salary\_by\_department;

You should see a result similar to this:

| **department\_id** | **avg\_salary** |
| --- | --- |
| 101 | 62500.00 |
| 102 | 77500.00 |
| 103 | 52500.00 |
| 104 | 92500.00 |

**Conclusion**

In this exercise, you learned how to:

1. **Create a table** (employee) to store employee data.
2. **Insert sample data** into the employee table.
3. **Define a Jinja macro** (calculate\_avg\_salary) to calculate the average salary for a given department.
4. **Use the macro in a dbt model** to create a report that shows the average salary for each department.
5. **Run the dbt model** to materialize the results as a table.

This approach helps you modularize the logic of calculating average salary by using a reusable Jinja macro. You can easily extend this to work with more departments or additional metrics as needed.