14 November 2024 18:16

Jinja Template (dbt)

Control Statements	{% set variable_a = 10 %} {% if name == "Bard": %} ..... {% endif %} {% for i in range(variable_a) %} ..... {% endfor %}
Expressions	{{ 10 + 20 }} {{ variable_a }}
Texts	SELECT UNION
Comments	{# This is a comment #}

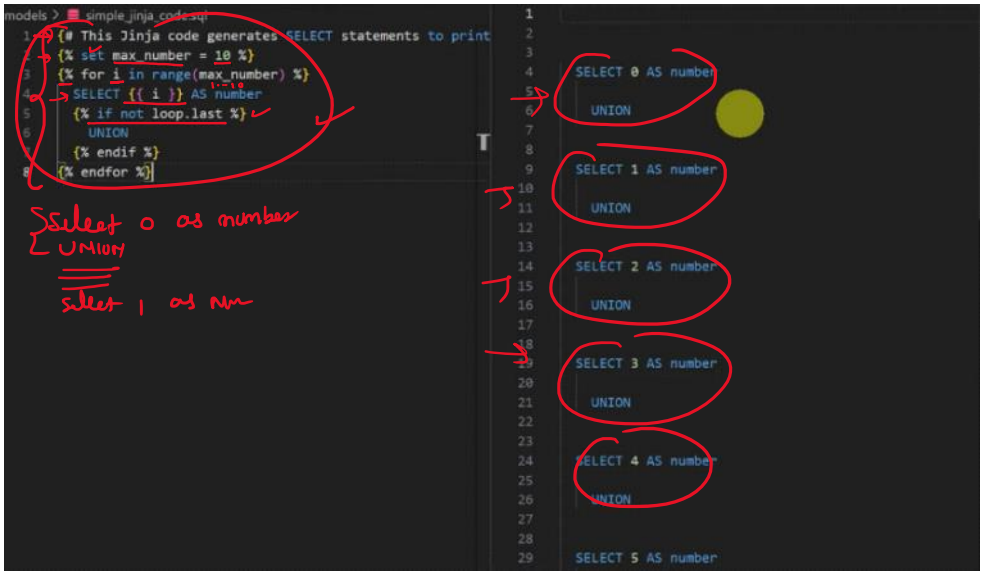


Table	Minimum Record Count needed for Proper Testing
customers	50
Employees	20
Stores	10
Suppliers	5
Products	100
OrderItems	1000
Orders	200

Testing

Select Count(*) from

```sql
-- Define the expected record counts for each table
{% set expected_counts = {
  'customers': 50,
  'Employees': 20,
  'Stores': 10,
  'Suppliers': 5,
  'Products': 100,
  'OrderItems': 1000,
  'Orders': 200
} %}

-- Test the count of records in each table
{% for table, expected_count in expected_counts.items() %}
  SELECT '{{ table }}' AS table_name,
         (SELECT COUNT(*) FROM {{ source('landing', table) }}) AS record_count,
         {{ expected_count }} AS expected_count
  WHERE (SELECT COUNT(*) FROM {{ source('landing', table) }}) < {{ expected_count }}
  {% if not loop.last %} UNION ALL {% endif %}
{% endfor %}
```

```sql
-- customers_record_count_check.sql
SELECT 'customers' AS table_name,
       (SELECT COUNT(*) FROM landing.customers) AS record_count,
       50 AS expected_count
WHERE (SELECT COUNT(*) FROM landing.customers) < 50;

-- Employees_record_count_check.sql
SELECT 'Employees' AS table_name,
       (SELECT COUNT(*) FROM landing.Employees) AS record_count,
       20 AS expected_count
WHERE (SELECT COUNT(*) FROM landing.Employees) < 20;

-- Stores_record_count_check.sql
SELECT 'Stores' AS table_name,
       (SELECT COUNT(*) FROM landing.Stores) AS record_count,
       10 AS expected_count
WHERE (SELECT COUNT(*) FROM landing.Stores) < 10;

-- Suppliers_record_count_check.sql
SELECT 'Suppliers' AS table_name,
       (SELECT COUNT(*) FROM landing.Suppliers) AS record_count,
       5 AS expected_count
WHERE (SELECT COUNT(*) FROM landing.Suppliers) < 5;
```

Python

Data Type	Ordered	Mutable	Example
Numbers	No	Yes	`123`, `1.23`, `1 + 2j`
Strings	No	No	`'This is a string'`, `"This is also a string"`
Booleans	No	No	`True`, `False`
Lists [ ]	Yes	Yes	`[1, 2, 3, 4, 5]`, `['a', 'b', 'c', 'd', 'e']`
Tuples ( )	Yes	No	`(1, 2, 3, 4, 5)`, `('a', 'b', 'c', 'd', 'e')`
Dictionaries { }	No	Yes	`{'key_1': 'value_1', 'key_2': 'value_2', 'key_3': 'value_3'}`, `{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}`
Sets	No	Yes	`{1, 2, 3, 4, 5}`, `{'a', 'b', 'c', 'd', 'e'}`

-- macros/calculate_growth.sql

```
{% macro calculate_growth(current_period_column, previous_period_column) %}
  CASE
    WHEN {{ previous_period_column }} != 0 THEN
      ({{ current_period_column }} - {{ previous_period_column }}) / {{ previous_period_column }} * 100
    ELSE
      NULL
  END
{% endmacro %}
```

-- models/quarterly_revenue.sql

```
select
  quarter,
  current_quarter_revenue,
  previous_quarter_revenue,
  {{ calculate_growth('current_quarter_revenue',
  'previous_quarter_revenue') }} as qoq_growth
from q_revenue_stg
```

-- models/monthly_revenue.sql

```
select
  month,
  current_month_revenue,
  previous_month_revenue,
  {{ calculate_growth('current_month_revenue',
  'previous_month_revenue') }} as mom_growth
from m_revenue_stg
```

```sql
SELECT
  city_code,
  city,
  month,
  avg_temp_fahrenheit,
  ROUND((avg_temp_fahrenheit - 32) * 5/9, 1)
 as avg_temp_celsius
FROM SLEEKMART_OMS.TRAINING.city_temperature
```

94 2

to_celsius.

to_Celsius(94, 2)

```
{% macro to_celsius(fahrenheit_column, decimal_places=1) %}
  ROUND((({{ fahrenheit_column }} - 32) * 5/9, {{ decimal_places }})
{% endmacro %}
```

macro_example_model.sql

```
SELECT
  city_code,
  city,
  month,
  avg_temp_fahrenheit,
```

```
1  SELECT
2    city_code,
3    city,
4    month,
5    avg_temp_fahrenheit,
6
```

```
city,
month,
avg_temp_fahrenheit,
{{ to_celsius('avg_temp_fahrenheit') }}
as avg_temp_celsius
FROM SLEEKMART_OMS.TRAINING.city_temperature
```

```
month,
avg_temp_fahrenheit,

ROUND((avg_temp_fahrenheit - 32) * 5/9, 1)

as avg_temp_celsius
FROM SLEEKMART_OMS.TRAINING.city_temperature
```

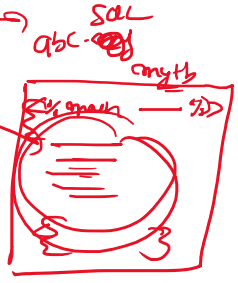
walmart

S.No	Source Tables	Target Tables
1	sales_us	profit_us
2	sales_uk	profit_uk
3	sales_india	profit_india
4	sales_canada	profit_canada
5	sales_germany	profit_germany
6	sales_france	profit_france
7	sales_japan	profit_japan
8	sales_mexico	profit_mexico
9	sales_russia	profit_russia
10	sales_china	profit_china

10 .sql

sales_us.sql

```
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_sold * unit_purchase_cost) as total_profit
FROM {{ source('training', 'sales_us') }}
GROUP BY sales_date
```



Sales_uk.sql

```
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_sold * unit_purchase_cost) as total_profit
FROM {{ source('training', 'sales_us') }}
GROUP BY sales_date
```

sales_us

```
{% macro generate_profit_model(table_name) %}
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_sold * unit_purchase_cost) as total_profit
FROM {{ source('training', table_name) }}
GROUP BY sales_date
{% endmacro %}
```

```
{{ generate_profit_model('sales_us') }}
```

```
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_sold * unit_purchase_cost) as total_profit
FROM SLEEKMART_OMS.training.sales_us
GROUP BY sales_date
```

```
{{ generate_profit_model('sales_us') }}
```

```
{{ generate_profit_model('sales_us') }}
```

```
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_so
FROM SLEEKMART_OMS.TRAINING.sales_us
GROUP BY sales_date
```

```
SELECT
  sales_date,
  SUM(quantity_sold * unit_sell_price) as total_revenue,
  SUM(quantity_sold * unit_purchase_cost) as total_cost,
  SUM(quantity_sold * unit_sell_price) - SUM(quantity_so
FROM SLEEKMART_OMS.TRAINING.sales_us
GROUP BY sales_date
```

```
{% macro aggregate_sales(months) %}
    {% set sql_parts = [] %}

    {% for month in months %}
        {% set total_column = 'SUM(sales_' ~ month ~ ') AS total_sales_' ~ month
%}
        {% set avg_column = 'AVG(sales_' ~ month ~ ') AS avg_sales_' ~ month %}

        {% do sql_parts.append(total_column) %}
        {% do sql_parts.append(avg_column) %}
    {% endfor %}
    -- Join the SQL parts into one string
    {% set final_sql = sql_parts | join(', ') %}

    -- Return the final SQL part
    {{ return(final_sql) }}
{% endmacro %}
```

Data Preview

	CUSTOMER_ID	CUSTOMER_NAME	EMAIL	ORDER_COUNT	TOTAL_ORDER_AMOUNT
1	1	Alice Johnson	alice@example.com	2	550.00
2	2	Bob Smith	bob@example.com	1	150.00
3	3	Charlie Brown	charlie@example.com	1	200.00

Data Preview

	CUSTOMER_ID	CUSTOMER_NAME	EMAIL	ORDER_COUNT	TOTAL_ORDER_AMOUNT
1	1	Alice Johnson	alice@example.com	2	550.00
2	2	Bob Smith	bob@example.com	1	150.00
3	3	Charlie Brown	charlie@example.com	1	200.00