

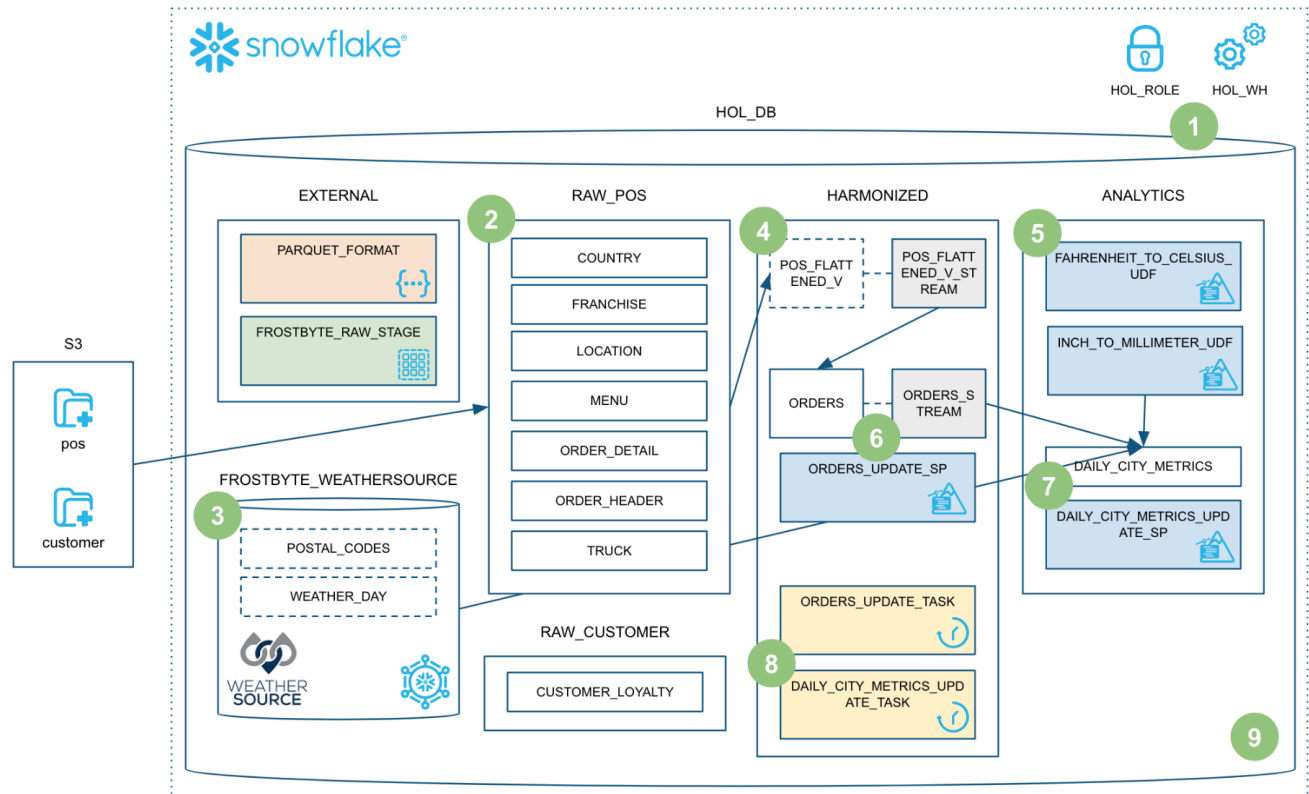


# Sun Life Snowpark Hands-On Lab

## Foundations for Advanced Analytics with Snowpark

### Overview

This HOL is a version of a [Snowflake Data Engineering Quickstart](#) -- customized for Sun Life



1 - Account configuration is managed by Sun Life. We'll talk through some details during the lab.

2 - Raw data for this lab has been loaded into the Sandbox account in advance.

3 - Snowflake offers a marketplace for data, where providers may offer free or paid data sets to consumers. In this case, you will be a consumer of weather data from [Weather Source](#).

Because ACCOUNTADMIN is required to provision new marketplace data sets, we'll go through this step as a group during the lab.

4 - The "HARMONIZED" schema depicted in the diagram is where the fun begins i.e. this is where you will start creating Snowflake objects.

**BE SURE TO PREFIX EVERY SINGLE OBJECT YOU CREATE WITH YOUR**  
**<first name>\_<last name>**

Because all users are sharing the schemas, code will fail if you don't update the object names.

**5a** - The “ANALYTICS” schema depicted in the diagram is sometimes called a “Consumption Layer”. Just like in the previous section, be sure to prefix every database object with your `<first name>_<last name>`


**5b** - You will create a UDF (user-defined function) which is created and deployed once, in one language, and can be called many times by any permitted user in any language. In this case, unit conversion may differ in precision or rounding from one calculation to the next and a UDF can prevent that!

**6** - The ORDERS\_UPDATE\_SP [Stored Procedure](#) will merge the POS\_FLATTENED\_V\_STREAM into the ORDERS table. The complexity of combining raw tables is contained in the POS\_FLATTENED\_V view... While the high water mark is managed automatically by the ‘stream’ which is built on top of the view. That way, in order to merge all of the incremental changes in raw data into a standardized format, all the stored procedure needs to do is a simple MERGE from the stream into the ORDERS table. It also runs some checks to make sure the tables exist.

**7** - The DAILY\_CITY\_METRICS\_UPDATE\_SP Stored Procedure is similar to the Orders Update Stored Proc, but contains more code within the stored procedure itself. We also include calculations in this stored procedure, which leverage the functions we created earlier for unit conversion. This is a great example of abstracting some of the complexity of in-line code by reusing a pre-built function.

**8** - Job Orchestration -- This is where we tie all the components together. The job orchestration script is fairly simple, using Snowflake tasks to schedule operations. The tasks call our stored procedures created in steps 6 and 7, which makes job orchestration very straightforward.

# Lab Execution

1. Claim a Username from This Spreadsheet:  Sun Life HOL Usernames

**USE THE LOGIN FROM THIS SPREADSHEET, NOT SSO**

2. Navigate to this URL and login:

<https://sunlife-corpdpep.snowflakecomputing.com/>

ROLE: SNOWPARK\_HOL\_ROLE

WAREHOUSE: SNOWPARK\_HOL\_VWH

3. Navigate to the [Lab Scripts in Github](#)

## Step 1: Snowflake Setup

### As a group...

Account configuration is managed by account admins. This includes, but is not limited to the creation of the account itself, warehouses, roles, databases and schemas. A Role-based Access Control framework has been developed centrally to align permissions to functional roles within Sun Life.

**Sun Life's best practices for snowflake can be found at the following link:**

[ [Sun Life Sharepoint for Snowflake](#) ]

<https://sunlifefinancial.sharepoint.com/:f:/r/sites/DT/DNA/DMS/DBServ/Snowflake/HOL?csf=1&web=1&e=yzvpe5>

## Step 2: Load Raw

Raw data provisioning has been done in advance. The objective of this class is to familiarize participants with working with data *already loaded* to Snowflake.

## Step 3: Load Weather

*Prior to the Lab, Weather Data (Weather Source LLC) has already been from the Snowflake Data Marketplace. Imported Data sets can be viewed in Snowsight using the Data Products tab*

That's it... we don't have to do anything from here to keep this data updated. The provider will do that for us and data sharing means we see the latest data they have published.

## Step 4: Create the POS View and Stream

- Open 04\_create\_pos\_view\_python.py and paste the contents into a new **Python** worksheet.
- If you like, you can name the worksheet '04\_create\_pos\_view\_python'

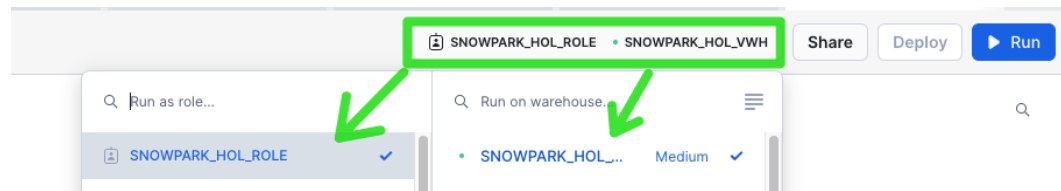


Sun Life

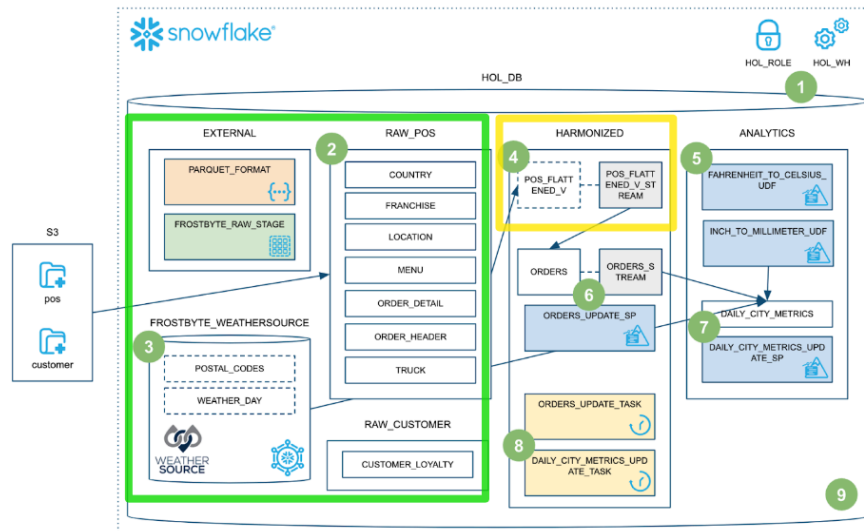


snowflake®


- Select your warehouse and role from the dropdown menu



- Update the code with your `<firstname>_<lastname>` anywhere that you see `<firstname>_<lastname>`
- This view contains all of the logic to join the individual raw tables into a more consumable view. But more importantly simplifies the pipeline logic.
  - That's because the Stream we just created will capture incremental changes on ANY of the underlying tables. So rather than capturing operations against 7 different tables individually, we only need to create one stream on the view.
  - Still not sure what the "Stream" is in Snowflake? Just raise your hand!
- Run the Script (press "Run")



## Step 5: Deploy Fahrenheit to Celsius UDF

- Open 05\_fahrenheit\_to\_celsius\_udf.sql and paste the contents into a new **SQL** worksheet
- If you like, you can name the worksheet '05\_fahrenheit\_to\_celsius\_udf'
- Select your SNOWPARK\_HOL\_VWH warehouse and SNOWPARK\_HOL\_ROLE from the dropdown menu
- Run all of the statements in order. To run them all automatically, highlight the entire script and click the play button  or Ctrl+Enter.
- Note that the final statement in the script tests the UDF you just deployed. Feel free to replace the value in parentheses with any value to see the conversion from Fahrenheit to Celsius.



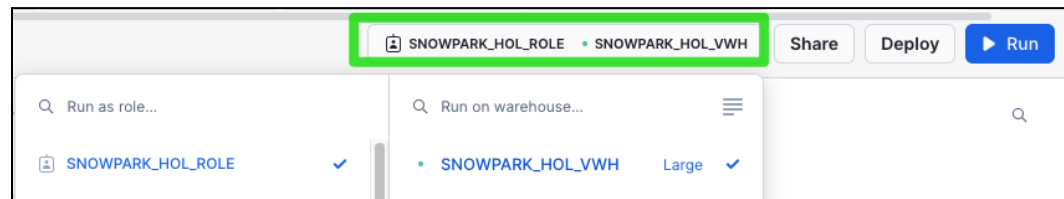
- This UDF, `<name>_FAHRENHEIT_TO_CELSIUS_UDF`, is now available for use by anyone with the `SNOWPARK_HOL_ROLE` role.

## Step 5a: CHALLENGE -- Deploy a Inch to Millimeter UDF

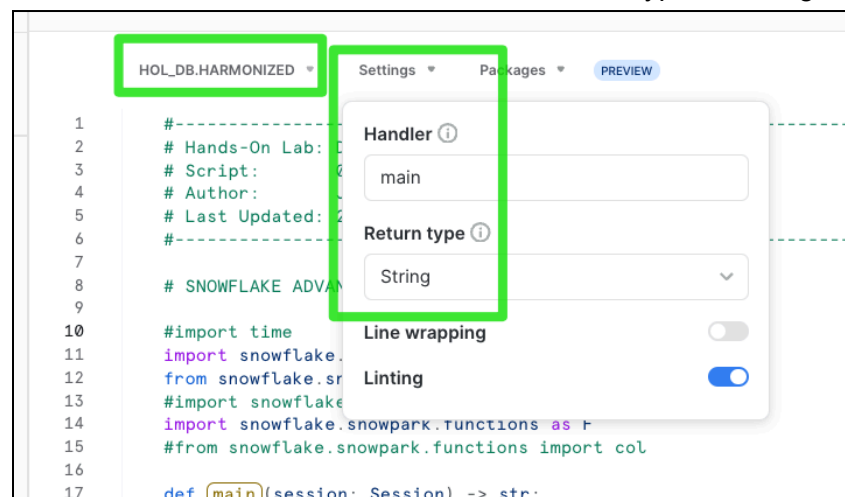
- Using the Fahrenheit to Celsius UDF script as a template, create your own UDF called `INCH_TO_MILLIMETER_UDF` which takes a float `x` as an input and returns a float `x*25.4` as an output, i.e. the conversion of your inches into mm.
- If you get stuck or want to skip the challenge just raise your hand and we'll give you the script.

## Step 6: Deploy Orders Update Stored Procedure

- Open `06_orders_update_sp.py` and paste the contents into a new **Python** worksheet
- If you like, you can name the worksheet '`06_orders_update_sp`'
- Choose your masterclass role and warehouse from the dropdown menu



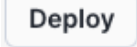
- Select the `SNOWPARK_HOL_DB.HARMONIZED` schema
- Ensure that the Handler is "main" and the return type is "String":



- Update all of the object names to use your username prefix
- Rather than running this script, we want to deploy it as a Python UDF. This is a handy feature Snowflake has created to simplify the process of deploying Stored Procedures -- rather than an explicit `CREATE PROCEDURE` with all the associated syntax requirements, you can simply write a script, configure the settings, test/update and then deploy it directly as a procedure! Using the "Create

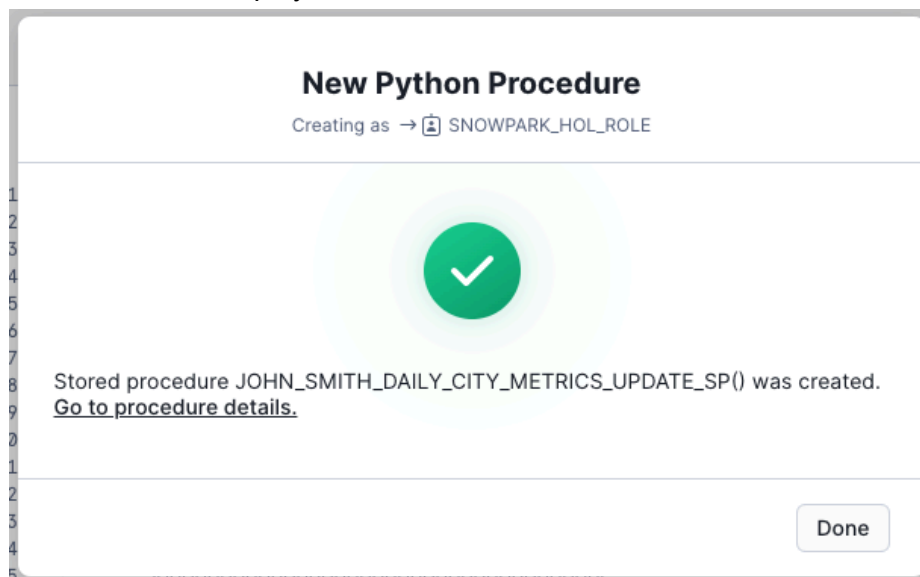


stored procedure” is nice for programmatic deployment via CI/CD, but this UI method is easier for development.

- i. Click 
- ii. Enter the procedure name  
“<firstname>\_<lastname>\_ORDERS\_UPDATE\_SP”
- iii. Select “main” as the handler
- iv. Click “Deploy”
- v. Optionally, once deployed, click “Go to procedure details” to see what you’ve created (You’ll need to navigate back to your worksheets after).

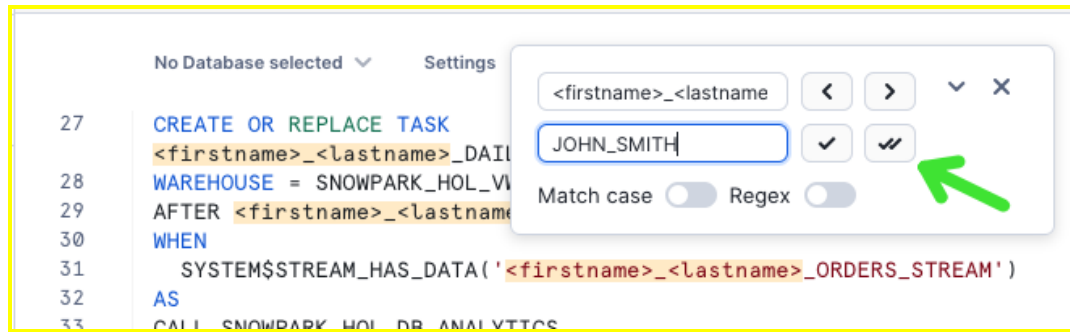
## Step 7: Daily City Metrics Stored Procedure

- Open 07\_daily\_city\_metrics\_update\_sp.py and paste the contents into a new **Python** worksheet
- Choose your HOL role and warehouse from the dropdown menu
- Once again, ensure that the Handler is “main” and the return type is “String”
- Select the HOL database, this time choose the **ANALYTICS schema**.
- Once again, update the object names with your user prefix
- Deploy the script as a stored procedure
  - i. Click “Deploy”
  - ii. Enter the procedure name  
“<firstname>\_<lastname>\_DAILY\_CITY\_METRICS\_UPDATE\_SP”
  - iii. Select “main” as the handler
  - iv. Click “Deploy”



## Step 8: Orchestrate Jobs using Snowflake Tasks

- Open 08\_orchestrate\_jobs.sql and paste the contents into a **SQL worksheet**
- Again, be sure to prefix every instance of <first name>\_<last name>



- We are only creating two objects in this step -- the two tasks which will run automatically to execute the entire data pipeline.
- Update your username prefix in the script
- **Run the entire script** to;
  - i. Set the warehouse and schema (HARMONIZED)
  - ii. Create the two tasks
  - iii. Resume the daily city metrics task which will run every time the orders update task runs, and
  - iv. Run the orders update task! Note that the orders update task doesn't have a trigger. For this exercise, we will only trigger it manually so that we can control the data flow. Generally, it would be scheduled with a cron schedule to run every minute, hour, day etc. or whatever interval we expect new data to come in.
    - **Monitor your task!** You just kicked it off with the script.
      - Read the comments in the script in order to understand how to monitor tasks that are running
    - While the task is running, feel free to take a break before we rejoin the group all together for the rest of the lab. It should take a little more than 5 minutes.

## Step Ω: Rejoin the group and watch a live pipeline execution!

- Because we're working on a shared account, we'll sync up as a group to watch what happens when incremental transactional data lands in the raw zone.
- **Admin to run incremental load \*\*\***

```
INSERT INTO SNOWPARK_HOL_DB.RAW_POS.ORDER_DETAIL
SELECT * FROM SNOWPARK_HOL_DB.RAW_POS.ORDER_DETAIL_INCREMENTAL;
```
- Once the data has landed....
  - i. Run the following command...
 

```
SELECT COUNT(1) FROM
      <first name>_<last
name>_POS_FLATTENED_V_STREAM;
```

 ... and note the count of incremental records loaded into the raw zone

- ii. Run the following command...

```
SELECT COUNT(1) FROM  
    <first name>_<last name>_DAILY_CITY_METRICS
```

... And note how many records there are in the consumption table

- iii. Re-execute your ORDERS\_UPDATE\_TASK

```
EXECUTE TASK <first name>_<last name>_ORDERS_UPDATE_TASK;
```

- iv. Count the records in the stream again

```
SELECT COUNT(1) FROM  
    <first name>_<last name>_POS_FLATTENED_V_STREAM
```

... Where did the records go???

- v. Do another count from the consumption table

```
SELECT COUNT(1) FROM  
    <first name>_<last name>_DAILY_CITY_METRICS
```

... Oh, there they are!



## What Next??

- You are now equipped with the foundation to develop a myriad of use cases on Snowflake using any of the libraries found in [Snowflake's Anaconda Repo](#).
- Or choose your own adventure from a selection of >100 Quick Start guides
  - [Snowflake Quickstart Guides](#)

