8 Snowflake Functions

8.1 Lab Introduction

8.1.1 Purpose

The purpose of this lab is to familiarize you with a few of the functions in Snowflake's large, built-in function library.

8.1.2 What you'll learn

- How to work with scalar functions
- How to work with regular and windowing aggregate functions
- How to use table and system functions
- How to use FLATTEN to work with semi-structured (JSON) data

8.1.3 How to complete this lab

In order to complete this lab, you can type the SQL commands below directly into a worksheet. It is not recommended that you cut and paste from the workbook pdf as that sometimes results in errors.

You can also use the SQL code file for this lab that was provided at the start of the class. You would simply need to open it in TextEdit (Mac) or Notepad (Windows), and then copy and paste the SQL code directly into a worksheet.

8.2 Scalar Functions

8.2.1 Open a worksheet and name it *Functions* and set the context:

```
USE ROLE TRAINING_ROLE;
CREATE WAREHOUSE IF NOT EXISTS LEARNER_WH;
USE WAREHOUSE LEARNER_WH;
CREATE DATABASE IF NOT EXISTS LEARNER_DB;
USE LEARNER_DB.PUBLIC;

ALTER SESSION SET USE_CACHED_RESULT = FALSE;
```

8.2.2 Run the statement below to convert c_name to uppercase.

```
SELECT c_name, UPPER(c_name)
    FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."CUSTOMER";
```


8.2.3 Run the following statements using the conditional function IFF to see what it does:

```
SELECT
    o_orderkey,
    o_totalprice,
    o_orderpriority,
    IFF(o_orderpriority like '1-URGENT', o_totalprice * 0.01, o_totalprice *
       0.005)::number(16,2)       ShippingCost
FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."ORDERS";
```

8.2.4 Use a CASE expression to print out text for preferred customer:

```
SELECT (c_salutation || ' ' || c_first_name || ' ' || c_last_name) AS full_name,
    CASE
        WHEN c_preferred_cust_flag LIKE 'Y'
            THEN 'Preferred Customer'
        WHEN c_preferred_cust_flag LIKE 'N'
            THEN 'Not Preferred Customer'
        END AS customer_status
FROM "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."CUSTOMER"
LIMIT 100;
```

8.2.5 Use the following statements to generate data. Note that `random()` with no argument generates a different number every time. If you provide a seed value, for example, `random(12)`, the same value will be returned every time:

```
SELECT RANDOM() AS random_variable;

SELECT RANDOM(100) AS random_fixed;
```

8.2.6 Run this query to use some time and date functions:

```
SELECT CURRENT_DATE(), DATE_PART('DAY', CURRENT_DATE()), CURRENT_TIME();
```

8.2.7 Snowflake has many string functions. Here is one you can run:

```
SELECT STRTOK_TO_ARRAY(query_text)
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE query_text LIKE 'select%'
AND query_text NOT LIKE 'select 1%'
LIMIT 5;
```

8.2.8 Run this statement to change an array back to a string with a separator:


```
SELECT STRTOK_TO_ARRAY(query_text),
       ARRAY_TO_STRING(STRTOK_TO_ARRAY(query_text),'#')
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
WHERE query_text LIKE 'select%'
AND query_text NOT LIKE 'select 1%'
LIMIT 5;
```

8.3 Use Regular and Windows Aggregate Functions

Aggregate functions work across rows to perform mathematical functions such as MIN, MAX, COUNT, and a variety of statistical functions.

Many of the aggregate functions can work with the OVER clause enabling aggregations across a group of rows. This is called a WINDOW function. This capability was shown earlier.

In this section you will work with some of the aggregate window functions.

8.3.1 Use WINDOW aggregate functions:

```
SELECT DAYOFMONTH(start_time), start_time, end_time,  warehouse_name,
   credits_used, SUM(credits_used)
    OVER (PARTITION BY DAYOFMONTH(start_time), warehouse_name
    ORDER BY DAYOFMONTH(start_time), warehouse_name )
AS day_tot_credits
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
    ORDER BY warehouse_name;
```

8.3.2 Query the query history to produce numeric aggregates on query total times:

```
SELECT MONTH(qh.start_time) AS "month", DAYOFMONTH(qh.start_time) AS dom,
qh.warehouse_name,
AVG(qh.total_elapsed_time),
MIN(qh.total_elapsed_time),
MAX(qh.total_elapsed_time),
STDDEV(qh.total_elapsed_time)
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY qh
WHERE query_text LIKE 'select%'
AND query_text NOT LIKE 'select 1%'
GROUP BY "month", dom, qh.warehouse_name
ORDER BY "month", dom, qh.warehouse_name;
```

8.4 Use TABLE and System Functions

Table functions return a set of rows instead of a single scalar value. Table functions appear in the FROM clause of a SQL statement and cannot be used as scalar functions.

System functions are functions that allow you to execute actions in the system, or that return information about queries or the system itself.

8.4.1 Use a table function to retrieve 1 hour of query history:

```
SELECT * FROM TABLE(information_schema.query_history
      (DATEADD('hours', -1, CURRENT_TIMESTAMP()), CURRENT_TIMESTAMP()))
ORDER BY start_time;
```

8.4.2 Use the RESULT_SCAN function to return the last result set:

```
SELECT * FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()));
```

8.4.3 Use the SHOW command with the RESULT_SCAN function to have SQL generate a list of commands to describe tables:

```
SHOW TABLES;
SELECT CONCAT('DESC ',"name",';')
FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()));
```

8.4.4 Note that Snowflake's SQL is generally case-insensitive: it changes everything to upper case unless it is enclosed in quotes. In this example, the "name" column must be lower case, so it is quoted.

8.4.5 Use the SYSTEM$WHITELIST function to see information on hosts that should be unblocked for Snowflake to work. Click on the item in the first row to see what the column contains:

```
SELECT SYSTEM$WHITELIST();
```

8.4.6 For an easier-to-read version, use the FLATTEN table function to flatten the data, which is in a semi-structured format:

```
SELECT VALUE:type AS type,
       VALUE:host AS host,
       VALUE:port AS port
FROM TABLE(FLATTEN(INPUT => PARSE_JSON(system$whitelist())));
```

8.4.7 Suspend and resize the warehouse

```
ALTER WAREHOUSE LEARNER_WH SET WAREHOUSE_SIZE=XSmall;
ALTER WAREHOUSE LEARNER_WH SUSPEND;
```


8.5 Key Takeaways

- Snowflake has many of the functions you are already accustomed to using in other software programs.
- You can query the query history table (SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY) to produce numeric aggregates on query total times.
- Table functions return a set of rows rather than a scalar value.
- System functions are functions that allow you to execute actions in the system, or that return information about queries or the system itself.

