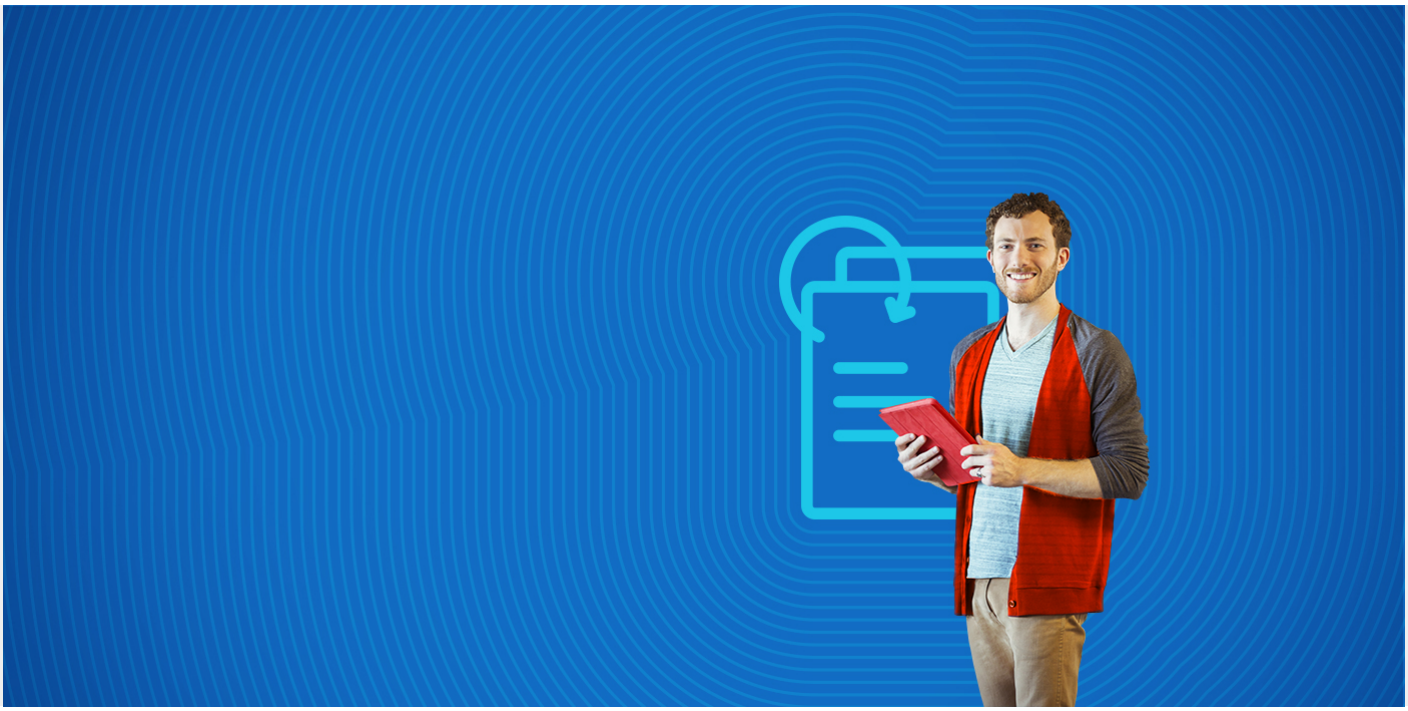
USING STORED PROCEDURES IN SNOWFLAKE



Snowflake supports stored procedures. Stored procedures let you write a series of commands and store them for later use.

(This tutorial is part of our Snowflake Guide; navigate to other tutorials using the right-hand menu.)

When to use a stored procedure

We've previously covered user defined functions (UDFs), which you use on a column. A stored procedure runs by itself. In other words, it goes off and "does something," like update a table. That's why stored procedures are good for batch-like actions.

You can also conditionally tie stored procedures to database events, not unlike what's called a trigger in other database products.

Data engineers can make a data pipeline with stored procedures, too. (A subject we will explore in depth in following posts.)

JavaScript for stored procedures

Snowflake stored procedures must be written in JavaScript. It makes sense that you must use a programming language beside SQL, since SQL does not support variable assignment. You would create variables to run calculations, etc.

Don't worry if you don't know JavaScript—you can simply copy boilerplate code and put SQL into the proper location. The SQL is really the only part that varies, for the most part. So, there is not much you need to understand.

Snowflake JavaScript is bare bones JavaScript. It does not let you import libraries that are external to the language. So, you can create arrays, variables, simple objects and there is error handling. But you could not, for example, use math functions form the Math library

Line by line tutorial: Stored procedure

We will put one simple example here and explain each line of the code. (So, you don't need any sample data.)

Look at the function below. Note the following:

- You put parameters to the function **functionname(parameters type)**
- You call the function by writing **call functionname(parameters)**.
- The function must return some value, even if it is just doing an update. Otherwise you will get the error NULL result in a non-nullable column. This is because the worksheet editor in Snowflake needs something to display, even if it's a null (Nan) value.
- The basic procedure is to use **execute()** to run SQL code that you have stored in a string. Database programmers know that creates what is called a **resultset**. So, you need to pull the first returned value into scope by calling **next()**. There is a result set with a select statement, delete, insert, and even update—even though you would not expect those to return any values.
- If the SQL statement returns more than one row, like in a SELECT, you would use a while **(rs.next())** to loop through the results.
- For whatever reason the parameters can be lowercase but must be uppercase inside the JavaScript code. You will get an error if you try to use lowercase letters.

```
create or replace procedure setprice(ORDERNUMBER varchar(100))
    returns float
    not null
    language javascript
    as
    $$
    sql_command = "update orders set price = 2 where ordernumber = " +
ORDERNUMBER   ;
    var stmt = snowflake.createStatement(
           {
           sqlText: sql_command
           }
        );
    var res = stmt.execute();
    res.next()
    price = res.getColumnValue(1);
    return price;
    $$
    ;

 call setprice(489)
```

Additional resources

For more tutorials like this, explore these resources:

- BMC Machine Learning & Big Data Blog
- How To Import Amazon S3 Data to Snowflake
- Snowflake Lag Function and Moving Averages
- Amazon Braket Quantum Computing: How To Get Started