SQL INSERT ... SELECT and CTAS Statements

As you should know by now, the SELECT statement in SQL returns a result set. Typically, you either view this result set, or else store it in a file or in memory on your local computer where you might use it to generate a report or data visualization. However, you can do more with the SELECT statement than just view the results or store them locally. The result set from a SELECT statement has the same basic structure as a table, so you can use a result set to materialize a new table. As you'll learn in this reading, it is possible to do this with a single command. This command allows you to save the result of a query to a table, so that you can later run another query to analyze or retrieve that result.

The way to do this is by combining together an INSERT and a SELECT into a single command. Using an INSERT statement, you can specify the name of the destination table, followed by a SELECT statement. This compound type of statement is known as an INSERT ... SELECT statement. Hive or Impala executes the SELECT statement then saves the results into the specified table. Note that the destination table must already exist. If you want to replace any existing data, use INSERT OVERWRITE; if you want to retain any existing data, use INSERT INTO.

For example, you might want to create a new table, chicago_employees,that contains only the rows of the employees table for employees in the Chicago office (office_id = 'b'). After you create this table (perhaps using LIKE employees in the CREATE TABLE command), the following command will populate the table with the desired rows. Any previously existing records in the chicago_employees table are deleted.

```
INSERT OVERWRITE chicago_employees

    SELECT * FROM employees WHERE office_id='b';
```

The line breaks and indentation used in these examples are all optional. The indentation serves to show that the SELECT statement is actually part of the INSERT statement.

As mentioned above, an INSERT … SELECT statement requires that the destination table already exists. But there is a different statement that does not require this: the CREATE TABLE AS SELECT (CTAS) statement. A CTAS statement creates a table and populates it with the result of a query, all in one command.

For example, the following CTAS statement creates the chicago_employees table and loads the same data as the INSERT … SELECT command above, but it does it all in one command.

```
CREATE TABLE chicago_employees AS

    SELECT * FROM employees WHERE office_id='b';
```

CTAS effectively combines a CREATE TABLE operation and an INSERT … SELECT operation into a single step. The column names and data types for the newly created table are determined based on the names and types of the columns queried in the SELECT statement.

However, the other attributes of the newly created table, like the delimiter and storage format, are *not* based on the format of the table you're querying; you must specify these attributes, or else the newly created table will use the defaults. So in the example here, since there is no ROW FORMAT clause, Hive or Impala will use the default field delimiter, which is the ASCII Control+A character. If you wanted comma-delimited files, you would need to include a ROW FORMAT clause. You must put this (and any other clauses that specify properties of the new destination table) *before* the AS keyword, as shown in the example below.

```
CREATE TABLE chicago_employees

    ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

AS

    SELECT * FROM employees WHERE office_id='b';
```

With both CTAS and INSERT … SELECT, you could select only some columns to include in the new table by listing only the desired columns in the SELECT list. For example, since all the records are in the same office, the office_id column isn't really needed. You could use:

```
CREATE TABLE chicago_employees AS
    SELECT empl_id, first_name, last_name, salary
        FROM employees
        WHERE office_id='b';
```

Note that if the data in the employees table is updated *after* the chicago_employees table is created, then the data in the chicago_employees table would be stale—any new employees in Chicago are *not* automatically added to the chicago_employees table. So if you are using a CTAS or INSERT … SELECT statement to create a derivative table that you want kept up to date with the original table, then you need to set up some process to keep it up to date. For example, you could schedule a job that runs an INSERT OVERWRITE … SELECT statement to repopulate the derivative table nightly, so the data there is never more than 24 hours stale.

If you complete the optional honors lessons in Week 6 of this course, you will learn another way to take a query and turn it into something that can itself be queried like a table: using a *view*. The main difference is that a view in Hive and Impala does not materialize (store) the results of the query like an INSERT … SELECT statement does. See Week 6 for more about this.

Try It!

1. Use the information above to create the chicago_employees table in two steps—create the table using LIKE employees (so the office_id column will be included) and populate it with the INSERT … SELECT statement above. Verify that the new table has only two rows, for Virginia and Luzja.

2. Drop the chicago_employees table (deleting the data—delete it manually if you created the table as an EXTERNAL table), and then recreate it *without* the office_id column, using a CTAS statement. Again verify that the new table has the two rows.