6 Data Loading and Unloading

6.1 Lab Introduction

6.1.1 Purpose

The purpose of this lab is to introduce you to data loading and data unloading.

6.1.2 What you'll learn

- How to load data from a file in an external stage into a table using the COPY INTO command.
- How to define a GZIP file format.
- How to review the properties of a stage.
- How to load a GZipped file from an external stage into a table
- How to unload table data into a Table Stage in Pipe-Delimited File format

6.1.3 How to complete this lab

In order to complete this lab, you can key 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 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 code directly into a worksheet.

6.2 Loading Data from an External Stage into a Table Using COPY INTO

In this exercise you will learn how to load a file from an external stage into a table using the COPY INTO command.

6.2.1 Scenario

You are a data engineer at Snowbear Air. You need to create and populate a table that will be used in reporting. The table will be called REGION and you will populate it from a pre-existing file (`region.tbl`) in an external stage. This headerless file is pipe-delimited and contains the following five rows:

```
0|AFRICA|lar deposits. blithely final packages cajole. regular waters are final
   requests. regular accounts are according to |
1|AMERICA|hs use ironic, even requests. S|
2|ASIA|ges. thinly even pinto beans ca|
3|EUROPE|ly final courts cajole furiously final excuse|
4|MIDDLE EAST|uickly special accounts cajole carefully blithely close requests.
   carefully final asymptotes haggle furiousl|
```

NOTE: There is a delimiter at the end of every line, which by default is interpreted as an additional column by the COPY INTO statement.

Let's get started!

6.2.2 Create a new folder called Data Loading and Unloading

6.2.3 Create a new worksheet named *Load Structured Data.*

6.2.4 Use the following SQL to 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;
```

6.2.5 Execute all of the CREATE TABLE statements:

```
CREATE OR REPLACE TABLE REGION (
       R_REGIONKEY NUMBER(38,0) NOT NULL,
       R_NAME      VARCHAR(25)  NOT NULL,
       R_COMMENT   VARCHAR(152)
);
```

6.2.6 Create a file format called `MYPIPEFORMAT`, that will read the pipe-delimited `region.tbl` file:

```
CREATE OR REPLACE FILE FORMAT MYPIPEFORMAT
  TYPE = CSV
  COMPRESSION = NONE
  FIELD_DELIMITER = '|'
  FILE_EXTENSION = 'tbl'
  ERROR_ON_COLUMN_COUNT_MISMATCH = FALSE;
```

6.2.7 Create a file format called `MYGZIPPIPEFORMAT` that will read the compressed version of the region.tbl file. It should be identical to the `MYPIPEFORMAT`, except you will set `COMPRESSION = GZIP`.

```
CREATE OR REPLACE FILE FORMAT MYGZIPPIPEFORMAT
 TYPE = CSV
 COMPRESSION = GZIP
 FIELD_DELIMITER = '|'
 FILE_EXTENSION = 'tbl'
 ERROR_ON_COLUMN_COUNT_MISMATCH = FALSE;
```

6.3 Load the *region.tbl* File

The files for this task have been pre-loaded into a location on AWS. The external stage that points to that location has been created for you. The stage is in the TRAININGLAB schema of the TRAINING_DB database. In this task

you will review the files in the stage, and load them using the file formats you created.

6.3.1 Review the properties of the stage:

```
DESCRIBE STAGE TRAINING_DB.TRAININGLAB.ED_STAGE;
```

NOTE: The file format defined in the stage is not quite right for this data. In particular, the field delimiter is set to a comma. You have two choices - you could either modify they file format definition in the stage itself, or you could specify a different file format with the COPY INTO command. You will use your MYPIPEFORMAT file format.

6.3.2 Confirm the file is in the external stage with the list command:

```
LIST @training_db.traininglab.ed_stage/load/lab_files/ pattern='.*region.*';
```

6.3.3 Load the data from the external stage to the REGION table, using the file format you created in the previous task:

```
COPY INTO REGION
FROM @training_db.traininglab.ed_stage/load/lab_files/
FILES = ('region.tbl')
FILE_FORMAT = (FORMAT_NAME = MYPIPEFORMAT);
```

6.3.4 Select and review the data in the REGION table, either by executing the following command in your worksheet or by using **Preview Data** in the sidebar:

```
SELECT * FROM REGION;
```

6.4 Loading a GZip Compressed File on an External Stage into a Table

The scenario for this activity is fundamentally the same as the previous activity. The difference is that you will load the REGION Table from a gzip compressed file that is in the external stage. You will re-use the MYGZIPPIPEFORMAT file format you created in the previous part of this lab.

For these next steps, you will use a smaller warehouse.

6.4.1 Empty the REGION Table in the PUBLIC schema of **LEARNER**_DB:

```
TRUNCATE TABLE region;
```


6.4.2 Confirm that the `region.tbl.gz` file is in the external stage:

```
LIST @training_db.traininglab.ed_stage/load/lab_files/ pattern='.*region.*';
```

6.4.3 Reload the REGION table from the region.tbl.gz file. Review the syntax of the COPY INTO command used in the previous task. Specify the file to COPY as 'region.tbl.gz'.

```
COPY INTO region
FROM @training_db.traininglab.ed_stage/load/lab_files/
FILES = ('region.tbl.gz')
FILE_FORMAT = ( FORMAT_NAME = MYGZIPPIPEFORMAT);
```

6.4.4 Query the table to confirm the data was successfully loaded:

```
SELECT * FROM region;
```

6.5 Unloading table data into a Table Stage in Pipe-Delimited File format

This activity is essentially the opposite of the previous two activities. Rather than load a file into a table, you are going to take the data you loaded and unload it into a file in a table stage.

6.5.1 Open a new worksheet and set the context as follows:

```
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;
```

6.5.2 Create a fresh version of the `REGION` table with 5 records to unload:

```
create or replace table REGION as
select * from SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.REGION;
```

6.5.3 Unload the data to the REGION table stage.

Remember that a table stage is automatically created for each table. Use the slides, workbook, or Snowflake documentation for questions on the syntax. You will use MYPIPEFORMAT for the unload:

```
COPY INTO @%region
FROM region
FILE_FORMAT = (FORMAT_NAME = MYPIPEFORMAT);
```


6.5.4 List the stage and verify that the data is there:

```
LIST @%region;
```

6.5.5 Remove the file from the REGION table's stage:

```
REMOVE @%region;
```

6.6 Use a SQL statement containing a JOIN to Unload a Table into an internal stage

This activity is essential the same at the previous activity. The difference is that you are going to unload data from more than one table into a single file.

6.6.1 Do a SELECT with a JOIN on the `REGION` and `NATION` tables. You can JOIN on any column you wish. Review the output from your JOIN.

```
SELECT *
FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."REGION" r
JOIN "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."NATION" n ON r.r_regionkey =
   n.n_regionkey;
```

6.6.2 Create a named stage (you can call it whatever you want):

```
CREATE OR REPLACE STAGE mystage;
```

6.6.3 Unload the JOINed data into the stage you created:

```
COPY INTO @mystage FROM
(SELECT * FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."REGION" r JOIN
   "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1"."NATION" n
ON r.r_regionkey = n.n_regionkey);
```

6.6.4 Verify the file is in the stage:

```
LIST @mystage;
```

6.6.5 Remove the file from the stage:

```
REMOVE @mystage;
```


6.6.6 Remove the stage:

```
DROP STAGE mystage;
```

6.6.7 Suspend and resize the warehouse

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

6.7 Key Takeaways

- Files to be loaded can be compressed or not compressed.
- A corresponding table stage is created automatically when a table is created.
- The COPY INTO command can be used to load or unload data.
- Data from multiple tables can be unloaded using a JOIN statement.
- You can use the LIST command to see what files are in a stage.

