Broadcom CA Test Data Manager and Snowflake Data Cloud

Continuous Testing Solution Engineering Team

DRAFT version 0.5

June, 2021

BROADCOM®

Table of Contents

Introduction

The purpose of this document is to provide information about configuring Broadcom Test Data Manager with the Snowflake database - a cloud-based data warehouse/data lake solution. **The steps described below are for a PROOF OF CONCEPT implementation.**

Note:

- ***Snowflake db is not formally supported by Broadcom for all TDM functions.***

TDM Architecture Diagram

The below diagram shows a basic TDM – Snowflake deployment architecture.

TBD

Broadcom CA Test Data Manager and Snowflake Demo Overview

For Test Data purposes, we'll use the supported "SnowSQL" CLI client to assist with TDM operations.

SnowSQL documentation is located here: https://docs.snowflake.net/manuals/user-guide/snowsql.html

There are 2 primary Use Cases where TDM & SnowSQL are suited:

Synthetic Data Generation:

(1) Generate data to a .csv representation
(2) Snowflake bulk load utility - https://docs.snowflake.com/en/user-guide-data-load.html
    a. Snowflake – Stage Data Files from a Local File System
    b. Snowflake – Copying Data from an Internal Stage

Masking:

(1) Snowflake unload utility - https://docs.snowflake.com/en/user-guide/data-unload-overview.html
(2) In-place masking by Fast Data Masker
(3) Snowflake put - https://docs.snowflake.com/en/sql-reference/sql/put.html
(4) Snowflake merge - https://docs.snowflake.com/en/sql-reference/sql/merge.html

Setup

Pre-requisites:

Snowflake ID, Snowflake database available

Install & Configure:

On your Windows TDM Server, download and install the SnowSQL client.

Setup the Default Connection if you wish to simplify this exercise, otherwise perform whatever options are required for your organization & cloud standards for OAuth, OKTA, 2MFA, etc.

On your Windows TDM Server, download and install the 32-bit Snowflake ODBC driver. Configure your connection to the Snowflake DB server **using ODBC Data Sources (32-bit)**.

Summary

Synthetic Data Generation

Initial setup:

In order to initialize the TDM Generator, we'll need to register the data structures.

Datamaker can connect via 32-bit ODBC connection to the Snowflake DB.   It appears you can directly interrogate the data catalog and register tables using Datamaker.

Configuring the generator:

As a Test Data Engineer (TDE) you are responsible for:

- documenting table relationships and ensuring that the generated data is referentially intact as you specify the formulas for each field
- identify any business rules that constrain the values
- identify field formats so the generated results are compatible once uploaded

Publishing the data:

The Publish should be generated to File, File Type=.csv for upload into Snowflake using the SnowSQL client.   See Internal Named Stages for details on how to configure a target when uploading data destined for multiple tables.

Configuring the Upload Using simple batch scripts

As the SnowSQL client can be executed as a Batch Script, you can:

a) Configure the connection information in variables
b) Execute the Batch script / command line as a Post-Publish action

Masking

Initial setup:

In order to initialize Fast Data Masker, we'll need to have a .csv representation of each table that we wish to mask so we can connect with the data structures.

You would execute a series of SELECT TOP 1 FROM queries, with output to .csv for each table.   Once completed,

Configuring FDM generator:

Launch FDM and specify FILE as the masking type.   Specify the directory where the .csv files have been downloaded.   You'll need to create a File Definition for each .csv file.   You can do this one-by-one from the FDM dialog, or manually via a text editor.

After connecting, use FDM to configure the masking rules for each field.

As a Test Data Engineer (TDE) you are responsible to:

- identify any business rules that constrain the masked values
- identify field formats so the generated results are compatible once uploaded

Masking the data:

The masking will generate .csv.scramble files for upload/merge into Snowflake using the SnowSQL client.

Configuring the process Using Javelin.

Once the steps have been vetted, you can utilize Javelin to automate the Unload, Mask, Put, and Merge steps within a flow.

Detail

Data Generation Detailed Steps:

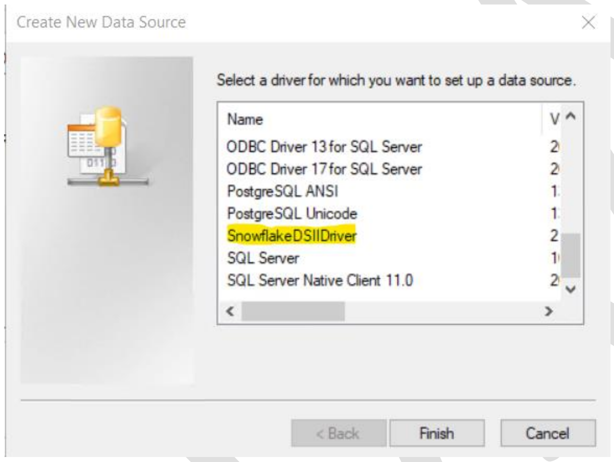
Pre-requisites:

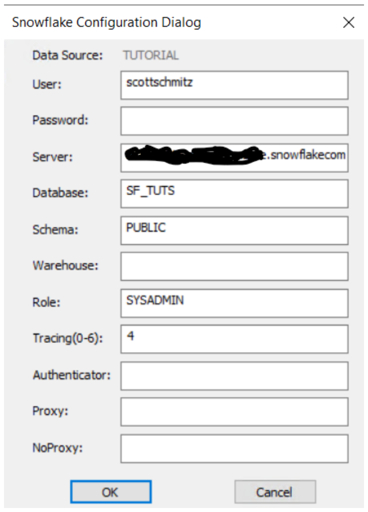
- You've completed the Snowflake in 20 Minutes tutorial up thru Step 5 or use your own data
- IMPORTANT NOTE: If you use the tutorial data, FDM will not tolerate the special characters in the City names – replace the "e" & "o" with a double-dot and "i" with an accent character for the following Cities prior to import in the tutorial: Semënovskoye, Kardítsa, Norrköping
- You've setup a config file for connection to the Snowflake database "example" as seen below
- You've created a TDM Project & Version

Datamaker and ODBC

Prerequisite: Snowflake 32-bit ODBC drivers are installed.

Use the "ODBC Data Sources (32-bit)" windows application to define the connection to the Snowflake DB:

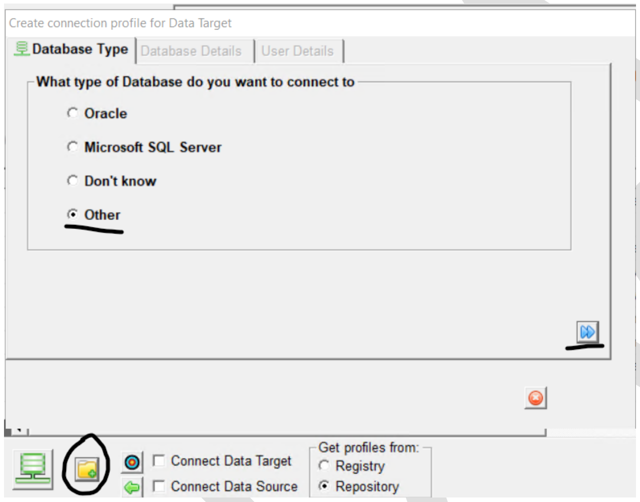

Create New Data Source
Select a driver for which you want to set up a data source.
Name
V
ODBC Driver 13 for SQL Server
ODBC Driver 17 for SQL Server
PostgreSQL ANSI
PostgreSQL Unicode
SnowflakeDSIIDriver
SQL Server
SQL Server Native Client 11.0
< Back
Finish
Cancel


Snowflake Configuration Dialog

Data Source: TUTORIAL
User: scottschmitz
Password:
Server: [redacted].snowflakecom
Database: SF_TUTS
Schema: PUBLIC
Warehouse:
Role: SYSADMIN
Tracing(0-6): 4
Authenticator:
Proxy:
NoProxy:

Launch Datamaker.

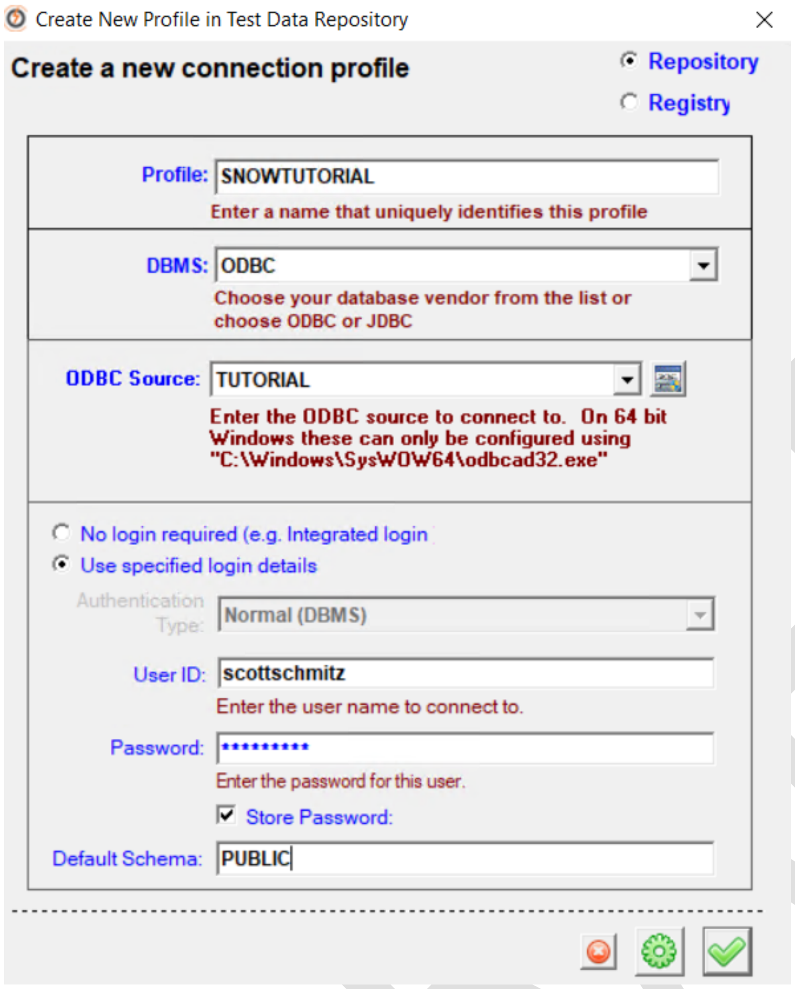
Create a new Database Connection:


Create connection profile for Data Target

What type of Database do you want to connect to

Oracle [ ] Microsoft SQL Server [ ] Don't know [ ] [✓ Other]

Get profiles from: Registry [✓ Repository]



Create New Profile in Test Data Repository

Create a new connection profile

(● Repository) ○ Registry

Profile: SNOWTUTORIAL

Enter a name that uniquely identifies this profile

DBMS: ODBC

Choose your database vendor from the list or choose ODBC or JDBC

ODBC Source: TUTORIAL

Enter the ODBC source to connect to. On 64 bit Windows these can only be configured using "C:\Windows\SysWOW64\odbcad32.exe"

○ No login required (e.g. Integrated login

● Use specified login details

Authentication Type: Normal (DBMS)

User ID: scottschmitz

Enter the user name to connect to.

Password: *********

Enter the password for this user.

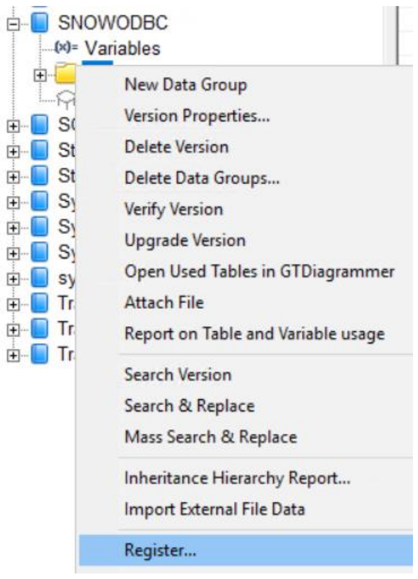
✓ Store Password:

Default Schema: PUBLIC

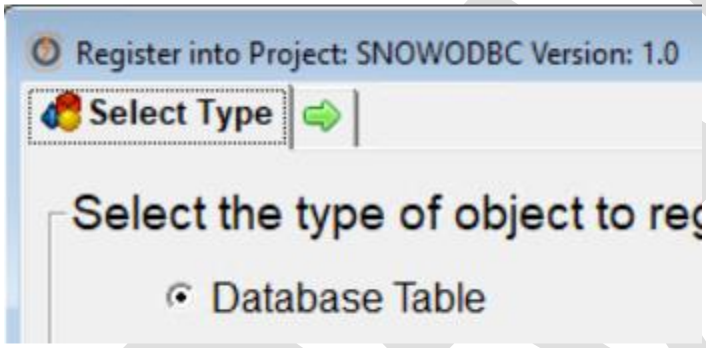
Test the connection, then save.

In Datamaker, set the Project and Version to the values created above.

Expand the Project folder structure on the left until you see the top-level folder.   Right mouse on the folder and select Register

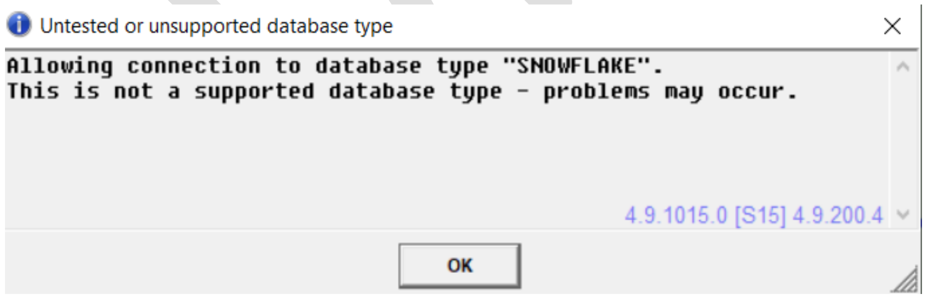


Select Database Table and click the green arrow


Register into Project: SNOWODBC Version: 1.0
Select Type
Select the type of object to reg
Database Table

When Prompted, set the Database Connection to the Snowflake connection you configured above.

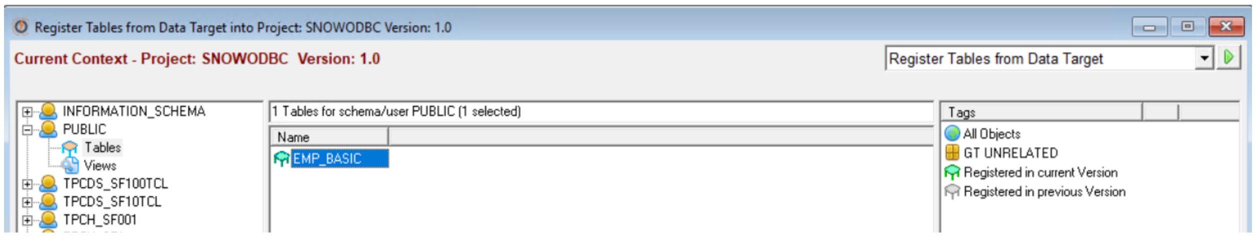
You'll get this warning, click ok.


Untested or unsupported database type

```
Allowing connection to database type "SNOWFLAKE".
This is not a supported database type - problems may occur.
```

4.9.1015.0 [S15] 4.9.200.4

Select the Table(s) and Register.


Register Tables from Data Target into Project: SNOWODBC Version: 1.0
Current Context - Project: SNOWODBC Version: 1.0
Register Tables from Data Target
INFORMATION_SCHEMA
PUBLIC
Tables
Views
TPCDS_SF100TCL
TPCDS_SF10TCL
TPCH_SF001
1 Tables for schema/user PUBLIC (1 selected)
Name
EMP_BASIC
Tags
All Objects
GT UNRELATED
Registered in current Version
Registered in previous Version

Now that the object has been registered, in the TDM Portal, navigate to the Project/Version, select the Generators tab on the left, then click the Create Generator button

Name the Generator “Generate New Employees”, open it, and open the emp_basic table.

Click the +r on the right side to add a row to the table.   Enter formulas into the fields.   Some samples:

FIRST_NAME	@randlov(0,@seedlist(FirstName)@)@
LAST_NAME	@randlov(0,@seedlist(LastName)@)@
EMAIL	^FIRST_NAME^.@collapse(^LAST_NAME^)@@atsign(1)@snowflakedemo.com

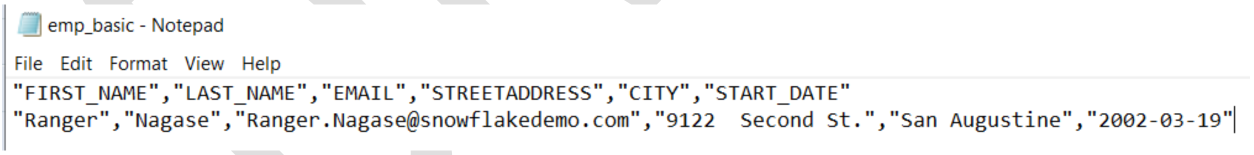
STREETADDRESS

@randrange(1,9999)@ @percval(10%N.,5%North,10%E.,5%East,10%S.,5%South,10%W.,5%West,40%)@ @percval(10%Second St.,10%Main St.,10%Park Ave.,10%Oak St.,10%Pine St.,10%Maple Ln.,10%Washington St.,10%Lake Dr.,10%Hill Ave.,10%Ninth St.)@

CITY	@randlov(0,@seedlist(US Zip-Codes)@,3)@

Publish one record to type File, format CSV.

Once the publish is complete, download the zip file and extract the emp_basic.csv file.   For example:

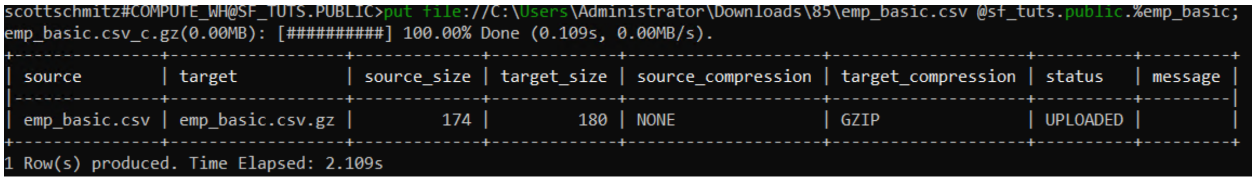


```
"FIRST_NAME","LAST_NAME","EMAIL","STREETADDRESS","CITY","START_DATE"
"Ranger","Nagase","Ranger.Nagase@snowflakedemo.com","9122  Second St.","San Augustine","2002-03-19"
```

Launch the SNOWSQL client, and use the PUT command to upload the contents to the Snowflake staging table (substitute your UserID & jobID as highlighted):

```
put file://C:/Users/Administrator/Downloads/85/emp_basic.csv @sf_tuts.public.%emp_basic;
```

NOTE: In the example above, we are using a Table Stage “%emp_basic”.   This is only valid when working with a single table.   Otherwise, we need to use an Internal Named Stage.

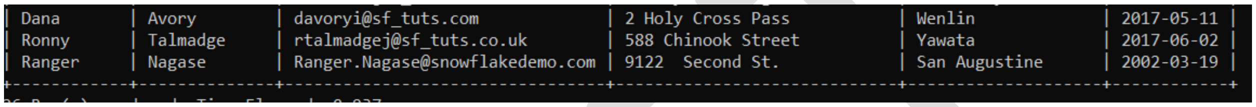


```
scottschmitz#COMPUTE_WH@SF_TUTS.PUBLIC>put file://C:\Users\Administrator\Downloads\85\emp_basic.csv @sf_tuts.public.%emp_basic;
emp_basic.csv_c.gz(0.00MB): [##########] 100.00% Done (0.109s, 0.00MB/s).
+---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------+
| source        | target           | source_size | target_size | source_compression | target_compression | status   | message |
|---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------|
| emp_basic.csv | emp_basic.csv.gz |         174 |         180 | NONE               | GZIP               | UPLOADED |         |
+---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------+
1 Row(s) produced. Time Elapsed: 2.109s
```

Then use the COPY INTO command to migrate the data from Staging to the Snowflake table, telling it to ignore the 1st line as it includes the headers.

copy into emp_basic from @%emp_basic file_format = (type = csv field_optionally_enclosed_by='"' skip_header = 1);

The result is that the newly generated Synthetic Data has been inserted into the table.



```
| Dana        | Avory         | davoryi@sf_tuts.com              | 2 Holy Cross Pass            | Wenlin              | 2017-05-11 |
| Ronny       | Talmadge      | rtalmadgej@sf_tuts.co.uk         | 588 Chinook Street           | Yawata              | 2017-06-02 |
| Ranger      | Nagase        | Ranger.Nagase@snowflakedemo.com  | 9122  Second St.             | San Augustine       | 2002-03-19 |
+-----------+--------------+--------------------------------+-----------------------------+--------------------+------------+
```

So you've mastered the basics of TDM Data Generation and insertion into Snowflake.   The next step is to automate the insertion via a "Post-Publish Action" in TDM.

Creating the .bat file for the Post-Publish Action:

Ensure that you have setup Snowflake variables for the output path of the generated files and the PUBJOBID variable that will be pulled from TDM at generation time.

Snowflake config file example:

```
[variables]

PUBJOBOUTPATH="C:/ProgramData/CA/CA Test Data Manager Portal/Jobs/Job_"

PUBJOBID=1
```

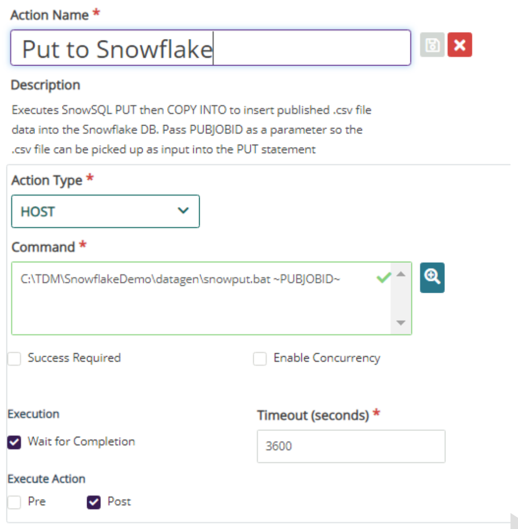
Create a new “snowput.bat” file containing the following line:

```
snowsql -c example -d SF_TUTS -s public –D PUBJOBID=%1 -f C:\TDM\SnowflakeDemo\datagen\put-copy-into-emp-basic.sql
```

Create the “put-copy-into-emp-basic.sql” file with the following contents:

```
USE DATABASE SF_TUTS

;

USE SCHEMA PUBLIC

;

put 'file://&{PUBJOBOUTPATH}&{PUBJOBID}/emp_basic.csv' @sf_tuts.public.%emp_basic overwrite=true

;

copy into SF_TUTS.PUBLIC.EMP_BASIC from @sf_tuts.public.%emp_basic file_format = (type = csv
field_optionally_enclosed_by='"' skip_header = 1)

;
```

Return to the TDM Generator defined before and click on the ACTIONS button. Define a new ACTION. Specify the full path to the .bat file you created above, followed by a space, followed by the TDM Variable ~PUBJOBID~ which will be passed as the first and only parameter (and referenced as %1 on the above snowsql command).


Action Name *

Put to Snowflake

Description

Executes SnowSQL PUT then COPY INTO to insert published .csv file data into the Snowflake DB. Pass PUBJOBID as a parameter so the .csv file can be picked up as input into the PUT statement

Action Type *

HOST

Command *

```
C:\TDM\SnowflakeDemo\datagen\snowput.bat ~PUBJOBID~
```

☐ Success Required ☐ Enable Concurrency

Execution

☑ Wait for Completion

Timeout (seconds) *

3600

Execute Action

☐ Pre ☑ Post

If the HOST actions are not configured for this TDM Portal, update the portal's application.properties file:

```
tdmweb.enableHostActions=true
```

and restart the TDM Portal so it can pickup the configuration change.

IMPORTANT NOTE: Because SnowSQL can only pickup the Password from the ~/.snowsql/config file for basic authentication, your Portal MUST be configured to run under a User Account with this file in the %USERPROFILE% path and <u>not</u> "Local System Account" as the .bat file will be executed under that user.

Execute a Publish and confirm that the new job's data is inserted into the Snowflake table via the Post-Publish Action automation.

Masking Detailed Steps:

Pre-requisites:

- Connection setup for SnowSQL
- Table/Columns to mask are known
- Mask types identified per column

Create a .bat file to export the table contents to a .csv file:

export-from-snowflake.bat

*snowsql -c example -d sf_tuts -s public -q "select * from emp_basic" -o output_format=csv -o header=true -o timing=false -o friendly=false > C:\TDM\SnowflakeDemo\masking\emp_basic.csv*

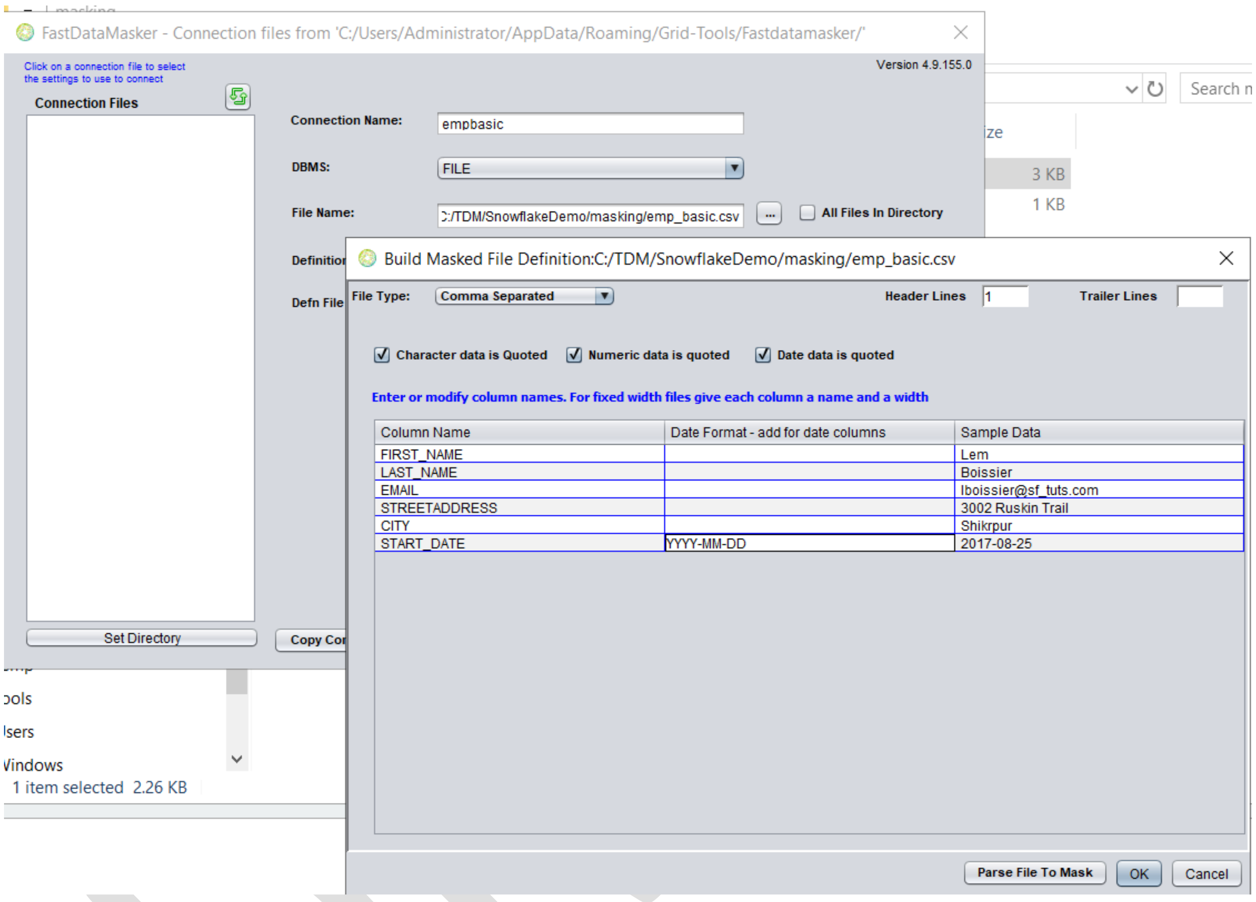
NOTE: In this simple example, we are exporting all columns in the table. In general, you should not do this! You will only need to download:

(1) The column that can unique identify this record (EMAIL in our case)
(2) The column(s) to be masked (START_DATE in our case)

Limiting the amount of data crossing between the FDM/TDM server and the Snowflake interface will minimize the data transfer time.

Execute the .bat file to create the .csv to allow FDM to interrogate the data structure.

Launch Fast Data Masker, change to FILE mask type, and use the “Create Definition File” button to Parse the File to Mask as shown below. Add the Date Format (and tab out of that field) to tell FDM how to interpret the date.


FastDataMasker - Connection files from 'C:/Users/Administrator/AppData/Roaming/Grid-Tools/Fastdatamasker/'

Version 4.9.155.0

Connection Name: empbasic

DBMS: FILE

File Name: :/TDM/SnowflakeDemo/masking/emp_basic.csv

Build Masked File Definition:C:/TDM/SnowflakeDemo/masking/emp_basic.csv

File Type: Comma Separated

Header Lines: 1

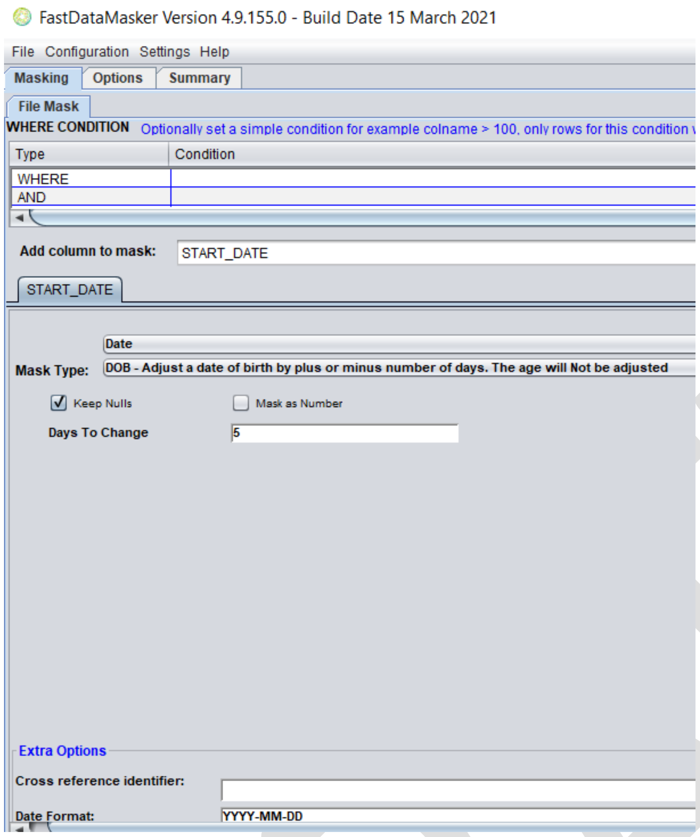
Trailer Lines:

[✓] Character data is Quoted [✓] Numeric data is quoted [✓] Date data is quoted

Enter or modify column names. For fixed width files give each column a name and a width

Column Name	Date Format - add for date columns	Sample Data
FIRST_NAME		Lem
LAST_NAME		Boissier
EMAIL		lboissier@sf_tuts.com
STREETADDRESS		3002 Ruskin Trail
CITY		Shikrpur
START_DATE	YYYY-MM-DD	2017-08-25

Choose the field(s) in Fast Data Masker to mask. START_DATE is used in this example.



FastDataMasker Version 4.9.155.0 - Build Date 15 March 2021

WHERE CONDITION Optionally set a simple condition for example colname > 100, only rows for this condition

Type	Condition
WHERE	
AND	

Add column to mask: START_DATE

START_DATE

Date

Mask Type: DOB - Adjust a date of birth by plus or minus number of days. The age will Not be adjusted

[✓] Keep Nulls   [ ] Mask as Number

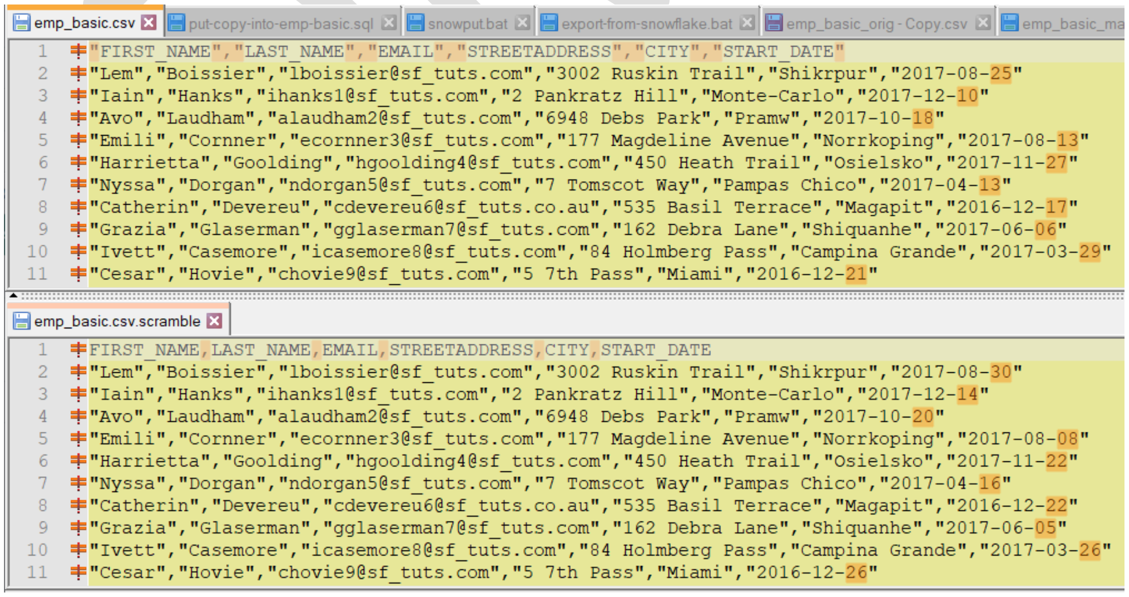
Days To Change: 5

Cross reference identifier:

Date Format: YYYY-MM-DD

Add any options such as AUDIT file and Save & Run the Masking job.

A .scramble file will be produced.   You can inspect this and the audit file to verify the dates have changed plus or minus 5 days as specified.


emp_basic.csv
put-copy-into-emp-basic.sql
snowput.bat
export-from-snowflake.bat
emp_basic_orig - Copy.csv
emp_basic_ma
emp_basic.csv.scramble

```
"FIRST_NAME","LAST_NAME","EMAIL","STREETADDRESS","CITY","START_DATE"
"Lem","Boissier","lboissier@sf_tuts.com","3002 Ruskin Trail","Shikrpur","2017-08-25"
"Iain","Hanks","ihanks1@sf_tuts.com","2 Pankratz Hill","Monte-Carlo","2017-12-10"
"Avo","Laudham","alaudham2@sf_tuts.com","6948 Debs Park","Pramw","2017-10-18"
"Emili","Cornner","ecornner3@sf_tuts.com","177 Magdeline Avenue","Norrkoping","2017-08-13"
"Harrietta","Goolding","hgoolding4@sf_tuts.com","450 Heath Trail","Osielsko","2017-11-27"
"Nyssa","Dorgan","ndorgan5@sf_tuts.com","7 Tomscot Way","Pampas Chico","2017-04-13"
"Catherin","Devereu","cdevereu6@sf_tuts.co.au","535 Basil Terrace","Magapit","2016-12-17"
"Grazia","Glaserman","gglaserman7@sf_tuts.com","162 Debra Lane","Shiquanhe","2017-06-06"
"Ivett","Casemore","icasemore8@sf_tuts.com","84 Holmberg Pass","Campina Grande","2017-03-29"
"Cesar","Hovie","chovie9@sf_tuts.com","5 7th Pass","Miami","2016-12-21"
```

```
FIRST_NAME,LAST_NAME,EMAIL,STREETADDRESS,CITY,START_DATE
"Lem","Boissier","lboissier@sf_tuts.com","3002 Ruskin Trail","Shikrpur","2017-08-30"
"Iain","Hanks","ihanks1@sf_tuts.com","2 Pankratz Hill","Monte-Carlo","2017-12-14"
"Avo","Laudham","alaudham2@sf_tuts.com","6948 Debs Park","Pramw","2017-10-20"
"Emili","Cornner","ecornner3@sf_tuts.com","177 Magdeline Avenue","Norrkoping","2017-08-08"
"Harrietta","Goolding","hgoolding4@sf_tuts.com","450 Heath Trail","Osielsko","2017-11-22"
"Nyssa","Dorgan","ndorgan5@sf_tuts.com","7 Tomscot Way","Pampas Chico","2017-04-16"
"Catherin","Devereu","cdevereu6@sf_tuts.co.au","535 Basil Terrace","Magapit","2016-12-22"
"Grazia","Glaserman","gglaserman7@sf_tuts.com","162 Debra Lane","Shiquanhe","2017-06-05"
"Ivett","Casemore","icasemore8@sf_tuts.com","84 Holmberg Pass","Campina Grande","2017-03-26"
"Cesar","Hovie","chovie9@sf_tuts.com","5 7th Pass","Miami","2016-12-26"
```

Now we need to build the upload and merge scripts.

Using snowsql, create an internal stage to receive the PUT (**mandatory** to support the merge):

create stage EMPTMP FILE_FORMAT=(TYPE=CSV,field_optionally_enclosed_by='"');

Create a .bat file for the snowsql execution, containing the following line:

snowsql -c example -d SF_TUTS -s public -f C:\TDM\SnowflakeDemo\masking\put-merge-emp-basic.sql

Create the .sql file containing the PUT and merge commands

put-merge-emp-basic.sql

```
USE DATABASE SF_TUTS

;

USE SCHEMA PUBLIC

;

put 'file://C:/TDM/SnowflakeDemo/masking/emp_basic.csv.scramble' @EMPTMP overwrite=true;

;

MERGE INTO SF_TUTS.PUBLIC.EMP_BASIC

USING

  (

    SELECT

      $3 EMAIL

      , $6 START_DATE

    FROM @SF_TUTS.PUBLIC.EMPTMP/emp_basic.csv.scramble.gz

  ) EMP_TMP_DESC

ON

  EMP_BASIC.EMAIL=EMP_TMP_DESC.EMAIL

WHEN MATCHED THEN

  UPDATE SET

      EMP_BASIC.START_DATE = EMP_TMP_DESC.START_DATE

;
```

NOTE: The $3 and $6 notations mean that these are the 3rd and 6th columns in the file. If you only exported the “id” (EMAIL) and the column to mask (“START_DATE”), they will be $1 and $2.

Execute the scripts.

Review the masked results using Datamaker, your SNOWSQL client or Snowflake Web Client.

Automating the masking process:

Use Javelin or a top-level .bat file to create a “round-trip” export-mask-import process. ***You’ll need to ensure that the mask step completes (and you’ll need to wait some seconds after that for FDM to flush the file to disk)*** before invoking the last script to perform the import (put-merge).