Broadcom CA Test Data Manager

Setting up a Simple Hadoop/Hive Data Generation Demo

Continuous Testing Solution Engineering Team

DRAFT version 1.0

January, 2021

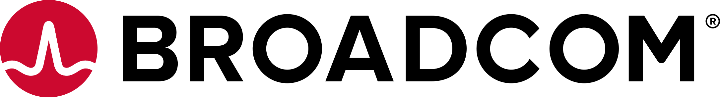


Table of Contents

[Introduction 3](#_Toc62560197)

[Pre-Requisites 3](#_Toc62560198)

[TDM / Hadoop Setup 4](#_Toc62560199)

[Sqoop 15](#_Toc62560200)

# 

# Introduction

The purpose of this document is to provide information to allow Solution Engineers, Territory Managers, or customers to setup a prototype Test Data Manager synthetic data generation environment for Hadoop.

**Note**: This document is an example, and the steps listed here are not supported by Broadcom. This example is for SQL Server.

# Pre-Requisites

1. Test Data Manager installed
2. Accessibility to a Microsoft SQL Server data server
3. Hadoop / Hive server available

# TDM / Hadoop Setup

This document serves as a companion to the “Hadoop Demo Environment Setup – 2020” document that describes how to install/configure and execute a simple Hadoop masking use case.

For Synthetic Data Generation, we’ll be using “sqoop” to migrate generated data from a Relational Database to HDFS. In this setup, we’ll use a SQL Server “staging” schema for the interim data storage.

***The first task for the Test Data Engineer (TDE) is to understand the formats of the Hive/HDFS data that will need to be generated. They should collect information about the HDFS data model, including Tables, Column Names/Data Types, and any relationships between columns across tables.***

For this simple demo setup, we’ll start with two tables with the following Hadoop/Hive definitions:

create table customer (custid int, custgivenname string, custfamilyname string, custaddress1 string, custaddress2 string, custcity string, custstate string, custzip string, custphone string, custstorenum int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

create table store (storenum int, storename string, storeaddress1 string, storeaddress2 string, storecity string, storestate string, storezip string, storephone string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

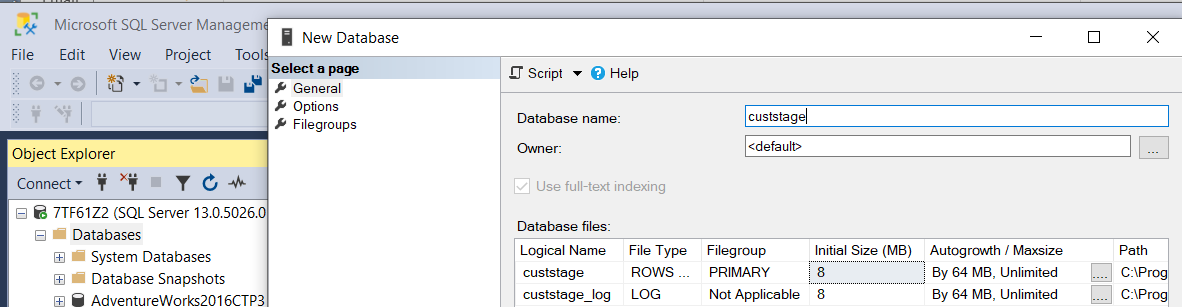
The custstorenum in the customer table is related to the storenum in the store table.

Store.csv

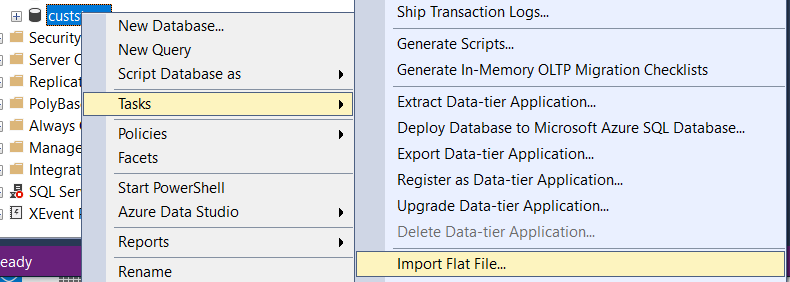
Storenum,storename,storeaddress1,storeaddress2,storecity,storestate,storezip,storephone

1,Downtown,13 N. Main St,City Square,Atlanta,GA,30303,303-456-7890

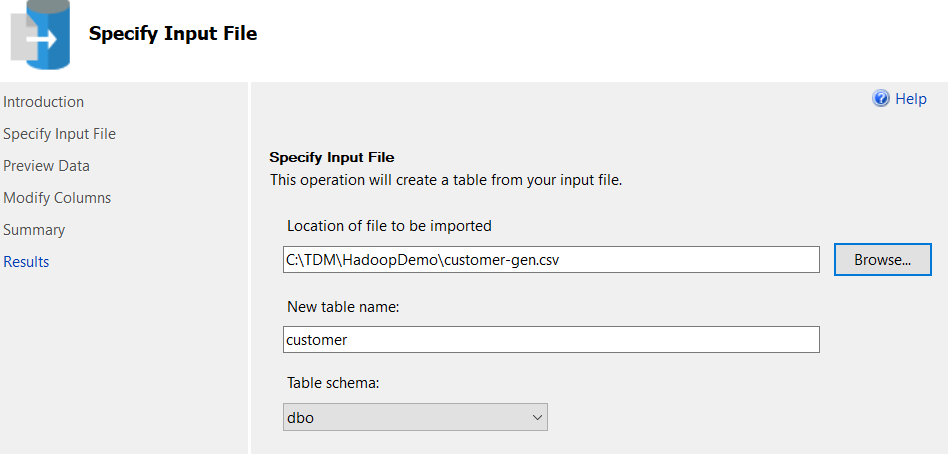
**Create the staging database in SQL Server**

****

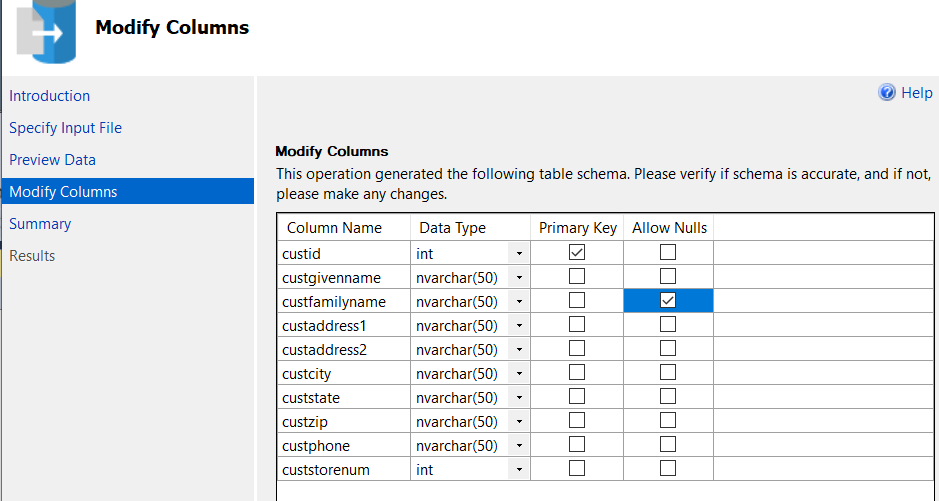
**Import the data as a flat file**

****

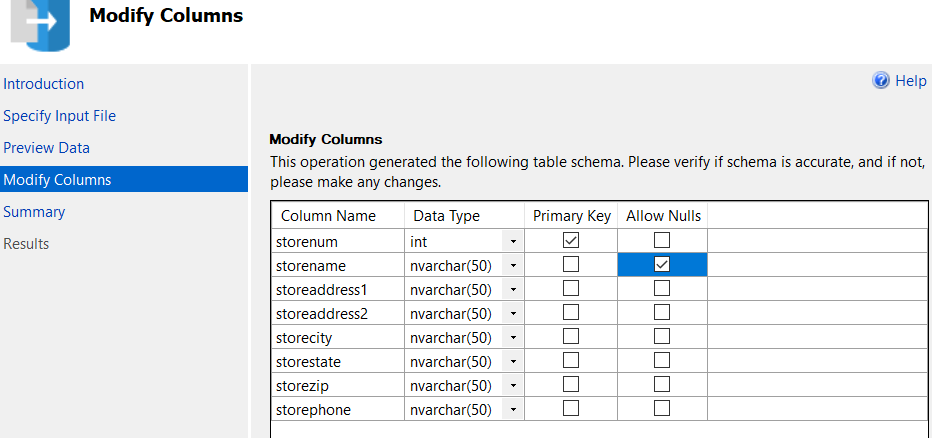
**We’ll start with the customer data**

****

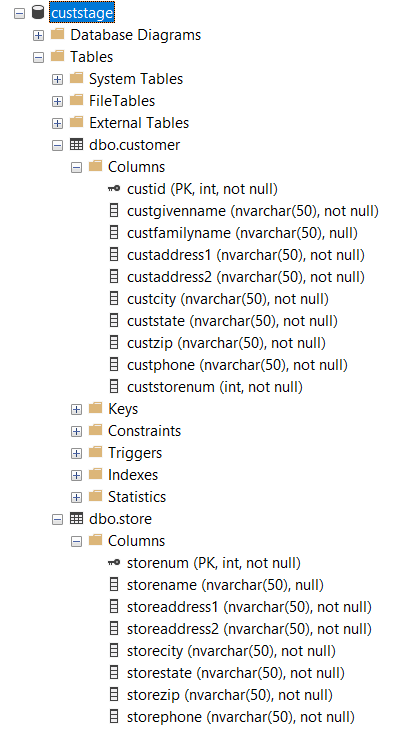
**As the .csv file has no real format/context, ensure you update the field definitions with data types and business rules for length before accepting the table schema. If one column is unique, set a Primary Key.**

****

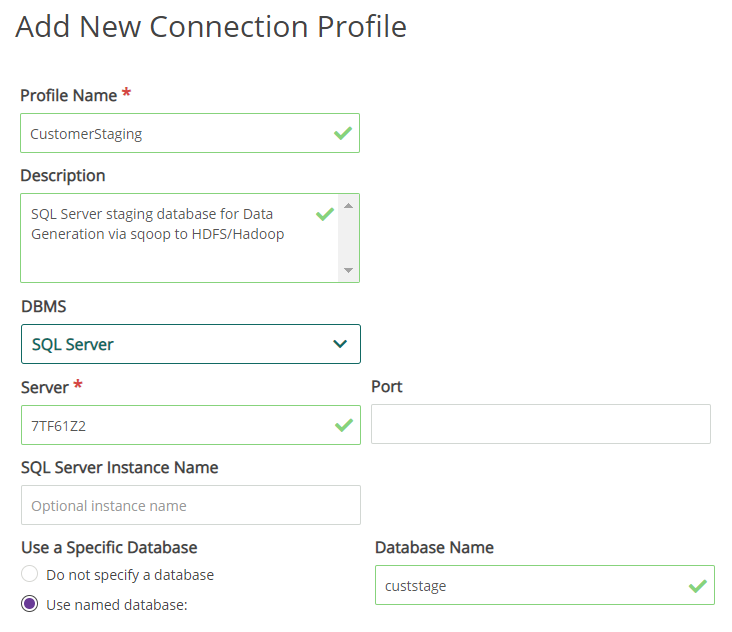
**After the wizard finishes, repeat for store**

****

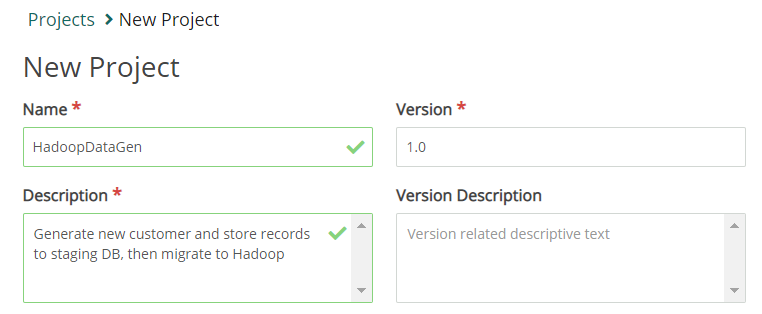
**Refresh MS Management Studio to see the new tables**

****

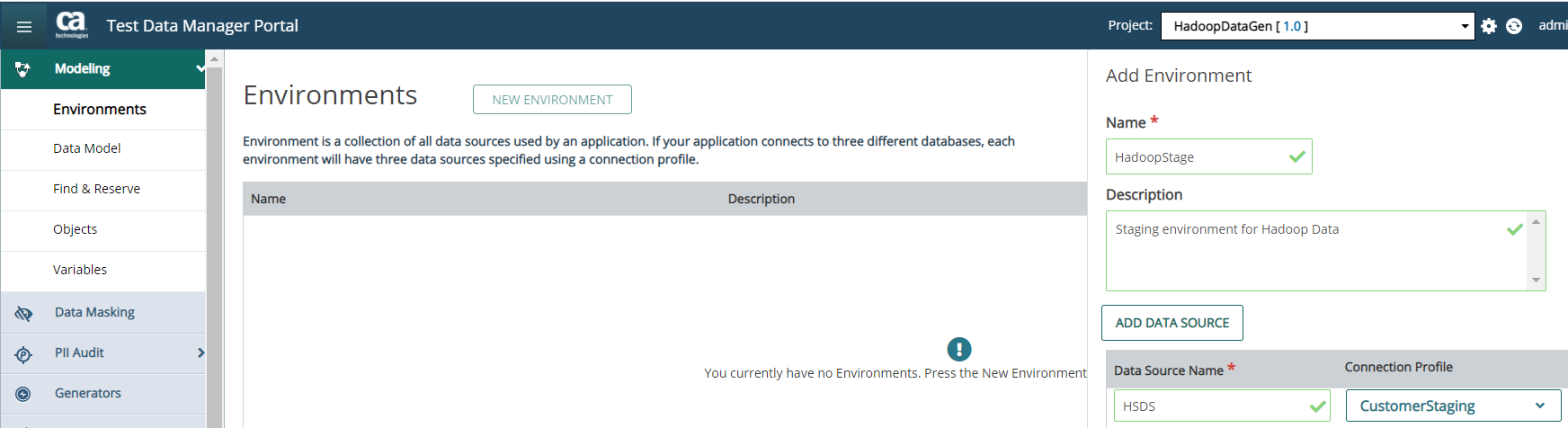
**Create a New Connection Profile in TDM Portal, pointing to the staging DB.**



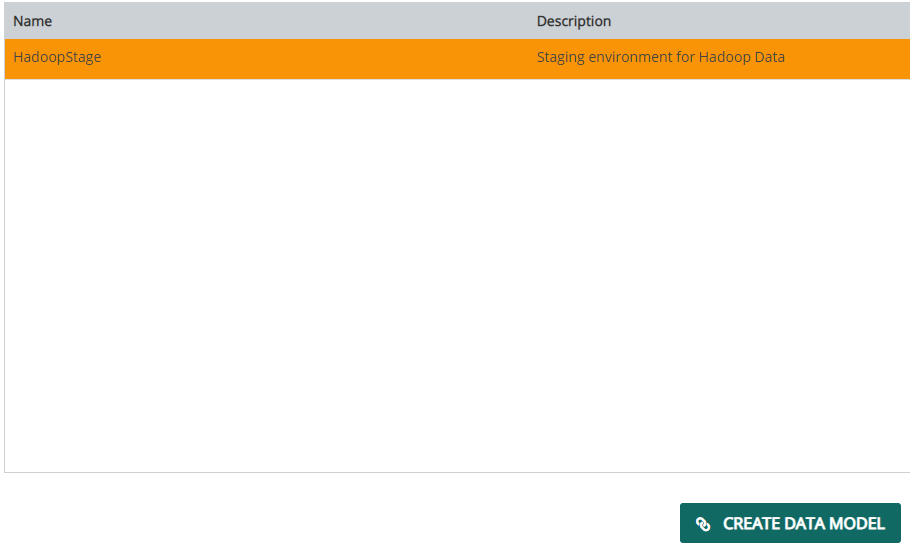
**Create a New Project in Portal.**

****

With the project selected in the top bar, choose Modeling->Environments and create an environment pointing to the newly created connection profile.

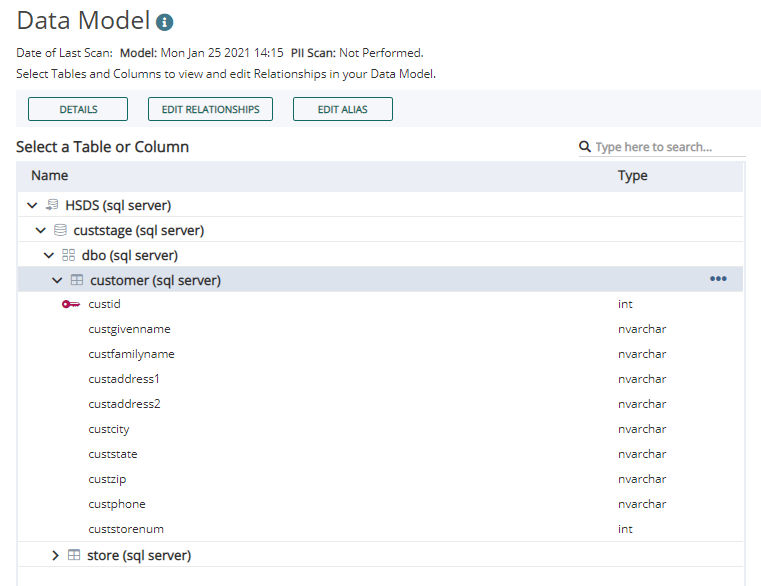


After saving, click the row that shows the Environment and click the Create Data Model button.

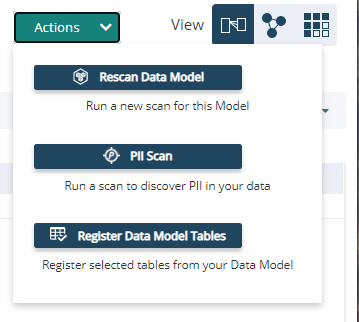


Click the SCAN button on the next page, keeping all the default values.

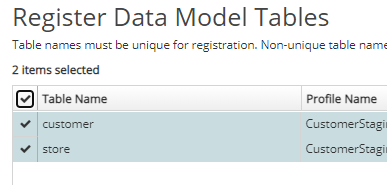
The resulting Data Model should reflect the SQL Server structures you create above.



Click the Actions button, and select Register Data Model Tables.

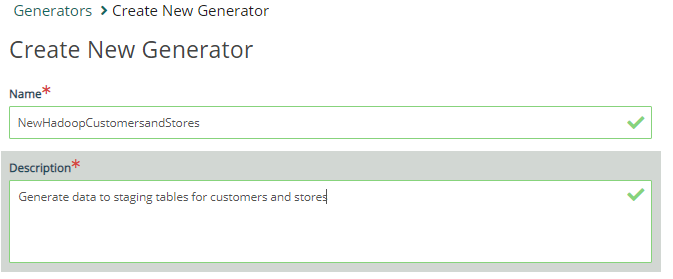


Click the checkbox in the header row to select all, then click the Register button.



The status will update to “Registered” once complete.

**Create a Generator**

****

Fill in the values as needed. Below are suggestions, replace the seedlists for your requirements.

Table Name Column Name Definition

customer custid ~NEXT~

customer custgivenname @randlov(0,@seedlist(FirstName)@)@

customer custfamilyname @randlov(0,@seedlist(LastName)@)@

customer custaddress1 @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.)@

customer custaddress2 @percval(3%Ste. @randrange(1,999)@,10%Apt. @randrange(1,9999)@ ,87% )@

customer custcity @randlov(0,@seedlist(US City State Zip County)@,1)@

customer custstate @randlov(0,@seedlist(US City State Zip County)@,2)@

customer custzip @randlov(0,@seedlist(US City State Zip County)@,3)@

customer custphone @randrange(100,999)@-@randrange(100,999)@-@leftpad(@randrange(100,999)@,0,4)@

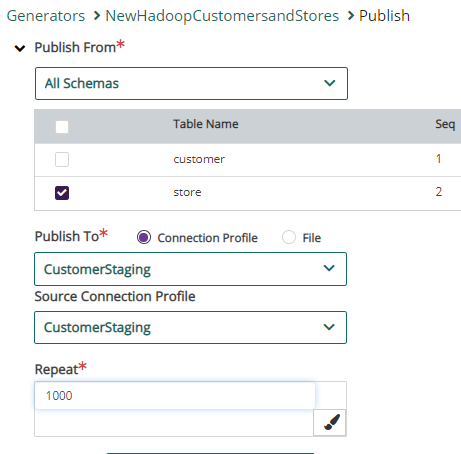
customer custstorenum @randlov(0,@sqllist(PCustomerStaging,Select storenum from Store)@)@

store storenum ~NEXT~

store storename @percval(50% ,50%@percval(20%West,20%East,20%North,20%South,20%Central)@ @randlov(0,@seedlist(US City State Zip County)@,1)@)@

(And similar values as the customer table for the remaining columns).

Click Publish, and set for a higher value (1000 in this example), to be published to the SQL Server Staging Tables. Deselect customer as we’ll only be populating the Stores the first time thru to allow the customers to reference existing store numbers in the custstorenum field.

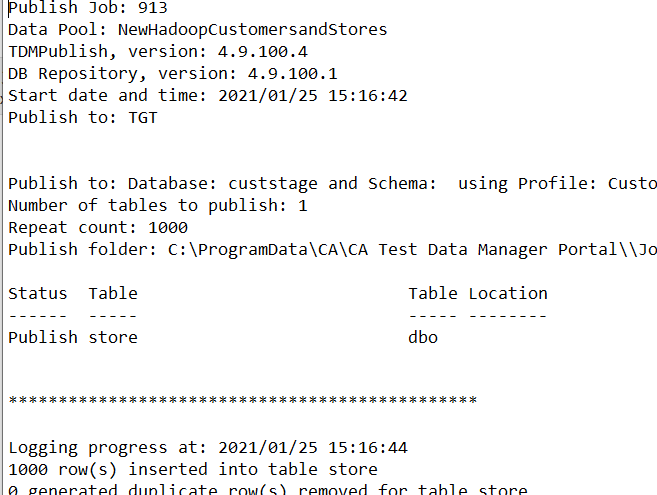


**(If you have data in those SQL Server tables, truncate them before the publish)**

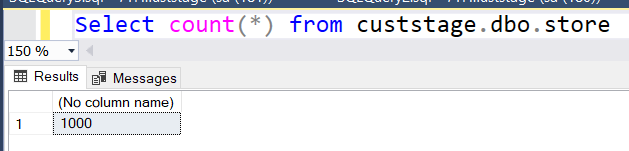
Review the results of the Publish in the Submitted Requests (Generator Pulldown), and download the logs from the link on the right.

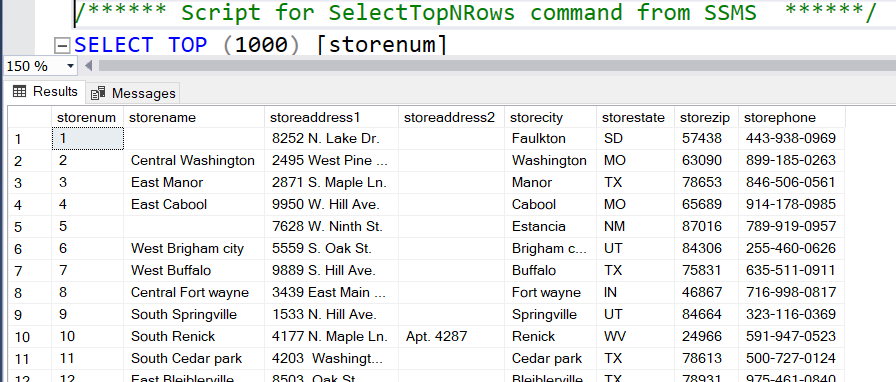


You’ll see confirmation of the inserts

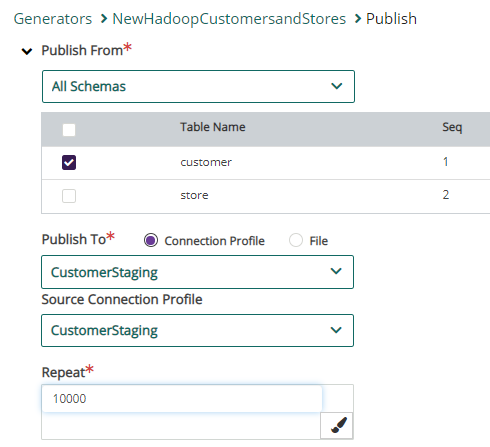


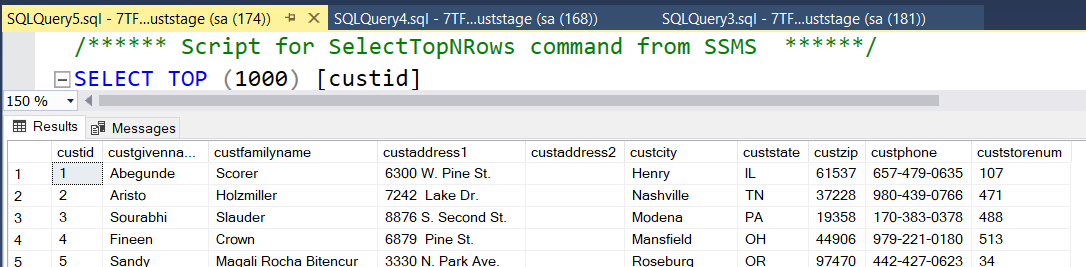
And you can use SQL Server Management Studio to confirm





Now we are ready to generate the customers. Return to the Generator, Publish, and select only the Customers to be generated.





Now we’ll setup Sqoop to migrate the data to Hadoop

# Sqoop

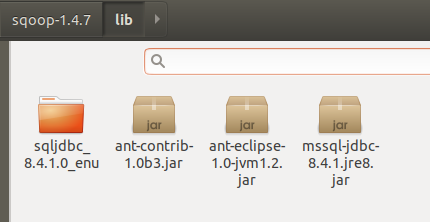
Download and extract [Sqoop](https://sqoop.apache.org/) per the instructions: (get the .gz file with hadoop in the name).

User Guide:

<https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html>

Setup:

Download the [SQL Server Drivers](https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15), and move the .jar file for the current version of java into the sqoop/lib directory, and the $HADOOP\_HOME/lib directory



**Check basic connectivity:**

./sqoop-1.4.7.bin\_\_hadoop-2.6.0/bin/sqoop list-databases --connect "jdbc:sqlserv

er://192.168.68.118;databaseName=custstage" --username sqoop --password sqoopTDM

123

You should see a list of the databases on the SQL Server **(prereq: you’ve setup a SQL user with access to the SQL Server staging DB)**

We’ll use the import-all-tables option to get both customer and stores. If the tables already exist, use the overwrite switch.

**1st try results in error:**

*Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/commons/lang/StringEscapeUtils*

*at org.apache.sqoop.orm.ClassWriter.serializeRawColName(ClassWriter.java:1126)*

[This response](https://stackoverflow.com/questions/55323057/sqoop-import-from-mysql-failed-with-exception-in-thread-main-java-lang-noclass) says to [download](https://commons.apache.org/proper/commons-lang/download_lang.cgi) commons-lang-2.6.jar & put in the sqoop/lib folder.

**2nd try does import into HDFS, but when it tries to create the Hive table, get**

HIVE\_CONF\_DIR error.

[This response](https://stackoverflow.com/questions/51661049/error-hive-hiveconfig-could-not-load-org-apache-hadoop-hive-conf-hiveconf-make) says to download hive-common-0.10.0.jar & put in the sqoop/lib folder.

**3rd try says /user/hdoop/customer already exists**

hadoop fs -rm -R /user/hdoop/customer to remove HDFS table

**4th try – success!**

**./sqoop-1.4.7.bin\_\_hadoop-2.6.0/bin/sqoop import-all-tables --connect "jdbc:sqls**

**erver://192.168.68.118;databaseName=custstage" --username sqoop --password sqoop**

**TDM123 --hive-import --hive-overwrite**

10,000 customer records + 1,000 store records took 27 minutes to migrate using an underpowered VM.

**Advanced Topic:**

Automate the 2 publishes and the remote sqoop execution via Javelin using the TDM Rest API and ssh capabilities.