**SECURING DBMS DATA**

**WITH**

**DATAGUISE DgSecure**

****



**Walk-Through Guide**

Table of Contents

[Introduction: Using DgSecure in the Hortonworks Sandbox 3](#_Toc445797795)

[Getting Started 3](#_Toc445797796)

[Logging into DgSecure interface 3](#_Toc445797797)

[Create Policy 4](#_Toc445797798)

[Discover Sensitive Data within SQL Server Database 7](#_Toc445797799)

[Connection Manager 7](#_Toc445797800)

[Create a Discovery Task 9](#_Toc445797801)

[Connect Directly to Database 15](#_Toc445797802)

[Masking the SQL Database Person Table 17](#_Toc445797803)

[Create 1st Masking Task 18](#_Toc445797804)

[Masking the SQL Database Address Table 24](#_Toc445797805)

[Create 2nd Masking Task 25](#_Toc445797806)

[Summary 33](#_Toc445797807)

## Introduction: Using DgSecure in the Hortonworks Sandbox

This document will guide you through several use cases around Discovering and protecting sensitive data within a SQL Server database. DBAs nowadays are no longer involved in the cradle-to-grave life cycle of databases. They are now separate departments that maintain hundreds of databases and are no longer familiar with every database, every table and every column. The need to now protect sensitive data within those databases for test/dev users becomes harder and harder without DBAs being tightly involved day after day. Dataguise’s DgSecure for DBMS will show you how to find all of your sensitive data within these databases and then (with that Discovery knowledge), now confidently mask these copies this protecting your organization from insider breaches.

This Guide will step you through the DgSecure interface and how to successfully protect your company’s sensitive data and now confidently utilize the cloud for your testing and developing of applications. This guide will utilize the SQL Server AdventureWorks database which is hidden behind the scenes during this evaluation.

## Getting Started

You will receive an email with the actual url for Dataguise.

It will look something like:

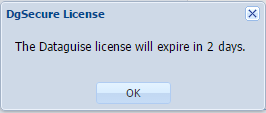
[*https://dnsux3bbrn5lkubuntu.westus.cloudapp.azure.com:10182/DgSecure/index.html#dgshome*](https://dnsux3bbrn5lkubuntu.westus.cloudapp.azure.com:10182/DgSecure/index.html#dgshome)

Enter that url into I.E. or Chrome and this screen sill pop up.

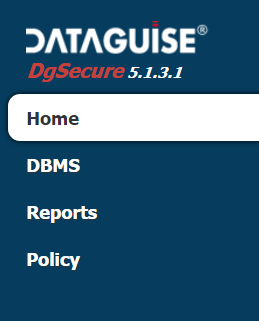


## Logging into DgSecure interface

* Enter username: **dataguise**
* Enter password: **dataguise**
* You will get a message about the limited license:



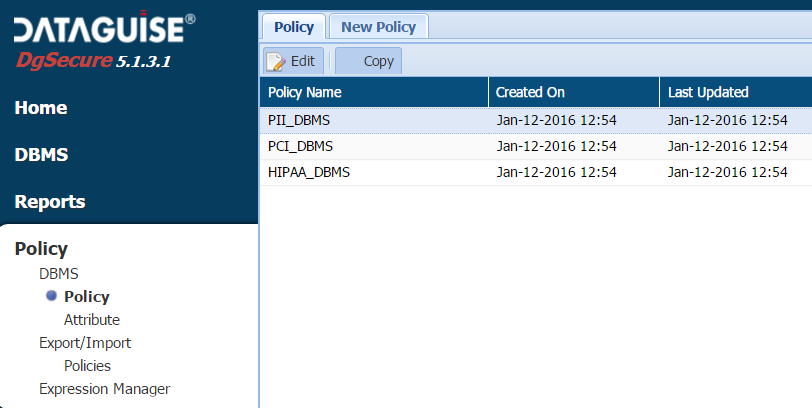
* On the left side of the DgSecure interface, you will see these sections:



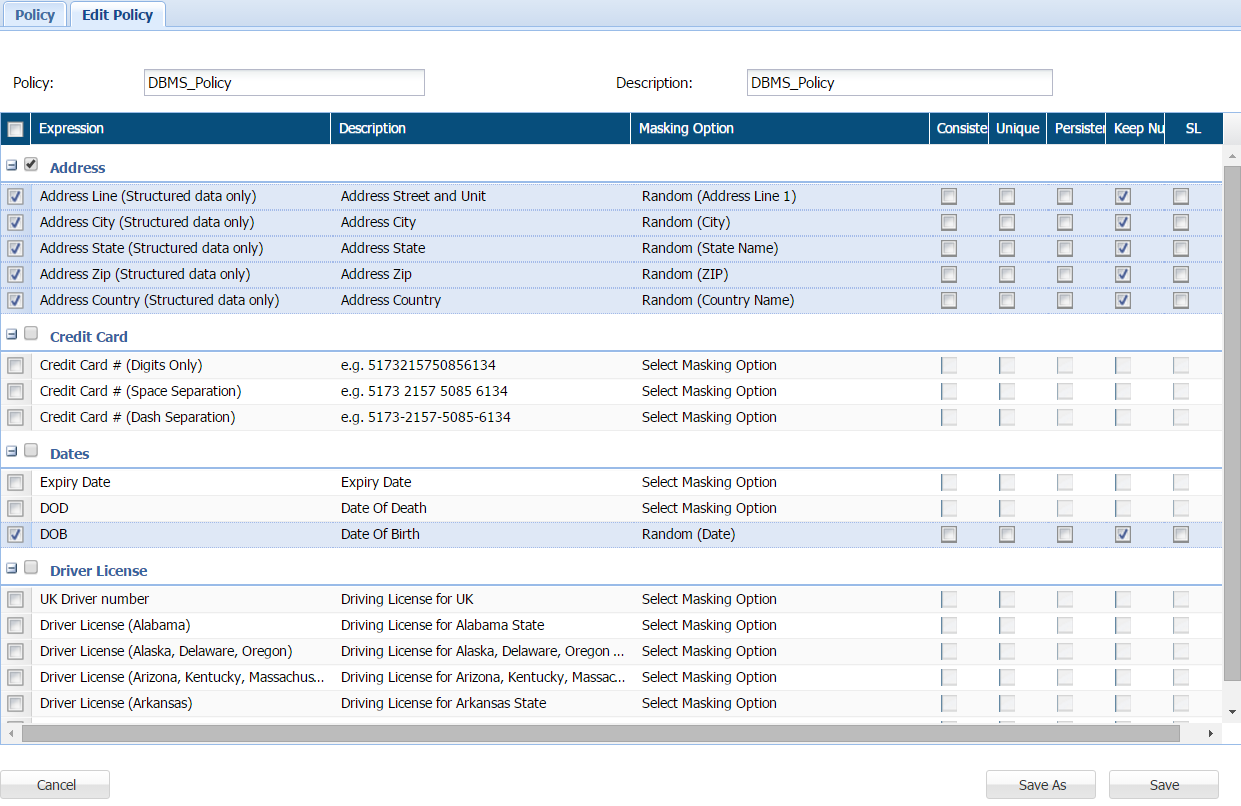
We will start with the Policy Section

## Create Policy

* On the left pane, click Policy – it will expand and the screen should be similar to below

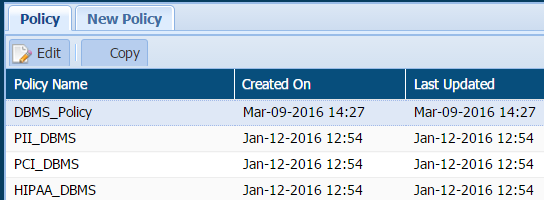


* On the top pane, highlight **‘PII\_DBMS’** policy
* Press the **Copy** button



* Fill in the Policy Name and Description field:  **DBMS\_Policy**
* Click **Save** (lower right)

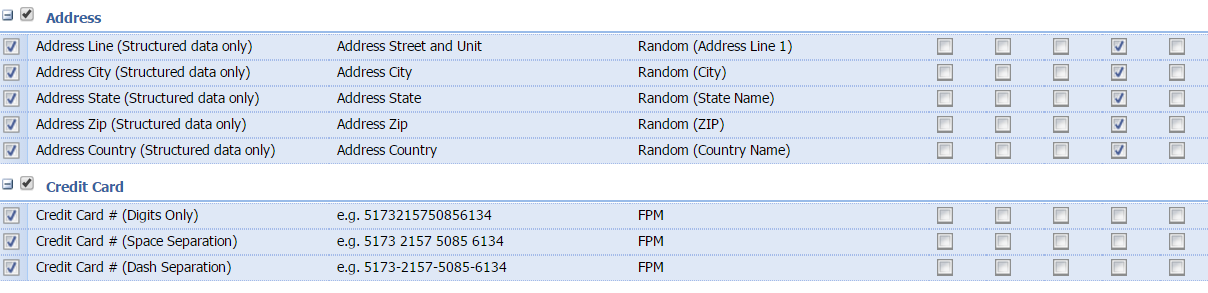
Your new policy should be listed



* Select **Edit**

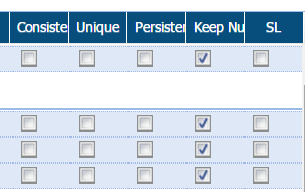
There are already numerous sensitive elements already checked within this policy, but you can add more as needed. Check the following sensitive elements: Credit card and IP Address. Press Save.

**NOTE:** There are many masking options that are already set at the policy level such as Random addresses, SSNs, Names, plus FPM (Format Preserve Masking). You can also check off additional entries to activate them in your new **DBMS\_Policy** such as Credit card. Check the Credit Card entries (as shown below).



Press **Save.**

Leave the existing masking option alone for this evaluation. Also do not check off any of the additional columns on the right side of the policy (shown below). We will demonstrate many masking options when we get to the masking of specific tables.



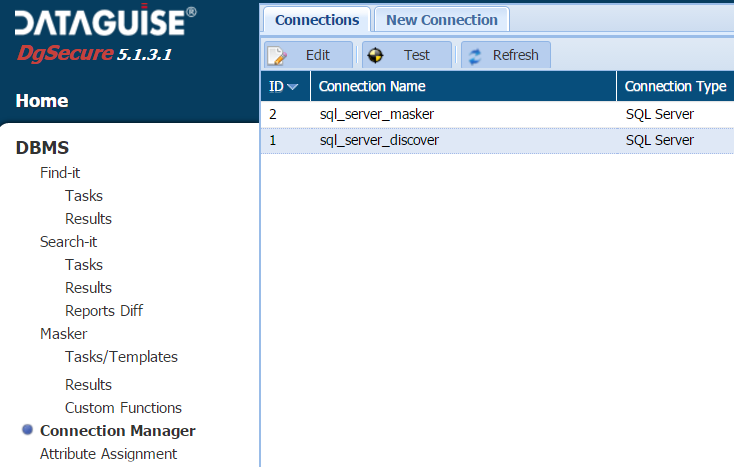
## Discover Sensitive Data within SQL Server Database

Dataguise’s Database Discovery is a very powerful option for today’s DBAs. Understanding every column in every table in every database is almost impossible nowadays. Dataguise will scan all of the targeted databases and tables to find all potential sensitive fields and bring awareness to areas within the database that should be masked when the time arrives.

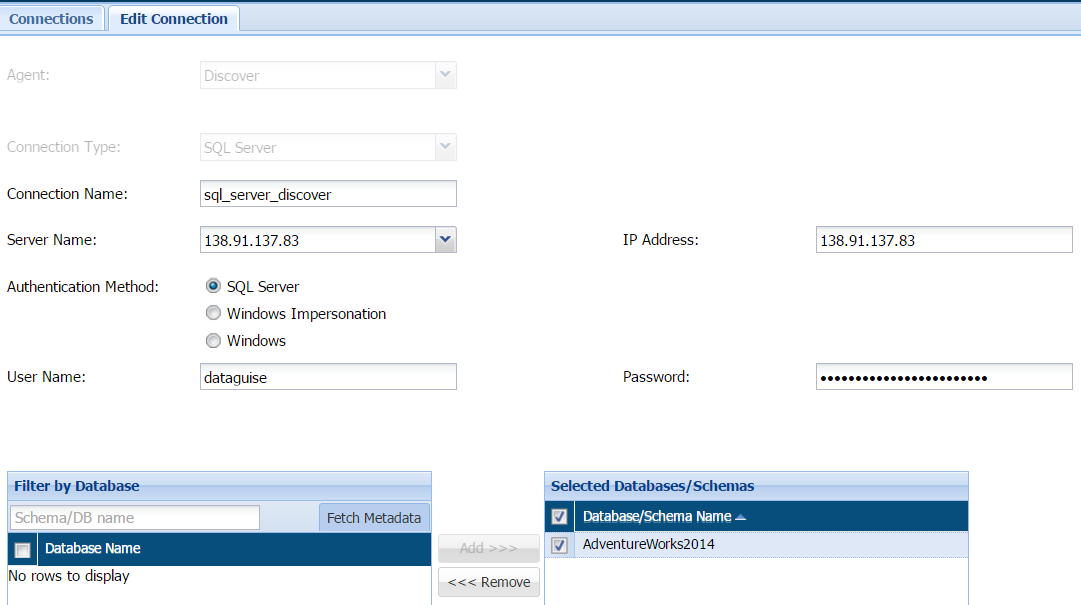
In order for DgSecure to scan the database tables, it requires a database user that has access to read all of the targeted tables. This is accomplished via the Connection Manager.

## Connection Manager

The Connection Manager contains the connectivity information for DgSecure to establish a connection with the targeted databases. Select DBMS 🡪 Connection Manager



* Highlight **sql\_server\_discover**
* Select **Edit**. 

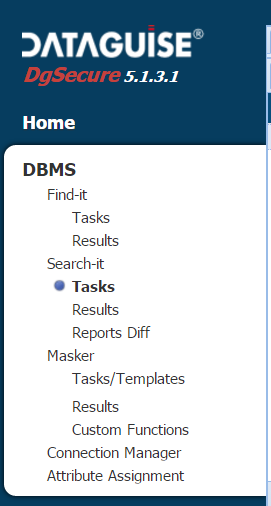


This screen displays the basic IP address, and userid/pwd to connect to this SQL Server database.

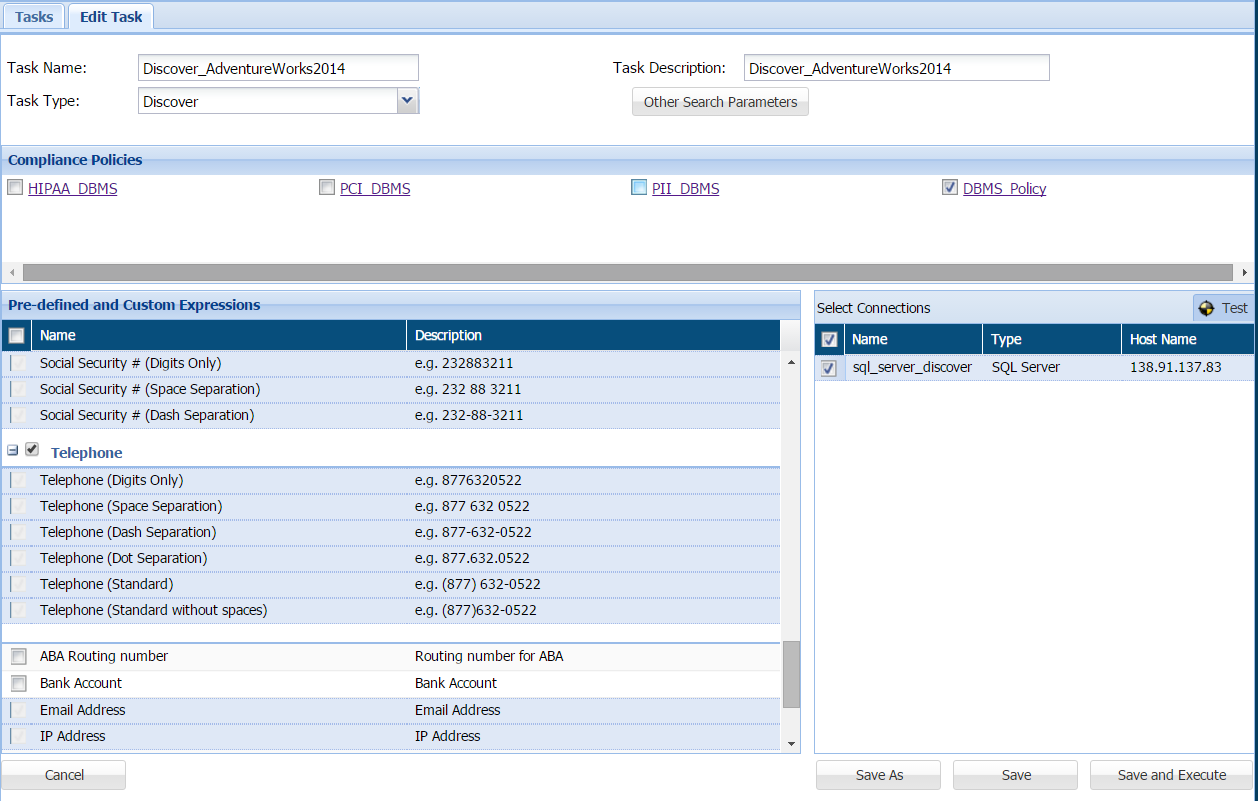
It has been prebuilt for this test database. If multiple databases exist on each database – the Fetch Metadata button extracts all databases and users can add each to the right side for Discovery. This demo will concentrate only on 1 test database.

## Create a Discovery Task

* On the left pane, click **DBMS 🡪 Search-it**

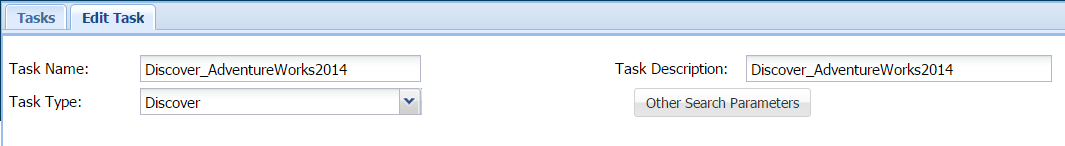


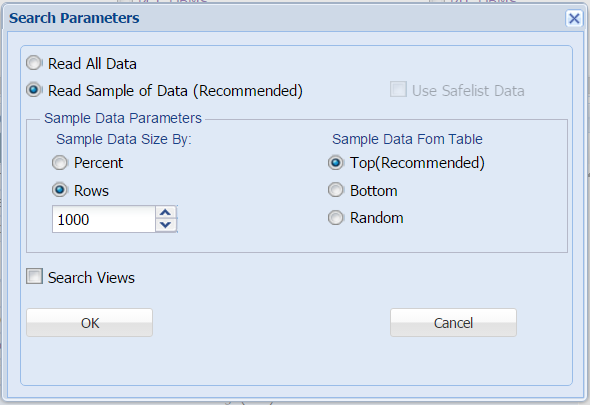
* This will bring up the Search-it Task Window
* 
* Select **New Task**



* A blank task will be displayed.
* Enter in the Task Name: **Discover\_AdventureWorks2014**
* Enter into Task Description: **Discover\_AdventureWorks2014**
* For Task Type: Select **Discover**
* Also check the **DBMS\_Policy** box. All selected sensitive items in that policy will be displayed in box on the bottom left area.

If you want to view the scanning options, press the Other Search Parameters button on the top right area of the Task.

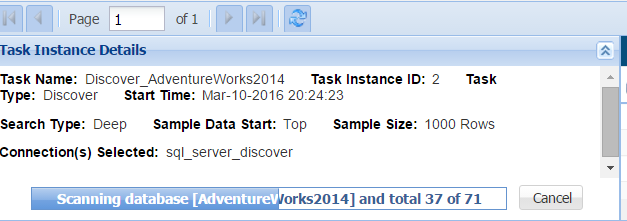




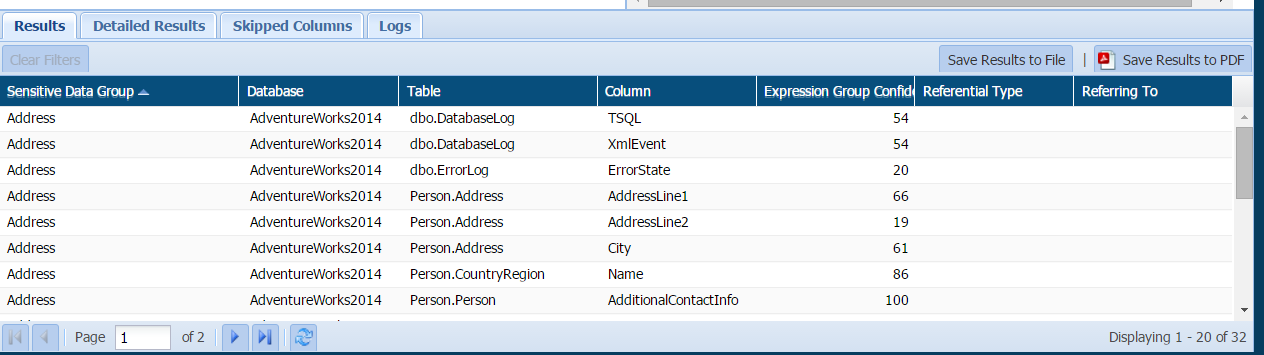
These are the default options. DgSecure will read 1000 rows from each table. You can modify this to scan a percentage of the table of the entire table.

* Press **Save and Execute** on the task screen (bottom right)

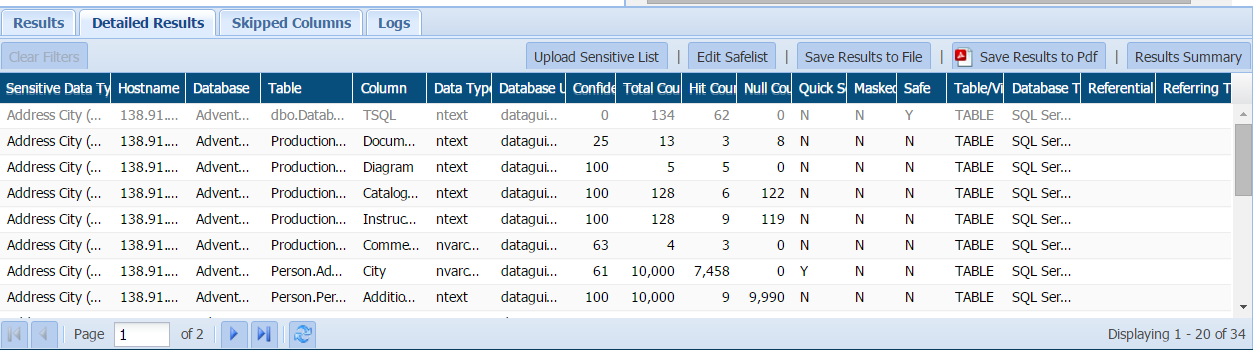
The job will proceed canning all of the tables. It is a multithreaded process that will split up all tasks between all available CPUs. You can watch the progress in the box below. (37 of 71 tables).



When the task completes, the bottom half of the task window will populate with columns of discovered entries.

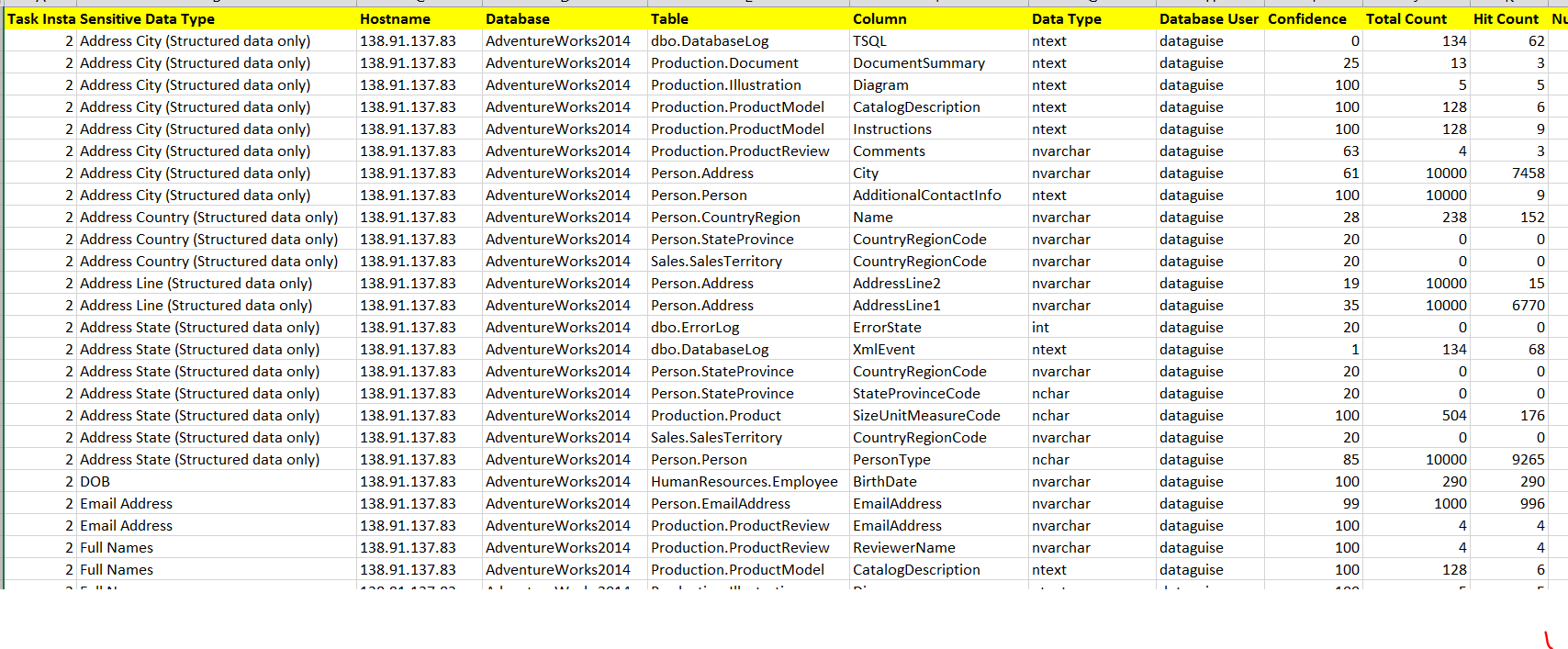


Click on the Detailed Results tab. This tab contains the Confidence Level information.

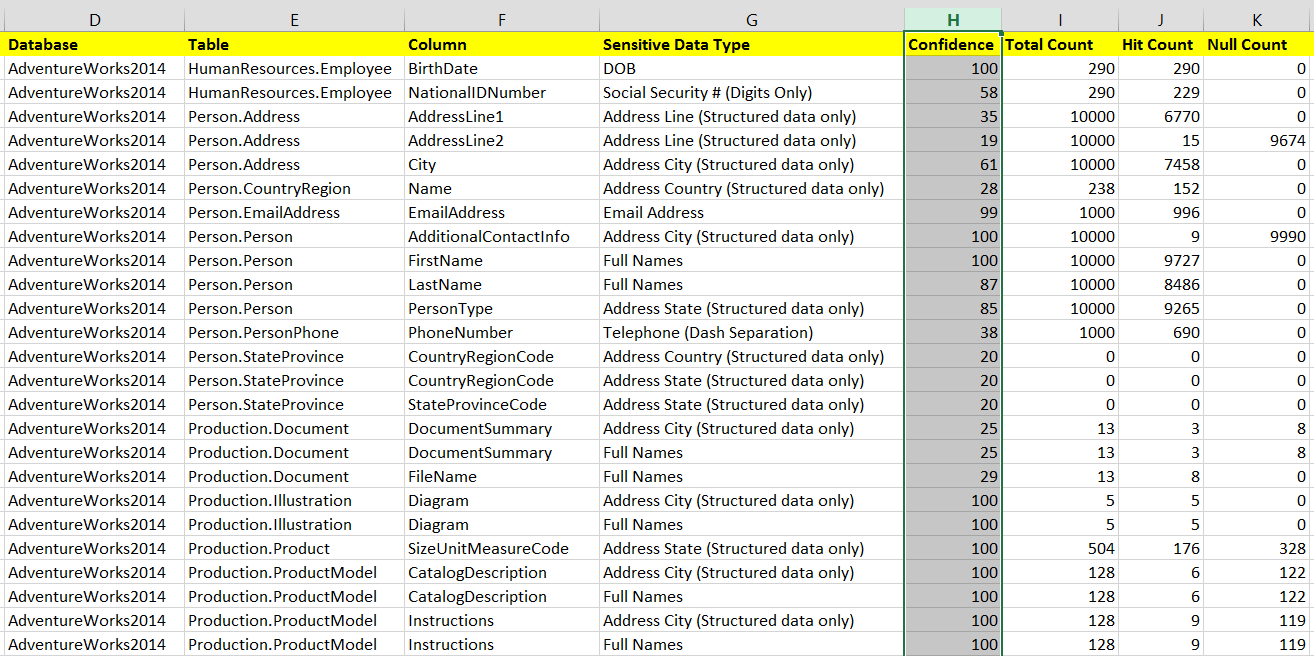


Press **Save Results to File** button (shown above).

This will open up a spreadsheet with all of the discovered sensitive entries. I colored the column headings yellow.



Looking at this spreadsheet in more detail – sorting the columns, moving some of the columns around and also highlighting the Confidence column shows the following:



This is the very informative power of DgSecure. The very first column shows that the Table **HumanResources.Employee**, and the column **BirthDate** has a **100%** confidence that the DOB column contains Date of Birth information. The **Confidence level is 100** because the **Total Count** was 290 rows and the **Hit Count** was also 290, plus the **Null Count** was 0 – meaning all 290 rows had a value that resembles DOB.

Looking down at another example, it shows that the Table **Person.Person**, and the column **LastName** has an **87%** confidence that the LastName column contains Full Name information. The **Confidence level is 87** because the **Total Count** was 10,000 rows and the **Hit Count** was also 8,486, plus the **Null Count** was 0 – meaning all 8,486 out of the 10,000 scanned had LastName values. DgSecure will go past the default 1000 rows if the data collected does not match the same hit rate as the batches within that 1000. Meaning it will read another 1000 and get that Confidence level. If it is close to the first 1000, then it will stop. In this case, it did 10 rounds to find a consistent grouping around 87%.

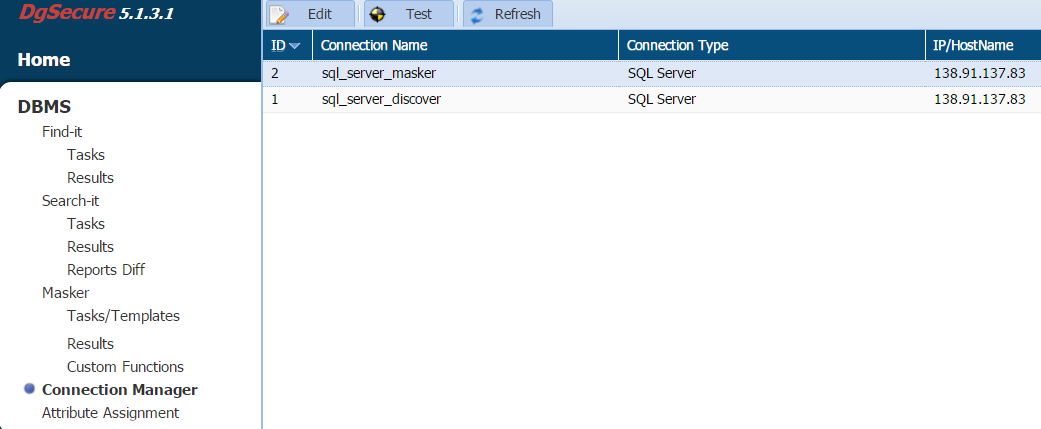
Another example shows that the Table **Production.Illustration**, and the column **Diagram** has a **100%** confidence that the **Diagram** column contains Name information. The **Confidence level is 100** because the **Total Count** was 5 rows and the **Hit Count** was also 5, plus the **Null Count** was 0 – meaning DgSecure found 5 Names in all 5 rows.

These spreadsheets are next reviewed by the security teams to verify that these columns do contain sensitive information. There could be entries that were flagged as potentially sensitive and those entries can be added to the SafeList – meaning do not flag these table/row

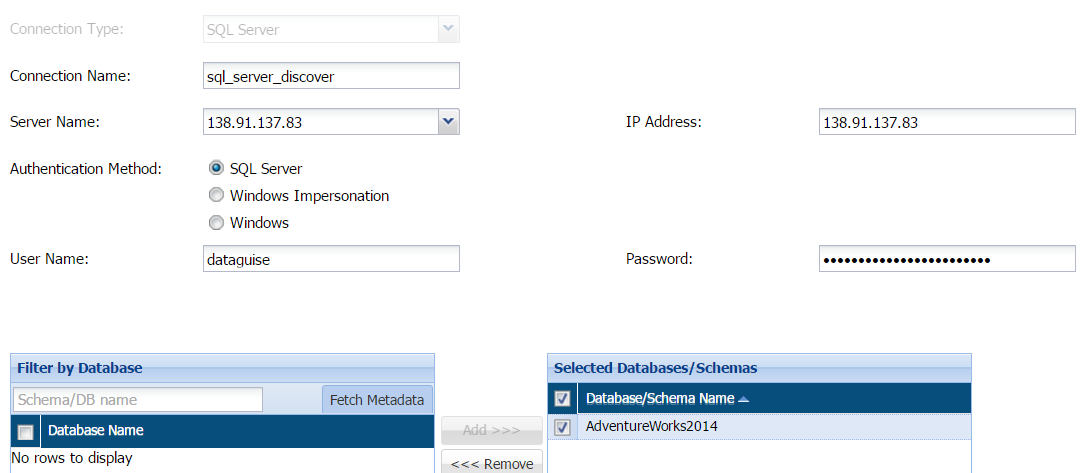
## Connect Directly to Database

In order to actually look at the table data to verify the sensitive data fields discovered above, some users may want to connect directly and view the data. This can also be helpful to view data both before and after masking.

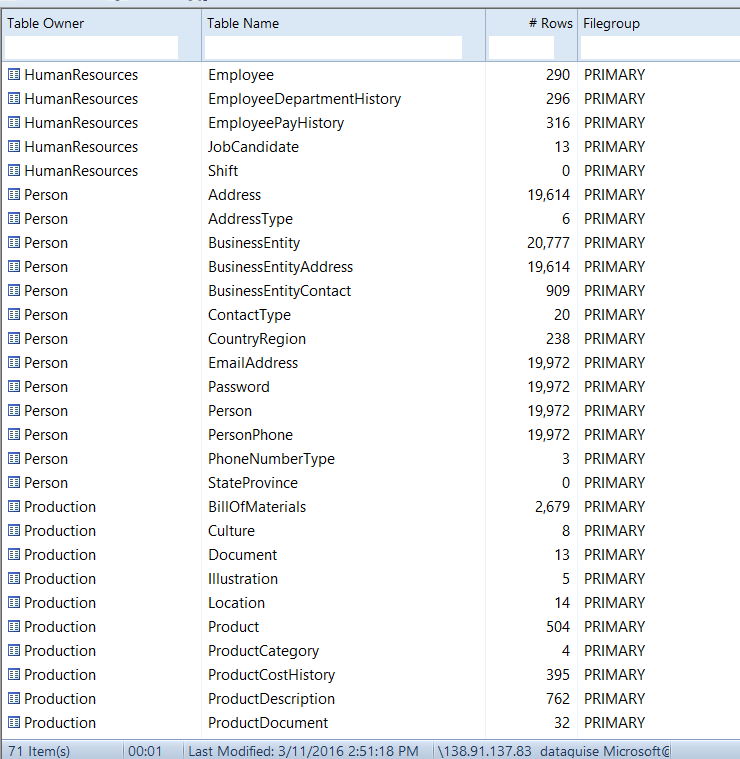
* On the left pane, click DBMS 🡪 Connection Manager



* The IP Address of the test database us displayed on the right side of the screen. To simply view the connection (and not make changes) highlight the **sql\_server\_discover** entry and select **Edit**



* The IP Address is displayed. Yours may be different
* The userid is **dataguise** (is displayed) and the password is **Microsoft@123**
* You can utilize any DBA or developer tool to connect to the listed IP and view the tables in AdventureWorlks2014. My screenshots are using the DBA tool: DBArtisan.
* This screenshot of DBArtisan shows a portion of the AdventureWorks2014 tables and row count



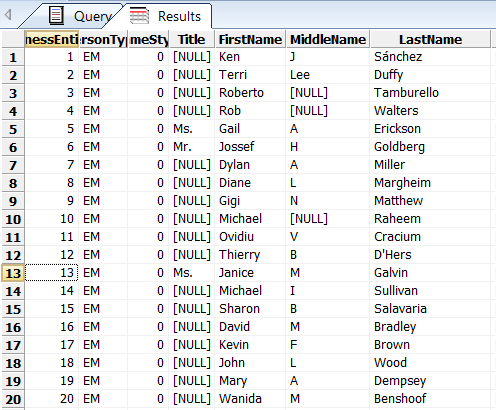
## Masking the SQL Database Person Table

Based on the findings of your Discovery Search-it task earlier, we can now target tables and columns to mask.

* We will start off simple by masking the Person.Person table
* We found sensitive entries for the **FirstName** and **LastName** columns
* Below is an excerpt from the Discovery Results spreadsheet showing the Person.Person table and the initial 20 rows of FirstName and Lastname – sorted by BusinessEntityID

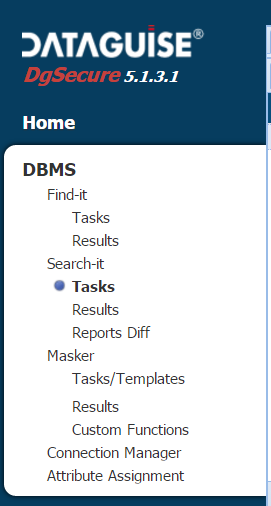


* Looking at the table data via any DBA Tool shows the first 20 rows

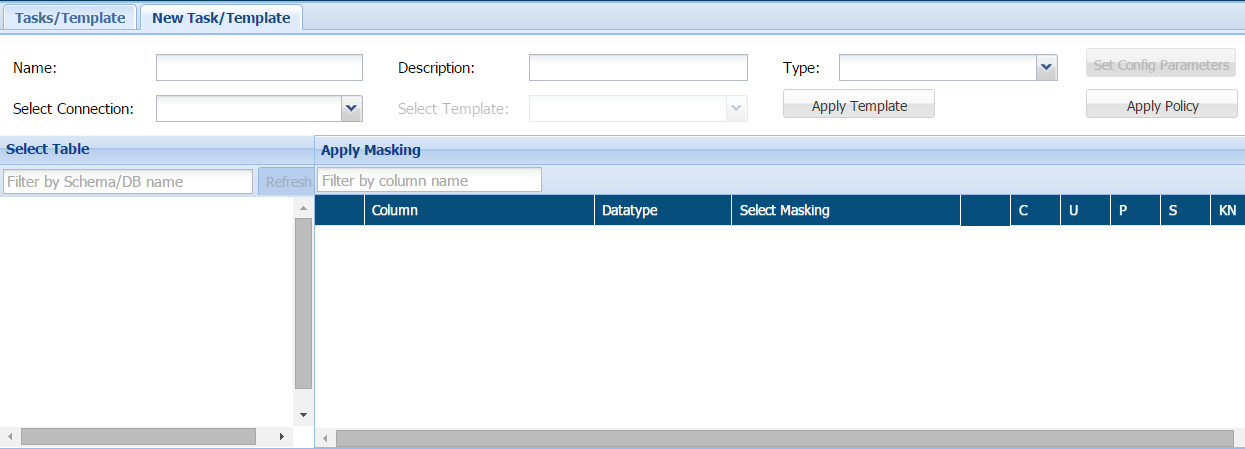


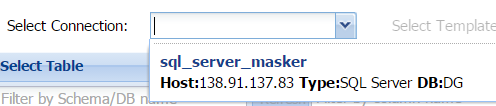
## Create 1st Masking Task

* On the left pane, click **DBMS 🡪 Masker 🡪 Task/Templates**



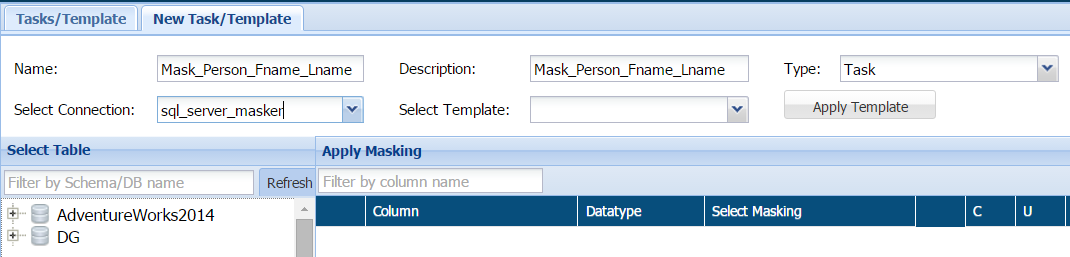
* This will bring up the Masker Task Window
* 
* Select **New Task/Template**



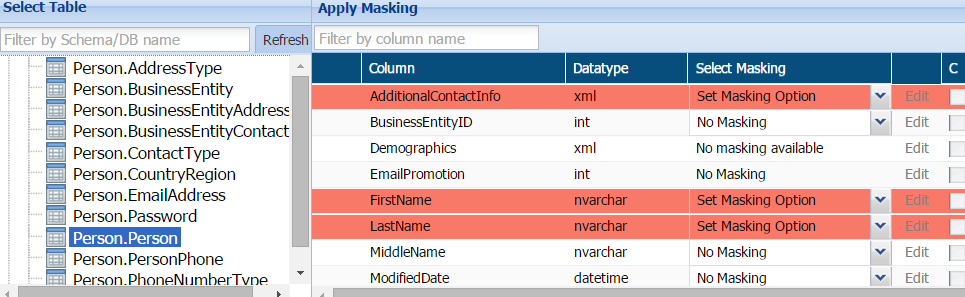
* The entries will be blank as shown above
* First define the Name: **Mask\_Person\_Fname\_Lname**
* Enter Description: **Mask\_Person\_Fname\_Lname**
* Type: **Task**
* Select Connection: There should be 1 pulldown – select that entry
* 

The databases AdventureWorks2014 and DG should appear. DG is the database that Dataguise will create to install all of the tables, procedures and functions that will be utilized during the masking process. Remember Dataguise DgSecure is not for Production databases, but only for copies of Production that will be directly masked and released to development. Running the code directly against the local database makes this efficient and very fast. It is also a multi-threaded architecture that will split up all of the masking tasks into smaller parallel threads as available.

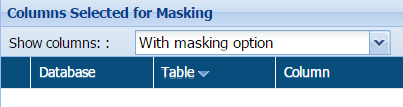
* You should see this at this point



* Expand out the AdventureWorks2014 database until you select the **Person.Person** table
* The sensitive columns discovered will be colored

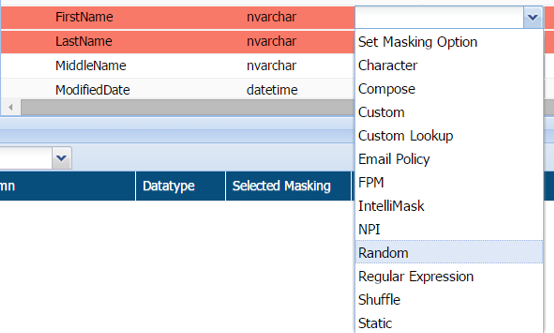


* Also for the bottom pane, select the pulldown to show **With masking option**

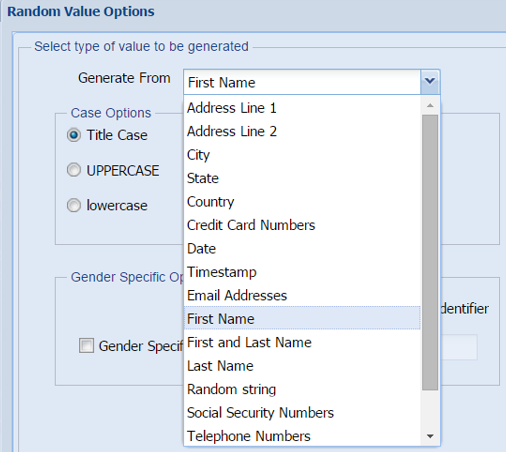


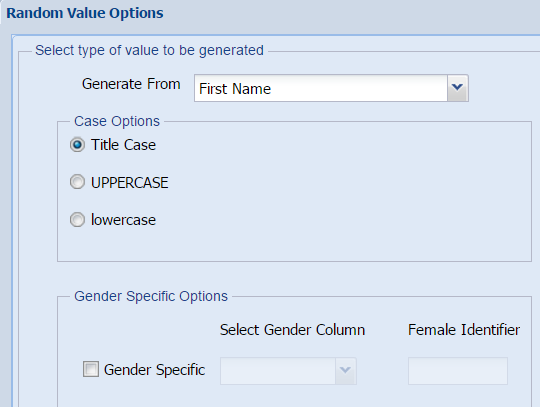
NOTE: DgSecure can apply your entire policy to all sensitive data in 1 step, but for this demo script, we will walk you through individual tables and individual masking options.

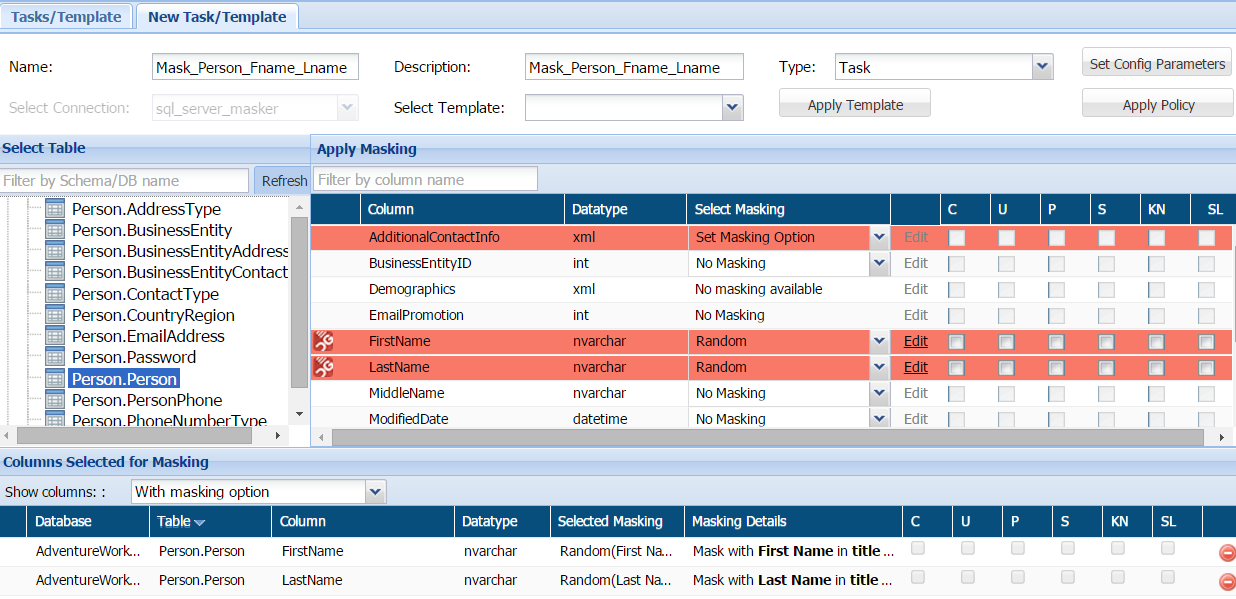
* The next step is to select the masking options for the **FirstName** and **LastName** columns
* Select the pulldown arrow next to **FirstName** and this shows the multiple masking options available. Select **Random**



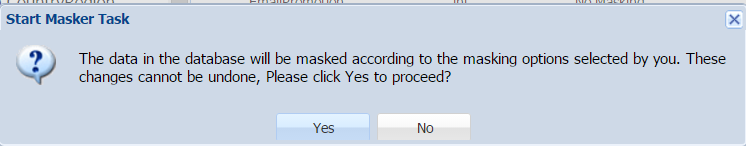
* In the next sub-window, select **First Name** as the Random option



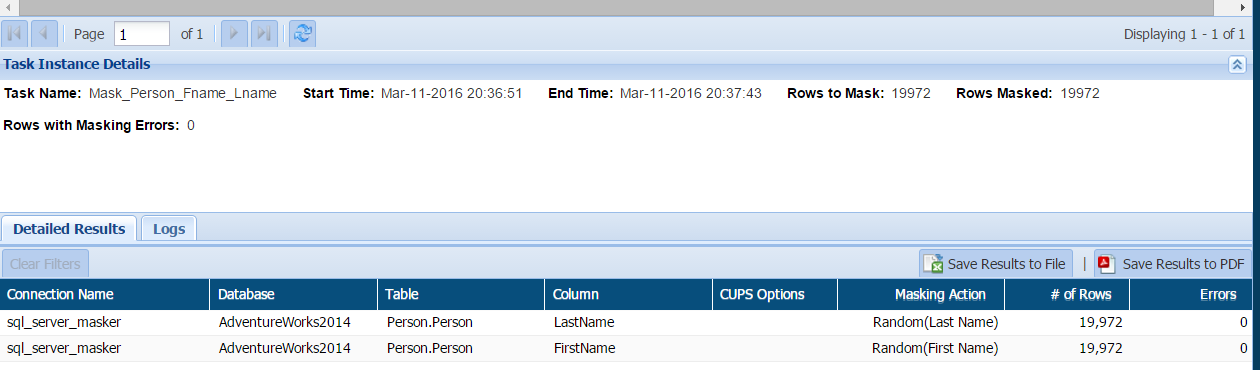
* Select **Title Case** (meaning first letter capitalized)
* 
* There is no M/F column in the Person table to associate male and female names.
* Select **OK**.
* Do this again for **LastName** column and select **Last Name** for the Random entry
* We will not be selecting any of the **C U P S KN SL** options for this first test
* Your screen should resemble below



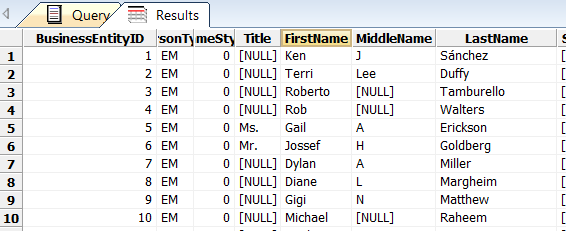
* Select **Save Task and Execute** (bottom right)
* This warning will appear



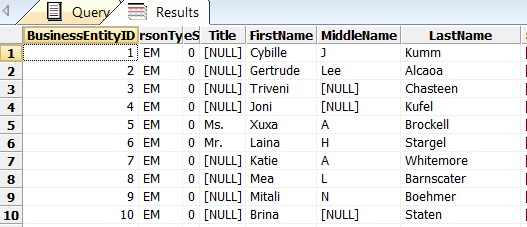
* Press **Yes**
* The very first time masking any table on this database will take longer because it will be creating all of the tables, 1400+ procedures, functions etc. in the DG schema that were not there yet.
* Once done, the summary screen will look similar to below



* 19,972 rows masked and 0 errors
* Now let’s look at the after-masking data.
* Here is the 1st 10 rows of data **BEFORE** masking FirstName and LastName



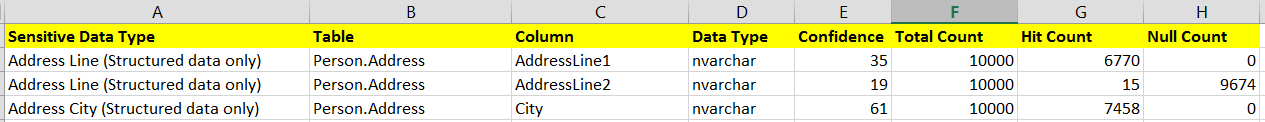
* Here is the 1st 10 rows of data **AFTER** masking FirstName and LastName



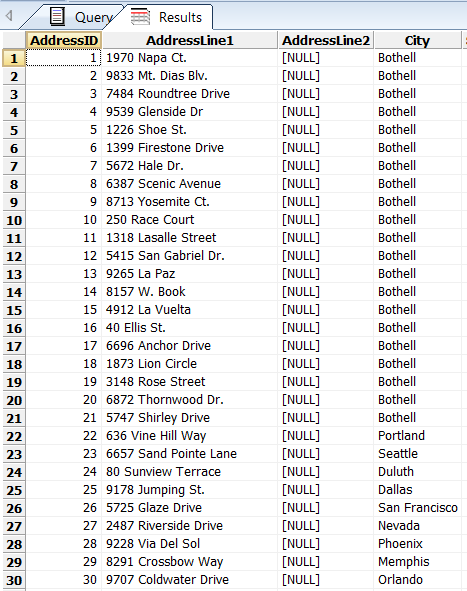
## Masking the SQL Database Address Table

Based on the findings of your Discovery Search-it task earlier, we can now target tables and columns to mask.

* We will now mask the Person.Address table
* We found sensitive entries for the **AddressLine1, AddressLine2 and City** columns.
* Below is an excerpt from the Discovery Results spreadsheet showing the Person.Address table and the initial 20 rows of **AddressLine1, AddressLine2** and **City** – sorted by AddressID



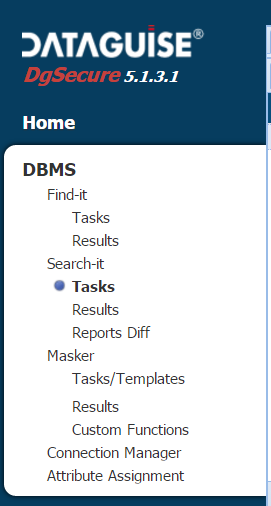
* Looking at the table data via any DBA Tool shows the first 30 rows



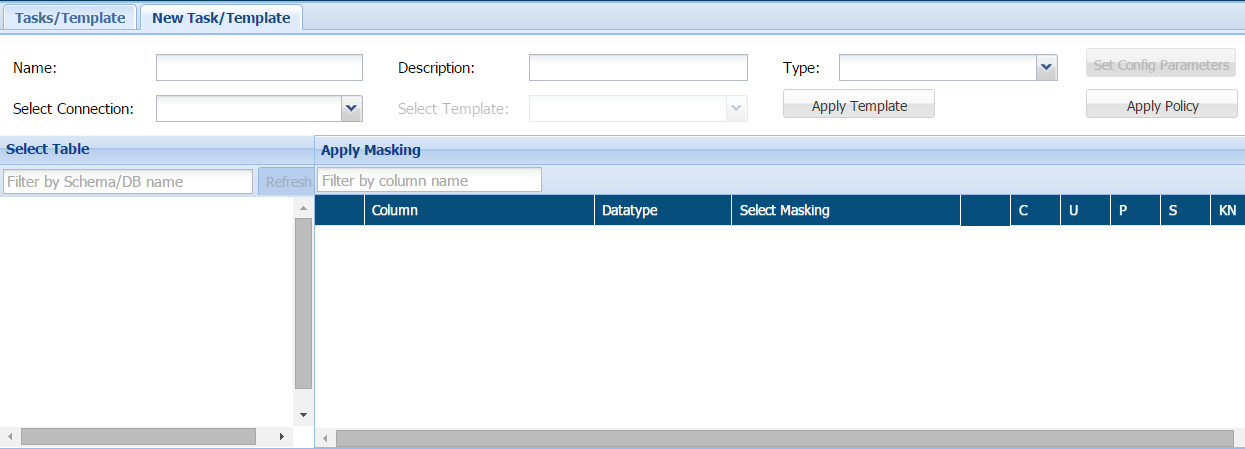
* Looking at the data values shows normal AddressLine1 values, but many nulls for AddressLine2. City is another sensitive field.

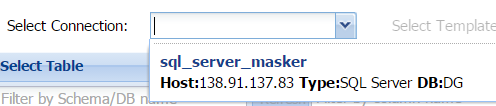
## Create 2nd Masking Task

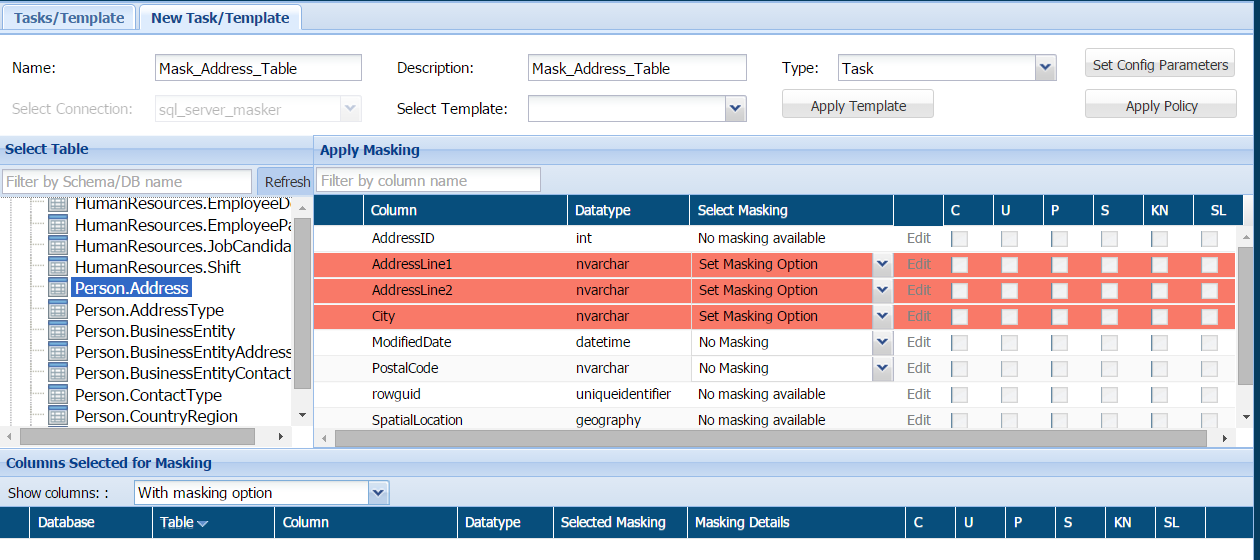
* On the left pane, click **DBMS 🡪 Masker 🡪 Task/Templates**



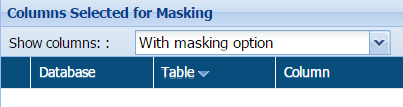
* This will bring up the Masker Task Window
* 
* Select **New Task/Template**



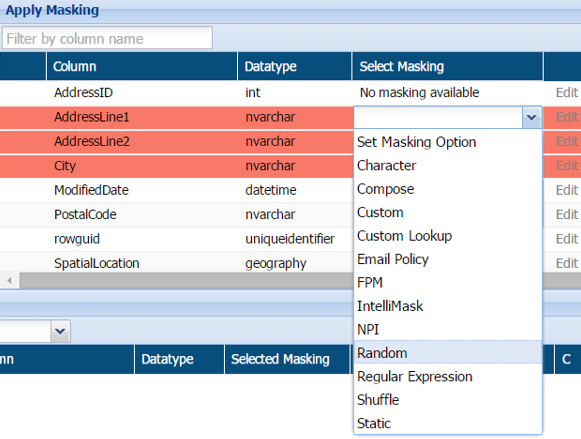
* The entries will be blank as shown above
* First define the Name: **Mask\_Address\_Table**
* Enter Description: **Mask\_Address\_Table**
* Type: **Task**
* Select Connection: There should be 1 pulldown – select that entry
* 
* Expand out the AdventureWorks2014 database until you select the **Person.Address** table
* The sensitive columns discovered will be colored



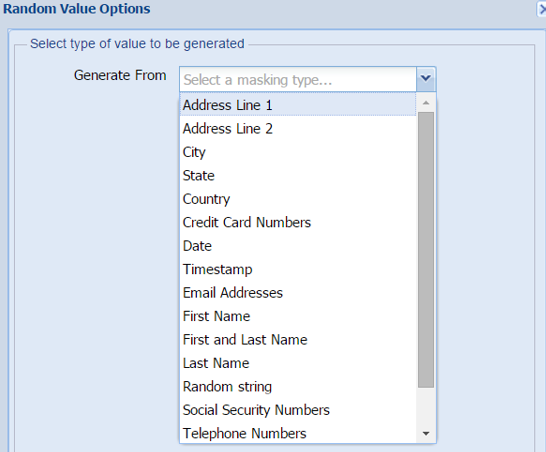
* Also for the bottom pane, select the pulldown to show **With masking option**

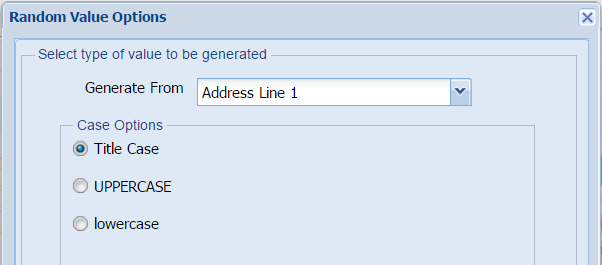


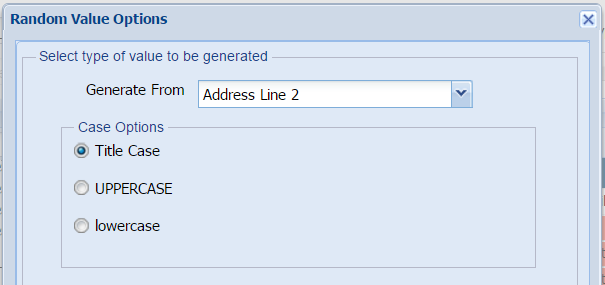
* The next step is to select the masking options for the **AddressLine1, AddressLine2 and City** columns
* Select the pulldown arrow next to A**ddressLine1** this shows the multiple masking options available. Select **Random**



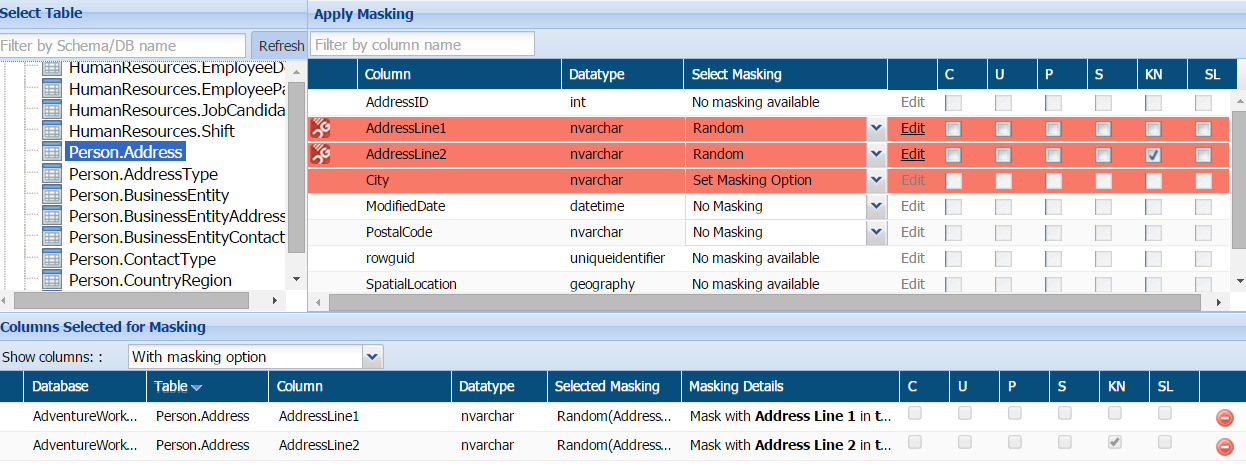
* In the next sub-window, select **AddressLine1** as the Random option



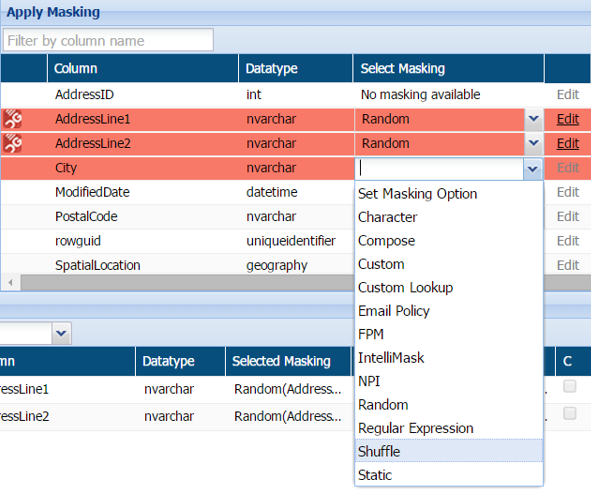
* Select **Title Case** (meaning first letter capitalized)
* 
* Select **OK**.
* Do this again for **AddressLine2** column and select **Address Line 2** for the Random entry plus Title Case.

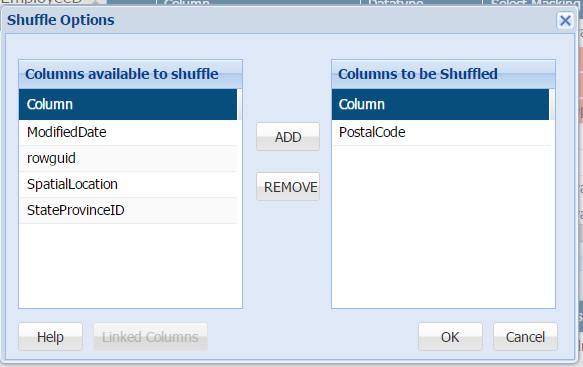


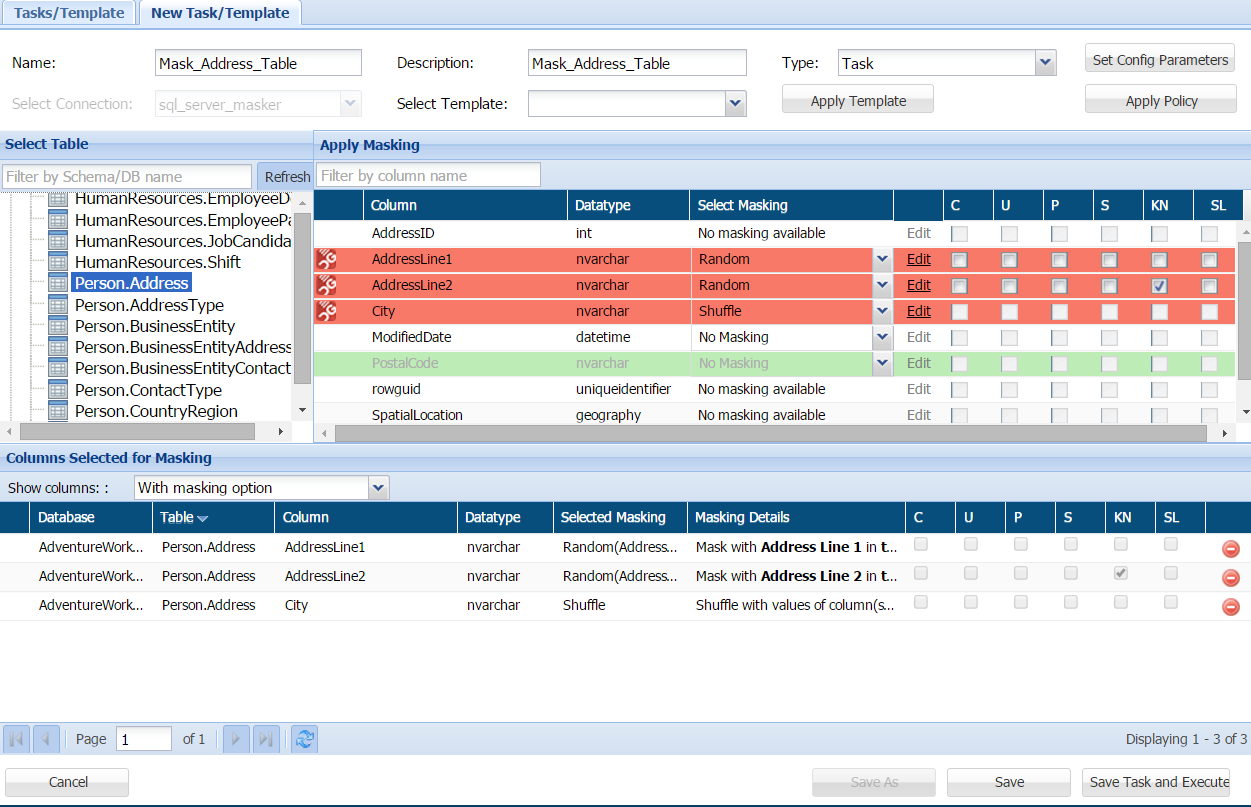
* If you noticed above, most of the AddressLine2 entries were null. In fact most of the 19,614 entries were null so we will allow those to stay null, and insert new random addresses for those that have entries.
* We will be selecting the **KN (meaning Keep Null)** option in the **C U P S KN SL** options for this column.
* Check the **KN** box next to AddressLine2.
* Your screen should resemble below sofar



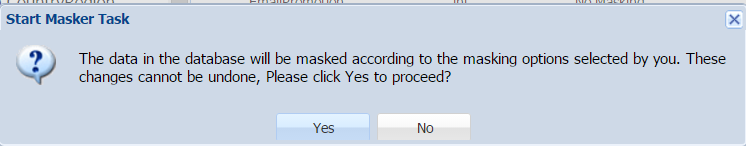
* We will also mask the **City** column.
* Select the pulldown arrow next to City and select **Shuffle could also group**



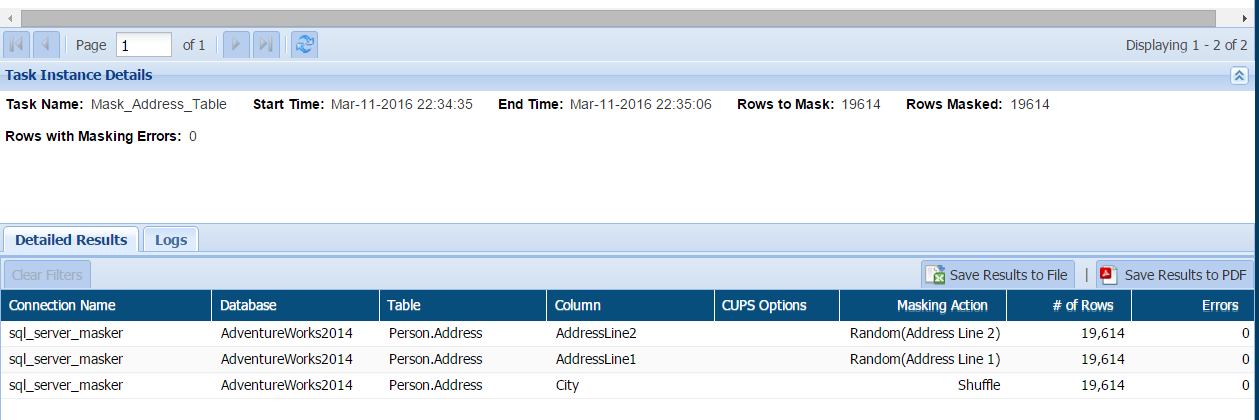
* This option is like shuffling a deck of cards. This will maintain the complete list of Cities but shuffle them around. This means that is there are 13 entries for Dallas in the table, there will also be 13 entries for Dallas but shuffled around. This is a good option to maintain distributions for geographic areas. You could also group the State and Zipcode if you wanted and keep them all intact in a shuffle.
* Let’s associate PostalCode with City and keep them together.
* 
* Select **PostalCode** and select **Add** to move it into the right-hand column.
* Your screen should look similar to this



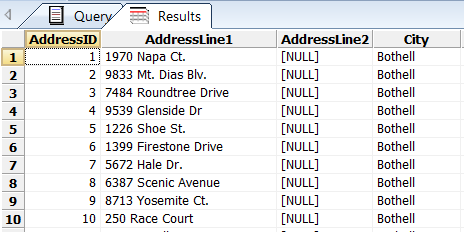
* The PostalCode column is now green signifying it is linked to a Shuffle operation.
* Select **Save Task and Execute** (bottom right)
* This warning will appear



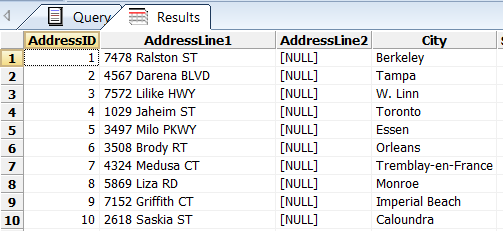
* Press **Yes**
* The 2nd masking job should be very fast because all of the tables, 1400+ procedures, functions etc. in the DG schema were created during the last task.
* Once done, the summary screen will look similar to below



* 19,614 rows masked and 0 errors
* Now let’s look at the after-masking data.
* Here is the 1st 10 rows of data **BEFORE** masking **AddressLine1, AddressLine2 and CIty**



* Here is the 1st 10 rows of data **AFTER** masking **AddressLine1, AddressLine2 and CIty**



The AddressLine1 entries received new addresses.

The AddressLine2 (if null) remained null. Others got new addresses.

The City column was shuffled around.

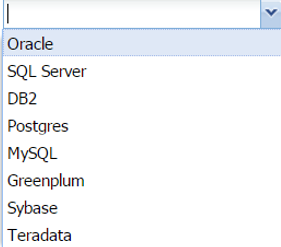
## Summary

This was a simple guide to navigate a novice user through the basic screens related to DgSecure and the Discovery and asking of sensitive information.

There is much more depth and functionality to this product than what we showed you here in this guide. The product picks up all physical referential integrity constraints within each database and is smart enough to build the masking scripts in the proper order for parent-child relationships, when to disable constraints and when to re-enable them.

There are 40+ basic masking operations that cover all of the sensitive values we provided within the policy earlier on. You can also augment your policy to include custom attributes such as an ACCTID, or any database entry that you also want to consider sensitive. The AdventureWorks2014 database tables are not complex but are perfect for this first learning experience.

As you can see, the product is easy to navigate and easy to use. It also covers many other database platforms:



If you want to evaluate this product against your own company data, please contact sales and we can arrange it.

Thank you for your time.