In addition, the IDMF lets you analyze the database and maintain an optimal database size by providing the purge and archive functions.

Purge: R*emoves or deletes data from a set of related entities, or tables, from the production database*

*Archive: moves data from all related tables from the production database to a standby database called the archive database. Users can use the archive database for reporting purposes but cannot update it*

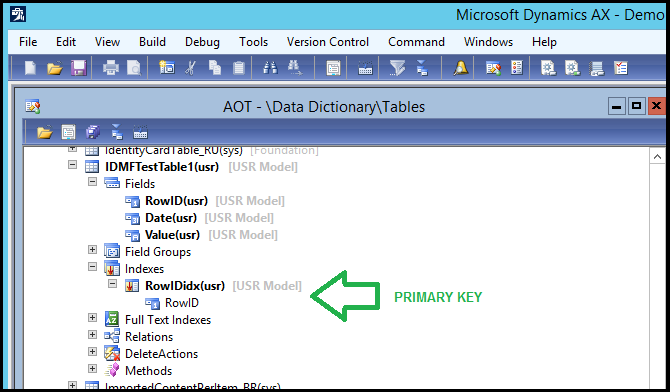
*Link:* [*https://community.dynamics.com/365/financeandoperations/b/axsupport/archive/2016/03/22/idmf-archiving-restoring-and-purging-data-from-a-custom-table*](https://community.dynamics.com/365/financeandoperations/b/axsupport/archive/2016/03/22/idmf-archiving-restoring-and-purging-data-from-a-custom-table)

Consider a scenario where you are running **AX 2012 R3** and you want to **archive** or **purge** data from a **custom table in AX** using **IDMF**.

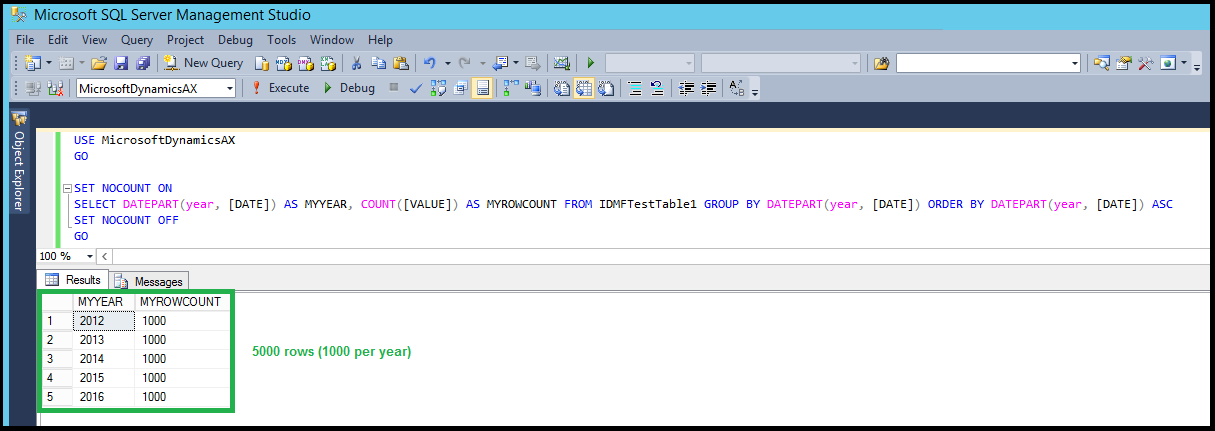
The screenshots are from a **VM** **TEST environment** with **CONTOSO DEMO DATA** that is running **AX 2012 R3 CU10** on **SQL Server 2014** and the latest version of **IDMF**.

**1. The custom table**

I’ve created a simple custom table called **IDMFTestTable1** which has three fields (**RowID, Date, Value**) and where **RowID** is the **Primary Key** (i.e. unique):

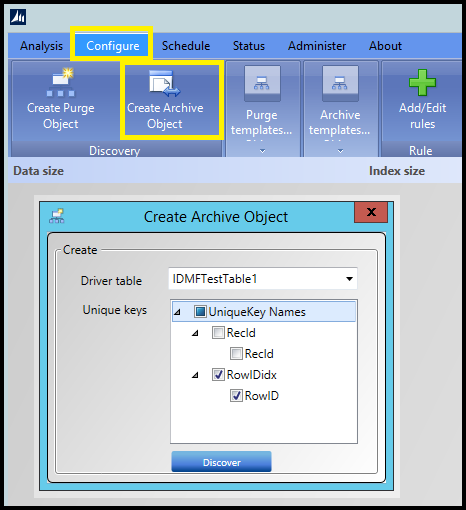
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA1.png)

I’ve inserted **1000 rows for each year between 2012 and 2016** in order to simulate data row growth over time, so I can demonstrate date based purging and archiving:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA2.png)

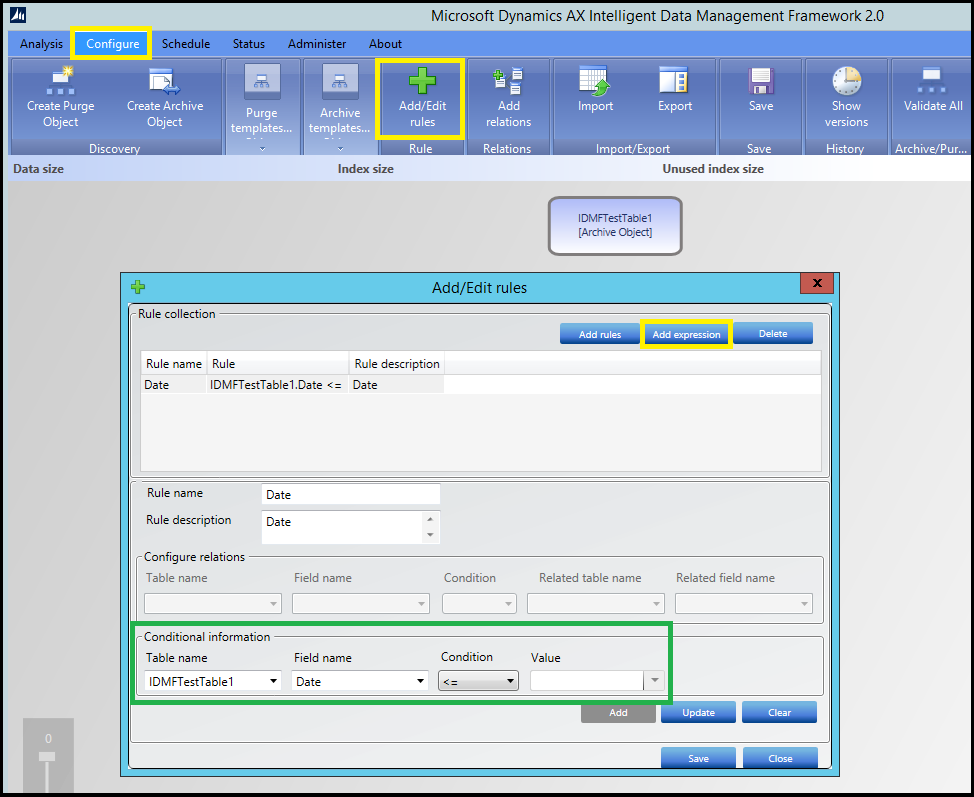
**2. Archiving data**

I begin by creating a new **Archive Object** from the **Configure** menu:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA3.png)

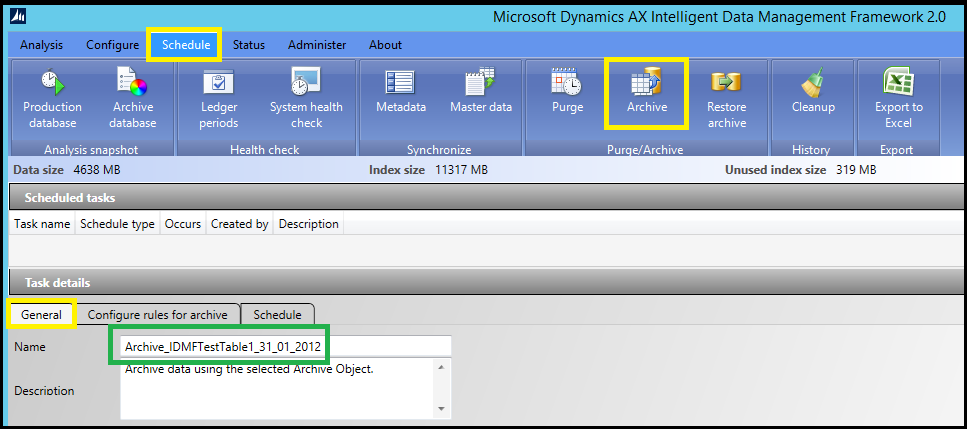
Note that I’ve chosen the **Primary Key** on **RowID** by checking it.

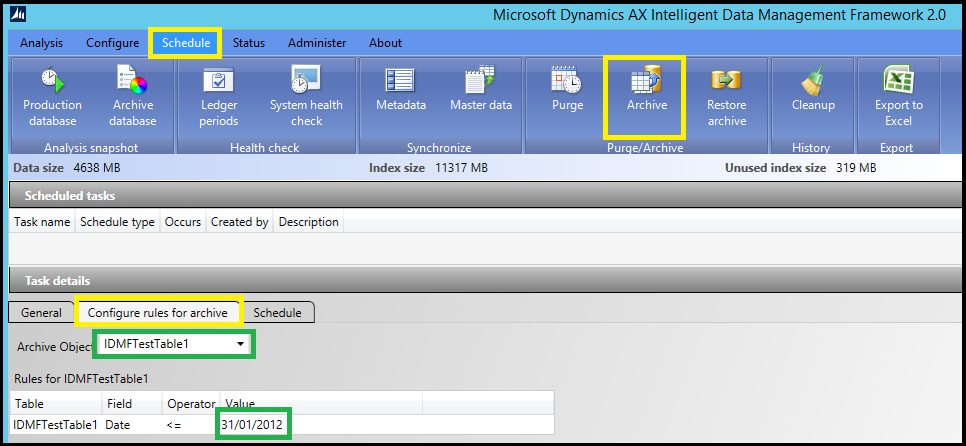
In this example, I am going to add an expression under by clicking on **Add/Edit rules** and then choosing **Add expression**:

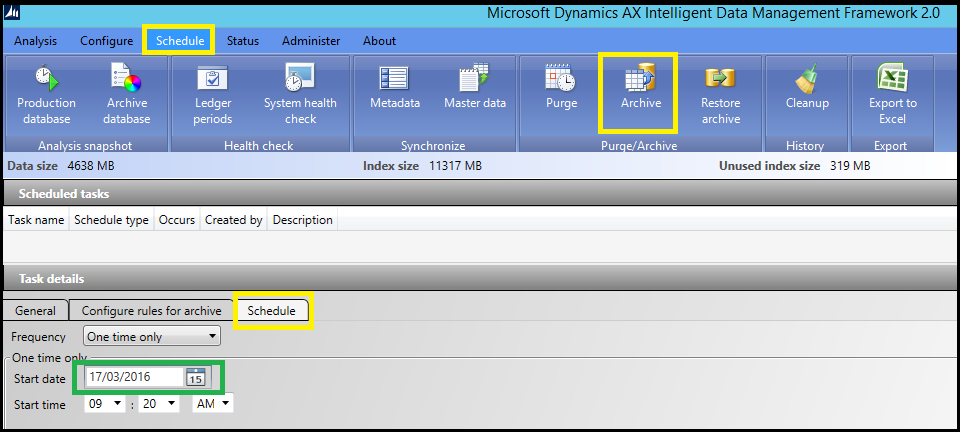
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA4.png)

Note that I’ve configured the rule to **only archive** rows from the custom table where the **Date field is <= to the value I enter when I schedule the archive job to run**.

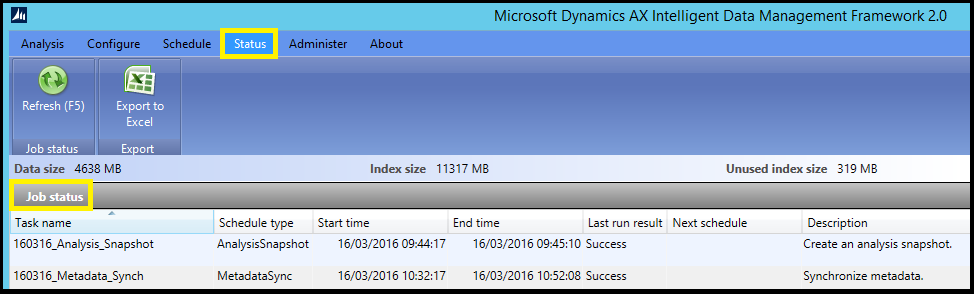
I now **schedule** the **Archive** job to run and **add the value I want to use for Date <=.** All this is done from the **Schedule** menu:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA5.png)

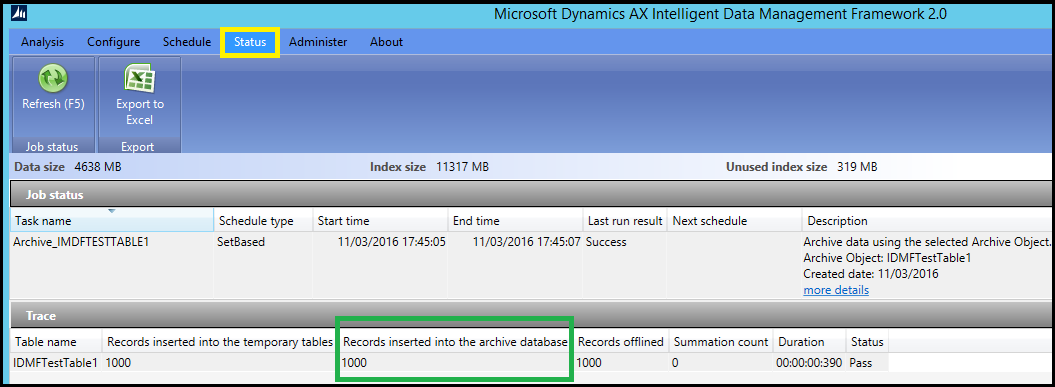
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA6.png)

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA7.png)

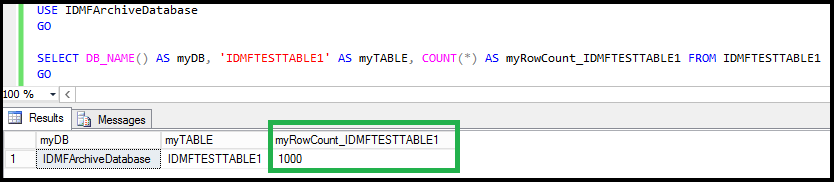
**Once you’ve scheduled the job**, it should appear in the **job status** list under **Status**:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA8.png)

When the job has run **successfully**, you can see **how many rows it inserted** into the **Archive** database:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA9.png)

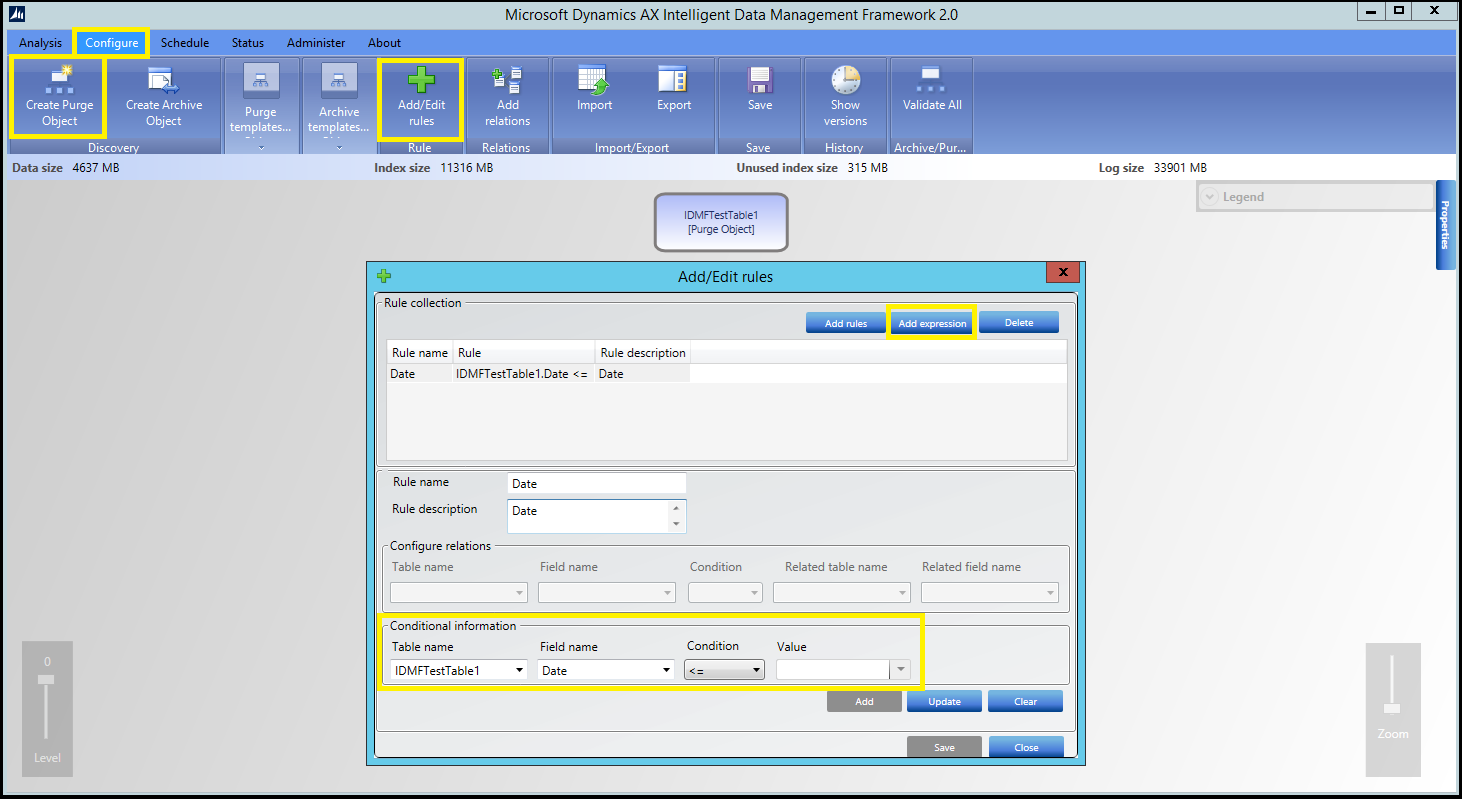
If I check my **Archive** Database, I can see that there are now **1000 rows** in the **archive table** stored there:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfA10.png)

**3. Purging data**

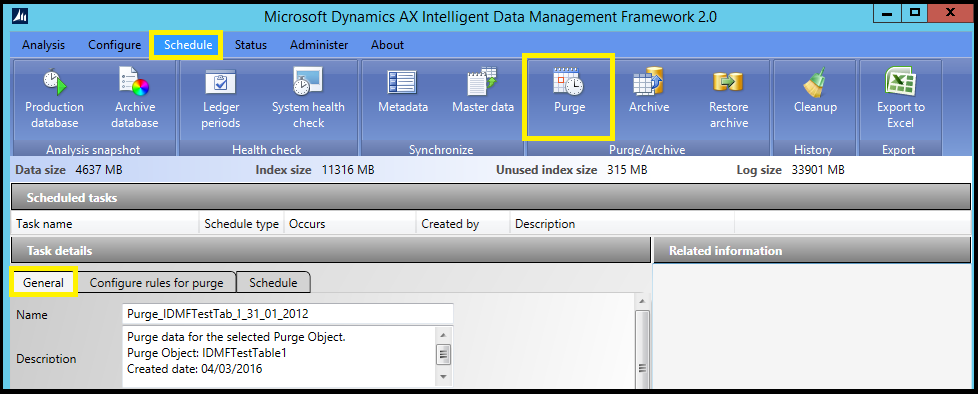
I begin by creating a new **Purge Object** from the **Configure** menu.

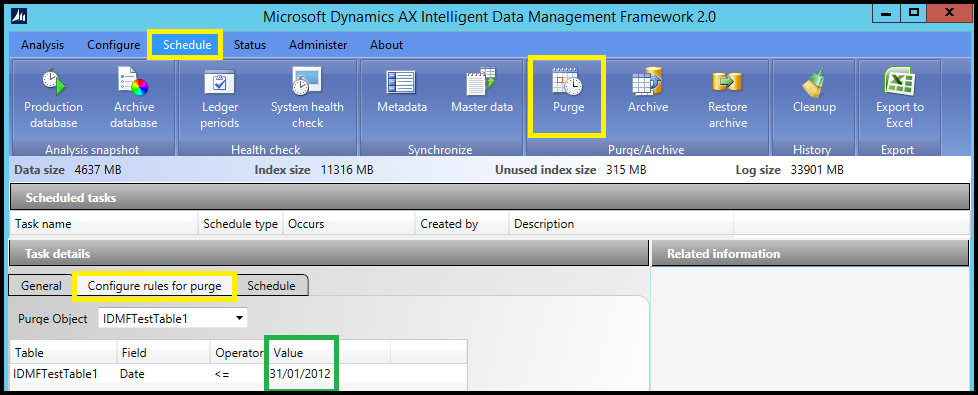
In this example, I am going to add an expression under by clicking on **Add/Edit rules** and then choosing **Add expression**:

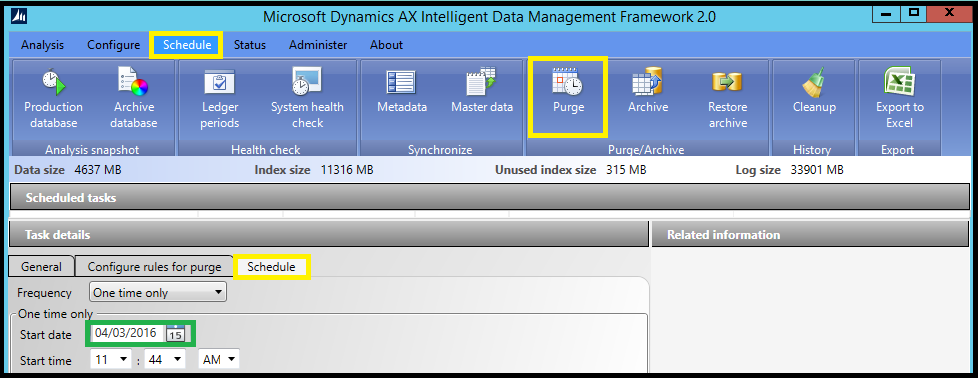
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfP1.png)

Note that I’ve configured the rule to **only purge** rows from the custom table where the **Date field is <= to the value I enter when I schedule the purge job to run**.

I now **schedule** the **Purge** job to run and **add the value I want to use for Date <=.** All this is done from the **Schedule** menu:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfP2.png)

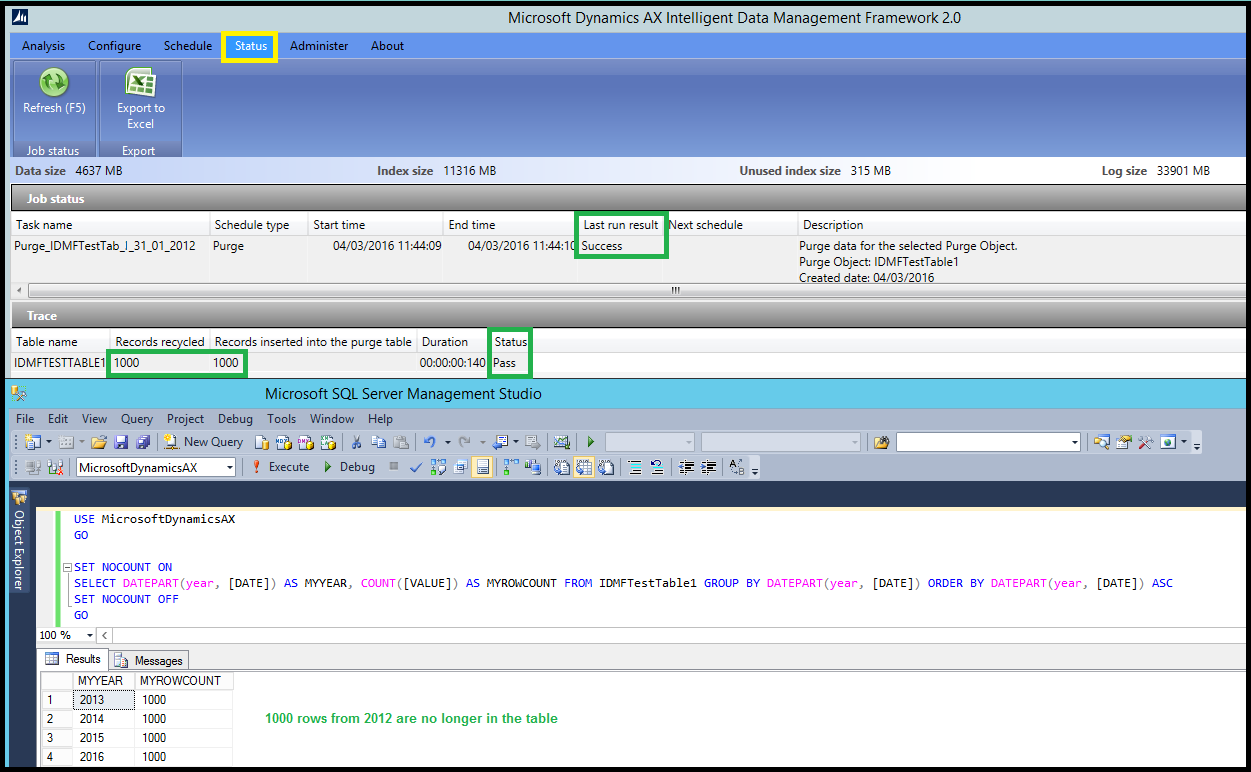
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfP3.png)

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfP4.png)

**Once you’ve scheduled the job**, it should appear in the **job status** list under **Status**.

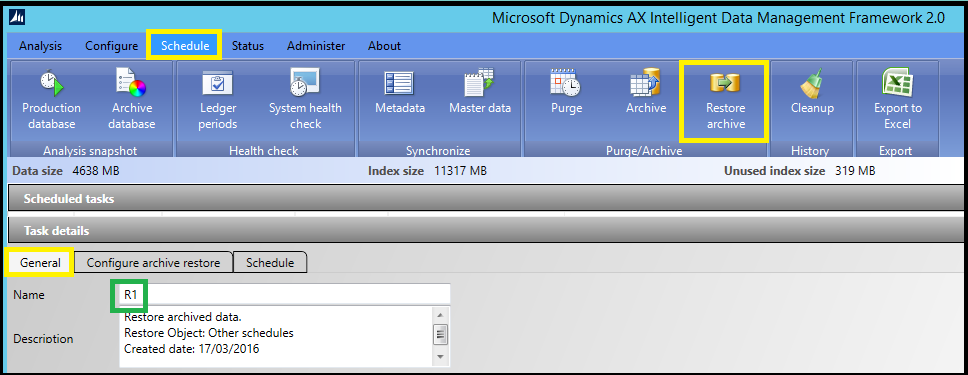
When the job has run **successfully**, you can see **how many rows it purged**.

If I check my **Production** Database, I can see that there are now **1000 rows** **less** in the **table** I purged the data from, as intended:

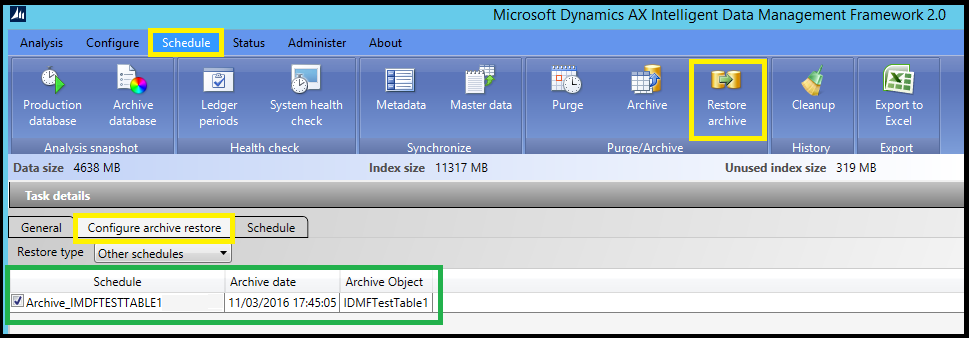
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfP5.png)

**4. Restore archived data**

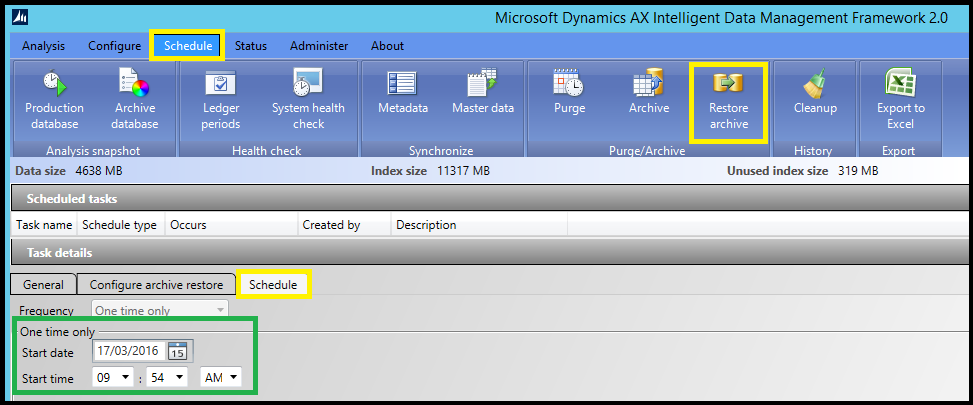
To **restore archived data**, go to the **Schedule** menu and click on the **Restore archive** button.

Name the job under the **General tab:**  
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfR1.png)

Select the relevant schedule under the **Configure archive restore tab**:

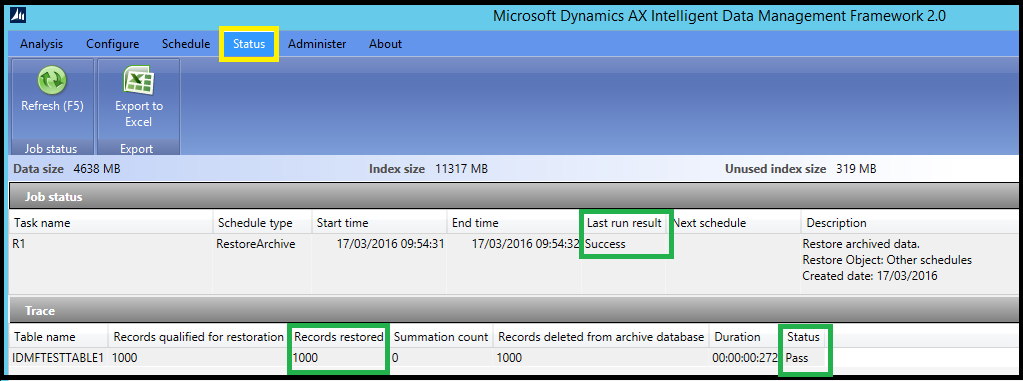
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfR2.png)

And finally, schedule the job to run when convenient under the **Schedule tab**:

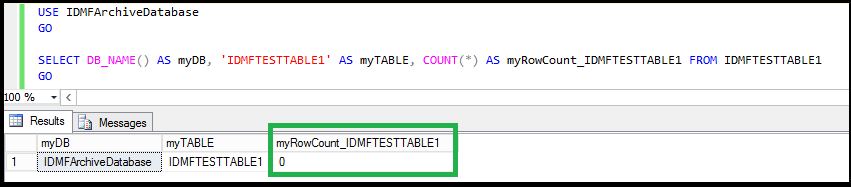
[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfR3.png)

**Once you’ve scheduled the job**, it should appear in the **job status** list under **Status**.

When the job has run **successfully**, you can see **how many rows it restored**.

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfR4.png)

If I check my **Archive** Database, I can see that there are now **no rows** in the **table** I restored the data from, as intended:

[](https://msdnshared.blob.core.windows.net/media/2016/03/idmfR5.png)

**5. Testing**

As always, ensure you’ve set up a proper **Dynamics AX TEST environment** first, and that you are familiar with the impact of your **IDMF archiving, restoring, or purging actions** before you perform them in **PROD**.