ThoughtSpot

ThoughtSpot in Practice

Version 6.0 December 17, 2019

910 Hermosa Court, Sunnyvale, California 94085

Table of Contents

Introduction to Data Integration

This guide explains how to integrate ThoughtSpot with other data sources for loading data. It also includes information on installing and using the ThoughtSpot clients (ODBC, JDBC, and more).

ThoughtSpot Clients

ThoughtSpot provides certified clients to help you load data easily from your ETL tool or another database. These include ODBC and JDBC drivers.

You can obtain the ThoughtSpot client downloads from the Help Center. Always use the version of the ThoughtSpot clients that corresponds with the version of ThoughtSpot that you are running. When upgrading, make sure to upgrade your clients as well.

> ⚠ **Important:** The ETL tool must add a data transformation step if the source column data type does not exactly match the target's, ThoughtSpot's, column data type. The driver does not do any implicit conversions.

Methods for loading data

There are several ways to load data into ThoughtSpot, depending on your goals and where the data is located. Always consider your requirements for recurring loads when planning how best to bring the data into ThoughtSpot.

Here are the options, with information on where to find the documentation for each method:

Method	Description
ThoughtSpot Loader (tsload)	ThoughtSpot Loader is a command line tool to load CSV files into an existing database schema in ThoughtSpot. This is the fastest way to load extremely large amounts of data, and it can be run in parallel. You can also use this method to script recurring loads. See the ThoughtSpot Administrator Guide for details.

Method	Description
User Data Import	Users can upload a spreadsheet through the web interface with User Data Import. This is useful for giving everyone easy access to loading small amounts of their own data. See the ThoughtSpot Administrator Guide for details.
ODBC	ThoughtSpot provides an ODBC (Open Database Connectivity) driver to enable transferring data from your ETL tool into ThoughtSpot.
JDBC	ThoughtSpot provides a JDBC (Java Database Connectivity) driver to enable transferring data from your ETL tool into ThoughtSpot.
Microsoft SSIS (SQL Server Integration Services)	You can use the ODBC driver to connect to SSIS and import data into ThoughtSpot. Basic instructions are included in this guide.
Connect to Pentaho	You can use the JDBC driver to connect to Pentaho and import data into ThoughtSpot. Basic instructions are included in this guide.

Where to go next

- **Server-side prerequisites for using JDBC/ODBC to import data**

  You must follow setup prerequisites for importing data using JDBC/ODBC.

- **About the ODBC Driver**

  You can use the ThoughtSpot ODBC driver to bring data into ThoughtSpot from your ETL tool or database.

- **About the JDBC Driver**

  Java Database Connectivity (JDBC) is a Java standard API that allows applications to interact with databases in a standard manner. ThoughtSpot has JDBC support through a JDBC driver that we provide.

Embrace overview

Summary: Using Embrace, you can perform live query on external databases.

If your company stores source data externally in data warehouses, you can use ThoughtSpot Embrace to directly query that data and use ThoughtSpot's analysis and visualization features, without moving the data into ThoughtSpot. If you decide later you want to copy your data into ThoughtSpot, you can also do that with Embrace.

Embrace supports the following external databases:

- Snowflake
- Amazon Redshift (*in beta*)

To enable Embrace, contact ThoughtSpot support.

How it works

You create a connection to the external database, choosing the columns from each table that you want to explore in your live query. Primary key and foreign key relationships are imported along with the primary and foreign key tables. If there are any joins in the tables of your connection, they are also imported. After your connection is complete, it becomes a **linked** data source in ThoughtSpot that allows you to query the external database directly. It's easy to apply transformations and filter the data also.

Key benefits

- Set up and deploy ThoughtSpot faster by connecting directly to the external database.
- Eliminate the need to move data into ThoughtSpot for analysis.
- Centralize data management and governance in the external database.
- Save significant time and money by avoiding ETL pipelines.
- Set up and schedule sync of data into ThoughtSpot.
- Connect to multiple external databases.

Embrace modes

Embrace has two operating modes:

- **Linked**: ThoughtSpot queries your data in the external database.
- **Synced**: ThoughtSpot queries a copy of your data stored in ThoughtSpot.

When you create your connection to an external database, by default, it is a **Linked** connection. If you want to copy the external data into ThoughtSpot, you must sync the data. The features available with Linked and Synced tables are slightly different.

Features in Embrace modes

Feature	Linked Tables	Synced Tables
Simple Search	Yes	Yes
Complex searches like Versus, Inline Subquerying, Growth	Yes	Yes
Search Suggestions for column names	Yes	Yes
Search Suggestions for column values	Yes	Yes
Headlines at the bottom that summarize tables	Yes	Yes
All Chart Types & Configurations	Yes	Yes
SpotIQ Instant Insights	No	Yes
SpotIQ pre-computed insights	No	Yes
Table and Column Remapping	Yes	N/A
Custom Calendar	No	Yes
Materialized Views	No	Yes
Indexing of table columns	Yes	Yes

Next steps

- **Add a connection**

Create the connection between ThoughtSpot and tables in an external database.

- **Sync** Set your connection to copy tables from the external database into ThoughtSpot.
- **Modify a connection**

  Edit, remap or delete a connection to tables in an external database.
- **Connectors reference**

  Source cloud data connectors, and their connection credentials, supported by Embrace.

JDBC and ODBC setup prerequisites

Before you can use JDBC or ODBC to import data into ThoughtSpot, you must do the following server-side configuration:

1. Open up the ThoughtSpot firewall to allow incoming requests to Simba server.

   ```
   tscli firewall open-ports --ports 12345
   ```

2. Confirm that the `simba_server` process is up. Output of the command below should contain exactly one line, as shown below.

   ```
   ps -ef | grep simba_server | grep -v grep
   admin    26679 25672  0 Jul13 ?        00:01:49 simba_se
   rver_main --logbufsecs=0
   ```

   For assistance, contact ThoughtSpot Support.

Overview of the ODBC Driver

Summary: Use the ODBC driver to bring data in from your ETL tool or database.

ThoughtSpot comes packaged with an ODBC (Open Database Connectivity) driver, so that you can transfer data between ThoughtSpot and other databases. Basic knowledge of ODBC data source administration is helpful when setting up ODBC.

Supported operating systems for the ODBC driver are:

- Microsoft Windows 32-bit
- Microsoft Windows 64-bit
- Linux 32-bit
- Linux 64-bit

Version compatibility and connection parameters

To ensure compatibility, always use the ODBC driver with the same version number as the ThoughtSpot instance to which you are connecting. You can make a secure ODBC connection to the ThoughtSpot database by configuring a user and password combination with the driver. For detailed information about connection parameters, see the ODBC and JDBC configuration properties

Supported Data Types

The ODBC driver supports these data types:

- `INT`
- `BIGINT`
- `BOOLEAN`
- `DOUBLE`
- `FLOAT`
- `DATE`
- `TIME`
- `TIMESTAMP`
- `DATETIME`

- `CHAR`
- `VARCHAR`

Source and target data compatibility

By default, ThoughtSpot takes a permissive approach to data type compatibility between source and target data in ODBC. In this mode, ThoughtSpot *assumes* that the incoming data matches exactly with the target data types and loads the table as is.

Alternatively, you can explicitly require that ThoughtSpot match the source data types exactly and, if it can't find a match, it returns an error and the data load fails. In this mode, for example, if the target ThoughtSpot data type for a column is INT, the source data type for that column must be INT in order for the data load to succeed.

By toggling ***strict*** and ***permissive*** `true` and `false` options, you can configure settings along a scale of behavior between the permissive, automatic approach and the strictness of the "must match" approach.

		Strictness	
		true	false
Permissiveness	true	Data types are inferred and automatically converted. ThoughtSpot returns an error in cases where the data conversion is not possible. Data load fails in its entirety if any data contains mismatches. You must correct the problem in the source data and try the load again.	Data types are inferred and automatically converted. No error is thrown even if source and target data types don't match. Data load continues even when the source and target data types don't match. This means your data load may contain data types that you do not intend or that are not helpful. You are responsible for checking and validating the data in this case.

false	The source and target data types must match. If any data contains mismatches, ThoughtSpot returns an error to the client a data load fails in its entirety. You must correct the problem in the source data and try the load again. This is the strictest configuration.	No data types are inferred and conversion does not check for matches. This is the most permissive configuration.

Your customer support engineer can assist you in configuring custom ODBC behavior. Regardless of the configuration you choose, you must validate that the results of data loading *as they appear* in ThoughtSpot are what you require.

Data type conversion matrix

The following table describes the conversion matrix between SQL data types and ThoughtSpot data types.

Source SQL Data Types	BOOL	INT32	INT64	DOUBLE	FLOAT	CHAR	DATE	TIME	DATETIME
SQL_BIT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_TINYINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_SMALLINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_INTEGER	Y	Y	Y	Y	Y	Y	–	–	–
SQL_BIGINT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_CHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_LONGVARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQL_BINARY	–	–	–	–	–	Y	–	–	–
SQL_VARBINARY	–	–	–	–	–	Y	–	–	–
SQL_LONGVARBINARY	–	–	–	–	–	Y	–	–	–

Source SQL Data Types	BOOL	INT32	INT64	DOUBLE	FLOAT	CHAR	DATE	TIME	DATETIME
SQL_DOUBLE	Y	Y	Y	Y	Y	Y	–	–	–
SQL_REAL	Y	Y	Y	Y	Y	Y	–	–	–
SQL_FLOAT	Y	Y	Y	Y	Y	Y	–	–	–
SQL_NUMERIC	Y	Y	Y	Y	Y	Y	–	–	–
SQL_GUID	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_MINUTE_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_HOUR_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_HOUR_TO_MINUTE	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_SECOND	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_MINUTE	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_DAY_TO_HOUR	–	–	–	–	–	Y	–	–	–
SQL_INTERVAL_YEAR	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_MONTH	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_DAY	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_HOUR	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_MINUTE	–	Y	Y	–	–	Y	–	–	–
SQL_INTERVAL_SECOND	–	Y	Y	–	–	Y	–	–	–
SQL_TYPE_TIME	–	–	–	–	–	Y	–	Y	Y
SQL_TYPE_DATE	–	–	–	–	–	Y	Y	–	Y
SQL_TYPE_TIMESTAMP	–	–	–	–	–	Y	Y	Y	Y

If a conversion is not possible, an error is returned to the client to indicate conversion failure. The ETL tool must add a data transformation step if the source column data type does not exactly match the target's ThoughtSpot column data type. The driver does not do any implicit conversions.

Install the ODBC driver on Windows

Summary: Use this procedure to obtain the Microsoft Windows ODBC driver and install it.

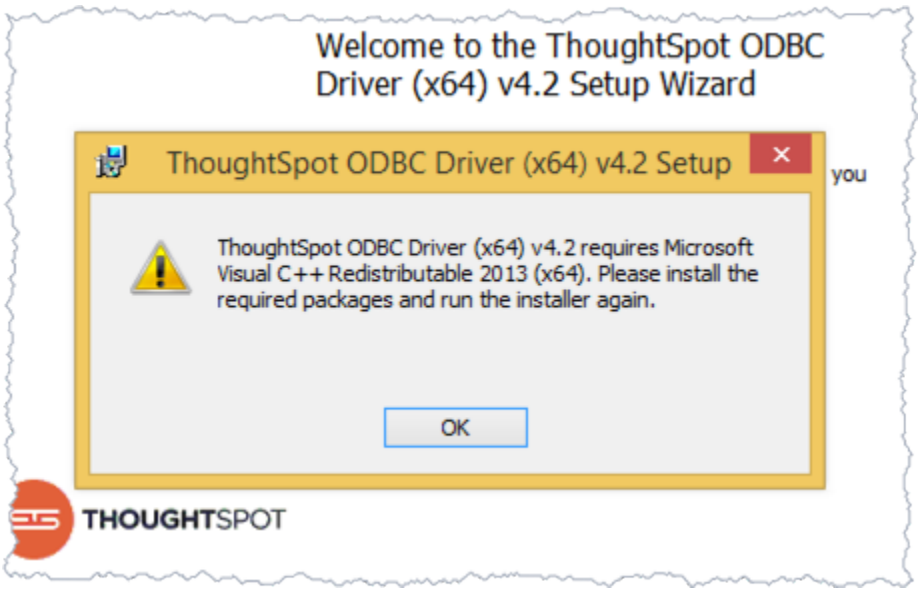
ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the Simba ODBC driver on a Windows workstation.

Make sure you have read the overview material in the ODBC driver overview. This workstation is the same machine where you plan to run your ETL activities.

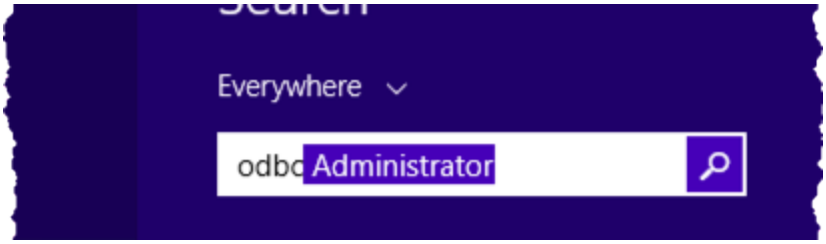
Prerequisites

These instructions include directions to use the `ssh` command. Make sure your Windows workstation is equipped with a tool such as Putty for making `ssh` connections to your ThoughtSpot server.

The ODBC driver for Windows requires Visual C++ Redistributable for Visual Studio 2013. You are prompted to install it during installation of the driver if it isn't already installed.



To check if this Microsoft tool is already installed, search for it on your workstation.



If it isn't installed, make sure you download and install it before continuing.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

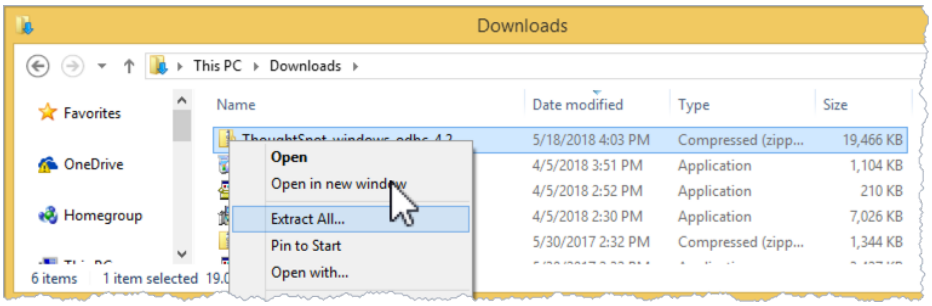
```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Download the driver

On the workstation where you want to connect from, do the following:

1. Navigate to the **Downloads** page.
2. Download the **ODBC Driver for Windows**.
3. Unzip the file you downloaded at a convenient location on your workstation.



4. Take a moment to examine the contents of the new directory.

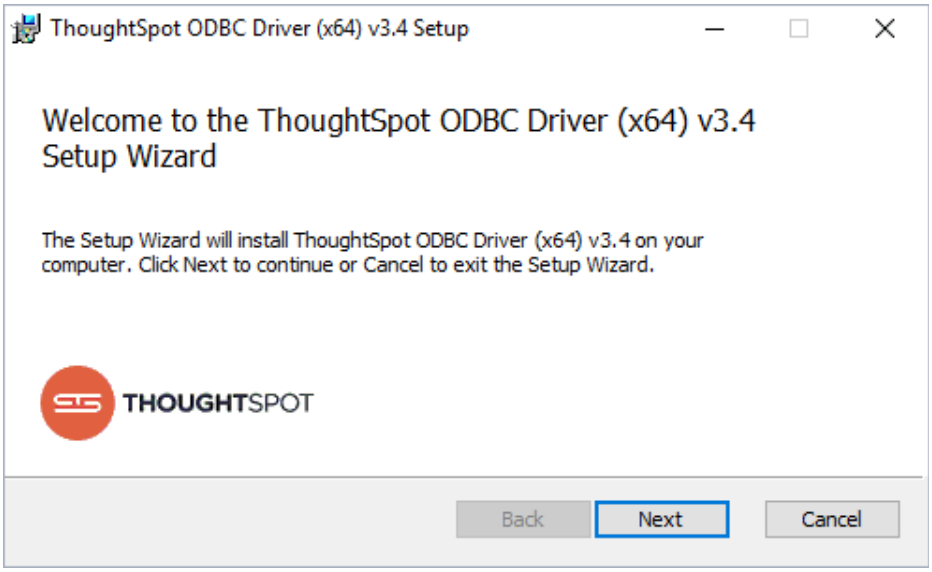
   There are two different Windows ODBC installers included in the file you downloaded.

   - ThoughtSpotODBC (x86).msi for Windows 32-bit
   - ThoughtSpotODBC (x64).msi for Windows 64-bit

Install the driver and supporting software

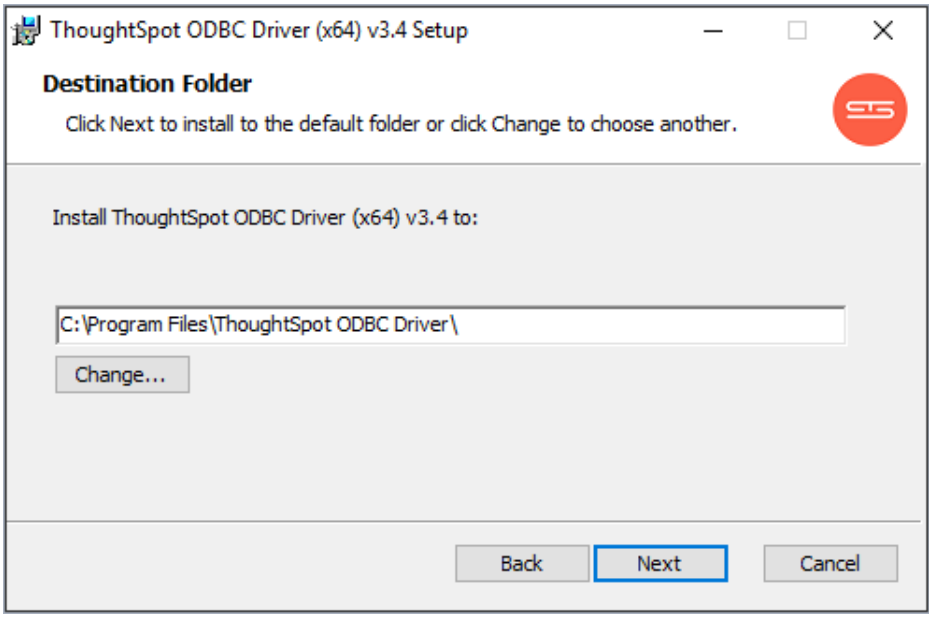
The installation process installs the Simba driver and adds the ODBC Administrator software to your workstation. You use this software to configure the driver.

1. Launch the installer for your version of Windows.
2. Click **Next** to continue.

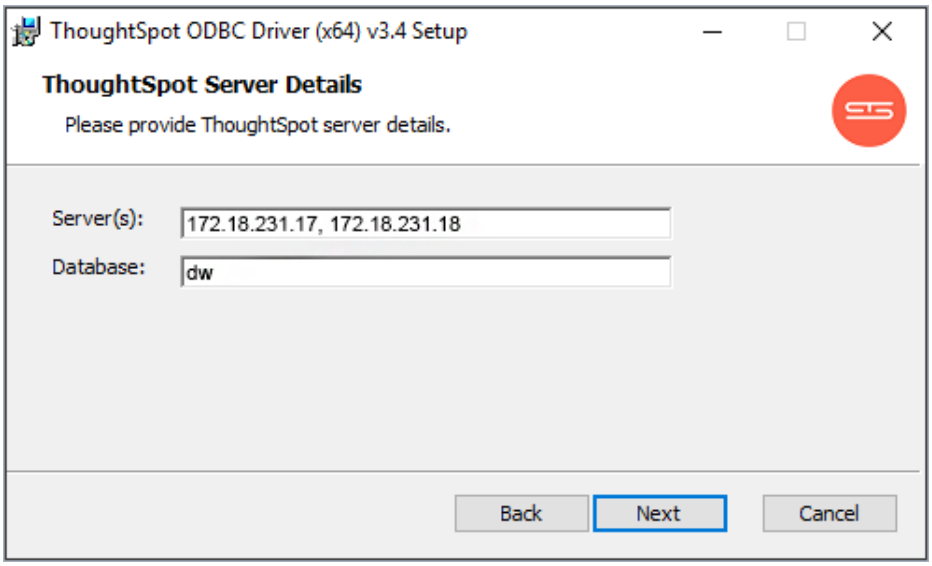


3. Accept the End User License Agreement (EULA), and click **Next**.

4. Specify the destination folder where the driver will be installed.



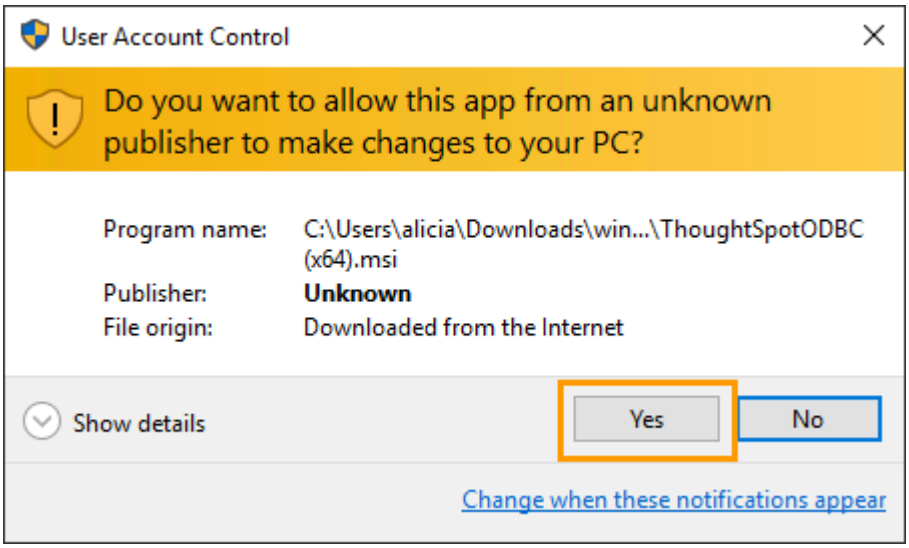
5. Enter the ThoughtSpot server details, and click **Next**.



- For **Server(s)**, provide a comma separated list of the IP addresses of each node on the ThoughtSpot instance.
- For **Database**, optionally specify the database to use. If you skip this entry, you must provide the database each time you connect using ODBC.

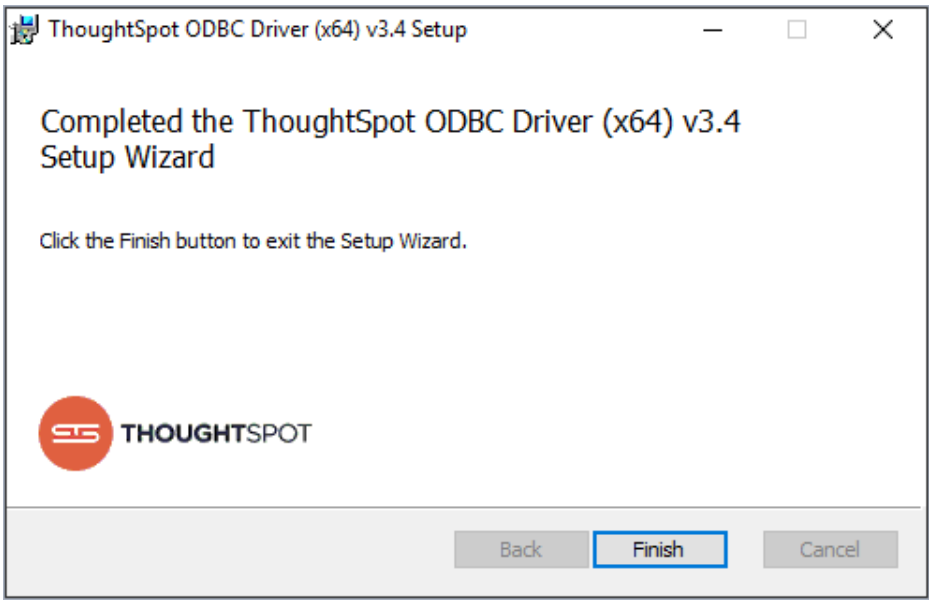
6. Confirm that the install can begin by clicking **Install**.

7. You may see a security warning.



8. Click **Yes** to continue.

   A confirmation message appears when the installation is complete.



9. Click **Finish**.

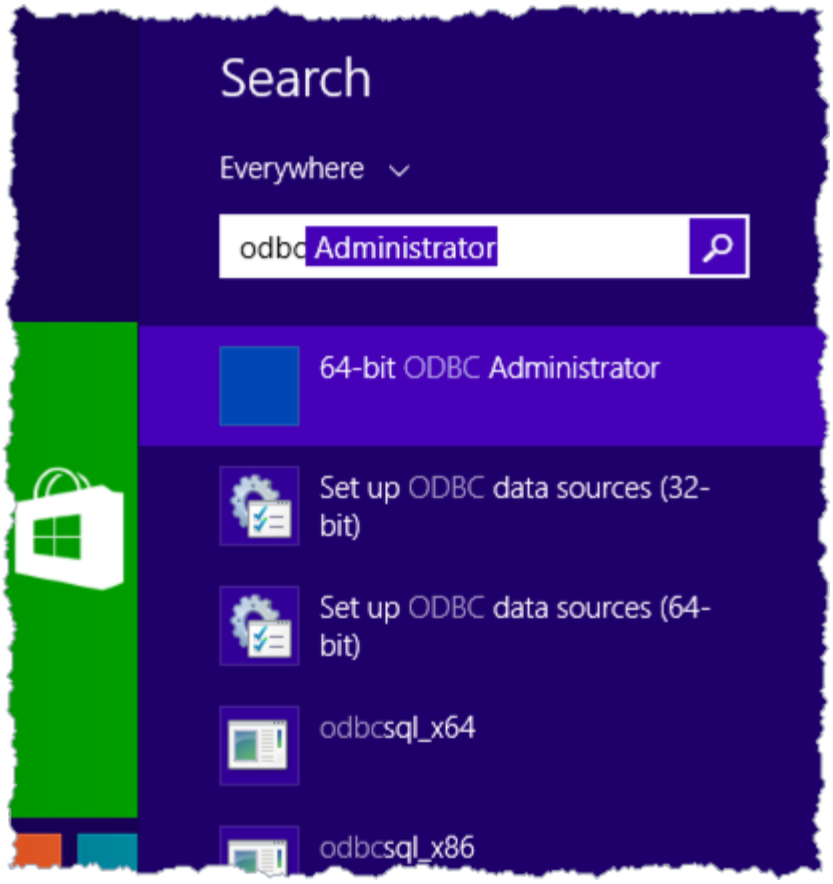
Configure the driver and test your connection

After installation completes, use the ODBC Administrator to configure the ODBC connection on your Windows workstation. For example, you may want to add a default schema or change the server IP address or the default database.

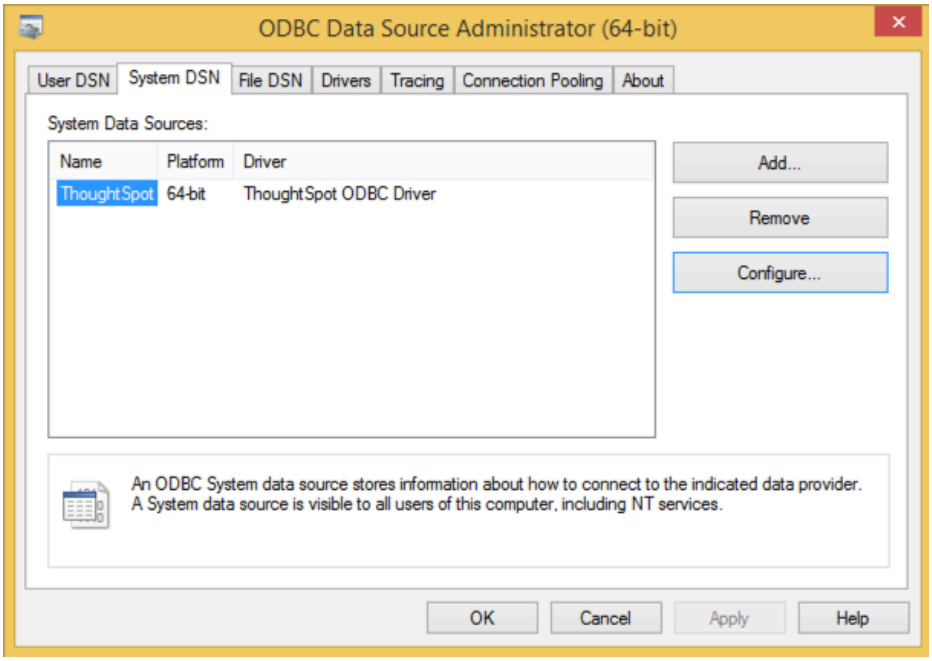
It is recommended to add a default schema. If you don't specify a default schema, you must supply it every time you use the ODBC driver.

At this point, you can test your ODBC connection to ThoughtSpot. It is important to recall that the username/password you use belongs to a ThoughtSpot application user. Typically, this user is a user with data management or administrative privileges on the application.

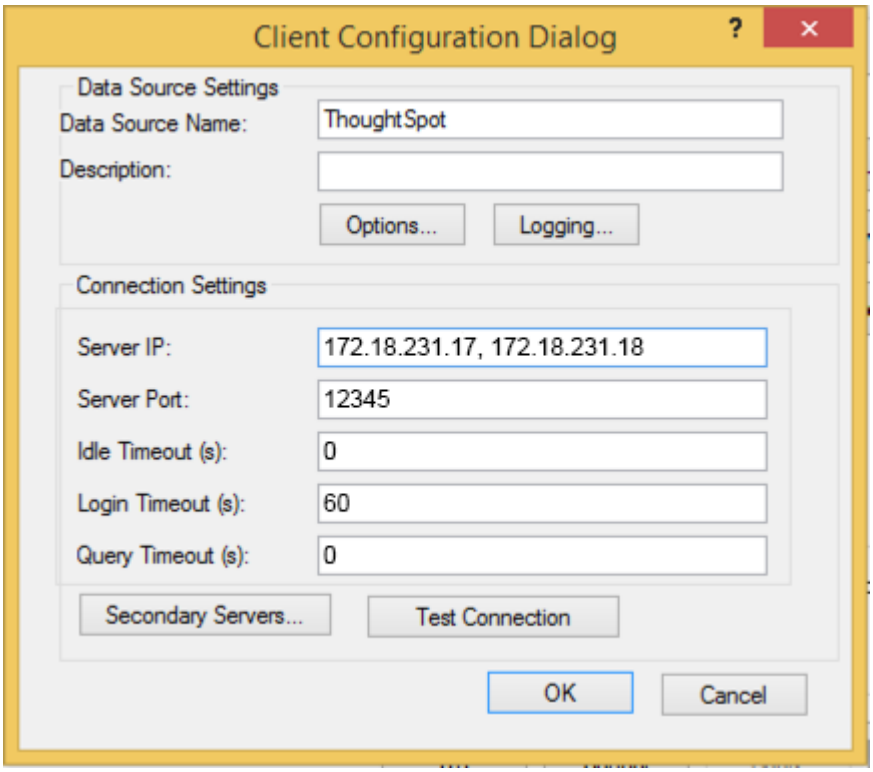
1. Before trying the ODBC connection, confirm a username/password that can log into the ThoughtSpot applications.
2. Click the **Data** tab, and confirm the user's privileges.
3. Return to your workstation.
4. Locate and open the **ODBC Data Source Administrator (64-bit)** application.



5. Click the **System DSN** tab.



6. Select **ThoughtSpot** and click **Configure…**



7. Click **Options…**

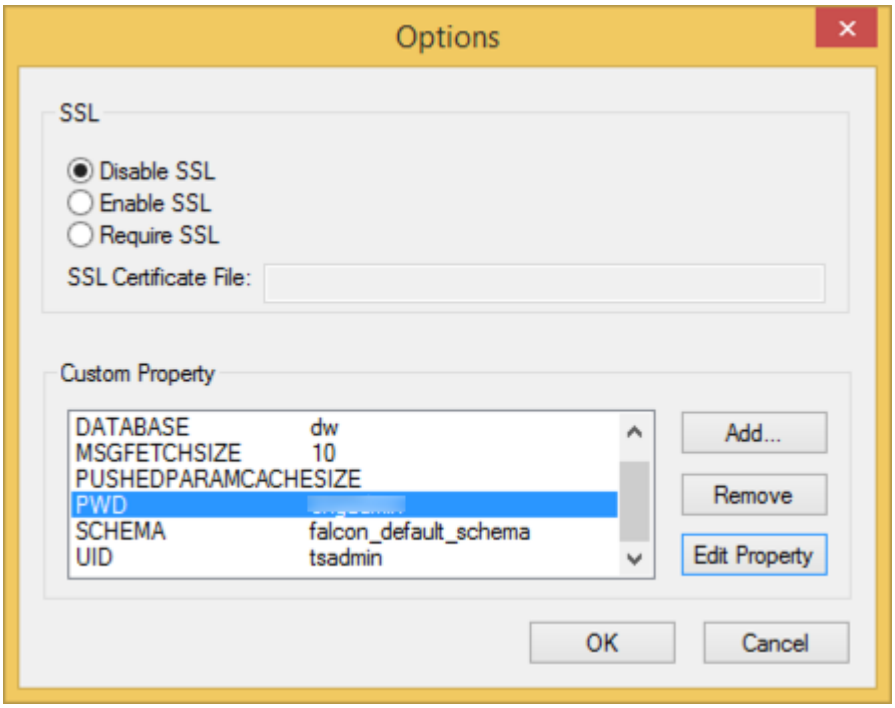
8. Ensure you have the following **Custom Property** values set:

Custom Property	Value
`SCHEMA`	`falcon_default_schema` is the default
`UID`	The username of a user with data management privilege.
`PWD`	The password for the username you specify.

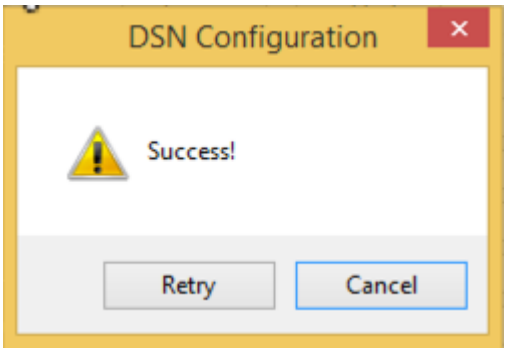
You don't have to use the ThoughtSpot default schema. You can specify your own. We recommend that you define a default schema. Otherwise, you must supply a schema every time you use the ODBC driver. Moreover, without a schema (or if the schema is not present), the ODBC driver returns an error that states that the schema could not be found.

Similarly, adding the `UID` and `PWD` properties are not required. If you don't add them, you are prompted to supply them each time you connect.

When you are done, your options should look similar to the following:



9. When you are done, click **OK** to save your new properties.
10. Click **Test Connection** to test your database connection.



11. Click **Cancel** to close the **DSN Configuration** dialog.
12. Click **OK** to close the **Client Configuration Dialog** the dialog.
13. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application.

Now, you are ready to begin using the connection you've configured.

Related information

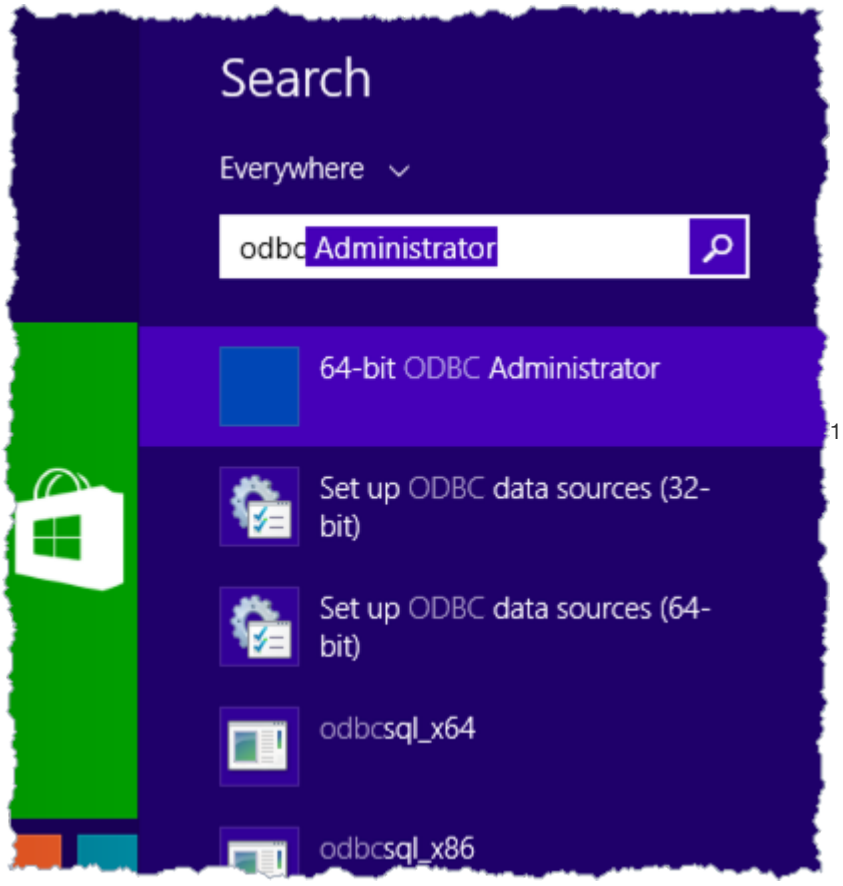
- Enable ODBC logs.
- Configure multiple connections on Windows.

Configure multiple connections on Windows

Summary: You can add multiple ODBC data sources.

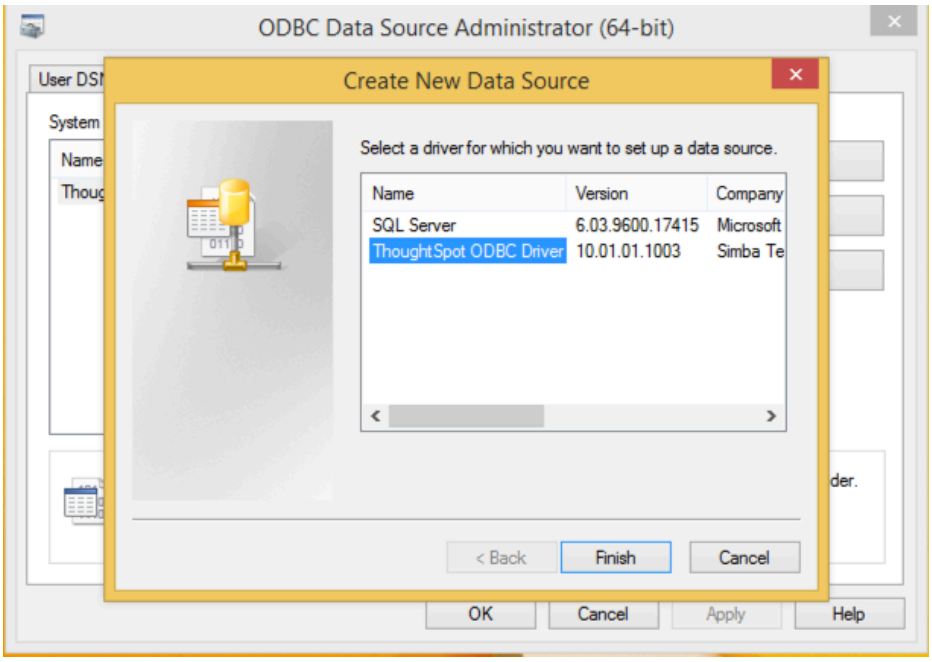
Use this procedure if you want to add an additional data source after creating a single source succeeds. The main reason for needing to set up multiple ThoughtSpot ODBC data sources is that you have a production cluster and a test or development cluster.

1. Locate and open the **ODBC Data Source Administrator (64-bit)** application.



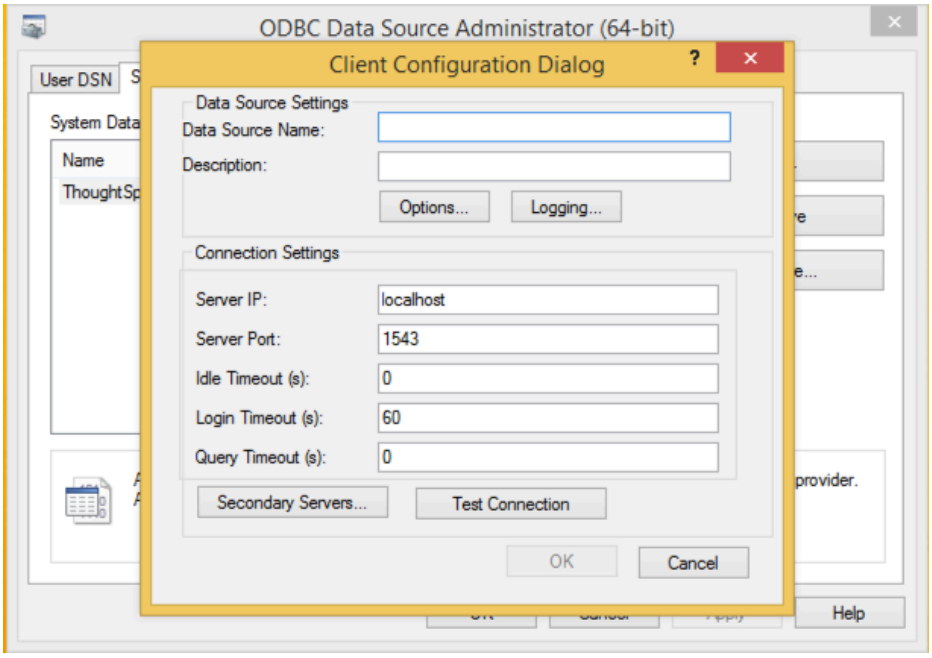
2. Click the **System DSN** tab.
3. Select **Add**.

   The system lists the available drivers.



4. Choose the **ThoughtSpot ODBC Driver** and click **Finish**.

   The system displays the **Client Configuration Dialog** dialog.

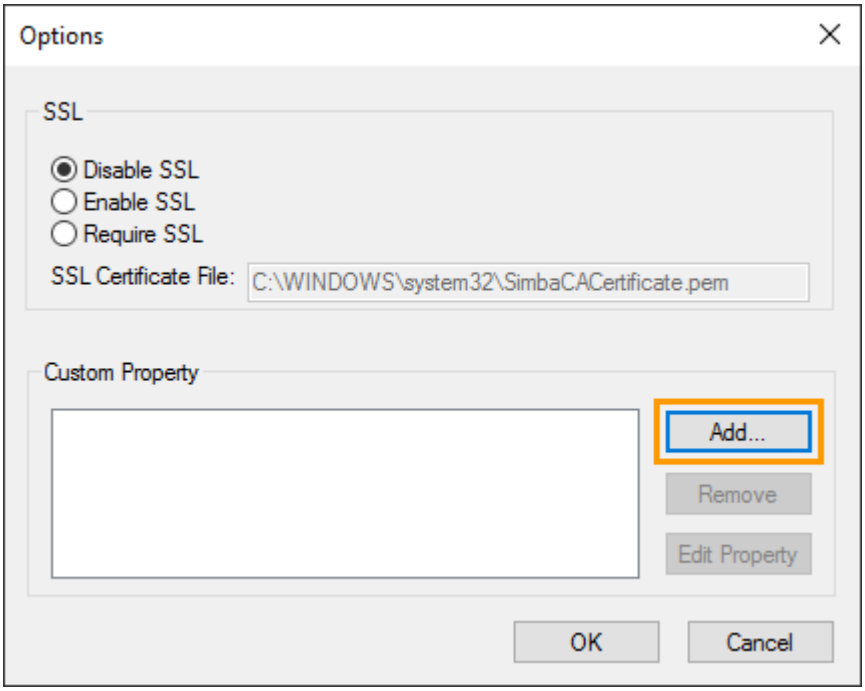


5. Enter the your data source configuration.

Configuration Property	Value
Data Source Name	The name you want to call the data source.
Description	A description of the data source.
Server IP	A list of the IP addresses for each node, separated by commas.
Server Port	12345
Idle Timeout	Time in seconds after which an idle ODBC connection times out.
Login Timeout	Time in seconds after which a login request times out.
Query Timeout	Time in seconds after which a query times out.

6. Configure custom properties by clicking **Options**.

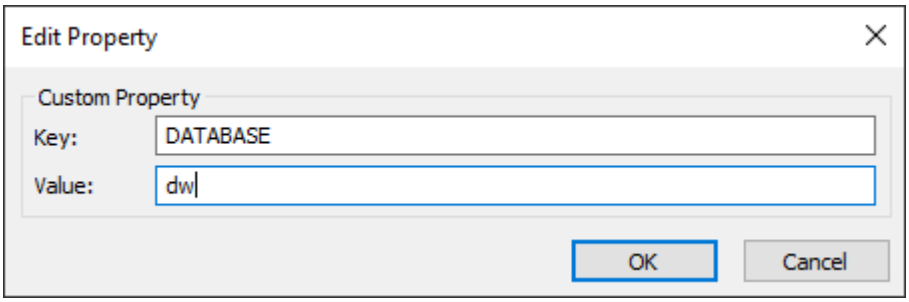
   The system displays the **Options** dialog.



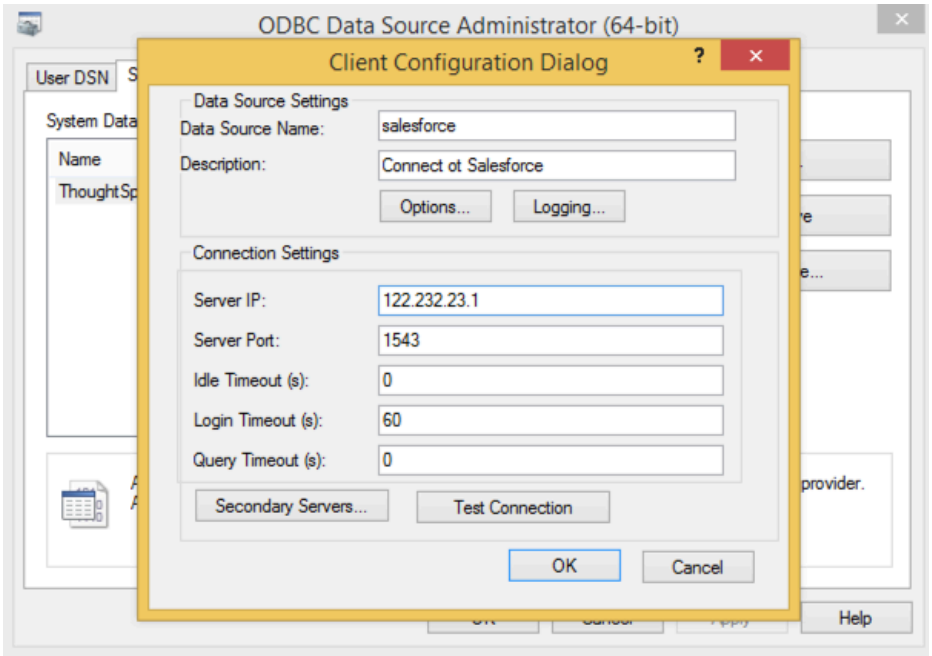
7. Add these properties using the **Add** to enter an option and click **OK** after to save an option.

Option	Value
DATABASE	The default database to connect to.
SCHEMA	The default schema to connect to. Use `falcon_default_schema` if you aren't sure.
CONNECTIONTIMEOUT	Optional. Seconds before an idle connection times out.

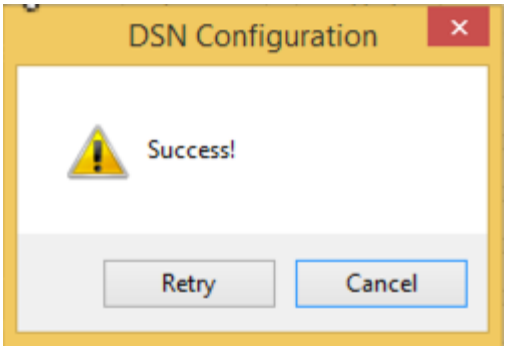
The key must be defined exactly as it appear here, using all capital letters. You can find other supported properties in ODBC and JDBC configuration properties.



8. When you are done, click **OK** to save your new configuration.



9. Click **Test Connection** to test your database connection.



If your test connection fails, enable ODBC logging to troubleshoot.

10. Click **Cancel** to close the **DSN Configuration** dialog.
11. Click **OK** to close the **Client Configuration Dialog** the dialog.
12. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application

Deploy SSL with ODBC on Windows

You can configure a secure ODBC connection between your ThoughtSpot cluster and a remote Windows Machine. This article explains the SSL resources and ODBC configuration options you need to enable SSL for an ODBC connection.

Prerequisites

Before configuring SSL over the ThoughtSpot ODBC connection, make sure that your system administrator has created and configured your network's Certificate Authority. Additionally, the system administrator should have available both the proper Private Key and Server Certificate.

Configure the ThoughtSpot cluster nodes

> ⚠ **Important:** Portions of this procedure require that you work with your ThoughtSpot Customer Service or Support Engineer.

The SimbaServer Configuration Properties reference includes full details on SSL Configuration Properties.

Before you change your ODBC configuration, decide on a path where you will store the Private Key and Server Certificate, for example, you could decide to use `/home/admin/Simba_SSL/` as the path.

Then, do the following on *every ThoughtSpot node* in your cluster.

1. Create the path on the node.
2. Copy the SSL certificate and private key to this path.
3. Edit the node's `/etc/thoughtspot/simba.ini` file (Simba server configuration) with your favorite editor.
4. Add the following lines:

```
 SslCertfile=/home/admin/Simba_SSL/Server-Certificate.pe
m
 SslKeyfile=/home/admin/Simba_SSL/Private-Key.pem
 UseSsl=Required
```

5. Restart the Simba service.

   You must work with your ThoughtSpot Customer Success or Support Engineer to do this.

Deploy the certificate on your windows workstation

Please note that the SSL settings on the server and client are interdependent.

The SimbaClient for ODBC Configuration Properties reference describes how to set parameters on the client to use SSL (scroll down to useSsl section at the end). The Simba documentation also provides a chart showing configuration properties for SSL where you can see how different combinations of SSL settings on client and server will behave. For example:

- Setting both server and client to `UseSsl=Enabled` provides the ability for clients to connect with or without SSL.
- Setting both server and client to `UseSsl=Required` requires that all clients use SSL.

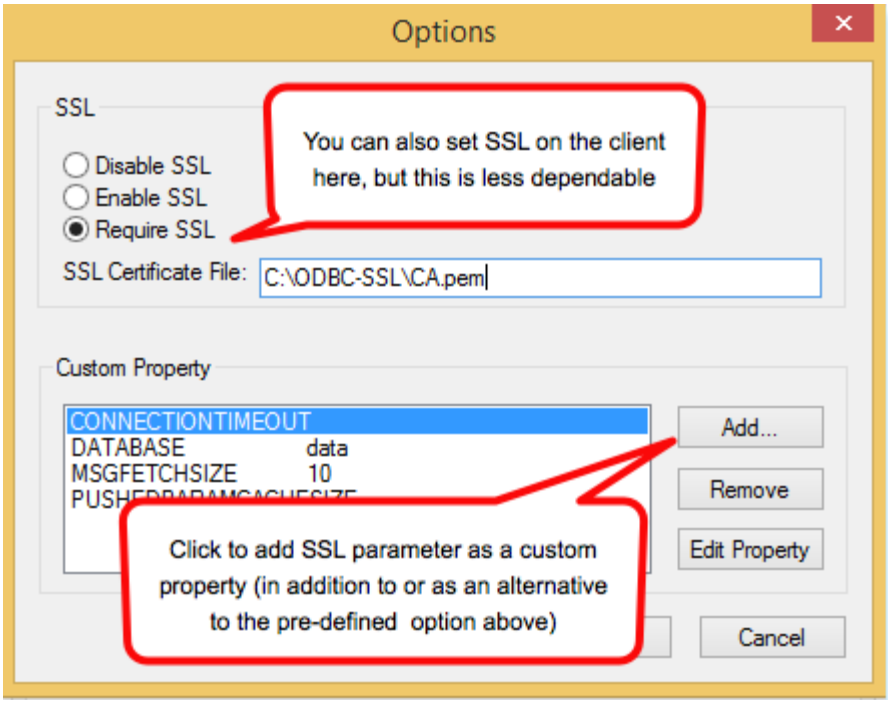
> ℹ **Note:** Note that the SSL and certificate parameters can be set through the pre-defined options on the options dialog, but customers have reported that these are not always reliable. In the following procedure, we recommend using custom properties to define these settings (either preemptively, or as a solution if the ODBC connection over SSL does not work with the pre-defined options). There is no harm in setting both. Example settings are: `UseSSL = Required` and `SslCACertfile = C:\ODBC-SSL\CA.pem`

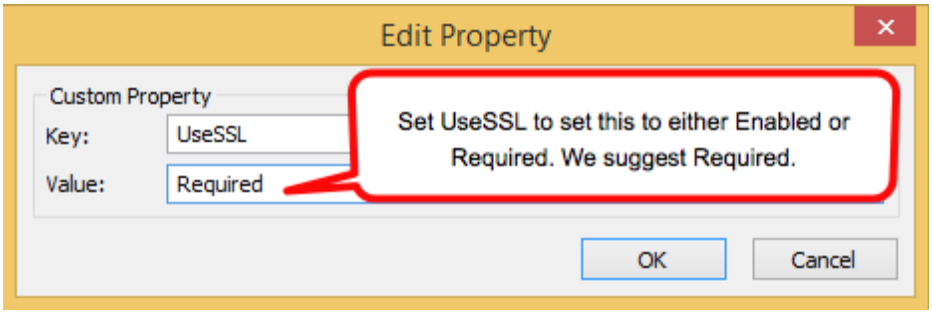
On the workstation you want to use for your ODBC connection, specify the level of SSL you want to use on the client along with the path to the CA certificate, and then test the connection.

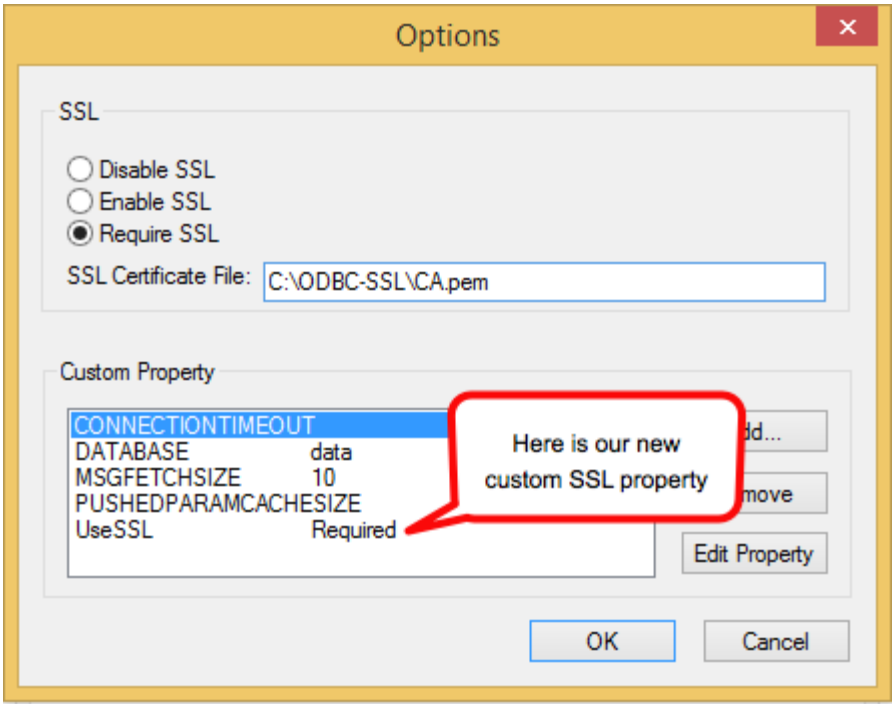
1. Save the CA certificate to a secure location on the workstation disk.

Choose a location where the certificate is unlikely to be deleted by mistake, for example, `C:\ODBC-SSL\CA.pem` is an example of a full path to such a location.

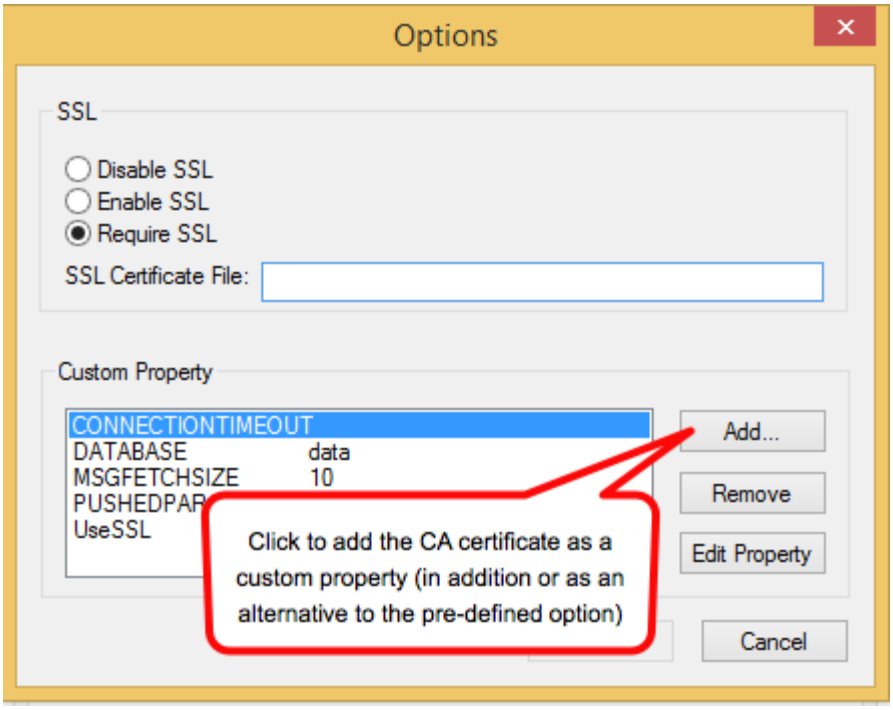
2. Open your ThoughtSpot ODBC connection configuration dialog.
3. Click **Options**.
4. Check the **Require SSL** option and/or add SSL as a custom property.

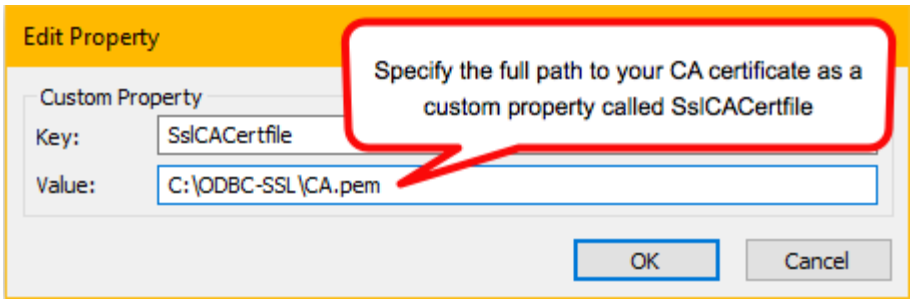

Options
SSL
Disable SSL
Enable SSL
Require SSL
You can also set SSL on the client here, but this is less dependable
SSL Certificate File:
C:\ODBC-SSL\CA.pem
Custom Property
CONNECTIONTIMEOUT
DATABASE data
MSGFETCHSIZE 10
Add...
Remove
Edit Property
Click to add SSL parameter as a custom property (in addition to or as an alternative to the pre-defined option above)
Cancel

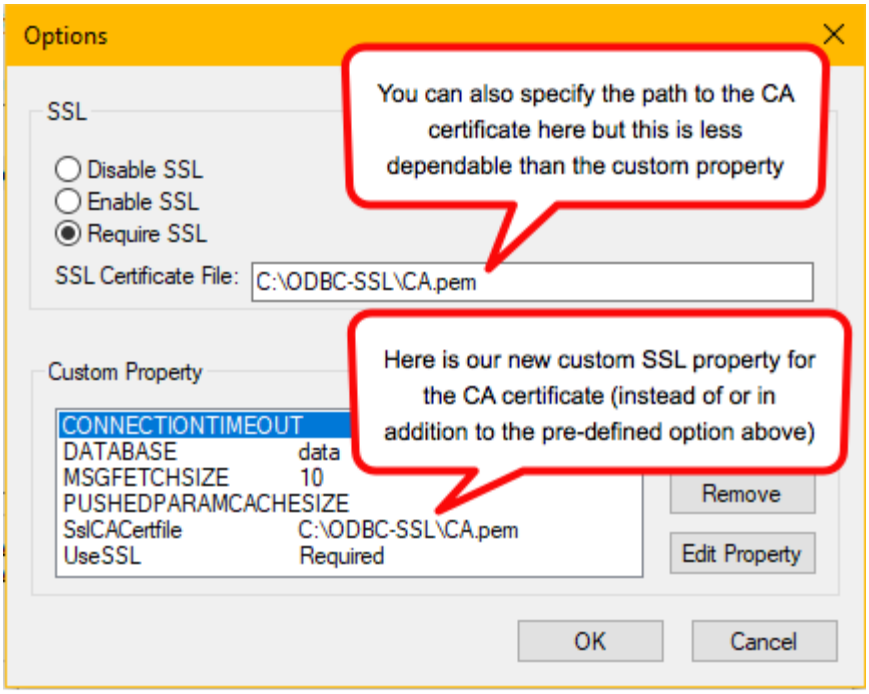

Edit Property
Custom Property
Key:
UseSSL
Value:
Required
Set UseSSL to set this to either Enabled or Required. We suggest Required.
OK
Cancel



5. Enter the location of the CA certificate in the **SSL Certificate File** field and/or add the CA certificate as a custom property. Be sure to provide the full path to the certificate (`{certificate_directory}\{CA_certificate}.pem`).

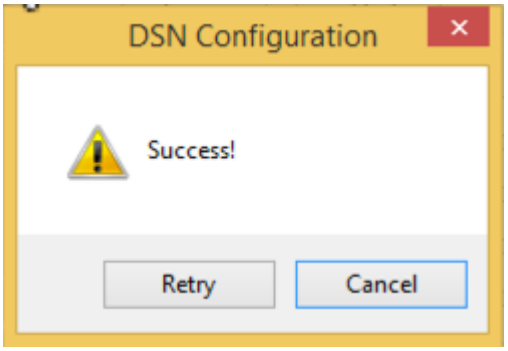

Options
SSL
Disable SSL
Enable SSL
Require SSL
SSL Certificate File:
Custom Property
CONNECTIONTIMEOUT
DATABASE data
MSGFETCHSIZE 10
PUSHEDPAR
UseSSL
Add...
Remove
Edit Property
Cancel
Click to add the CA certificate as a custom property (in addition or as an alternative to the pre-defined option)


Edit Property
Custom Property
Key: SslCACertfile
Value: C:\ODBC-SSL\CA.pem
OK
Cancel
Specify the full path to your CA certificate as a custom property called SslCACertfile



6. When you are done, click **OK** to save your new properties.

7. Click **Test Connection** to test your database connection.



8. Click **Cancel** to close the configuration dialog.
9. Click **OK** to close the **Client Configuration Dialog** the dialog.
10. Click **OK** to close the **ODBC Data Source Administrator (64-bit)** application.

Set up the ODBC Driver for SSIS

Summary: Use SSIS to set up the ODBC Driver.

Microsoft SSIS (SQL Server Integration Services) is a data integration and workflow applications platform you can use to connect to ThoughtSpot. The platform is a component of the Microsoft SQL Server database software.

You can use a SSIS connection to perform data migration tasks. Its data warehousing tool is useful for data ETL (extraction, transformation, and loading). The SSIS Import/Export Wizard creates packages that transfers data with no transformations. It can move data from a variety of source types to a variety of destination types, including text files and other SQL Server instances.

Use SSIS to set up the ODBC Driver by creating a connection manager. This manager connects an OLE DB Source and the ODBC Destination.

Prerequisites

On Windows 64-bit, you have to install both the 32-bit and 64-bit ThoughtSpot ODBC drivers. In addition, they must be named the same, such as ThoughtSpot. By default they are named ThoughtSpot-32 and ThoughtSpot-64. This is required because the 64-bit SSIS shows a list of 32-bit ODBC drivers when you configure an ODBC target. However, it executes the 64-bit driver. If the drivers aren't named the same, then you can get an error stating the driver doesn't exist.

Set up the driver

To set up the ODBC driver using SSIS:

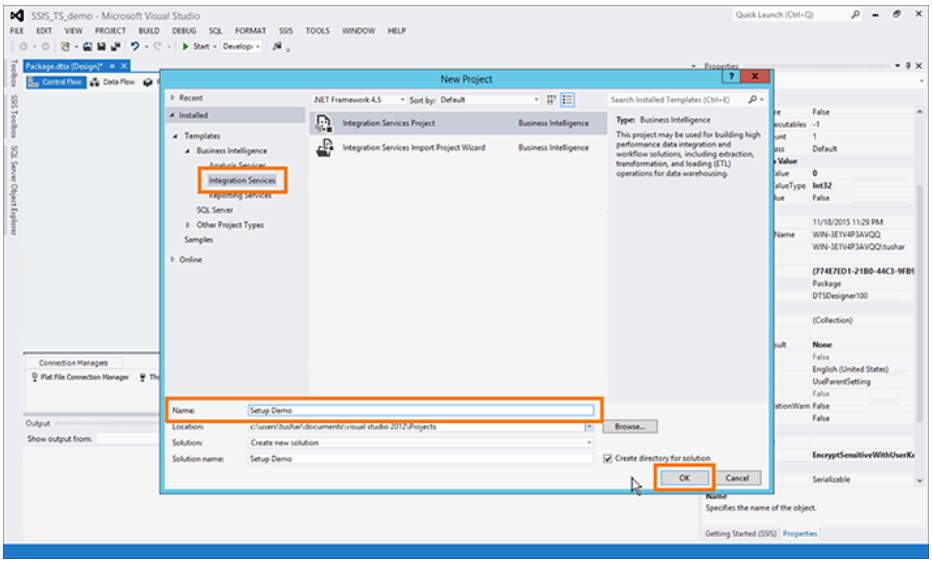
1. Open your SQL Server visual development tool that is based on Microsoft Visual Studio.
2. Select **OLE DB Source**, and click **New**.

   Where ODBC provides access only to relational databases, OLE DB provides access to data regardless of its format or location.
3. Add the server by name from the machine accessible list.

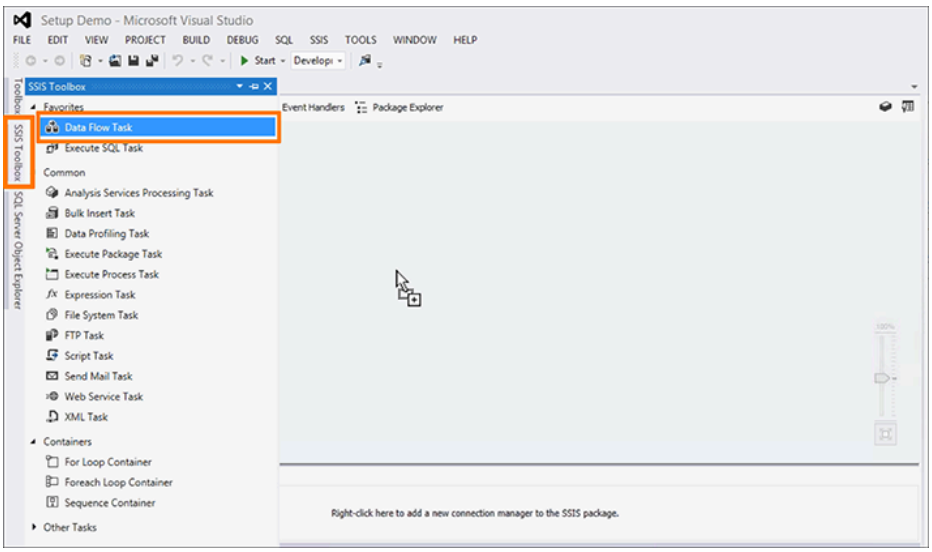
4. Enter the authentication information: db name, user name, password, and test connection.

   You can add the UID and password by clicking on **Options**.

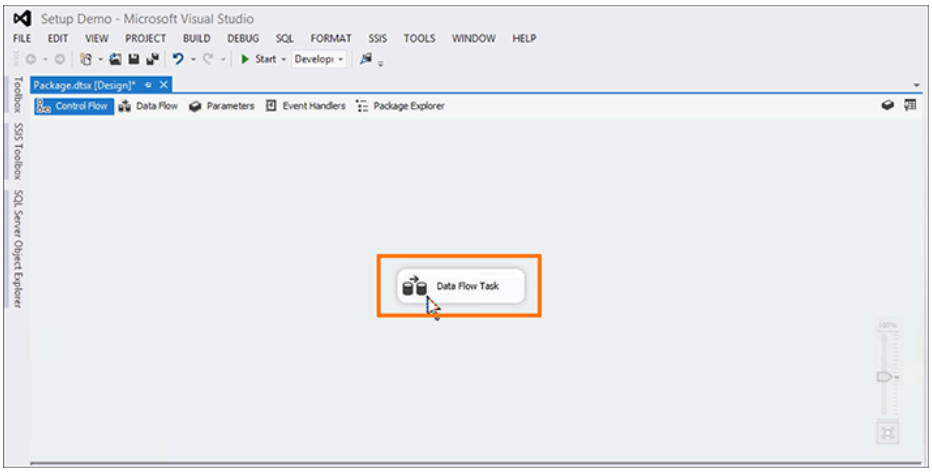
5. Click **File** and select **New**, then **Project**.
6. Select the **Integration Services** tab under **Installed > Templates > Business Intelligence**.
7. Enter a name in the **Name** field and click **OK**.



8. Select the **SSIS Toolbox** tab on the left hand side of the platform, and drag and drop **Data Flow Task** to the main window.



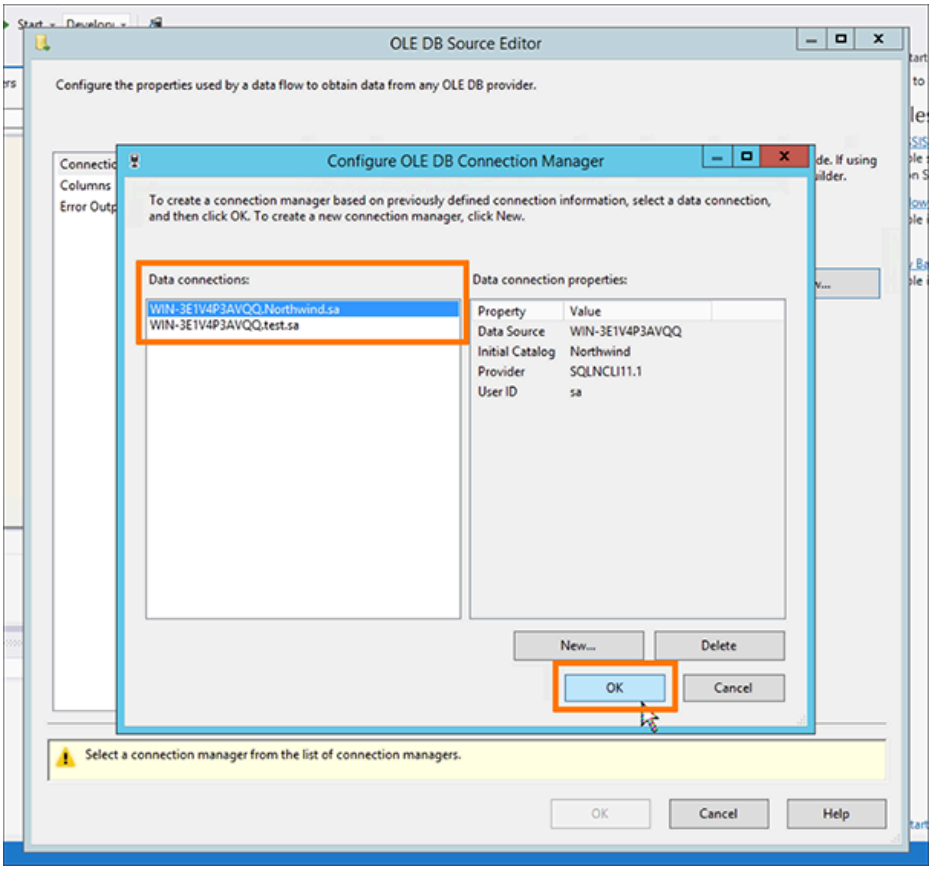
9. Double click the **Data Flow Task** icon when it appears in the center of the page.



10. Navigate back to the **SSIS Toolbox** tab. You now want to create sources and destinations.

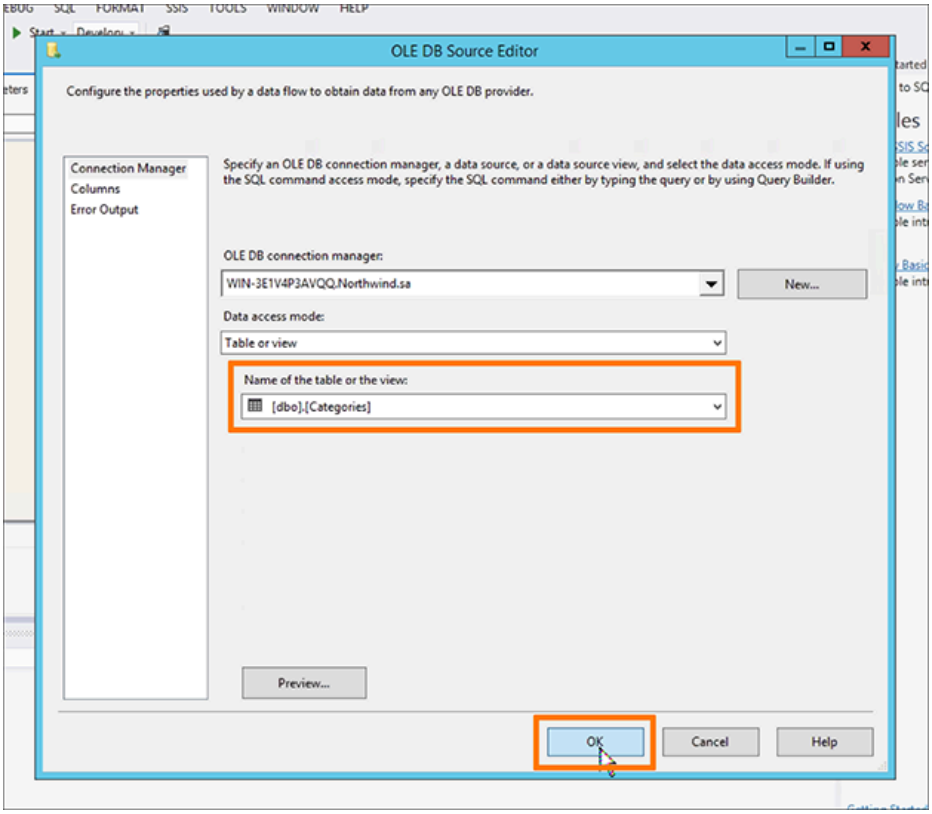
Create sources and destinations

1. Under **Other Sources**, find **OLE DB Source** and drag and drop it to the main window.
2. Double click the **OLE DB Source** icon when it appears in the center of the page to open the OLE DB Source Editor.
3. Select a new OLE DB connection manager by clicking **New**.
4. In the Configure OLE DB Connection Manager window, select your **Data connection** and click **OK**.



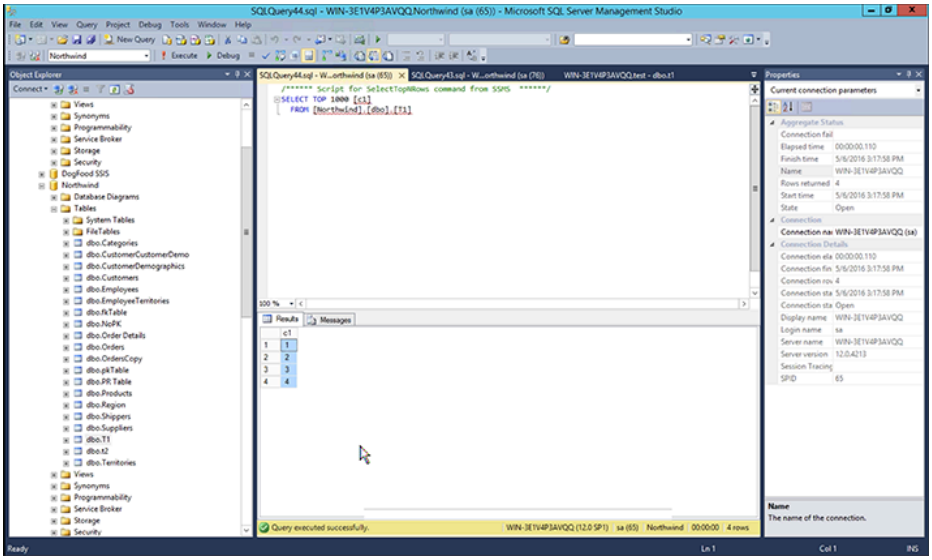
If you do not see your data connection, you will have to create a new one in the Connection Manager by clicking **New**.

5. Back in the OLE DB Source Editor, select the **Name of the table or the view**, and click **OK**.



6. Select the table, and see what columns are in it.

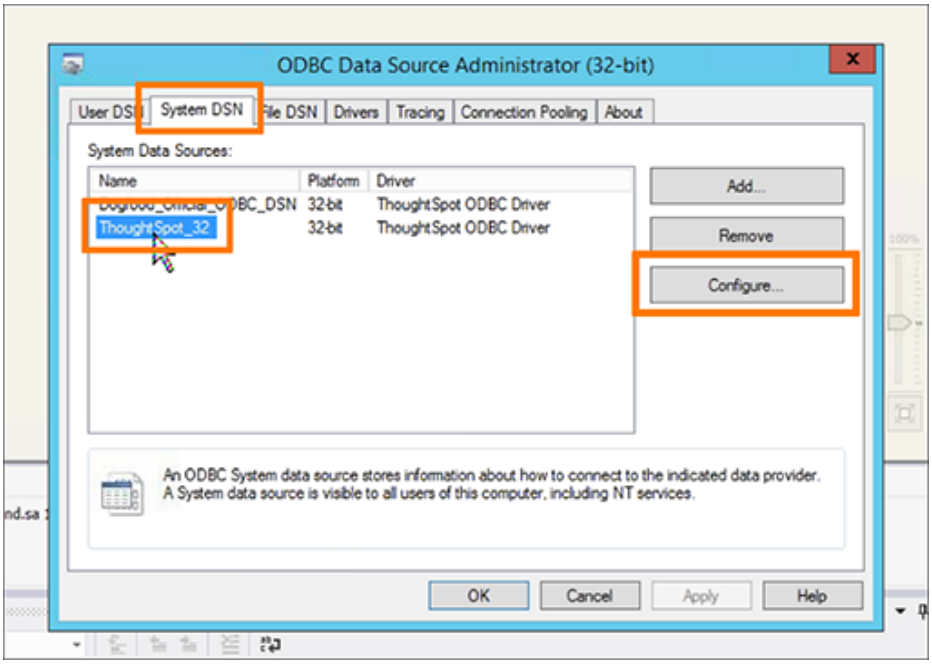
   In this example, a single column, `c1`, is selected.



Configure the ODBC Data Source Administrator

The ODBC Data Source Administrator has to be configured to connect to ThoughtSpot and bring the table in.

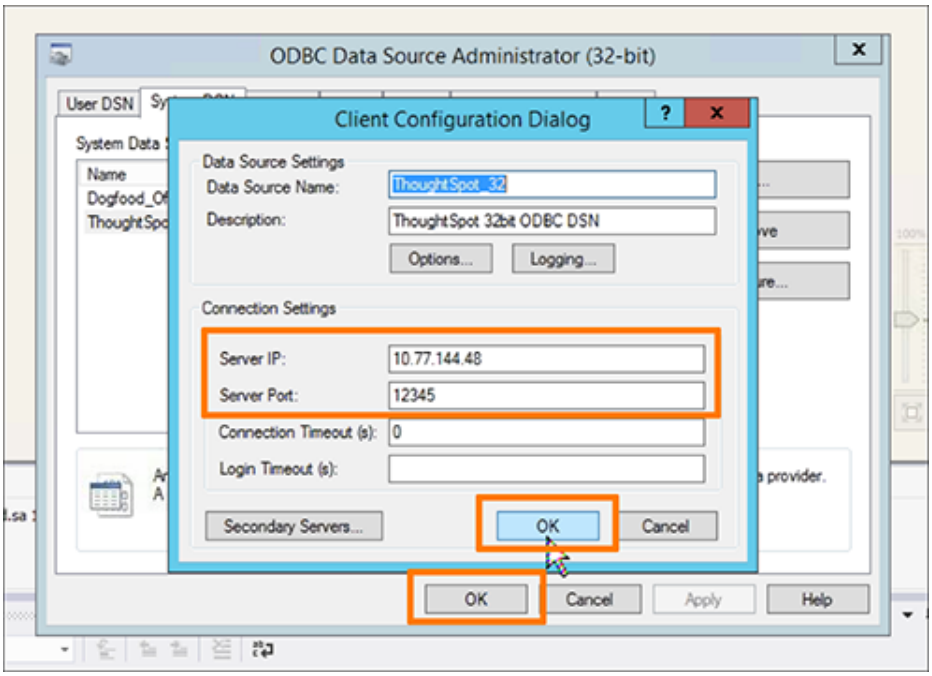
1. Search for and open your **ODBC Data Sources (32-bit)** program.
2. Click the **System DSN** tab and select **ThoughtSpot_32**.
3. Click **Configure**.



4. In the Client Configuration Dialog, enter the **Server IP** and **Server Port**.

   Enter any node IP that has Simba server running on it. In **Secondary Servers**, you must specify all node IPs, because ThoughtSpot must resolve to the server Simba runs on, and that server can change after an upgrade. Enter one server IP per line. The line return serves as a separator. Comma separated values are not supported.

5. Click **OK** twice to close the Client Configuration Dialog and the ODBC Data Source Administrator.

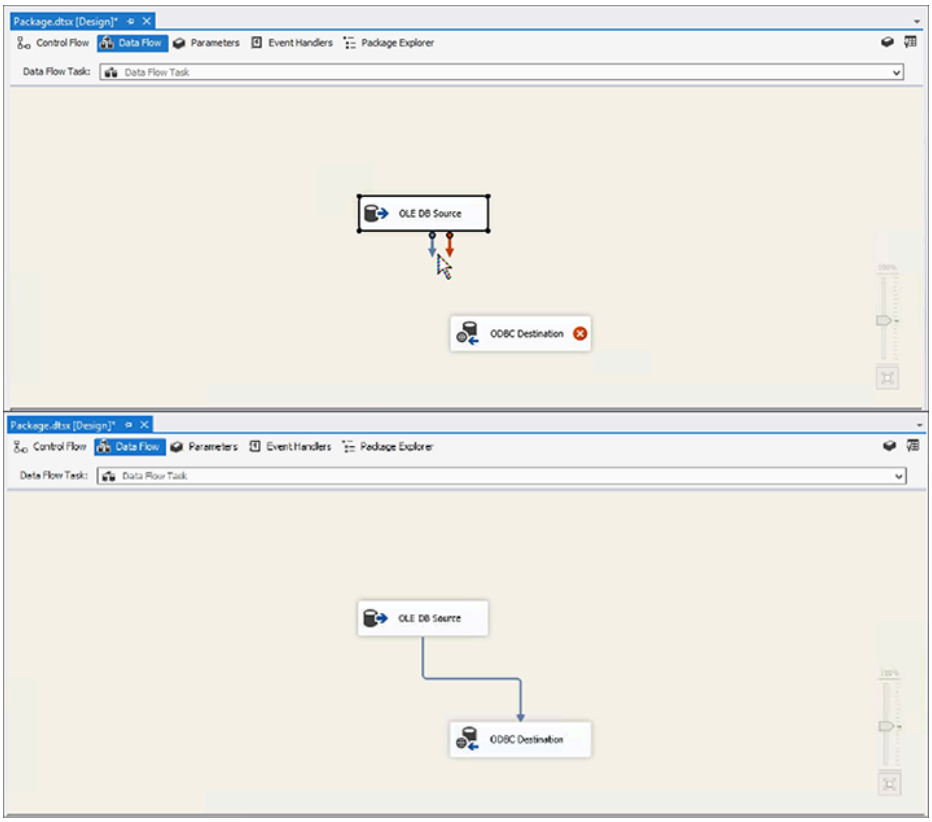


Create a file to take the feed

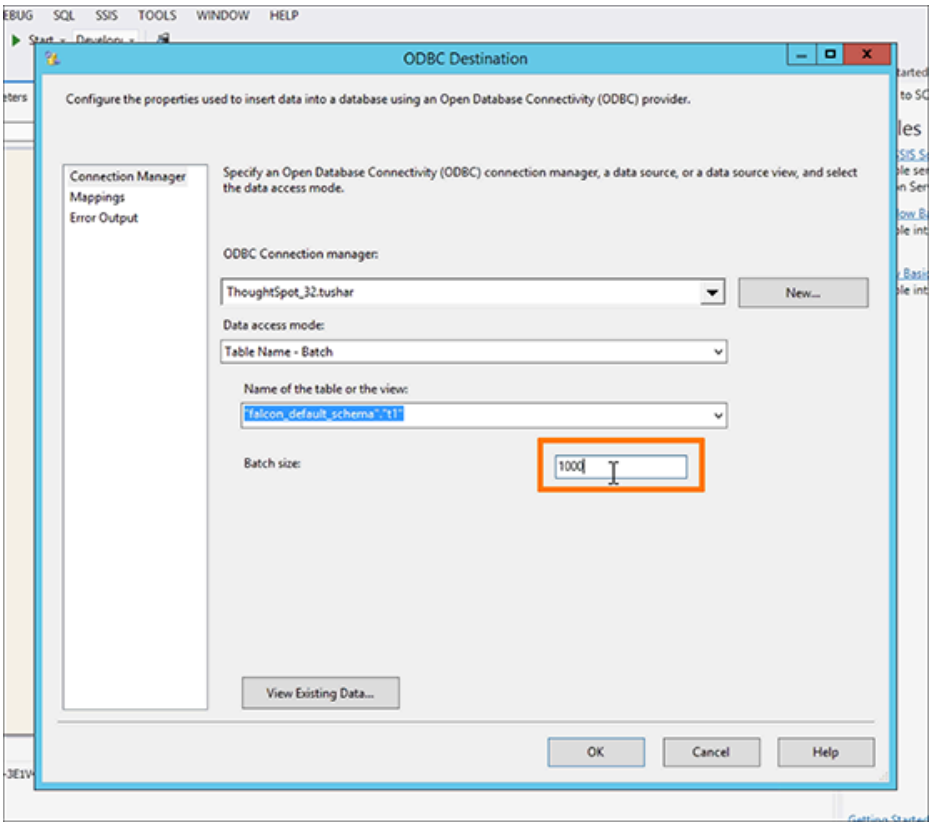
Now that you have set up your source, create the empty table in ThoughtSpot to take this feed. SSIS does not allow you to create the table in ThoughtSpot. You have to do this first in TQL. In Pentaho, it will create the table in ThoughtSpot, but not in SSIS.

Create the ODBC Destination. Use the one you created and named in the ODBC Data Source Administrator.

1. In the **SSIS Toolbox** tab, under **Other Destinations**, drag and drop **ODBC Destination** to the main window.
2. Drag the **blue arrow** to connect the OLE DB Source icon to the ODBC Destination icon.
3. Double click the **ODBC Destination** icon.



4. Use ODBC Destination to set the **Batch size** for the connection in the Connection Manager tab. You can set the size to be up to 10,000.



If the load fails, the entire batch will be lost, and you will have to start that load over again.

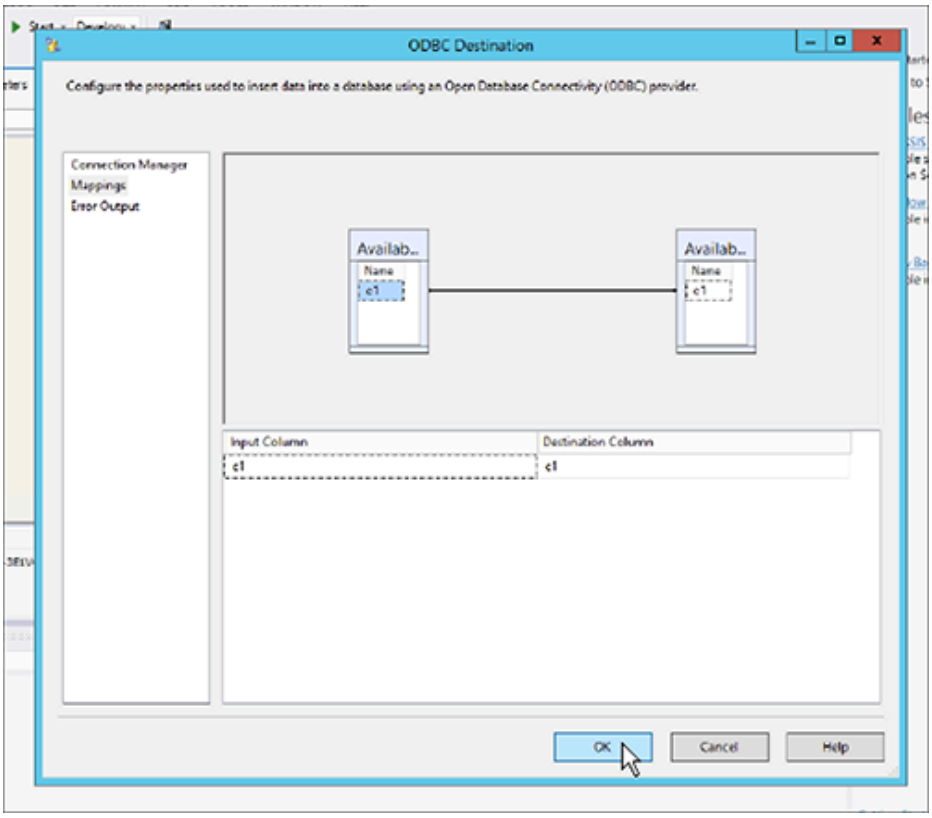
5. Set the **Transaction Size** to match the total number of rows that are expected to be loaded in the load cycle.

   Your transaction size can be quite large—even spanning a million rows. However, too many small batches can leave the cluster in a rough state. This is because each batch acts as a separate transaction and creates a separate commit. Too many of these will slow down our system since each transaction creates a “data version” in our system. In Pentaho, the transaction size setting is called Commit Size.

6. Set the **Transaction Option** attribute of the Data Flow Task to **Supported**.

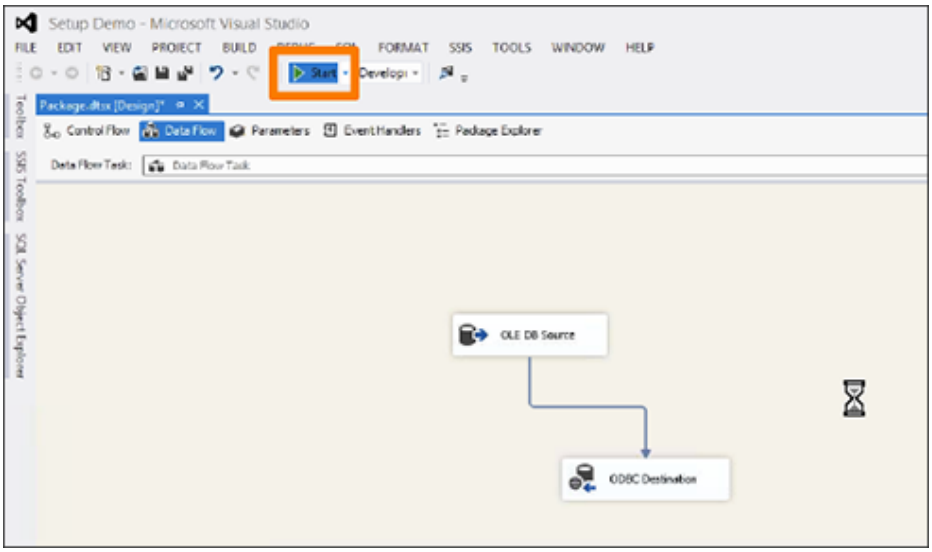
7. In the **Mappings** tab, validate the mapping or change it.

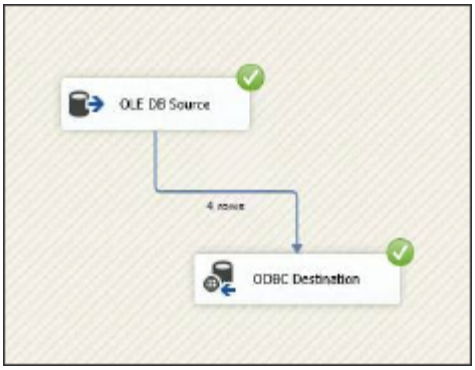
   You can have different column names in each database if you map them. Of course, they must be of the same or compatible datatype.



8. Start the import job by clicking the **Start** button.

   You should see an animation indicating that the data is transferring over. When the import is complete, the number of successfully transferred rows is displayed.





You can validate the import using TQL or from the **Data** screen.

Install the ODBC Driver on Linux

Summary: Use this procedure to obtain the Linux ODBC driver and install it.

ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the Simba ODBC driver on a Linux workstation.

Make sure you have read the overview material in the ODBC driver overview. This workstation is the same machine where you plan to run your ETL activities.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Install the Simba client

On your workstation, where you want to connect from, do the following to get the ODBC driver:

1. Open a browser on your workstation.

2. Navigate to the **Downloads** page.
3. Click **ODBC Driver for Linux** to download the driver.
4. Open a terminal on your workstation.
5. Change directory to the location where you downloaded the file.
6. Optionally, move the file to a permanent location on your machine.

   When you expand the downloaded file it will create a directory in the location.

7. Unzip the zip file:

   ```
   gunzip ThoughtSpot_linux_odbc_<version>.tar.gz
   ```

8. Extract the contents of the `tar` file.

   ```
   tar –xvf ThoughtSpot_linux_odbc_<version>.tar
   ```

   This extracts a subdirectory called `linux` into the current directory.

9. Take a moment to examine the contents of the new directory.

   The structure contains a Simba client library, supporting libraries and setup files for two different architectures. It also continues error messages for multiple languages.

```
├── Bin
│   ├── Linux_x86
│   │   └── libSimbaClient.so
│   └── Linux_x8664
│       └── libSimbaClient.so
├── ErrorMessages
│   ├── de-DE
│   │   ├── ...
│   ├── en-US
│   │   ├── CSCommonMessages.xml
│   │   ├── ClientMessages.xml
│   │   ├── JNIDSIMessages.xml
│   │   ├── ODBCMessages.xml
│   │   ├── SQLEngineMessages.xml
│   │   └── ServerMessages.xml
│   ├── es-ES
│   ├── fr-FR
│   ├── it-IT
│   └── ja-JP
├── Lib
│   ├── Linux_x86
│   │   ├── libcrypto.so
│   │   ├── libcrypto.so.1.0.0
│   │   ├── ...
│   └── Linux_x8664
│       ├── ...
└── Setup
    ├── .simba.javaquickstart.ini
    ├── .simba.javaultralight.ini
    ├── .simba.quickstart.ini
    ├── .simba.ultralight.ini
    ├── .simbaclient.ini
    ├── .simbaserver.javaquickstart.ini
    ├── .simbaserver.javaultralight.ini
    ├── .simbaserver.quickstart.ini
    ├── .simbaserver.ultralight.ini
    ├── odbc.ini
    └── odbcinst.ini

14 directories, 92 files
```

The `linux/Setup` directory contains the key ODBC configuration files and sample Simba client configurations you can use later in this procedure.

10. You must know your workstation architecture to continue, confirm your workstation's architecture.

    You can use the `arch` or the `uname` command or both.

    ```
     $ arch
       x86_64
     $ uname -a
       Linux nebula-docs-production-4vfnv 4.4.108-1.el7.elre
    po.x86_64 #1 SMP Mon Dec 25 09:55:39 EST 2017 x86_64 x8
    6_64 x86_64 GNU/Linux
    ```

    In previous examples, the workstation is a 64 bit workstation. Your workstation may be 32-bit. You can use this architecture information in the procedures that follow.

(Optional) Install unixODBC tools for testing

The procedures on this page rely on the unixODBC tools to test your configuration and connection. If you are experienced with ODBC and want to skip this, you can. Simply substitute your preferred mechanism in the subsequent procedures where references are made to the unixODBC tools.

> ❗ **Warning:** Your ThoughtSpot installation contains a version of the unixODBC tools. These tools are incompatible with CentOS. Do not use these tools if you are performing this procedure on your ThoughtSpot server.

1. Search for the unixODBC tools on your system.

   The `yum` package manager searches for software already installed or available on your system or from the configured repositories. Depending on your workstation configuration, you may need to use the `sudo` command with your workstation.

```
$ yum search unixODBC
...
 * updates: repos-lax.psychz.net
================ N/S matched: unixODB
C============================
opensips-unixodbc.x86_64 : OpenSIPS unixODBC Storage sup
port
unixODBC-devel.i686 : Development files for programs whi
ch will use the unixODBC library
unixODBC-devel.x86_64 : Development files for programs w
hich will use the unixODBC library
erlang-odbc.x86_64 : A library for unixODBC support in E
rlang
freeradius-unixODBC.x86_64 : Unix ODBC support for freer
adius
unixODBC.i686 : A complete ODBC driver manager for Linux
unixODBC.x86_64 : A complete ODBC driver manager for Lin
ux
```

Make note of the correct package to install for your architecture.

2. Install the appropriate package for your architecture.

In this case the command installs the tools for a 64-bit architecture. A 32-bit package needs the `unixODBC.i686` package.

```
 [admin@nebula-docs-odbc-test-cxmrn ~]$ yum install unix
ODBC.x86_64
 Loaded plugins: fastestmirror, ovl
 Loading mirror speeds from cached hostfile
  * base: mirror.linuxfix.com
  * elrepo: repos.lax-noc.com
  * epel: mirror.hmc.edu
  * extras: centos-distro.cavecreek.net
  * rpmforge: mirror.lstn.net
  * updates: repos-lax.psychz.net
 Resolving Dependencies
 --> Running transaction check
 ---> Package unixODBC.x86_64 0:2.3.1-11.el7 will be ins
talled
 ...
 Complete!
```

3. Verify the files were installed.

```
$ ls /usr/bin/isql
/usr/bin/isql
$ ls /usr/bin/odbcinst
/usr/bin/odbcinst
```

Set up your environment

In this section, you set parameters in your workstation to support your ODBC connection.

1. Copy the library for your architecture from the `Lib` directory on your Linux machine.

Library	Architecture
`/linux/Lib/Linux_x86`	32-bit
`/linux/Lib/Linux_x8664`	64-bit

2. Add the location's path to the `LD_LIBRARY_PATH` environment variable.

   For example if your architecture is 64-bit and you keep the library in your `home` directory:

```
export LD_LIBRARY_PATH=~/linux/Lib/Linux_x8664/
```

3. Use the `echo` command to verify the path was added correctly.

```
echo $LD_LIBRARY_PATH
```

4. Copy the `odbc.ini` file to the `/etc` directory.

```
$ cp ~/linux/Setup/odbc.ini  /etc
```

If you have trouble making the copy, use the `sudo` command to make the move.

5. Copy the `odbcinst.ini` file to the `/etc` directory.

```
$ cp ~/linux/Setup/odbcinst.ini  /etc
```

6. Copy the hidden `.simba.quickstart.ini` file to the `/etc` directory, renaming it in the process to `simbaclient.ini`.

```
$ cp ~/linux/Setup/.simba.quickstart.ini  /etc/simbacli
ent.ini
```

7. Update your environment with the `ODBCSYSINI` and `ODBCINI` variables.

```
$ export ODBCSYSINI=/etc/
$ export ODBCINI=/etc/odbc.ini
```

8. Use the `/usr/bin/odbcinst` command to confirm your settings:

```
$ /usr/bin/odbcinst -j
unixODBC 2.3.1
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /etc/odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8
```

Edit the /etc/simbaclient.ini file

When you are ready, follow this procedure to configure the driver.

1. Edit the `/etc/simbaclient.ini` file with your favorite editor. 2.

2. Change the `ErrorMessagesPath` property to point to the location where you unzipped the client.

```
[Driver]
ErrorMessagesPath=<path_to_error_messages_directory>
```

3. Comment out the `# Generic ODBCInstLib` value.
4. Uncomment the `ODBCInstLib` property.

   When you are done, your file looks like the following:

```
# Generic ODBCInstLib
#   iODBC
#ODBCInstLib=libiodbcinst.so

#   SimbaDM / unixODBC
ODBCInstLib=libodbcinst.so
```

5. Save and close the `/etc/simbaclient.ini` file.

Edit the odbcinst.ini file

The `odbcinst.ini` file is a registry and configuration file for ODBC drivers. Depending on your workstation architecture, you configure the 32-bit or 64-bit driver.

1. Open the file `/etc/odbcinst.ini` in your favorite editor.
2. Comment out the driver that you don't need.

   For example, if you are using 64-bit, comment out 32-bit.

3. Edit the `Driver` line so that it contains the path to the file `libSimbaClient.so`

   Use the path where you copied the library files. For example, for the 64-bit ODBC driver:

```
 [ThoughtSpot(x64)]
 APILevel            = 1
 ConnectFunctions    = YYY
 Description         = ThoughtSpot 64bit ODBC driver
 Driver              = /home/admin/linux/Bin/Linux_x866
4/libSimbaClient.so
 DriverODBCVer       = 03.52
 SQLLevel            = 1
```

4. Make sure the remaining driver is named `ThoughtSpot` without any special characters.

   When you are done, you should see something similar to the following:

```
 #[ThoughtSpot]
 #APILevel            = 1
 #ConnectFunctions    = YYY
 #Description         = ThoughtSpot 32bit ODBC driver
 #Driver              = /usr/local/scaligent/toolchain/l
ocal/simba/odbc/linux/Bin/Linux_x86/libSimbaClient.so
 #DriverODBCVer       = 03.80
 #SQLLevel            = 1

 [ThoughtSpot]
 APILevel            = 1
 ConnectFunctions    = YYY
 Description         = ThoughtSpot 64bit ODBC driver
 Driver              = /home/admin/linux/Bin/Linux_x866
4/libSimbaClient.so
 DriverODBCVer       = 03.80
 SQLLevel            = 1
```

5. Save and close the `/etc/odbcinst.ini` file.

Edit the odbc.ini file

The `odbc.ini` file is a registry and configuration file for ODBC DSNs (Data Source Names). This file relies on the drivers registered in the `/etc/odbcinst.ini` file. Depending on your workstation architecture, you configure the 32-bit or 64-bit driver.

1. Open the file `/etc/odbc.ini` in the editor of your choice.

2. Comment out the configuration that you don't need.

   For example, if you are using 64-bit, comment out 32-bit.

3. Locate the `Description` section for the type of Linux you are using (32-bit or 64-bit).
4. Locate the line that begins with `ServerList`.
5. Replace `127.0.0.1` with a comma separated list of the IP addresses of each node on the ThoughtSpot instance.

   The syntax for the `ServerList` is:

   ```
    ServerList = <node1_IP> 12345, <node2_IP> 12345 [, <nod
   e3_IP> 12345, ...]
   ```

   If you need to obtain the IP addresses of the ThoughtSpot cluster nodes, run the command `tscli node ls` from a Linux shell on a ThoughtSpot appliance.

6. Do not edit the port number, leave it as `12345`.

   When you are done, your entry will look similar to the following (this example is for the 64-bit ODBC driver):

   ```
    [ThoughtSpot]
    Description = ThoughtSpot 64-bit ODBC Driver
    Driver = ThoughtSpot
    ServerList =  172.18.231.17 12345
    Locale = en-US
    ErrorMessagesPath = /home/admin/linux/ErrorMessages
    UseSsl = 0
    #SSLCertFile = # Set the SSL certificate file path. Th
   e certificate file can be obtained by extracting the SD
   K tarball
    #LogLevel = 0 # Set log level to enable debug logging
    #LogPath = # Set the debug log files path
    DATABASE = # Set the default database to connect to
    SCHEMA = # Set the default schema to connect to
   ```

7. Save and close the `odbc.ini` file.

Test your ODBC connection

At this point, you can test your ODBC connection to ThoughtSpot. It is important to recall that the username/password you use belongs to a ThoughtSpot application user. Typically, this user is a user with data management or administrative privileges on the application.

1. Before trying the ODBC connection, make sure you can use this username/password to login into the ThoughtSpot application.
2. Confirm the user's privileges by going to the **Data** tab.
3. Go back to your workstation's terminal shell.
4. Use the `/usr/bin/isql` and confirm you can connect.

   Specify the `ThoughtSpot` DSN:

```
/usr/bin/isql –v ThoughtSpot tsadmin adminpwd
+---------------------------------------+
| Connected!                            |
|                                       |
| sql–statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

Now, you are ready to begin using the connection you've configured.

Best Practices for Using ODBC

Summary: To successfully use ODBC, following these best practices is recommended.

When developing tools that use the ODBC driver, use these best practices:

- When setting up ODBC for the first time, begin by using the ThoughtSpot `tsload` for the initial data loads. This allows you to do more in-depth troubleshooting on any initial loading issues. After initial loads work properly, switch to ODBC to perform incremental loads.

- You should create the parameterized SQL statement outside of ODBC. Using this method, the SQL statement can be sent to ThoughtSpot in batches by the ODBC driver, so you only have to update the memory itself. ETL tools have this implemented already (end users shouldn't have to actually write the `INSERT` statement). But as a developer, you may be writing code that leverages the ODBC driver, so this tip can help you write your SQL for the best performance with the driver.

- Data can be loaded into a table through multiple parallel connections. You can achieve this by splitting the input data into multiple parts. Then, load those individual parts through multiple parallel connections. You can use parallel loading even while loading to a single table or multiple tables at the same time.

- When doing an incremental data load, note that the same `UPSERT` behavior that occurs in TQL also occurs. This means that if you import a row whose primary key matches an existing row, the existing row will be updated with the new values.

Related information

- Enable ODBC logs
- Introduction to loading and managing data
- Loading and constraints

JDBC Driver Overview

Summary: Use JDBC to interact with databases in a standard manner.

Java Database Connectivity (JDBC) is a Java standard API that allows applications to interact with databases in a standard manner. ThoughtSpot has JDBC support through a JDBC driver that we provide.

Connector type

There are different types of JDBC connectors. Driver types categorize the technology used to connect to the database. The ThoughtSpot JDBC drive is a type 4 connector. It uses Java to implement a networking protocol for communicating with ThoughtSpot.

This driver is Java driver. There is no client installation or configuration.

When to use JDBC

JDBC can be used whenever you want to connect to ThoughtSpot to insert data programmatically from a Java program or application. You should begin by using the ThoughtSpot Loader for initial data loads and then use JDBC for incremental loads. This is because the ThoughtSpot Loader is generally faster than JDBC. Information on using the ThoughtSpot Loader is available in the ThoughtSpot Administrator Guide.

Version Compatibility

To ensure compatibility, always use the JDBC driver with the same version number as the ThoughtSpot instance to which you are connecting.

Performance Considerations

These are some general recommendations for maximizing the performance of JDBC:

- Insert in batches rather than doing single inserts at a time using the

`PreparedStatement::addBatch()` and `PreparedStatement::executeBatch` commands.

- If you need to upload a lot of data, consider running multiple connections with batch inserts in parallel.

Note: The ETL tool must add a data transformation step if the source column data type does not exactly match the target's, ThoughtSpot's, column data type. The driver does not do any implicit conversions.

Use the JDBC Driver

Summary: How to configure the JDBC driver.

ThoughtSpot's ODBC connection relies on the SimbaEngine X SDK to connect through ODBC or JDBC to ThoughtSpot's remote data stores. The instructions on this page explain how to configure the JDBC driver.

The ThoughtSpot JDBC driver is supplied by a `.jar` file you install on a workstation. This workstation is the same machine where you plan to run your ETL activities.

JDBC configuration parameters

Information	Description
Driver name	`com.simba.client.core.jdbc4.SCJDBC4Driver`
Server IP address	The ThoughtSpot appliance URL or IP address.
Simba port	The simba port, which is `12345` by default.
Database name	This is not the machine login username. The ThoughtSpot Database name to connect to.
username	The name of a ThoughtSpot user with administrator permissions.
password	The password of a ThoughtSpot application user. This is not the machine or SSH userpassword.

For more JDBC configuration options, see also:

- JDBC properties reference in this ThoughtSpot documentation
- SimbaClient for JDBC Configuration Properties reference

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports 12345`
5. Exit or close the shell.

Install the driver

The JDBC driver is a `.jar` packaged application. To use the package, you download it, install it

1. Log in to the local machine where you want to install the JDBC driver.
2. Click **Here** to download the JDBC driver.
3. Click **JDBC Driver** to download the file `thoughtspot_jdbc<version>.jar`.
4. Move the driver to the desired directory on your local machine.
5. Add the downloaded JDBC driver to your Java class path on the local machine.

Write your application

Using JDBC with ThoughtSpot is the same as using any other JDBC driver with any other database. You must provide the connection information, create a connection, execute statements, and close the connection.

Specify each of the nodes in the cluster in the connection string, as shown. This enables high availability for JDBC connections. To find out the nodes in the cluster, you can run the command `tscli node ls` from the Linux shell on the ThoughtSpot instance.

The format for the connection is:

```
jdbc:simba://<node1>:12345,<node2>:12345,<node3>:12345[,…];
           LoginTimeout=<seconds>;DATABASE=<db>;SCHEMA=<schema>
```

For example:

```
jdbc:simba://192.168.2.248:12345,192.168.2.249:12345,192.16
8.2.247:12345;
           LoginTimeout=5;DATABASE=test;SCHEMA=falcon_default_s
chema
```

As shown, the `DATABASE` and `SCHEMA` parameters need to be in all caps. For the `simba` JDBC driver to work with Spark, the `DATABASE` and `SCHEMA` must be specified in the URL. They cannot be specified as a name/value pair as a map or property. For example:

```
val tssqldf1 = sparkSession.read.format("jdbc").options(Map("ur
l" ->
"jdbc:simba://10.84.78.181:12345;DATABASE=movieratings;SCHEMA=f
alcon_default_schema", "driver" ->
"com.simba.client.core.jdbc4.SCJDBC4Driver", "dbtable" -> "Movi
es", "user" ->
"tsadmin", "password" -> "admin")).load()
```

This `InsertData.java` example shows how to use ThoughtSpot with JDBC. This is an example of a reference JDBC application:

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class InsertData {

  // JDBC class to use.
  private static final String DB_DRIVER = "com.simba.client.cor
e.jdbc4.SCJDBC4Driver";
  // jdbc_example should be an existing database.

  private static final String DB_CONNECTION = "jdbc:simba://19
2.168.2.129:12345;
     192.168.2.249:12345,192.168.2.247:12345;
     LoginTimeout=5;DATABASE=jdbc_example;SCHEMA=falcon_defaul
t_schema";

  private static final String TABLE_NAME = "jdbc_example";
  private static final String DB_USER = "<username>";
  private static final String DB_PASSWORD = "<password>";

  // Assuming everything in local directory use:
  //   javac InsertData.java
  //   java -cp .:thoughtspot_jdbc4.jar InsertData
  public static void main(String[] argv) {

    try {
      insertRecordsIntoTable();
    }
    catch (SQLException e) {
      System.out.println(e.getMessage());
    }
  }

  /**
   * Insert some records using batch updates.
   *  Assumes a table exists:  CREATE TABLE "jdbc_example" ( "t
ext" varchar(10) );
   */
  private static void insertRecordsIntoTable() throws SQLExcept
ion {

    System.out.println("Inserting records.");
    Connection dbConnection = getDBConnection();
```

```
    PreparedStatement preparedStatement = null;
    String insertTableSQL = "INSERT INTO falcon_default_schem
a.jdbc_example (text) VALUES (?)";

    try {
      preparedStatement = dbConnection.prepareStatement(insertT
ableSQL);

      // Create multiple statements and add to a batch update.
      for (int cnt = 1; cnt <= 10; cnt++) {
        preparedStatement.setString(1, "some string " + cnt);
        preparedStatement.addBatch();
        System.out.println("Record " + cnt + " was added to th
e batch!");
      }
      preparedStatement.executeBatch();  // For large numbers o
f records, recommend doing sets of executeBatch commands.
      System.out.println("Records committed");

    }
    catch (SQLException sqle) {
      sqle.printStackTrace();
    }
    finally {

      if (preparedStatement != null) {
        preparedStatement.close();
      }
      if (dbConnection != null) {
        dbConnection.close();
      }
    }
  }

  /** Create a connection to the database. */
  private static Connection getDBConnection() {
    Connection dbConnection = null;
    try {
      Class.forName(DB_DRIVER);
    }
    catch (ClassNotFoundException e) {
      System.out.println(e.getMessage());
    }
    try {
      dbConnection = DriverManager.getConnection(DB_CONNECTIO
```

```
N, DB_USER,DB_PASSWORD);
      return dbConnection;
    }
    catch (SQLException sqle) {
      System.out.println(sqle.getMessage());
    }

    return dbConnection;
  }

}
```

Related Information

- Enable JDBC logs
- Connection configuration
- Supported SQL commands

Set up the JDBC driver for Pentaho

Summary: JDBC to connect to the ThoughtSpot Simba server from Pentaho.

You can use the Pentaho Data Integration (PDI) to create a JDBC connection. The Pentaho Data Integration (PDI) suite is a comprehensive data integration and business analytics platform. You can use it to create a JDBC connection to ThoughtSpot.

PDI consists of a core data integration (ETL) engine and GUI applications that allow you to define data integration jobs and transformations. Through Pentaho, we primarily use the JDBC driver to set up a connection. The process is not as complicated as with SSIS, and is much more lenient.

Community and enterprise editions of PDI are available. Using the community edition is sufficient, though you may use the enterprise edition, which is subscription based, and therefore contains extra features and provides technical support.

Use JDBC to connect to the ThoughtSpot Simba server from Pentaho. The connection will be made between a new ThoughtSpot Table Input and Output objects.

Check the ThoughtSpot IP and the simba_server status

Before you begin, you need to know the IP address or DNS name of the server you intend to connect your server to.

1. SSH as `admin` or the `thoughtspot` user to your ThoughtSpot node.
2. Verify the node IP(s).

```
$ tscli node ls
172.18.231.17
172.18.231.18
```

3. Make a note of each IP; there may be more than one.
4. Configure the ThoughtSpot firewall to allow connections from your ETL client, by running the following command on any ThoughtSpot node: `tscli firewall open-ports --ports`

`12345`

5. Exit or close the shell.

Install the Simba drivers in the Pentaho directories

Before starting the Pentaho Data Integration (PDI) client and creating the connection, ensure that the Simba JDBC client libraries are present in the Pentaho client/server machines. This will ensure that the drivers picked up at runtime.

1. Log in to the local machine where you have alredy installed the Pentaho Data Integration (PDI) client.
2. Click **Here** to download the JDBC driver.
3. Click **JDBC Driver** to download the file `thoughtspot_jdbc<version>.jar`.
4. Copy the `thoughtspot_jdbc<version>.jar` file to the following directories:
    - `<Pentaho_install_dir>/server/data-integration-server/tomcat/webapps/pentaho-di/WED-INF/lib/`
    - `<Pentaho_install_dir>/design-tools/data-integration/lib/`
    - `<Pentaho_install_dir>/server/data-integration-server/tomcat/lib/`
    - `<Pentaho_install_dir>/design-tools/data-integration/plugins/spoon/agile-bi/lib/`

Set up the driver

This section explains how to set up the JDBC driver using Pentaho. These instructions use Spoon, the graphical transformation and job designer associated with the PDI suite. It is also known as the Kettle project.

Create a transformation

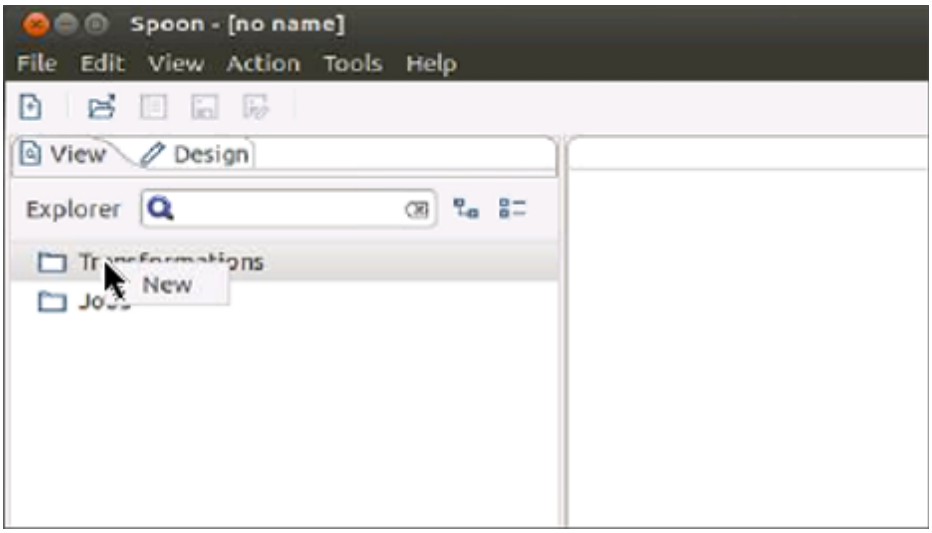
Do the following on your ETL workstation with the Pentaho client:

1. Open the PDI client.

```
./spoon.sh &>/dev/null &
```

2. Right click **View > Transformations** tab.

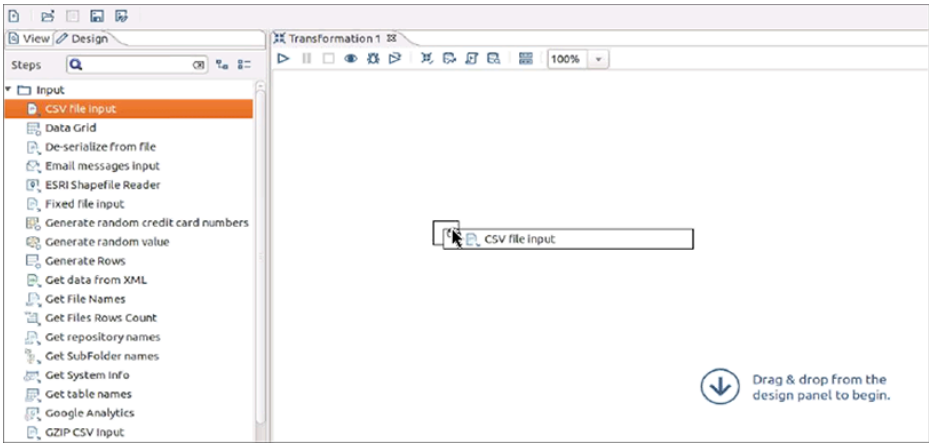
3. Click **New** to create a new transformation.



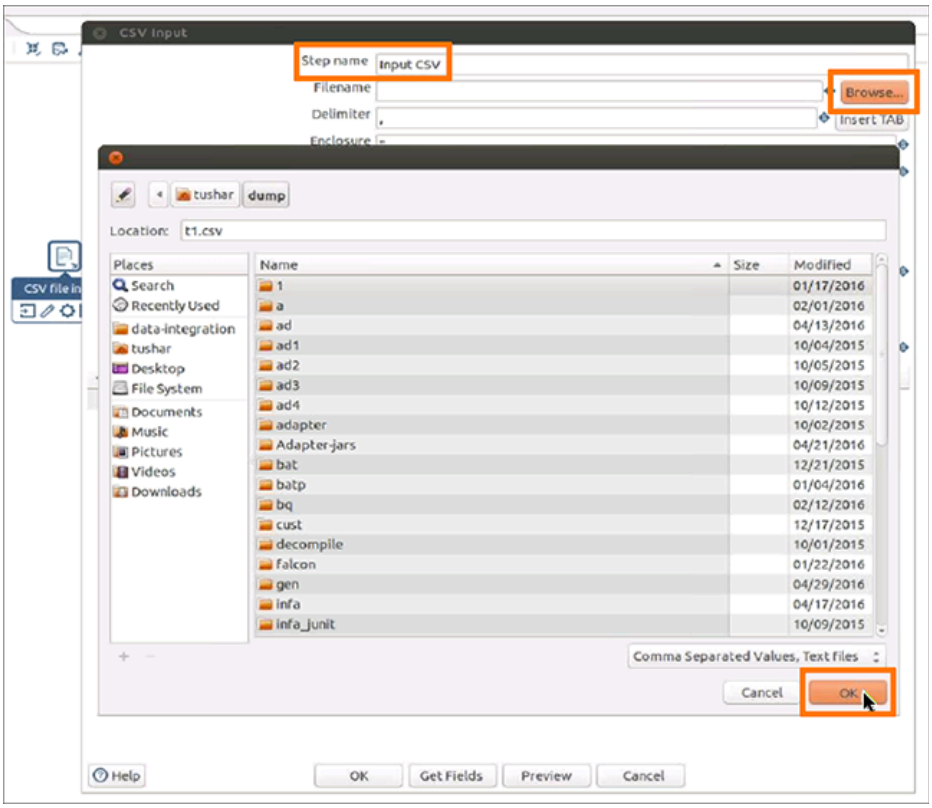
4. Click **Input** under the **Design** tab to expand it.

5. Drag and drop **CSV File Input** to the **Transformation** window.

   This opens a new CSV file.



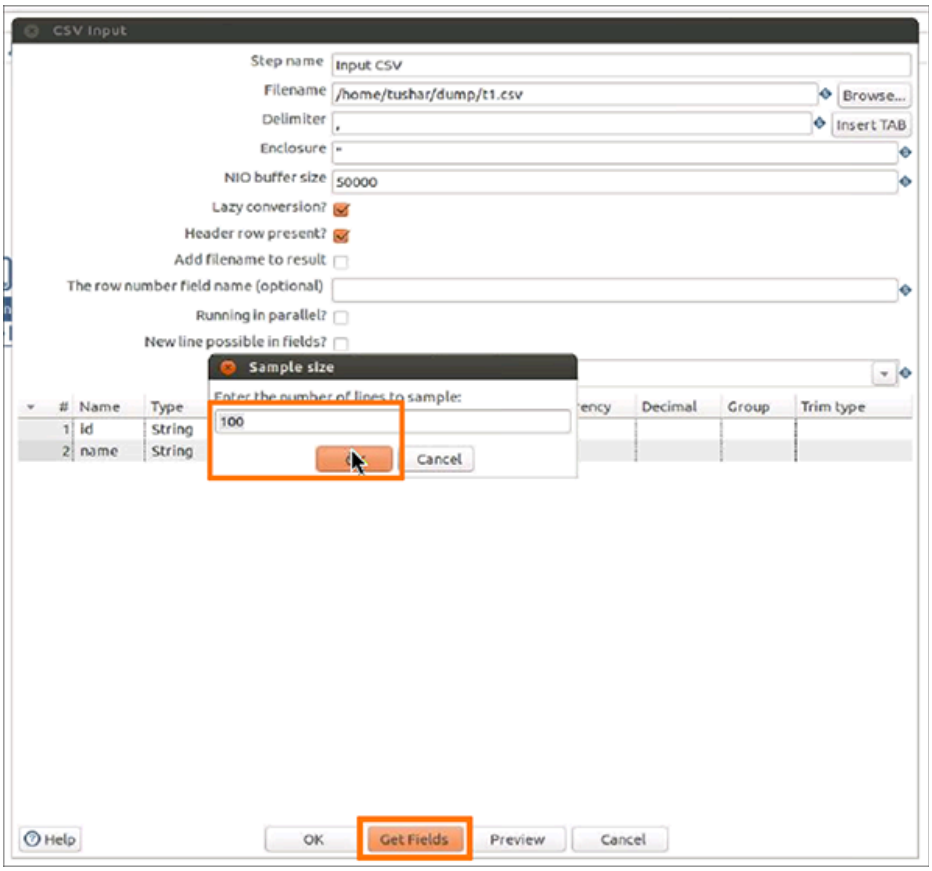
6. Double-click the **CSV File Input** icon to open the **CSV Input** dialog .
7. Name the **Step**.
8. Click **Browse** next to the **Filename** field and provide the file you want to read from.
9. Click **OK**.



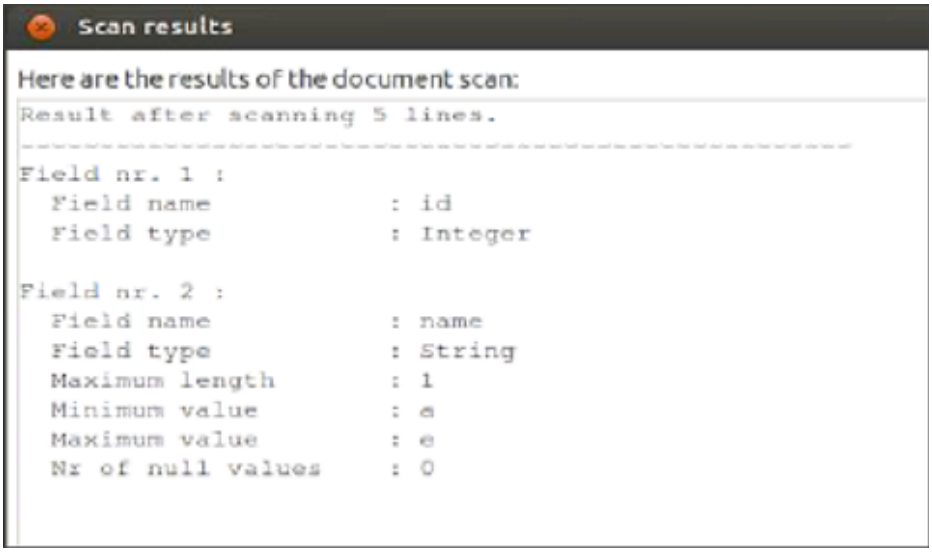
10. In the CSV Input dialog, click **Get Fields**.
11. Enter the number of lines you would like to sample in the Sample size dialog.

The default setting is 100.

1. Click **OK** when you are ready.



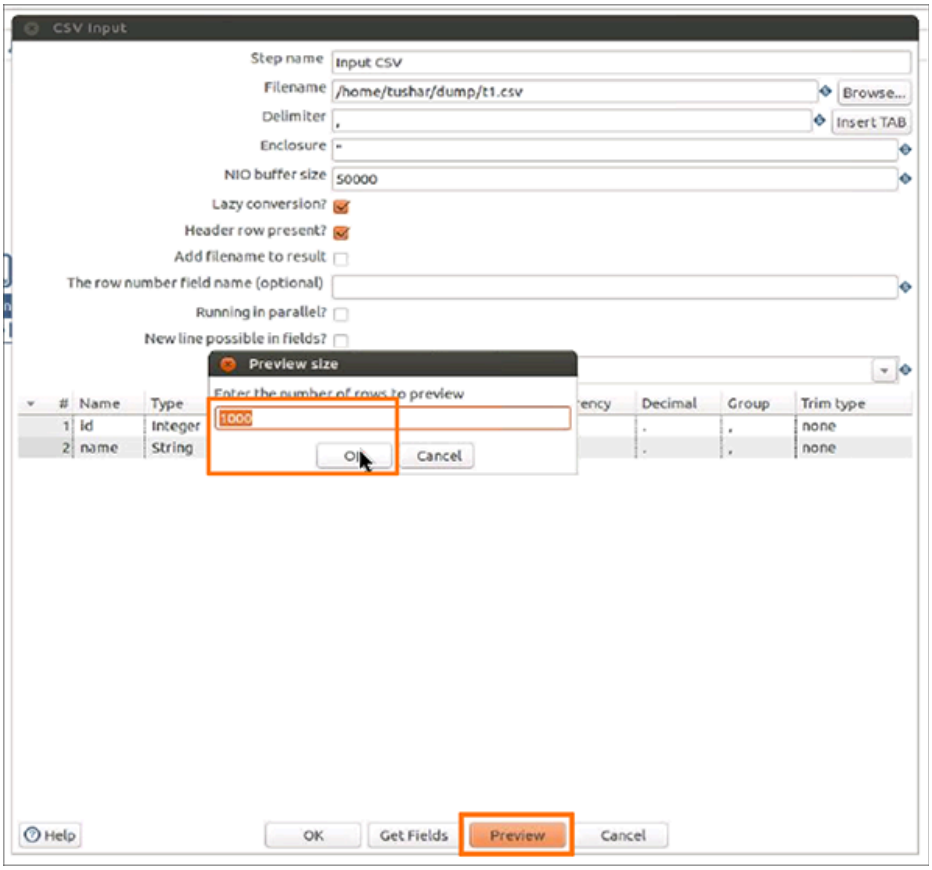
The tool reads the file and suggests the field name and type.



2. Click **Preview** to preview the data.

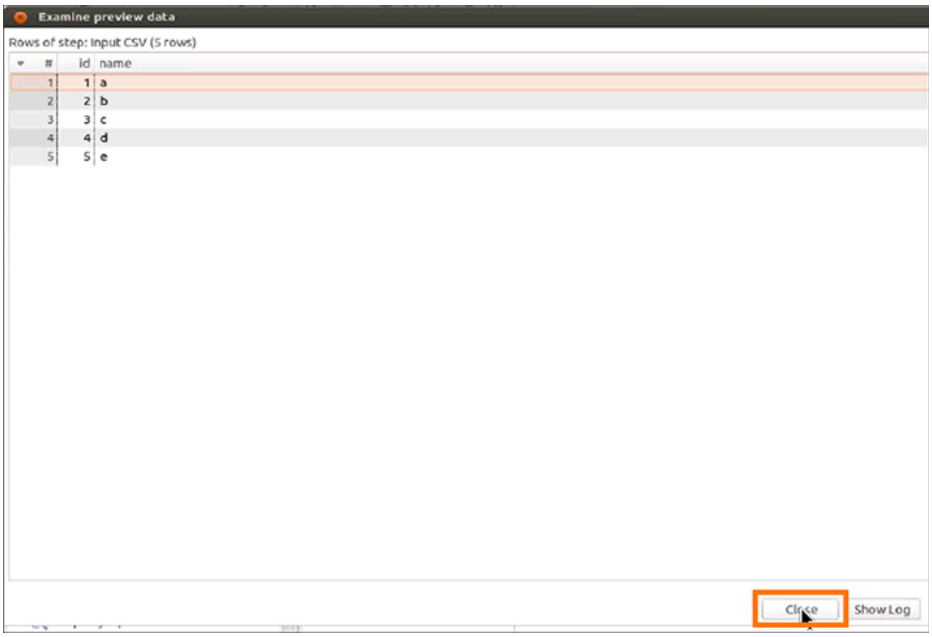
3. Enter the number of rows to preview in the **Preview size** dialog.

The default setting is 1000. Click **OK** to start the transformation in preview.



4. Examine the preview data, then click **Close**.

You may want to verify that you are able to read the data using the SQL query from ThoughtSpot.

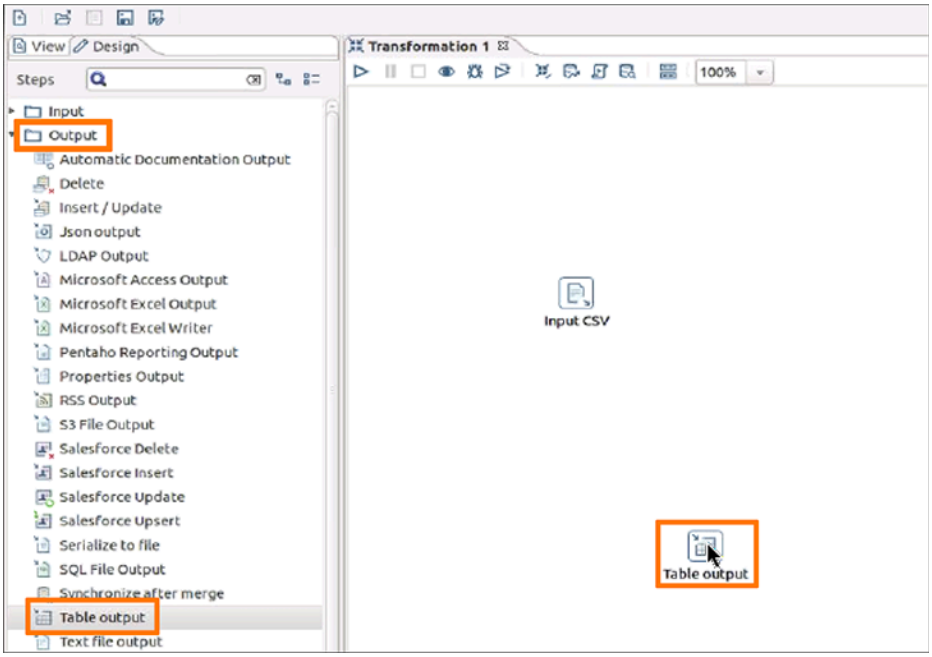


5. Click **OK** in the CSV Input dialog to confirm your CSV input settings.

Define the Output

1. Click **Design > Output**.

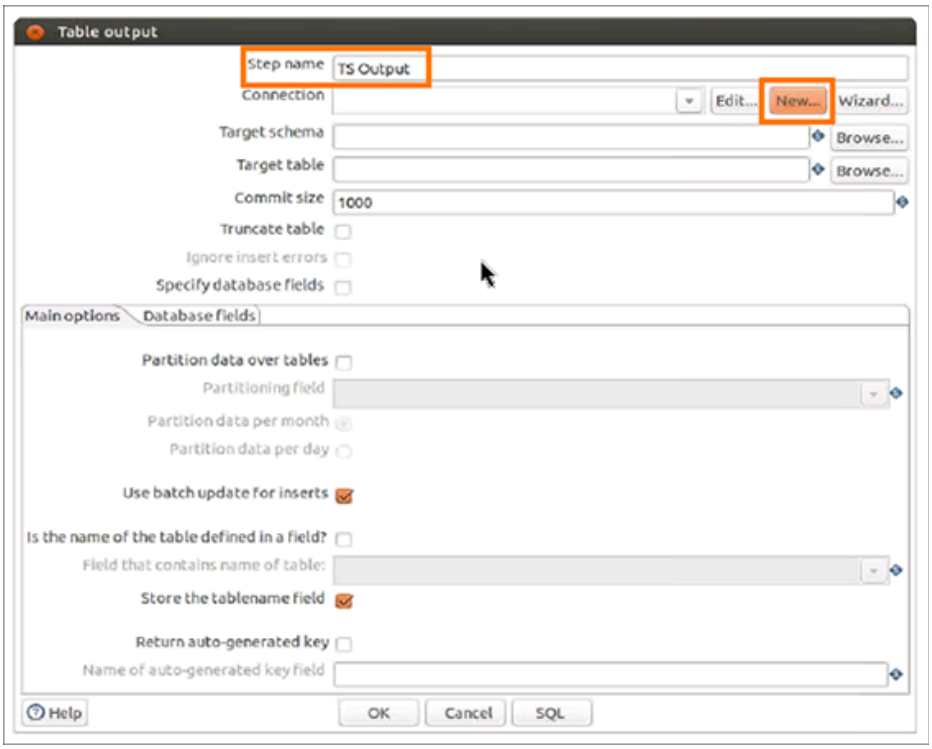
2. Drag and drop **Table output** to the **Transformation** window.



3. Double click the **Table output** icon to open the Table output dialog.

4. Enter a **Step name**.

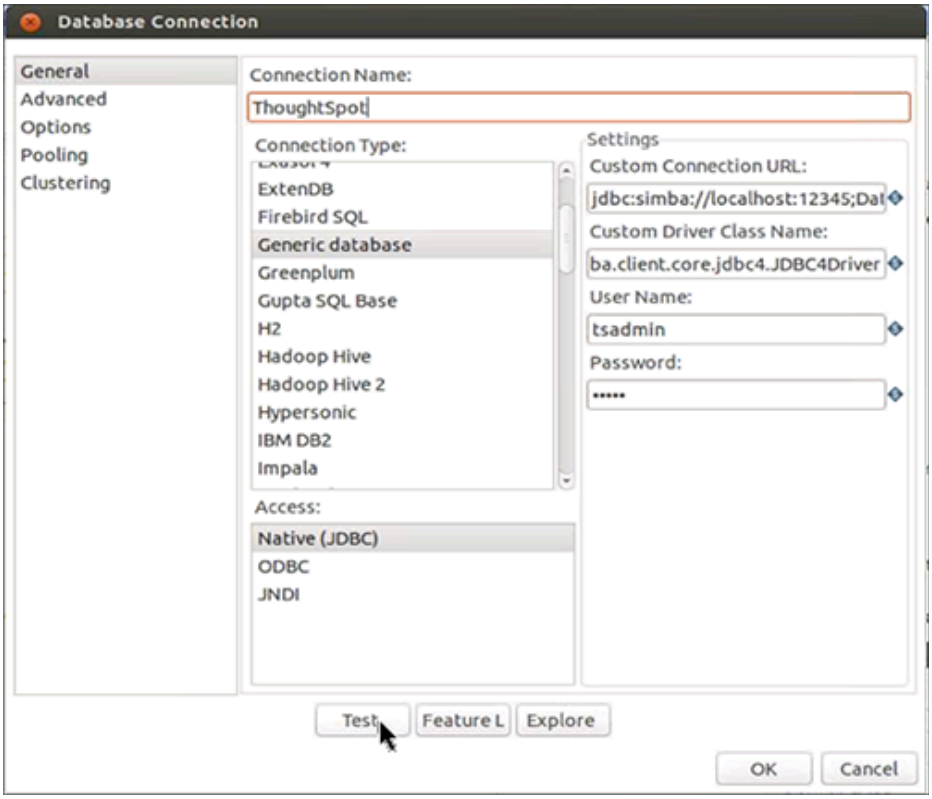
5. Click **New** to create a new connection.



6. Enter or select the following information in the Database Connection dialog:

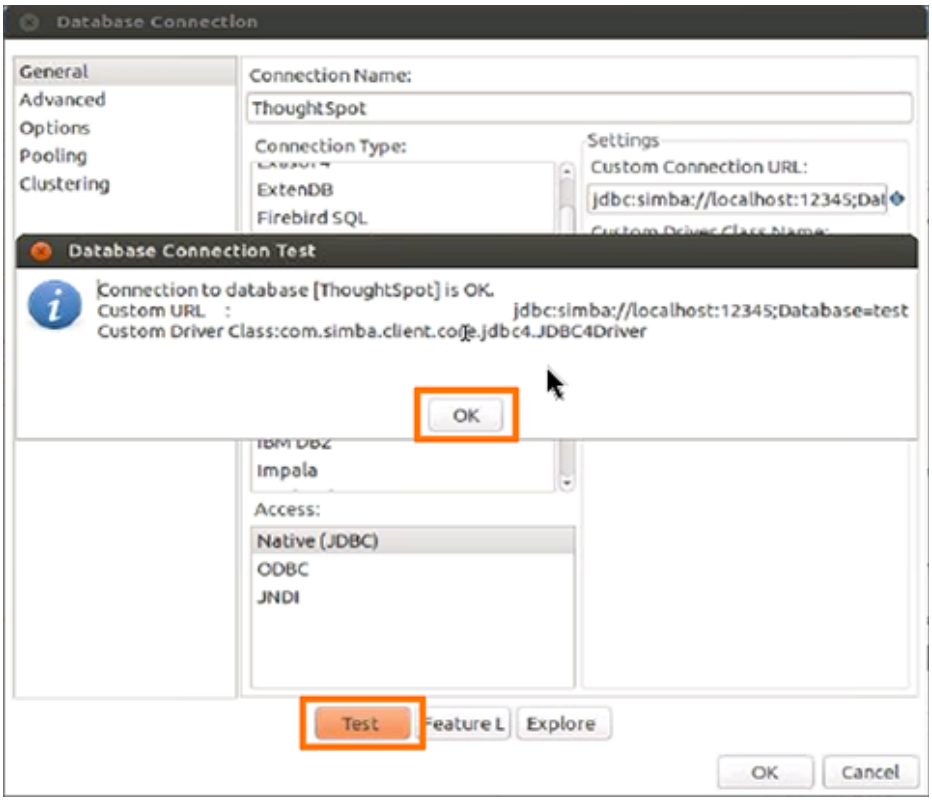
Field	Description
Connection Name	Any string.
Connection Type	Generic database
Access	Native (JDBC)
Custom Connection URL	`jdbc:simba://SERVER_IP:12345;Database=DATABASE_or_SCHEMA_NAME </code>` The IP is a node in your ThoughtSpot cluster. The name or schema of the database you want to connect to. Use TQL to create a database name if needed. Ensure that there are no leading or trailing spaces.

Custom Driver Class Name	`com.simba.client.core.jdbc4.JDBC4Driver` Ensure that there are no leading or trailing spaces.
User Name	A ThoughtSpot username. If you leave this empty, you are prompted for it at connection time. This user should have **Data Management** privileges on ThoughtSpot.
Password	The password for the **User Name**. If you leave this empty, you are prompted for it at connection time.



7. Click **Test** to test your database connection.

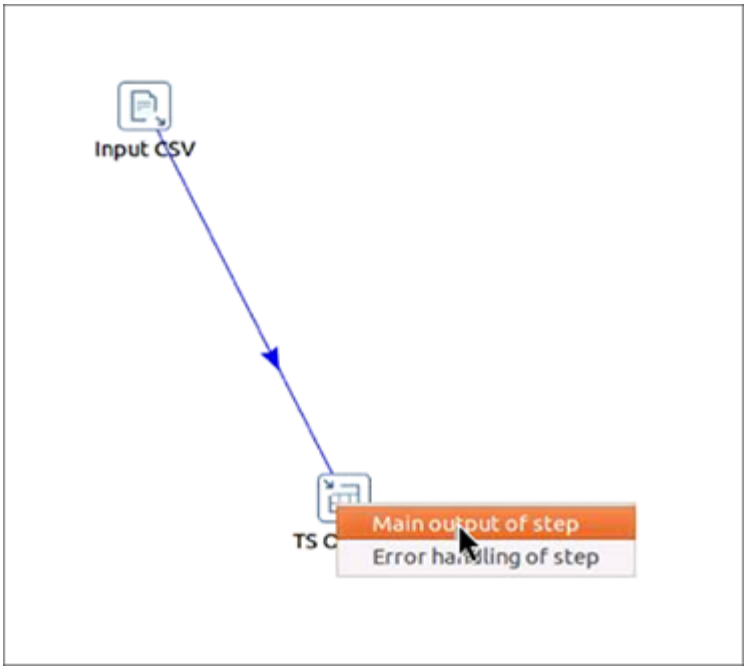
8. If you are able to make a successful connection to the ThoughtSpot Simba Server, click **OK**.



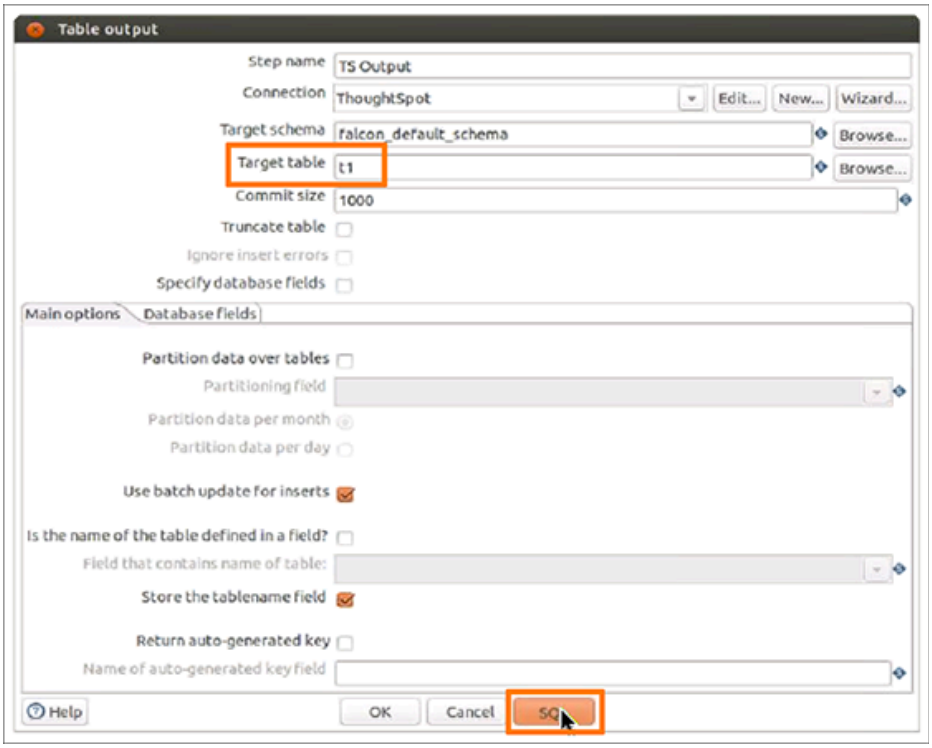
9. Click **OK** in the Database Connection dialog to create the new connection.

Import data

1. In the **Table output** dialog, select the connection you just created.
2. Click **Browse** next to the **Target schema** field and select your **Target schema**.
3. Click **OK** when you are done.
4. Connect the **Input CSV** icon to the **Table output** icon by clicking and dragging an arrow.
5. When prompted, choose **Main output of step**.

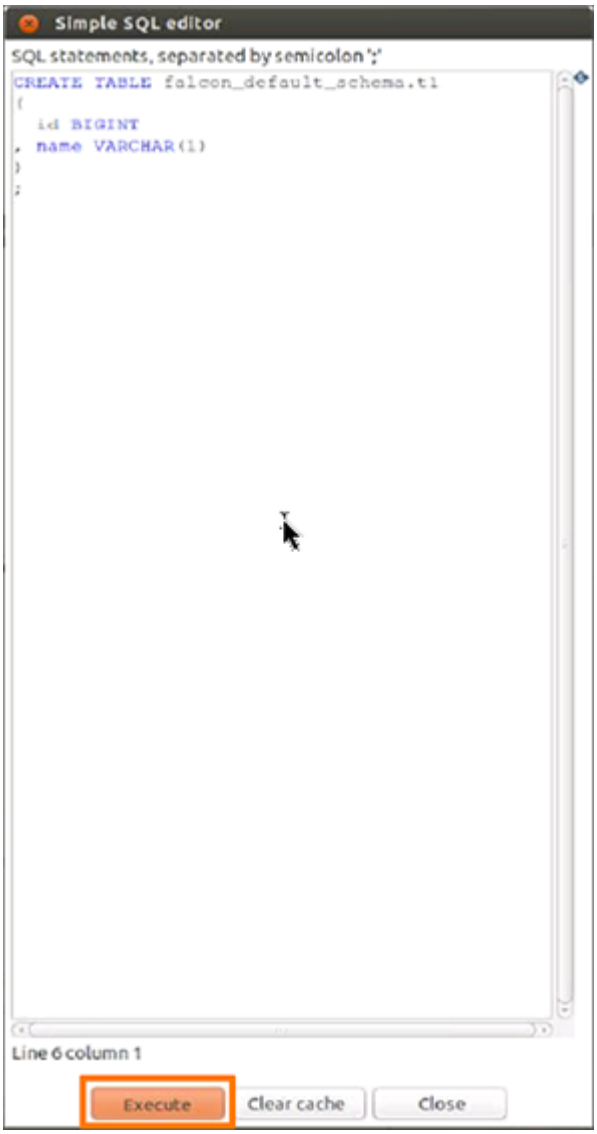


6. Double click the **Table output** icon to reopen the **Table output** dialog.
7. Enter a **Target table name**.
8. Click **SQL**.



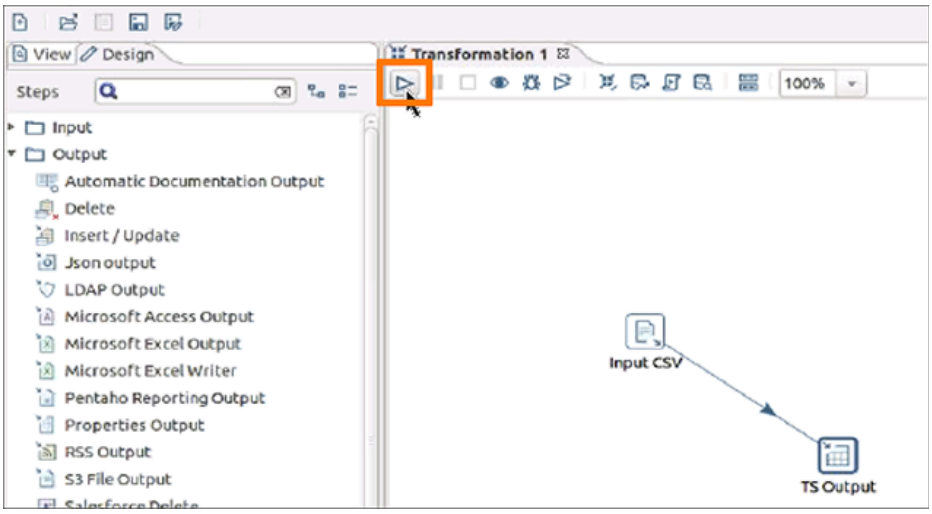
9. In the **Simple SQL editor** dialog, click **Execute**.

   The system processes and then displays the results of the SQL statements.

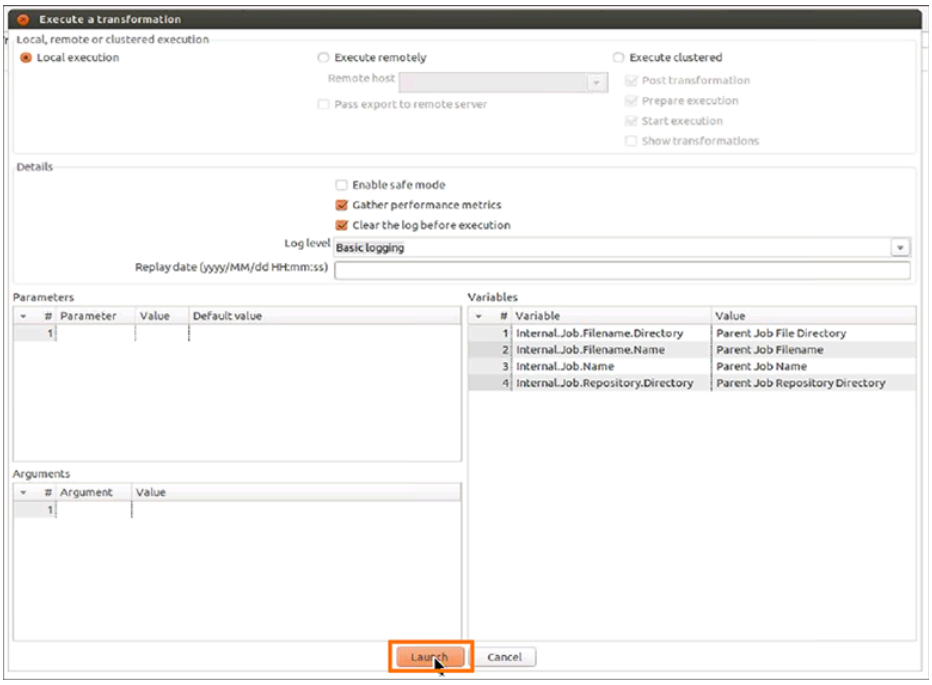


10. Close all open dialogs.

11. Click the **Play** button at the top of the **Transformation** window to execute the transformation.

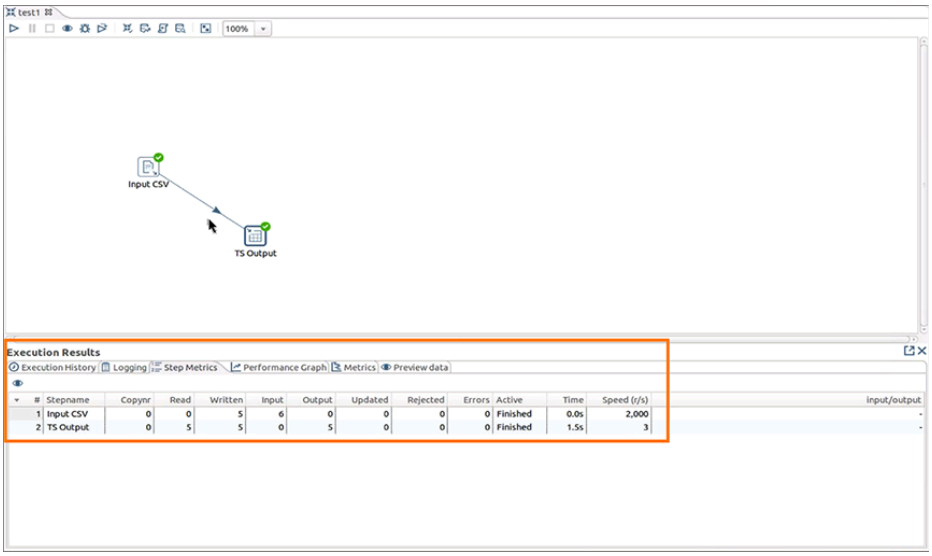


12. Click **Launch** in the **Execute a transformation** dialog.



The system prompts you to save it if you have not already.

13. View the **Execution Results**.


test1
100%
Input CSV
TS Output
Execution Results
Execution History
Logging
Step Metrics
Performance Graph
Metrics
Preview data
Stepname Copynr Read Written Input Output Updated Rejected Errors Active Time Speed (r/s) input/output
1 Input CSV 0 0 5 6 0 0 0 0 Finished 0.0s 2,000 -
2 TS Output 0 5 5 0 5 0 0 0 Finished 1.5s 3 -

Troubleshooting Data Integrations

Summary: Learn how to fix connection issues.

This section can help if you're having trouble creating a connection or need to find out more information about what is going on with ODBC or JDBC.

The information contained here is very basic, and mostly about how to enable logs on the client side. If you need more detailed troubleshooting information or help, please contact ThoughtSpot Support.

- **Enable ODBC Logs**

  If you need more information in order to troubleshoot ODBC connections, you can enable logging for ODBC.
- **Enable JDBC Logs**

  To enable logging for JDBC, add the logging parameters to the connect string. Logs are stored on ThoughtSpot.
- **Schema not found error with ODBC**

  When connecting with ODBC, you need to specify both the database and schema to connect to. If no schema is supplied, you will get an error indicating that the schema could not be found.
- **How to improve throughput of the load**

  The transaction/commit size value can improve the throughput of the load when setting up the ODBC Driver.
- **ODBC tracing on Windows**

  Using logs to aid in troubleshooting.

Enable ODBC Logs

Summary: Learn how to troubleshoot ODBC connections.

If you need more information in order to troubleshoot ODBC connections, you can enable logging for ODBC on the workstation you use for connecting to ThoughtSpot. There are two points where you can enable logging:

- the workstation where you run your ETL activities
- the server where the Simba service is running

On both workstation and servers, the verbosity of the log is controlled by the `LogLevel` property. This property can be one of the following:

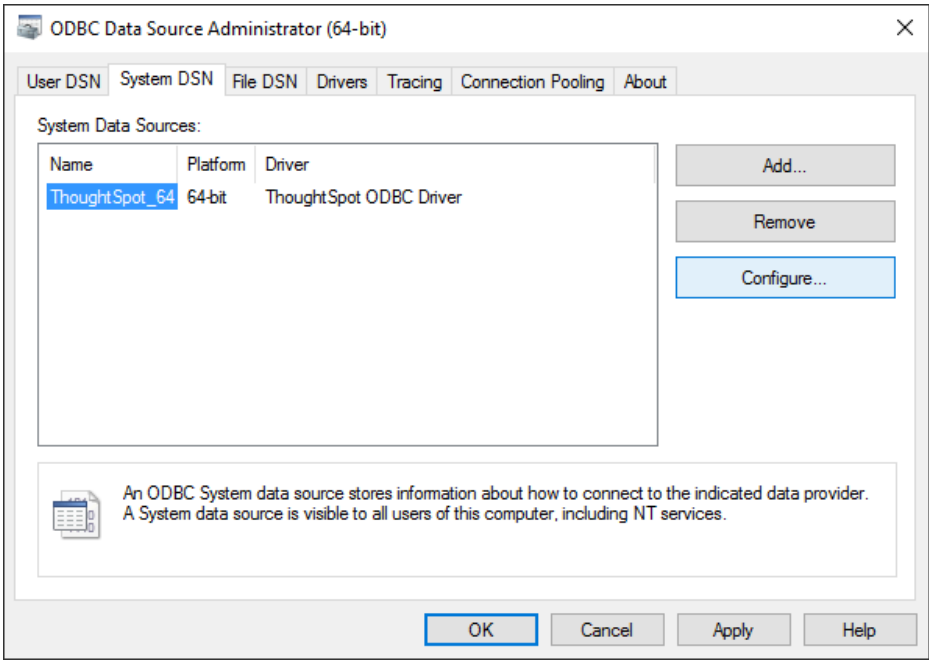
- `0` or `LOG_OFF`: no logging occurs
- `1` or `LOG_FATAL`: only log fatal errors
- `2` or `LOG_ERROR`: log all errors
- `3` or `LOG_WARNING`: log all errors and warnings
- `4` or `LOG_INFO`: log all errors, warnings, and informational messages
- `5` or `LOG_DEBUG`: log method entry and exit points and parameter values for debugging
- `6` or `LOG_TRACE`: log all method entry points

Larger values include the information from lessor values. For example, if you set `3` or `LOG_WARNING`, you log all warnings *and* all errors.

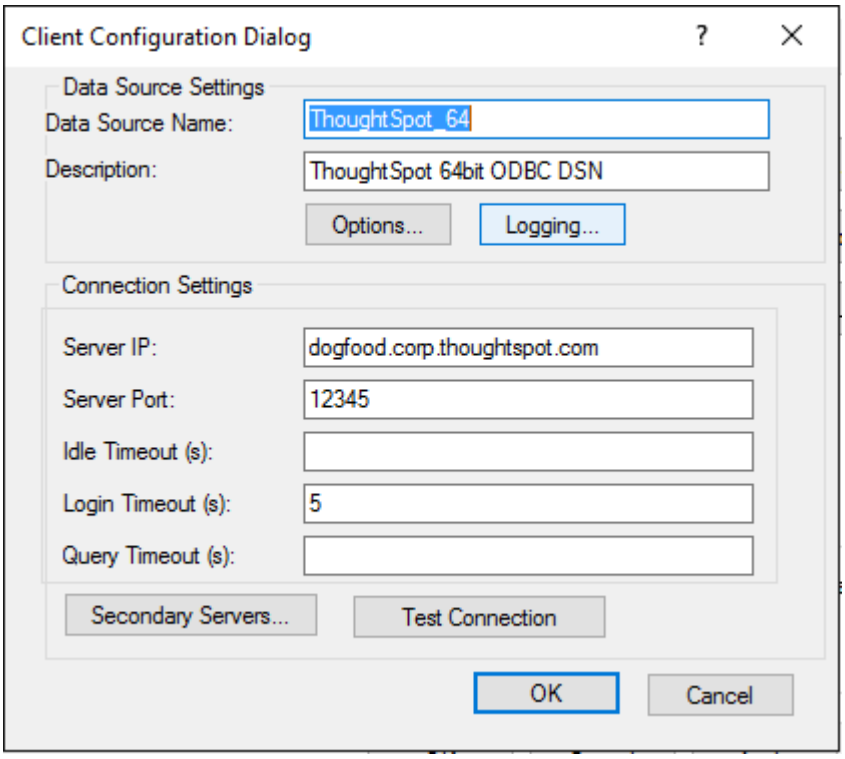
Enable ODBC logs on a Windows workstation

To enable ODBC logs on Windows:

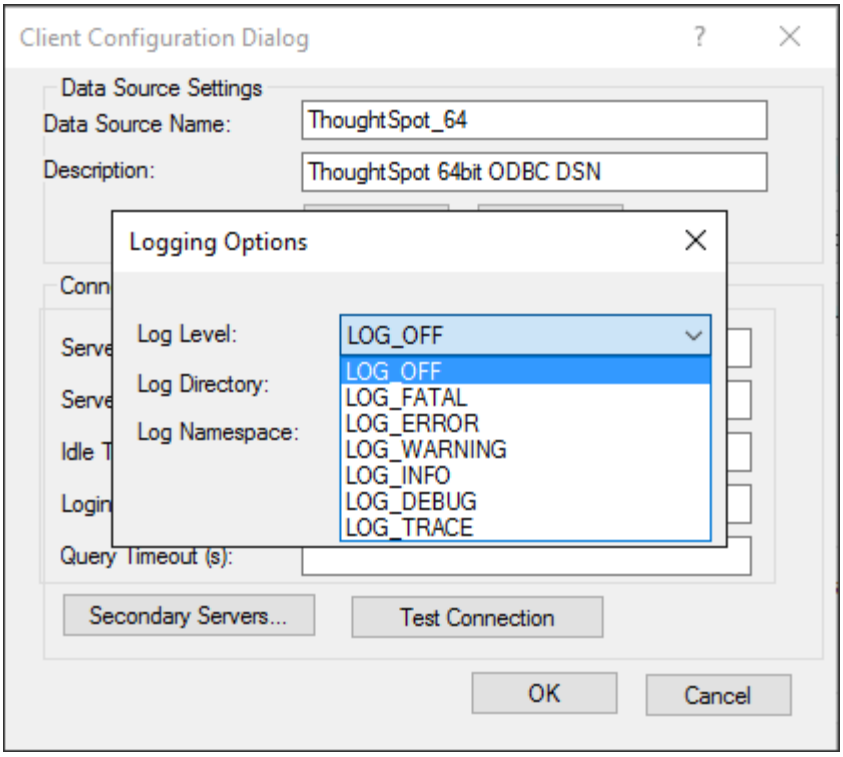
1. Open the **ODBC Data Source Administrator** and select the **System DSN** tab.
2. Select your ThoughtSpot data source and click **Configure**.



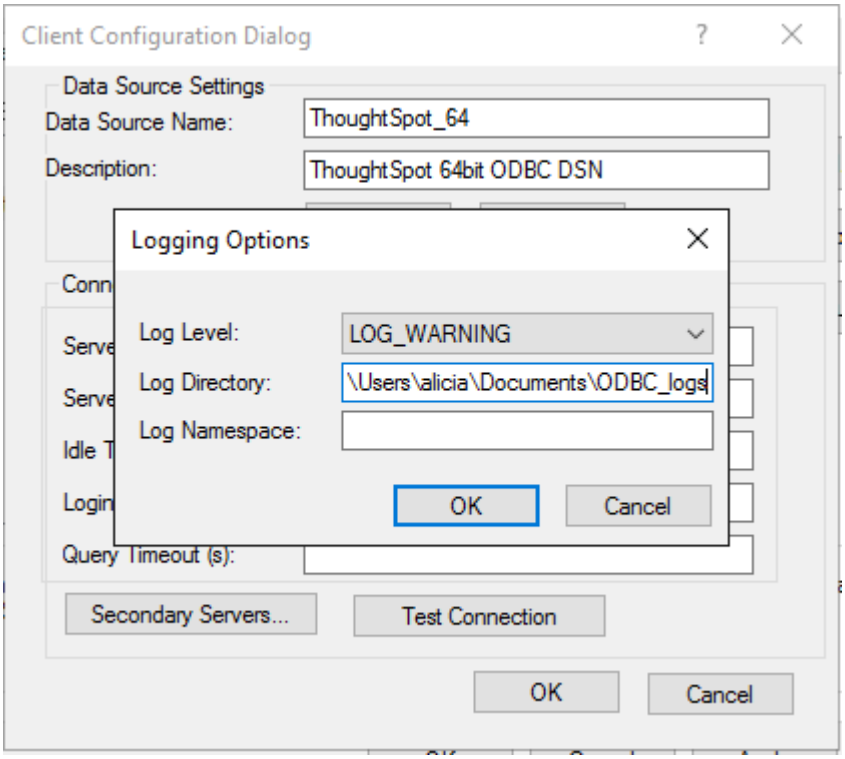
3. In the Client Configuration Dialog, click **Logging**.



4. Choose a **Log Level**, depending on what level of verbosity you want to show in the logs.



5. For **Log Directory:**, type in the fully qualified path where you want the logs to be saved.



6. Click **OK** to save your settings, and **OK** again, to dismiss the ODBC Data Source Administrator.
7. Run the ODBC load.
8. Locate the log file that was generated, and send it to ThoughtSpot Support with a description of the problem.

Enable ODBC logs on a Linux workstation

To enable logging on Linux, follow these instructions:

1. Navigate to the directory where you installed ODBC.
2. Open the `odbc.ini` file in a text editor.

   This file is the registry and configuration file for ODBC.
3. Locate the `LogLevel` and `LogPath` properties.
4. Uncomment the properties.
5. Enter a value for the `LogLevel` the.

   Acceptable values are from 1 to 6 with 6 being the most verbose.
6. Enter the fully qualified path for the `LogPath` values.

   The log will be written here. Your file will look similar to the following: Example for Linux 64-bit:

```
[ThoughtSpot]
Description = ThoughtSpot 64-bit ODBC Driver
Driver = ThoughtSpot
ServerList =  172.18.231.17 12345
Locale = en-US
ErrorMessagesPath = /home/admin/linux/ErrorMessages
UseSsl = 0
#SSLCertFile = # Set the SSL certificate file path. The
certificate file can be obtained by extracting the SDK t
arball
LogLevel = 3 # Set log level to enable debug logging
LogPath = /home/admin/odbc-logs # Set the debug log file
s path
DATABASE = # Set the default database to connect to
SCHEMA = # Set the default schema to connect to
```

7. Save and close the file.
8. To test the configuration, run the ODBC load and review the log files.

Control logs from the Simba server

You may want to collect logs from the Simba service. Do the following to procedure on every ThoughtSpot node running the Simba service.

1. SSH into the ThoughtSpot node.
2. Edit the `/etc/thoughtspot/linux.ini` file.

```
 ...
 [Driver]

 ## Note that this default DriverManagerEncoding of UT
F-32 is for iODBC. unixODBC uses UTF-16 by default.
 ## If unixODBC was compiled with -DSQL_WCHART_CONVERT,
then UTF-32 is the correct value.
 ## Execute 'odbc_config --cflags' to determine if you n
eed UTF-32 or UTF-16 on unixODBC
 DriverManagerEncoding=UTF-32
 DriverLocale=en-US
 ErrorMessagesPath=/usr/home/linux/ErrorMessages/
 LogLevel=0
 LogNamespace=
 LogPath=

 ....
```

3. Uncomment the `LogLevel` setting.

   The `LogLevel` is the level of logging to capture (0-6).

4. Set `LogPath` to a directory to save the logs.

   The `LogPath` is the fully qualified path where ThoughtSpot should write the logs.

5. Work with ThoughtSpot Support to restart the Simba serivce.

   The node IP may change because of the restart. If this happens, repeat the entire procedure.

Enable JDBC Logs

Summary: Configure logging parameter strings.

To enable logging for JDBC, add the logging parameters to the connect string. Logs are stored on ThoughtSpot. Before enabling JDBC logging, you need:

- The level of logging you want to capture.
- The path on the ThoughtSpot server where the logs will be written. Make sure the directory has the correct permissions so that the “admin” Linux user can write logs to it.

To enable JDBC logging:

1. When forming the connect string for JDBC, add these two parameter, separated by “&”:

   For example:

   ```
    jdbc:simba://192.168.2.248:12345;SERVERS=192.168.2.24
   9:12345,
    192.168.2.247:12345;Database=test;Schema=falcon_defaul
   t_schema;**LogLevel=3;LogPath=/usr/local/scaligent/log
   s**
   ```

   The `LogLevel` is the level of logging to capture (0-6). The `LogPath` is the fully qualified path where logs will be written on ThoughtSpot.

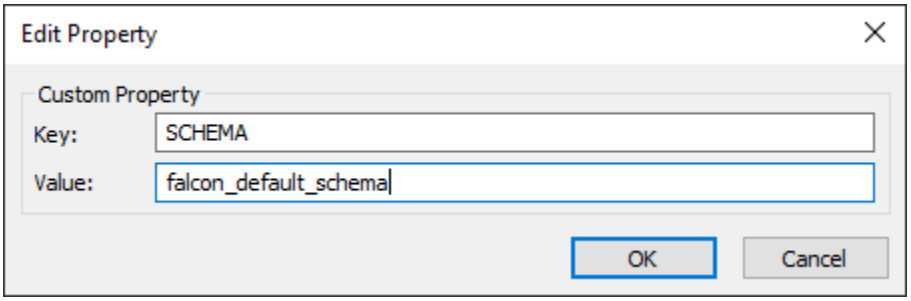
2. Run the JDBC code that uses the connection you modified.
3. Check the `LogPath` directory for logs generated by JDBC.

Schema not found error with ODBC

> **Summary:** Correct schema not found errors.

When connecting with ODBC, you need to specify both the `DATABASE` and `SCHEMA` parameters. This is true even if you do not use schema names in ThoughtSpot. If you don't supply a `SCHEMA`, you get an error indicating that the schema could not be found.

The default schema name in ThoughtSpot is `falcon_default_schema`. To set the `SCHEMA` on Windows, adding a custom property with the key `SCHEMA` and the value `falcon_default_schema`.



On Linux, you can edit the properties in the `odbc.ini` file for the driver you are using:

```
[ThoughtSpot]
Description = ThoughtSpot 64-bit ODBC Driver
Driver = ThoughtSpot
ServerList =  172.18.231.17 12345
Locale = en-US
ErrorMessagesPath = /home/admin/linux/ErrorMessages
UseSsl = 0
#SSLCertFile = # Set the SSL certificate file path. The certifi
cate file can be obtained by extracting the SDK tarball
#LogLevel = 0 # Set log level to enable debug logging
#LogPath = # Set the debug log files path
DATABASE = # Set the default database to connect to
SCHEMA = # Set the default schema to connect to
```

Related information

- Configuring ODBC on Windows
- Configuring ODBC on LINUX
- ODBC and JDBC configuration properties

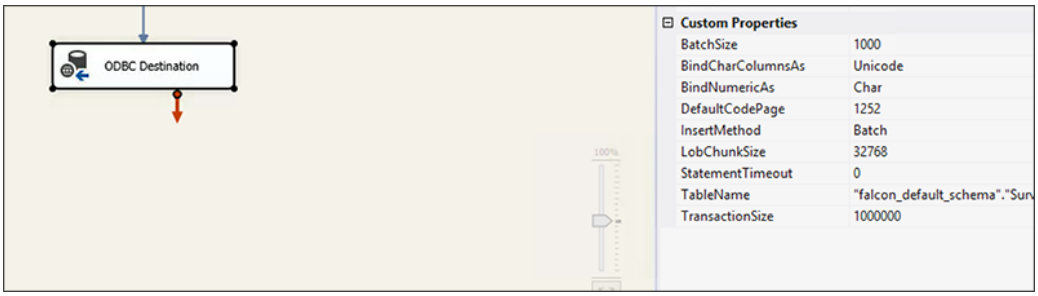
How to improve throughput

> **Summary:** Adjusting the transaction size may correct poor performance and low throughput.

The transaction/commit size value can improve the throughput of the load when setting up the ODBC Driver.

Adjusting the transaction size may correct poor performance and low throughput issues. The transaction size should be set to match the total number of rows that are expected to be loaded in the load cycle. However, increasing this value even higher should help improve throughput of the load.

> ❗ **Warning:** A high transaction size may slow down the ThoughtSpot system.



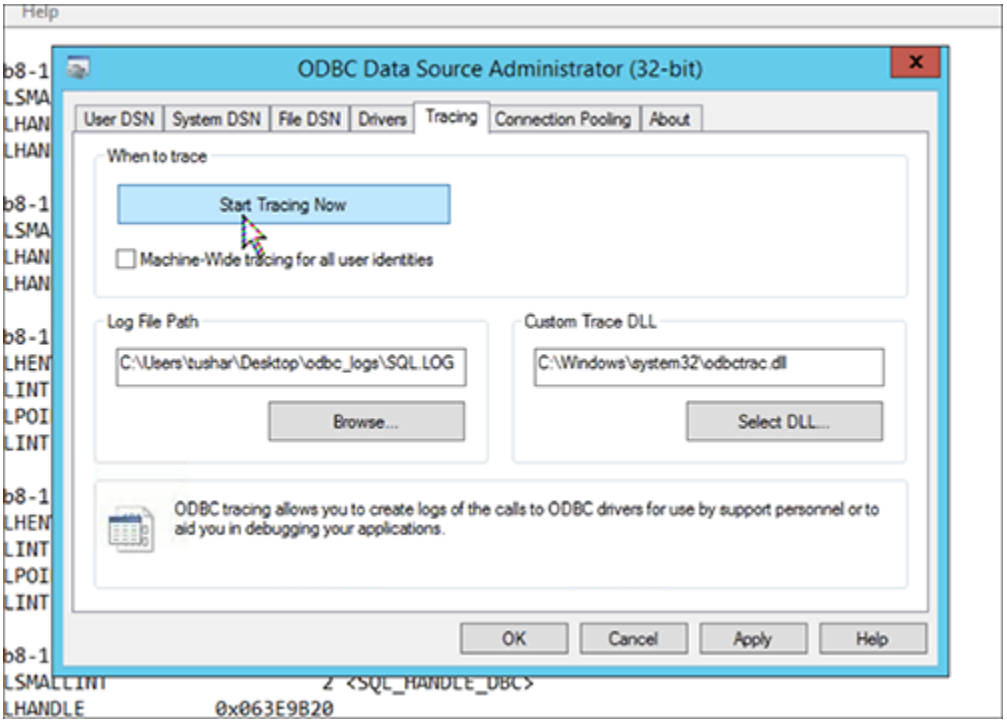
This is where the transaction size field exists for SSIS. Clicking on the ODBC destination reveals the properties on the right hand side, where the **Transaction Size** can be found.

See Set up the ODBC Driver for SSIS for more details on setting the transaction size.

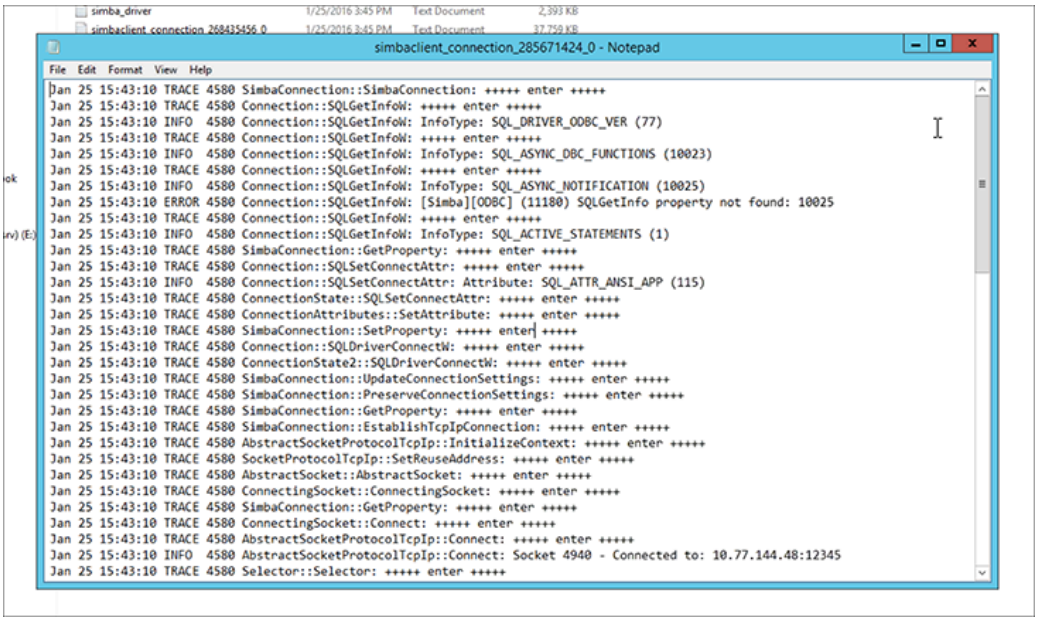
ODBC tracing on Windows

Summary: Using logs to aid in troubleshooting.

Windows shows ODBC specific tracing in the ODBC Data Source Administrator Tracing tab. You can start tracing there by clicking **Start Tracing Now**. This logs every ODBC call from this system, and prints the input and output for the call.



Although this is lower level information, it can still be helpful in troubleshooting. When you are not sure if it is our driver or the tool causing an issue, doing this trace will help narrow the inquiry.



If you start or stop tracing, make sure you do not have the SSIS client open. Close it, change the trace, and reopen.

Supported SQL commands

> **Summary:** The ThoughtSpot connection drivers support a limited set of SQL commands.

The ODBC and JDBC drivers support a limited set of SQL commands. When developing software that uses a ThoughtSpot ODBC driver, use this reference of supported commands. This reference is intended for developers using other tools (ETL, etc.) to connect to ThoughtSpot through the ODBC or JDBC driver.

> **Note:** ThoughtSpot displays `VARCHAR` fields using lower case, regardless of what the original casing of your loaded data is.

ODBC

These SQL commands are supported for ODBC:

- `CREATE TABLE`

  Creates a table with the specified column definitions and constraints. The table is replicated on each node.

  ```
  CREATE TABLE country_dim (id_number int, country varch
  ar, CONSTRAINT PRIMARY KEY (id_number));
  ```

- `INSERT`

  Creates placeholders in the table to receive the data.

  ```
  INSERT INTO TABLE country_dim (?, ?);
  ```

- `DELETE FROM <table>`

Deletes `ALL` rows from the specified table. Use the `WHERE` clause to specify only certain rows to be deleted. Example: You could remove all data for sales before a certain date to free up space in ThoughtSpot.

```
  DELETE FROM country_dim;
```

- `SELECT <cols_or_expression> FROM <table_list> [WHERE <predicates>] [GROUP BY <expressions>] [ORDER BY <expressions>]`

  Fetches the specified set of table data.

```
  SELECT id_number, country FROM country_dim WHERE id_nu
mber > 200;
```

JDBC

`TRUNCATE` is not supported. Instead, use `DELETE FROM TABLE` which is functionally equivalent to “truncate table” in terms of table compression and so forth.

Connection configuration

> **Summary:** Lists the properties you can set for ODBC or JDBC connections

This section lists the properties you can set for ODBC or JDBC connections.

Setting Properties for ODBC

Not all the parameters Simba accepts are supported by the ThoughtSpot ODBC clients, and ThoughtSpot has added some properties, which are listed separately here. All configuration properties use the type String (text).

You can set these properties on Windows by using the ODBC Administrator client. For Linux, the properties are located in three files, depending on the property type:

Property Type	Location
DSN	`odbc.ini` file
Driver	`odbsinst.ini` file
SimbaSetting Reader	`simbaclient.ini` file

Setting Properties for JDBC

For JDBC, these properties are passed as key value pairs in the connect string. For more information, see Use the JDBC Driver.

Properties Reference

The following tables summarize the configuration properties.

Property	Type	Description
DATABASE	DSN or Driver	The default database to connect to.
SCHEMA	DSN or Driver	The default schema to connect to.
Description	DSN	A brief, human-readable description of the DSN. This describes the DSN to users who are deciding which DSN to use.
Driver	DSN or Driver	In the driver configuration location, Driver should contain the path to the driver binary. In the DSN configuration location, Driver could contain the path to the driver binary, or it could contain the driver entry in the registry.
IdleTimeout	DSN	The time to wait for a response from the server, in seconds. This property is optional, but SimbaClient will wait indefinitely for SimbaServer to respond to a request made to the server unless you specify a timeout period. IdleTimeout specifies how many seconds that SimbaClient will wait before aborting the attempt and returning to the application with an error. This timeout corresponds to ODBC's CONNECTION_TIMEOUT property and is only used when more specific timeouts, such as QUERY_TIMEOUT or LOGIN_TIMEOUT aren't applicable.
Locale	DSN	The connection locale. If this value is set, it overrides the driver-wide locale. For example, the driver-wide locale could be en-US. If the client would prefer fr-CA, it can set the connection locale to fr-CA. Values are composed of a 2-letter language code (in lower case), and an optional 2-letter country code (in upper case). If the country code is specified, it must be separated from the language code by a hyphen (-).
LoginTimeout	DSN	The timeout, in seconds, to wait for a response from the server when attempting to log in. A value of 0 means no timeout. The default value is 60.
QueryTimeout	DSN	The timeout, in seconds, to wait for a response from the server during Prepare, Execute, and ExecuteDirect. A value of 0 means no timeout. The default value is 60.
ServerList	DSN	A comma separated list of all servers (IP address and port number) to connect to. SimbaClient must be able to find SimbaServer on the network. This property enables server discovery. SimbaClient will try to make a network connection to the servers in the order specified until a connection is made.
LogLevel	SimbaSettingReader	Controls the granularity of the messages and events that are logged. With this keyword, you can control the amount of log output by controlling the kinds of events that are logged. Possible values (case sensitive): • 0 or LOG_OFF : no logging occurs • 1 or LOG_FATAL : only log fatal errors • 2 or LOG_ERROR : log all errors • 3 or LOG_WARNING : log all errors and warnings • 4 or LOG_INFO : log all errors, warnings, and informational messages • 5 or LOG_DEBUG : log method entry and exit points and parameter values for debugging • 6 or LOG_TRACE : log all method entry points

Property	Type	Description
`LogPath`	SimbaSetting Reader	Specifies the directory where the log files are created. For example: `LogPath=C:\Simba Technologies\Temp` If this value is not set, the log files are written to the current working directory of the SimbaClient.
`LogFileSize`	SimbaSetting Reader	The size of each log file, in bytes. The default values is 20971520 bytes. When the maximum size of the file is reached, a new file is created.
`LogFileCount`	SimbaSetting Reader	The number of log files to create. When the maximum number of log files has been created, the oldest file will be deleted and a new one created. The default value is 50.
`username`	UID	Part of a user username/password combination. This combination should correspond to a ThoughtSpot application user with permissions appropriate to your ETL requirements. Typically, this user is a user with data management or administrative privileges on the application.
`password`	Password	Part of a user username/password combination. This combination should correspond to a ThoughtSpot application user with permissions appropriate to your ETL requirements. Typically, this user is a user with data management or administrative privileges on the application.

ThoughtSpot in Practice

Summary: This guide demonstrates the power of ThoughtSpot to solve real solutions we developed for our clients.

The purpose of this section is to guide you through a few solutions we created for our clients, so you can leverage our experience to quickly and confidently employ ThoughtSport in meeting your own business objectives.

Each topic and scenario includes a real-world data modeling problem, and how we solved it with ThoughtSpot technology.

Reaggregation scenarios in practice

Summary: We provide real world scenarios for using flexible aggregation in ThoughtSpot.

The following scenarios showcase the use of the `group_aggregate` function in the real world. We provide them to demonstrate to you how the function works, and the scenarios where it proved useful.

Best practices for flexible aggregations

The `group_aggregate` function enables you to calculate a result at a specific aggregation level, and then returns it at a different aggregation level. For this reaggregation result to return correctly, follow these syntax guidelines:

- Wrap `group_aggregate` in an aggregate function, such as `sum` or `average`
- The wrapping function must be the immediate preceding function, such as `sum(group_aggregate(...))`
- Do not use with conditional operators. For example, the following expression does not reaggregate the data because the `if` precedes `group_aggregate`:

```
(if(group_aggregate(...)))
```

Scenario 1: Supplier tendering by job

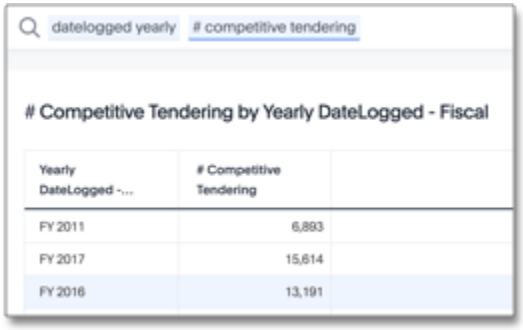
We have a fact table at a job or supplier tender response aggregation level. There are many rows for each job, where each row is a single row from a supplier. A competitive tender is a situation when multiple suppliers bid on the same job.

Our objective is to determine what percentage of jobs had more than 1 supplier response. We want to see high numbers, which indicate that many suppliers bid on the job, to we can select the best response.

Valid solution

A valid query that meets our objective may look something like this:

```
sum(group_aggregate(if(sum(# trades tendered ) > 1) then 1 els
e 0,
                    query_groups() + {claimid, packageid},
                    query_filters()))
```



Resolution

1. The `sum ( # trades tendered )` function aggregates to these attributes:
    - `{claimid, packageid}`

      The job-level identifier
    - `query_groups( )`

      Adds any additional columns in the search to this aggregation. Here, this is the `datelogged` column at the yearly level.
    - `query_filters ( )`

      Applies any filters entered in the search. Here, there are no filters.

2. For each row in this virtual table, the conditional `if() then else` function applies. So, if the sum of tendered responses is greater than 1, then the result returns 1, or else it returns 0.

3. The outer function, `sum()`, reaggregates the final output as a single row for each `datelogged` yearly value.

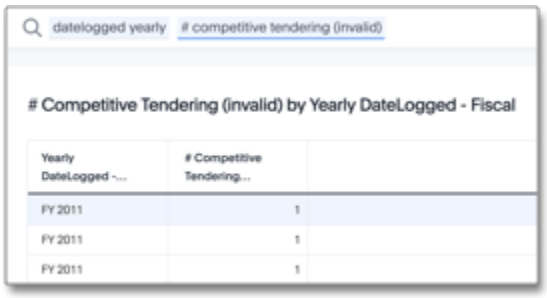
    - This reaggregation is possible because the conditional statement is inside the `group_aggregate` function.
    - Rather than return a row for each `{claimid,packageid}`, the function returns a single row for `datelogged yearly`.
    - The default aggregation setting does not reaggregate the result set.

Non-Aggregated Result

We include the following result to provide contrast to an example where ThoughtSpot does not reaggregate the result set. Reaggregation requires the aggregate function, `sum`, to precede the `group_aggregate` function.

In the following scenario, the next statement is the conditional `if` clause. Because of this, the overall expression does not reaggregate. The returned result is a row for each `{claimid,packageid}`.

```
sum(if(group_aggregate (sum (# trades tendered),
                              query_groups() + {claimid, packag
eid},
                              query_filters ( ) )>1) then 1 els
e 0)
```



Scenario 2: Average rates of exchange

The Average rate of exchange calculates for the selected period. These average rates provide a mechanism to hedge the value of loans against price fluctuations in the selected period. We apply the average rate *after the aggregation*.

The pseudo-logic that governs the value of loans is `sum(loans) * average(rate)`.

The data model has two tables: a primary fact table, and a dimension table for `rates`.

- The `loans` column is from the primary fact table.
- The `rate` column is from the `rates` table.

These tables are at different levels of aggregation:

- The primary fact table uses a lower level of aggregation, on `product`, `department`, or `customer`.
- The `rates` dimension table use a higher level of aggregation, on `daily`, `transaction currency`, or `reporting currency`.

The two tables are joined through a relationship join on `date` and `transaction currency`.

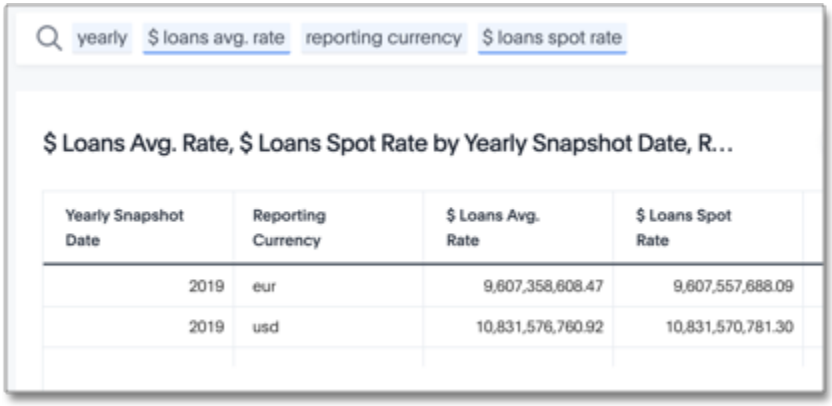
To simplify the scenario, we only use a single `reporting currency`. The join ensures that a single rate value returns each day for each transaction currency.

Valid solution

A valid query that meets our objective may look something like this:

```
sum(group_aggregate (sum(loans)*average (rate),
                     query_groups () + {transaction_currency},
                     query_filters () ))
```

The following search and resulting response returns the dollar value for each year, for each target reporting currency. Note that the dataset contains both euro (€) and US dollars ($). The `$ Loans Avg. Rate` calculates the average rate of exchange for the entire period. The `$ Loans Spot Rate` applies the rate of exchange on the day of the transaction.



Resolution

1. The **`sum(loans)`** function aggregates to these attributes:
    - **`{transaction_currency}`** and **`query_groups()`**

      Add additional search columns to this aggregation. Here, this at the level of `reporting currency` and `year`.
    - **`query_filters( )`**

      Applies any filters entered in the search. Here, there are no filters.

2. Similarly, the **`average(rate)`** function aggregates to these attributes:
    - **`{transaction_currency}`** and **`query_groups()`**

      Add additional search columns to this aggregation. Here, this at the level of `reporting currency` and `year`.
    - **`query_filters( )`**

      Applies any filters entered in the search. Here, there are no filters.

3. For each row in this virtual table, the exchange rate applies to the sum of loans: `sum(loans) * average(rate)`.

4. The outer `sum()` function reaggregates the final output as a single row for each yearly reporting currency value.

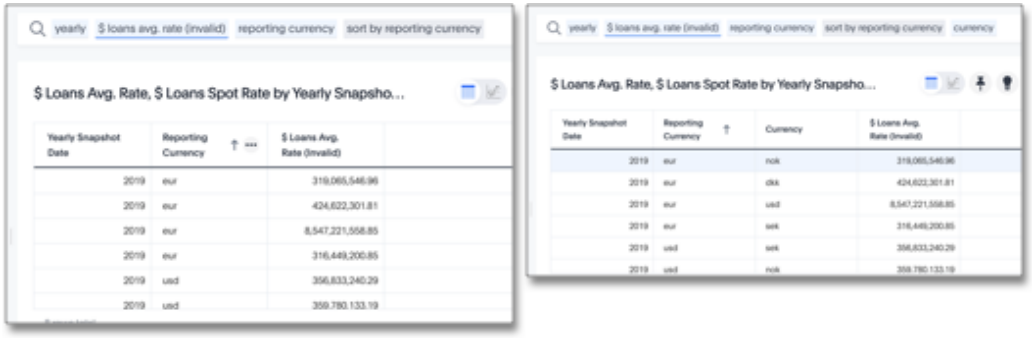
   Note that the default aggregation setting does not reaggregate the result set.

Non-Aggregated Result

We include the following result to provide contrast to an example where ThoughtSpot does not reaggregate the result set. Reaggregation requires the aggregate function, `sum`, to precede the `group_aggregate` function.

In the following scenario, the formula assumes that the default aggregation applies. Here, the result returns 1 row for each `transaction currency`.

```
group_aggregate (sum(loans )*average (rate ),
                 query_groups() + {transaction_currency},
                 query_filters())
```



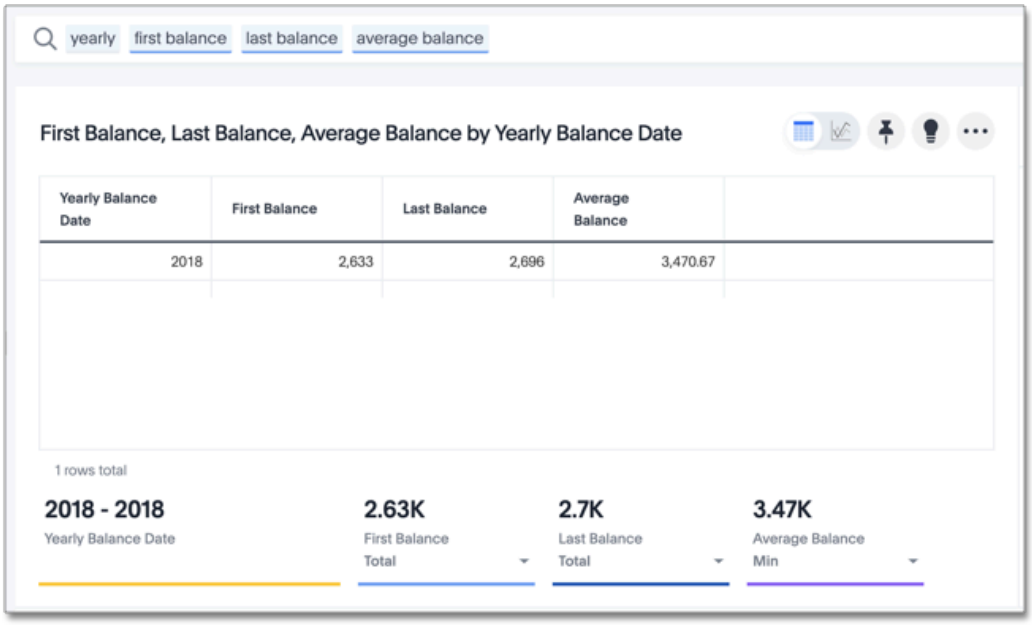
Scenario 3: Average period value for semi-additive numbers I

Semi-additive numbers may be aggregated across some, but not all, dimensions. They commonly apply to specific time positions. In this scenario, we have daily position values for home loans, and therefore cannot aggregate on the date dimension.

Valid solution

A valid query that meets our objective may look something like this:

```
average(group_aggregate(sum(loan balance),
                        query_groups() + {date(balance date)},
                        query_filters()))
```



Resolution

1. The `sum(loan balance)` function aggregates to the following attributes:
    - `{date(balance date)}` and `query_groups()`

      Add additional search columns to this aggregation. Here, this at the `yearly` level.
    - `query_filters ( )`

      Applies any filters entered in the search. Here, there are no filters.

2. The `sum(loan balance)` function returns a result for each row in this virtual table.

3. The outer `average()` function reaggregates the final output as a single row for each `year` value.

Scenario 4: Average period value for semi-additive numbers II

Semi-additive numbers may be aggregated across some, but not all, dimensions. They commonly apply to specific time positions. In this scenario, we have daily position values for home loans, and therefore cannot aggregate on the date dimension.

Here, we consider a somewhat different situation than in Scenario 3. In some financial circumstances, the average daily balance has to be calculated, even if the balance does not exist. For example, if a banking account was opened on the 15th of June, business requirements have to consider all the days in the same month, starting with the 1st of June. Importantly, we cannot add these 'missing' data rows to the data set; note that the solution used in Scenario 3 returns an average only for the period that has data, such as June 15th to 30th, not for the entire month of June. The challenge is to ensure that in the daily average formula, the denominator returns the total days in the selected period, not just the days that have transactions:

```
sum(loans) / sum(days_in_period)
```

To solve for this, consider the data model:

- The fact table `transactions` reports the daily position for each account, and uses a `loan` column.
- The dimension table `date` tracks information for each date, starting with the very first transaction, all the way through the most recent transaction. This table includes the expected `date` column, and `days_in_period` column that has a value of 1 in each row.
- Worksheets use the `date` column with keywords such as *weekly*, *monthly*, *yearly* to change the selected period.
- When users run a search with the *monthly* keyword, the denominator must reflect the number of days in each month.

Valid solution

A valid query that meets our objective may look something like this:

The following code *in the denominator definition* returns the total number of days for the period, regardless whether there are transactions, or what filters apply:

```
group_aggregate (sum(days_in_period),{Date},{})
```

Resolution

1. The **`sum(days_in_period)`** function aggregates to:
    - **`{Date}`**

      No other search columns appear.
    - `{}`

      We require the entire period, so there are no filters.

    Note that the `date` keywords *yearly*, *quarterly*, *monthly*, and *weekly* apply because we use the same column in both the search and the aggregation function. So, the function will result in the following output when it runs with the *yearly* keyword in search:

Year	Result
2016	366
2017	365
2018	365
2019	365
2020	366

2. This data is not reaggregated because we want to return the result at the appropriate `date` level.

Alternate Solution

To return only the number of days that have existing transactions, use the following code in the denominator:

```
sum(days_in_period)
```