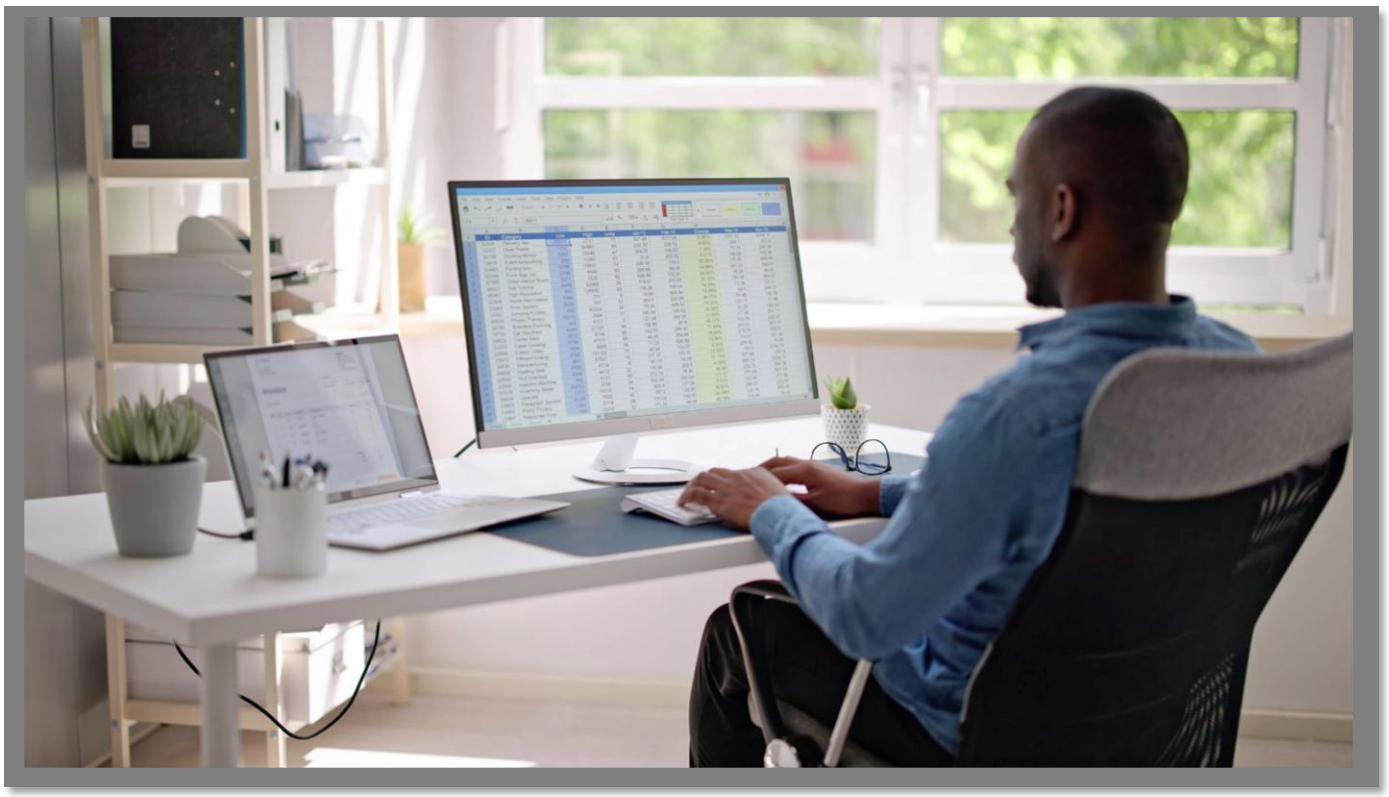
Arbutus Connectors

Google BigQuery
CONFIGURATION GUIDE



ARBUTUS
Powerful Analytics Simplified

Contents

Arbutus Connector – Google BigQuery

A. Introduction

The purpose of this Guide is to provide assistance with configuring the Arbutus Google BigQuery Connector using the ODBC Data Source Administrator. The configuration process can involve several technical steps that require a good understanding of IT systems and database management.

To make the most of this guide, it's advisable to have a good understanding of database connectivity, driver installation, and system settings. The ODBC Data Source Administrator, which is used as part of the configuration process, allows for the setup and management of data sources, enabling applications to access data from various database systems.

Due to the complexity and potential impact of these configurations, it is recommended that only those individuals with IT or database expertise undertake this task. In addition, it should also be understood that each client's network environment is different. A one-size-fits-all approach is rarely effective, as what works well in one environment may not be suitable in another.

B. About Google BigQuery

Databricks is a cloud-based platform for big data processing and analytics, integrating with Apache Spark to simplify building, managing, and scaling data pipelines, machine learning models, and analytics workflows. It is widely used for data engineering, data science, and AI tasks, with collaborative features and seamless integration with cloud services like AWS, Azure, and Google Cloud.

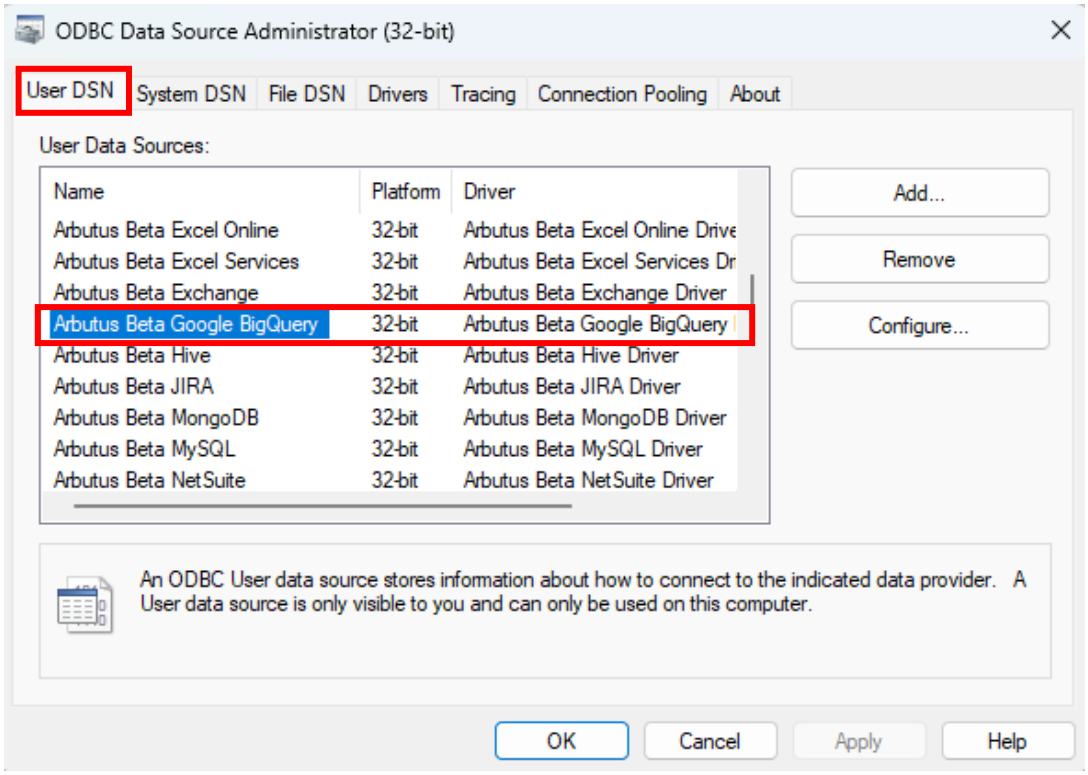
In Databricks, data is stored in cloud storage (AWS, Azure, or Google Cloud) and supports both structured and unstructured data. Databases can be defined to organize tables, where data is stored and queried. While Databricks is not a traditional relational database, it supports relational-like operations using SQL and Spark SQL for creating tables, defining schemas, and running queries.

C. Determining if the Connector exists prior to configuring

Installation of the Arbutus Google BigQuery Connector is done at the time of installing the Arbutus software. For more information on this, please see the **Overview Guide Document**.

Once the Connector has been installed, the next step is to configure it.

Prior to configuring it, you can check to see if the Connector has been installed by opening the **32-bit ODBC Data Source Administrator**, pictured below, and clicking the **User DSN** tab. Included below is information on how you can access the **ODBC Data Source Administrator**.



- If the Arbutus Google BigQuery Connector appears in the list, it can be considered as installed.
- If it is not listed, it is likely that you did not select it during the installation or modification of the Arbutus software. In this case, it is recommended to reinstall the Arbutus software and choose the **Modify** option when prompted. For more details, please refer to the **Overview Guide Document**.

Below is the file path to access and run the **ODBC Data Source Administrator** application:

C:\Windows\SysWOW64\odbcad32.exe

Alternative, you can also try locating and opening the **ODBC Data Source Administrator** application by doing a search on your desktop application.

D. Configuring the Connector after it has been installed

Once you have verified that the Arbutus Connector has been installed, it is time to configure it.

This process is done using the **ODBC Data Source Administrator**. It can be described as "**editing the DSN configuration"**.

DSN, Drivers, and Data Sources
What is a DSN? DSN stands for Data Source Name, and is a unique name used to create a data connection to a database using open database connectivity (ODBC). A DSN is a data structure that contains the information required to connect to a database. It is essentially a string that identifies the source database, including the driver details, the database name, and often authentication credentials and other necessary connection parameters. DSNs facilitate a standardized method for applications to access databases without needing hard-coded connection details, enhancing flexibility and scalability in database management.

Arbutus Connectors

- ***Drivers*** are the components that process ODBC requests and return data to the application. If necessary, drivers modify an application's request into a form that is understood by the data source. The **Drivers** tab in the **ODBC Data Source Administrator** dialog box lists all drivers installed on your computer, including the name, version, company, file name, and file creation date of each driver.

- ***Data sources*** are the databases of files accessed by a driver and are identified by a data source name (DSN). You use the ODBC Data Source Administrator to add, configure, and delete data sources from your system.

All ODBC connections require that a DSN be configured to support the connection. When a client application wants to access an ODBC-compliant database, it references the database using the DSN.

The types of DSNs are:

- **User DSN** – User DSNs are local to a computer and can be used only by the current user. They are registered in the HKEY_Current_USER registry subtree.

- **System DSN** – System DSNs are local to a computer rather than dedicated to a user. The system or any user with privileges can use a data source set up with a system DSN. System DSNs are registered in the HKEY_LOCAL_MACHINE registry subtree.

- **File DSN** – File DSNs are file-based sources that can be shared among all users who have the same drivers installed and therefore have access to the database. These data sources need not be dedicated to a user nor be local to a computer. File data source names are identified by a file name with a .dsn extension.

User and system data sources are collectively known as *machine* data sources because they are local to a computer.

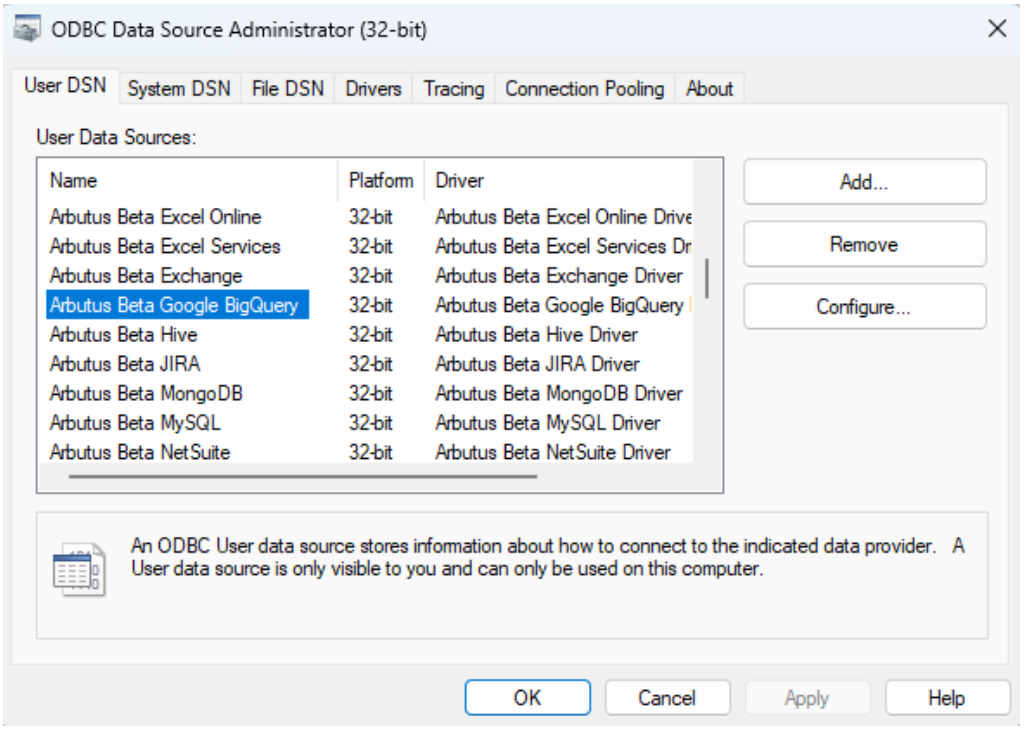
Each of these DSNs has a tab in the **ODBC Data Source Administrator** dialog.

The Arbutus ODBC Driver for Google BigQuery enables real-time access to Google BigQuery data, directly from any applications that support ODBC connectivity, the most widely supported interface for connecting applications with data.

Arbutus Connectors

Follow these steps to edit the DSN configuration and configure the Connector.

1. First open the **ODBC Data Source Administrator**.



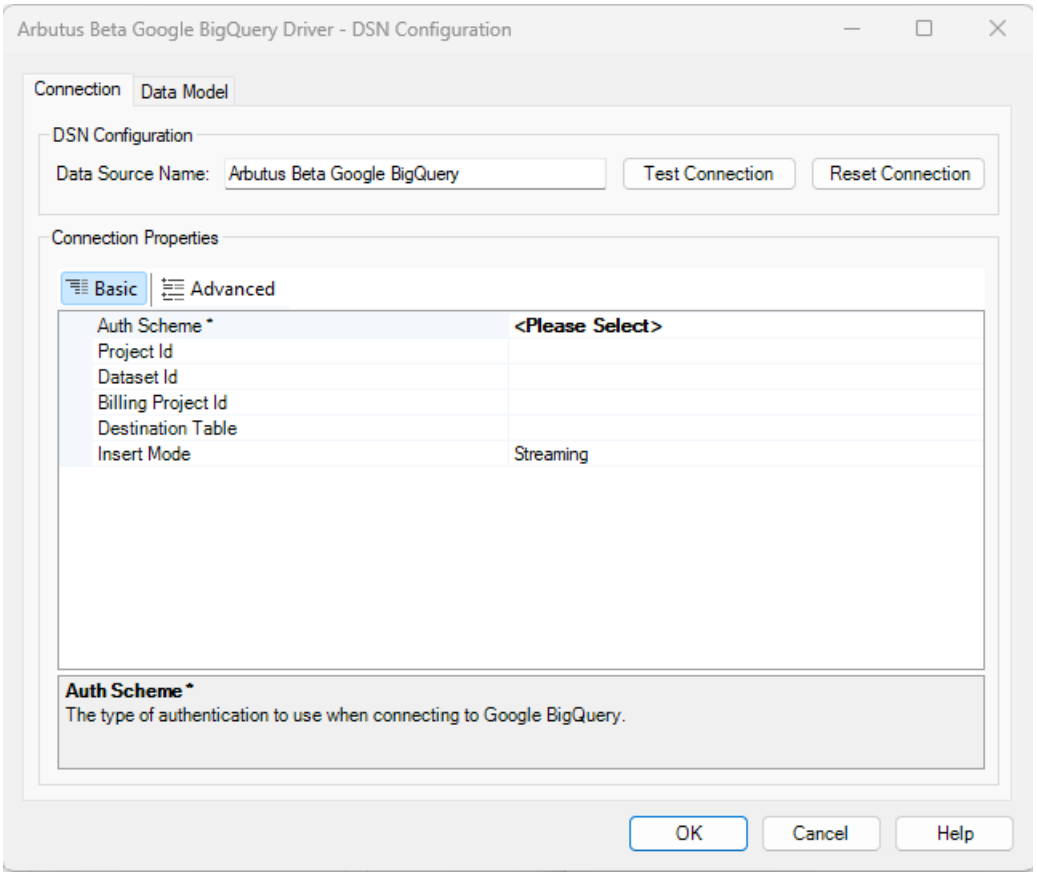
2. Click the **User DSN** tab.

   Selected data connectors are installed as **User DSN's** in Window's 32 Bit **ODBC Data Source Administrator**.

   Also, each of the data connector's names is prefaced with Arbutus, for example, **Arbutus Google BigQuery.**

3. Select the Arbutus Connector, in this case it is **Arbutus Google BigQuery**.
4. Click **Configure**.

This opens the **Arbutus Google BigQuery Driver – DSN Configuration** dialog.



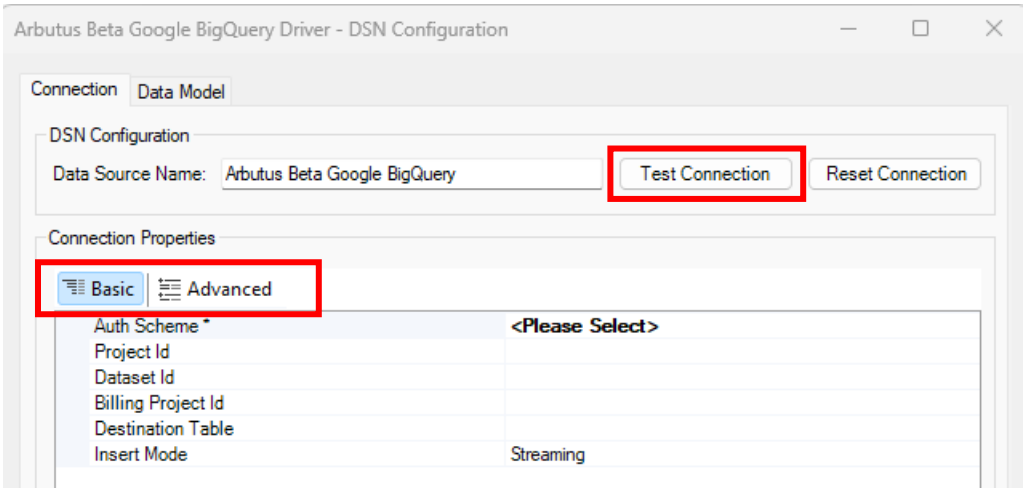
E. Editing the DSN properties – the Basic and Advanced tabs

With the DSN Configuration dialog open, the next step is to edit the DSN properties, where necessary, in the **Basic** and **Advanced** tabs. For example, editing the **Auth Scheme properties** (per screenshot below) to ensure correct authentication to the server is applied.

E1. Editing the DSN properties in the Basic tab

The properties listed in the **Basic** tab are typically the ones that are most commonly used, and as such are designed to be more user-friendly and straightforward, allowing you to quickly make changes without needing in-depth technical knowledge.

Once you have completed editing the properties in the **Basic** tab, you can go ahead and try testing the connection to the Google BigQuery system by clicking the **Test Connection** button, as highlighted in the screenshot below.



In the **Basic** tab, there are **six** main properties to review:

1. **Auth Scheme –** click the dropdown to select from the list the appropriate type of authentication to use when connecting to Google BigQuery. The options available for selection are as follows:
   - OAuth – select this to perform OAuth authentication using a **standard user account** to gain access to Google services like BigQuery.

   - OAuth JWT (JSON Web Token) – select this to perform OAuth authentication using an **OAuth service account**.

     You would select **OAuth JWT (JSON Web Token)** to perform authentication when you need a **secure, token-based authentication method** for **server-to-server** communication, particularly in scenarios where traditional OAuth authentication isn't sufficient or practical.

It provides a more secure, stateless authentication method. It involves creating a **signed JSON Web Token (JWT)** that proves the identity of the sender. This makes it more reliable and secure for sensitive applications or services where high security is required.

Selecting **OAuth JWT** requires you to specify the following:

- OAuth JWT Cert – this is the JWT Certificate store for the client certificate.

  The **OAuth JWT Cert Type** field (see below) specifies the type of the certificate store specified by **OAuth JWT Cert**. If the store is password protected, specify the password in **OAuth JWT Cert Password** (see below)

  OAuth JWT Cert is used in conjunction with the **OAuth JWT Cert Subject** field (see below) in order to specify client certificates. If **OAuth JWT Cert** has a value, and **OAuth JWT Cert Subject** is set, a search for a certificate is initiated. Please refer to the **OAuth JWT Cert Subject** field for details.

  Designations of certificate stores are platform-dependent.

  The following are designations of the most common User and Machine certificate stores in Windows:

MY	A certificate store holding personal certificates with their associated private keys
CA	Certifying authority certificates
ROOT	Root certificates
SPC	Software publisher certificates

  In Java, the certificate store normally is a file containing certificates and optional private keys.

When the certificate store type is PFXFile, this property must be set to the name of the file. When the type is PFXBlob, the property must be set to the binary contents of a PFX file (i.e. PKCS12 certificate store).

- OAuth JWT Cert Type – select from the dropdown list the type of key store containing the JWT Certificate.

  The options available for selection are:

  - USER
  - MACHINE
  - PFXFILE
  - PFXBLOB
  - JKSFILE
  - JKSBLOB
  - PEMKEY_FILE
  - PEMKEY_BLOB
  - PUBLIC_KEY_FILE
  - PUBLIC_KEY_BLOB
  - SSH PUBLIC_KEY_FILE
  - SSH PUBLIC_KEY_BLOB
  - P7BFILE
  - PPKFILE
  - XMLFILE
  - XMLBLOB
  - GOOGLEJSON
  - GOOGLEJSONBLOB

  The default value is **GOOGLEJSON**.

  If required, more information on this property, including the descriptions for each of the key stores listed above, can be provided.

- OAuth JWT Subject – this is the user subject for which the application is requesting delegated access. Enter the email address of the user for which the application is requesting delegated access.

- GCPInstanceAccount – select this to get Access Token from Google Cloud Platform (GCP) instance to authenticate and access Google services.

GCPInstanceAccount is used for **server-to-server authentication** on Google Cloud instances, where an application or service running on a GCP instance needs to authenticate automatically to interact with other Google Cloud services, e.g., BigQuery.

2. **Project Id** – this is the Project Id used to resolve unqualified tables and execute jobs.

   This property and **Billing Project Id** (see #4 below) are used to determine billing for jobs and resolve unqualified table names.

   The driver must create a job within Google BigQuery to execute certain kinds of queries. For example, complex SELECT statements, UPDATE and DELETE statements, and INSERT statements (when **Insert Mode** (see #6 below) is **DML**) are all executed using jobs. The project where a job executes determines how the job is billed.

   Job execution: The driver determines the billing project using these rules. Note that only the first two rules apply when **Query Pass through** (in the **Miscellaneous** section of the **Advanced** tab of the **DSN Configuration** dialog) is enabled. Either this property or **Billing Project Id** must be set to execute passthrough queries.
   a. The **Billing Project Id** is used if that property is not empty
   b. Then this property is used
   c. If both properties are empty, the project is determined from the catalog of the first table in the query. The job created for the following query executes in the **psychic-valve-137816** project.

      SELECT FirstName, LastName FROM `psychic-valve-137816`.`Northwind`.`customers`

Table Resolution: In addition to setting the billing project, the driver also uses this property to determine the default data project. The data project is used to resolve tables included in queries when they are not fully qualified.

/ Unqualified, resolved against connection properties */*
SELECT FirstName, LastName FROM `Northwind`.`customers`

/ Qualified, project specified as catalog */*
SELECT FirstName, LastName FROM `psychic-valve-137816`.`Northwind`.`customers`

Any unqualified table references in the query are resolved using the following rules. Note that only methods 1 and 2 are supported when **Query Pass through** is enabled. This means that any tables outside the default data project must be explicitly qualified.

a. This property is used if it is not empty
b. Then the **Billing Project Id** property is used
c. If both properties are empty, the catalog from the first table in the query is used. In the following query, the 'Northwind', 'Orders' table is treated as if it comes from the **psychic-valve-137186** project.

SELECT … FROM `psychic-valve-137816`.`Northwind`.`customers`
INNER JOIN `Northwind`.`orders`
ON …

3. **Dataset Id** – this is the Dataset Id used to resolve unqualified tables.

When a query refers to a table it can leave the dataset implicit, or qualify the dataset directly as the schema portion of the table:

/ Implicit, resolved against connection string */*
SELECT FirstName, LastName FROM `customers`

/ Explicit, dataset specified as schema */*
SELECT FirstName, LastName FROM `psychic-valve-137816`.`Northwind`.`customers`

Any unqualified table references in the query are resolved using the following rules. Note that only method 1 is supported when **Query Pass through** is enabled. This means that passthrough queries must set this property or qualify all tables.

a. If this property is set, then the specified dataset is used
b. Otherwise, the schema from the first table in the query is used. In the following query the `orders` table is treated as if it comes from the Northwind dataset

SELECT ... FROM `psychic-valve-137816`.`Northwind`.`customers`
INNER JOIN `orders`
ON ...

4. **Billing Project Id** – this is the Project Id of the billing project for executing jobs. This property is used with **Project Id** (see #2 above) to determine the project the driver executes jobs under. Please refer to that section for more information.

5. **Destination Table** – this is to determine where query results are stored in Google BigQuery.

   Google BigQuery queries have a maximum amount of data they are allowed to return directly. If this limit is exceeded, then queries will fail with an error message like *Response too large to return*. When this option is enabled the response limit does not apply, because all query responses are stored in a Google BigQuery table before being returned.

This option is set differently depending upon whether your connection is using **Use Legacy SQL** or not. By default this option is set using the standard SQL syntax:

DestinationTable=project-name.dataset-name.table-name

When **Use Legacy SQL** is enabled, this option is set using the legacy table syntax:

DestinationTable=project-name:dataset-name.table-name

When using this option with multiple connections, make sure that each connection has its own destination table. Sharing a table between connections can lead to results getting lost because parallel queries can overwrite each others results.

6. **Insert Mode** – this is a dropdown selection to specify what kind of method to use when inserting data. The options available for selection are:
   - Streaming – this uses the Google BigQuery streaming API (also called **insert All**). The **Streaming** API is intended for use where the most important factor is being able to insert quickly. However, rows which are inserted via the API are queued and only appear in the table after a delay. Sometimes this delay can be as high as 20-30 minutes which makes this API incompatible with cases where you want to insert data and then run other operations on it immediately. You should avoid modifying the table while any rows are in the streaming queue: Google BigQuery prevents DML operations from running on the table while any rows are in the streaming queue, and changing the table's metadata (name, schema, etc.) may cause streamed rows that haven't been committed to be lost.

- DML – this uses the Google BigQuery query API to generate INSERT SQL statements which insert individual rows. The **DML** mode API uses Standard SQL INSERT queries to upload data. This is by the most robust method of uploading data because any errors in the uploaded rows will be reported immediately. The driver also uses this API in a synchronous way so once the INSERT is processed, any rows can be used by other operations without waiting. However, it is by far the slowest insert method and should only be used for small data volumes.

- Upload – this uses the Google BigQuery upload API to create a load job which copies the rows from temporary server-side storage. The **Upload** mode uses the multipart upload API for uploading data. This method is intended for performing low-cost medium to large data loads within a reasonable time. When using this mode the driver will upload the inserted rows to Google-managed storage and then create a load job for them. This job will execute and the driver can either wait for it (see **Wait For Batch Results** in the **Uploading** section of the **Advanced** tab of the **DSN Configuration** dialog) or let it run asynchronously. Waiting for the job will report any errors that the job encounters but will take more time. Determining if the job failed without waiting for it requires manually checking the job status via the job stored procedures.

- GCSStaging – this is similar to the Upload mode except that it uses your Google Cloud Storage account instead of public storage. The **GCSStaging** mode is the same as Upload except that it uses your Google Cloud Storage account to store staged data instead of Google-managed storage. The driver cannot act asynchronously in this mode because it must delete the file after the load is complete, which means that **Wait For Batch Results** (in the **Uploading** section of the **Advanced** tab of the DSN Configuration dialog) has no effect.

Because this depends on external data, you must set the **GCS Bucket** (in the **Uploading** section of the **Advanced** tab of the **DSN Configuration** dialog) to the name of your bucket and ensure that **Scope** (in the **OAuth** section of the **Advanced** tab of the **DSN Configuration** dialog), a space delimited set of scopes, contains at least the scopes:
https://www.googleapis.com/auth/bigquery and
https://www.googleapis.com/auth/devstorage.read_write

The devstorage scope used for GCS also requires that you connect using a service account because Google BigQuery does not allow user accounts to use this scope.

When **Use Legacy SQL** (in the **BigQuery** section of the **Advanced** tab of the **DSN Configuration** dialog) is only Streaming and Upload modes are allowed. The Legacy SQL dialect does not support DML statements.

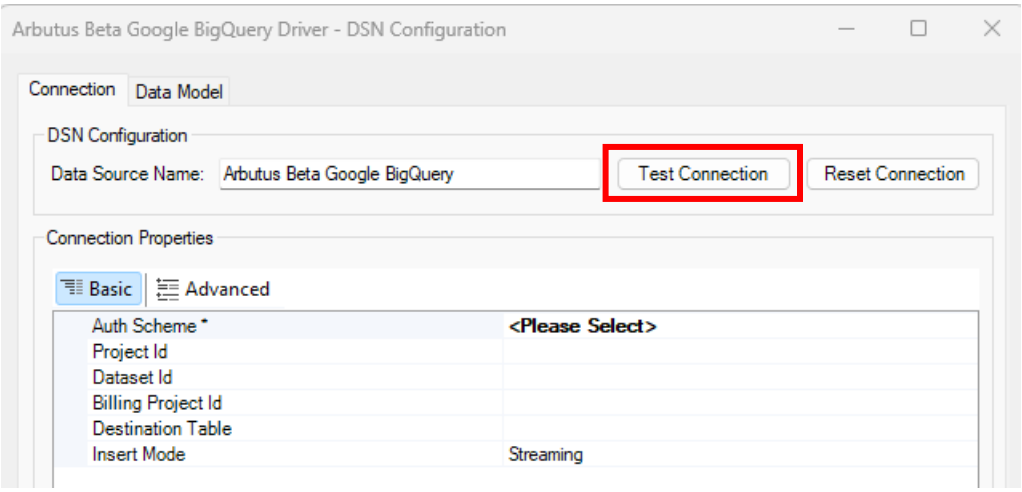
If required, more information on these properties and their settings can be provided. Detail driver settings may be needed in advanced integrations.

The default value is **Streaming**.

E2. Editing the DSN properties in the Advanced tab

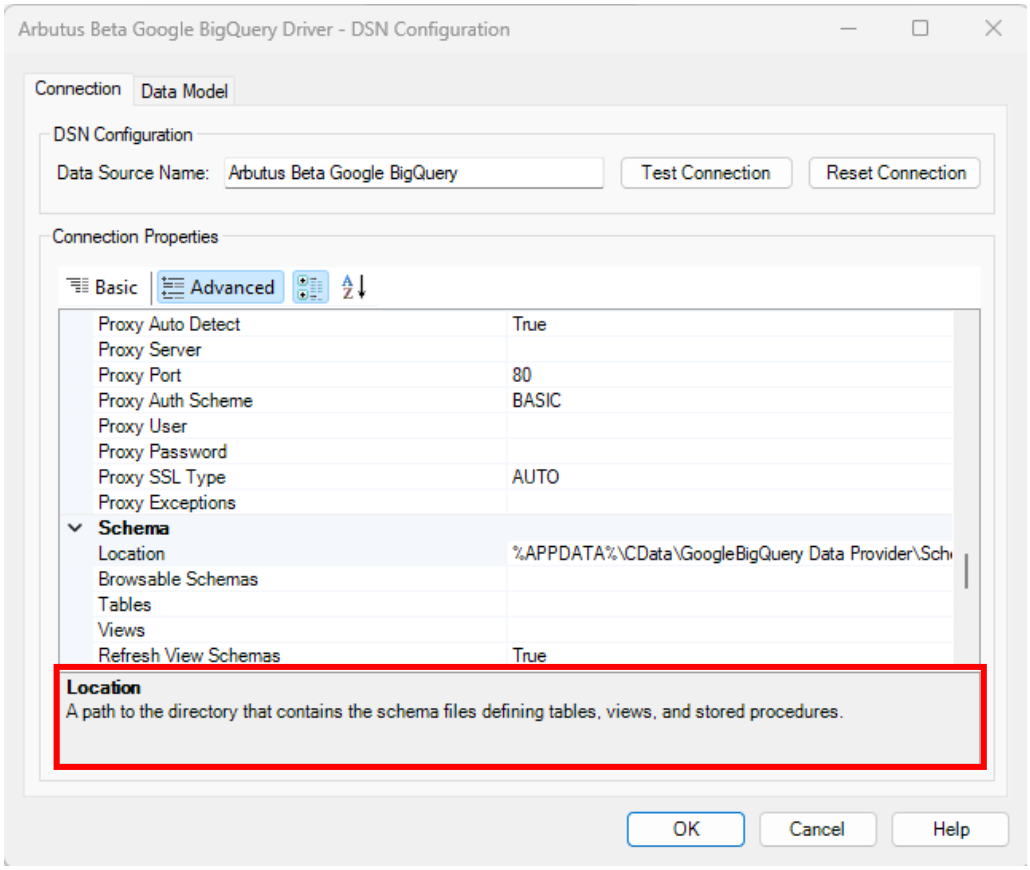
This tab includes more detailed and technical properties. It is intended for those users who need more control over the configuration and are comfortable with more complex options. The **Advanced** tab often includes properties that can fine-tune the behaviour of the system feature.

If you have already completed editing the properties in the **Basic** tab, as required, you do not necessarily need to also complete editing the properties in the **Advanced** tab. Instead, once you have completed editing the properties in the **Basic** tab, you may opt to proceed to testing the connection to the Google BigQuery system by clicking the **Test Connection** button.



There are a lot more properties included for editing in the **Advanced** tab.

However, it is useful to know that each property does provide a short description of it and as such serves as a guide in terms of what to edit and/or enter. This short description can be seen at the bottom of the **DSN Configuration** dialog box, as seen in the screenshot below.



If it is deemed necessary to complete some/all the properties in the **Advanced** tab, it is recommended that you refer to the description shown for any of the properties being edited and/or entered.

If required, more information on the properties listed in the **Advanced** tab can also be provided.

F. Other questions and/or request for assistance

There may be times when you need to consult with the technical support team at Arbutus Software. If so, please send an email request to support@ArbutusSoftware.com.

For more information, please refer to the CONTACT US page on our website.