



# USER GUIDE

BETA 2.0.0

## Contents

What is Snowflake Data Integrity?.....	2
Useful Links .....	2
SNOWDay88 Data Integrity Check Process.....	3
The SNOWDay88 Graphical User Interface (GUI) .....	3
Database Constraints Options .....	4
Connection Options .....	6
Logging Options .....	8
Load and Save Options.....	9
Actions .....	10
The SNOWDay88 Command Line Interface .....	11
Filename Variables.....	14
Console Application Usage Examples .....	14

## What is Snowflake Data Integrity?

Snowflake is a powerful, advanced Software-as-a-Service (SaaS) data platform that enables data storage, processing and analytic solutions. Snowflake is designed for performance, and to that end, constraints that could adversely affect performance are not enforced by the Snowflake database engine. To ensure data integrity, Snowflake recommends that you perform validations on your data after it is loaded. Until now, doing so required writing custom code to test the integrity of all the data in your database; and when your database structures changed, custom code updates were required to accommodate the changes.

Dynamic data integrity is an optimized process by which we can validate the data in your database automatically based on the primary key, unique key, and foreign key constraints that you define using Snowflake's standard SQL syntax. Dynamic data integrity automatically adjusts itself based on changes in your database. If you add new tables, remove tables, or change the definitions of your constraints, dynamic quality validation automatically adjusts its analysis to your new database schema.

SNOWDay88 bridges the Snowflake data quality gap. It uses your primary key, unique key, and foreign key constraint definitions to dynamically validate the integrity of your data after you load it. SNOWDay88 also has the ability to verify adherence to best practices for constraint declarations.

SNOWDay88 is designed to make efficient use of network bandwidth and cloud CPU resources to perform its functions, including granular logging capability, and can output results in any of several formats designed for simple manual review or automated processing. SNOWDay88 comes with an easy-to-use graphical user interface (GUI) and a full-featured command-line interface that can be run using an enterprise scheduler for regular scheduling of data integrity checks against your most critical data.

## Useful Links

[Installation](#) | [FAQs](#)

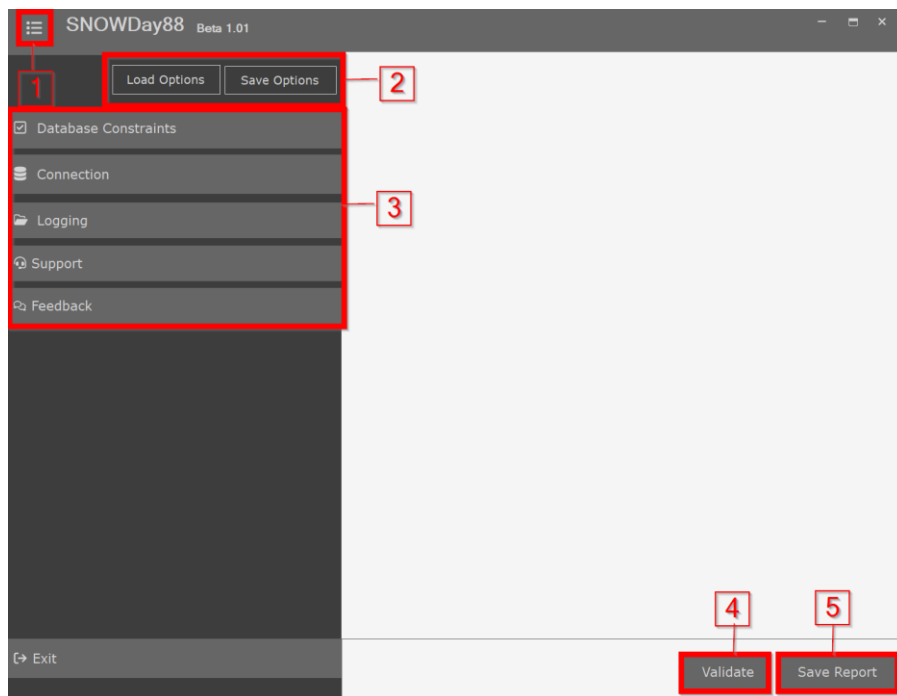
## SNOWDay88 Data Integrity Check Process

By regularly checking the quality of your data, you can ensure the results you generate, and the information you distribute to end users and decision makers, is as accurate and error-free as possible. The results may also lead to the discovery of previously unknown data loading process issue that can introduce errors into your data. The style of reporting provided by the SNOWDay88 makes it easier to identify and address data quality issues at the most granular level of your data.

The SNOWDay88 process is straightforward. Simply run SNOWDay88 against your Snowflake database and it will perform the data integrity checks based on the constraints you have defined in your database. Then review the results of the analysis and update the data in your database appropriately to resolve the issues reported. The results can be output in any of several human-readable formats or in formats optimized for automated processing.

## The SNOWDay88 Graphical User Interface (GUI)

The SNOWDay88 GUI is designed for compactness and simplicity. The startup screen is shown below.



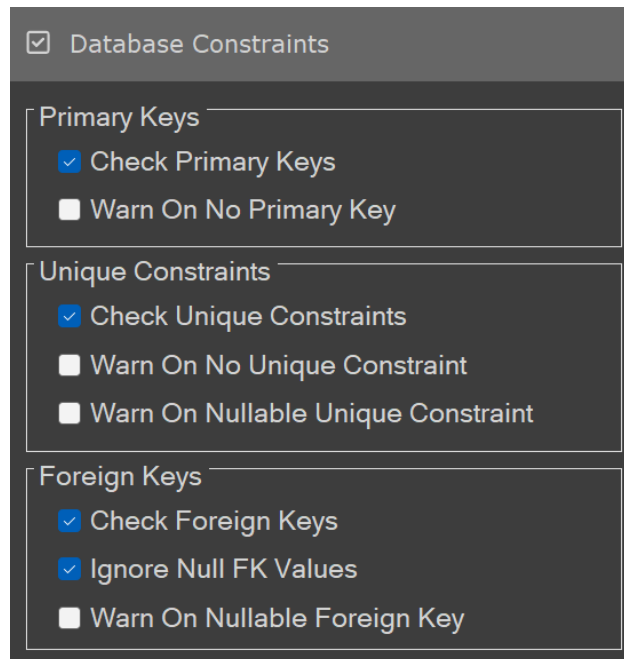
SNOWDay88 GUI

The major components of the launch screen are:

- 1 Flyout menu button:** Shows or hides the flyout menu on the left side of the screen.
- 2 Load/Save Options buttons:** Load from options file or save current configuration.
- 3 Flyout menu:** Contains application functions which are described in detail below.
- 4 Validate button:** The validate button starts the data quality validation process.
- 5 Save Report button:** The save report button saves your validation report in Tab-delimited format.

### Database Constraints Options

Clicking on the Database Constraints header reveals the Database Constraints section options.



*SNOWDay88 Database Constraints Options*

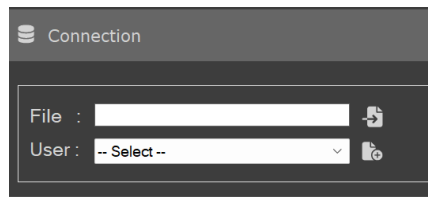
This section has the options that control the validations performed by SNOWDay88. The available options are:

- **Check Primary Keys:** When turned on, this option checks for duplicates in tables that have Primary Keys defined on them. If duplicate Primary Key entries are found in a given table, the duplicates are reported as errors in the validation report.
- **Warn On No Primary Key:** When checked, this option will generate a warning in the validation report for any table that does not have a Primary Key defined on it. Defining Primary Keys on tables is considered a best practice. This option is only available when **Check Primary Keys** is turned on.
- **Check Unique Constraints:** When turned on, this option checks for duplicates in tables that have Unique Constraints defined on them. If duplicate Unique Constraint entries are found in a given table, the duplicates are reported as errors in the validation report.
- **Warn On No Unique Constraint:** When turned on, this option generates a warning in the validation report for any table that does not have a Unique Constraint defined on it. Defining Unique Constraints on tables is considered a best practice when your Primary Key is used as a surrogate key. This option is available only when **Check Unique Constraints** is turned on.
- **Warn on Nullable Unique Constraint:** When turned on, this option generates a warning in the validation report for any table that has a Unique Constraint that includes nullable columns. Some organizations have standards that indicate Unique Constraints should only be defined on non-nullable columns. This option is available only when **Check Unique Constraints** is turned on.
- **Check Foreign Keys:** When turned on, this option checks all your Foreign Keys to ensure that they do not contain invalid values (values that do not exist in the tables they reference). If any invalid reference values are identified, they are reported as errors in the validation report.
- **Ignore NULL FK Values:** When turned on, this option ignores mismatches caused by NULL values in your Foreign Key columns. If you have a Foreign Key that contains nullable columns, NULL values in those columns won't join to the referenced table without special handling (using the COALESCE function in JOINS, for instance). This option is only available when **Check Foreign Keys** is turned on.

- **Warn on Nullable Foreign Key:** When turned on, this option generates a warning when a Foreign Key that contains a nullable column in its definition is encountered. In some database architectures (data marts, for instance), best practice is to only use non-nullable columns in Foreign Key definitions. This option is only available when **Check Foreign Keys** is turned on.

## Connection Options

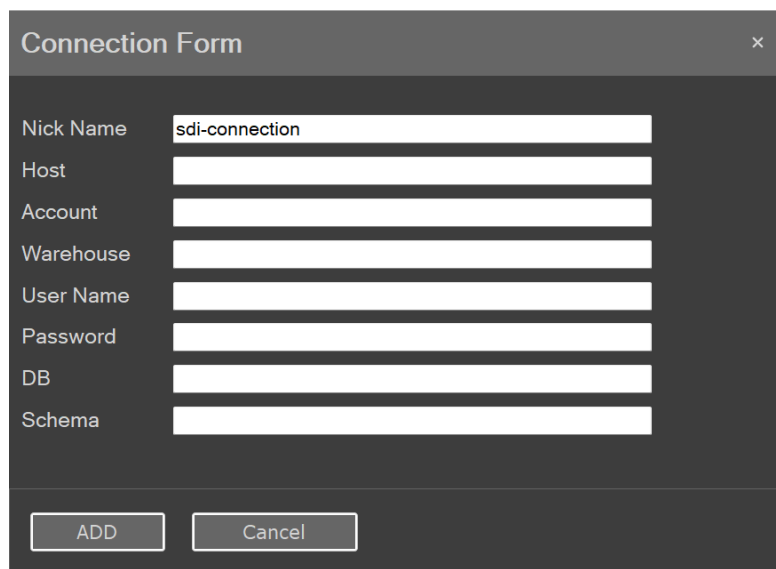
Clicking on the Connection header reveals the options for connecting to your database:



*SNOWDay88 Connection Option*

The Connection option GUI allows you to create and save new users for easily connecting to your databases. The connection information is saved in an encrypted connection file in C:\Users\USER\AppData\Roaming\SNOWDay88. This folder will need to be manually deleted when uninstalling the program. When moving to a new computer you can copy this folder to retain access to your users.

Clicking the  button opens up the connection form to easily add a new user.


A screenshot of a 'Connection Form' dialog box. It has a title bar with the text 'Connection Form' and a close button. The form contains several input fields: 'Nick Name' (pre-filled with 'sdi-connection'), 'Host', 'Account', 'Warehouse', 'User Name', 'Password', 'DB', and 'Schema'. At the bottom, there are two buttons: 'ADD' and 'Cancel'.

*Connection Form*

When a user is selected the new user button will turn into an edit button, this will open up the Connection Form to quickly modify user information.



There is also a delete button to permanently delete the user. This cannot be undone. Alternatively you can browse to the APPDATA folder and move the user files you wish to delete to the recycle bin.

Clicking the  button shows a file open dialog, this will allow you to select a JSON-formatted connection file and fill in the Connection Form fields automatically. You can then edit the user before saving it.

There is a sample connection file included in the installation, but it is not populated yet. You can edit the file in a text editor to insert the appropriate values to connect to your server. Below is a sample file:

```
{
  "HOST": "zzz12345.us-east-1.snowflakecomputing.com",
  "ACCOUNT": "zzz12345",
  "WAREHOUSE": "COMPUTE_WH",
  "USER": "my_username",
  "PASSWORD": "my_password",
  "DB": "MY_DATABASE",
  "SCHEMA": "MY_SCHEMA",
  "AUTHENTICATOR": "SNOWFLAKE"
}
```

### *Sample JSON Connection File*

The sample connection file shown has no usable connection data in it; attempting to connect using these parameters will fail. You should change the parameters in your connection file using the following descriptions as a guide:

- **HOST:** The HOST parameter is your Snowflake host information.
- **ACCOUNT:** The ACCOUNT parameter is your Snowflake account ID. Generally, this is the same as the first part of your HOST parameter.
- **WAREHOUSE:** The WAREHOUSE parameter is the Snowflake warehouse that you want to use to perform processing.
- **USER:** The USER parameter is your login username.

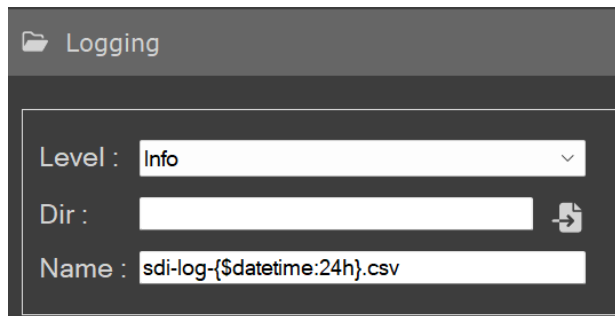


- **PASSWORD:** The PASSWORD parameter is your login password.
- **DB:** The DB parameter is the name of the database you want to connect to and validate.
- **SCHEMA:** The SCHEMA parameter is the name of the default schema to connect to. Though you specify a default schema for connection and processing purposes, tables in all schemas in the database are validated.
- **AUTHENTICATOR:** The AUTHENTICATOR parameter specifies the method of authentication. The default value of SNOWFLAKE indicates you are using the standard Snowflake username/password authentication method.

You can create multiple Connection files if you want to validate multiple databases from the GUI.

### Logging Options

Clicking on the Logging header reveals the logging options:



*SNOWDay88 Logging Options*

From this menu, you can choose one of the following runtime **Logging Levels**:

- **Critical:** Reports only critical errors, which stop execution of the application.
- **Error:** Reports all errors, including critical errors.
- **Warning:** Reports all warnings, errors, and critical errors.
- **Info:** Reports all informational messages, warnings, errors, and critical errors.
- **Debug:** Reports all informational messages, warnings, errors, critical errors, and additional debugging information.
- **None:** Turns all logging off.

The **Dir** option allows you to select the directory where the log will be written. Clicking the ... button shows a directory selection dialog, allowing you to change the directory.

The **File** option allows you to define the output filename for your log file. You can use the following variables in your filename:

- **{\$datetime}** or **{\$datetime:24h}** inserts the current date/timestamp in the format *YYYY-MM-DD-hh-mm-ss*, using a 24-hour clock.
- **{\$datetime:12h}** inserts the current date/timestamp in the format *YYYY-MM-DD-hh-mm-ss-[A/P]*. The [A/P] indicates AM/PM. This designator is culture-specific, and may appear as a different character for depending on your computer's language settings.
- **{\$date}** inserts the current date in *YYYY-MM-DD* format.
- **{\$time}** or **{\$time:24h}** inserts the current time in *hh-mm-ss* format, using a 24-hour clock.
- **{\$time:12h}** inserts the current time in *hh-mm-ss-[A/P]* format. The [A/P] indicates AM/PM. This designator is culture-specific.
- **{\$guid}** or **{\$uuid}** inserts a randomly generated GUID (globally unique identifier) string.
- **{\$machine}** inserts the local machine name.
- **{\$user}** inserts the current username.
- **{\$domain}** inserts the current domain name.

### Load and Save Options

The Load Options button allows you to load a JSON-formatted Options file (.opt extension). Load Options allows you to load all the options in the Options flyout menu from a file.

Below is a sample Options file:

```
{
  "conn-file": "C:\\sdq\\sdq-connection.con",
  "check-pk-flag": true,
  "check-uc-flag": true,
  "check-fk-flag": true,
  "warn-no-pk-flag": true,
  "warn-no-uc-flag": true,
  "warn-nullable-uc-flag": true,
  "ignore-null-fk-flag": true,
  "warn-nullable-fk-flag": true,
  "log-file": "C:\\Windows\\Temp\\SDQ-LOG-{$datetime}.txt",
  "log-format": "Tab",
  "logging-level": "Debug"
}
```

### Sample Options File Contents

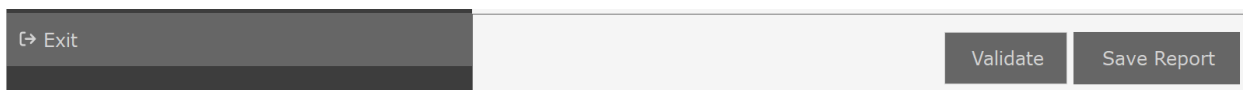
The parameters in the Options file are:

- **conn-file:** The full path to the JSON-formatted .con connection file.
- **check-pk-flag:** The Check Primary Keys option flag, set to true or false.
- **check-uc-flag:** The Check Unique Constraints option flag, set to true or false.
- **check-fk-flag:** The Check Foreign Keys option flag, set to true or false.
- **warn-no-pk-flag:** Logs a warning when a table is encountered with no primary key. Value can be true or false.
- **warn-no-uc-flag:** Logs a warning when a table is encountered with no unique constraint. Value can be true or false.
- **warn-nullable-fk-flag:** Logs a warning when a nullable foreign key, one that contains a nullable column, is encountered. Value can be set to true or false.
- **warn-nullable-uc-flag:** Logs a warning when a nullable unique constraint, one that contains a nullable column, is encountered. Value can be set to true or false.
- **ignore-null-fk-flag:** Ignores null values encountered when validating foreign key constraint relationships. Value can be set to true or false.
- **log-file:** Sets the full path and filename for the log file.
- **logging-level:** The logging level. Possible values are CRITICAL, ERROR, WARNING, INFO, DEBUG, and NONE.
- **log-format:** Sets the log file format. The default log file format is set to tab-delimited format for the GUI application. *This setting has no effect and is reserved for future use.*

The Save Options button saves your current SNOWDay88 GUI options settings to an Options file.

#### Actions

The GUI has three Action buttons, shown below.



GUI Action Buttons

- **Validate:** The Validate button connects to your Snowflake database and performs all the dynamic validations based on your options.
- **Save Report:** The Save Report button is disabled until after your validation is complete. Once the validation report is generated it is displayed in the GUI and the Save Report button is enabled. The Save Report action allows you to save your results to a tab-delimited results text file.
- **Exit:** The Exit button closes the GUI and exits the application.

### Filtering Options

From the GUI, SNOWDay88 provides basic filtering options for schema and table when viewing your data. If you have a large dataset and require more advanced filtering, we recommend exporting to a flat file and opening it in Microsoft Excel or your data software of choice.



*SNOWDay88 filtering options*

### The SNOWDay88 Command Line Interface

The SNOWDay88 command line interface can be run directly from the Windows command line through the sdi-cmd.exe utility. This option allows you to run data quality validations in a script or batch file, or via an enterprise scheduler of your choice.

When running SNOWDay88 from the command line, you configure the application via command-line parameters. The following parameters are available:

Command / Option	Required?	Purpose
<b>connect</b>	<b>Yes</b>	<b>Defines connection to Snowflake database</b>
--file <filename> -f <filename>	Yes	Specifies the full path to JSON Snowflake connection file. Variables are not allowed in this filename.
<b>check</b>	<b>No</b>	<b>Defines validation check</b>

<pre>--ignore [pk , fc , uc] -i [pk , fc , uc]</pre>	No	Ignore validation of specific constraint types. Can be any combination of <b>PK</b> (primary keys), <b>FK</b> (foreign keys), and <b>UC</b> (unique constraints). By default, all constraint types are validated.
<pre>--missing [pk , uc] -m [pk , uc]</pre>	No	Report on missing <b>PK</b> (primary keys) and <b>UC</b> (unique constraints). By default, missing PK and UC constraints are not reported.
<pre>--warn [fk , uc] -w [fk , uc]</pre>	No	Report on nullable <b>FK</b> (foreign keys) and <b>UC</b> (unique constraints). A nullable constraint is any constraint that has one or more nullable columns in its definition. By definition, PK constraints cannot be nullable. The default is that nullable FK and UC constraints are not reported.
<pre>--null-ignore -n</pre>	No	Ignore nulls in foreign key constraint data. NULLs that exist in foreign key column data will not join to primary key tables without special handling in queries.
<b>output</b>	<b>Yes</b>	<b>Defines report output file name and format</b>
<pre>--file &lt;filename&gt; -f &lt;filename&gt;</pre>	Yes	Specifies the full path to the report output file. Variables can be used in this filename (see below).
<pre>--format [csv   tab   pipe   xml   json] -r [csv   tab   pipe   xml   json]</pre>	No	Defines the format for the output report to one of the following: <ul style="list-style-type: none"> <li>• <b>csv</b>: comma-separated values text file</li> <li>• <b>tab</b>: tab-delimited values text file</li> <li>• <b>pipe</b>: pipe-delimited values text file</li> <li>• <b>xml</b>: xml file</li> <li>• <b>json</b>: JSON file</li> </ul> <p>The default value is <b>tab</b>.</p>
<b>log</b>	<b>No</b>	<b>Defines the log file output destination and format.</b>

<pre>--file &lt;filename&gt; -f &lt;filename&gt;</pre>	<p>No</p>	<p>Specifies the full path to the log file. If not provided, the default is <code>.\SNOWDay88-log.log</code>. Variables can be used in this filename (see below).</p>
<pre>--format [csv   tab            pipe   xml   json] -f [csv   tab      pipe   xml   json]</pre>	<p>No</p>	<p>Indicates the logging file format. Can be one of the following:</p> <ul style="list-style-type: none"> <li>• <b>csv</b>: comma-separated values text file</li> <li>• <b>tab</b>: tab-delimited values text file</li> <li>• <b>pipe</b>: pipe-delimited values text file</li> <li>• <b>xml</b>: xml file</li> <li>• <b>json</b>: JSON file</li> </ul> <p>The default value is <b>tab</b>.</p>
<pre>--level [error   info            critical   warning            debug   none] --l [error   info       critical   warning       debug   none]</pre>	<p>No</p>	<p>Sets the logging level. Can be one of the following:</p> <ul style="list-style-type: none"> <li>• <b>critical</b>: Reports only critical errors, which stop execution of the application.</li> <li>• <b>error</b>: Reports all errors, including critical errors.</li> <li>• <b>warning</b>: Reports all warnings, errors, and critical errors.</li> <li>• <b>info</b>: Reports all informational messages, warnings, errors, and critical errors.</li> <li>• <b>debug</b>: Reports all informational messages, warnings, errors, critical errors, and additional debugging information.</li> <li>• <b>none</b>: Turns all logging off.</li> </ul> <p>The default is <b>none</b>.</p>
<pre>--help -h -?</pre>	<p>No</p>	<p><b>Displays the console help screen.</b></p>
<pre>--version -v</pre>	<p>No</p>	<p><b>Displays the application version information.</b></p>

If any required parameters are missing from the command line, or invalid values are passed in, when the application is called, an error message is returned, and the console application will terminate.

### Filename Variables

Filenames may contain variables, delimited by curly braces and preceded by a \$ symbol (“\${...}”). These variables are replaced with specified system values at runtime to form complete filenames. The following variables are available for use in the output and log filenames.

Variable	System Value
{ \$datetime} { \$datetime:24h}	The current system timestamp in YYYYMMDD_hhmmss format, based on a 24-hour clock. Example: 20220716_160105.
{ \$datetime:12h}	The current system timestamp in YYYYMMDD_hhmmssX format, based on a 12-hour clock. The X indicator is “A” for AM or “P” for PM. Example: 20220716_040105P.
{ \$date}	The current system date in YYYYMMDD format. Example: 20220716.
{ \$time} { \$time:24h}	The current system time in hhmmss format, based on a 24-hour clock. Example: 160105.
{ \$time:12h}	The current system time in hhmmssX format, based on a 12-hour clock. Example: 040105P.
{ \$guid} { \$uuid}	A system generated globally unique identifier (GUID), an alphanumeric identifier with 36 characters including hyphens. Example: 8aaa34af-a051-4a02-bd60-ea692014dd0c
{ \$machine}	The current server or workstation computer name. Example: APPSERVER01
{ \$user}	The current user name. Example: SVCACCOUNT18
{ \$domain}	The current user’s domain. Example: CORPDOMAIN6
{ \$userdir}	The current user’s user profile directory. Example: C:\USERS\LJOHNSON
{ \$appdata}	The current user’s application data directory. Example: C:\USERS\LJOHNSON\APPDATA\ROAMING

### Console Application Usage Examples

The console application can be called with many different commands and options. Here is the most basic possible usage:

```
sdi-cmd.exe CONNECT -f .\ sdi-connection.con  
OUTPUT -f .\sdi-report.txt
```

This will call the sdi-cmd console application specifying the database connection and the output file name. All other options are defaulted in this case.

A more complex example shows how to add logging and set additional options:

```
sdi-cmd.exe CONNECT -f .\sdi-connection.con  
OUTPUT -f .\results.json -r json  
CHECK -i "fk,uc"  
LOG -f .\sdi-log.csv -r csv -l debug
```

This example specifies the database connection file and sets the report output to JSON format. The analysis will ignore foreign key and unique constraints, focusing only on primary keys because of the ignore (-i option) in the CHECK command. The logging is set to debug mode and output in CSV format.

The previous example can be modified to include variables in the output file names for more flexibility, as shown in the following:

```
sdi-cmd.exe CONNECT -f .\sdi-connection.con  
OUTPUT -f .\results-{$datetime:24h}.json -r json  
CHECK -i "fk,uc"  
LOG -f .\SNOWDay88-log-{$datetime:24h}-{$guid}.csv -r csv -l debug
```

In this modified example, the output file will now have the current system timestamp in YYYYMMDD\_hhmmss format (24-hour clock) appended to the name. The log file will also have the current system timestamp and a GUID appended to its name. Refer to the previous section for more details on filename variables.