

# SNOWFLAKE SNOWCONVERT FOR TERADATA

SnowConvert App Version 1.16.1.0

Engine Version 30.6.1

Executed on 22/8/2025 at 11:09:37 am

Conversion Time:  00:00:04

Conversion Speed: 18 lines/sec

The purpose of this document is to summarize the technical considerations and code analysis in migrating SQL to Snowflake from Teradata that either have an impact on the automated code conversion or cannot be handled by automated code conversion, as well as provide a high-level inventory and automation capability of the code that will need to be addressed. A glossary of terms used is located at the end of the document.

##### KEY TERMS

Code Unit (CU) -SnowConvert breaks down code for reporting here into code units.  Please see the documentation “[here](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/reports/top-level-code-units-report)” for an explanation of how code units are defined.

Code Unit Parent Category (CUPC) - For summary purposes in certain sections of this document you code units are grouped together to display conversion rates, counts and other metrics.  All details for code units can still be found and analyzed in the top level code units document. For information on how code units are grouped, please refer to the documentation [here](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/reports/top-level-code-units-report).

##### CODE COMPLETENESS SCORE

SnowConvert results are only as good as the completeness of the provided code.  A full lineage of information is needed in order to properly convert many objects.  The Code Completeness score is an indication of how complete the provided code base is.  Anything less than a score of 100 means SnowConvert identified missing object references in the code. It is advisable to convert dependent objects together to avoid getting missing object remarks. As an example, a Procedure definition converted individually without the dependent tables or functions would result in missing dependency remarks.

|  |
| --- |
| **50** |

[Learn more](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/reports/assessment-report/code-completeness-score)

For details on objects reported as missing from the submitted code, see the section on Missing Dependent Objects.  For best results, revise the submitted code base to include a complete set of code.

**EXCLUDED SCOPE SUMMARY**

SnowConvert only supports [certain file types and code units](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/snowconvert-scopes).  The following outlines what was identified in the submitted code and has been ***excluded*** from the scope of this assessment.  These excluded items are not converted by SnowConvert and do not affect the overall conversion rates reported in this assessment.  For additional information see the section on Excluded Scope Breakdown.

| **Excluded from Assessment:**  **Files: Out of Scope 50%** | |  | **Excluded from Conversion:**  **Code Units:  Out of Scope 0%** | |
| --- | --- | --- | --- | --- |
| Unsupported extensions: | 1 |  | TRIGGER | 0 |
|  |  |  | GRANT  Functions with unsupported language | 0  0 |

##### ASSESSED CONVERSION SCOPE SUMMARY

| Files | 1 | Fully Converted Code Units | 100% |
| --- | --- | --- | --- |
| Code Units | 2 | Lines of Code Conversion Rate | 100% |
| Lines of Code | 78 | Functional Difference Messages | 2 |
| Files Not Generated | 0 | Performance Reviews | 0 |
| Parsing EWIs | 0 | Missing Dependent Objects | 1 |
| Other EWIs | 0 |  |  |

## TABLE OF CONTENTS

[**SNOWFLAKE SNOWCONVERT FOR TERADATA 1**](#_heading=h.gjdgxs)

[TABLE OF CONTENTS 2](#_heading=h.sa6ljsdy5qff)

[**CONVERSION SETTINGS 4**](#_heading=h.1fob9te)

[**ENVIRONMENT SETTINGS 5**](#_heading=h.3znysh7)

[ANSI Mode versus Teradata Mode: 5](#_heading=h.j7d80z4nyvl0)

[Parameters: 5](#_heading=h.bbgmzwynhylc)

**EXCLUDED** [**SCOPE**](https://docs.google.com/document/d/1Q4jhga2CEGHuypo1rTxuZkUkjhyobqfG/edit#heading=h.tyjcwt) **BREAKDOWN** [**5**](#_heading=h.tyjcwt)

[File Details 6](#_heading=h.3dy6vkm)

[Code Unit Detail 6](#_heading=h.1t3h5sf)

[Unsupported Language Objects 6](#_heading=h.2s8eyo1)

[**CODE UNITS SUMMARY** 7](#_heading=)

[Top Level Code Units 7](#_heading=h.3rdcrjn)

[**CONVERSION REMARKS DETAIL 7**](#_heading=h.26in1rg)

[Functional Difference Messages (FDMs) 7](#_heading=h.lnxbz9)

[Errors, Warnings, & Issues (EWIs) 8](#_heading=h.35nkun2)

[Missing Dependent Objects (MDOs) 9](#_heading=)

[Performance Reviews (PRFs) 9](#_heading=)

[**CODE UNIT CONSIDERATIONS DETAIL 10**](#_heading=h.2jxsxqh)

[Databases & Schemas 10](#_heading=h.z337ya)

[Tables 10](#_heading=h.3j2qqm3)

[Views 15](#_heading=h.1y810tw)

[Join Indexes 15](#_heading=h.4i7ojhp)

[Functions 15](#_heading=h.2xcytpi)

[Procedures 16](#_heading=h.1ci93xb)

[Macros 16](#_heading=h.f3fujd3h8shc)

[**COMPLEX CODE PATTERNS 16**](#_heading=h.3as4poj)

[Subqueries 16](#_heading=h.j1vyjx8cyoz6)

[Cursors 17](#_heading=h.hr6m5xmun1e4)

[Dynamic SQL 18](#_heading=h.2p2csry)

[CONSIDERATIONS 19](#_heading=h.n7x4xmj7hjz1)

[**GLOSSARY 19**](#_heading=h.147n2zr)

## 

## CONVERSION SETTINGS

SnowConvert settings which were used to run this assessment/conversion which affect certain behaviors and translations in the conversion engine.

| **General** | |
| --- | --- |
| Comment objects with missing dependencies | Off |
| Generate XML-tags for SQL statements in Stored Procedures | Off |
| Separate Period Data-type definitions and usages into begin and end Data-Time fields | Off |
| Disable topological level reorder | Off |
| Encoding of the input files | Unicode (UTF-8) |
| Session Mode | TERA |
| Use COLLATE for Case Specification | Off |

| **DB Object Names** | |
| --- | --- |
| Database | None |
| Schema | None |
| Default | On |
| Use Existing Name Qualification | Off |

| **Format Conversion** | |
| --- | --- |
| Character to number default scale | 10 |
| Default TIMESTAMP format | YYYY/MM/DD HH:MI:SS |
| Default DATE format | YYYY/MM/DD |
| Source TIMEZONE | GMT-5 |
| Default TIME format | HH:MI:SS |

| **Target Language for BTEQ, Procedures/Macros** | |
| --- | --- |
| BTEQ | Python |
| Procedures/Macros | Snowflake Scripting |

| **Join Index** | |
| --- | --- |
| Target lag | 1 days |
| Warehouse | UPDATE\_DUMMY\_WAREHOUSE |

## ENVIRONMENT SETTINGS

These are key settings that need to be considered in the Teradata migration.

## 

### ANSI Mode versus Teradata Mode:

[Teradata mode](https://docs.teradata.com/r/Enterprise_IntelliFlex_VMware/SQL-Request-and-Transaction-Processing/Transaction-Processing/Comparison-of-Transactions-in-ANSI-and-Teradata-Session-Modes) has a NOT CASESPECIFIC default configuration and typically ignores only trailing spaces while ANSI mode has a CASESPECIFIC default configuration and ignores both leading and trailing spaces during comparison. Teradata mode has diverse behaviors, but migration supports only string comparison-related statements for this mode. This setting is important to understand when converting code.

## 

### Parameters:

QUOTED\_IDENTIFIERS\_IGNORE\_CASE

(Snowflake) – Should be set to TRUE for the database for a

Teradata migration. This is necessary as there are many identifiers that have to use quotes in Snowflake that are not necessary in Teradata (any name with a # symbol for example).

TIMESTAMP\_TYPE\_MAPPING (Snowflake) – Depending on customer requirements this alias mapping will need to be set to TIMESTAMP\_LTZ for the default behavior of the TIMESTAMP function to match the default behavior in Teradata.

ROUNDHALFWAYMAGUP

(Teradata) – This Teradata parameter controls banker rounding and should be looked at to determine if banker rounding is being used.

## EXCLUDED SCOPE BREAKDOWN

This section contains details on the scope which was submitted but was not assessed or converted by SnowConvert. [See the SnowConvert Documentation for more detail](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/snowconvert-scopes)[.](https://docs.snowconvert.com/snowconvert/general/getting-started/running-snowconvert/review-results/snowconvert-scopes)

### Excluded from Assessment

### File Details

| **File Path/Name** | **File Size** | **Reason** |
| --- | --- | --- |
| storage | 56.0 KB | File Type |

### 

### Excluded from Conversion

##### 

##### Code Unit Detail

| **Type** | **Number of Objects** | **Lines of Code** |
| --- | --- | --- |
| Grant | 0 | 0 |
| Trigger | 0 | 0 |

### Unsupported Language Objects

##### 

| **Type** | **Language** | **Number of Objects** | **Lines of Code** |
| --- | --- | --- | --- |
| Function | Java | 0 | 0 |
| Procedure | Java | 0 | 0 |
| Function | C++ | 0 | 0 |
| Procedure | C++ | 0 | 0 |
| Function | C | 0 | 0 |
| Procedure | C | 0 | 0 |

## CODE UNITS SUMMARY

### Top Level Code Units

Code units are used to holistically count code across multiple types of files and scenarios and then grouped into categories here for summarization.  For information on how CUs are determined please see the [SnowConvert documentation here](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/reports/top-level-code-units-report)[.](https://docs.snowconvert.com/snowconvert/general/getting-started/running-snowconvert/review-results/reports/top-level-code-units-report)

| **Code Unit Parent Category** | **Code Unit** | **Conversion Rate** | | **Total # of Code Units** | **Lines of Code** | **Parsing EWIs** | **Other EWIs** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Fully Converted  Code Units** | **Lines of**  **Code** |
| TABLE | CREATE TABLE | 100 % | 100 % | 1 | 25 | 0 | 0 |
| VIEW | CREATE VIEW | 100 % | 100 % | 1 | 53 | 0 | 0 |

## CONVERSION REMARKS DETAIL

### Functional Difference Messages (FDMs)

In the course of converting your legacy platform code to Snowflake, it is essential to acknowledge that Teradata and Snowflake are distinct platforms, each with its unique set of features, functions, and capabilities.  In many cases, full functional equivalence cannot be achieved through brute force automation and involvement is required to bridge these functional differences.  SnowConvert calls out all known cases of these functional differences as Functional Difference Messages where the code has been successfully converted to the extent possible and can only be further improved by reviewing details that lie outside of the code which may include business use cases, data ingestion processes or other architectural considerations.

##### Purpose

Customer Review: The purpose of this remark is to ensure transparency and provide visibility into areas of the conversion where human intervention may be required. It allows you, as the customer, to review and address these platform-specific differences according to your specific business requirements.

Customization Opportunities: By highlighting these differences, it enables you to assess whether any custom adaptations, workarounds, or alternative solutions may be required to achieve the desired functionality on the Snowflake platform.

Full Compatibility: It aims to ensure that your Snowflake environment, after migration, aligns with your expectations, while also complying with Snowflake's architecture and capabilities.

##### Action Required

Your review of these platform differences is essential. Depending on your specific use case and business needs, there could be substantial work to perform both outside and inside the code in order to achieve a successful migration.  You may need to collaborate with database administrators or developers to implement solutions or workarounds to achieve functionality equivalent to what was present in your legacy platform.  Each of the FDMs should be reviewed and any impacts fully understood prior to deploying any of the code and moving into testing.

Please reach out to our support team for further guidance in addressing these platform differences.

##### Summary

|  | **FDMs** |
| --- | --- |
| # of total remarks | 2 |
| # of unique remarks | 2 |

##### Detail

| **Code** | **Description** | **Instances** |
| --- | --- | --- |
| [SSC-FDM-0007](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/technical-documentation/issues-and-troubleshooting/functional-difference/generalFDM#ssc-fdm-0007) | Element with missing dependencies | 1 |
| [SSC-FDM-TD0024](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/technical-documentation/issues-and-troubleshooting/functional-difference/teradataFDM#ssc-fdm-td0024) | SET TABLE not supported | 1 |

### 

### Errors, Warnings, & Issues (EWIs)

EWIs in the output code are generated by SnowConvert in places where full automation is not implemented, they require review and manual remediation. They typically produce a functional or runtime difference. The SnowConvert team uses internal metrics to classify EWIs based on how much effort, on average, it takes to correct the code.

| **LOW** | **MEDIUM** | **HIGH** | **CRITICAL** |
| --- | --- | --- | --- |
| The user may have to invest a low amount of manual effort to complete the conversion. | The user may have to invest a medium amount of manual effort to complete the conversion. | The user may have to invest a high amount of manual effort to complete the conversion. | Errors that cause exceptions in SnowConvert.  The user may have to invest a substantial amount of manual effort to complete the conversion. |

*For more information about EWIs, such as their severity and examples of each type, please visit our* [*documentation page*](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/technical-documentation/issues-and-troubleshooting/README)

##### Parsing

#### SUMMARY

Total Unparsed Lines of Code:

|  | **CRITICAL** |
| --- | --- |
| # of occurrences | 0 |
| # of unique issues | 0 |

#### DETAIL

| **Error Code** | **Description** | **Instances** | **Severity** |
| --- | --- | --- | --- |
| [N/A](https://docs.snowflake.com/en/migrations/README) | N/A | N/A | N/A |

##### Other

#### SUMMARY

|  | **LOW** | **MEDIUM** | **HIGH** | **CRITICAL** |
| --- | --- | --- | --- | --- |
| # of occurrences | 0 | 0 | 0 | 0 |
| # of unique issues | 0 | 0 | 0 | 0 |

#### 

#### DETAIL

| **Error Code** | **Description** | **Instances** | **Severity** |
| --- | --- | --- | --- |
| [N/A](https://docs.snowflake.com/en/migrations/README) | N/A | N/A | N/A |

### 

### Missing Dependent Objects (MDOs)

Completeness Score (0-100)      50

Total Missing Object References 1

Unique Missing Object References       1

### 

### Performance Reviews (PRFs)

##### Summary

|  | **Informational** |
| --- | --- |
| # of remarks | 0 |
| # of unique remarks | 0 |

##### Detail

| **Code** | **Description** | **Instances** |
| --- | --- | --- |
| [N/A](https://docs.snowflake.com/en/migrations/README) | N/A | N/A |

*If you are using the full version of SnowConvert, you can find out the exact file and location of each error in the Issues Report in the output reports folder created by SnowConvert.*

## CODE UNIT CONSIDERATIONS DETAIL

This section considers both top-level and embedded code units, i.e. tables defined inside procedures or script files will also be counted in the table section. That is the reason why the number of code units here might differ compared to the Top-level code unit breakdown.

The sections below summarize many different key considerations for some specific code unit categories.  Much of the information covered is identified by the different remarks in the code that SnowConvert inserts (FDMs, EWIs, PRFs, etc..).

### Databases & Schemas

Teradata does not have the concept of schemas and a decision must be made when migrating Teradata environments and databases. The simplest approach is for a single Teradata environment to equate to a single Snowflake Database. The individual Teradata databases are then deployed as individual schemas within the Snowflake Database. However, there may be cases where workloads will need to split into separate Snowflake Databases for purposes of managing the resources (sizing) dedicated to a specific database.

##### Databases

Number of Databases Containing Objects: 2

2 database(s) had objects defined (tables, views, macros, join indexes, procedures). The size of this list is important for conversion strategy if converting Teradata databases to Snowflake databases as opposed to Snowflake schemas within a single database.

No databases were found with the name PUBLIC or INFORMATION\_SCHEMA which are default Snowflake schemas.

### 

### Tables

* Invalid column names – These column names are not supported in Snowflake and are converted to XXX\_<column name>\_XXX. This could require changes to external reporting/ETL systems. Examples include ‘current\_date’, ‘localtimestamp’.
* Identifiers Requiring Double Quotes – These objects require double quotes for defining in Snowflake and as a result are case sensitive unless the parameter QUOTED\_IDENTIFIERS\_IGNORE\_CASE has been set to True.

##### Table Types

Table properties may not perfectly line up between Snowflake and your source code platform. Here are some key type differences between Snowflake and your source language.

##### 

| **Tables Types** | **Instances** | **Tables Impacted** | **Percent Impacted** |
| --- | --- | --- | --- |
| SET TABLES | 1 | 1/1 | 100% |
| TEMPORAL TABLES | 0 | 0/1 | 0% |
| GLOBAL TEMPORARY TABLES | 0 | 0/1 | 0% |
| QUEUE TABLES | 0 | 0/1 | 0% |

If a hyphen (‘-’) is listed in the table above, it means no objects of that kind were found in the input folder.

##### Notes

* *SET Tables* – Set tables in Teradata automatically prevent duplicate rows by checking for duplicates upon insert. If a primary key is defined it will base the duplicate check on the primary key, otherwise it compares the entire row - which is very performance intensive. Snowflake does not have this functionality and will require rearchitecting the data ingestion process to check for duplicates prior to insertion if the SET functionality of these tables is being leveraged for de-duplication.
* *Temporal Tables* – Temporal tables operate as pre-built slowly changing dimensions, capturing history as records are inserted/updated/deleted. Snowflake does not have an equivalent table type. Some of this functionality can be mimicked utilizing Streams, Tasks and Time Travel but use cases must be considered due to process timing and change capture requirements.
* *Global Temporary Tables* – Global temporary tables allow for a table to be permanently defined but the contents of the table are specific and temporary to users’ sessions. Snowflake does not currently have a comparable feature and use of these must be re-architected based upon the use case.
* *Queue Tables* – Persistent table type used to handle queue‑oriented data, such as event processing and asynchronous data loading applications. Snowflake does not have a like functionality and a process will need to be defined to emulate the functionality.

##### Data Types

Just as with table types, data properties may not perfectly line up between Snowflake and your source platform. Here are some key data type differences between Snowflake and your source language.

##### 

| **Data Types** | **Instances** | **Tables Impacted** | **Percent Impacted** |
| --- | --- | --- | --- |
| NUMBER without precision | 0 | 0/1 | 0% |
| INTERVAL | 0 | 0/1 | 0% |
| PERIOD | 0 | 0/1 | 0% |
| BLOB | 0 | 0/1 | 0% |
| CLOB | 0 | 0/1 | 0% |
| GEOSPATIAL | 0 | 0/1 | 0% |
| XML/JSON | 0 | 0/1 | 0% |

If a hyphen (‘-’) is listed in the table above, it means no objects of that kind were found in the input folder.

##### Notes

* This assessment can be used as a complement to a full data profile. A complete inventory for this assessment is coming soon.
* *NUMBER without precision* – Number when defined without a scale/precision in Teradata allows for a flexible scale value for any given record from 0 to 38 provided the total precision never exceeds 38. Snowflake does not allow this and always has a fixed scale/precision.
* *INTERVAL* – Interval data types are not supported in Snowflake and are converted to varchars and manipulated as such using custom functions. Heavy use of intervals can cause performance issues and may require some re-architecture of the DDL and SQL.
* *PERIOD* – Period data types are not supported in Snowflake and are converted to varchars. They are a pair of dates or timestamps. Heavy use of periods can cause performance issues and may require some re-architecture of the DDL and SQL.
* *BLOB* – BINARY can be used instead but has a maximum of 8,388,608 bytes.
* *CLOB* – VARCHAR can be used instead but has a maximum of 16,777,216 bytes (for single-byte).
* *GEOSPATIAL* – Geospatial types have recently had support released. Some specific use cases may require evaluation for workarounds.
* *XML/JSON* – Special column types that are converted to VARIANT columns in Snowflake and may need additional attention in testing/data loading.

##### Data Values

Data values that have significance in a migration.

##### 

| **Data Types** | **Instances** | **Tables Impacted** | **Percent Impacted** |
| --- | --- | --- | --- |
| Binary Defaults | 0 | 0/1 | 0% |
| TIME Defaults | 0 | 0/1 | 0% |
| FLOAT Defaults | 0 | 0/1 | 0% |
| IDENTITY Columns | 0 | 0/1 | 0% |
| SESSION Defaults | 0 | 0/1 | 0% |

If a hyphen (‘-’) is listed in the table above, it means no objects of that kind were found in the input folder.

##### Notes

* *Binary Defaults* – Binary data types do not currently support a default value being defined.
* *TIME Defaults* – Time with TimeZone is not a supported type in Snowflake
* *FLOAT Default* – Timestamp can be defined as a default value for a FLOAT column in Teradata but is not supported in Snowflake. This is typically a column required for an auxiliary system such as SAS and many times will not be an issue once the system has been adjusted to work with Snowflake.
* *IDENTITY Columns* – These are converted to sequences and can represent an area that requires additional testing. No unique definitions were found. A large number of these can represent an area of increased testing, particularly if sequences are required to be kept in progression at cutover.
* *SESSION Defaults* – These columns are defined as integers in Teradata and must be varchars in Snowflake to hold the Snowflake current\_session value.

##### Unique Conversion Elements

Other elements that may not explicitly fit into one of the previous categories that have significance in a migration.

##### 

| **Data Types** | **Instances** | **Tables Impacted** | **Percent Impacted** |
| --- | --- | --- | --- |
| Formats | 3 | 1/1 | 100% |
| Partition By | 0 | 0/1 | 0% |
| UPPERCASE | 0 | 0/1 | 0% |
| NOT CASESPECIFIC | 8 | 1/1 | 100% |

If a hyphen (‘-’) is listed in the table above, it means no objects of that kind were found in the input folder.

##### Notes

* Formats – Default format definitions by column are not supported in Snowflake. This can have implications for reporting or data extracts that are created for consumption. No unique format definitions were found.
* Partition By – Partitions are not supported in Snowflake but have a similar functionality called CLUSTER BY. By default, the PARTITION BY statements are converted to CLUSTER BY but are commented out as CLUSTER BY does not support all the same cases that can be defined in Partitions, so some must be modified before enabling. Each CLUSTER BY statement will need to be individually evaluated based upon the necessity for performance.
* UPPERCASE – Not supported in Snowflake and must be analyzed to determine if case insensitive collation is sufficient.
* NOT CASESPECIFIC – This collation setting can cause performance degradation in Snowflake when executing queries. Its use should be analyzed to decide if removing the collation is needed or not. Notice that if the conversion setting “*TERA mode string comparison*” is enabled, the collation settings will not appear in the migration.

### Views

Code Unit Conversion Rate: 100 %

Lines of Code Conversion Rate: 100 %

Number of Views: 1

Number of Views created with only SELECT \* FROM: 0

Number of Views with more than 10 nested Select Statements: 0

Lines of Code: 53

Total Parsing Errors: 0

Conversion Errors: 0

### Join Indexes

Join indexes are converted to dynamic tables in Snowflake.  How join indexes are refreshed is a consideration that must be made outside of the automated conversion process.

Code Unit Conversion Rate: 0 %

Conversion Rate by Lines of Code: 0 %

Number of Join Indexes: 0

Lines of Code: 0

Parsing Errors: 0

Conversion Errors: 0

Number of Join Indexes referencing multiple objects: 0

### Functions

Code Unit Conversion Rate: 0 %

Conversion Rate by Lines of Code: 0 %

Number of Functions: 0

Lines of Code: 0

Parsing Errors: 0

Conversion Errors: 0

Functions with C Body: 0

Functions with C++ Body: 0

Functions with Java Body: 0

Functions with SQL Body: 0

### Procedures

Code Unit Conversion Rate: 0 %

Conversion Rate by Lines of Code: 0 %

Number of Procedures: 0

Number of Procedures with InOut or Out Parameters: 0

Lines of Code: 0

Parsing Errors: 0

Conversion Errors: 0

Dynamic Result Sets: 0

Cursors: -

Loops (While, For, and Loop): -

ACTIVITY\_COUNT Occurrences: 0

### Macros

Code Unit Conversion Rate: 0 %

Conversion Rate by Lines of Code: 0 %

Number of Macros: 0

Lines of Code: 0

Parsing Errors: 0

Conversion Errors: 0

## COMPLEX CODE PATTERNS

### Subqueries

| **Subquery pattern** | **Instances** | **Views Impacted** | **Procedures Impacted** | **Macros Impacted** |
| --- | --- | --- | --- | --- |
| Subquery clause | 1 | 1/1 | 0/0 | 0/0 |
| Correlated subquery | 0 | 0/1 | 0/0 | 0/0 |

##### Unsupported subqueries

To determine if a subquery is supported, Snowflake considers two characteristics of the subquery: if it is correlated and if it is scalar.

A query is correlated if it contains a reference to any column from outside the query, while a query is scalar if it returns a single value (a single row containing a single column). For correlated subqueries, to determine if a query is scalar Snowflake evaluates the query statically, meaning the query should have only an aggregate function and no group by.

According to the [Snowflake documentation](https://docs.snowflake.com/en/user-guide/querying-subqueries.html#types-supported-by-snowflake), the supported subqueries are:

* Uncorrelated scalar subqueries in any place where a value expression can be used.
* Correlated scalar subqueries in WHERE clauses.
* EXISTS, ANY / ALL, and IN subqueries. These subqueries can be correlated or uncorrelated.

The above list is not extensive, meaning there would be cases that do not comply with any of the list conditions but are still valid.

The following table shows a set of subquery patterns found in the input code that normally invalidate subqueries in Snowflake.

| **Subquery patterns** | **Instances** | **Views Impacted** | **Procedures Impacted** | **Macros Impacted** |
| --- | --- | --- | --- | --- |
| More than one item in the query column list | 0 | 0/1 | 0/0 | 0/0 |
| Subquery outside a WHERE clause | 0 | 0/1 | 0/0 | 0/0 |

### Cursors

##### 

##### Cursor complex patterns

The following breakdown gives information about cursor complex patterns.

##### 

| **Pattern type** | **Instances** | **Procedures Impacted** |
| --- | --- | --- |
| Cursor FOR-LOOP  (SSC-PRF-0004) | 0 | 0 |
| Fetch inside loops  (SSC-PRF-0003) | 0 | 0 |

##### 

##### Cursor relevant information

​​The following breakdown provides more relevant information that might be helpful to better understand the migration.

| **Cursor Definitions** | **Instances** | **Procedures Impacted** |
| --- | --- | --- |
| Basic Cursor | 0 | 0 |
| Cursor FOR UPDATE | 0 | 0 |
| Cursor with variables | 0 | 0 |

| **Cursor Usages** | **Instances** | **Procedures Impacted** |
| --- | --- | --- |
| Fetch | 0 | 0 |
| FETCH FIRST | 0 | 0 |
| UPDATE WHERE CURRENT OF | 0 | 0 |
| DELETE WHERE CURRENT OF | 0 | 0 |

### Dynamic SQL

Tables created with dynamic SQL:  0

Views created with dynamic SQL:  0

| **Statements** | **Instances** | **Procedures Impacted** | **Functions impacted** |
| --- | --- | --- | --- |
| EXECUTE IMMEDIATE | 0 | 0 | 0 |
| Procedure call | 0 | 0 | 0 |

Total statements with usages of dynamic SQL:  0

* The data inside “Procedures Impacted” and “Functions Impacted” includes Procedures and Functions as Top Level Code Units as well as Procedures and Functions defined within Package Body.

## 

## CONSIDERATIONS

Find information about SnowConvert calculations and other report details in the [considerations documentation](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/technical-documentation/considerations/teradata).

## GLOSSARY

Visit the [glossary](https://docs.snowflake.com/en/migrations/snowconvert-docs/general/getting-started/running-snowconvert/review-results/reports/README#glossary) to understand the terminology used in multiple report documents generated by SnowConvert.