

**Snowflake SQL**

**Course Number:** SQL-700  
**Duration:** 3 days

**Overview**

Snowflake is a popular data warehouse platform that supports SQL. In this Snowflake SQL training course, attendees will greatly expand their ability to query Snowflake using SQL.

**Prerequisites**

The default 3-day length of this class presumes that attendees have prior SQL SELECT experience and that at least the first two chapters will be review.  If your group is new to writing SQL queries, we recommend that we expand the class to 4 days.

**Materials**

All students receive comprehensive courseware.

**Software Needed on Each Student PC**

All attendees will need access to a Snowflake SQL environment with Accelebrate’s sample data loaded.

**Objectives**

* Work with the Snowflake database to run SQL statements, retrieve advanced analytics, and create graphs and charts
* Learn a wide variety of Snowflake analytics
* Use advanced Order By methods to sort the answer sets retrieved
* Fully understand and use Joins and Subqueries
* Interrogate the data using Case, Coalesce, and Decode
* Be able to fully use all of the different Snowflake temporary options
* Create tables and views on the Snowflake system
* Utilize the many different Snowflake date functions
* Use Aggregation and advanced Aggregation techniques
* Handle and manipulate Strings
* Learn and write SQL for advanced Statistical Aggregate Functions

**Outline**

* Introduction
* Basic SQL Functions
  + Introduction
  + Setting Your Default Database and Schema
  + SELECT \* (All Columns) in a Table
  + SELECT Specific Columns in a Table
  + Commas in the Front or Back?
  + Place your Commas in front for better Debugging Capabilities
  + Sort the Data with the ORDER BY Keyword
  + Use a Column Name or Number in an ORDER BY Statement
  + Two Examples of ORDER BY using Different Techniques
  + Changing the ORDER BY to Descending Order
  + NULL Values sort Last in Ascending Mode (Default)
  + Using the Nulls First Command
  + NULL Values sort First in Descending Mode (DESC)
  + Using the Nulls Last Command
  + Major Sort vs. Minor Sort
  + Multiple Sort Keys using Names vs. Numbers
  + Sorts are Alphabetical, NOT Logical
  + Using A Valued CASE Statement to Sort Logically
  + Using A Searched CASE Statement to Sort Logically
  + Quiz – Can you Add a Minor Sort?
  + Answer – Can you Add a Minor Sort?
  + Using Decode to Sort Logically
  + How to ALIAS a Column Name
  + A Missing Comma can by Mistake become an Alias
  + Comments using Double Dashes are Single Line Comments
  + Comments for Multi-Lines
  + Comments for Multi-Lines as Double Dashes per Line
  + Comments are a Great Technique for Finding SQL Errors
  + Popular Snowflake Functions
  + Move Data to the Snowflake Effortlessly
  + Move Data to the Cloud Effortlessly
* The WHERE Clause
  + The WHERE Clause limits Returning Rows
  + The WHERE Clause Needs Single-Quotes for Character Data
  + Using a Column ALIAS in the WHERE Clause
  + Numbers Don't Need Single or Double Quotes
  + Searching for NULL Values Using Equality Returns Nothing
  + Use IS NULL or IS NOT NULL when dealing with NULLs
  + Using Greater Than OR Equal To (>=)
  + AND in the WHERE Clause
  + Troubleshooting AND
  + OR in the WHERE Clause
  + Troubleshooting OR
  + WHY OR must utilize the Column Name Each Time
  + Troubleshooting Character Data
  + Using Different Columns in an AND Statement
  + What is the Order of Precedence?
  + Using Parentheses to change the Order of Precedence
  + Using an IN List in place of OR
  + The IN List is an Excellent Technique
  + IN List vs. OR brings the same Results
  + The IN List Can Use Character Data
  + Using a NOT IN List
  + Null Values in a NOT IN List Return No Rows
  + A Technique for Handling Nulls with a NOT IN List
  + The BETWEEN Statement is Inclusive
  + The NOT BETWEEN Statement is also Inclusive
  + The BETWEEN Statement Works for Character Data
  + LIKE uses Wildcards Percent ‘%’ and Underscore ‘\_’
  + LIKE command Underscore is Wildcard for one Character
  + Using Upper and Lower to Handle Case Issues
  + Using ILIKE Handle Case Issues
  + Finding Anyone Who Name End in 'Y'
  + Escape Character in the LIKE Command changes Wildcards
  + Escape Characters Turn off Wildcards in the LIKE Command
* Distinct, Group By, Top, and Pivot
  + The Distinct Command
  + Distinct vs. GROUP BY
  + TOP Command
  + TOP Command is brilliant when ORDER BY is Used!
  + The FETCH Clause
  + Sample and Tablesample
  + TOP vs. Sample
  + The Pivot Command
* Aggregation
  + There are Five Aggregates
  + Troubleshooting Aggregates
  + GROUP BY when Aggregates and Normal Columns Mix
  + GROUP BY Delivers one row per Group
  + GROUP BY Dept\_No or GROUP BY 1 the same thing
  + Limiting Rows and Improving Performance with WHERE
  + WHERE Clause in Aggregation limits unneeded Calculations
  + Keyword HAVING tests Aggregates after they are Totaled
  + Keyword HAVING is like an Extra WHERE Clause for Totals
  + Three types of Advanced Grouping
  + GROUP BY Grouping Sets
  + GROUP BY Grouping Sets Answer Set
  + GROUP BY Rollup
  + GROUP BY Rollup ResultSet
  + GROUP BY Cube
  + GROUP BY Cube ResultSet
* Join Functions
  + A two-table join using Non-ANSI Syntax
  + A two-table join using Non-ANSI Syntax with Table Alias
  + You Can Fully Qualify All Columns
  + A two-table join using ANSI Syntax
  + Both Queries have the same Results and Performance
  + Left Outer Join
  + Left Outer Join Results
  + LEFT OUTER JOIN Using (+)
  + RIGHT OUTER JOIN
  + Right Outer Join Example and Results
  + RIGHT OUTER JOIN Using (+)
  + Full Outer Join
  + Full Outer Join Results
  + Which Tables are the Left, and which are the Right?
  + INNER JOIN with Additional AND Clause
  + ANSI INNER JOIN with Additional AND Clause
  + ANSI INNER JOIN with Additional WHERE Clause
  + OUTER JOIN with Additional WHERE Clause
  + OUTER JOIN with Additional AND Clause
  + The DREADED Product Join
  + The DREADED Product Join Results
  + Cartesian Product Join with Traditional Syntax
  + Cartesian Product Join with ANSI Syntax
  + The CROSS JOIN
  + The CROSS JOIN Answer Set
  + The Self Join
  + An Associative Table is a Bridge that Joins Two Tables
  + The 5-Table Join – Logical Insurance Model
* Date Functions
  + Current Date
  + Current\_Date, Current\_Time, and Current\_Timestamp
  + Current\_Time vs. LocalTime With Precision
  + Local\_Time and Local\_Timestamp With Precision
  + Add or Subtract Days from a date
  + The ADD\_MONTHS Command
  + Using the ADD\_MONTHS Command to Add 1 Year
  + Using the ADD\_MONTHS Command to Add 5 Years
  + Formatting a Date Using the To\_Char Command
  + Formatting Date and Time With To\_Char
  + The To\_Char command to format Dollar Signs
  + The To\_Char Command for Formatting Numbers
  + The EXTRACT Command
  + EXTRACT from DATES and TIME
  + Using Extract
  + EXTRACT from DATES and TIME Optional Syntax
  + Another Option for Extracting Portions of Dates and Times
  + Using Date\_Part to Extract
  + Implied Extract of Day, Month and Year using to\_char
  + The Date\_Part Function Using a Date
  + Great Date Functions to Know
  + DAYOFWEEK and a CASE Statement
  + Year and Days for the First/Last Weeks of the Year
  + First Day and Last Day Functions
  + Incrementing Date Values Using the Dateadd Function
  + Incrementing Time Values Using Dateadd
  + The Datediff command
  + The Datediff Function on Column Data
  + Calculating Days Between using the DATEDIFF Function
  + Changing the Date to a Timestamp
  + Find the First Day of the Current Month
  + Using Intervals
  + Using Day, Month, and Year Intervals
  + Complex Interval
* Analytics
  + The Row\_Number Command
  + Find the Top Two Students Per Class\_Code using Qualify
  + Find the Top Two Students using a Derived Table
  + The RANK Command
  + Getting RANK to Sort in DESC Order
  + RANK () OVER and PARTITION BY
  + RANK() OVER and a Qualify Statement
  + RANK() OVER and a WITH Derived Table
  + RANK vs. DENSE\_RANK
  + DENSE\_RANK() OVER and PARTITION BY
  + DENSE\_RANK() OVER and QUALIFY
  + PERCENT\_RANK () OVER with 14 rows in Calculation
  + PERCENT\_RANK () OVER with 21 rows in Calculation
  + PERCENT\_RANK() OVER and PARTITION BY
  + CSUM
  + CSUM – The Sort Explained
  + CSUM – Rows Unbounded Preceding Explained
  + The CSUM – Making Sense of the Data
  + CSUM – The Major and Minor Sort Key(s)
  + The ANSI OLAP – Reset with a PARTITION BY Statement
  + Totals and Subtotals through Partition BY
  + Moving SUM
  + Moving SUM every 3-rows Vs. a Continuous Average
  + Partition By Resets the Calculations
  + Moving Average
  + How the Moving Average Calculates
  + How the Sort works for Moving Average (MAVG)
  + Moving Average every 3-rows Vs. a Continuous Average
  + Partition BY Resets an ANSI OLAP
  + Moving Difference using ANSI Syntax
  + Moving Difference using ANSI Syntax with Partition By
  + Finding a Value of a Column in the Next Row with MIN
  + Finding a Value of a Column in the Next Row with PARTITION BY
  + Finding Multiple Values of a Column in Upcoming Rows
  + Finding The Next Date using MAX
  + COUNT OVER for a Sequential Number
  + COUNT OVER using ROWS UNBOUNDED PRECEDING
  + The MAX OVER Command
  + MAX OVER with PARTITION BY Reset
  + The MIN OVER Command
  + The MIN OVER Command with PARTITION BY
  + Finding Gaps Between Dates
  + The CSUM For Each Product\_Id For The First 3 Days
  + Using FIRST\_VALUE
  + FIRST\_VALUE
  + FIRST\_VALUE with Partitioning
  + FIRST\_VALUE Combined with Row\_Number and Qualify
  + FIRST\_VALUE and Row\_Number with a Derived Table
  + Using LEAD
  + Using LEAD with a PARTITION Statement
  + Using LEAD With an Offset of 2
  + Using LEAD With an Offset of 2 and a PARTITION
  + Using LAG
  + Using LAG with a PARTITION Statement
  + Using LAG With an Offset of 2
  + Using LAG With an Offset of 2 and a PARTITION
  + CUME\_DIST
  + CUME\_DIST With a Partition
  + CURRENT ROW AND UNBOUNDED FOLLOWING
  + Different Windowing Options
  + LISTAGG Basic Example
  + Another Example of LISTAGG
  + LISTAGG With a Pipe-Separated List
  + LISTAGG With a Comma-Separated List in Groups
  + MEDIAN Function
  + MEDIAN Example
  + MEDIAN with Partitioning and a WHERE Clause
  + MEDIAN with Partitioning
  + NTILE Function
  + How Ntile Works
  + Ntile
  + Ntile Continued
  + Ntile Percentile
  + Another Ntile Example
  + Using Quantiles (Partitions of Four)
  + NTILE Using a Value of 10
  + NTILE With a Partition
  + NTH\_VALUE Function and Syntax
  + NTH\_VALUE Arguments
  + NTH\_VALUE
  + NTH\_VALUE With Partition
  + NTH\_VALUE With Partition and Ignore Nulls
  + PERCENTILE\_CONT Function Description and Syntax
  + Final Result Information About PERCENTILE\_CONT
  + PERCENTILE\_CONT Function Arguments
  + PERCENTILE\_CONT Example
  + PERCENTILE\_CONT Example with Percentage Change
  + PERCENTILE\_CONT With PARTITION Example
  + PERCENTILE\_CONT With PARTITION and (0.4)
  + PERCENTILE\_DISC Function Description and Syntax
  + PERCENTILE\_DISC Function Arguments
  + PERCENTILE\_DISC Example
  + PERCENTILE\_DISC Example with Percentage Change
  + PERCENTILE\_DISC With PARTITION Example
  + PERCENTILE\_DISC With PARTITION and (0.4)
  + RATIO\_TO\_REPORT Function
  + RATIO\_TO\_REPORT Example
  + RATIO\_TO\_REPORT Example with Partitioning
  + SUM(SUM(n))
* Temporary Tables
  + CREATING A Derived Table
  + Naming the Derived Table
  + Aliasing the Column Names in the Derived Table
  + CREATING A Derived Table using the WITH Command
  + Derived Query Examples with Three Different Techniques
  + Most Derived Tables Are Used To Join To Other Tables
  + The Three Components of a Derived Table
  + Visualize This Derived Table
  + Our Join Example With the WITH Syntax
  + An Example of Two Derived Tables in a Single Query
  + MULTIPLE Derived Tables using the WITH Command
  + WITH RECURSIVE Derived Table Hierarchy
  + WITH RECURSIVE Derived Table Query
  + WITH RECURSIVE Derived Table Definition
  + WITH RECURSIVE Derived Table Seeding
  + WITH RECURSIVE Derived Table Looping
  + WITH RECURSIVE Derived Table Looping in Slow Motion
  + WITH RECURSIVE Derived Table Looping Continued
  + WITH RECURSIVE Derived Table Ends the Looping
  + WITH RECURSIVE Derived Table Final Report
  + Creating a Temporary Table
  + Creating a Temporary Table using a CTAS
  + Dropping a Temporary Table
* Sub-query Functions
  + An IN List is much like a Subquery
  + An IN List Never has Duplicates – Just like a Subquery
  + An IN List Ignores Duplicates
  + The Subquery
  + How a Basic Subquery Works
  + These are Equivalent Queries
  + The Final Answer Set from the Subquery
  + Should you use a Subquery of a Join?
  + The Basics of a Correlated Subquery
  + The Top Query always runs first in a Correlated Subquery
  + Correlated Subquery Example vs. a Join with a Derived Table
  + NOT IN Subquery Returns Nothing when NULLs are Present
  + Fixing a NOT IN Subquery with Null Values
  + How the Double Parameter Subquery Works
  + More on how the Double Parameter Subquery Works
  + IN is equivalent to =ANY
  + Using a Correlated Exists
  + How a Correlated Exists matches up
  + The Correlated NOT Exists
* Strings
  + UPPER and lower  Functions
  + The Length Command Counts Characters
  + LENGTH Works on Fixed Length Columns
  + LENGTH and OCTET\_LENGTH
  + The TRIM Command trims both Leading and Trailing Spaces
  + The RTRIM and LTRIM Command trims Spaces
  + Concatenation
  + Concatenation and SUBSTRING
  + Four Concatenations Together
  + LPAD and RPAD
  + The SUBSTR and SUBSTRING Command
  + How SUBSTR Works with NO ENDING POSITION
  + Using SUBSTR and LENGTH Together
  + The LEFT and RIGHT Functions
  + The POSITION Command finds a Letters Position
  + The POSITION Command is brilliant with SUBSTR
  + CHARINDEX Finds a Letter(s) Position in a String
  + The CHARINDEX Command is brilliant with SUBSTRING
  + The CHARINDEX Command Using a Literal
  + The REPLACE Function
  + REGEXP\_REPLACE
  + REGEXP\_INSTR
  + SOUNDEX Function to Find a Sound
  + The ASCII Function
  + The CHAR Function
  + The UNICODE Function
  + The Reverse String Function
  + The RIGHT Function
* Interrogating the Data
  + Numeric Manipulation Functions
  + Finding the Cube Root
  + Ceiling Gets the Smallest Integer Not Smaller Than X
  + Floor Finds the Largest Integer Not Greater Than X
  + The Round Function and Precision
  + The COALESCE Command
  + COALESCE is Equivalent to this CASE Statement
  + A Rounding Example Using CAST
  + CAST will Round Values up or Down
  + Valued Case vs. Searched Case
  + Combining Searched Case and Valued Case
  + Nested Case
  + The CASE Challenge
  + The CASE Challenge Answer
  + The Decode Command
  + A Trick for getting a Horizontal Case
  + Put a CASE in the ORDER BY
  + Using Decode to Sort Logically
* View Functions
  + The Fundamentals of Views
  + Creating a Simple View to Restrict Sensitive Columns
  + Creating a Simple View to Restrict Rows
  + Creating a View to Join Tables Together
  + Join Views Allow Users to Merely Select Columns
  + Sometimes we Create Views for Formatting
  + Basic Rules for Views
  + How to Modify a View
  + The Exception to the ORDER BY Rule inside a View
  + Derived Columns in a View Should Contain a Column Alias
  + The Standard Way Most Aliasing is Done
  + Another Way to Alias Columns in a View CREATE
  + What Happens When a View Column gets Aliased Twice?
  + Aggregates on View Aggregates
* UNION Set Operator
  + Rules of Set Operators
  + INTERSECT Explained Logically
  + INTERSECT Explained Logically
  + UNION Explained Logically
  + UNION Explained Logically
  + UNION ALL Explained Logically
  + UNION ALL Explained Logically
  + EXCEPT Explained Logically
  + EXCEPT Explained Logically
  + Minus Explained Logically
  + Minus Explained Logically
  + An Equal Number of Columns in both SELECT Lists
  + Columns in the SELECT list should be from the same Domain
  + The Top Query handles all Aliases
  + The Bottom Query does the ORDER BY
  + Great Trick:  Place your Set Operator in a Derived Table
  + UNION vs. UNION ALL
  + Using UNION ALL and Literals
  + A Great Example of how EXCEPT works
  + USING Multiple SET Operators in a Single Request
  + Changing the Order of Precedence with Parentheses
  + Using UNION ALL for speed in Merging Data Sets
  + Using UNION to be same as GROUP BY GROUPING SETS
* Creating Tables
  + Show Databases and Table DDL Commands
  + Finding Constraints
  + Create Table Syntax
  + Creating A Table in Snowflake
  + Creating Temporary and Transient Tables
  + Comparing Table Types
  + Data Types for Numeric, String, and Binary
  + Data Types for Date, Time, and Unstructured
  + Creating Tables with a Clustering Key
  + Joining Tables Can Have the Same Clustering Keys for Speed
  + Creating Tables with a Primary Key/Foreign Key Relationship
  + A Table with a NOT NULL Constraint
  + CREATE TABLE LIKE
  + CREATE a Temporary TABLE using LIKE
  + CREATE TABLE AS (CTAS) Populates the Table With Data
  + CREATE TABLE AS (CTAS) Can Choose Certain Columns
  + CREATE a Temporary Table AS (CTAS)
  + CREATE a Temporary Table AS (CTAS) Using a Join
* Data Manipulation Language (DML)
  + INSERT Syntax # 1
  + INSERT Syntax # 2
  + INSERT Example with Multiple Rows
  + Inserting Null Values into a Table
  + INSERT/SELECT Command
  + INSERT/SELECT to Build a Data Mart
  + UPDATE Examples
  + Subquery UPDATE Command Syntax
  + Example of Subquery UPDATE Command
  + Deleting Rows in a Table
* Statistical Aggregate Functions
  + The Stats Table
  + The KURTOSIS Function
  + A Kurtosis Example
  + The SKEW Function
  + A SKEW Example
  + The STDDEV\_POP Function
  + A STDDEV\_POP Example
  + The STDDEV\_SAMP Function
  + A STDDEV\_SAMP Example
  + The VAR\_POP Function
  + A VAR\_POP Example
  + The VAR\_SAMP Function
  + A VAR\_SAMP Example
  + The CORR Function
  + A CORR Example
  + Another CORR Example so you can Compare
  + The COVAR\_POP Function
  + A COVAR\_POP Example
  + Another COVAR\_POP Example so you can Compare
  + The REGR\_INTERCEPT  Function
  + A REGR\_INTERCEPT  Example
  + Another REGR\_INTERCEPT Example so you can Compare
  + The REGR\_SLOPE Function
  + A REGR\_SLOPE Example
  + Another REGR\_SLOPE Example so you can Compare
  + The REGR\_AVGX Function
  + A REGR\_AVGX Example
  + Another REGR\_AVGX Example so you can Compare
  + The REGR\_AVGY Function
  + A REGR\_AVGY Example
  + Another REGR\_AVGY Example so you can Compare
  + The REGR\_COUNT Function
  + A REGR\_COUNT Example
  + The REGR\_R2 Function
  + A REGR\_R2 Example
  + The REGR\_SXX Function
  + A REGR\_SXX Example
  + The REGR\_SXY Function
  + A REGR\_SXY Example
  + The REGR\_SYY Function
  + A REGR\_SYY Example
  + Using GROUP BY
* Conclusion