1.Merge into query

**Syntax:**

**MERGE** **INTO** <target\_table> **USING** <source> **ON** <join\_expr> { **matchedClause** | **notMatchedClause** } [ ... ]

**Examples**

Perform a simple merge:

Create and load the tables:

**CREATE** **TABLE** target\_table **(**ID **INTEGER,** description **VARCHAR);**

**CREATE** **TABLE** source\_table **(**ID **INTEGER,** description **VARCHAR);**

**INSERT** **INTO** target\_table **(**ID**,** description**)** **VALUES**

**(**10**,** 'To be updated (this is the old value)'**)**

**;**

**INSERT** **INTO** source\_table **(**ID**,** description**)** **VALUES**

**(**10**,** 'To be updated (this is the new value)'**)**

**;**

Execute the MERGE statement:

Syntax:

**MERGE** **INTO** target\_table **USING** source\_table

**ON** target\_table**.**id **=** source\_table**.**id

**WHEN** **MATCHED** **THEN**

**UPDATE** **SET** target\_table**.**description **=** source\_table**.**description**;**

**o/p:**

**+**-----------------------------------+

| number of rows updated |

|-----------------------------------|

| 1 |

**+**-----------------------------------+

Display the new value(s) in the target table (the source table is unchanged):

**SELECT** **\*** **FROM** target\_table**;**

**o/p:**

**+**----+--------------------------------------------------+

| ID | DESCRIPTION |

|----+---------------------------------------------------|

| 10 | To be updated (this is the new value) |

**+**----+---------------------------------------------------+

**SELECT** **\*** **FROM** source\_table**;**

**o/p:**

**+**----+---------------------------------------------------+

| ID | DESCRIPTION |

|----+---------------------------------------------------|

| 10 | To be updated (this is the new value) |

**+**----+---------------------------------------------------+

**Syntax:**

**MERGE** **INTO** target\_table **USING** source\_table

**ON** target\_table**.**id **=** source\_table**.**id

**WHEN** **MATCHED** **THEN**

**UPDATE** **SET** target\_table**.**description **=** source\_table**.**description

**WHEN** **NOT** **MATCHED** **THEN**

**INSERT** **(**ID**,** description**)** **VALUES** **(**source\_table**.**id**,** source\_table**.**description**);**

**o/p:**

**+**-------------------------+----------------------------------------------+

| number of rows inserted | number of rows updated |

|-------------------------+-----------------------------------------------|

| 2 | 0 |

**+**-------------------------+----------------------------------------------+

Example2:

In the following example, the members table stores the names, addresses, and current fees (members.fee) paid to a local gym. The signup table stores each member’s signup date (signup.date). The MERGE statement applies a standard $40 fee to members who joined the gym more than 30 days ago, after the free trial expired:

Syntax:

**MERGE** **INTO** members m

**USING** **(**

**SELECT** id**,** **date**

**FROM** signup

**WHERE** DATEDIFF**(**day**,** CURRENT\_DATE**(),** signup**.date::DATE)** **<** **-**30**)** s **ON** m**.**id **=** s**.**id

**WHEN** **MATCHED** **THEN** **UPDATE** **SET** m**.**fee **=** 40**;**

**2.with query**

**Syntax:** Subquery:

[ **WITH**

<cte\_name1> [ ( <cte\_column\_list> ) ] **AS** ( **SELECT** ... )

[ , <cte\_name2> [ ( <cte\_column\_list> ) ] **AS** ( **SELECT** ... ) ]

[ , <cte\_nameN> [ ( <cte\_column\_list> ) ] **AS** ( **SELECT** ... ) ]

]

**SELECT** ...

**Syntax2**:

Recursive CTE:

[ **WITH** [ **RECURSIVE** ]

<cte\_name1> ( <cte\_column\_list> ) **AS** ( **anchorClause** **UNION** **ALL** **recursiveClause** )

[ , <cte\_name2> ( <cte\_column\_list> ) **AS** ( **anchorClause** **UNION** **ALL** **recursiveClause** ) ]

[ , <cte\_nameN> ( <cte\_column\_list> ) **AS** ( **anchorClause** **UNION** **ALL** **recursiveClause** ) ]

]

**SELECT** ...

**Examples**:

**1.Non-recursive examples:**

**Syntax:**

**with**

**albums\_1976 as (select \* from music\_albums where album\_year = 1976)**

**select album\_name from albums\_1976 order by album\_name;**

**o/p:**

**+----------------------+**

**| ALBUM\_NAME |**

**|----------------------|**

**| Amigos |**

**| Look Into The Future |**

**+----------------------+**

**Example2:**

**This next example uses a WITH clause with an earlier WITH clause; the CTE named journey\_album\_info\_1976 uses the CTE named album\_info\_1976. The output is the album “Look Into The Future”, with the name of the band:**

**Syntax:**

**with**

**album\_info\_1976 as (select m.album\_ID, m.album\_name, b.band\_name**

**from music\_albums as m inner join music\_bands as b**

**where m.band\_id = b.band\_id and album\_year = 1976),**

**Journey\_album\_info\_1976 as (select \***

**from album\_info\_1976**

**where band\_name = 'Journey')**

**select album\_name, band\_name**

**from Journey\_album\_info\_1976;**

**+----------------------+-----------+**

**| ALBUM\_NAME | BAND\_NAME |**

**|----------------------+-----------|**

**| Look Into The Future | Journey |**

**+----------------------+-----------+**

**3.how to join two tables in snowflake**

**Types of Joins**

**Snowflake supports the following types of joins:**

* **Inner join.**
* **Outer join.**
* **Cross join.**
* **Natural join.**
* [**ASOF JOIN**](https://docs.snowflake.com/en/sql-reference/constructs/asof-join)

### Inner Join

An inner join pairs each row in one table with the matching row(s) in the other table.

The example below uses an inner join:

Syntax:

**SELECT** p**.**project\_ID**,** project\_name**,** employee\_ID**,** employee\_name**,** e**.**project\_ID

**FROM** projects **AS** p **INNER** **JOIN** employees **AS** e

**ON** e**.**project\_id **=** p**.**project\_id

**ORDER** **BY** p**.**project\_ID**,** e**.**employee\_ID**;**

**+**------------+------------------+-------------+-----------------+------------+

| PROJECT\_ID | PROJECT\_NAME | EMPLOYEE\_ID | EMPLOYEE\_NAME | PROJECT\_ID |

|------------+------------------+-------------+-----------------+------------|

| 1000 | COVID-19 Vaccine | 10000001 | Terry Smith | 1000 |

| 1000 | COVID-19 Vaccine | 10000002 | Maria Inverness | 1000 |

| 1001 | Malaria Vaccine | 10000003 | Pat Wang | 1001 |

**+**------------+------------------+-------------+-----------------+------------+

**Outer Join:**

An outer join lists all rows in the specified table, even if those rows have no match in the other table. For example, a left outer join between projects and employees lists all projects, including projects that do not yet have any employee assigned.

Syntax:

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **LEFT** **OUTER** **JOIN** employees **AS** e

**ON** e**.**project\_ID **=** p**.**project\_ID

**ORDER** **BY** p**.**project\_name**,** e**.**employee\_name**;**

**+**---------------------+---------------------------+

| PROJECT\_NAME | EMPLOYEE\_NAME|

|------------------+------------------------------|

| COVID-19 Vaccine | Maria Inverness |

| COVID-19 Vaccine | Terry Smith |

| Malaria Vaccine | Pat Wang |

| NewProject | NULL |

**+**--------------------+---------------------------+

i.A right outer join lists all employees (regardless of project).

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **RIGHT** **OUTER** **JOIN** employees **AS** e

**ON** e**.**project\_ID **=** p**.**project\_ID

**ORDER** **BY** p**.**project\_name**,** e**.**employee\_name**;**

**+**----------------------+---------------------------+

| PROJECT\_NAME| EMPLOYEE\_NAME |

|------------------+-------------------------------|

| COVID-19 Vaccine | Maria Inverness |

| COVID-19 Vaccine | Terry Smith |

| Malaria Vaccine | Pat Wang |

| NULL | NewEmployee |

**+**---------------------+---------------------------+

ii.A full outer join lists all projects and all employees.

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **FULL** **OUTER** **JOIN** employees **AS** e

**ON** e**.**project\_ID **=** p**.**project\_ID

**ORDER** **BY** p**.**project\_name**,** e**.**employee\_name**;**

**+**------------------+-----------------+

| PROJECT\_NAME | EMPLOYEE\_NAME |

|------------------+-----------------|

| COVID-19 Vaccine | Maria Inverness |

| COVID-19 Vaccine | Terry Smith |

| Malaria Vaccine | Pat Wang |

| NewProject | NULL |

| NULL | NewEmployee |

**+**------------------+-----------------+

cross join:

The following query shows a cross join:

Syntax:

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **CROSS** **JOIN** employees **AS** e

**ORDER** **BY** p**.**project\_ID**,** e**.**employee\_ID**;**

**+**------------------+-----------------+

| PROJECT\_NAME | EMPLOYEE\_NAME |

|------------------+-----------------|

| COVID-19 Vaccine | Terry Smith |

| COVID-19 Vaccine | Maria Inverness |

| COVID-19 Vaccine | Pat Wang |

| COVID-19 Vaccine | NewEmployee |

| Malaria Vaccine | Terry Smith |

| Malaria Vaccine | Maria Inverness |

| Malaria Vaccine | Pat Wang |

| Malaria Vaccine | NewEmployee |

| NewProject | Terry Smith |

| NewProject | Maria Inverness |

| NewProject | Pat Wang |

| NewProject | NewEmployee |

**+**------------------+-----------------+

i.The output of a cross join can be made more useful by applying a filter in the WHERE clause:

Syntax:

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **CROSS** **JOIN** employees **AS** e

**WHERE** e**.**project\_ID **=** p**.**project\_ID

**ORDER** **BY** p**.**project\_ID**,** e**.**employee\_ID**;**

**+**------------------+-----------------+

| PROJECT\_NAME | EMPLOYEE\_NAME |

|------------------+-----------------|

| COVID-19 Vaccine | Terry Smith |

| COVID-19 Vaccine | Maria Inverness |

| Malaria Vaccine | Pat Wang |

**+**------------------+-----------------+

ii.The result of this cross join and filter is the same as the result of the following inner join:

Syntax:

**SELECT** p**.**project\_name**,** e**.**employee\_name

**FROM** projects **AS** p **INNER** **JOIN** employees **AS** e

**ON** e**.**project\_ID **=** p**.**project\_ID

**ORDER** **BY** p**.**project\_ID**,** e**.**employee\_ID**;**

**+**------------------------+-----------------------+

| PROJECT\_NAME| EMPLOYEE\_NAME|

|----------------------+-------------------------|

| COVID-19 Vaccine | Terry Smith |

| COVID-19 Vaccine | Maria Inverness |

| Malaria Vaccine | Pat Wang |

**+**-----------------------+------------------------+

**NATURAL** **JOIN:**

**Syntax:**

**SELECT** **\***

**FROM** projects **NATURAL** **JOIN** employees

**ORDER** **BY** employee\_ID**;**

**+**------------+------------------+-------------+-----------------+

| PROJECT\_ID | PROJECT\_NAME | EMPLOYEE\_ID | EMPLOYEE\_NAME |

|------------+------------------+-------------+-----------------|

| 1000 | COVID-19 Vaccine | 10000001 | Terry Smith |

| 1000 | COVID-19 Vaccine | 10000002 | Maria Inverness |

| 1001 | Malaria Vaccine | 10000003 | Pat Wang |

**+**------------+------------------+-------------+-----------------+

**ASOF JOIN :**

**An ASOF JOIN operation combines rows from two tables based on timestamp values that follow each other, precede each other, or match exactly.**

**Syntax:**

The following FROM clause syntax is specific to ASOF JOIN:

**FROM** <left\_table> **ASOF** **JOIN** <right\_table>

**MATCH\_CONDITION** ( <left\_table.timecol> <comparison\_operator> <right\_table.timecol> )

[ **ON** <table.col> = <table.col> [ **AND** ... ] | **USING** ( <column\_list> ) ]

Example:

**NULL-padded results**

Insert a new row into the trades table with a date that’s a day earlier than the existing rows in both trades and quotes:

Syntax:

**INSERT** **INTO** trades **VALUES(**'SNOW'**,**'2023-09-30 12:02:55.000'**,**3000**);**

**o/p:**

+----------------------------------+

| number of rows inserted |

|----------------------------------|

| 1 |

+----------------------------------+

4.rename the table names in snowflage

Syntax:

 ALTER TABLE old\_table\_name RENAME TO new\_table\_name;

Example:

**CREATE** **OR** **REPLACE** **TABLE** t1**(**a1 **number);**

**The following statement changes the name of the table to tt1:**

**ALTER** **TABLE** t1 **RENAME** **TO** tt1**;**

**SHOW** **TABLES** **LIKE** 'tt1'**;**