1.Create tables in snowflake

In snowflake there are three types of table creations

Syntax:

**CREATE** **OR** **REPLACE** **TABLE** some\_data **(**id **INTEGER,** **name** **VARCHAR);**

i.permanent

Example:

create table users (

id integer autoincrement, -- auto incrementing IDs

name varchar (100), -- variable string column

preferences variant, -- column used to store JSON type of data

created\_at timestamp

);

ii. temporary

You can also create temporary tables that will stick around for the duration of your session. This is helpful to break down your analysis into smaller pieces.

Example:

create temporary table inactive\_users (

id integer autoincrement,

name varchar(100) not null,

active boolean default false

);

iii. transient

Snowflake allows us to create transient tables which are a mix of permanent and temporary tables. They are used to store temporary data outside our session without having the need to implement a high level of data security and data recovery.

Example:

create transient table active\_users (

id integer autoincrement,

name varchar(100) not null,

active boolean default true

);

2.select queries

**Selecting all columns in the table**

**Syntax:**

**SELECT description, retail\_price, wholesale\_cost FROM ftable1;**

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

**| DESCRIPTION | RETAIL\_PRICE | WHOLESALE\_COST |**

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

**| bling | 14 | 6 |**

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

1.This example shows how to select all columns in employee\_table:

Syntax:

**SELECT** **\*** **FROM** employee\_table**;**

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

| EMPLOYEE\_ID | LAST\_NAME | FIRST\_NAME | DEPARTMENT\_ID |

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

| 101 | Montgomery | Pat | 1 |

| 102 | Levine | Terry | 2 |

| 103 | Comstock | Dana | 2 |

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

2.Selecting all columns with names that match a pattern

This example shows how to select all columns in employee\_table with names that contain id:

Syntax:

SELECT \* ILIKE '%id%' FROM employee\_table;

Output:

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

| EMPLOYEE\_ID | DEPARTMENT\_ID |

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

| 101 | 1 |

| 102 | 2 |

| 103 | 2 |

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

**3.Selecting all columns except one column**

This example shows how to select all columns in employee\_table except for the department\_id column:

Syntax:

**SELECT** **\*** **EXCLUDE** department\_id **FROM** employee\_table**;**

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

| EMPLOYEE\_ID | LAST\_NAME | FIRST\_NAME |

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

| 101 | Montgomery | Pat |

| 102 | Levine | Terry |

| 103 | Comstock | Dana |

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

4.**Selecting all columns and renaming one column**

This example shows how to select all columns in employee\_table and rename the department\_id column:

Syntax:

**SELECT** **\*** **RENAME** department\_id **AS** department **FROM** employee\_table**;**

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

| EMPLOYEE\_ID | LAST\_NAME | FIRST\_NAME | DEPARTMENT |

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

| 101 | Montgomery | Pat | 1 |

| 102 | Levine | Terry | 2 |

| 103 | Comstock | Dana | 2 |

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

**3.how to insert values one table to another table**

**Example:**

**Syntax:**

 INSERT INTO table2 (id, name, age) SELECT id, name, age FROM table1; This query will copy all rows from table1 into table2

Example:

**CREATE** **OR** **REPLACE** **TABLE** some\_data **(**id **INTEGER,** **name** **VARCHAR);**

**INSERT** **INTO** some\_data **(**id**,** **name)** **VALUES**

**(**1**,** 'a'**),**

**(**2**,** 'b'**);**

**4.replace query**

**Syntax:**

**REPLACE( <subject> , <pattern> [ , <replacement> ] )**

Example1:

Syntax:

**SELECT** **REPLACE(**'down'**,** 'down'**,** 'up'**);**

**Output:** **Replace the string down with the string up:**

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

| REPLACE('DOWN', 'DOWN', 'UP') |

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

| up |

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

Example2:

Replace the substring Athens in the string Vacation in Athens with the substring Rome:

Syntax:

**SELECT** **REPLACE(**'Vacation in Athens'**,** 'Athens'**,** 'Rome'**);**

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

| REPLACE('VACATION IN ATHENS', 'ATHENS', 'ROME') |

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

| Vacation in Rome |

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

Example3:

Replace the substring bc in the string abcd with an empty substring:

Syntax:

**SELECT** **REPLACE(**'abcd'**,** 'bc'**);**

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

| REPLACE('ABCD', 'BC') |

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

| ad |

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

Example4:

Replace the values in a table with new values.

Create and populate a table:

Syntax:

**CREATE** **OR** **REPLACE** **TABLE** replace\_example**(**

subject **VARCHAR(**10**),**

**pattern** **VARCHAR(**10**),**

replacement **VARCHAR(**10**));**

Replace the values in a table with new values.

Syntax:

**INSERT** **INTO** replace\_example **VALUES**

**(**'old car'**,** 'old car'**,** 'new car'**),**

**(**'sad face'**,** 'sad'**,** 'happy'**),**

**(**'snowman'**,** 'snow'**,** 'fire'**);**

**Example5:**

Replace strings in a value with a specified replacement:

Syntax:

**SELECT** subject**,**

**pattern,**

replacement**,**

**REPLACE(**subject**,** **pattern,** replacement**)** **AS** new

**FROM** replace\_example **ORDER** **BY** subject**;**

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

| SUBJECT | PATTERN | REPLACEMENT | NEW |

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

| old car | old car | new car | new car |

| sad face | sad | happy | happy face |

| snowman | snow | fire | fireman |

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

5.update query

**UPDATE**

Updates specified rows in the target table with new values.

**Syntax**

**UPDATE** <target\_table>

**SET** <col\_name> = <value> [ , <col\_name> = <value> , ... ]

[ **FROM** <additional\_tables> ]

[ **WHERE** <condition> ]

**Example1:**

Perform a standard update using two tables:

Syntax:

**UPDATE** t1

**SET** number\_column **=** t1**.**number\_column **+** t2**.**number\_column**,** t1**.**text\_column **=** 'ASDF'

**FROM** t2

**WHERE** t1**.**key\_column **=** t2**.**t1\_key **and** t1**.**number\_column **<** 10**;**

**Example:**

Update with join that produces nondeterministic results:

**select** **\*** **from** target**;**

o/p:

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

| K | V |

|---+----|

| 0 | 10 |

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

**Select** **\*** **from** src**;**

o/p:

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

| K | V |

|---+----|

| 0 | 11 |

| 0 | 12 |

| 0 | 13 |

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

Example3:

-- Following statement joins all three rows in src against the single row in target

**UPDATE** target

**SET** v **=** src**.**v

**FROM** src

**WHERE** target**.**k **=** src**.**k**;**

o/p:

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

| number of rows updated | number of multi-joined rows updated |

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

| 1 | 1 |

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