

Merge In SQL Server

The MERGE statement allows you to INSERT, UPDATE, or DELETE rows in a target table based on comparison with a source table in one single statement.

Basic Syntax

```
MERGE target_table AS T
USING source_table AS S
ON <match_condition>
WHEN MATCHED THEN
    UPDATE SET ...
WHEN NOT MATCHED BY TARGET THEN
    INSERT (...)
WHEN NOT MATCHED BY SOURCE THEN
    DELETE;
```

Example

Scenario

- Customer_Master → Active customers
- Customer_Staging → Latest customer list

Remove customers no longer present

Create Table Commands

```
CREATE TABLE Customer_Master
(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    City VARCHAR(30)
);
```

```
CREATE TABLE Customer_Staging
(
    CustomerID INT,
    CustomerName VARCHAR(50),
    City VARCHAR(30)
);
```

Insert Into commands

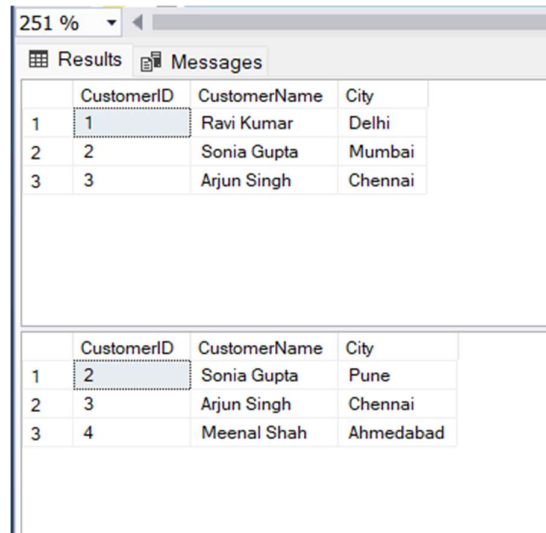
```
INSERT INTO Customer_Master VALUES
(1, 'Ravi Kumar', 'Delhi'),
(2, 'Sonia Gupta', 'Mumbai'),
(3, 'Arjun Singh', 'Chennai');
```

```
INSERT INTO Customer_Staging VALUES
(2, 'Sonia Gupta', 'Pune'), -- City changed
(3, 'Arjun Singh', 'Chennai');
```

```
(4, 'Meenal Shah', 'Ahmedabad'); -- New customer
```

```
select * from customer_master;  
select * from customer_staging;
```

Initial Data Output



The screenshot shows a database client interface with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying two tables. The first table, 'customer_master', has columns CustomerID, CustomerName, and City, with three rows: (1, Ravi Kumar, Delhi), (2, Sonia Gupta, Mumbai), and (3, Arjun Singh, Chennai). The second table, 'customer_staging', also has columns CustomerID, CustomerName, and City, with three rows: (2, Sonia Gupta, Pune), (3, Arjun Singh, Chennai), and (4, Meenal Shah, Ahmedabad).

	CustomerID	CustomerName	City
1	1	Ravi Kumar	Delhi
2	2	Sonia Gupta	Mumbai
3	3	Arjun Singh	Chennai

	CustomerID	CustomerName	City
1	2	Sonia Gupta	Pune
2	3	Arjun Singh	Chennai
3	4	Meenal Shah	Ahmedabad

Now using Merge to update master table.

```
MERGE Customer_Master AS T  
USING Customer_Staging AS S  
ON T.CustomerID = S.CustomerID  
  
WHEN MATCHED THEN  
    UPDATE SET  
        T.CustomerName = S.CustomerName,  
        T.City = S.City  
  
WHEN NOT MATCHED BY TARGET THEN  
    INSERT (CustomerID, CustomerName, City)  
    VALUES (S.CustomerID, S.CustomerName, S.City)  
  
WHEN NOT MATCHED BY SOURCE THEN  
    DELETE;
```

Resultant Table Data

```
select * from customer_master;  
select * from customer_staging;
```

Output

251 %

Results Messages

	CustomerID	CustomerName	City
1	2	Sonia Gupta	Pune
2	3	Arjun Singh	Chennai
3	4	Meenal Shah	Ahmedabad

	CustomerID	CustomerName	City
1	2	Sonia Gupta	Pune
2	3	Arjun Singh	Chennai
3	4	Meenal Shah	Ahmedabad

Observation

- Customer 2 → matched → **updated**
- Customer 4 → new → **inserted**
- Customer 1 → missing in source → **deleted**

Points to consider

1. Always use a **unique match condition**
2. Prefer running MERGE inside a **transaction**
3. Avoid MERGE on tables with heavy concurrent writes
4. Use OUTPUT clause to track changes (audit/logs)