**PROMPT : Give me a detailed documentation on delete and truncate statment in SQL with Examples.**

Both DELETE and TRUNCATE are SQL commands used to remove data from a table, but they operate very differently. DELETE is a Data Manipulation Language (DML) command that removes rows one by one and can be filtered, while TRUNCATE is a Data Definition Language (DDL) command that quickly removes all rows by deallocating the data pages.

## DELETE Statement

The DELETE statement is used to remove existing records from a table. It's a DML command, meaning it works with the data itself.

### Key Characteristics

* **Row-by-Row Operation**: DELETE removes rows one at a time and records an entry in the transaction log for each deleted row. This can make it slower than TRUNCATE for large tables.
* **WHERE Clause**: You can use a WHERE clause to specify which rows to remove. If you omit the WHERE clause, all rows in the table will be deleted.
* **Transactional**: Since it's a DML operation, a DELETE statement can be rolled back. You can wrap it in a transaction and undo it if needed.
* **Triggers**: DELETE operations fire ON DELETE triggers associated with the table because it operates on individual rows.
* **Identity Columns**: It does not reset the value of an identity column. If you delete all rows and then insert a new one, the new row will get an identity value continuing from where the last one left off.
* **Locks**: It places a lock on each row it affects, which can lead to more locking overhead compared to TRUNCATE.

### Syntax

To delete specific rows:

SQL

DELETE FROM table\_name  
WHERE condition;

To delete all rows:

SQL

DELETE FROM table\_name;

### Examples

Let's assume we have a table called Employees:

| EmployeeID | FirstName | LastName | Department |
| --- | --- | --- | --- |
| 1 | John | Doe | HR |
| 2 | Jane | Smith | IT |
| 3 | Peter | Jones | IT |
| 4 | Mary | Williams | Finance |

**1. Deleting Specific Rows**

To delete all employees from the 'IT' department:

SQL

DELETE FROM Employees  
WHERE Department = 'IT';

After this operation, the table will look like this:

| EmployeeID | FirstName | LastName | Department |
| --- | --- | --- | --- |
| 1 | John | Doe | HR |
| 4 | Mary | Williams | Finance |

**2. Deleting All Rows**

To delete all records from the Employees table:

SQL

DELETE FROM Employees;

This will leave the table empty, but its structure, indexes, and constraints will remain. If the EmployeeID was an auto-incrementing identity column, the next new record would have an EmployeeID of 5, not 1.

## TRUNCATE Statement

The TRUNCATE TABLE statement is used to remove all rows from a table quickly and efficiently. It's a DDL command, meaning it modifies the table's definition and structure at a low level.

### Key Characteristics

* **Deallocates Data Pages**: Instead of deleting rows one by one, TRUNCATE works by deallocating the data pages used to store the table's data. This results in minimal logging and makes it extremely fast.
* **No WHERE Clause**: You cannot use a WHERE clause with TRUNCATE. It always removes **all** rows from the table.
* **Minimally Logged**: The operation is minimally logged. While it can sometimes be rolled back (depending on the specific SQL dialect and transaction context), it doesn't log each row's deletion.
* **No Triggers**: TRUNCATE does not fire DELETE triggers because it doesn't perform individual row deletions.
* **Resets Identity**: It resets any identity columns in the table back to their seed value (usually 1).
* **Table Lock**: It typically places a schema lock on the table, which is more restrictive than a row-level lock.

### Syntax

SQL

TRUNCATE TABLE table\_name;

### Example

Using the same Employees table:

| EmployeeID | FirstName | LastName | Department |
| --- | --- | --- | --- |
| 1 | John | Doe | HR |
| 2 | Jane | Smith | IT |
| 3 | Peter | Jones | IT |
| 4 | Mary | Williams | Finance |

To remove all data from the Employees table:

SQL

TRUNCATE TABLE Employees;

The table is now empty. If a new row is inserted, its EmployeeID will be reset to 1 (its original seed value).

## Summary of Differences

| Feature | DELETE | TRUNCATE |
| --- | --- | --- |
| **Command Type** | DML (Data Manipulation Language) | DDL (Data Definition Language) |
| **Filtering** | Can use a WHERE clause to remove specific rows. | Cannot use a WHERE clause; always removes all rows. |
| **Performance** | Slower, as it removes rows one by one. | Much faster, as it deallocates data pages. |
| **Transaction Log** | Logs each row deletion individually. | Minimally logged (logs the page deallocations). |
| **Rollback** | Can be easily rolled back. | May not be possible to roll back, depending on the system. |
| **Triggers** | Fires DELETE triggers for each affected row. | Does not fire DELETE triggers. |
| **Identity Column** | Does not reset the identity value. | Resets the identity value to its original seed. |
| **Locks** | Places row-level locks. | Places a table-level lock. |
| **Foreign Keys** | Can be used with tables that have foreign key constraints. | Cannot be used on a table referenced by a foreign key constraint. |

## When to Use Which? 🤔

* Use **DELETE** when:
  + You need to remove only **specific rows** from a table.
  + You need the operation to fire triggers.
  + The table is referenced by foreign keys.
  + You want to be able to **roll back** the change with certainty.
* Use **TRUNCATE** when:
  + You want to remove **all rows** from a large table quickly.
  + You want to **reset the identity column**.
  + You do not need to fire DELETE triggers.
  + The operation's speed is more critical than individual row logging.