SQL TRUNCATE TABLE

A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

TRUNCATE TABLE Vs DELETE TABLE

Truncate table is faster and uses lesser resources than DELETE TABLE command.

TRUNCATE TABLE Vs DROP TABLE

Drop table command can also be used to delete complete table but it deletes table structure too. TRUNCATE TABLE doesn't delete the structure of the table.

What is TRUNCATE Command?

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extent of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing several integrity-enforcing mechanisms.

TRUNCATE TABLE table_name;

Differences Between DROP and TRUNCATE

The key differences between DROP and TRUNCATE statements are explained in the following table:

DROP	TRUNCATE
In the drop table data and its definition is deleted with their full structure.	It preserves the structure of the table for further use exist but deletes all the data.
Drop is used to eliminate existing complications and fewer complications in the whole database from the table.	Truncate is used to eliminate the tuples from the table.
Integrity constraints get removed in the DROP command.	Integrity constraint doesn't get removed in the Truncate command.
Since the structure does not exist, the View of the table does not exist in the Drop command.	Since the structure exists, the View of the table exists in the Truncate command.
Drop query frees the table space complications from memory.	This query does not free the table space from memory.
It is slow as there are so many complications compared to the TRUNCATE command.	It is fast as compared to the DROP command as there are fewer complications.

Step 1: Create a Table

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(50),
    Category VARCHAR(50),
    Price DECIMAL(10, 2),
    Stock INT
);
```

Step 2: Insert Sample Data

```
INSERT INTO Products (ProductID, ProductName, Category, Price, Stock) VALUES
(1, 'Laptop', 'Electronics', 1200.00, 30),
(2, 'Chair', 'Furniture', 150.00, 50),
(3, 'Desk', 'Furniture', 300.00, 20),
(4, 'Headphones', 'Electronics', 100.00, 100);
```

Step 3: Basic TRUNCATE TABLE Statement

-- Truncate the Products table

```
TRUNCATE TABLE Products;
```

Step 4: TRUNCATE TABLE vs. DELETE (Advanced Considerations)

Differences and Advanced Use-Cases:

- **Performance**: TRUNCATE TABLE is usually faster than DELETE because it doesn't generate individual row delete logs.
- **Resetting Identity**: It resets any auto-increment counters associated with the table.
- **Constraints Handling**: TRUNCATE cannot be used when there are foreign key constraints unless they are removed or disabled.

Step 5: TRUNCATE TABLE with RESTART IDENTITY (Advanced)

When working with tables that have auto-increment columns, you can use the RESTART IDENTITY option to reset the counter:

-- Truncate the table and restart identity

```
TRUNCATE TABLE Products RESTART IDENTITY;
```

This resets any identity columns to their seed value (e.g., ProductID starting from 1 again).

Step 6: TRUNCATE TABLE with CASCADE (Advanced)

When tables have foreign key dependencies, you can use CASCADE to truncate the referenced tables automatically. For this to work, foreign key constraints must be defined with ON DELETE CASCADE.

-- Creating another table Orders with a foreign key to Products

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    ProductID INT,
    Quantity INT,
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE
);
```

-- Insert sample data into Orders

```
INSERT INTO Orders (OrderID, ProductID, Quantity) VALUES
(1, 1, 2),
(2, 2, 5);
```

-- Truncate Products table along with Orders due to ON DELETE CASCADE

```
TRUNCATE TABLE Products CASCADE;
```

Step 7: Permissions and Safety (Advanced)

- **Permissions**: Only database administrators or users with the appropriate privileges can execute TRUNCATE TABLE.
- **Irreversible**: Unlike DELETE, TRUNCATE cannot be rolled back once committed (in some databases), especially when outside of a transaction.