SQL Cleanup Crew

Mastering SQL with DELETE Statement

Section 1: Learn

What is the DELETE Statement?

The DELETE statement is one of the core Data Manipulation Language (DML) commands in SQL. It is used to permanently remove one or more rows from a table. Unlike TRUNCATE, it can be used with conditions and participates in transactions, meaning changes can be rolled back if needed.

Why Use DELETE?

- To remove incorrect or duplicate data entries
- To clean up records that are no longer relevant (e.g., old logs, expired entries)
- To enforce business rules by cleaning inconsistent or outdated rows
- As part of routine data archival or purging processes

How DELETE Works:

1. The system scans the specified table.
2. Each row is evaluated against the WHERE condition.
3. Matching rows are removed from the table.

If no WHERE clause is provided, **all rows** in the table will be deleted.

Syntax Overview:

```
DELETE FROM table_name
WHERE condition;
```

You can also delete all rows without a condition:

```
DELETE FROM table_name;
```

This keeps the table structure intact but removes all records.

Important Points:

- **Transactions**: DELETE operations can be wrapped in transactions to allow rollback.
- **Referential Integrity**: If foreign keys are defined, deletes may fail unless cascading is enabled.
- **Performance**: Large deletes can be slow; use batching where needed.

Basic Example:

```
DELETE FROM employees

WHERE employee_id = 105;
```

Section 2: Practice

1. Delete a Single Row

```
DELETE FROM books

WHERE book_id = 12;
```

This removes the record where book_id equals 12.

2. Delete Multiple Rows Based on Condition

```
DELETE FROM students

WHERE grade = 'F';
```

Deletes all students who received a failing grade.

3. Delete All Rows from a Table

```
DELETE FROM archive_logs;
```

This clears the archive_logs table, keeping the structure for future inserts.

4. Conditional DELETE with Multiple Criteria

```
DELETE FROM orders

WHERE order_date < '2023-01-01' AND status = 'Cancelled';
```

This removes all canceled orders placed before 2023.

5. DELETE with Subquery

```
DELETE FROM customers

WHERE customer_id IN (

  SELECT customer_id

  FROM orders

  WHERE order_total = 0

);
```

Deletes all customers whose orders had zero total amount.

6. DELETE with EXISTS

```
DELETE FROM customers

WHERE EXISTS (

  SELECT 1 FROM orders

  WHERE orders.customer_id = customers.customer_id

  AND orders.order_date < '2022-01-01'
);
```

Removes customers with orders older than 2022.

7. DELETE with JOIN (MySQL-style)

```
DELETE orders
FROM orders
JOIN customers ON orders.customer_id = customers.customer_id
WHERE customers.status = 'Inactive';
```

Removes orders placed by inactive customers.

Section 3: Know More

Best Practices

- Always **preview** rows to be deleted using a SELECT with the same condition.
- Use transactions if supported:

```
BEGIN;
DELETE FROM logs WHERE log_date < '2023-01-01';
ROLLBACK; -- or COMMIT
```

- Avoid deleting in one large operation on large datasets—use batch deletion with LIMIT.

Frequently Asked Questions (FAQs)

1. **What happens if I omit the WHERE clause?**
    - All records will be deleted from the table.

2. **Can I undo a DELETE operation?**
    - Yes, but only within a transaction before it is committed. Use ROLLBACK to undo.
3. **Is DELETE the same as TRUNCATE?**
    - No. DELETE allows conditional row removal and is transactional. TRUNCATE deletes all rows and cannot be rolled back in many DBMS.
4. **Can I use DELETE with JOIN?**
    - Yes, especially in MySQL and PostgreSQL. It's used to delete rows based on matching records in another table.
5. **How do I safely test DELETE conditions?**
    - Run a SELECT with the same WHERE clause first:

```
SELECT * FROM users WHERE signup_date < '2020-01-01';
```

Then, confirm with:

```
DELETE FROM users WHERE signup_date < '2020-01-01';
```

6. **Does DELETE free up disk space immediately?**
    - It depends on the DBMS. Some free space instantly, while others require VACUUM or optimization commands.
