SQL Temporary Tables, Views and CTEs

LESSON

Generate a quiz about this unit

Summarize this unit

Have any doubt about this content? Ask!

Learning Objectives

By the end of this lesson, you will be able to:

- Identify the characteristics and uses of temporary tables in SQL.
- Differentiate between permanent and temporary tables.
- Describe the benefits and basic structure of views in SQL.
- Construct and apply Common Table Expressions (CTEs) to achieve modular querying.

Temporary Tables

Temporary tables, which exist only during a session or specific procedure, are short-lived structures ideal for storing intermediate results needed temporarily. Temporary tables, which exist only during a session or specific procedure, are short-lived structures ideal for storing intermediate results needed temporarily.

Basic Syntax

```
CREATE TEMPORARY TABLE temp_table_name (column1 datatype, co   Copy   yp
```

Explain this code

Key Points

- **Scope**: They are only visible to the session that created them.
- **Naming**: While not mandatory, it's common to prefix temporary tables with `temp_` or use a similar convention for clarity.

Example

Suppose we have the following table.

Table: employees

employee_name	salary
John Doe	60000
Jane Smith	55000
Lucy Liu	48000
Alan Walker	52000

In order to create a temporary table to store top-earning employees, we can use the following query: In order to create a temporary table to store top-earning employees, we can use the following query:

```sql
CREATE TEMPORARY TABLE top_employees AS
SELECT employee_name, salary
FROM employees
WHERE salary > 50000;
```

Copy

Explain this code

Resulting Temporary Table: top_employees

employee_name	salary
John Doe	60000
Jane Smith	55000
Alan Walker	52000

Views

A view is a virtual table based on the result set of a SQL statement. It does not store data physically but rather provides a way to access data from one or more tables in a simplified or aggregated manner.

Basic Syntax

```sql
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM tablename
WHERE condition;
```

Copy

Explain this code

Key Points

- **Simplicity**: Views can abstract away the complexity of underlying table structures.
- **Security**: They can restrict access to specific rows or columns.
- **Consistency**: Views ensure consistent results for recurring queries.
- **Read-Only**: Most views are read-only, but some databases support “updatable views.”

Example

We’ll continue using the table employees defined above. We’ll continue using the table employees defined above.

In order to create a view of employees with high salaries, we can use the following query: In order to create a view of employees with high salaries, we can use the following query:

```sql
CREATE VIEW high_salary_employees AS
SELECT employee_name, salary
FROM employees
WHERE salary > 50000;
```

Copy

Resulting View: high_salary_employees

employee_name	salary
John Doe	60000
Jane Smith	55000
Alan Walker	52000

Common Table Expressions (CTEs)

A CTE provides a temporary result set that can be referenced within a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. CTEs make complex queries more readable and modular.

Basic Syntax

```
WITH cte_name (column_name1, column_name2, ...) AS (
    SELECT statement
)
SELECT * FROM cte_name;
```

Copy

Explain this code

Key Points

- **Readability**: CTEs structure and segment complex queries for better readability.
- **Recursion**: Some databases support recursive CTEs, enabling hierarchical or iterative querying.
- **Scope**: A CTE is only available to the query in which it is defined.

Example

We'll continue with the table employees defined above.

To calculate the average salary of employees using a CTE, we can use the following query: To calculate the average salary of employees using a CTE, we can use the following query:

```
    SELECT AVG(salary) AS average
    FROM employees
)
SELECT employee_name, salary
FROM employees
WHERE salary > (SELECT average FROM avg_salary);
```

Explain this code

Resulting Set from CTE-based Query

employee_name	salary
John Doe	60000
Jane Smith	55000

Note: The average salary is calculated to be 53,750. Therefore only employees with salaries greater than this amount will be listed in the resulting set.

When to Use Which?

1. **Temporary Tables**: Use when you have large intermediate results that need to be referenced multiple times in various queries during a session.
2. **Views**: Use when you want a persistent “saved” query that abstracts away complexity or provides restricted access to data.
3. **Views**: Use when you want a persistent “saved” query that abstracts away complexity or provides restricted access to data.
4. **CTEs**: Use for breaking down complex queries into simpler parts for a single query execution, especially when recursion is involved.
5. **Subqueries**: Use for on-the-fly computations within a query when you don’t need to reuse the result.

Summary

Temporary tables, views, and CTEs are powerful SQL constructs that enhance flexibility, readability, and modularity in database querying. Whether you’re simplifying complex queries with CTEs, abstracting database structure with views, or using temporary tables for transient data manipulation, understanding these concepts is crucial for advanced SQL operations. Temporary tables, views, and CTEs are powerful SQL constructs that enhance flexibility, readability, and modularity in database understanding these concepts is crucial for advanced SQL operations.

PREVIOUS

← Group Lab Review

NEXT

SQL Temporary Tables, Views and CTEs Hands On →