**Prompt : Give me a documentation on update statement in postgres sql.**

Absolutely! Here's a well-structured documentation-style overview of the UPDATE statement in PostgreSQL, including syntax, usage, constraints, and examples.

**PostgreSQL UPDATE Statement Documentation**

🧾 Purpose

The UPDATE statement is used to modify existing records in a table. You can update one or more columns, conditionally or across all rows.

📚 **Syntax**

**UPDATE table\_name**

**SET column1 = value1,**

**column2 = value2,**

**...**

**WHERE condition;**

**table\_name: The name of the table to update.**

SET: Specifies the columns and new values.

WHERE: Filters which rows to update (optional but recommended).

⚠️ Without a WHERE clause, all rows in the table will be updated.

🧠 Key Concepts

✅ Column Updates

You can update:

A single column

Multiple columns at once

Columns based on expressions or other column values

**🔒 Constraints**

Updates must respect constraints like NOT NULL, UNIQUE, CHECK, and FOREIGN KEY.

Violating constraints will result in errors.

🔁 **Transactions**

Updates can be wrapped in transactions for rollback safety:

BEGIN;

UPDATE ...;

COMMIT;

🔍 RETURNING Clause

Returns the updated rows:

UPDATE employees

SET salary = salary \* 1.10

WHERE department = 'Sales'

RETURNING \*;

**📘 Example 1: Update a Single Column**

sql

UPDATE employees

SET salary = 80000

WHERE emp\_id = 101;

🔍 What It Does:

Sets the salary of employee with ID 101 to 80,000.

**📘 Example 2: Update Multiple Columns**

sql

UPDATE employees

SET department = 'Marketing',

salary = salary + 5000

WHERE emp\_id = 102;

🔍 What It Does:

Changes department and increases salary for employee 102.

**📘 Example 3: Update All Rows (Use with Caution)**

sql

UPDATE employees

SET is\_active = FALSE;

🔍 What It Does:

Marks all employees as inactive.

**📘 Example 4: Conditional Update Using Subquery**

sql

UPDATE employees

SET salary = salary + 10000

WHERE emp\_id IN (

SELECT emp\_id FROM performance WHERE rating = 'Excellent'

);

🧪 Best Practices

Always use a WHERE clause unless you intend to update all rows.

Use RETURNING to verify changes.

Wrap critical updates in transactions.

Test updates with SELECT before applying them.

**🔄 UPDATE with Triggers, Joins, and CTEs in PostgreSQL**

⚡ 1. UPDATE with Triggers

🧠 What Are Triggers?

Triggers are database callbacks that automatically execute before or after an INSERT, UPDATE, or DELETE on a table.

📘 Example: AFTER UPDATE Trigger

sql

CREATE TABLE audit\_log (

id SERIAL PRIMARY KEY,

emp\_id INT,

old\_salary NUMERIC,

new\_salary NUMERIC,

changed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE OR REPLACE FUNCTION log\_salary\_change()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO audit\_log(emp\_id, old\_salary, new\_salary)

VALUES (OLD.emp\_id, OLD.salary, NEW.salary);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER salary\_update\_trigger

AFTER UPDATE ON employees

FOR EACH ROW

WHEN (OLD.salary IS DISTINCT FROM NEW.salary)

EXECUTE FUNCTION log\_salary\_change();

🔍 What It Does:

Automatically logs salary changes into audit\_log whenever an employee's salary is updated.

**🔗 2. UPDATE with JOIN**

PostgreSQL supports UPDATE with FROM clause to join tables.

📘 Example: Update Using Join

sql

UPDATE employees e

SET salary = e.salary + b.bonus

FROM bonuses b

WHERE e.emp\_id = b.emp\_id;

🔍 What It Does:

Increases each employee’s salary by their corresponding bonus from the bonuses table.

✅ You can join multiple tables, use aliases, and apply complex conditions.

**🧵 3. UPDATE with CTEs (Common Table Expressions)**

CTEs allow you to define temporary result sets that can be referenced in the UPDATE.

📘 Example: UPDATE with CTE

sql

WITH top\_performers AS (

SELECT emp\_id

FROM performance

WHERE rating = 'Excellent'

)

UPDATE employees

SET salary = salary \* 1.20

WHERE emp\_id IN (SELECT emp\_id FROM top\_performers);

🔍 What It Does:

Boosts salary by 20% for employees identified as top performers.

**🧪 Bonus: UPDATE with CTE + RETURNING**

sql

WITH updated AS (

UPDATE employees

SET salary = salary \* 1.10

WHERE department = 'Sales'

RETURNING emp\_id, salary

)

SELECT \* FROM updated;

🔍 What It Does:

Updates salaries and immediately returns the affected rows.

🧠 Summary Table

**Feature Use Case Benefits**

**Triggers Automate logging or validation Ensures consistency and auditing**

**Joins Update based on related table data Powerful multi-table updates**

**CTEs Modular, readable conditional updates Cleaner logic and reuse**