**Promt : Give in detailed documentation of update command and syntax. Explain in detail with 2 examples as a beginner in postgresql.**

**UPDATE Command**

The **UPDATE** command is used to modify existing records in a table. It's a crucial part of **Data Manipulation Language (DML)**, allowing you to change data without having to delete and re-insert rows.

**Syntax**

**The basic syntax for the UPDATE command is:**

SQL

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

* UPDATE table\_name: Specifies the table you want to modify.
* SET column1 = value1, ...: This clause lists the columns you want to change and the new values you want to assign to them. You can update one or more columns at once.
* WHERE condition: This is the most important part of the statement. The WHERE clause specifies which rows should be updated. If you omit the WHERE clause, **all rows in the table will be updated**.

**Examples for Beginners**

Let's use our sample employees table for these examples.

SQL

-- Assume this table already exists

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

department VARCHAR(50),

salary NUMERIC(10, 2));

-- Let's populate the table with some data

INSERT INTO employees (first\_name, last\_name, department, salary) VALUES

('John', 'Doe', 'Sales', 50000.00),

('Jane', 'Smith', 'Marketing', 60000.00),

('Peter', 'Jones', 'Engineering', 75000.00);

**Example 1: Updating a single record ✍️**

Suppose John Doe gets a promotion and moves to the Engineering department, and his salary increases. We can update his record using the employee\_id to uniquely identify him.

SQL

UPDATE employees

SET department = 'Engineering', salary = 65000.00

WHERE employee\_id = 1;

* UPDATE employees: We are targeting the employees table.
* SET department = 'Engineering', salary = 65000.00: We are changing two columns: department and salary.
* WHERE employee\_id = 1: The WHERE clause ensures that **only the employee with an id of 1** (John Doe) is updated. The other employees' records will remain unchanged.

**Example 2: Updating multiple records 📊**

Imagine all employees in the "Sales" department are receiving a 10% raise. Instead of updating each person individually, you can update all of them at once.

SQL

UPDATE employees

SET salary = salary \* 1.10

WHERE department = 'Sales';

* SET salary = salary \* 1.10: This is a great example of how UPDATE can perform calculations. We are setting the new salary to be the current salary multiplied by 1.10 (a 10% increase).
* WHERE department = 'Sales': This condition ensures that **only employees in the 'Sales' department** are affected by the update. The other departments will not receive a salary change.

**Important Notes:**

* You must **use the correct data type** while assigning new values.
* Always use **single quotes** for strings and dates.
* Always include a WHERE clause unless you intentionally want to update every row.
* You can update **one column** or **multiple columns** in one statement.

**Common Mistakes to Avoid**

**❌ Forgetting WHERE Clause**

sql

CopyEdit

UPDATE students

SET age = 25;

➡️ This will set **age = 25 for all students**. Be very careful!

**❌ Using Wrong Data Types**

sql

CopyEdit

UPDATE students

SET age = 'twenty';

➡️ You’ll get a **data type mismatch error**. Always use numbers for INT, strings in quotes, dates in 'YYYY-MM-DD' format.

**❌ Violating Constraints**

sql

CopyEdit

UPDATE students

SET email = 'rahul@gmail.com'

WHERE student\_id = 3;

➡️ If another student already has this email, it will **violate the UNIQUE constraint**, and PostgreSQL will throw an error.

**Best Practices**

* ✅ Always use the WHERE clause.
* ✅ Use the **primary key** (student\_id, product\_id, etc.) for accurate updates.
* ✅ Double-check your query before running it — especially when working in real databases.