Inserting data into a database table is a fundamental operation in SQL, enabling you to add records to your database. This article will explain the concepts and provide detailed examples of how to insert data into a table.

Creating a Table

Before inserting data, we need a table to store the data. Here's an example of creating an Employees table:

```
USE employees;
DROP TABLE IF EXISTS Employees;
CREATE TABLE IF NOT EXISTS Employees (
  employee_id SERIAL PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  department_id INT,
  hire_date DATE
);
SELECT * FROM Employees;
```

In this example, we create an Employees table with the following columns:

- employee_id: A unique identifier for each employee, which is auto-incremented.
- first_name: The first name of the employee.
- last_name: The last name of the employee.
- department_id: The ID of the department the employee belongs to.
- hire_date: The date the employee was hired.

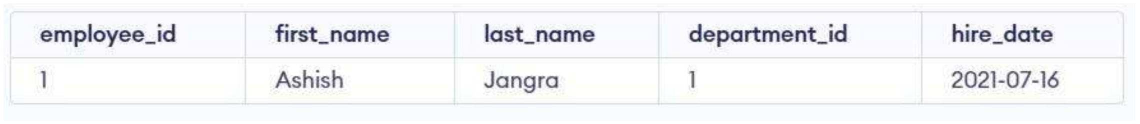
Inserting Data

1. Inserting a Single Row

To insert a single row into the Employees table, you use the INSERT INTO statement with the column names and the values to be inserted.

```
INSERT INTO Employees (employee_id, first_name, last_name, department_id,
hire_date)
VALUES (1, 'Ashish', 'Jangra', 1, '2021-07-16');
SELECT * FROM Employees;
```

This inserts one record into the Employees table with the specified values.



employee_id	first_name	last_name	department_id	hire_date
1	Ashish	Jangra	1	2021-07-16

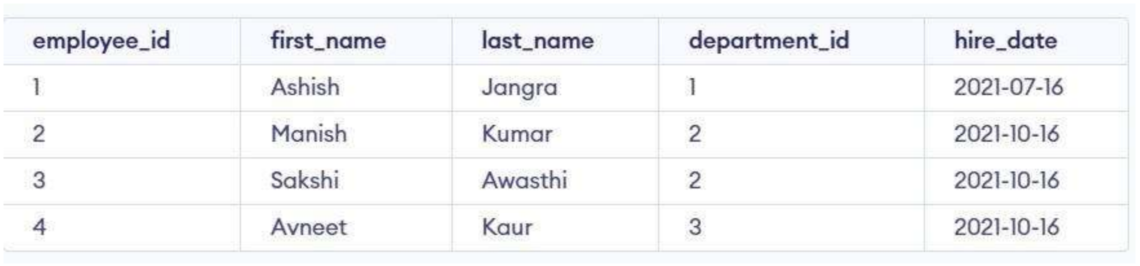
Output

Inserting Multiple Rows

You can also insert multiple rows in a single INSERT INTO statement by separating the values for each row with a comma.

```
INSERT INTO Employees (employee_id, first_name, last_name, department_id,
hire_date)
VALUES
    (2, 'Manish', 'Kumar', 2, '2021-10-16'),
    (3, 'Sakshi', 'Awasthi', 2, '2021-10-16'),
    (4, 'Avneet', 'Kaur', 3, '2021-10-16');
SELECT * FROM Employees;
```

This example adds three new rows to the Employees table.



employee_id	first_name	last_name	department_id	hire_date
1	Ashish	Jangra	1	2021-07-16
2	Manish	Kumar	2	2021-10-16
3	Sakshi	Awasthi	2	2021-10-16
4	Avneet	Kaur	3	2021-10-16

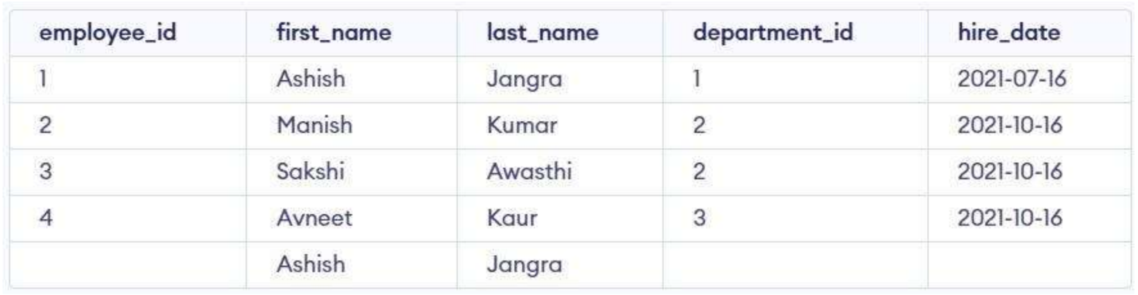
Output

Adding Partial Data

If you don't have values for all columns, you can insert partial data. Columns not listed in the INSERT INTO statement will be set to their default value or NULL if no default is specified.

```
INSERT INTO Employees (first_name, last_name)
VALUES ('Ashish', 'Jangra');
SELECT * FROM Employees;
```

In this case, only the first_name and last_name are provided. The other columns will use their default values or NULL.



employee_id	first_name	last_name	department_id	hire_date
1	Ashish	Jangra	1	2021-07-16
2	Manish	Kumar	2	2021-10-16
3	Sakshi	Awasthi	2	2021-10-16
4	Avneet	Kaur	3	2021-10-16
	Ashish	Jangra		

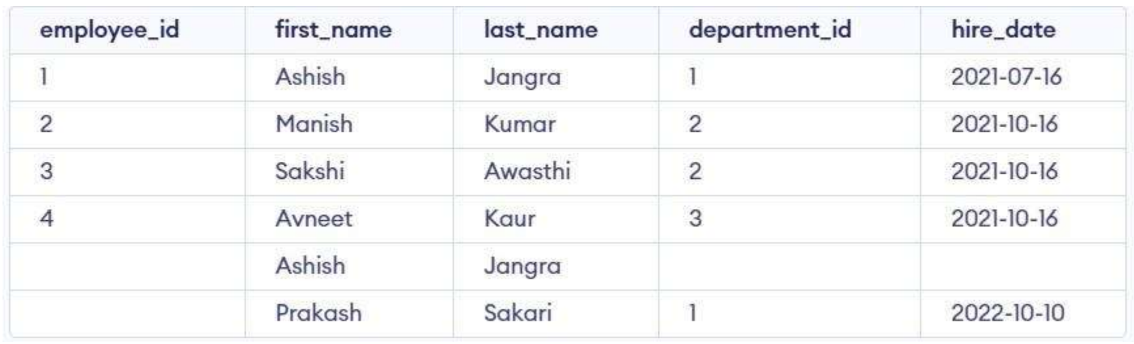
Output

Inserting Data in a Different Order

The order of the columns in the INSERT INTO statement does not have to match the order of the columns in the table. You can specify the columns in any order.

```
INSERT INTO Employees (department_id, hire_date, last_name, first_name)
VALUES (1, '2022-10-10', 'Sakari', 'Prakash');
SELECT * FROM Employees;
```

Here, the columns are specified in a different order, and the values are inserted accordingly.



employee_id	first_name	last_name	department_id	hire_date
1	Ashish	Jangra	1	2021-07-16
2	Manish	Kumar	2	2021-10-16
3	Sakshi	Awasthi	2	2021-10-16
4	Avneet	Kaur	3	2021-10-16
	Ashish	Jangra		
	Prakash	Sakari	1	2022-10-10

Output

Adding Data Without Order

You can also add data without specifying the column names if you provide values for all columns in the exact order they are defined in the table.

```
INSERT INTO Employees
VALUES (11, 'Ashish', 'Jangra', 5, '2024-10-10');
SELECT * FROM Employees;
```

This statement inserts values directly into all columns based on their order in the table definition.

Adding Current Date and Default Values

You can create a table with default values for some columns, such as the current timestamp or a default status.

```
DROP TABLE IF EXISTS ExampleTable;
CREATE TABLE IF NOT EXISTS ExampleTable (
   employee_id SERIAL PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   hire_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   status VARCHAR(20) DEFAULT 'Active'
);
SELECT * FROM ExampleTable;


INSERT INTO ExampleTable (name) VALUES ('Ashish');
SELECT * FROM ExampleTable;

INSERT INTO ExampleTable (name) VALUES ('Ashish'), ('Manish'), ('Parag');
SELECT * FROM ExampleTable;


INSERT INTO ExampleTable (name, status) VALUES ('Ashish', 'Inactive'), ('Manish',
'Inactive'), ('Parag', 'Inactive');
SELECT * FROM ExampleTable;
```

In this example, the **hire_date** column uses the current timestamp by default, and the status column defaults to **'Active'.** When inserting data, you can omit these columns if you want to use the default values.

Conclusion

Inserting data into a table is a fundamental aspect of working with databases. Understanding how to use the INSERT INTO statement with different variations allows you to add records efficiently and effectively. Whether you're inserting a single row, multiple rows, partial data, or using default values, mastering these techniques is essential for managing and manipulating data in SQL databases.