Normalization Techniques

What is Normalization? Normalization is a database design technique used to minimize data redundancy and improve data integrity. It involves organizing data into multiple related tables to eliminate duplicate data and ensure consistency.

Why is Normalization Important?

* Reduces redundancy and avoids duplicate data.
* Improves data integrity and consistency.
* Simplifies data maintenance and updates.
* Enhances query performance in relational databases.

Normalization Forms:

1. **First Normal Form (1NF):** Ensures that all columns contain atomic (indivisible) values and each row has a unique identifier (primary key).
2. **Second Normal Form (2NF):** Achieved by removing partial dependencies, ensuring that all non-key attributes are fully dependent on the primary key.
3. **Third Normal Form (3NF):** Eliminates transitive dependencies by ensuring that non-key attributes depend only on the primary key.
4. **Boyce-Codd Normal Form (BCNF):** A stricter version of 3NF where every determinant is a candidate key.
5. **Fourth Normal Form (4NF):** Eliminates multi-valued dependencies, ensuring that a table does not contain multiple independent relationships.
6. **Fifth Normal Form (5NF):** Ensures that a table is split into smaller tables without losing any data, preventing join dependencies.
7. **Sixth Normal Form (6NF):** Rarely used, focuses on handling temporal databases by decomposing tables even further.

By applying these normalization techniques, databases achieve efficiency, consistency, and better scalability.

SQL Joins

What are SQL Joins? SQL Joins are used to retrieve data from multiple tables based on related columns. They help in combining rows from two or more tables in a meaningful way.

Types of Joins:

1. **INNER JOIN**: Retrieves records that have matching values in both tables. If there is no match, the row is not included in the result.
   * Syntax:

SELECT columns FROM table1

INNER JOIN table2 ON table1.column = table2.column;

1. **LEFT JOIN (LEFT OUTER JOIN)**: Retrieves all records from the left table and only the matching records from the right table. If there is no match, NULL values are returned for columns from the right table.
   * Syntax:

SELECT columns FROM table1

LEFT JOIN table2 ON table1.column = table2.column;

1. **RIGHT JOIN (RIGHT OUTER JOIN)**: Retrieves all records from the right table and only the matching records from the left table. If there is no match, NULL values are returned for columns from the left table.
   * Syntax:

SELECT columns FROM table1

RIGHT JOIN table2 ON table1.column = table2.column;

By using these join types, data from multiple tables can be efficiently queried and analyzed based on relationships.