SQL Crash Course (1)

`SELECT` - this identifies which columns to return in the result set

`FROM` and `JOIN` -

`WHERE` - this filters the rows

Example Tables

Imagine we have two simple tables in a database:

1. `students` **Table:**

id	name	age	grade
1	Alice	14	9
2	Bob	15	10
3	Charlie	14	9

1. `subjects` **Table:**

id	student_id	subject	score
1	1	Math	95
2	1	Science	90
3	2	Math	88
4	3	Literature	93

SQL Clauses

SELECT

- **Purpose:** Specifies the columns that you want to retrieve from the database.

- **Example:** `SELECT name, age FROM students;`
- **Explanation:** This retrieves the `name` and `age` of all students.

FROM

- **Purpose:** Indicates the table from which to retrieve the data.
- **Example:** `SELECT name FROM students;`
- **Explanation:** This specifies that the `name` should be retrieved from the `students` table.

WHERE

- **Purpose:** Filters rows based on a specified condition.
- **Example:** `SELECT name FROM students WHERE age = 14;`
- **Explanation:** This retrieves the names of students who are 14 years old.

JOIN

- **Purpose:** Combines rows from two or more tables, based on a related column between them.
- **Example:** `SELECT students.name, subjects.subject FROM students JOIN subjects ON students.id = subjects.student_id;`
- **Explanation:** This retrieves a list of students and the subjects they are enrolled in by matching `students.id` with `subjects.student_id`.

GROUP BY

- **Purpose:** Groups rows that have the same values in specified columns into summary rows.
- **Example:** `SELECT grade, COUNT(*) FROM students GROUP BY grade;`
- **Explanation:** This counts how many students are in each grade.

HAVING

- **Purpose:** Filters groups based on a specified condition, used in conjunction with `GROUP BY`.

- **Example:** `SELECT grade, COUNT(*) FROM students GROUP BY grade HAVING COUNT(*) > 1;`
- **Explanation:** This shows grades that have more than one student.

ORDER BY

- **Purpose:** Sorts the result set in ascending or descending order.
- **Example:** `SELECT name, age FROM students ORDER BY age DESC;`
- **Explanation:** This retrieves the names and ages of students, sorted by age in descending order.

LIMIT

- **Purpose:** Specifies the maximum number of records to return, useful for pagination.
- **Example:** `SELECT name FROM students LIMIT 2;`
- **Explanation:** This retrieves the names of the first two students in the result set.

Summary

Each SQL clause serves a distinct function, from specifying which columns to retrieve (`SELECT`), where to retrieve them from (`FROM`), how to filter (`WHERE`), combine (`JOIN`), group (`GROUP BY`), and order (`ORDER BY`) the data, to how many records to return (`LIMIT`). By combining these clauses, you can construct complex queries to extract or manipulate data from relational databases efficiently.

JOINS

If you want to join two tables but only include specific information from each, you don't need multiple `SELECT` statements; instead, you specify exactly which columns you want in the single `SELECT` clause of your query. You can include columns from both tables in the `SELECT` clause and use the `JOIN` operation to combine the tables based on a related column. Here's how you structure such a query:

Example Scenario

Assume you have two tables, `students` and `subjects`, and you want to join them to get each student's name along with the subject names they are enrolled in, but you do not want to include all columns from both tables.

`students` Table:

id	name	age
1	Alice	14
2	Bob	15
3	Charlie	14

`subjects` Table:

id	student_id	subject
1	1	Math
2	1	Science
3	2	Math
4	3	Literature

```
SELECT students.name, subjects.subject
FROM students
JOIN subjects ON students.id = subjects.student_id
;
```

Explanation

- `SELECT students.name, subjects.subject`: This specifies that you only want to retrieve the `name` column from the `students` table and the `subject` column from the `subjects` table.
- `FROM students`: This indicates that the base table for the query is `students`.

- `JOIN subjects ON students.id = subjects.student_id`: This performs an inner join between `students` and `subjects`, using the `id` column from `students` and the `student_id` column from `subjects` to find matches. Only rows with matching values in these columns will be included in the result.

Result

A result set that includes each student's name along with each subject they are enrolled in. If a student is enrolled in multiple subjects, there will be multiple rows for that student, one for each subject.

This approach allows you to selectively include information from both tables in your query result without needing multiple `SELECT` statements. You can further refine which rows are included in the result using conditions in a `WHERE` clause, and you can control the order of the results with an `ORDER BY` clause.

Given the example tables and the SQL query provided, the result of the query would look something like this:

Result Table

name	subject
Alice	Math
Alice	Science
Bob	Math
Charlie	Literature

LEFT/RIGHT JOINS

Remember, a `LEFT JOIN` includes all rows from the **left table** and matched rows from the right table, while a `RIGHT JOIN` includes all rows from the **right table** and matched rows from the left table.

LEFT JOIN

`LEFT JOIN` includes all rows from the **left table** and matched rows from the right table

```
SELECT students.name, subjects.subject
FROM students
LEFT JOIN subjects ON students.id = subjects.student_id
;
```

This query says:

- Start with all rows from the `students` table.
- Look for matching rows in the `subjects` table where `students.id` equals `subjects.student_id`.
- For students who have matching subjects, include those subjects in the results.
- For students who do not have matching subjects, include the student names in the result but show `NULL` for the `subject`.

Result:

name	subject
Alice	Math
Alice	Science
Bob	Math
Charlie	Literature
(No matching student for History)	