DISTINCT IN SQL

"DISTINCT is a SQL keyword that can be used in a SELECT statement to return only unique rows based on the column expressions listed."

- DISTINCT applies to the result set as a whole, ensuring no duplicate rows.

Examples:

- If you select one column with DISTINCT → you get unique values of that column.
- If you select multiple columns with DISTINCT → you get unique combinations of those columns together.

DISTINCT VS GROUP BY

The below two queries return same results , unique services

```sql
SELECT service FROM patients GROUP BY service;
SELECT DISTINCT service FROM patients;
```

Which is better?

- Use DISTINCT when you only need unique values.
- Use GROUP BY when you also want to perform aggregations (e.g., COUNT, SUM, AVG).

Rule of thumb:

- DISTINCT → simpler, cleaner for uniqueness.
- GROUP BY → more powerful when combined with aggregate functions.

DISTINCT AND NULL VALUES

- DISTINCT treats NULL values as equal.
- This means if multiple rows contain NULL in the selected column(s), only one NULL will appear in the result set.

Example:

```
SELECT DISTINCT col1 FROM table;
```

- If col1 has values: A, A, NULL, NULL, B → Result: A, NULL, B
- Key takeaway: DISTINCT collapses duplicate NULLs into a single entry, just like any other duplicate value.

WHEN NOT TO USE DISTINCT

Example: Taking the Day 11 Challenge to understand when not to use DISTINCT to fetch unique values
“Find all unique combinations of service and event type from the services_weekly table where events are not null or none, along with the count of occurrences for each combination. Order by count descending.”

Thumb Rule:

DISTINCT = uniqueness only

GROUP BY = uniqueness + aggregation (“group similar rows together and then allow me to do math on each group”)

Since the above query needs uniqueness + counting,

GROUP BY is the correct and better choice.

WHY NOT?

Since COUNT(*) is derived from grouping, DISTINCT doesn’t change anything — each row is already unique by definition of GROUP BY.

Query:

```
SELECT service, event, COUNT(*) AS occurrences
FROM services_weekly WHERE event IS NOT NULL AND event <> 'none'
GROUP BY service, event ORDER BY occurrences DESC;
```