Here's a **PostgreSQL SQL evaluation order chart** — a handy reference to help you understand the order in which SQL clauses are evaluated and how they are used.

🧠 PostgreSQL Clause Evaluation Order (Mnemonic: *"Funky Wild Goats Have Sharp Dancing Orange Legs"* 🐐🕺)

Order	Clause	What It Does	Mnemonic
1️⃣	`FROM`	Identify tables, joins, and subqueries	**Funky**
2️⃣	`WHERE`	Row-level filtering (before grouping)	**Wild**
3️⃣	`GROUP BY`	Groups rows for aggregate functions	**Goats**
4️⃣	`HAVING`	Filters on aggregates (after grouping)	**Have**
5️⃣	`SELECT`	Picks and returns columns/expressions	**Sharp**
6️⃣	`DISTINCT`	Removes duplicate rows	**Dancing**
7️⃣	`ORDER BY`	Sorts results	**Orange**
8️⃣	`LIMIT/OFFSET`	Limits the number of rows	**Legs**

Mnemonic: "**F**rogs **J**ump **W**hen **G**ood **H**armony **S**ounds **O**ut **L**oudly"**

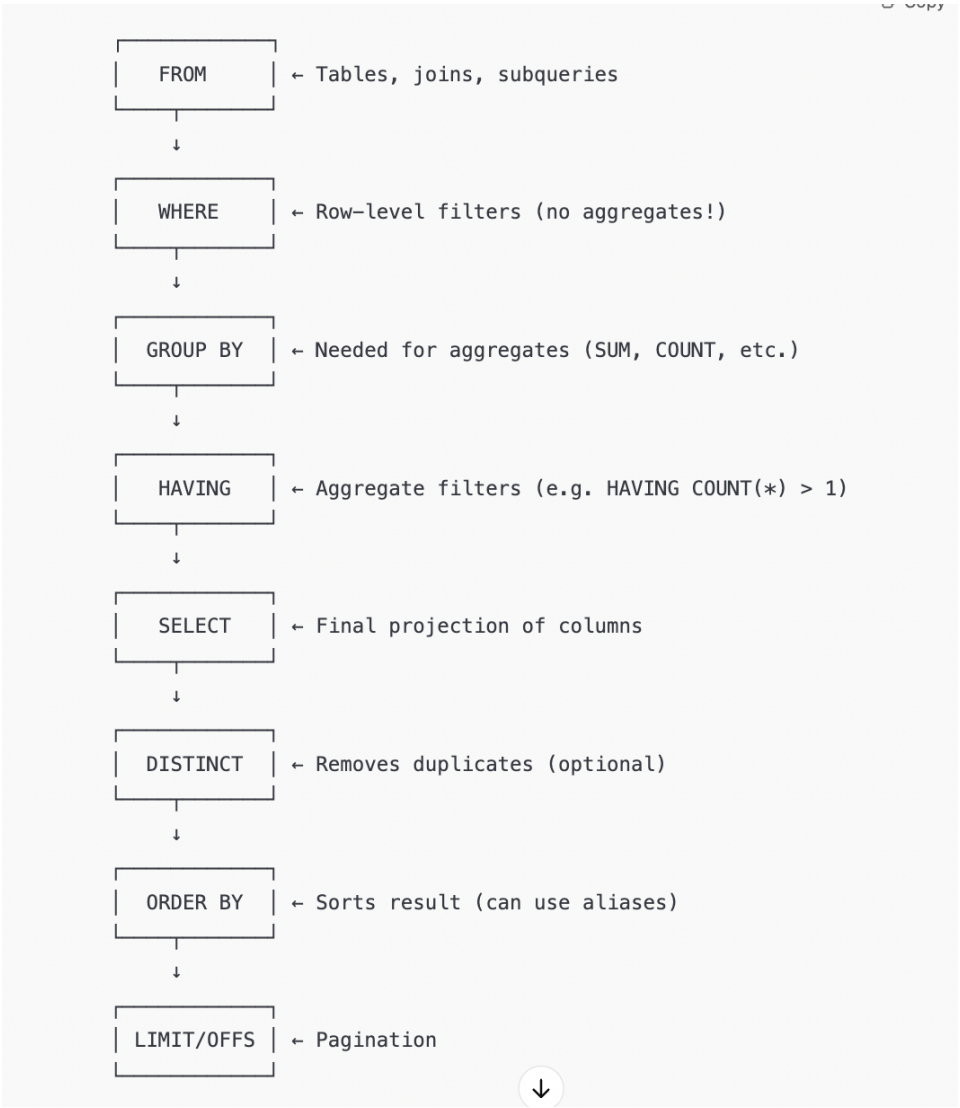
Clause Properties at a Glance

Clause	Filters...	Aggregates Allowed?	Aliases Allowed?	Mnemonic
FROM	Tables to scan	❌ No	❌ No	*"First, find the data"*
JOIN	Relationship logic	❌ No	❌ No	*"Connect related data"*
WHERE	**Individual rows**	❌ No	❌ No	*"Filter raw rows early"*
GROUP BY	Group definition	❌ No	❌ No	*"Bundle rows together"*
HAVING	**Entire groups**	✅ Yes	❌ No	*"Filter after bundling"*
SELECT	Columns to return	✅ Yes	✅ Yes	*"Choose what to show"*
ORDER BY	Sort order	✅ Yes	✅ Yes	*"Arrange the results"*
LIMIT	Row count	✅ Yes	✅ Yes	*"Trim to final output"*

🔁 Clause Reference Table

Clause	When It Runs	Used For
`FROM`	First	Pulls data from tables, joins, and subqueries
`WHERE`	After FROM	Filters rows BEFORE aggregation
`GROUP BY`	After WHERE	Groups rows for aggregation (`SUM` , `COUNT` , etc.)
`HAVING`	After GROUP BY	Filters groups (e.g. `HAVING COUNT(*) > 2`)
`SELECT`	After HAVING	Chooses what to return
`DISTINCT`	After SELECT	Removes duplicates from selected rows
`ORDER BY`	After DISTINCT	Sorts the final output
`LIMIT/OFFSET`	Last	Restricts number of output rows (pagination)

```sql
SELECT column1, AVG(column2)  -- 6. SELECT (choose output)
FROM table1                   -- 1. FROM (source)
JOIN table2 ON...             -- 2. JOIN (combine)
WHERE condition               -- 3. WHERE (row filter)
GROUP BY column1              -- 4. GROUP BY
HAVING AVG(column2) > 50      -- 5. HAVING (group filter)
ORDER BY column1 DESC         -- 7. ORDER BY
LIMIT 10;                     -- 8. LIMIT
```


FROM ← Tables, joins, subqueries
WHERE ← Row-level filters (no aggregates!)
GROUP BY ← Needed for aggregates (SUM, COUNT, etc.)
HAVING ← Aggregate filters (e.g. HAVING COUNT(*) > 1)
SELECT ← Final projection of columns
DISTINCT ← Removes duplicates (optional)
ORDER BY ← Sorts result (can use aliases)
LIMIT/OFFS ← Pagination