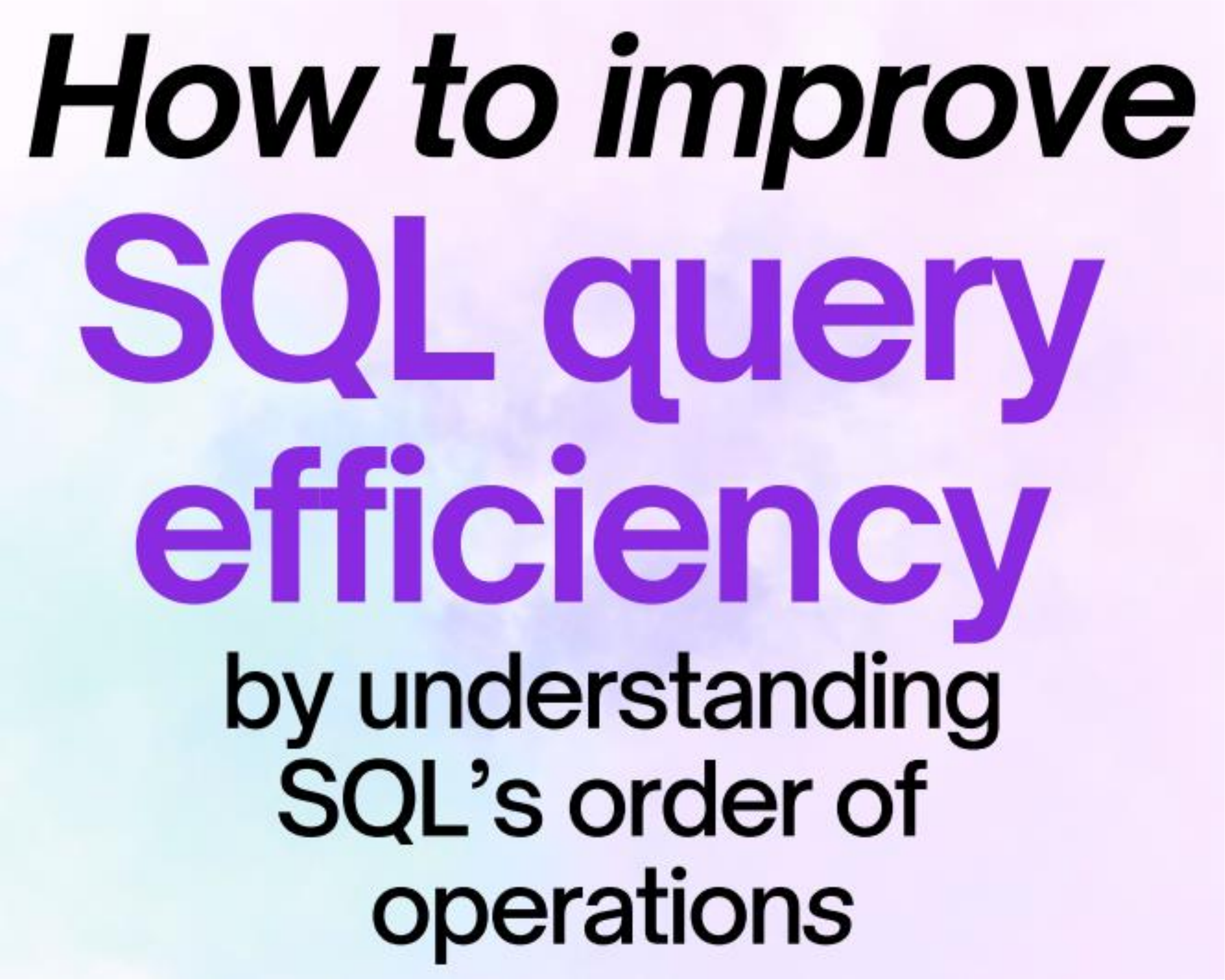
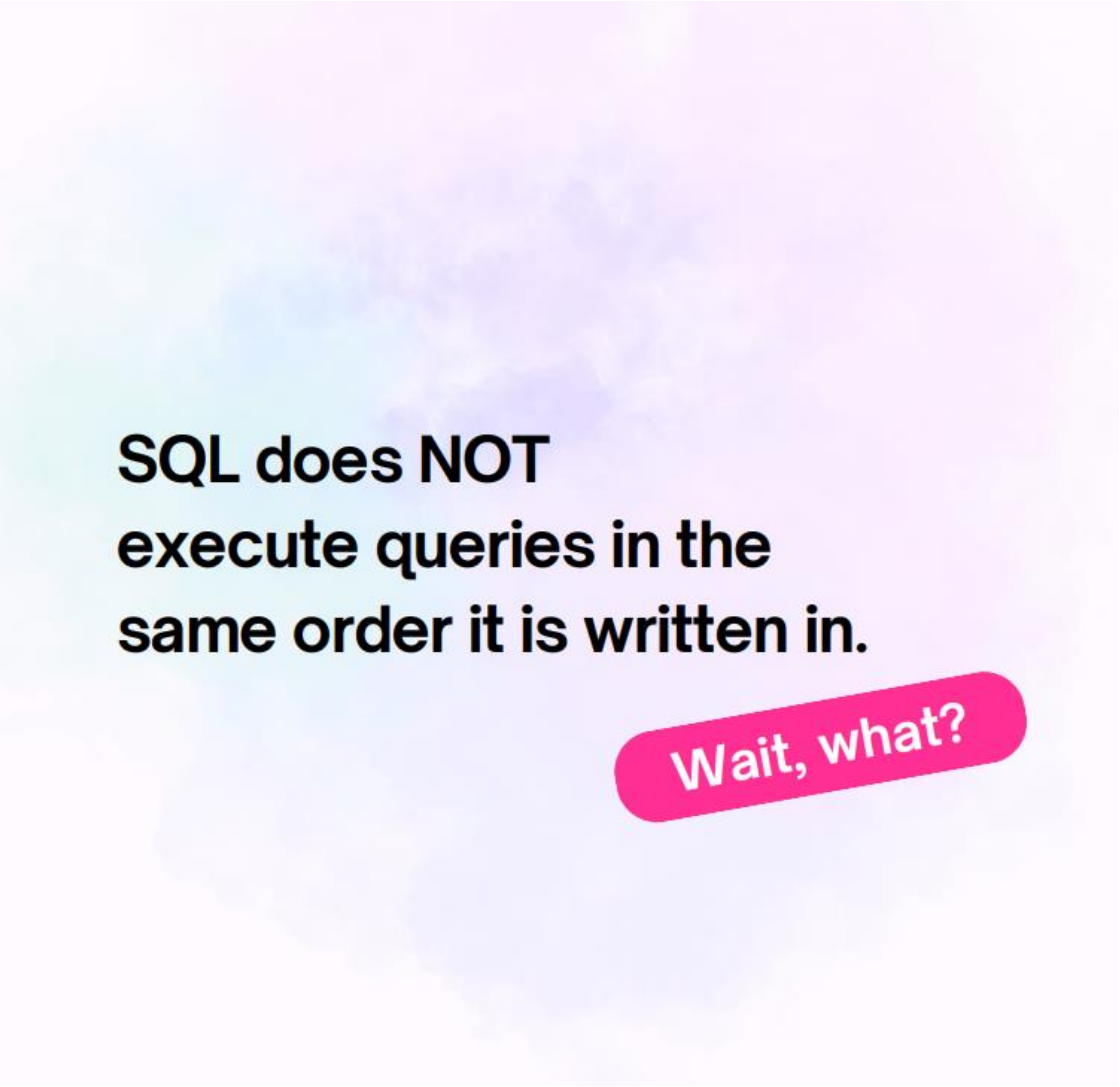
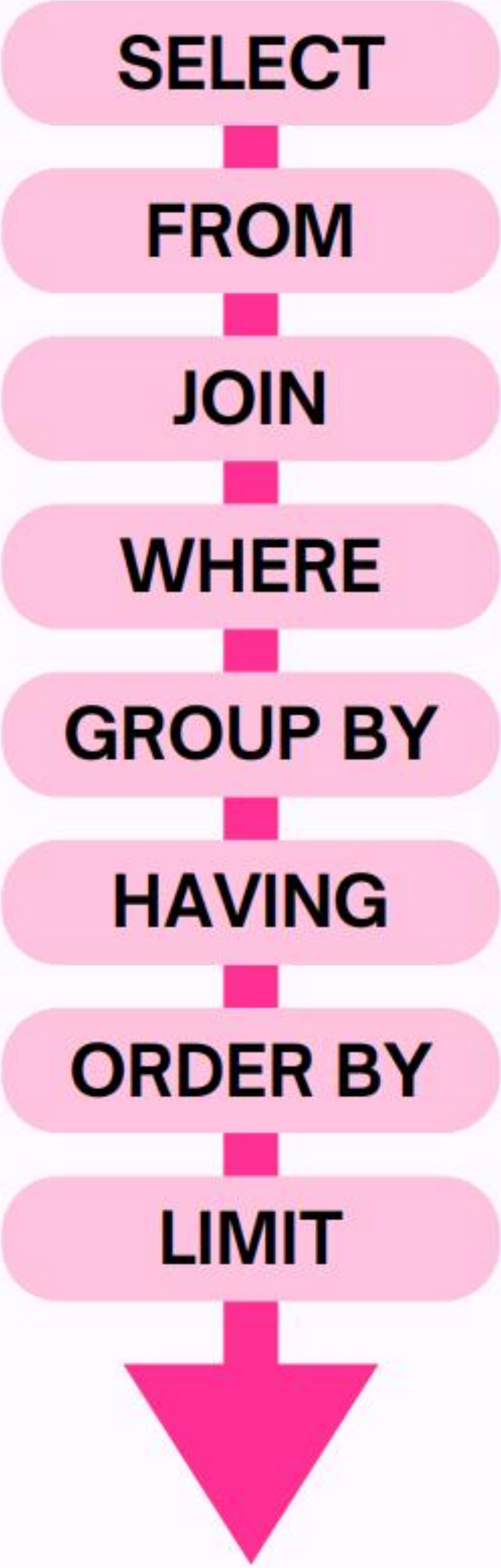


# How to improve SQL query efficiency by understanding SQL's order of operations



SQL does NOT execute queries in the same order it is written in.

Wait, what?



SELECT

FROM

JOIN

WHERE

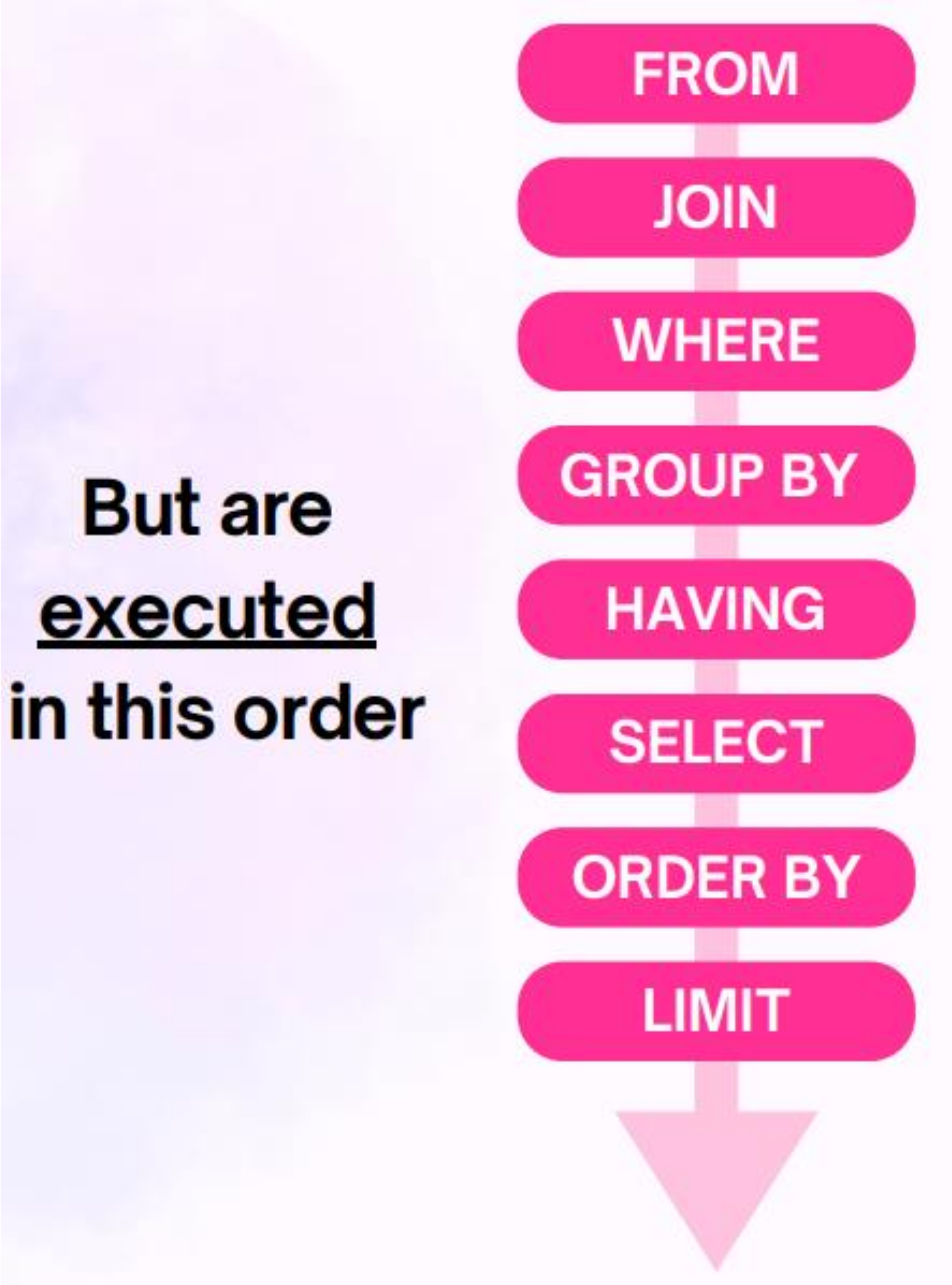
GROUP BY

HAVING

ORDER BY

LIMIT

## SQL commands are written in this order



But are **executed** in this order

FROM

JOIN

WHERE

GROUP BY

HAVING

SELECT

ORDER BY

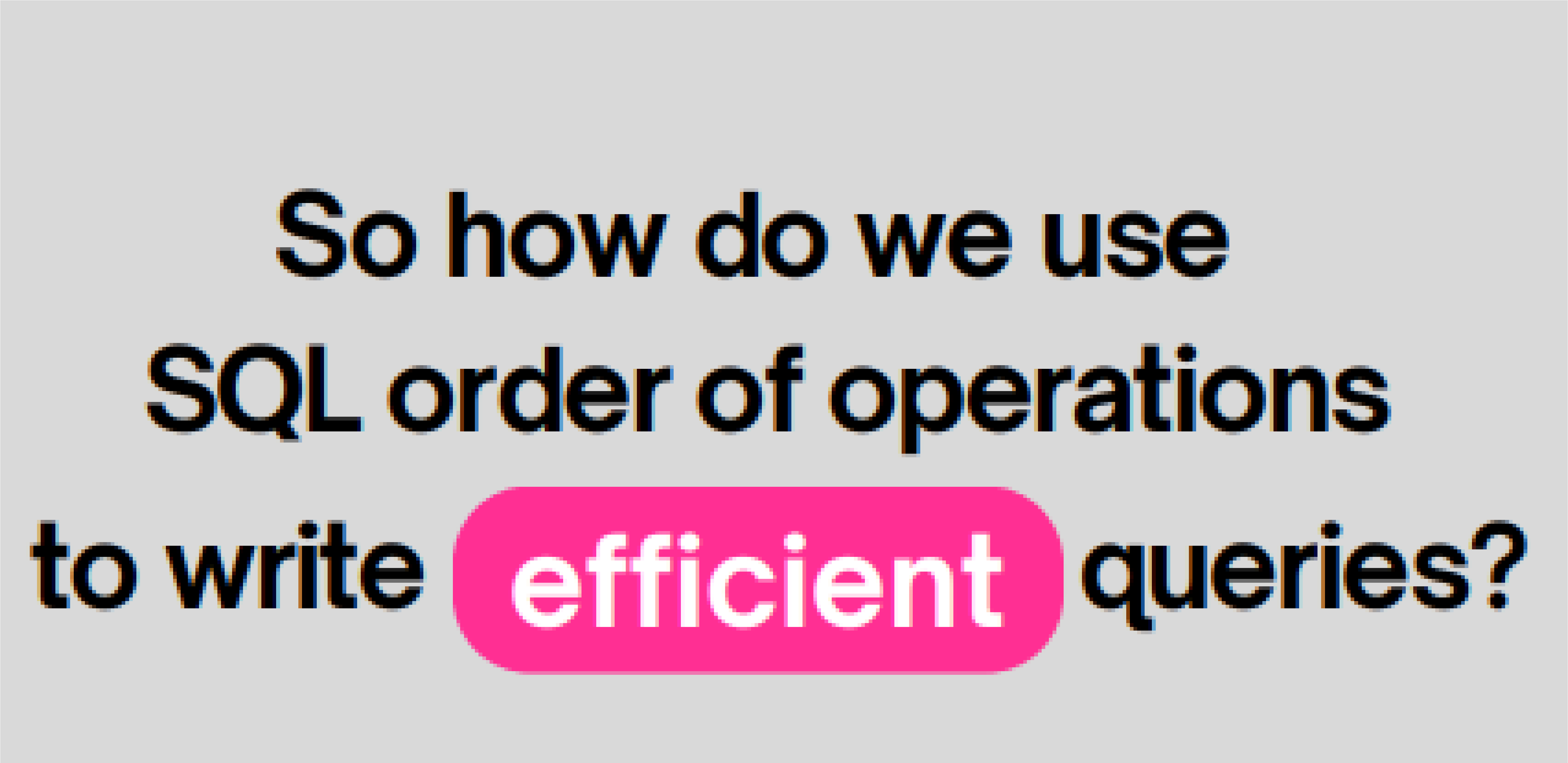
LIMIT

# We are providing Training in Below Courses

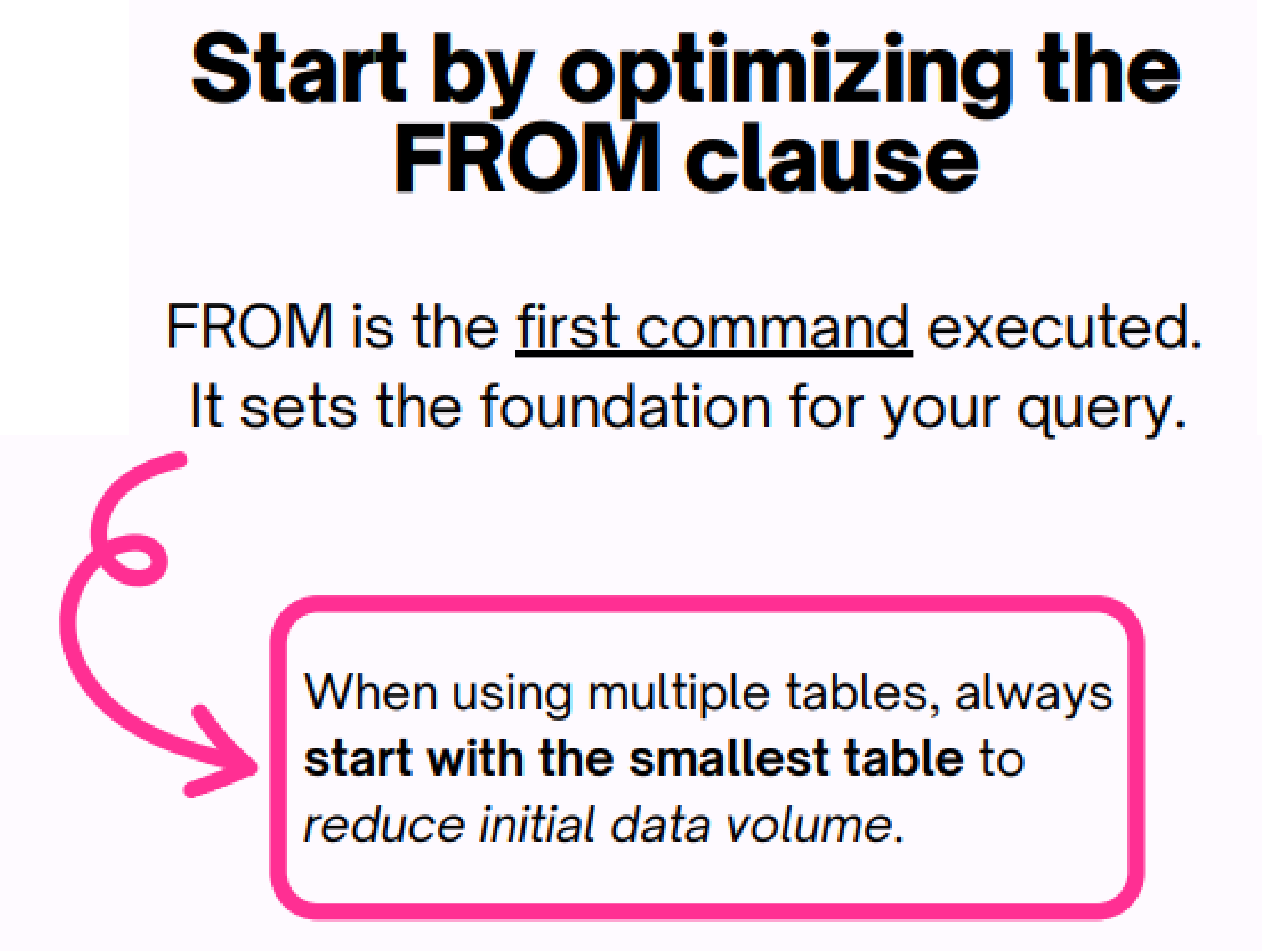
## Projects Training -:- Internship

| | | |
|---|---|---|
| C/C++ | Python / Django | Data Analysis |
| DSA | React Js / Angular Js | Power BI |
| DBMS | MERN / MEAN | AI / ML |
| Mongo DB | Full Stack Development | Data Science |
| Java/Ad. Java | | Generative AI |

***Certified Courses.***
***100% job assistance*.***
***Courses with Hands-On Projects.***



So how do we use SQL order of operations to write efficient queries?



# Start by optimizing the FROM clause

FROM is the first command executed.
It sets the foundation for your query.

When using multiple tables, always **start with the smallest table** to *reduce initial data volume.*

The real game-changer:

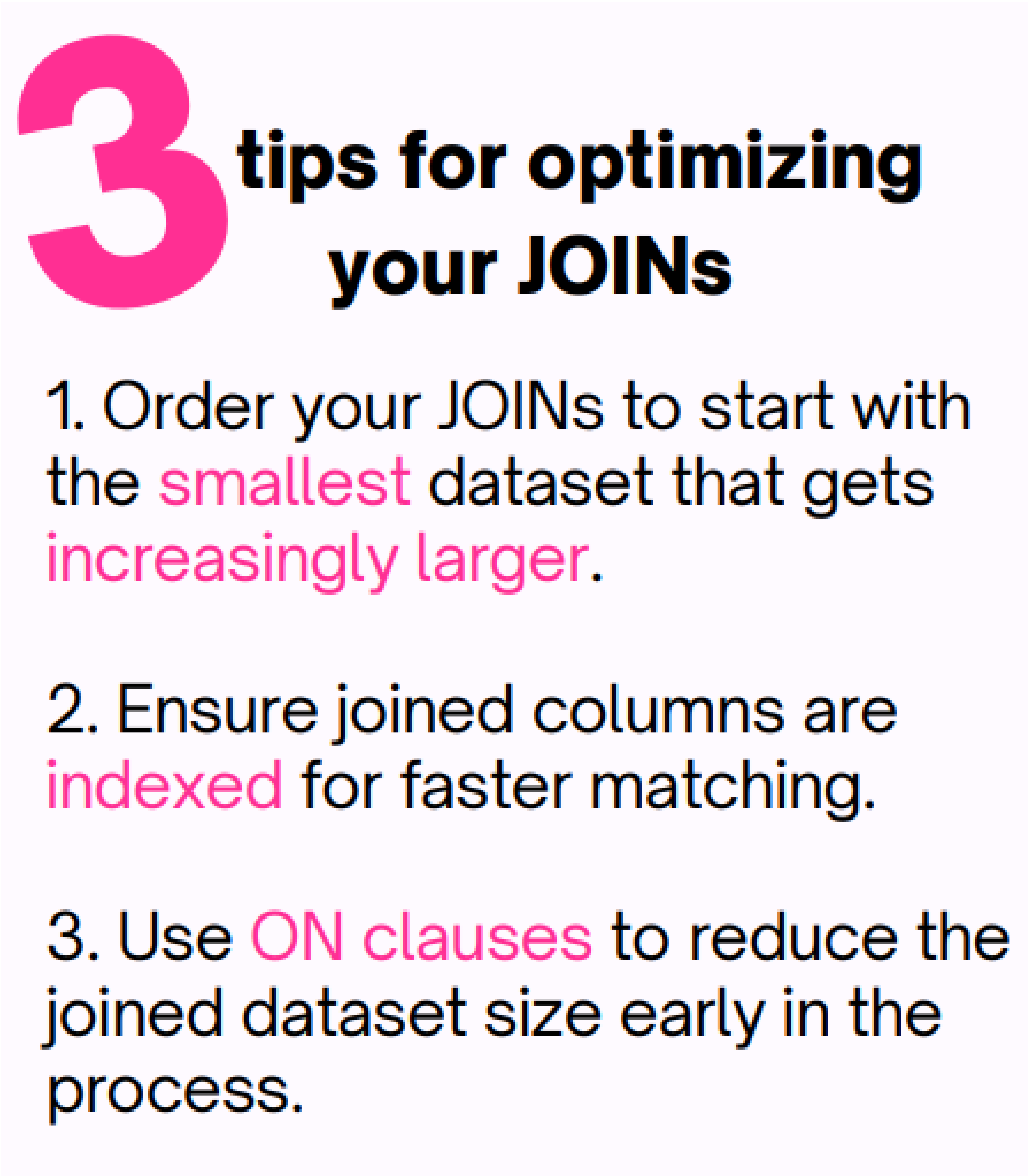
**is to optimize your JOINs using the SQL order of operations**

How though?

JOINs are one of the first to be executed

& they are computationally expensive

So, optimizing your JOINs are crucial for efficiency.

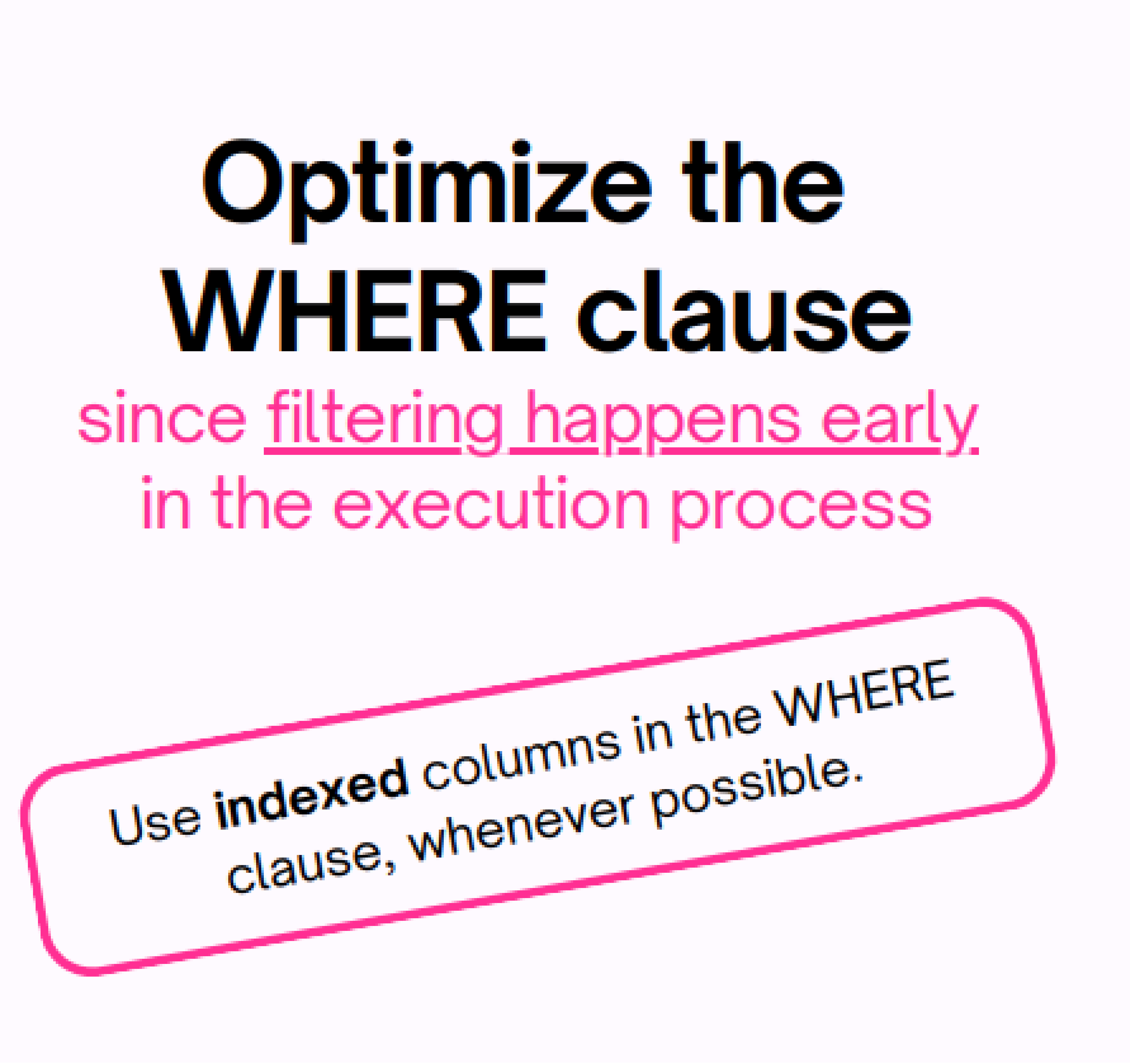


## 3 tips for optimizing your JOINs

1. Order your JOINs to start with the smallest dataset that gets increasingly larger.

2. Ensure joined columns are indexed for faster matching.

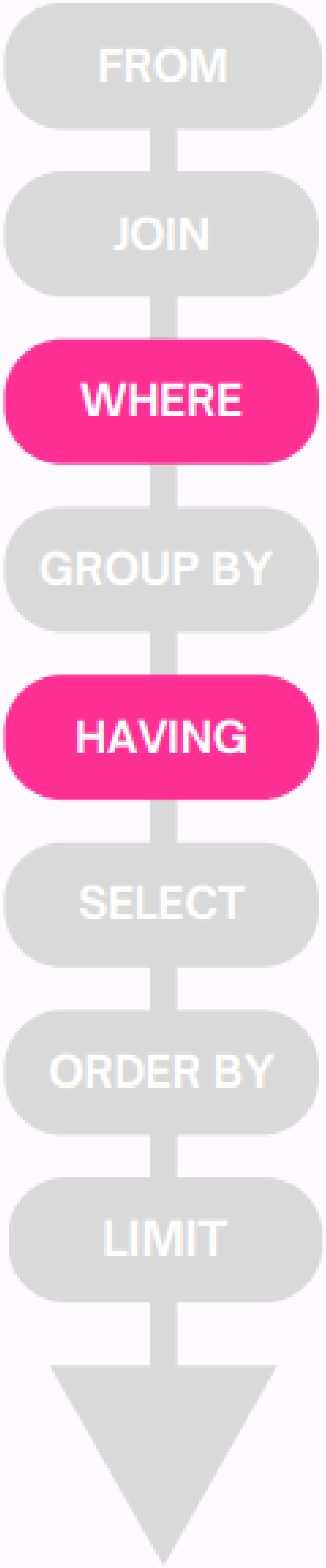
3. Use ON clauses to reduce the joined dataset size early in the process.



# Optimize the WHERE clause

since filtering happens early in the execution process

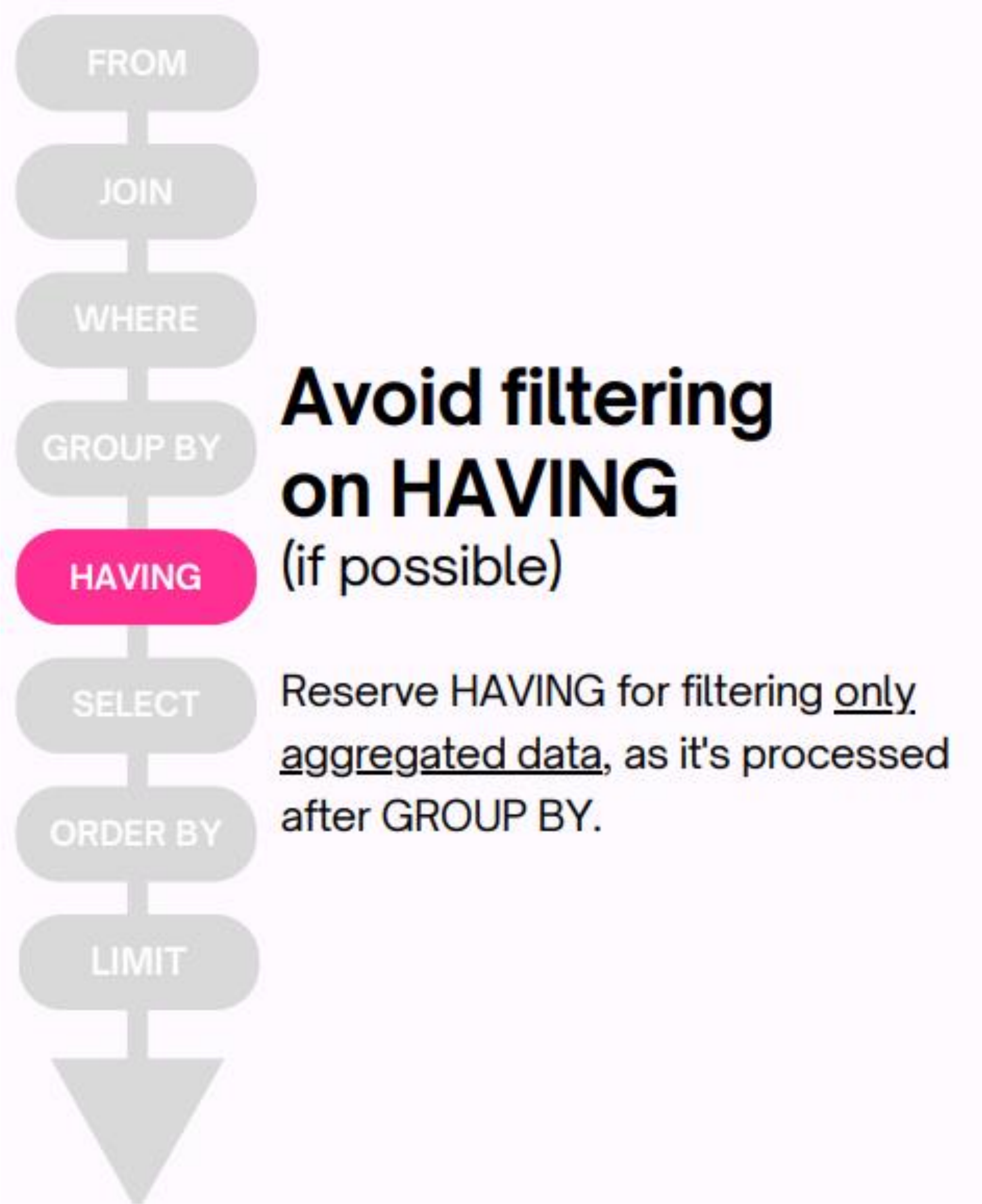
Use **indexed** columns in the WHERE clause, whenever possible.



# Use WHERE instead of HAVING

(if possible)

WHERE gets executed before HAVING, so put filters in the WHERE clause for better efficiency.



FROM

JOIN

WHERE

GROUP BY

HAVING

SELECT

ORDER BY

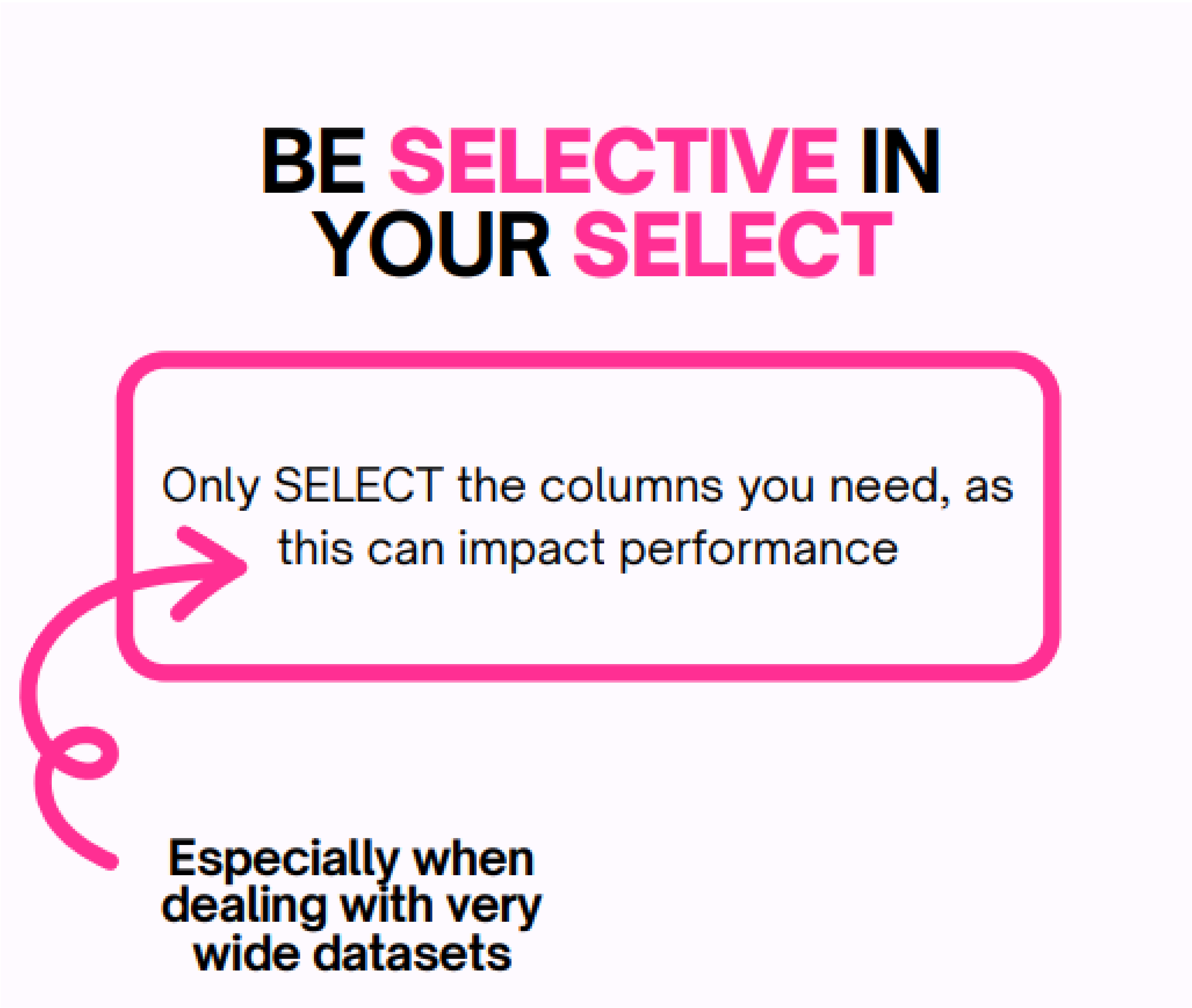
LIMIT

# Avoid filtering on HAVING

(if possible)

Reserve HAVING for filtering only aggregated data, as it's processed after GROUP BY.

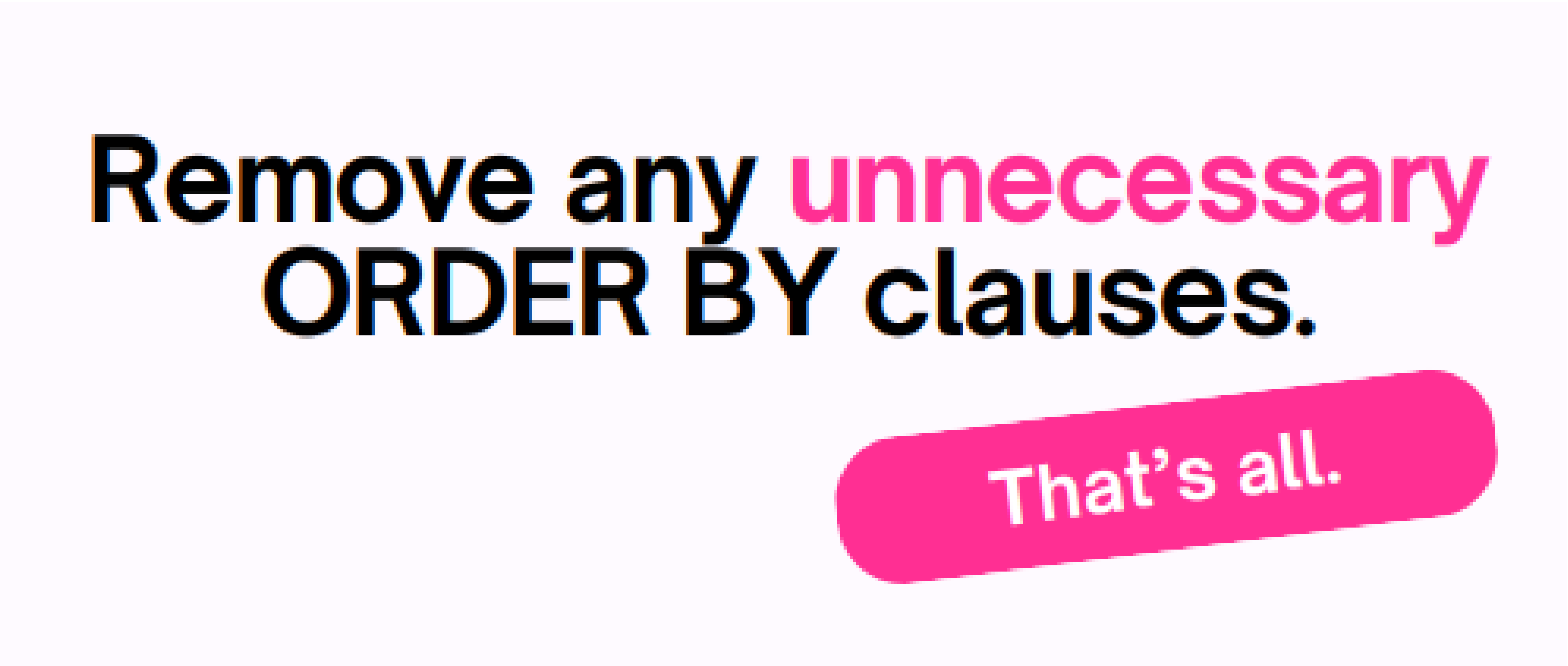
Finally, make sure to reduce redundancies



# BE SELECTIVE IN YOUR SELECT

Only SELECT the columns you need, as this can impact performance

Especially when dealing with very wide datasets



Remove any unnecessary ORDER BY clauses.

That’s all.

# We are providing Training in Below Courses

## Projects Training -:- Internship

| | | |
|---|---|---|
| C/C++ | Python / Django | Data Analysis |
| DSA | React Js / Angular Js | Power BI |
| DBMS | MERN / MEAN | AI / ML |
| Mongo DB | Full Stack Development | Data Science |
| Java/Ad. Java | | Generative AI |

***Certified Courses.***
***100% job assistance*.***
***Courses with Hands-On Projects.***