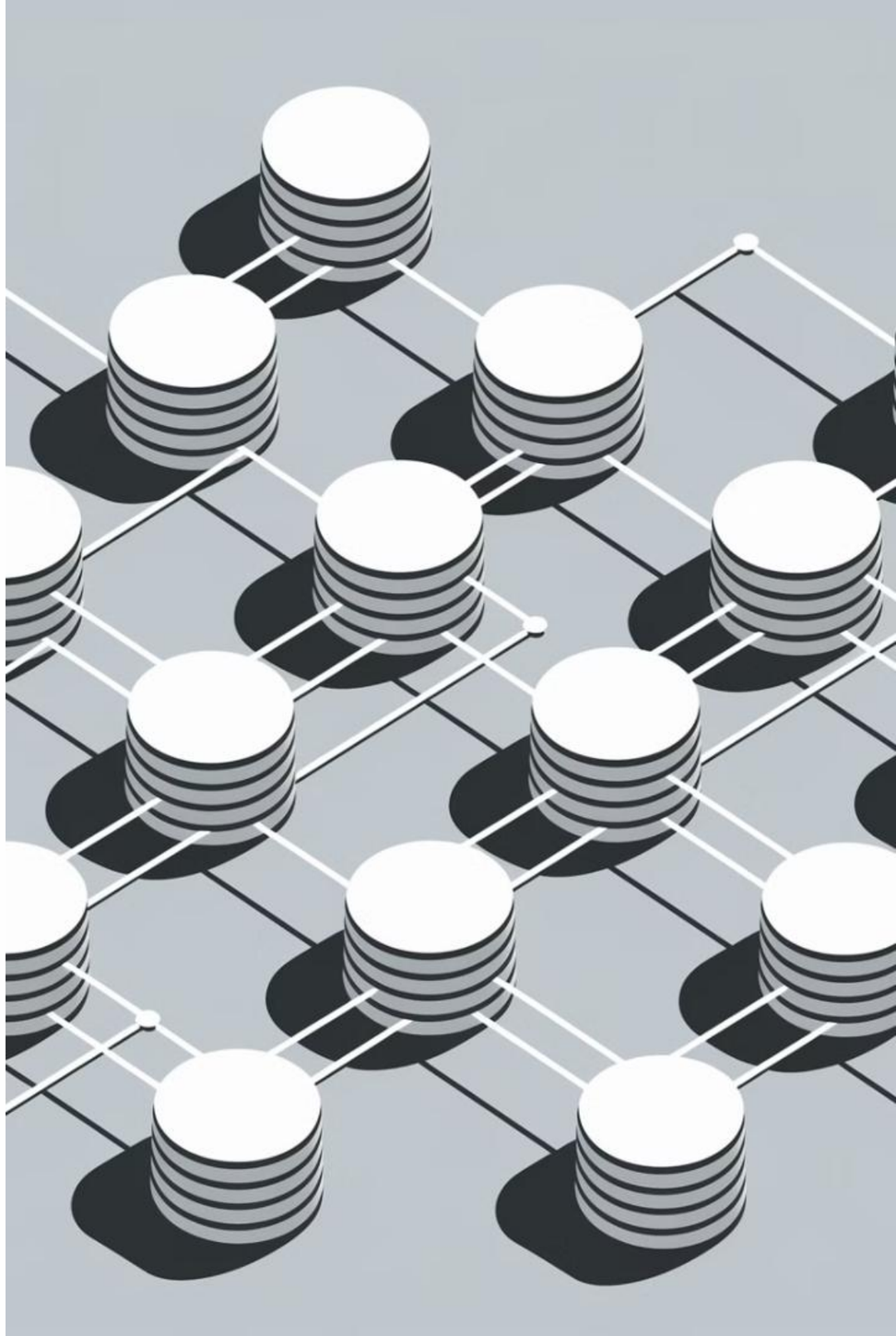
Understanding SQL Queries and Statements

A journey into the heart of SQL

Welcome to our exploration of SQL queries and statements. In this presentation, we'll dive deep into the world of database interactions, unraveling the power and flexibility of SQL. From basic commands to advanced techniques, we'll guide you through the essentials of crafting effective SQL queries.

por Mayko Silva



Welcome to the World of SQL!

Dive Into Queries

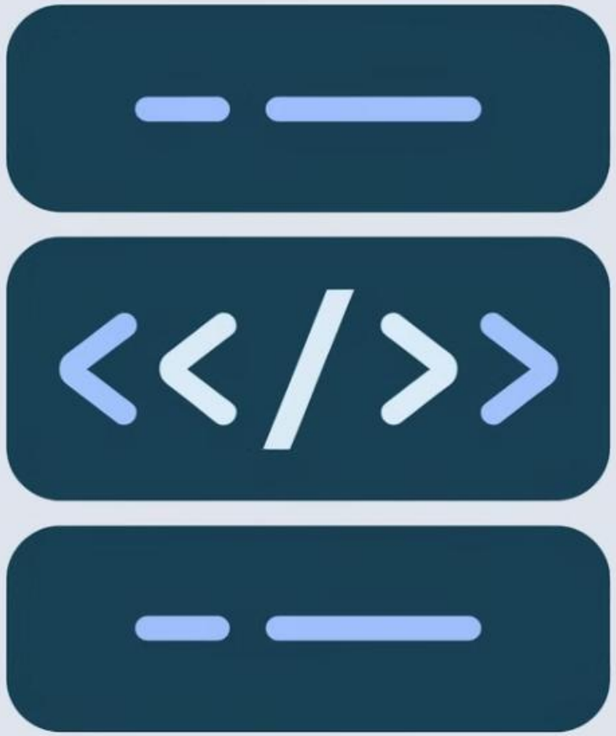
"Today, we will explore the heart of SQL: queries! Grab your notepad and get ready to learn!"

Powerful Tool

SQL queries are the key to unlocking valuable insights from your databases. They allow you to retrieve, filter, and manipulate data with precision and efficiency.

Essential Skill

Whether you're a data analyst, developer, or business professional, understanding SQL queries is crucial in today's data-driven world.



The Most Basic Yet Powerful Command

SELECT * FROM my_table;

This command means: "Hey, database, show me everything you have in my_table!" It's the simplest way to retrieve all data from a table.

The Power of *

The asterisk (*) is a wildcard that represents all columns. It's a quick way to see the entire structure and content of your table.

Caution

While powerful, use this command judiciously. For large tables, it can be resource-intensive and may return more data than you need.

SQL Writing Best Practices

Aa

Reserved Words

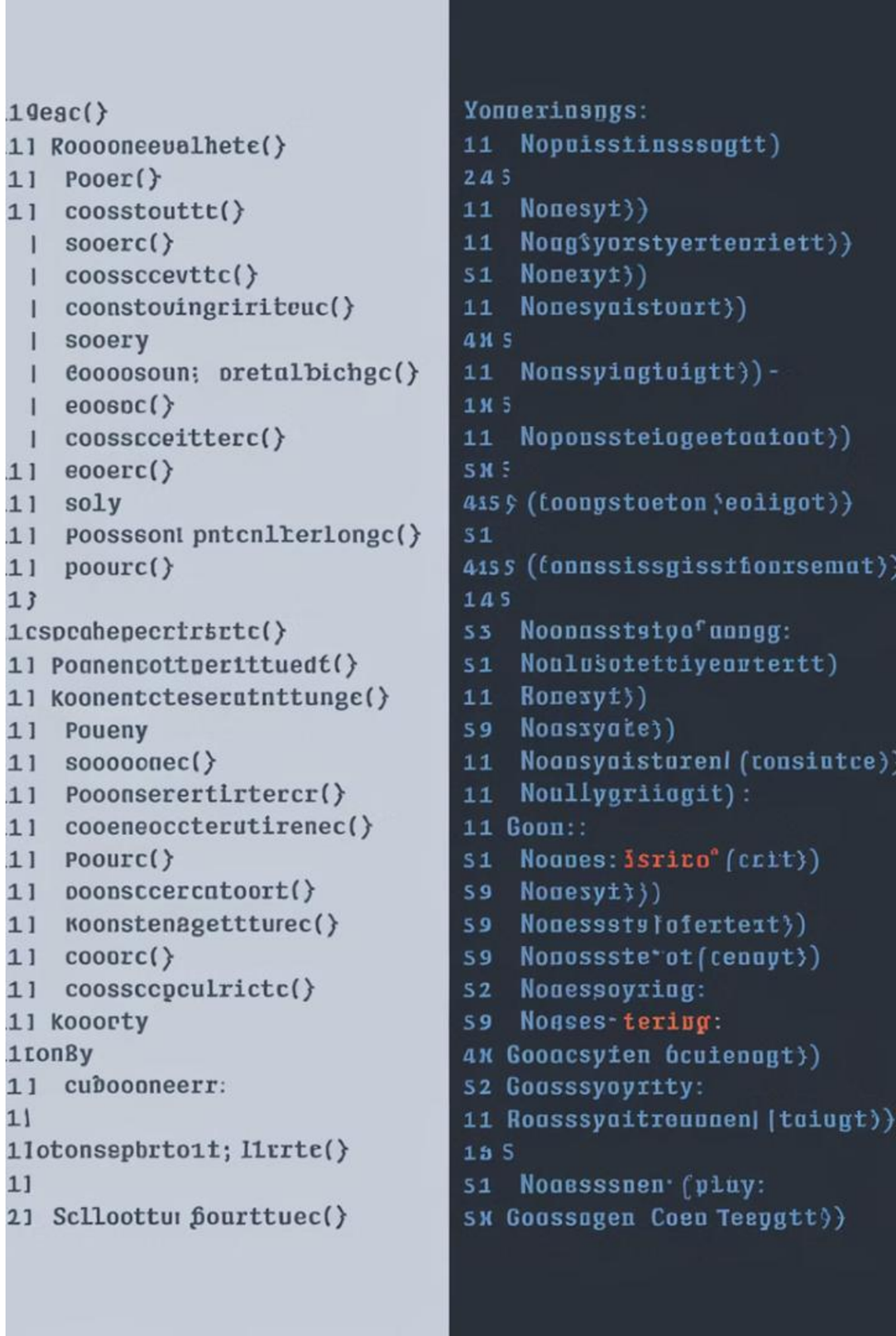
Words in uppercase (SELECT, FROM) are SQL reserved words. This makes your queries more readable and distinguishes SQL syntax from your table and column names.

Table and Column Names

Table and column names are usually written in lowercase for better readability. This convention helps differentiate between SQL keywords and your database objects.

Consistency is Key

Whatever style you choose, be consistent throughout your queries. Consistency makes your code easier to read and maintain, especially in larger projects.



Refining Queries

```
SELECT *
FROM my_table
WHERE column1 > 100
ORDER BY column2;
```

1. SELECT *

   Start with selecting all columns from your table.

2. FROM my_table

   Specify the table you want to query.

3. WHERE column1 > 100

   Filter the data based on specific conditions.

4. ORDER BY column2

   Sort the results according to your preferences.

We added **WHERE** to filter the data and **ORDER BY** to sort it. This allows for more precise and organized data retrieval.

The Correct Sequence When Writing SQL Queries

1. SELECT

   Choose the columns you want to retrieve.

   2. FROM

      Specify the table(s) you're querying.

      3. WHERE

         Filter the data based on conditions.

         4. GROUP BY

            Group rows that have the same values.

         5. HAVING

            Specify a search condition for a group.

      6. ORDER BY

         Sort the result set in ascending or descending order.

Mnemonics: "Sweaty Feet Will Give Horrible Odors" or "Start Fridays With Grandma's Homemade Oatmeal"

The Actual SQL Execution Order

1. FROM

Determines which table to retrieve data from.

2. WHERE

Filters the rows based on the specified condition.

3. GROUP BY

Groups rows that have the same values.

4. HAVING

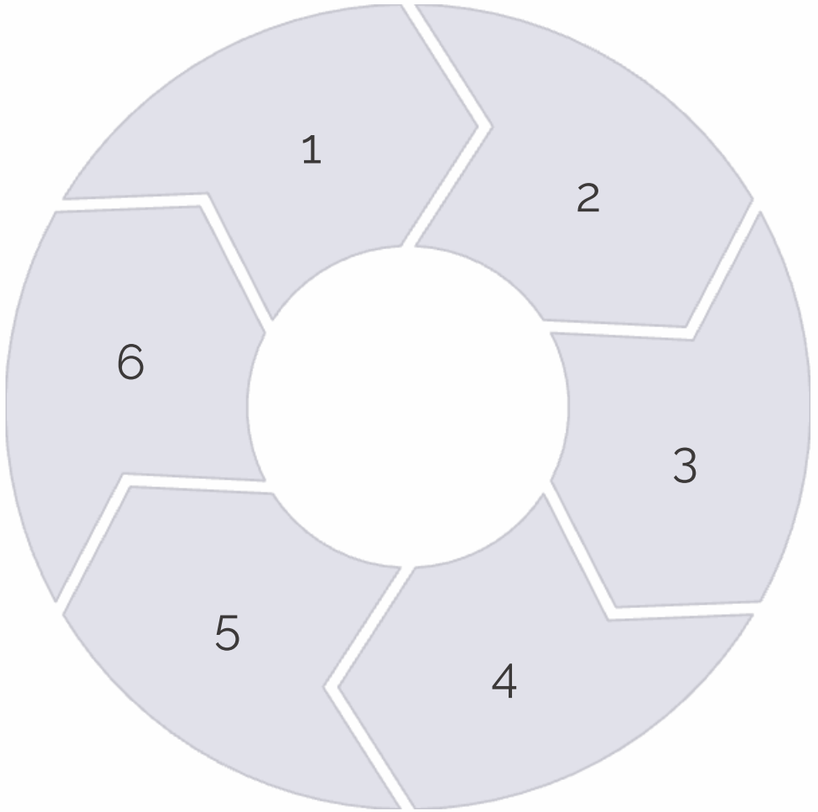
Filters the groups based on a specified condition.

5. SELECT

Selects the specified columns from the result.

6. ORDER BY

Sorts the final result set.



What Did We Learn Today?

1 Basic Structure of a SELECT Statement

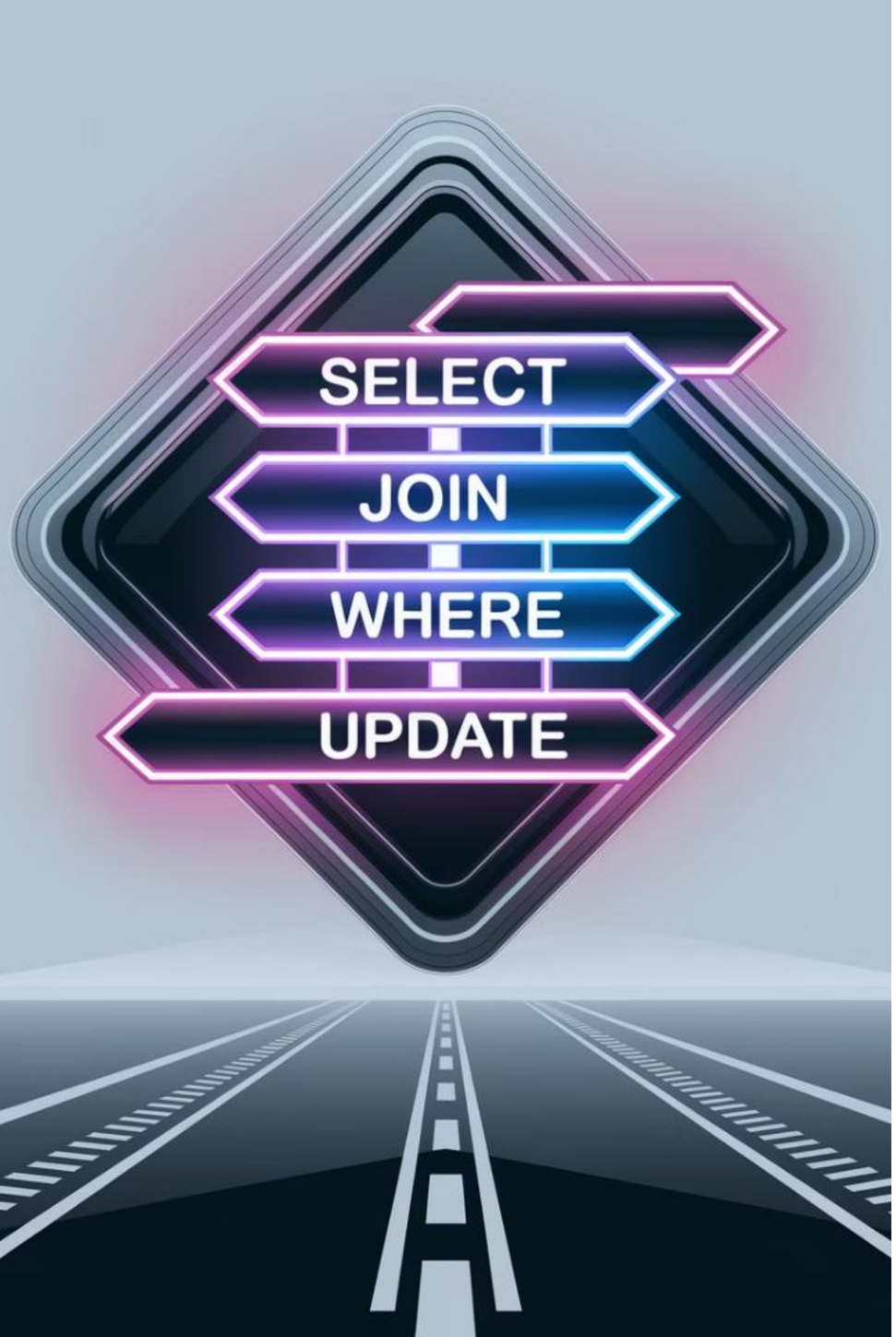
We explored the fundamental components of a SELECT statement, including FROM, WHERE, and ORDER BY clauses.

2 Importance of Clause Order

We learned the correct sequence for writing SQL queries and how it differs from the actual execution order.

3 Writing Order vs Execution Order

We discovered that the order in which we write SQL clauses is different from how the database engine executes them, which is crucial for understanding query optimization.



Next Steps in Your SQL Journey

1

Practice

Apply what you've learned by writing your own SQL queries. Start with simple SELECT statements and gradually add complexity.

2

Explore

Dive deeper into advanced SQL concepts like JOINs, subqueries, and aggregate functions.

3

Optimize

Learn about query optimization techniques to make your SQL queries more efficient and faster.

4

Apply

Use SQL in real-world scenarios. Try working with actual databases and solving practical problems.