WEEK 16

How to **Property** **Structure** a **Query**

PRESENTED BY **FILSAN MUSA & FADUMO DIRIYE**

TABLE OF CONTENT

Query Order

When writing a query in SQL we should know that the placements of different components are very important, however here is a general layout of where different components should be:

```
SELECT
FROM
JOIN
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

Note: Should be used in this exact order. Let’s look at a filled out example here: https://github.com/filsan95/Course-SQL_Fundamentals_for_Data_Analysis/edit/main/WEEK_16/notes_16.md/

Query Order (Cont.)

```sql
SELECT
    -- List columns of interest, and manipulate if necessary using arithmetic operators, CASE
    ,column_1
    ,column_2 * 2 AS doubled_column  -- Example of arithmetic manipulation
    ,CASE
        WHEN column_3 > 10 THEN 'High'
        ELSE 'Low'
    END AS column_3_status   -- Example of CASE WHEN
FROM
    -- List table, and database if necessary
    database_name.table_name
JOIN
    -- This is where you would list joining tables
    another_table ON table_name.id = another_table.id  -- Example of an INNER JOIN
WHERE
    -- Apply your conditions here
    column_1 = 'A' AND column_2 > 100  -- Example condition
GROUP BY
    -- Group by necessary columns (often for aggregation)
    column_1
HAVING
    -- Apply conditions to grouped results
    SUM(column_2) > 50  -- Example: Only include groups where sum of column_2 is greater than
ORDER BY
    -- Sort the results (can be ascending or descending)
    column_1 ASC, column_2 DESC  -- Sorting first by column_1 ascending, then column_2 descend
LIMIT
    -- Limit the number of rows if necessary
    10;  -- Limit the result to the first 10 rows
```

Query Format

When writing your query there is much left to personal preference, however there are some common best practices for writing easy to read/follow code. In all instances consistency is most important.

1. Uppercase or Lowercase - Generally Uppercase
2. Leading Commas, or Trailing Commas
3. Indents (*strongly preferred*)

```
SELECT column_1
      ,column_2
      ,column_3
      ,column_4
FROM table _a
WHERE column_1 > 10
AND (column_2 IN 'CANADA'
OR column_3 LIKE '%M%)
```

```
select column_1,
       column_2,
       column_3,
       column_4
from table _a
where column_1 > 10
and (column_2 in 'CANADA'
or column_3 like '%M%)
```

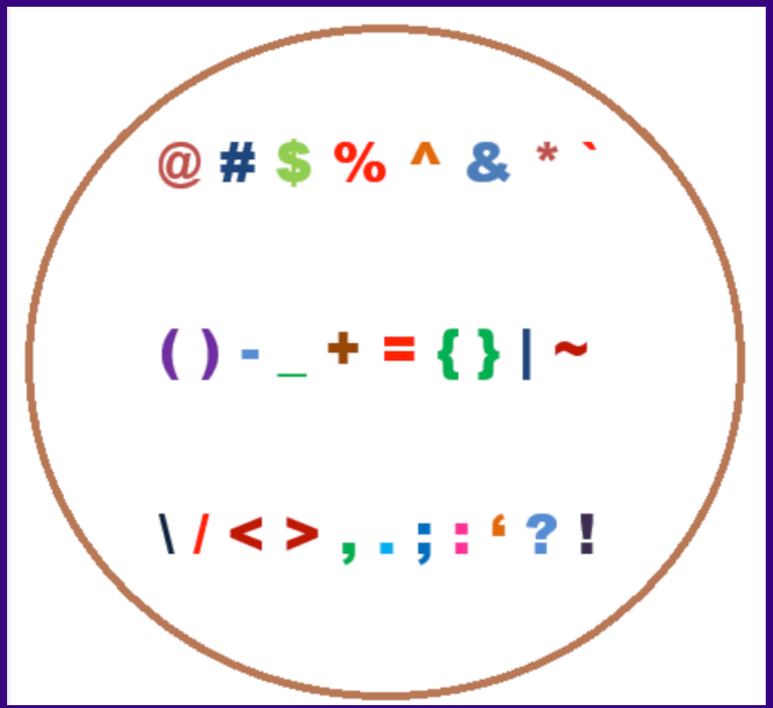
When writing code it's best to think of the UX, whether that's a manager, or colleague reading your code

Variable Naming Conventions

When considering naming conventions in SQL, it's important to adopt practices that promote readability, consistency, and maintainability of your code. While you have some flexibility in how you name your variables, there are certain rules to follow. Some practices should be avoided entirely to avoid potential issues.

1. **Avoid Using Special Characters:** Avoid using hyphens (`-`), spaces or other special character (ie. @, $, * etc.) in variable names, as these can cause syntax errors or other issues in SQL queries. SQL doesn't handle hyphens well because it interprets them as subtraction operators. As a general rule of thumb, refrain from ever using hyphens for naming variable since they can't be handled by most programming languages (ie. Python, C++, and Java to name a few).

Note: In other words, DO NOT use naming convention methods such as the Kebab Case (kebab-case) or Train Case (Train-Case).



Variable Naming Conventions (Cont.)

2. **Common Naming Conventions:**
   - **Snake Case (snake_case):** This convention uses underscores to separate words. It is often used in SQL to name columns, tables, and variables.
     Example: `order_total_amount`, `customer_id`.
   - **Camel Case (camelCase):** In this style, the first letter of the first word is lowercase, and the first letter of subsequent words is uppercase. This is often used in programming languages like JavaScript, but less common in SQL.
     Example: `orderTotalAmount`, `customerId`.
   - **Pascal Case (PascalCase):** Similar to CamelCase but with the first letter of the first word capitalized. Pascal case is more common in object-oriented programming but not widely used in SQL
     Example: `OrderTotalAmount`, `CustomerId`.

Naming Convention	Example Format
Pascal Case	PascalCase
Camel Case	camelCase
Snake Case	snake_case
Kebab Case	kebab-case
Flat Case	flatcase
Upper Flat Case	UPPERFLATCASE
Pascal Snake Case	Pascal_Snake_Case
Camel Snake Case	camel_Snake_Case
Screaming Snake Case	SCREAMING_SNAKE_CASE

Variable Naming Conventions (Cont.)

- **Best Practices:**
  - **Consistency:** Choose one naming convention and stick to it throughout your database schema and queries. Consistency will help keep your codebase organized and easy to read
  - **Clarity:** Use descriptive names for your variables that clearly indicate the nature of the information stored in said variable. Avoid abbreviations unless they are widely recognized (e.g., `uid` for unique identifier, or 'ip' for internet protocol address).
  - **Brief:** While being descriptive when naming your variable is desireable, avoid excessively long or overly descriptive variable names. Aim for brevity without sacrificing clarity.
  - **Avoid Reserved Keywords:** Never use SQL reserved keywords like SELECT, WHERE, or TABLE as variable names. This can lead to errors or ambiguity in your queries.

Common Errors & Debugging

Errors are often some of the most annoying and common issues to occur when writing code. Whether it's a simple syntax mistake or a more complex logic issue, errors are an inevitable part of programming. Knowing how to debug effectively is essential in resolving them and ensuring that the code runs smoothly.

Visible Errors:

1. **Syntax Errors:** These are the most basic type of errors. They occur when the code does not follow the correct syntax of the programming language. Common causes include missing parentheses, extra commas, or incorrect keyword usage. Syntax errors are often easy to spot and fix because they are usually pointed out by the compiler or interpreter.
2. **Logic Errors:** Logic errors occur when the code runs without crashing, but the output is not what you expect. These errors are trickier because they do not produce visible error messages and can be difficult to identify. They usually stem from incorrect algorithms, faulty conditions in loops, or wrong assumptions about the data being processed.

Common Errors & Debugging

Debugging Process:

To debug code, you first need to read and interpret the error message. Here's how to approach this:

- **Check the error message carefully:** If the message is not giving a specific error, it often includes the first affected line or a clue pointing to where things went wrong.
- **Understand the context:** Sometimes errors can arise from data issues or unexpected input. Make sure to check the inputs and the expected output before diving into the code.
- **Isolate the problem:** For SQL queries, the first error encountered will typically be the only one listed. If multiple issues exist, the first error message will often prevent the rest of the code from executing. It's a good idea to correct the first error, test the code, and then move on to subsequent errors.
- **Use debugging tools:** Tools like IDE debuggers, SQL EXPLAIN plans, or print statements can help you step through the code and see where things might be going wrong.
- **Test frequently:** Running your code in small increments and checking the output frequently helps catch errors early. It's much easier to fix smaller issues than to track down larger problems in a long piece of code.

Reference a Database

In the cases where you are encounter a large number of databases to pull data from, you may need to specify the database that contains the table you want to query. This is also helpful to avoid pulling from similarly named tables which may exist in another database.

```
SELECT column_1
      ,column_2
      ,column_3
FROM database_a.table_a
```

Note: “database_a.table_a” this syntax allows you to specify that you're querying the table_a table from database_a, ensuring the correct database context is used when executing the query.