SQL basics

SQL syntax structure

> The SQL syntax structure consists of **several key components** that include clauses, keywords, operators, identifiers, and comments that **make up the query**.

```
-- This comment explains the purpose of the query
SELECT
       last_name,
       salary
FROM
      employees
WHERE
      department_id = 5
      AND salary > 50000
ORDER BY
      last_name ASC;
```

Comments provide **explanatory notes** within the code which serve only as documentation.

SQL basics

SQL syntax structure

> The SQL syntax structure consists of **several key components** that include clauses, keywords, operators, identifiers, and comments that **make up the query**.

```
-- This comment explains the purpose of the query
SELECT
       last_name,
       salary
FROM
      employees
WHERE
      department_id = 5
      AND salary > 50000
ORDER BY
      last_name ASC;
```

SQL is built upon a set of **reserved keywords** that have specific meanings and functions. Examples of keywords include **SELECT, FROM, WHERE, ORDER BY, GROUP BY,** and **JOIN**.

SQL basics

SQL syntax structure

> The SQL syntax structure consists of **several key components** that include clauses, keywords, operators, identifiers, and comments that **make up the query**.

```
-- This comment explains the purpose of the query
SELECT
       last_name,
       salary
FROM
      employees
WHERE
      department_id = 5
       AND salary > 50000
ORDER BY
      last_name ASC;
```

Identifiers are used to name tables, columns, and other database objects.

SQL basics

SQL syntax structure

> The SQL syntax structure consists of **several key components** that include clauses, keywords, operators, identifiers, and comments that **make up the query**.

```
-- This comment explains the purpose of the query
SELECT
       last_name,
       salary
FROM
      employees
WHERE
      department_id = 5
      AND salary > 50000
ORDER BY
      last_name ASC;
```

Identifiers are used to name tables, columns, and other database objects.

SQL supports various **operators** for performing operations on data, including **arithmetic** (+, -, *, /), **comparison** (=, <, >, <=, >=, <>), **logical** (AND, OR, NOT), and **string concatenation** operators (+ or ||).

SQL basics

Naming conventions

> SQL is generally **not case-sensitive**. This means that whether you write keywords in uppercase or lowercase, the SQL engine will interpret them the same way.

However, to make them **easier to read and distinguishable from the identifiers**, it is recommended that keywords and clauses be written in **uppercase letters** and **begin on separate lines**.

Following this convention helps to maintain **consistency** in SQL code and makes it easier for **other developers to understand** and **work with** the code.

However, this **does not affect** the way the **SQL engine** interprets the statement.

```
SELECT
       last_name,
       salary
FROM
      employees
WHERE
      department_id = 5
      AND salary > 50000
ORDER BY
      last_name ASC;
```

SQL basics

Naming conventions

We use **relevant, descriptive, and consistent identifiers** for tables, columns, and other database objects. In most cases, **spaces and other special characters can cause errors**, so we avoid them. The most common naming conventions include:

camelCase

```
SELECT
       firstName,
       lastName
FROM
      employees
```

The **first letter** of each word is **lowercase**, and the **first letter of each subsequent concatenated word** is capitalised.

PascalCase

```
SELECT
       FirstName,
       LastName
FROM
      Employees
```

Each word **starts with an uppercase letter**, and there are **no separators** between words.

Underscore

```
SELECT
       first_name,
       last_name
FROM
      employees
```

Each word is **separated by an underscore**, and all letters are lowercase.

SQL basics

Semicolons

> In SQL, semicolons (;) are used as **statement terminators**. They indicate the end of a SQL statement and separate multiple SQL statements within a script.

```sql
INSERT INTO employees (first_name, last_name, salary)
VALUES ('John', 'Doe', 50000);

UPDATE employees
SET salary = 55000
WHERE employee_id = 1;

SELECT *
FROM employees;
```

By placing a semicolon at the end of each statement, we are able to **execute multiple queries in a single run**.

SQL basics

Commenting

> Comments are used to explain the reasoning behind a statement, provide reminders or to-do lists for future changes, temporarily disable some functions for debugging, making notes, etc.

Single-line commenting

```
-- This is a single-line comment
SELECT *
FROM employees; -- Retrieve all records from the
employees table
```

Single-line comments **begin with a double-dash (--)**. They are used to provide explanatory notes or documentation for a **specific line or statement**. Anything **after the double-dash until the end of the line** is considered a comment and is ignored by the SQL engine.

Multi-line commenting

```
/*
This is a multi-line comment.
It can span multiple lines.
*/
SELECT *
FROM employees;
```

Multi-line comments **begin with /* and end with */**. They are used to provide comments that **span multiple lines** or to temporarily disable a block of code. Anything between /* and */ is considered a comment and is ignored by the SQL engine.

SQL basics

Line breaks

> Line breaks, also known as **line terminators**, are characters used to **systematically separate lines of code** in SQL.

Functionality

- It is recommended to **use line breaks consistently** and strategically to **improve the clarity and readability** of SQL code.
- By using line breaks, SQL code becomes more **visually appealing** and **easier to read** and comprehend, especially **long and complex queries.**
- **SQL ignores line breaks**, so they do not affect the functionality or performance of the queries.

Example

```
SELECT last_name, salary FROM employees WHERE salary = 50000;
```

```
SELECT
       last_name,
       salary
FROM employees
WHERE
      salary > 50000;
```

SQL basics

Line breaks

> In SQL, there are **different systems for using line breaks** depending on personal preference and coding style. Here are the most commonly used:

Compact system

```
SELECT first_name, last_name
FROM employees
WHERE (department_id = 5
AND salary > 50000);
```

Line breaks are **used sparingly**, only **at the end of each statement**. This is useful for **shorter queries** or in situations where **minimising vertical space** is preferred.

- **Keywords and the related column** names start on a new line.
- **Parentheses** are typically **placed on the same line** as the corresponding expression or condition.

Aligned system

```
SELECT
       first_name,
       last_name
FROM
      employees
WHERE (
      department_id = 5
      AND salary > 50000
);
```

- **Keywords start on a new line** and related column names are given in new indented lines below.
- **Booleans** start on a new line **indented to the right**.
- Opening parentheses **terminate the line** and closing parentheses **align with the last keyword** on a new line.