SQL Fundamentals

- In general, SQL has three parts:
  - **Data Definition Language (DDL)**
  - **Data Manipulation Language (DML)**
  - **Data Control Language (DCL)**
- The first part is used to create and manage the structure of the data, the second part is used to manage the data itself, and the third part controls access to the data.

SQL Lexical Structure

- The minimum SQL instruction that can be executed by the database engine is a statement.
- It can also be called a command or query.
- For example, each of the following is a statement:
  - `SELECT car_id, number_of_doors FROM car_portal_app.car;`
  - `DELETE FROM car_portal_app.a;`
  - `SELECT now();`
- SQL commands are terminated by a semicolon, `;`.

- The following elements form the lexical structure of SQL:
  - **Keywords**: Determine what exactly is required from the database to be done
  - **Identifiers**: Refer to the objects in the database, such as tables, their fields, and functions
  - **Constants (or literals)**: Parts of expressions whose values are specified directly in the code
  - **Operators**: Determine how the data is processed in expressions
  - **Special characters (such as parenthesis, brackets, and commas)**: Have meanings other than simply being an operator
  - **Whitespaces**: Separate words from each other
  - **Comments**: Used to describe a particular piece of code and are ignored by the database

- **Keywords** are words such as `SELECT` or `UPDATE`.
- They have special meanings in SQL.
- The full list of keywords can be found in the documentation at http://www.postgresql.org/docs/current/static/sql-keywords-appendix.html

- **Identifiers** are the names of database objects.
- Objects such as tables or views can be referred to by the name of the schema they belong to followed by the dot symbol, `.`, and the name of the object.
- This is called a **qualified object name**.
- If the name of the schema is included in the `search_path` setting, or if the object belongs to the current user's schema, then it isn't necessary to use the schema name when referring to the object.
- In that case, it is called an **unqualified object name**.

- SQL is not case-sensitive.
- Both keywords and identifiers can contain any letters (`a–z`), digits (`0–9`), underscores (_), or dollar signs (`$`).
- However, they cannot start with a digit or a dollar sign.
- That makes them similar to each other and, without knowing the language, sometimes it's difficult to say whether a word is a keyword or an identifier.
- Usually, keywords are typed in uppercase.

- In identifiers, it's still possible to use symbols other than those mentioned earlier, by double-quoting them.
- Note that double-quoted identifiers are case-sensitive.
- It's also possible to create objects with the same names as keywords, but this isn't recommended.

- **Constants** in SQL are also called **literals**.
- PostgreSQL supports three types of implicitly-typed constants: numbers, strings, and bit strings.
- To use the constant values of any other data type, implicit or explicit conversion should be performed.

- Numeric constants contain digits and, optionally, a decimal point and an exponent sign.
- These are examples of selecting valid numeric constants:

```
car_portal=> SELECT 1, 1.2, 0.3, .5, 1e15, 12.65e-6;
 ?column? | ?column? | ?column? | ?column? |     ?column?     |  ?column?
----------+----------+----------+----------+------------------+------------
        1 |      1.2 |      0.3 |      0.5 | 1000000000000000 | 0.00001265
(1 row)
```

- String constants should be quoted.
- There are two kinds of syntax for string constants in PostgreSQL: single quoted constants, such as in the SQL standard, and PostgreSQL-specific `dollar-quoted` constants.
- Putting the letter `e` before a string constant makes it possible to use C-style backslash-escaped characters, such as `\n` for a new-line or `\t` for tabulation.
- A single quote character (`'`) inside a literal should be doubled (`''`) or used with an escape string `\'`.
- Putting the letter `U` with an ampersand (`&`) before the string without any spaces in between allows you to specify Unicode characters by their code after a backslash.

```
car_portal=> SELECT 'a', 'aa''aa', E'aa\naa', $$aa'aa$$, U&'\041C\0418\0420';
 ?column? | ?column? | ?column? | ?column? | ?column?
----------+----------+----------+----------+----------
 a        | aa'aa    | aa      +| aa'aa    | МИР
          |          | aa       |          |
(1 row)
```

- `dollar-quoted string` constants always have the same value as it is written in the code.
- No escape sequences are recognized and any kind of quotes will become part of the string.
- Dollar-quoted strings can have their names specified between the `$` signs, which makes it possible to use one `dollar-quoted string` inside another

```
car_portal=> SELECT $str1$SELECT $$dollar-quoted string$$;$str1$;
             ?column?
---------------------------------
 SELECT $$dollar-quoted string$$;
(1 row)
```

- Bit strings are preceded by the letter `B` and can contain only the digits `0` or `1`.
- Alternatively, they can be preceded by the letter `X` and contain any digits, along with the letters `A`–`F`.
  - In that case, they are hexadecimal strings.
- Most of the time, bit strings are converted into a numeric data type.

```
car_portal=> SELECT B'01010101'::int, X'AB21'::int;
 int4 | int4
------+-------
   85 | 43809
(1 row)
```

- **Operators** are basic elements of data processing.
- They are used in SQL expressions.
- They take one or two arguments and return a value.
- Examples of operators include addition, `+`, and subtraction `-`.
- PostgreSQL supports a wide range of operators for all data types.
- In the statements, the operators look like a single character or a short sequence of characters from this list: `+ - * / < > = ~ ! @ # % ^ & | ` ?`.

- Special characters in SQL include the following:
  - Parentheses (**()**): These are used to control the precedence of operations or to group expressions. They can also be used to identify tuples or attributes in a composite type, or as part of a function name. Some SQL commands have parentheses in their syntax.
  - Brackets (**[]**): These are used to select elements from an array.
  - Colons (**:**): These are used to access parts of arrays.
  - Double colons (**::**): These are used for type-casing.
  - Commas (**,**): These are used to separate elements of a list.
  - Periods (**.**): These are used to separate schema, table, and column names from each other.
  - Semicolon (**;**): This is used to terminate a statement.
  - Asterisk (*****): This is used to refer to all the fields of a table or all the elements of a composite value.

- **Whitespaces** separate words from each other.
- In SQL, any number of spaces, new lines, or tabulations are considered a single whitespace.

- **Comments** can be used in any part of SQL code.
- The server ignores comments, treating them as whitespace.
- Comments are quoted in pairs of `/*` and `*/`.
- Also, a whole line of code can be commented by using double dash `--`.
- In this case, a comment starts from the double dash and ends at the end of the line.