The SQL language and PL/pgSQL – a comparison

- we can write functions in C, SQL, and PL/pgSQL.
- There are some pros and cons to each approach.
- You can think of an SQL function as a wrapper around a parameterized `SELECT` statement.
- SQL functions can be inlined into the calling subquery, leading to better performance.
- Also, since the SQL function execution plan is not cached as in PL/pgSQL, it is often better in performance than PL/pgSQL.
- Moreover, caching in PL/pgSQL can have some surprisingly bad side effects, such as the caching of sensitive `timestamp` values, as shown in the documentation at http://www.postgresql.org/docs/current/interactive/plpgsql-implementation.html
- Finally, with the introduction of CTE, recursive CTE, window functions, and `LATERAL JOINS`, you can perform complex logic using only SQL.
- If function logic can be implemented in SQL, use an SQL function, instead of PL/pgSQL.

- The PL/pgSQL function execution plan is cached; caching the plan can help to reduce execution time, but it can also hurt it if the plan is not optimal for the provided function parameters.
- From a functionality point of view, PL/pgSQL is much more powerful than SQL.
- PL/pgSQL supports several features that SQL functions cannot support, including the following:
  - It provides the ability to raise exceptions and to raise messages at different levels, such as notice and debug.
  - It supports the construction of dynamic SQL, using the `EXECUTE` command.
  - It provides `EXCEPTION` handling.
  - It has a complete set of assignment, control, and loop statements.
  - It supports cursors.
  - It is fully integrated with the PostgreSQL trigger system. SQL functions cannot be used with triggers.