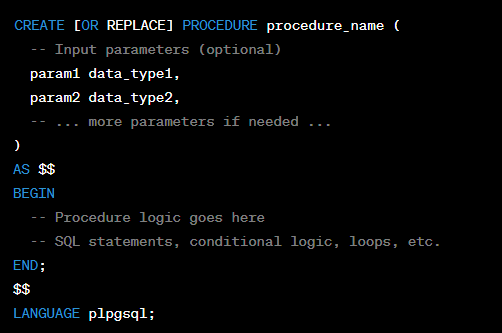
A stored procedure in PostgreSQL is a collection of SQL commands that are grouped together to perform a specific task. Stored procedures can be used to improve the performance of an application, to encapsulate complex logic, and to make data access more secure.

***Structure:***



* CREATE [OR REPLACE] PROCEDURE: This statement is used to create a new stored procedure with the specified name. The OR REPLACE clause allows you to modify the existing procedure if it already exists with the same name.
* procedure\_name: Replace this with the desired name of your stored procedure.
* param1 data\_type1, param2 data\_type2, ...: These are input parameters that the stored procedure can accept. They are optional and can be used to pass values into the procedure. Each parameter has a name (e.g., param1) and a data type (e.g., INTEGER, VARCHAR, etc.).
* AS $$: The $$ symbols are used as delimiters to enclose the body of the stored procedure. This allows you to include multiline code blocks without worrying about special characters or escape sequences.
* BEGIN: Marks the beginning of the procedural logic block.
* END: Marks the end of the procedural logic block.
* LANGUAGE plpgsql: This specifies the language used within the stored procedure. In this case, we are using PL/pgSQL, the procedural language for PostgreSQL.

***What type of input parameters PostgreSQL Stored Procedure can take?***

In PostgreSQL functions, you can use various types of input parameters to receive data and values from the caller. PostgreSQL supports a wide range of data types for function parameters, which allows you to handle different kinds of input data. Here are the types of input parameters that PostgreSQL functions can take:

1. Basic Data Types:

- Integer Types: `INTEGER`, `SMALLINT`, `BIGINT`

- Floating-Point Types: `REAL`, `DOUBLE PRECISION`

- Numeric Types: `NUMERIC`, `DECIMAL`

- Character Types: `CHAR`, `VARCHAR`, `TEXT`

- Boolean Type: `BOOLEAN`

2. Date and Time Types:

- `DATE`, `TIME`, `TIMESTAMP`, `INTERVAL`

3. Enumerated Types:

- User-defined enumeration types

4. Composite Types:

- User-defined composite types (a group of fields similar to a struct or record)

5. Array Types:

- Arrays of basic data types, composite types, or other arrays

6. Range Types:

- Continuous ranges of data types (e.g., `INT4RANGE`, `DATERANGE`)

7. Domain Types:

- User-defined domains based on existing data types

8. Special Types:

- `ANYELEMENT`: Accepts any data type

- `ANYARRAY`: Accepts any array type

- `ANYENUM`: Accepts any enumerated type

9. PostgreSQL Object Types:

- `JSON`, `JSONB`, `UUID`, `XML`, `TSVECTOR`, `HSTORE`, etc.

10. Composite Input Types:

- A function can take a single parameter of a composite type that encapsulates multiple values.

***What type of output parameter PostgreSQL stored procedure can have?***

A PostgreSQL stored procedure can have any of the following types of output parameters:

1. Scalar: A scalar output parameter is a single value. It can be any of the scalar types supported by PostgreSQL, such as integer, float, string, boolean, date, time, and timestamp.
2. Composite: A composite output parameter is a set of values. It can be any of the composite types supported by PostgreSQL, such as record, array, or table

***Limitations:***

1. Stored procedure doesn’t have any return type.
2. Can’t call stored procedure that has explicit transactional commands from java project.