PDP ACADEMY

PL/pgSQL (User-defined functions and Procedures)

Create Function

```
create [or replace] function function_name(param_list)
   returns return_type
   language plpgsql
  as
$$
declare
-- variable declaration
begin
 -- logic
end;
$$
```

Function Parameter Modes

The parameter modes determine the behaviors of parameters. PL/pgSQL supports three parameter modes: in, out, and inout. A parameter takes the **in mode** by default if you do not explicitly specify it.

IN	OUT	INOUT
The default	Explicitly specified	Explicitly specified
Pass a value to function	Return a value from a function	Pass a value to a function and return an updated value.
in parameters act like constants	out parameters act like uninitialized variables	inout parameters act like an initialized variables
Cannot be assigned a value	Must assign a value	Should be assigned a value

PDP ACADEMY

Function overloading

PostgreSQL allows multiple functions to share the same name as long as they have different arguments. If two or more functions share the same name, the function names are overloaded.

When you can call an overloading function, PostgreSQL select the best candidate function to execute based on the the function argument list.

Note: Parenthesized type modifiers are discarded by CREATE FUNCTION.

Postgresql documentation

PDP ACADEMY

Returning table

```
create [or replace] function function_name(parameter(s))
returns table ( column(s) )
language plpgsql
as $$
declare
-- variable declaration
begin
-- body
end; $$
```

PDP ACADEMY

Drop function

```
drop function [if exists] function_name(argument(s)) [cascade | restrict]
```

Procedures

A drawback of user-defined functions is that they cannot execute transactions. In other words, inside a user-defined function, you cannot start a transaction, and commit or rollback it.

PostgreSQL 11 introduced stored procedures that support transactions. To define a **new stored procedure**, you use the **create procedure statement**.

```
create [or replace] procedure procedure_name(parameter(s))
language plpgsql
as $$
declare
-- variable declaration
begin
-- stored procedure body
end; $$
```

PDP ACADEMY

Drop procedure

```
drop procedure [if exists] procedure_name (argument(s)) [cascade | restrict]
```

PDP ACADEMY

Interview Questions