

Practical 3

Aim:

3.5 Write a PL/SQL Function (StoredProcedure) to retrieve all the student information whose branch is CSE and using java program display all selected records on console.

Description:

PostgreSQL **functions**, also known as Stored Procedures, allow you to carry out operations that would normally take several queries and round trips in a single function within the database. Functions allow database reuse as other applications can interact directly with your stored procedures instead of a middle-tier or duplicating code.

Functions can be created in language of your choice like SQL, PL/pgSQL, C, Python, etc.

Syntax

The basic syntax to create a function is as follows:

```
CREATE [OR REPLACE] FUNCTION function_name (arguments)
RETURNS return_datatype AS $variable_name$
DECLARE
    declaration;
    [...]
BEGIN
    < function_body >
    [...]
    RETURN { variable_name | value }
END; LANGUAGE plpgsql;
```

Where,

- **function-name** specifies the name of the function.
- [OR REPLACE] option allows modifying an existing function.
- The function must contain a **return** statement.
- **RETURN** clause specifies that data type you are going to return from the function. The **return_datatype** can be a base, composite, or domain type, or can reference the type of a table column.

- **function-body** contains the executable part.
- The AS keyword is used for creating a standalone function.
- **plpgsql** is the name of the language that the function is implemented in. Here, we use this option for PostgreSQL, it Can be SQL, C, internal, or the name of a user-defined procedural language. For backward compatibility, the name can be enclosed by single quotes.

Program code:

JC5.java

```
import java.sql.*;
import java.util.*;
class JC5{
public static void main(String[] args){
    Connection c=null;
    PreparedStatement s=null;
    try{
        MyDb my=new MyDb();
        c=my.connectMyDb("s2b130050131031");
        s=c.prepareStatement("select display()");
        ResultSet r=s.executeQuery();
        System.out.println("130050131031\nid\tname\tbranch");
        while(r.next()){

            System.out.println( r.getString("display"));

        }
    }
    catch(Exception e){
        System.out.println(e.getMessage());
    }
}
}
```

PLPGSQL Function:

```
CREATE OR REPLACE FUNCTION public.display()
    RETURNS TABLE(id integer, name character varying, branch character varying) AS
$BODY$
BEGIN RETURN QUERY
SELECT * FROM student WHERE student.branch = 'cse';
END;
```

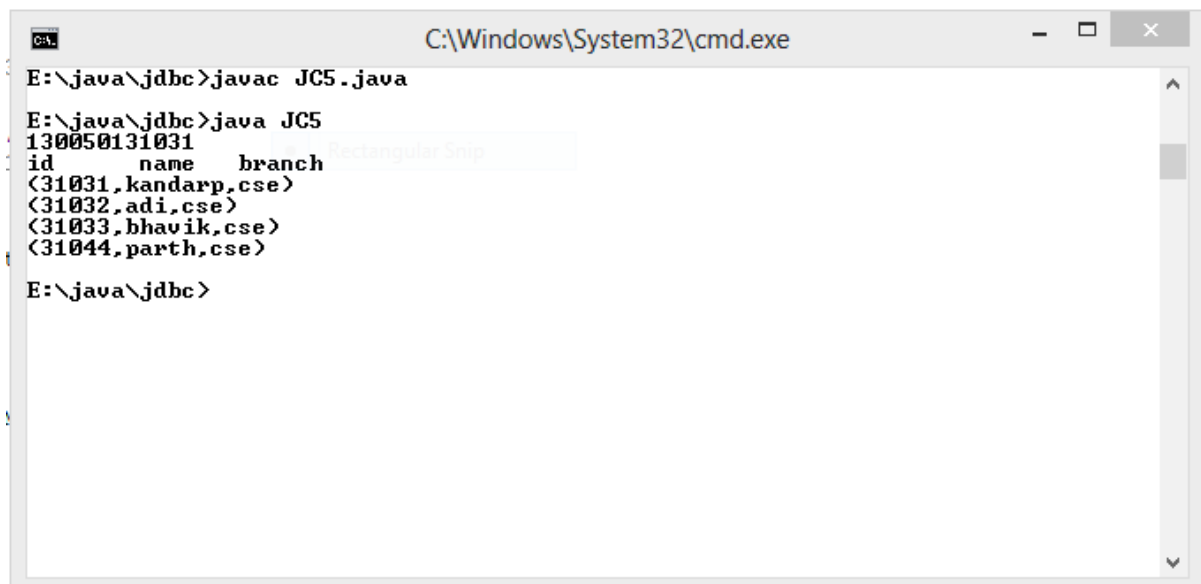
\$BODY\$
LANGUAGE plpgsql STABLE

MyDB.java

```
import java.sql.*;
class MyDb{
    public Connection connectMyDb(String database){
        Connection c=null;
        try{
            Class.forName("org.postgresql.Driver");

c=DriverManager.getConnection("jdbc:postgresql://localhost:5432/"+database,
"postgres","1014");
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
        return c;
    }
}
```

Input Output:



```
C:\Windows\System32\cmd.exe

E:\java\jdbc>javac JC5.java

E:\java\jdbc>java JC5
130050131031
id      name      branch
<31031,kandarp,cse>
<31032,adi,cse>
<31033,bhavik,cse>
<31044,parth,cse>

E:\java\jdbc>
```