unionbank

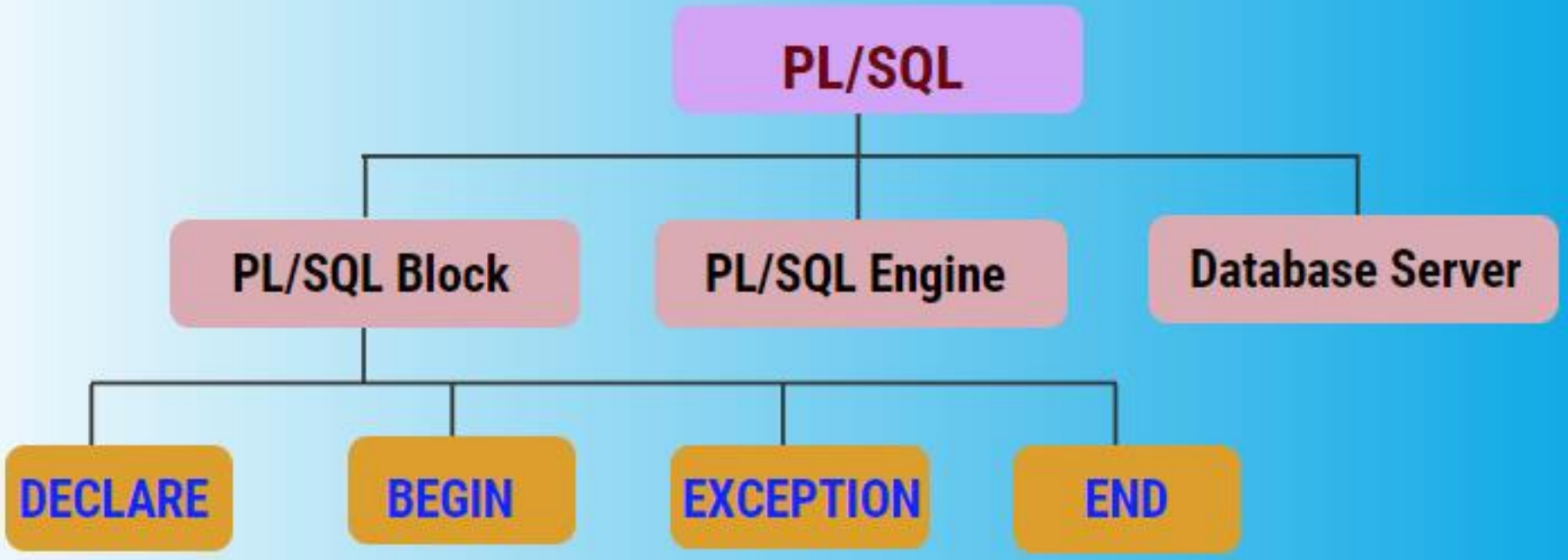
TURBOCHARGING **PRODUCTIVITY**

PL/SQL Training

Course Content

- Declaring PL/SQL variables
- Writing executable statements: Introducing stored procedures and functions
- SQL statements in PL/SQL programs
- Writing control structures
- Working with composite data types using explicit cursors
- Handling exceptions

What is PL/SQL?



www.educba.com

What is PL/SQL?

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Disadvantages of SQL:

- SQL doesn’t provide the programmers with a technique of condition checking and looping.
- SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.
- SQL has no facility of error checking during manipulation of data.

What is PL/SQL?

Features of PL/SQL:

- PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
- PL/SQL can execute several queries in one block using single command.
- One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
- PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
- Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
- PL/SQL Offers extensive error checking.

What is PL/SQL?

Differences between SQL and PL/SQL:

SQL	PL/SQL
SQL is a single query that is used to perform DML and DDL operations.	PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
It is declarative, that defines what needs to be done, rather than how things need to be done.	PL/SQL is procedural that defines how the things needs to be done.
Execute as a single statement.	Execute as a whole block.
Mainly used to manipulate data.	Mainly used to create an application.
Cannot contain PL/SQL code in it.	It is an extension of SQL, so it can contain SQL inside it.

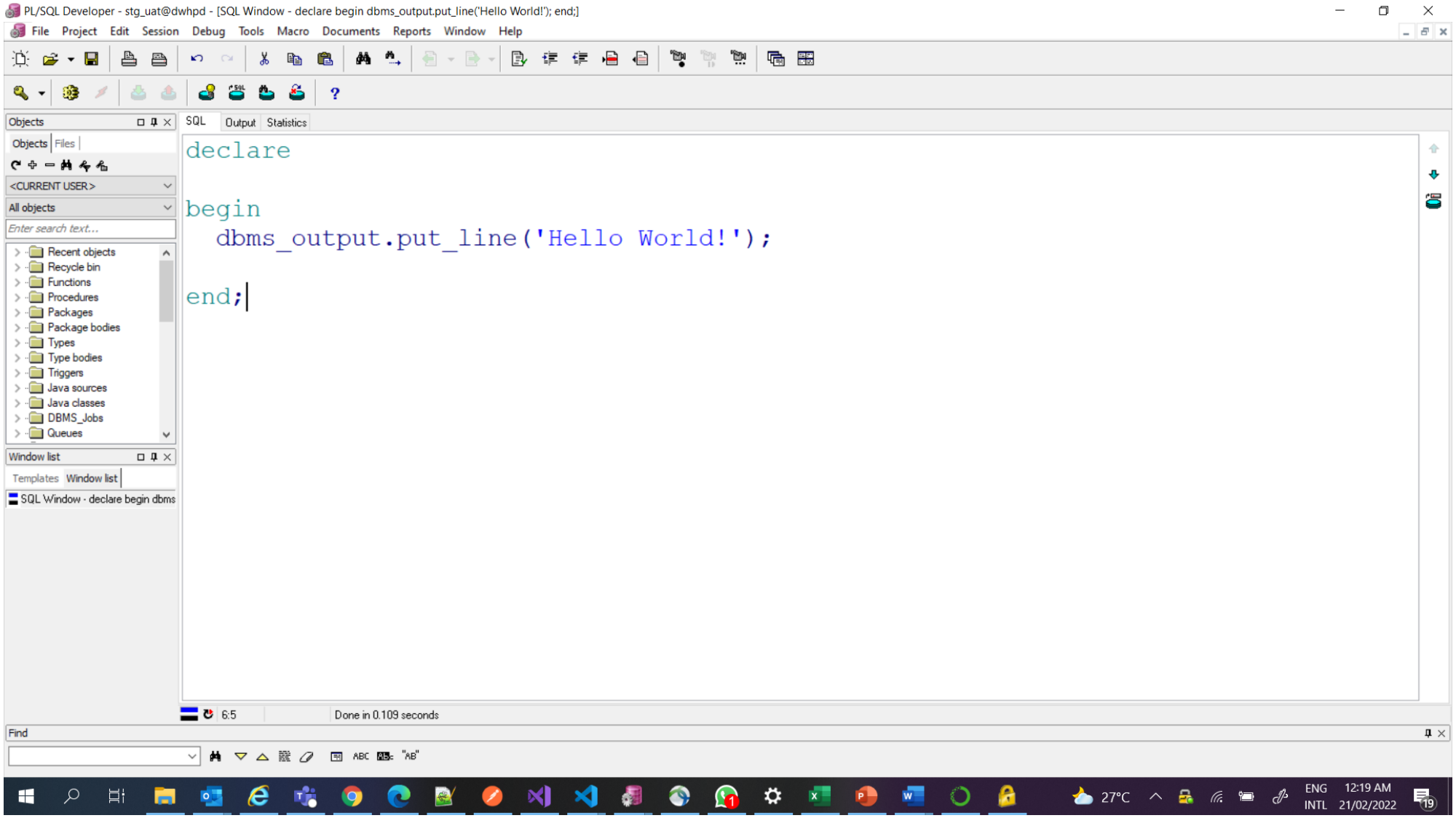
PL/SQL Structure

Structure of PL/SQL Block:

```
DECLARE (optional)
    declaration statements;

BEGIN (mandatory)
    executable statements

EXCEPTIONS (optional)
    exception handling statements

END (mandatory);
```

- Declare section starts with **DECLARE** keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.
- Execution section starts with **BEGIN** and ends with **END** keyword. This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands and DDL commands.
- Exception section starts with **EXCEPTION** keyword. This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

PL/SQL Structure



Declaring PL/SQL Variables

Declaring PL/SQL Variables

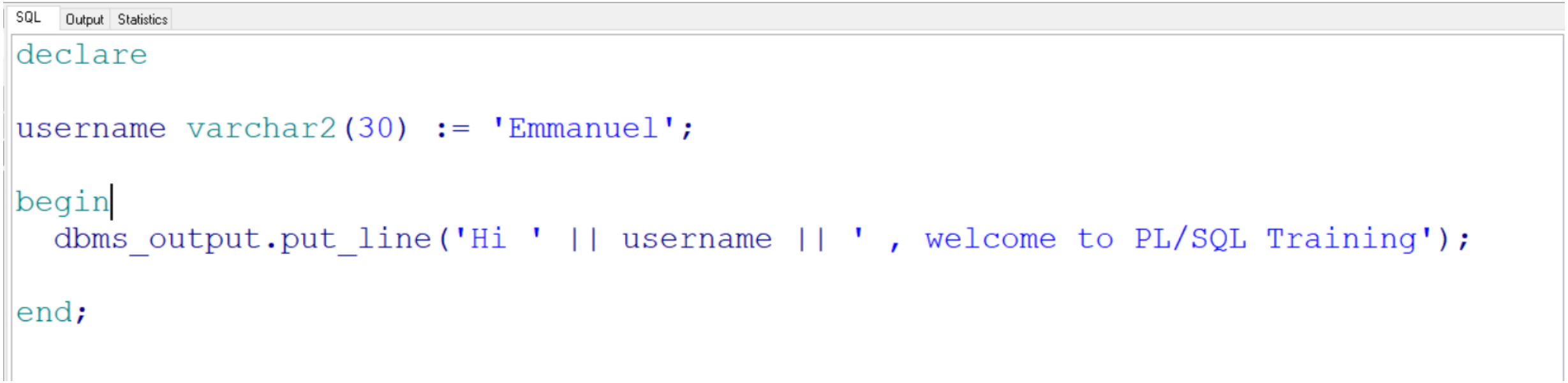
Like several other programming languages, variables in PL/SQL must be declared prior to its use.

They should have a valid name and data type as well.

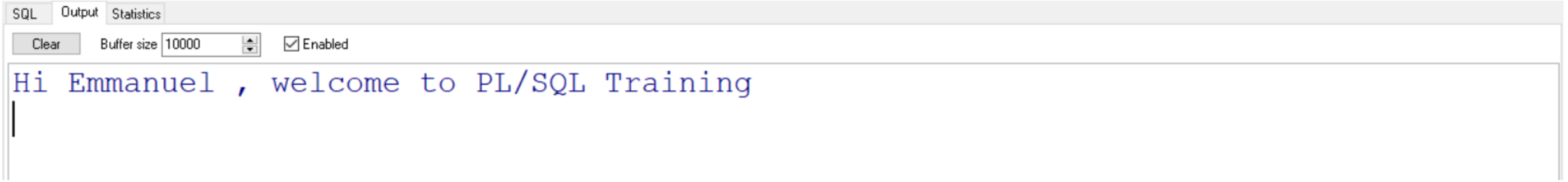
Syntax for declaration of variables:

```
variable_name datatype [NOT NULL := value ];
```

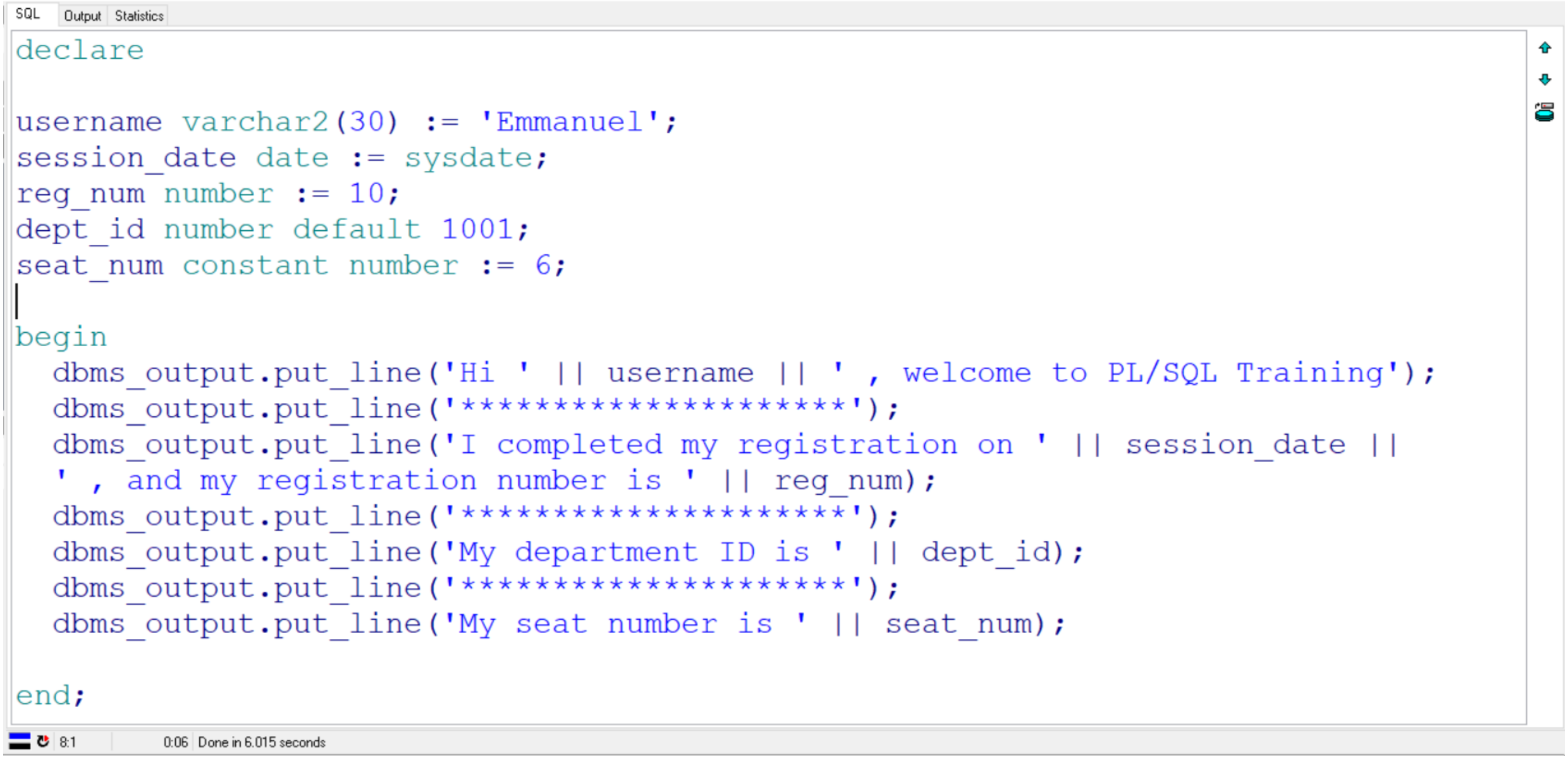
Declaring PL/SQL Variables



Output

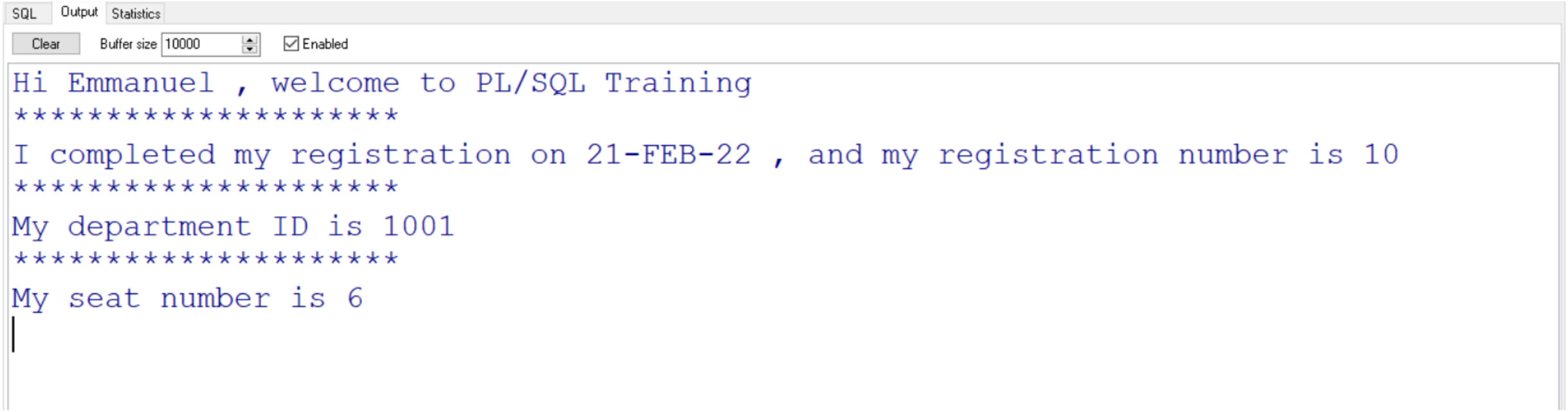


Declaring PL/SQL Variables



Declaring PL/SQL Variables

Output



Declaring PL/SQL Variables

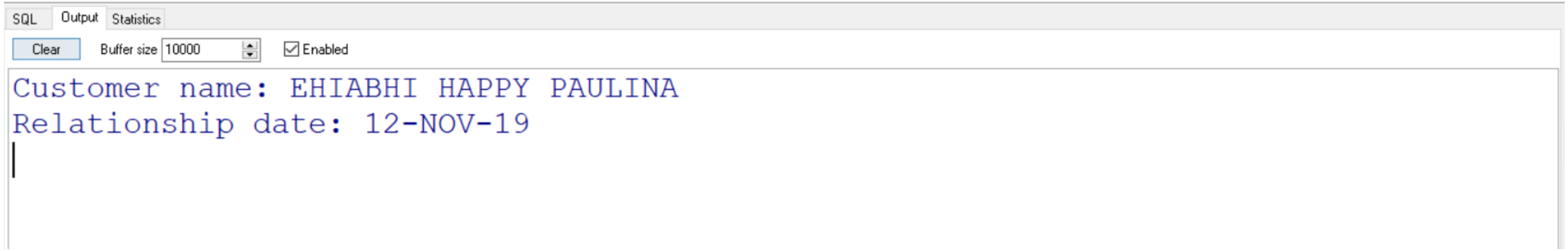
SQL | Output | Statistics

```
declare
  account_num        varchar2(10) := '0124495895';
  customer_num       varchar2(9);
  cust_name          varchar2(100);
  relationship_date  date;
begin
  select cust_no
    into customer_num
    from account_master
   where cust_ac_no = account_num;

  select customer_name, cif_creation_date
    into cust_name, relationship_date
    from customer_master
   where customer_no = customer_num;

  dbms_output.put_line('Customer name: ' || cust_name);
  dbms_output.put_line('Relationship date: ' || relationship_date);
end;
```

3:18 | 0:05 | Done in 5.407 seconds

Declaring PL/SQL Variables

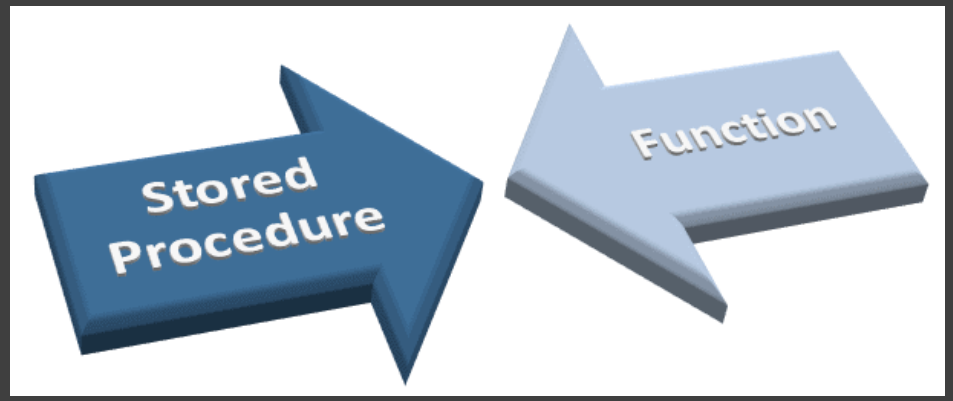
Output



Exercise 1

1. Write a program to declare 3 variables with datatype as below and display their values.
    - Number
    - Varchar
    - Date
2. Write a program to compute the area of a circle of radius 7 cm. (Formula is $A = \pi r^2$, use $\pi = 3.142$). Display the are of the circle.
3. Write a program to calculate 25% increase in sales for a sale amount of N200,000. Display the new sales amount.
4. Write a program to calculate customer's relationship age in months for the account number – 0124495895. Display the customer's account number, customer ID, customer name and relationship age.

Writing Executable Statements: Introducing Stored Procedures And Functions



Stored Procedures & Functions

A stored procedure or function in PL/SQL is nothing but a series of declarative SQL statements which can be stored in the database catalogue. A procedure can be thought of as a function or a method. They can be invoked through triggers, other procedures, or applications on Java, PHP etc. All the statements of a block are passed to Oracle engine all at once which increases processing speed and decreases the traffic.

Advantages of a Stored Procedure or Function

1. **Improves Database Performance**
   - Compilation is automatically done by oracle engine.
   - Whenever the calling of procedure or function is done, the oracle engine loads the compiled code into a memory which makes execution faster.

Stored Procedures & Functions

Advantages of a Stored Procedure or Function

2. **Provides Reusability and avoids redundancy**
   - The same block of code for procedure or function can be called any number of times for working on multiple data.
   - Due to which number of lines of code cannot be written repeatedly.

3. **Maintains Integrity**
   - Integrity means accuracy. Use of procedure or function ensures integrity because they are stored as database objects by the oracle engine.

Stored Procedures & Functions

Advantages of a Stored Procedure or Function

4. **Maintains Security**
   - Use of stored procedure or function helps in maintaining the security of the database as access to them and their usage can be controlled by granting access/permission to users while the permission to change or to edit or to manipulate the database may not be granted to users.

5. **Saves Memory**
   - Stored procedure or function have shared memory. Due to which it saves memory as a single copy of either a procedure or a function can be loaded for execution by any number of users who have access permission.

Stored Procedures & Functions

Difference between Stored Procedure and Function

Stored Procedure	Function
May or may not returns a value to the calling part of program.	Returns a value to the calling part of the program.
Uses IN, OUT, IN OUT parameter.	Uses only IN parameter.
Returns a value using “ OUT” parameter.	Returns a value using “RETURN”.
Does not specify the datatype of the value if it is going to return after a calling made to it.	Necessarily specifies the datatype of the value which it is going to return after a calling made to it.
Cannot be called from the function block of code.	Can be called from the procedure block of code.

Terminologies in PL/SQL Subprograms

Parameter:

The parameter is variable or placeholder of any valid PL/SQL datatype through which the PL/SQL subprogram exchange the values with the main code. This parameter allows to give input to the subprograms and to extract from these subprograms.

These parameters should be defined along with the subprograms at the time of creation.

These parameters are included in the calling statement of these subprograms to interact the values with the subprograms.

The datatype of the parameter in the subprogram and the calling statement should be same.

Based on their purpose parameters are classified as

Terminologies in PL/SQL Subprograms

IN Parameter:

- This parameter is used for giving input to the subprograms.
- It is a read-only variable inside the subprograms. Their values cannot be changed inside the subprogram.
- In the calling statement, these parameters can be a variable or a literal value or an expression, for example, it could be the arithmetic expression like ‘5*8’ or ‘a/b’ where ‘a’ and ‘b’ are variables.
- By default, the parameters are of IN type.

OUT Parameter:

- This parameter is used for getting output from the subprograms.
- It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.

Terminologies in PL/SQL Subprograms

OUT Parameter:

- In the calling statement, these parameters should always be a variable to hold the value from the current subprograms.

IN OUT Parameter:

- This parameter is used for both giving input and for getting output from the subprograms.
- It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.
- In the calling statement, these parameters should always be a variable to hold the value from the subprograms.

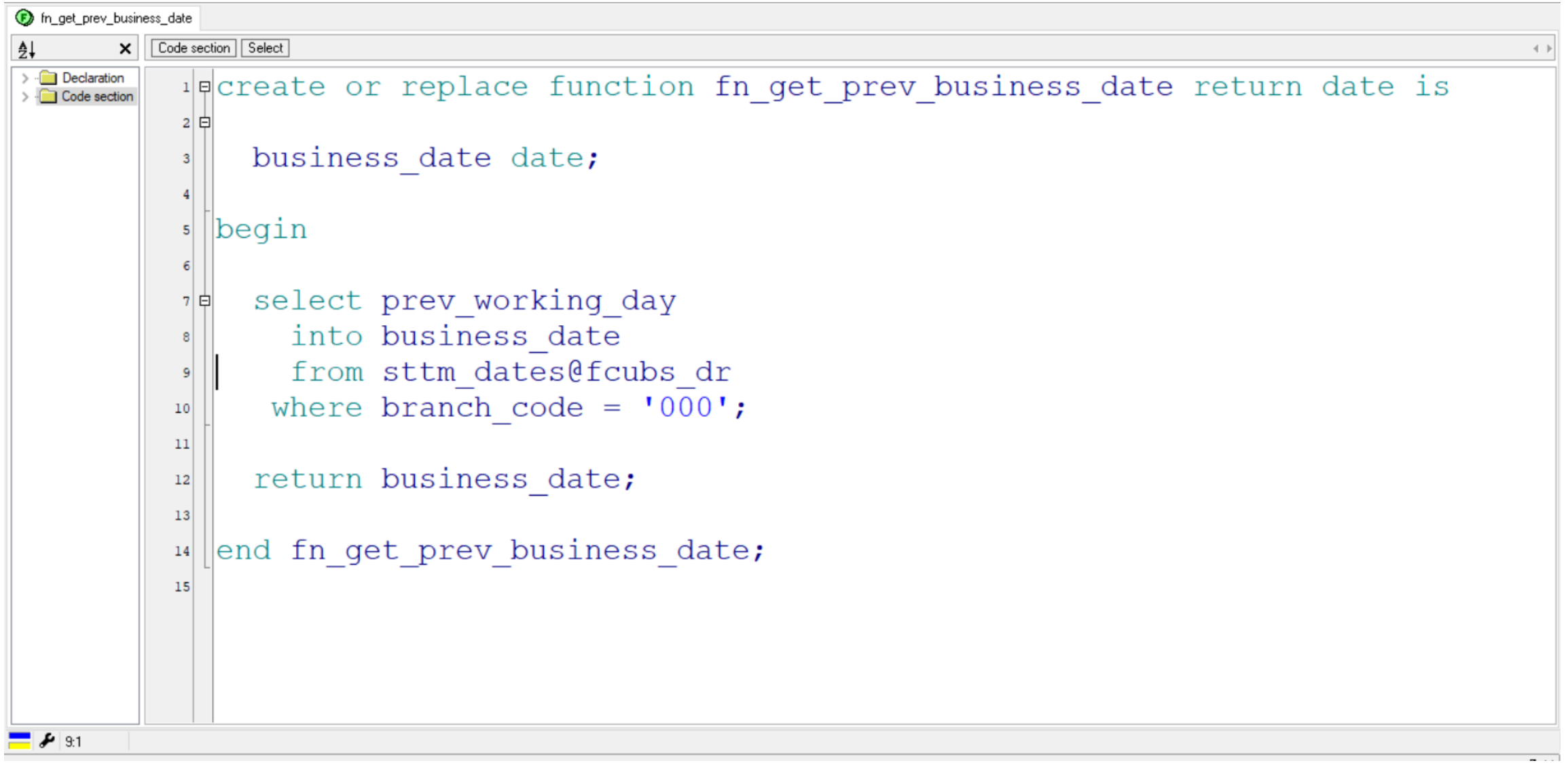
Syntax – Stored Procedure

```
CREATE OR REPLACE PROCEDURE

<procedure_name>
        (
        <parameter IN/OUT <datatype>
        ..
        .
        )

[ IS | AS ]
        <declaration_part>

BEGIN
        <execution part>

EXCEPTION
        <exception handling part>

END <procedure_name>;
```

Syntax – Function

```
CREATE OR REPLACE FUNCTION

<function_name>
(
<parameter IN <datatype>
)

RETURN <datatype>

[ IS | AS ]
        <declaration_part>

BEGIN
        <execution part>

EXCEPTION
        <exception handling part>

END;
```

Syntax – Function (example)



Syntax – Function (example)

D: > DAO > DAO Webinar > function_example.sql

```sql
create or replace function fn_create_new_user(p_username in varchar2)
  return varchar2 is

  user_status varchar2(100);
  user_count  number;

begin
  insert into table_name
    select sys_guid() || upper(username) user_id,
           full_name,
           sysdate maker_date_time,
           'false' login_status,
           0 failed_password_count
      from stg_uat.ubn_employee_details@linkcognos
     where trim(lower(username)) = p_username
       and trim(lower(username)) not in
           (select username from alero_users where username = p_username);
  commit;

  select count(*)
    into user_count
    from alero_users
   where username = p_username
     and trunc(maker_date_time) = trunc(sysdate);

  if user_count = 0 then
    user_status := 'User already exist';
  elsif user_count = 1 then
    user_status := 'User created successfully';
  elsif user_count > 1 then
    user_status := 'User have multiple profile, kindly review';
  else
    user_status := 'Check user status';
  end if;

  return user_status;
end fn_create_new_user;
```

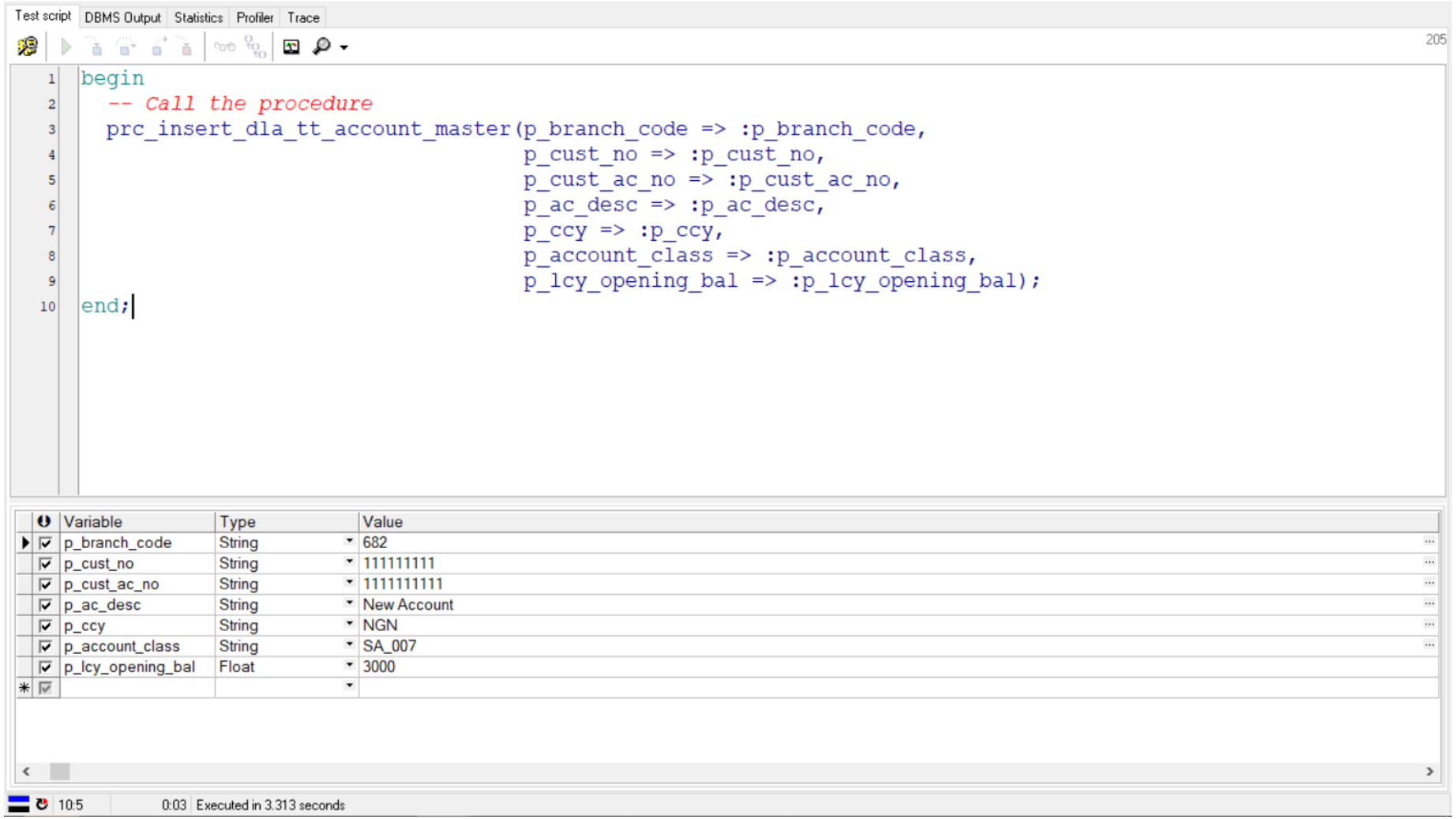
Syntax – Stored Procedure (example)

C: > Users > eokon > Downloads > procedure_sample.sql

```
create or replace procedure prc_insert_dla_tt_account_master(p_branch_code     varchar2,
                                                             p_cust_no         varchar2,
                                                             p_cust_ac_no      varchar2,
                                                             p_ac_desc         varchar2,
                                                             p_ccy             varchar2,
                                                             p_account_class   varchar2,
                                                             p_lcy_opening_bal number) is

  v_lcy_opening_bal number := p_lcy_opening_bal + 5000;
  --v_lcy_opening_bal number;

begin

  --v_lcy_opening_bal := p_lcy_opening_bal + 5000;

  insert into dla_tt_account_master
    (branch_code,
     cust_no,
     cust_ac_no,
     ac_desc,
     ccy,
     account_class,
     ac_open_date,
     lcy_opening_bal)
  values
    (p_branch_code,
     p_cust_no,
     p_cust_ac_no,
     p_ac_desc,
     p_ccy,
     p_account_class,
     sysdate,
     v_lcy_opening_bal);

  commit;

end prc_insert_dla_tt_account_master;
```

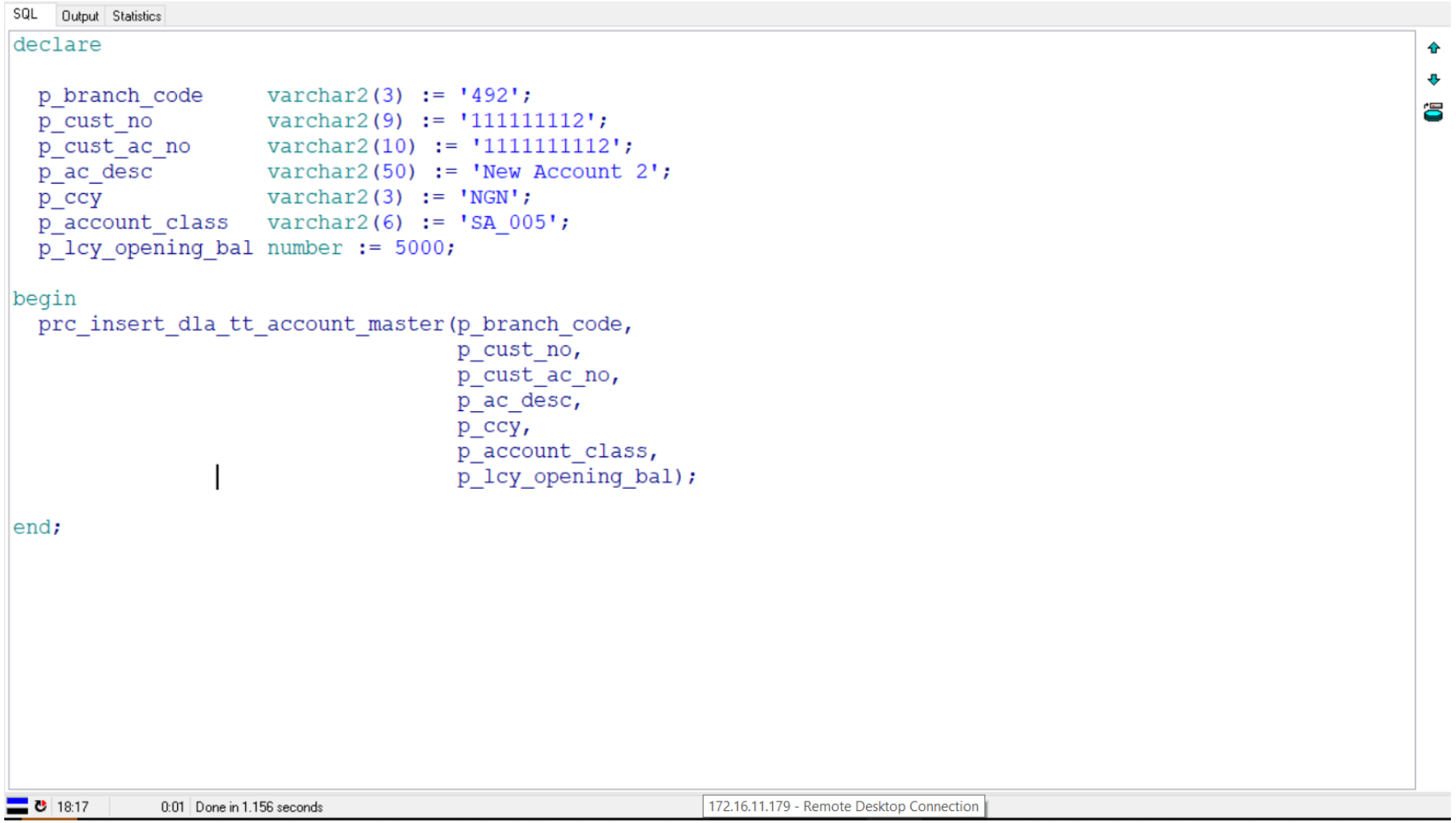
Sample procedure to insert into a table

Syntax – Stored Procedure (example)



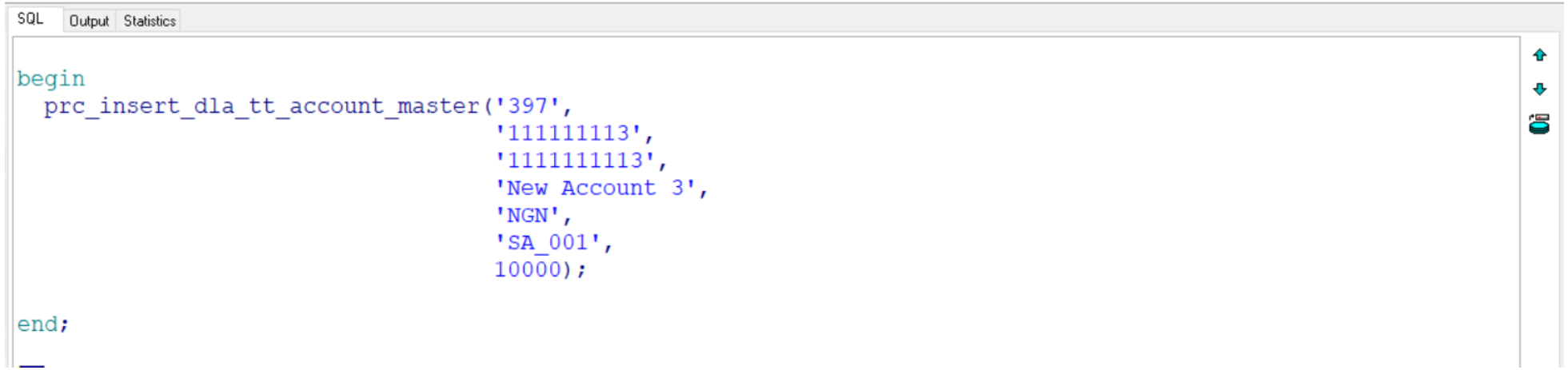
Sample procedure usage

Syntax – Stored Procedure (usage)



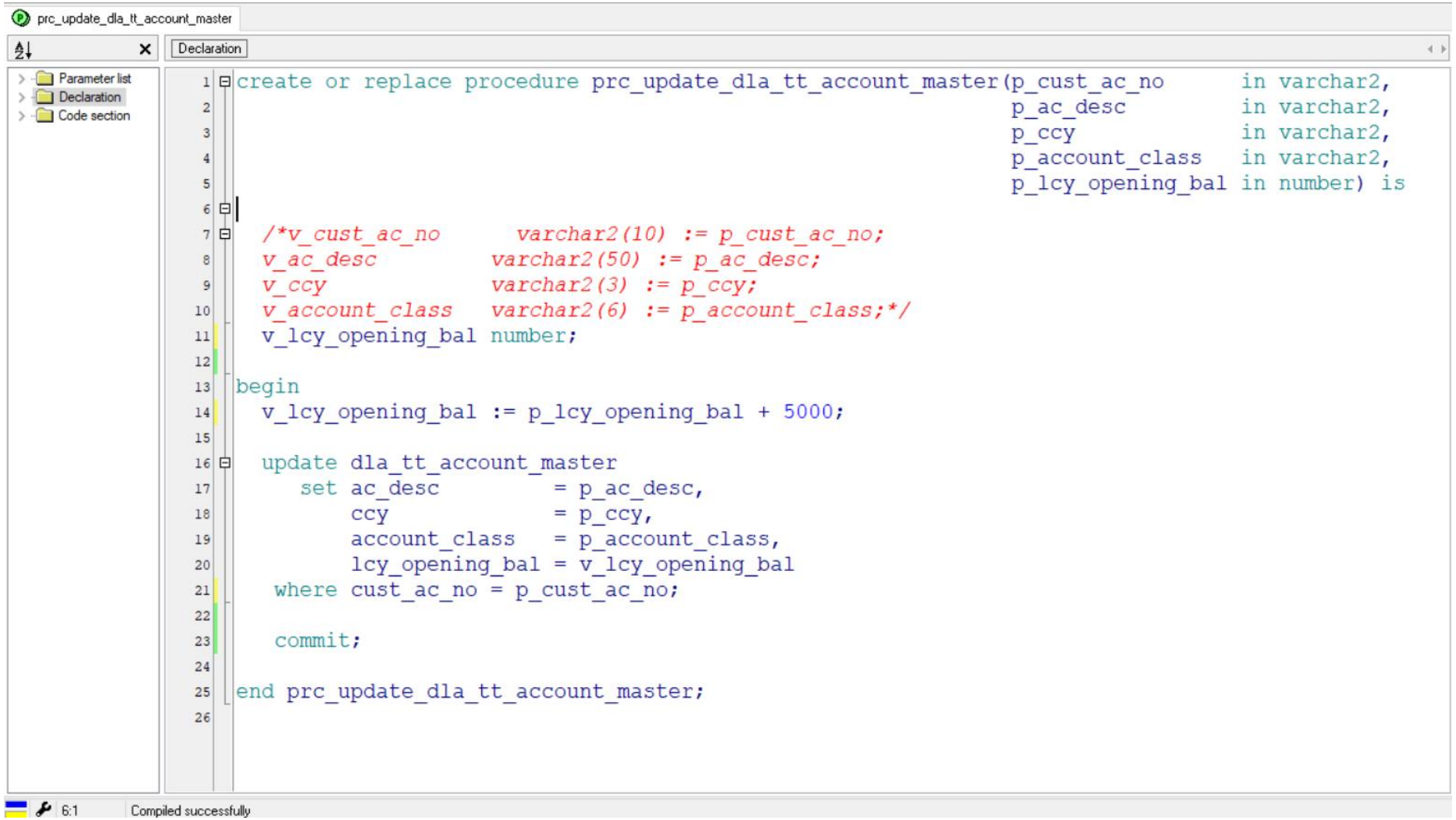
Sample procedure usage

Syntax – Stored Procedure (example)



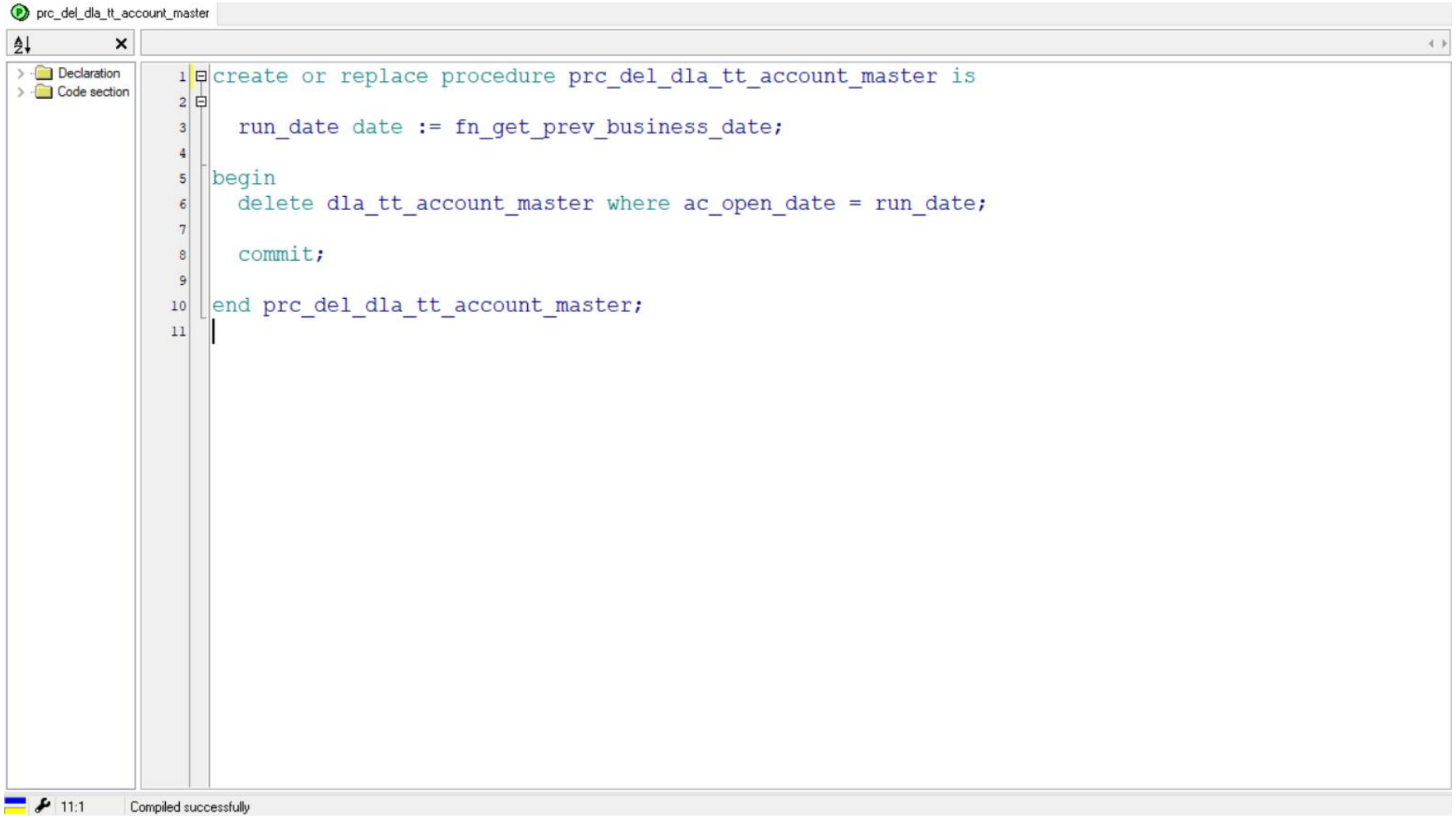
Sample procedure usage

Syntax – Stored Procedure (example)



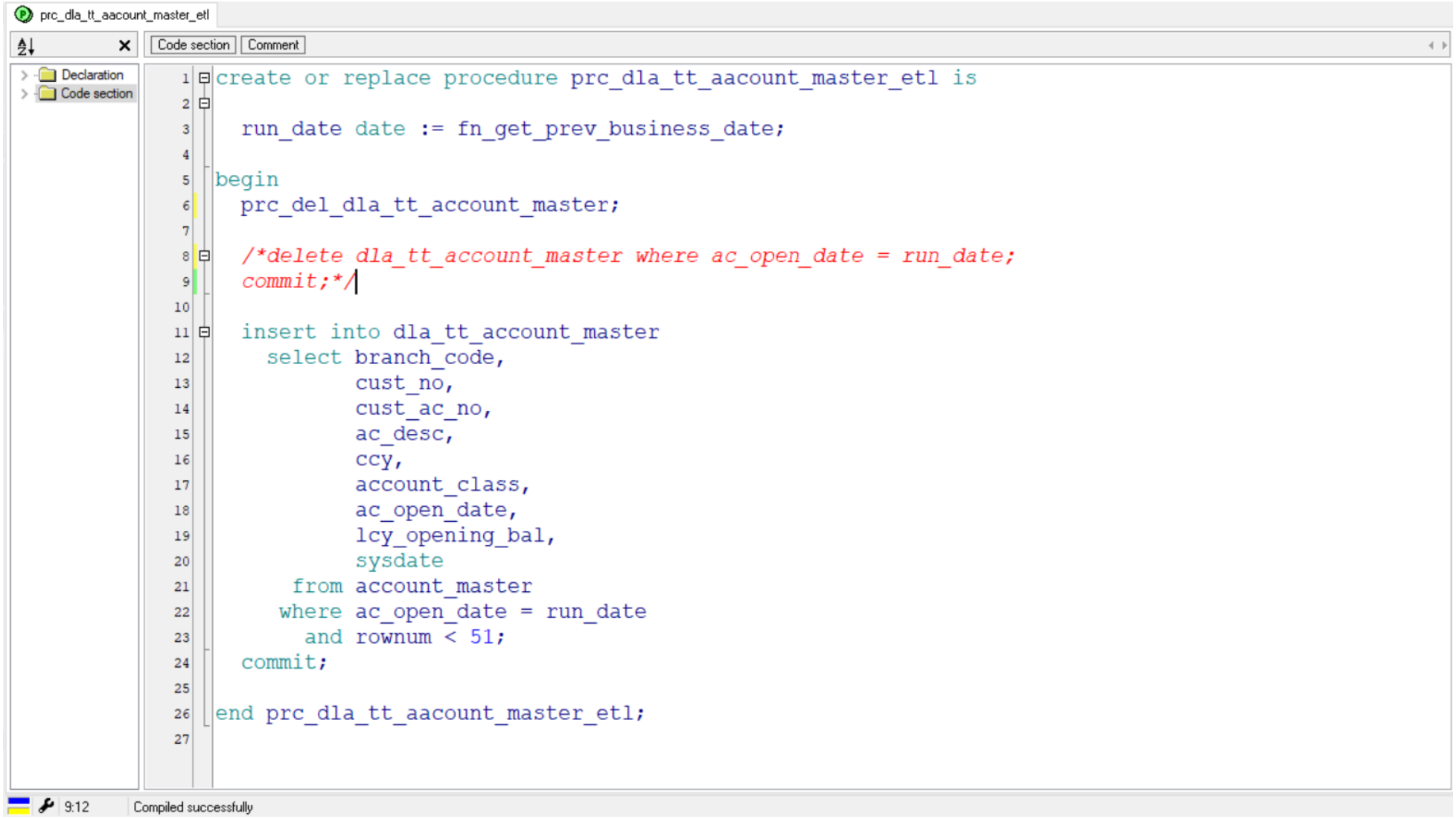
Sample procedure to update a record

Syntax – Stored Procedure (example)



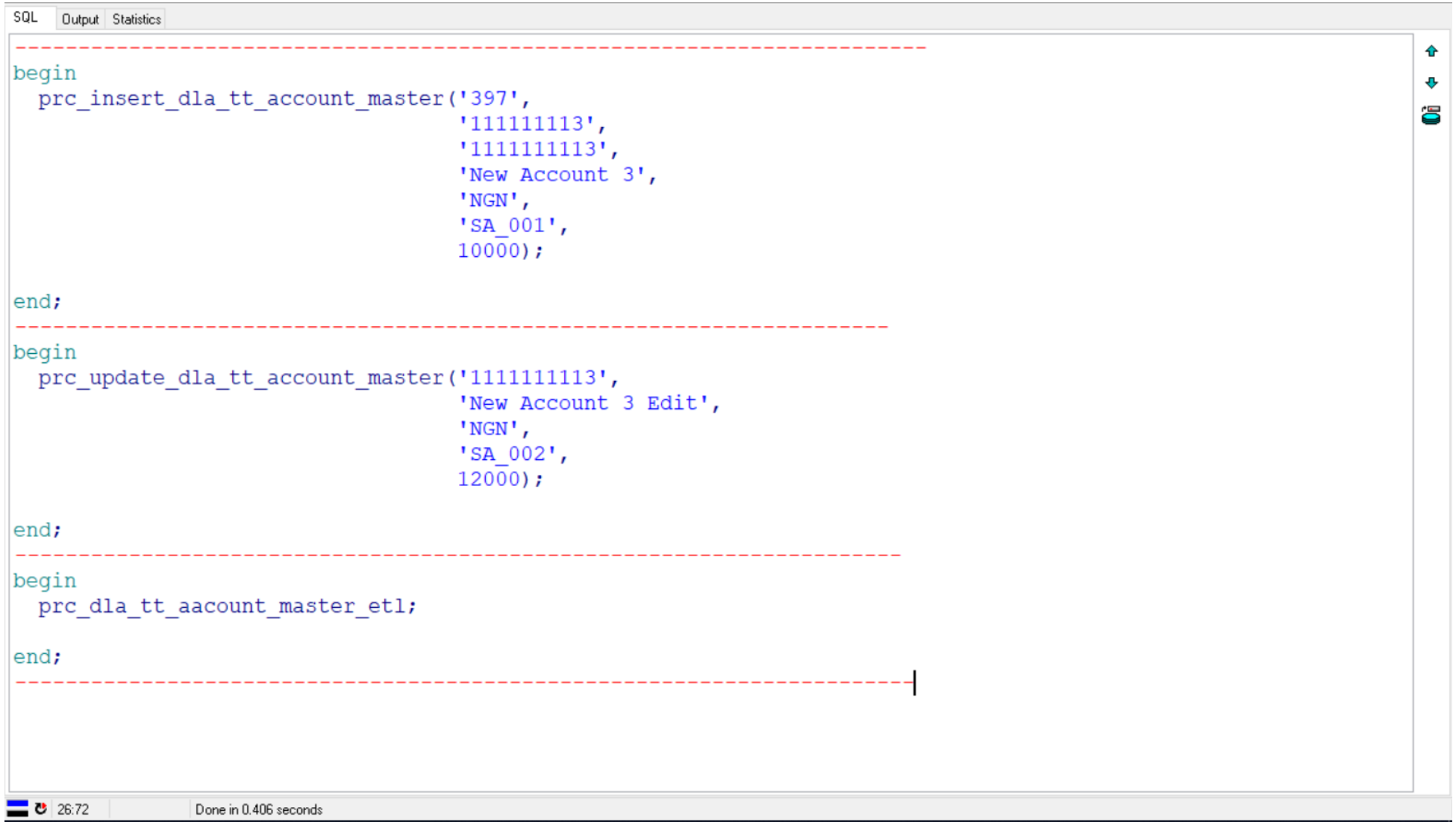
Sample procedure to delete a record

Syntax – Stored Procedure (example)



Sample procedure to show how to call a procedure in another procedure

Syntax – Stored Procedure (example)



Sample calls

Thank You