Harmony IT Solution

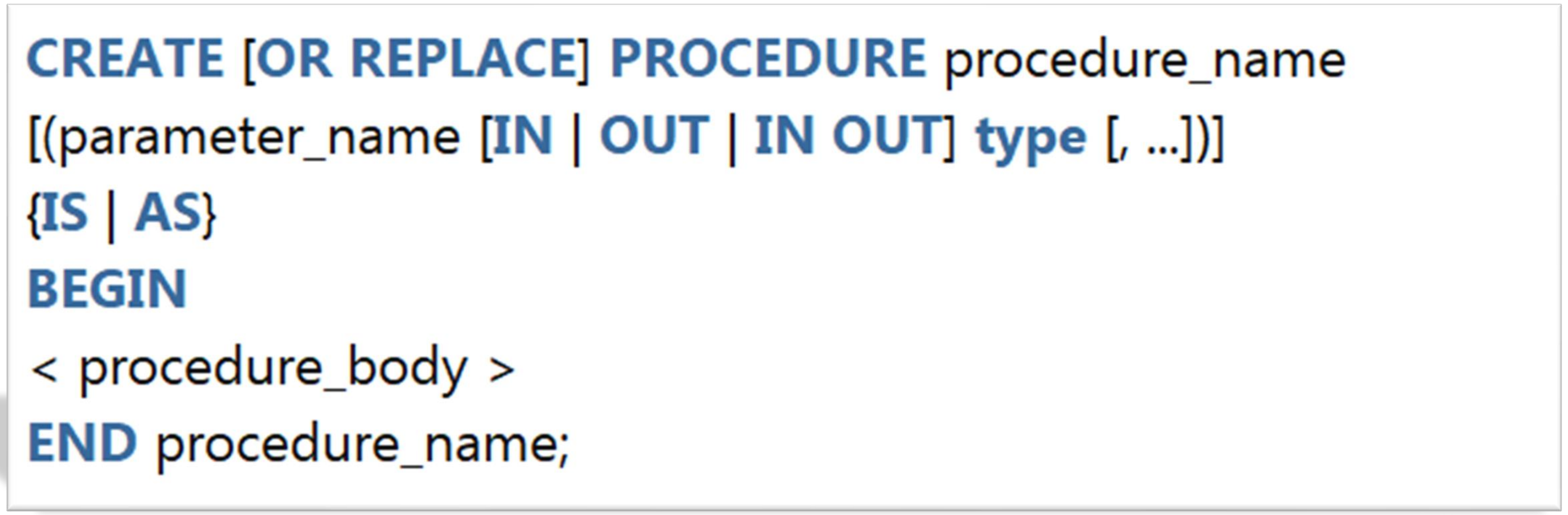
Database Design and Programming

Tahaluf Training Center 2022

Overview of Stored Procedure

- **A PL/SQL procedure** is a reusable unit used to encapsulate a specific business logic of the application.

- Procedure is a named block stored in the Oracle Database.



```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
< procedure_body >
END procedure_name;
```

Each parameter can be in either IN, OUT, or INOUT mode:

1. **IN parameter:** is read only, It means that if you do not specify the mode for a parameter explicitly, Oracle will use the IN mode.

2. **OUT parameter:** is writable, Setting a returned value for the OUT parameter and return it to call a program.

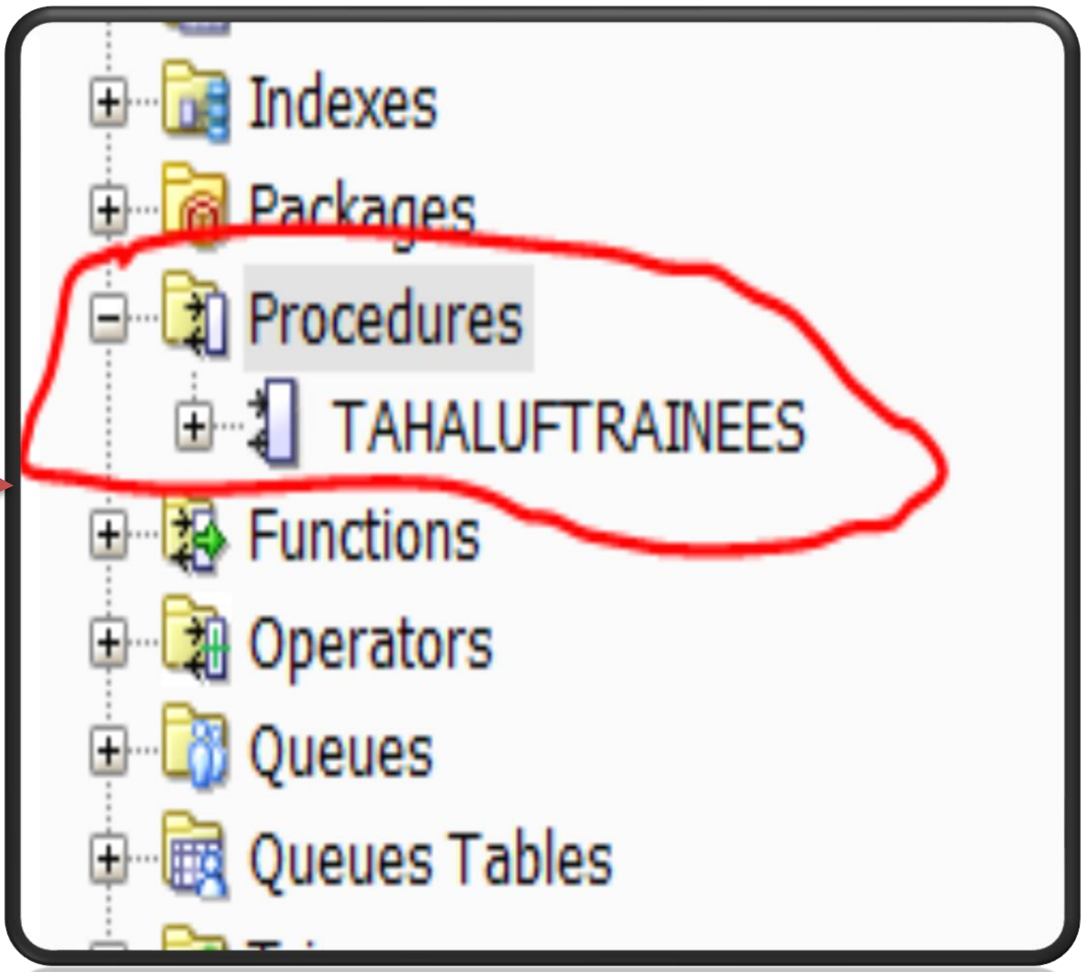
3. **INOUT parameter:** is both writable and readable. The procedure can modify and read it.

A procedure ignores the value supplies an OUT parameter.

Create Stored Procedure

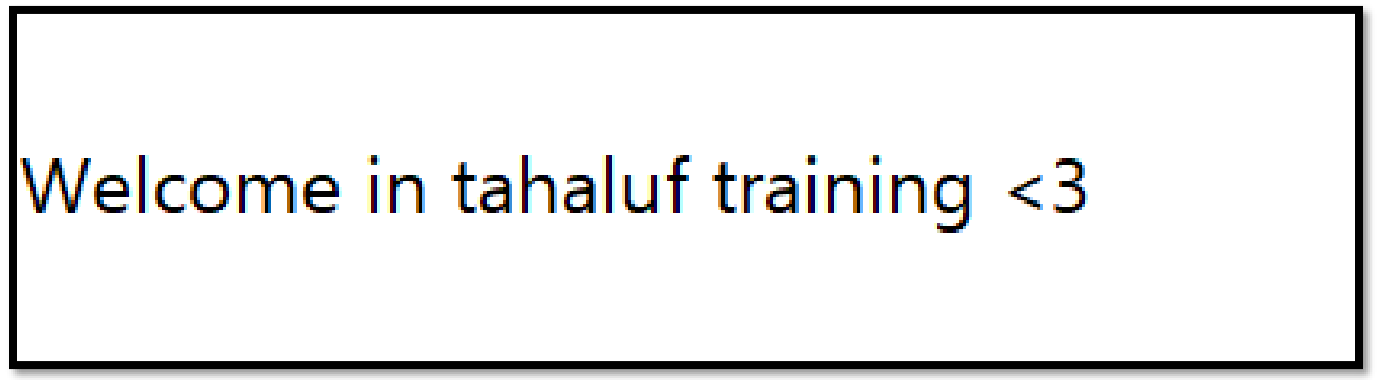
Example :

```
create or replace procedure TahalufTrainees
as
begin
 DBMS_Output.put_line('Welcome in tahaluf training <3 ');
end ;
```



Using Stored Procedure:

```
EXECUTE TahalufTrainees;
```



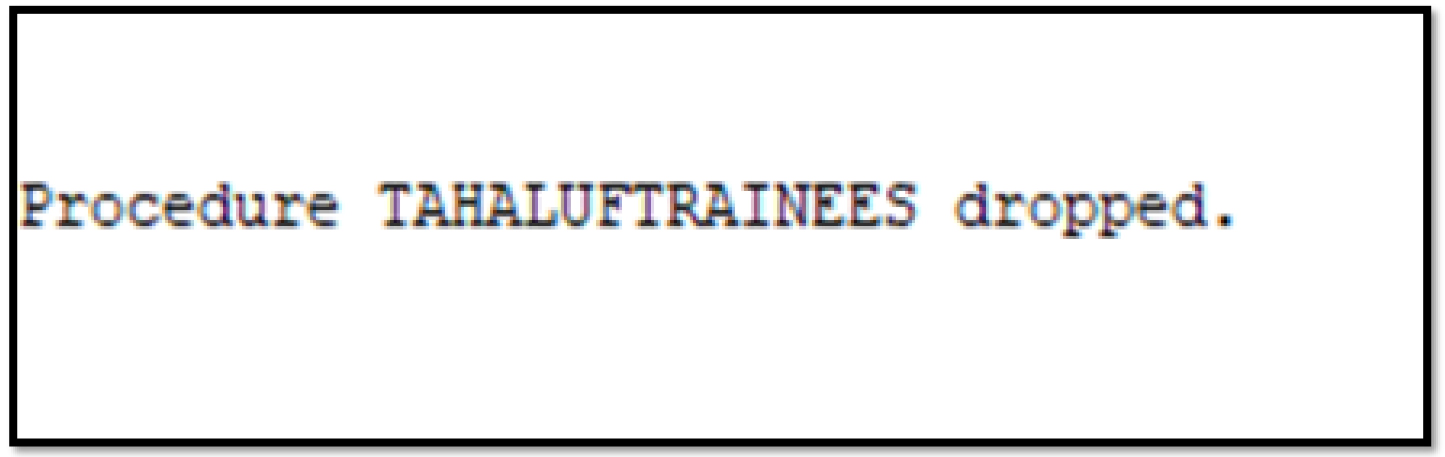
```
Welcome in tahaluf training <3
```

DROP Stored Procedure

```
DROP PROCEDURE procedure-name;
```

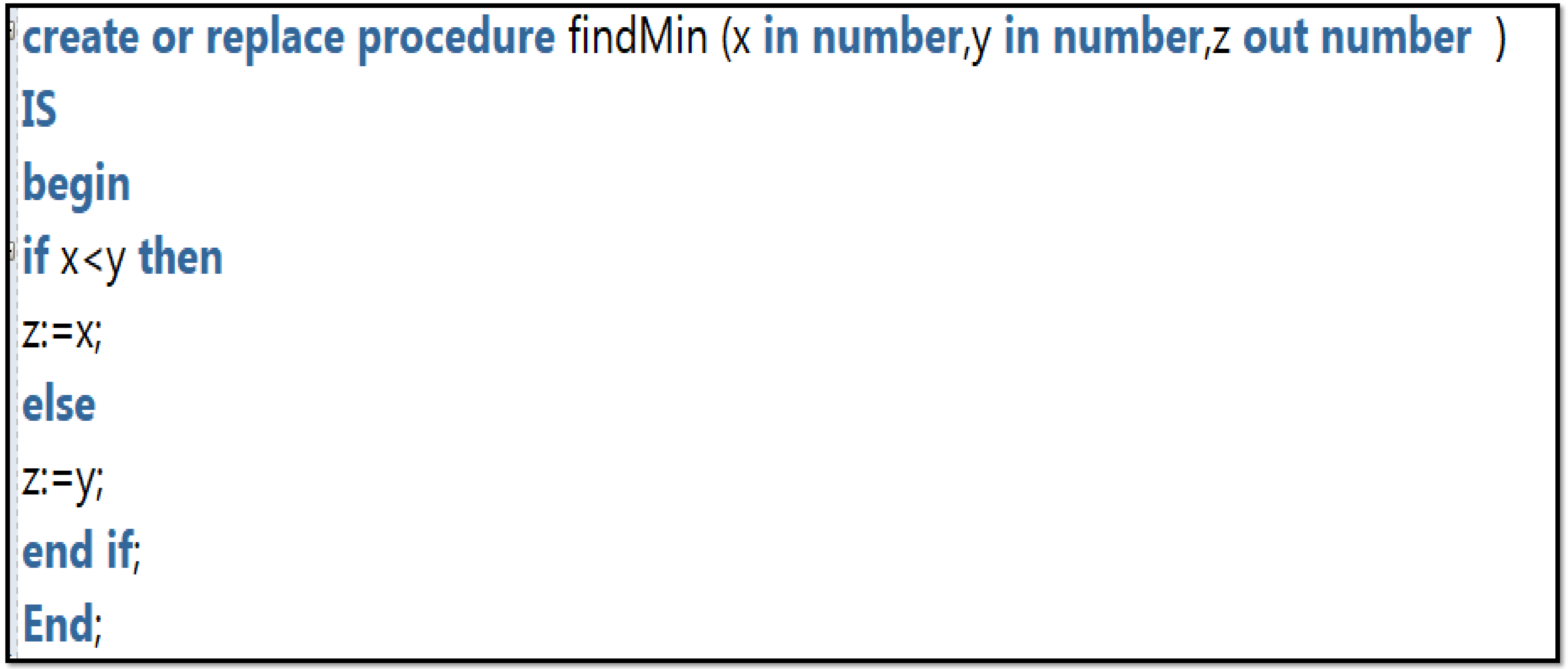
Example:

```
DROP PROCEDURE TahalufTrainees;
```



```
Procedure TAHALUFTRAINEES dropped.
```

Example :

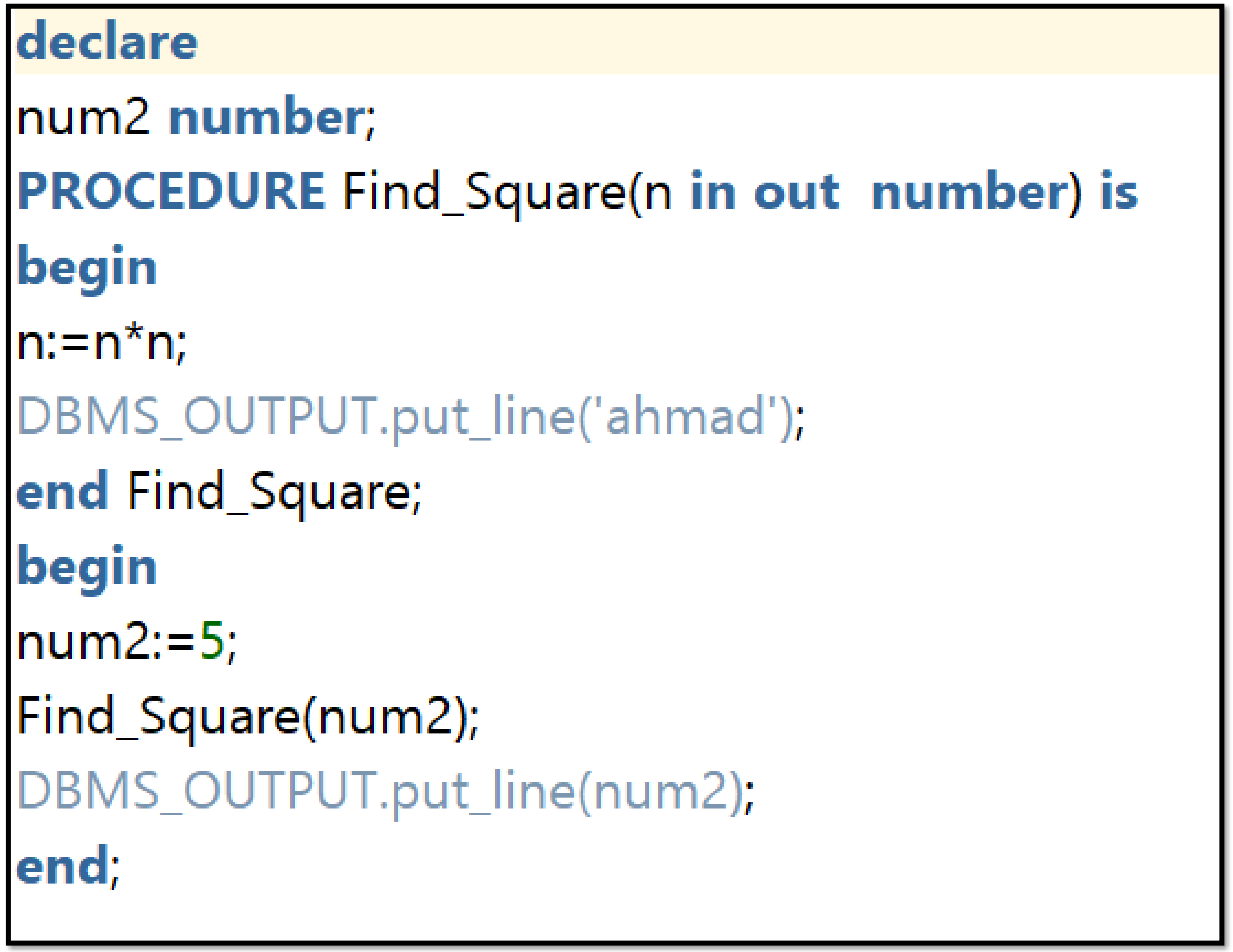


```
create or replace procedure findMin (x in number,y in number,z out number  )
IS
begin
if x<y then
z:=x;
else
z:=y;
end if;
End;
```

Use FindMin Procedure:

```
declare
a number;
b number;
c number;
begin
a:=23;
b:=45;
findMin(a,b,c);
 DBMS_Output.put_line('Minimum of (23,45)='||c);
 End;
```

Declare Stored Procedure without using **Create** :



```
declare
num2 number;
PROCEDURE Find_Square(n in out  number) is
begin
n:=n*n;
DBMS_OUTPUT.put_line('ahmad');
end Find_Square;
begin
num2:=5;
Find_Square(num2);
DBMS_OUTPUT.put_line(num2);
end;
```

Overview of Function

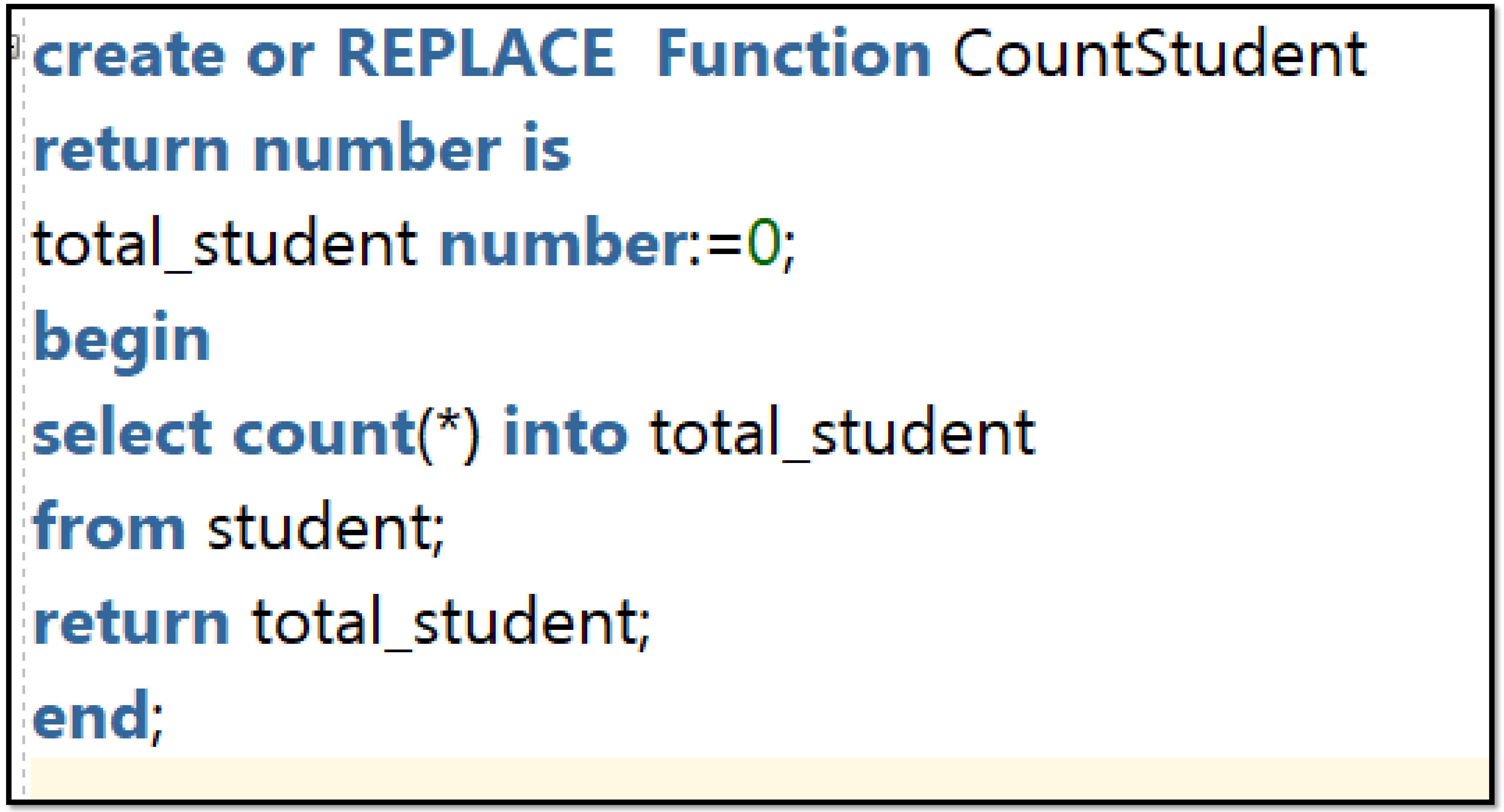
- **Function** is a multi-tenant, fully managed, on-demand, highly scalable and service platform.

- It is built on enterprise grade Cloud Infrastructure and powered by open source engine.

- Using Functions to focus on writing code to meet business needs.

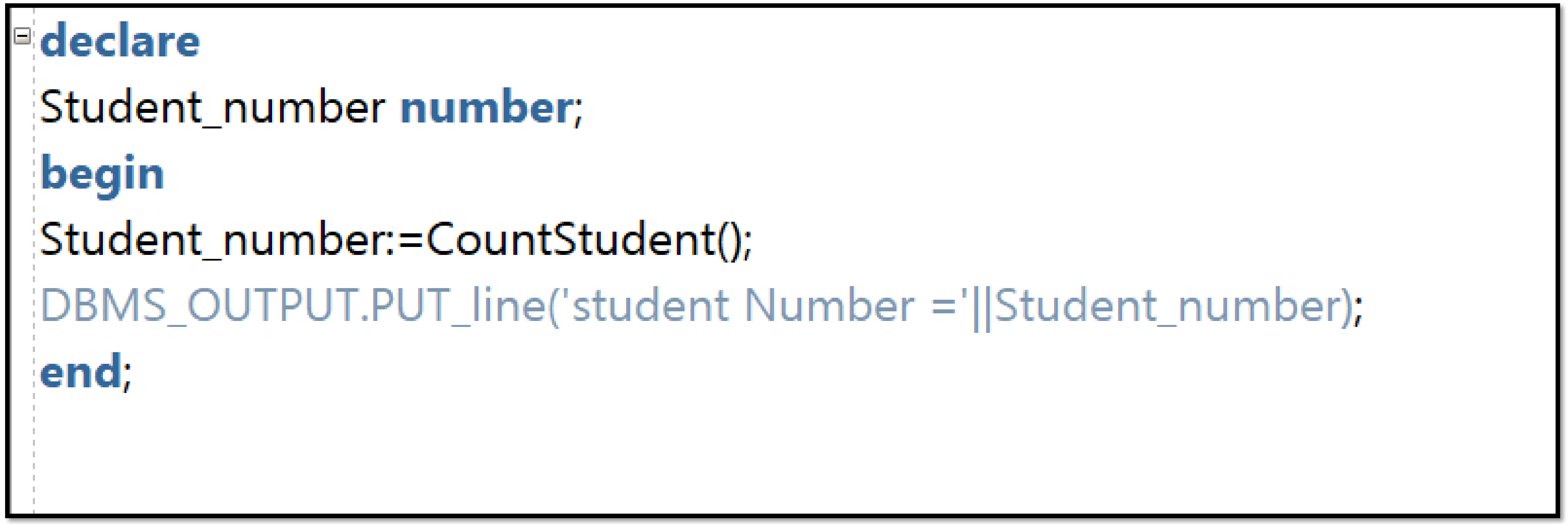
Create Function

Example :



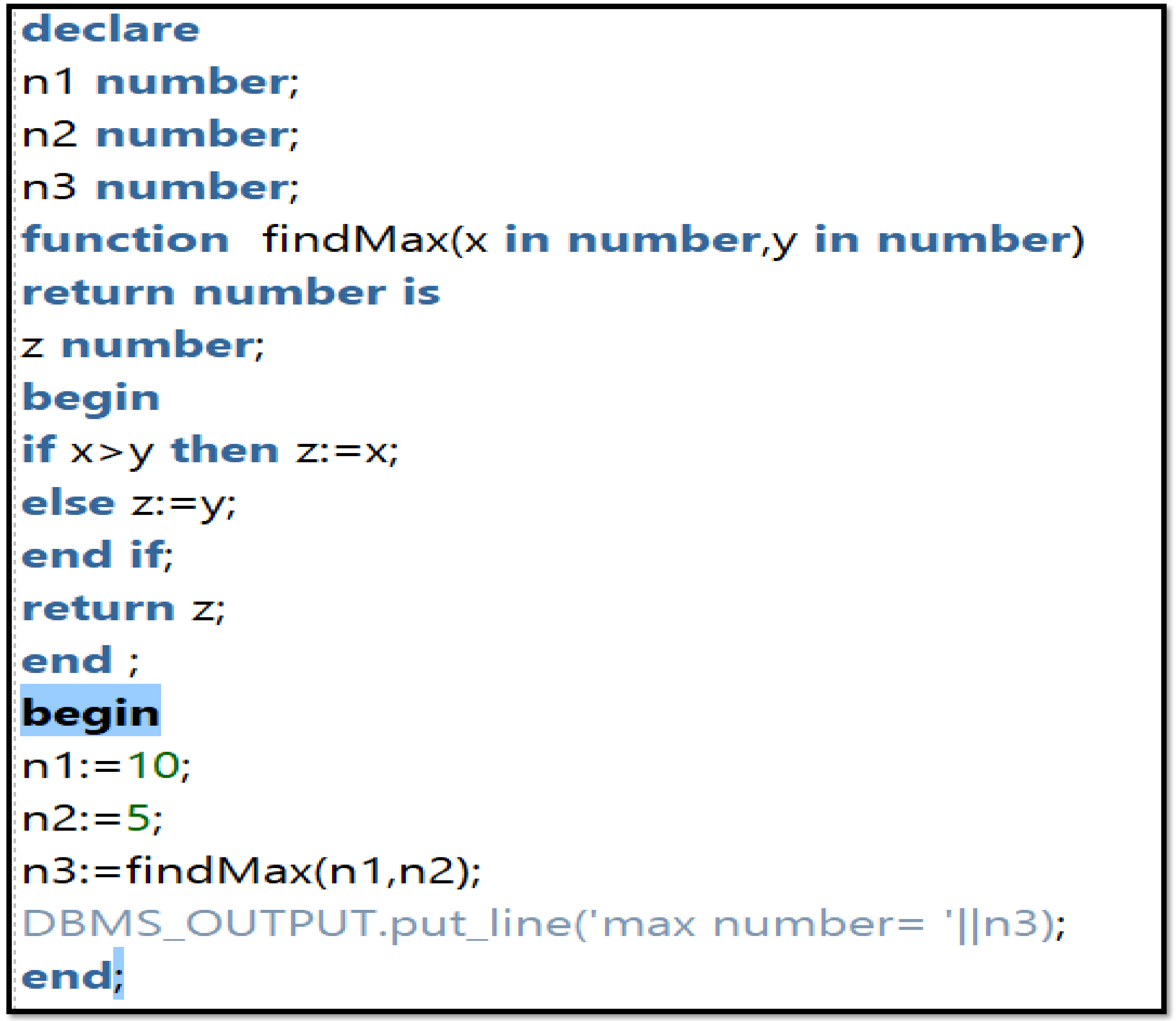
```
create or REPLACE  Function CountStudent
return number is
total_student number:=0;
begin
select count(*) into total_student
from student;
return total_student;
end;
```

Use CountStudent Function:



```
declare
Student_number number;
begin
Student_number:=CountStudent();
DBMS_OUTPUT.PUT_line('student Number ='||Student_number);
end;
```

Declare Function without Using **Create** :



```
declare
n1 number;
n2 number;
n3 number;
function  findMax(x in number,y in number)
return number is
z number;
begin
if x>y then z:=x;
else z:=y;
end if;
return z;
end ;
begin
n1:=10;
n2:=5;
n3:=findMax(n1,n2);
DBMS_OUTPUT.put_line('max number= '||n3);
end;
```

Overview of Packages

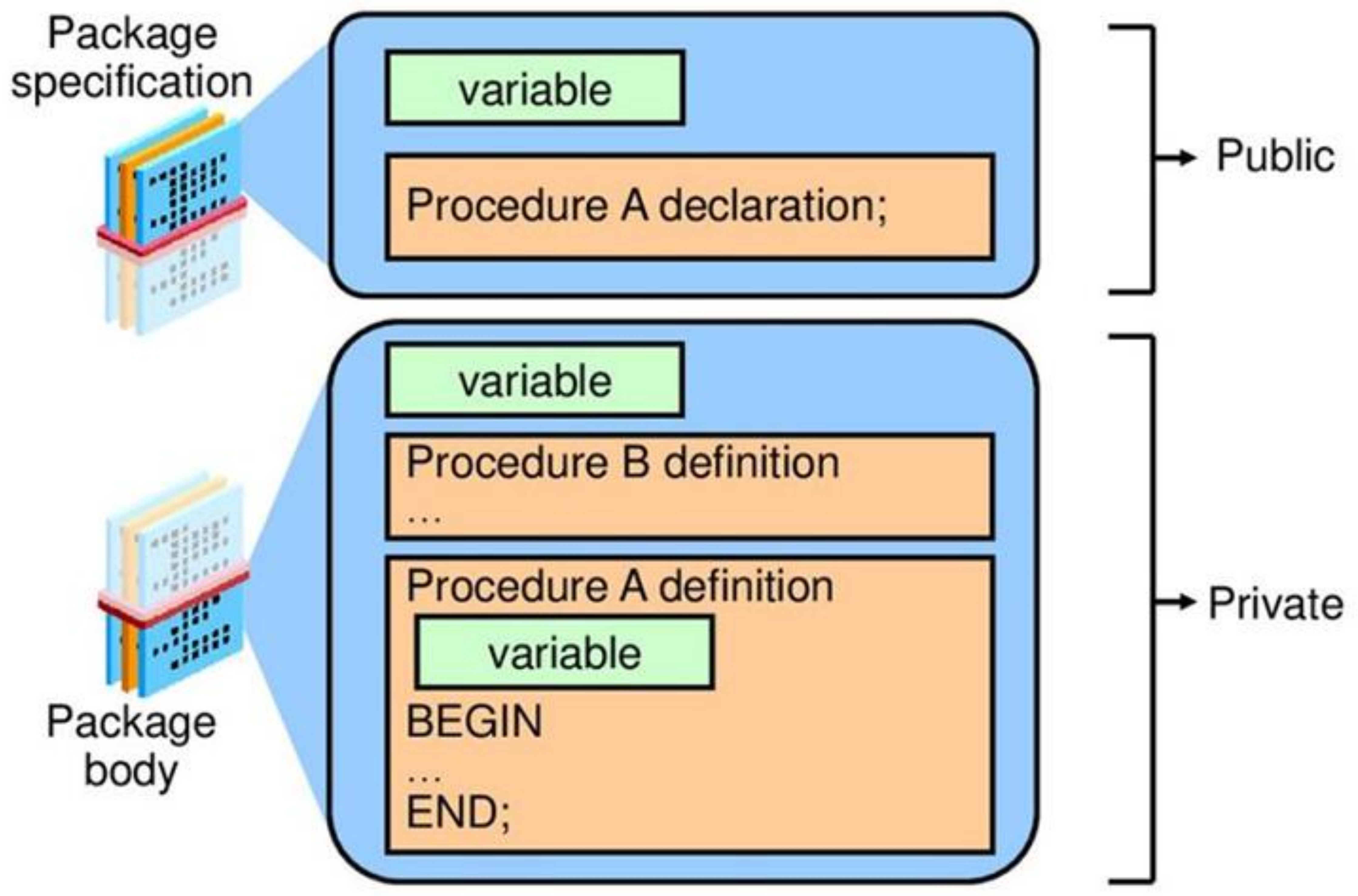
- **Packages** are objects that groups logically related variables, types, and subprograms.

- A package will have two parts:

  ✓Package specification.

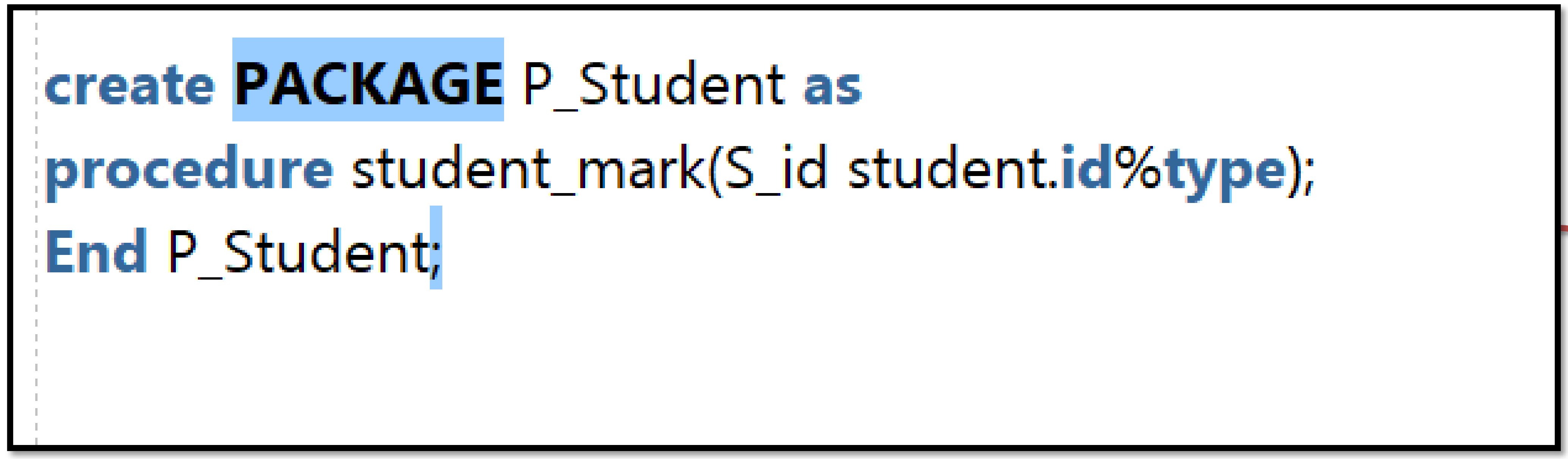
  ✓Package body or definition.

Components of a PL/SQL Package

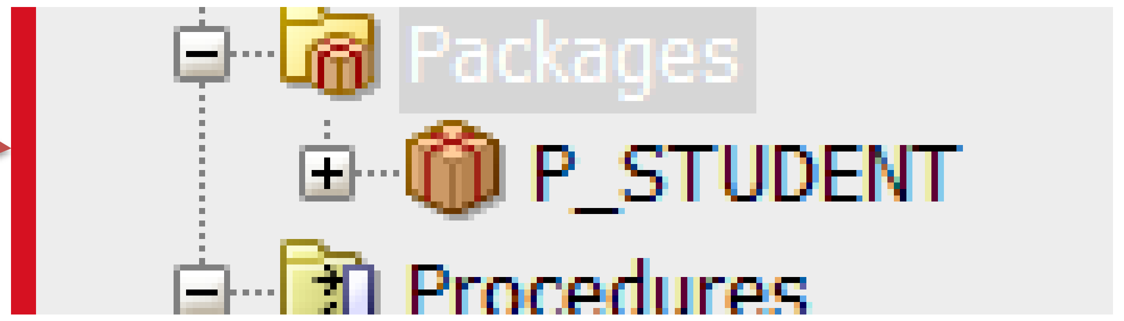


Create Packages

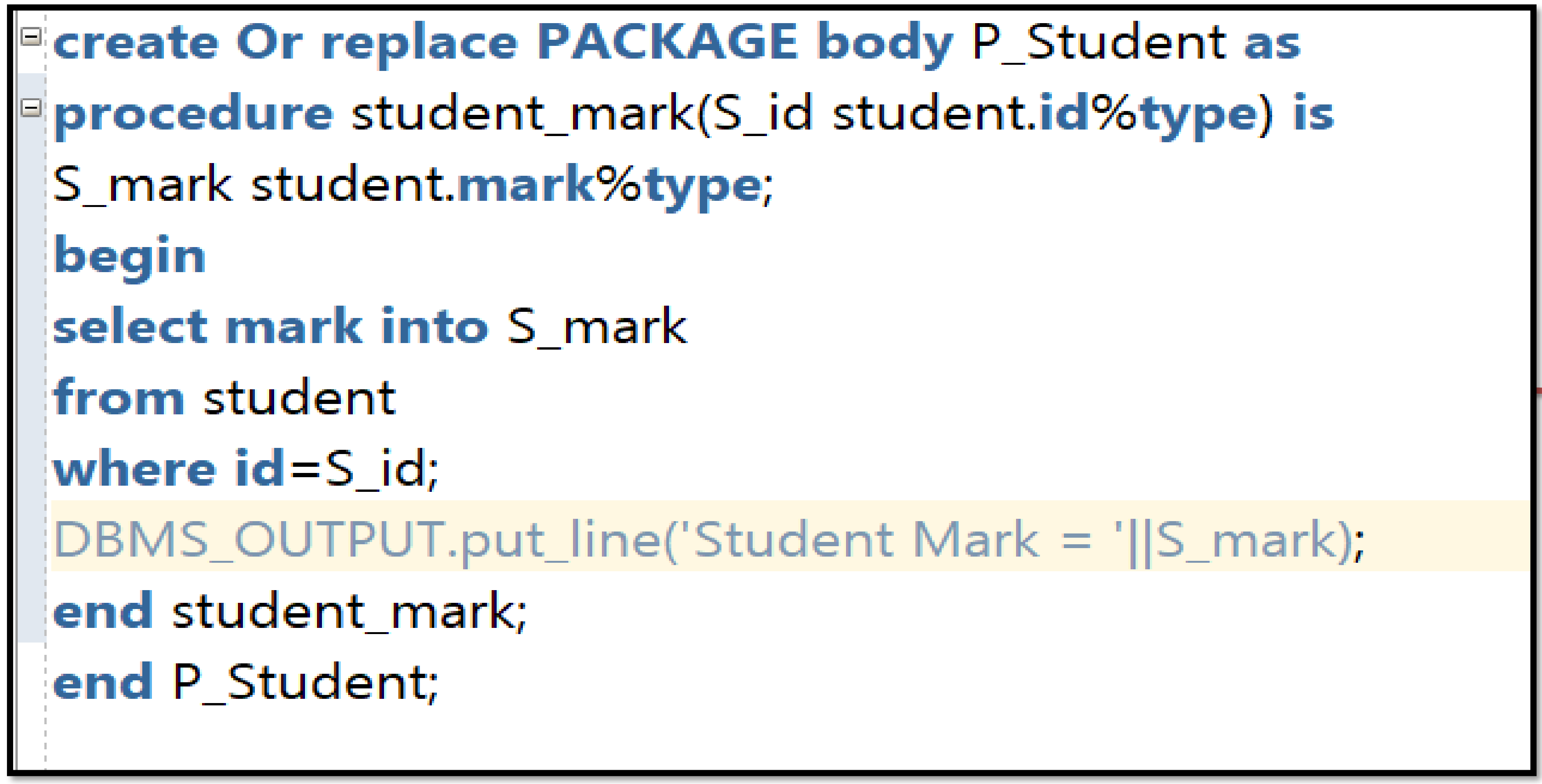
Create Package Header:



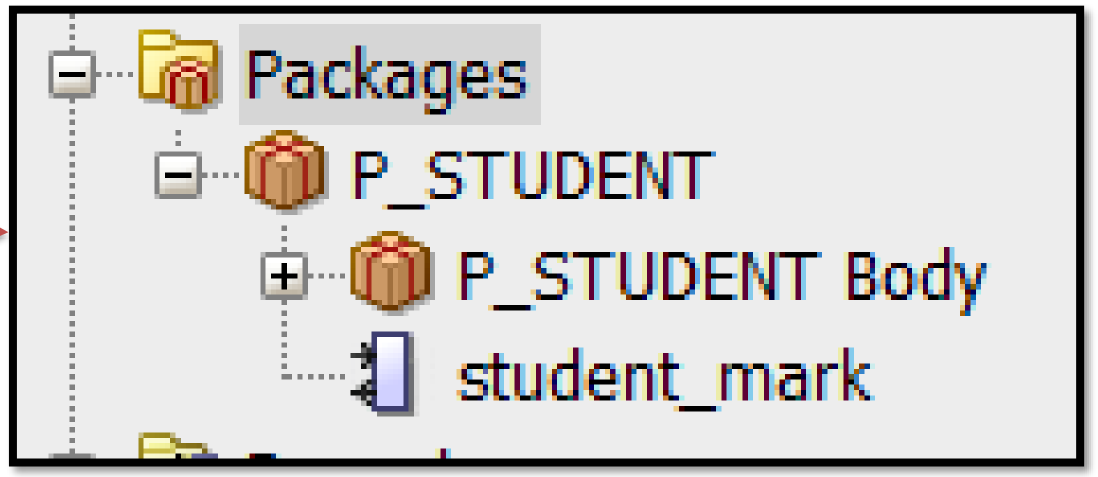
```
create PACKAGE P_Student as
procedure student_mark(S_id student.id%type);
End P_Student;
```



Create Package Body:

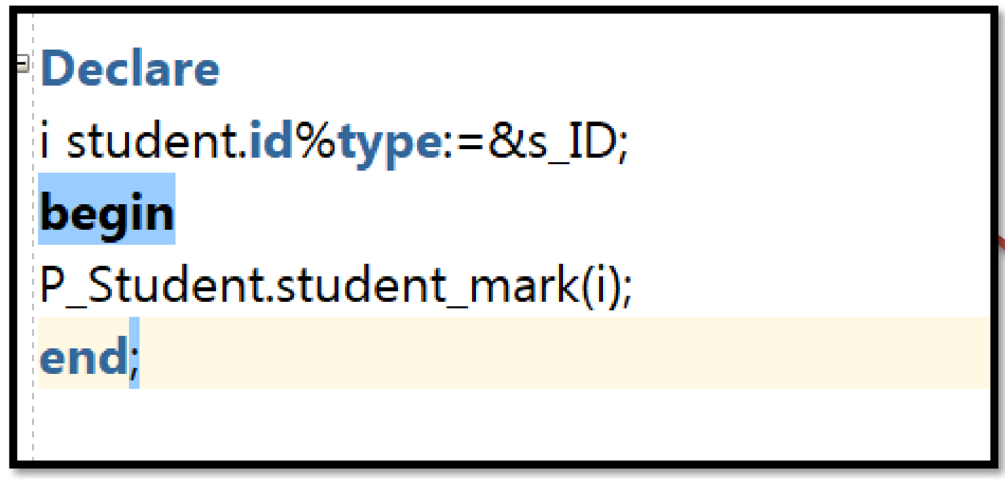


```
create Or replace PACKAGE body P_Student as
procedure student_mark(S_id student.id%type) is
S_mark student.mark%type;
begin
select mark into S_mark
from student
where id=S_id;
DBMS_OUTPUT.put_line('Student Mark = '||S_mark);
end student_mark;
end P_Student;
```

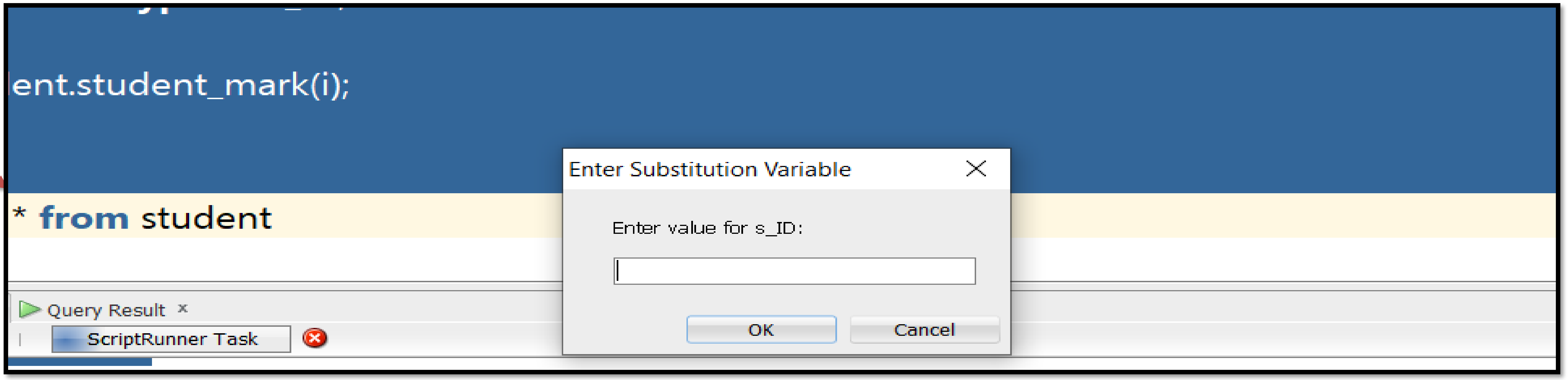


```
Packages
  P_STUDENT
    P_STUDENT Body
    student_mark
```

Execute Code :



```
Declare
i student.id%type:=&s_ID;
begin
P_Student.student_mark(i);
end;
```



```
ent.student_mark(i);

* from student
```

Enter Substitution Variable

Enter value for s_ID:

Overview of Triggers

Triggers are written to be executed in response when any of the following events occurs:

- A **database manipulation** (DML) commands (DELETE, INSERT, or UPDATE).
- A **database definition** (DDL) commands (CREATE, ALTER, or DROP).
- A **database operation** (LOGOFF, SERVERERROR, STARTUP, or SHUTDOWN).

Benefits of Triggers :

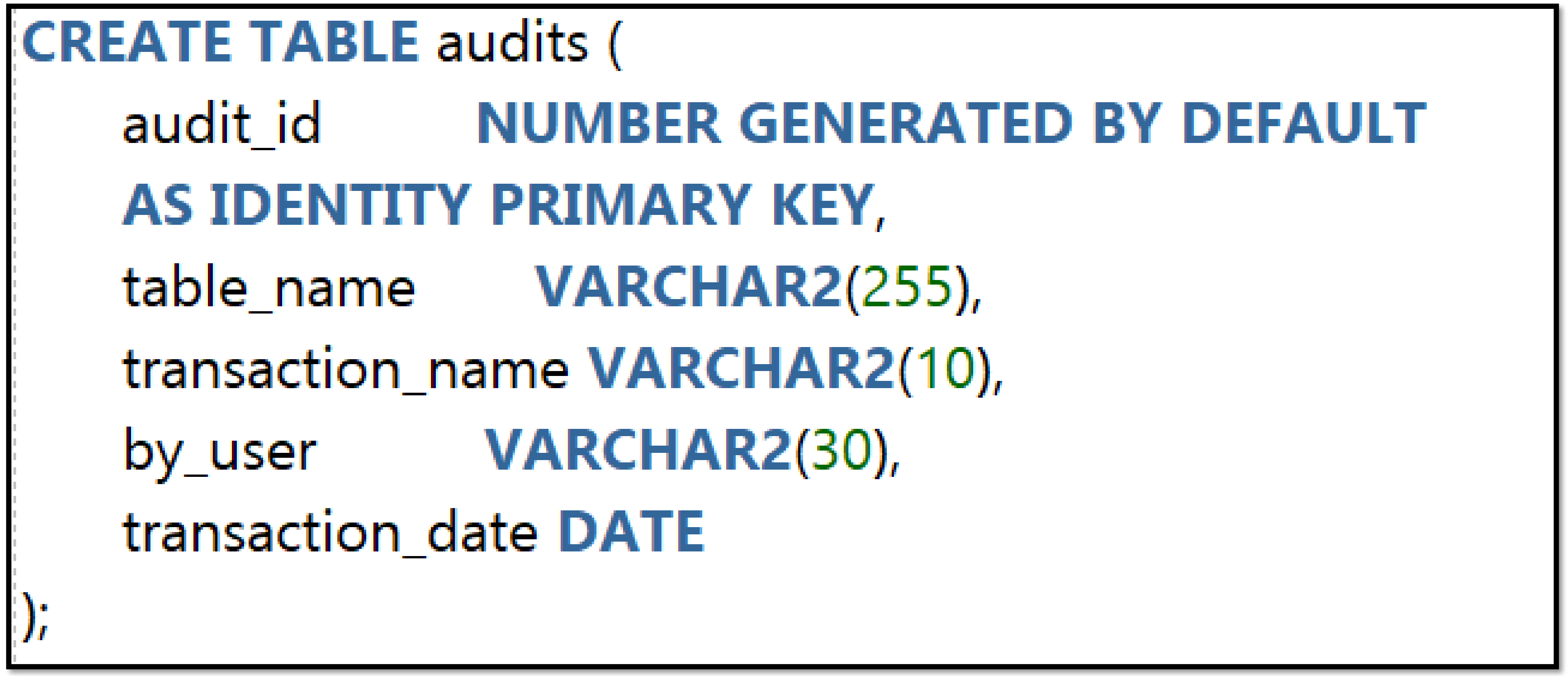
1. Generate a some of derived column values automatically.
2. Event storing and logging information on table access.
3. Synchronous replication of tables.
4. Imposing security authorizations.
5. Preventing invalid transactions.
6. Enforcing referential integrity.
7. Auditing.

Syntax :

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
Declaration-statements
BEGIN
Executable-statements
EXCEPTION
Exception-handling-statements
END;
```

Create Triggers

1- Create Table Audits :



```
CREATE TABLE audits (
    audit_id       NUMBER GENERATED BY DEFAULT
    AS IDENTITY PRIMARY KEY,
    table_name       VARCHAR2(255),
    transaction_name VARCHAR2(10),
    by_user        VARCHAR2(30),
    transaction_date DATE
);
```

2- Create Triggers :

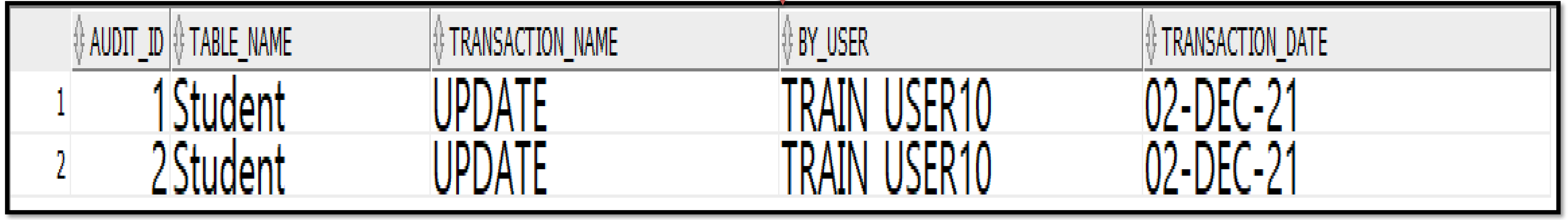
```
CREATE OR REPLACE TRIGGER student_audit
    AFTER
    UPDATE OR DELETE
    ON student
    FOR EACH ROW
    DECLARE
    l_transaction VARCHAR2(10);
    BEGIN
        l_transaction := CASE
        WHEN UPDATING THEN 'UPDATE'
        WHEN DELETING THEN 'DELETE'
   END;
INSERT INTO audits (table_name, transaction_name, by_user, transaction_date)
VALUES('Student', l_transaction, USER, SYSDATE);
END student_audit;
```

2- Example 1: Execute **Update** Command :

```
UPDATE
  student
SET
   name= 'Mutaz'
WHERE
   id =1;

SELECT * FROM audits;
```

Select * from Audits

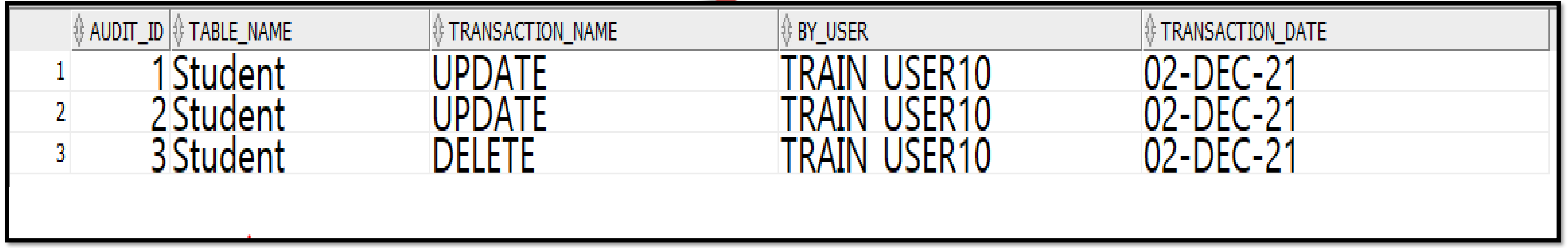


	AUDIT_ID	TABLE_NAME	TRANSACTION_NAME	BY_USER	TRANSACTION_DATE
1	1	Student	UPDATE	TRAIN USER10	02-DEC-21
2	2	Student	UPDATE	TRAIN USER10	02-DEC-21

3- Example 2: Execute **Delete** Command :

```sql
DELETE FROM student
WHERE id = 4;
```

Select * from Audits



	AUDIT_ID	TABLE_NAME	TRANSACTION_NAME	BY_USER	TRANSACTION_DATE
1	1	Student	UPDATE	TRAIN_USER10	02-DEC-21
2	2	Student	UPDATE	TRAIN_USER10	02-DEC-21
3	3	Student	DELETE	TRAIN_USER10	02-DEC-21

References :

[1]. PL/SQL Procedure: A Step-by-step Guide to Create a Procedure (oracletutorial.com)
[2]. Oracle INSTEAD OF Triggers By Practical Examples (oracletutorial.com)

Harmony IT Solution