Nama	: Afina Putri Dayanti
NIM	: 825200049
Jurusan	: Sistem Informasi
Mata Kuliah	: Database Design and Management (Praktikum)

Vocabulary

Identify the vocabulary word for each definition below:

Subprograms	Named PL/SQL blocks that are compiled and stored in the database.
Anonymous Blocks	Unnamed executable PL/SQL blocks that cannot be reused or stored in the database for later use.
Procedure	Named PL/SQL blocks that can accept parameters and are compiled and stored in the database.

Try It / Solve It

1. What is the difference between the following two pieces of code?

```
--CODE SAMPLE A
DECLARE
  v_empid            employees.employee_id%TYPE := 100;
  v_percent_increase NUMBER(2,2) := .05;
BEGIN
  UPDATE employees
  SET salary = (salary * v_percent_increase) + salary
  WHERE employee_id = v_empid;
END;

--CODE SAMPLE B
CREATE PROCEDURE pay_raise(
  p_empid            employees.employee_id%TYPE,
  p_percent_increase NUMBER) IS
BEGIN
  UPDATE employees
  SET salary = (salary * p_percent_increase) + salary
  WHERE employee_id = p_empid;
END pay_raise;
```

   Answer : Code sample A is an anonymous blocks, while code sample B is a subprograms (procedures)

2. In your own words, list the benefits of subprograms.
   Answer : Subprograms are easy to maintain, can be re-used, improve data security and data integrity, improve performance, and improve readability of code

3. In your own words, describe a stored procedure.
   Answer : A procedure is a named subprogram that is compiled and stored in the database as an object in a schema. It can accept parameters and can return values. A procedure performs an action and can be re-used.

4. The remaining questions in this practice use a copy of the employees table.

a. Create the copy by executing the following SQL statement:
   CREATE TABLE employees_dup AS SELECT * from employees;
b. Create the following procedure in Application Express:

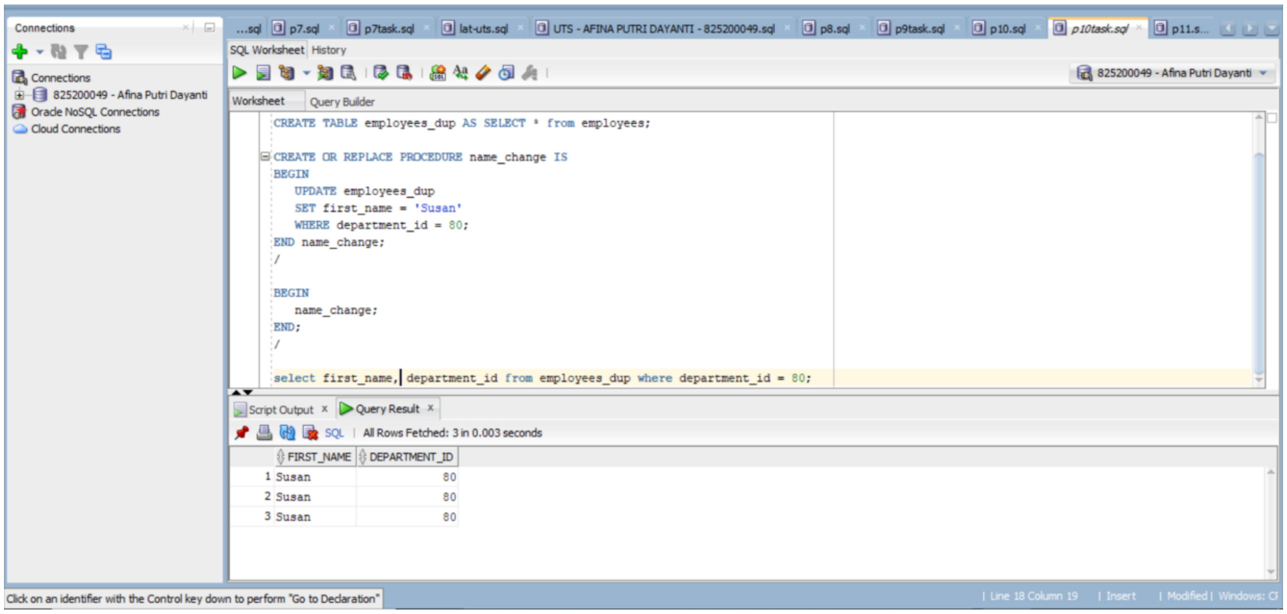
```
CREATE OR REPLACE PROCEDURE name_change IS
BEGIN
  UPDATE employees_dup
  SET first_name = 'Susan'
  WHERE department_id = 80;
END name_change;
```

c. Save the definition of your procedure in case you need to modify it later. In the “Save SQL” popup, name your saved work “My name change procedure”.
d. Execute the procedure by running the following anonymous block:

```
BEGIN
  name_change;
END;
```

e. SELECT from the table to check that the procedure has executed correctly and performed the UPDATE:
   Answer :

```
SELECT first_name, department_id
FROM employees_dup
WHERE department_id = 80;
```



```
CREATE TABLE employees_dup AS SELECT * from employees;

CREATE OR REPLACE PROCEDURE name_change IS
BEGIN
   UPDATE employees_dup
   SET first_name = 'Susan'
   WHERE department_id = 80;
END name_change;
/

BEGIN
   name_change;
END;
/

select first_name, department_id from employees_dup where department_id = 80;
```

All Rows Fetched: 3 in 0.003 seconds

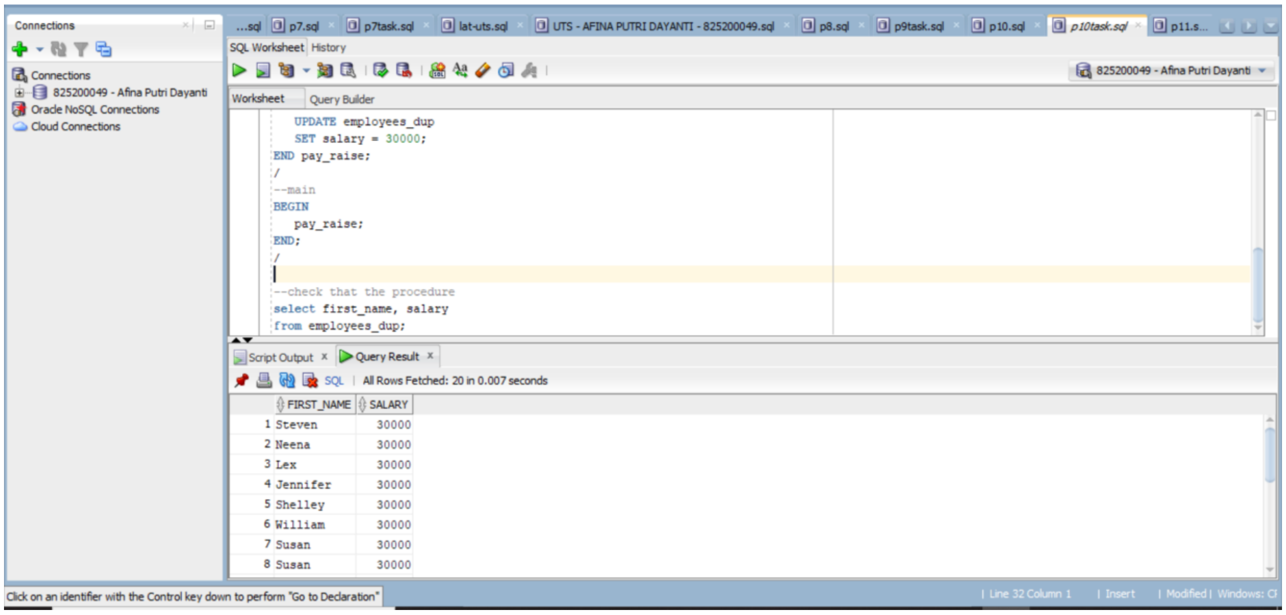
```
  FIRST_NAME  DEPARTMENT_ID
1 Susan                  80
2 Susan                  80
3 Susan                  80
```

Click on an identifier with the Control key down to perform "Go to Declaration"

5. Create a second procedure named pay_raise which changes the salary of all employees in employees_dup to a new value of 30000. Execute the procedure from an anonymous block, then SELECT from the table to check that the procedure has executed correctly.
   Answer :

```
--procedure
CREATE OR REPLACE PROCEDURE pay_raise IS
BEGIN
  UPDATE employees_dup
  SET salary = 30000;
END pay_raise;
```

```
/

--main
BEGIN
  pay_raise;
END;
/

--check that the procedure
SELECT first_name, salary
FROM employees_dup;
```



```
   UPDATE employees_dup
   SET salary = 30000;
END pay_raise;
/
--main
BEGIN
   pay_raise;
END;
/

--check that the procedure
select first_name, salary
from employees_dup;
```

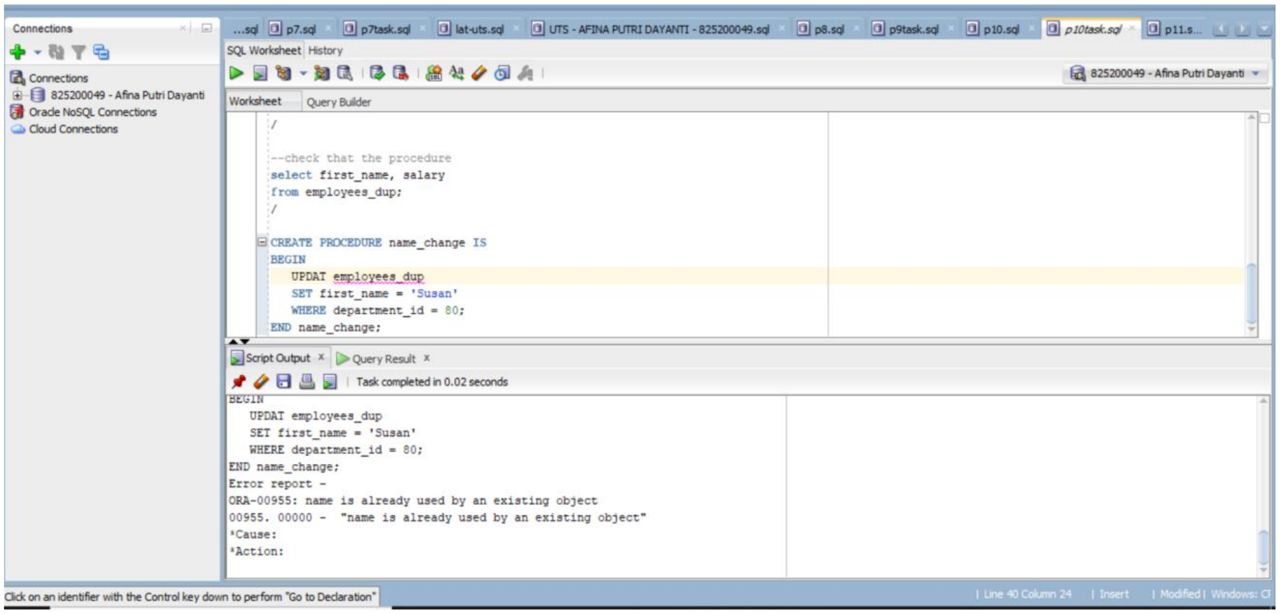
```
All Rows Fetched: 20 in 0.007 seconds

   FIRST_NAME  SALARY
 1 Steven       30000
 2 Neena        30000
 3 Lex          30000
 4 Jennifer     30000
 5 Shelley      30000
 6 William      30000
 7 Susan        30000
 8 Susan        30000
```

Click on an identifier with the Control key down to perform "Go to Declaration" | Line 32 Column 1 | Insert | Modified | Windows: CF

6. Retrieve your first name_change procedure by clicking on its name in the Saved SQL window. Modify the code to remove OR REPLACE from the CREATE statement, and introduce a deliberate error into the code, for example by misspelling a keyword: UPDAT employees_dup. Execute your code to recreate the procedure. What happens?

Answer :

```
CREATE PROCEDURE name_change IS
BEGIN
  UPDAT employees_dup
  SET first_name = 'Susan'
  WHERE department_id = 80;
END name_change;

/* error : ORA-00955: name is already used by an existing object
```

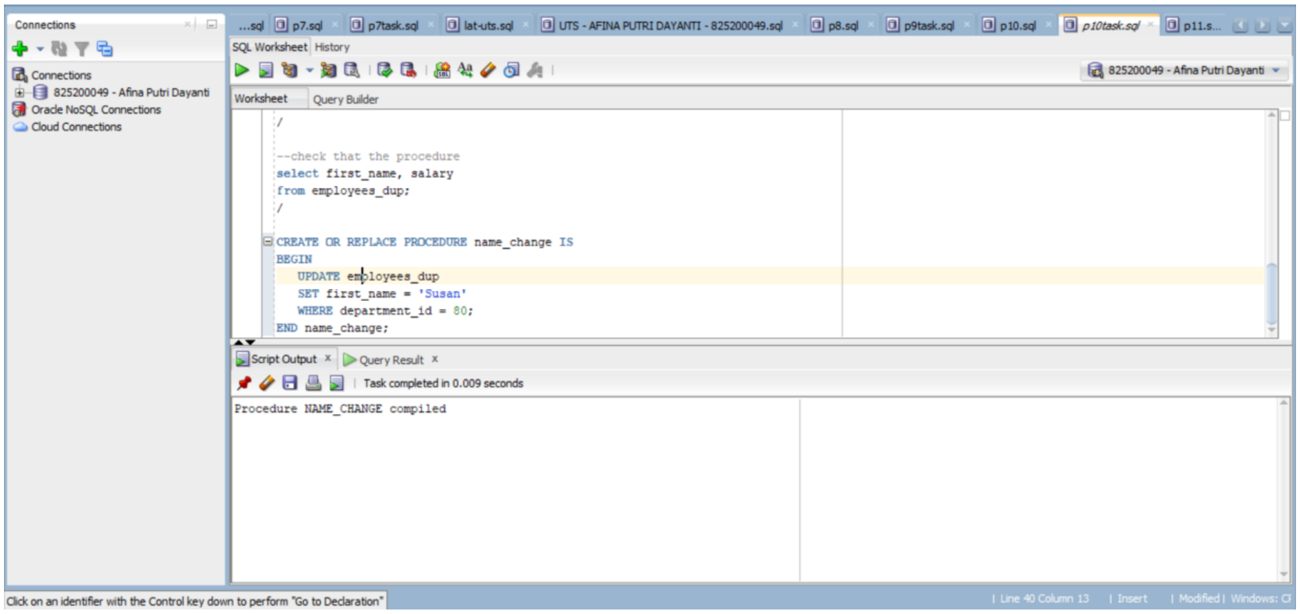


```
/

--check that the procedure
select first_name, salary
from employees_dup;
/

CREATE PROCEDURE name_change IS
BEGIN
   UPDAT employees_dup
   SET first_name = 'Susan'
   WHERE department_id = 80;
END name_change;
```

```
BEGIN
   UPDAT employees_dup
   SET first_name = 'Susan'
   WHERE department_id = 80;
END name_change;
Error report -
ORA-00955: name is already used by an existing object
00955. 00000 -  "name is already used by an existing object"
*Cause:
*Action:
```

7. Now correct the procedure code by reinserting the OR REPLACE clause and correcting your deliberate spelling error. Execute your code to recreate the procedure. Now what happens?
Answer :

```
CREATE OR REPLACE PROCEDURE name_change IS
BEGIN
  UPDATE employees_dup
  SET first_name = 'Susan'
  WHERE department_id = 80;
END name_change;

/* Procedure NAME_CHANGE compiled
```



```
/

--check that the procedure
select first_name, salary
from employees_dup;
/

CREATE OR REPLACE PROCEDURE name_change IS
BEGIN
   UPDATE employees_dup
   SET first_name = 'Susan'
   WHERE department_id = 80;
END name_change;
```

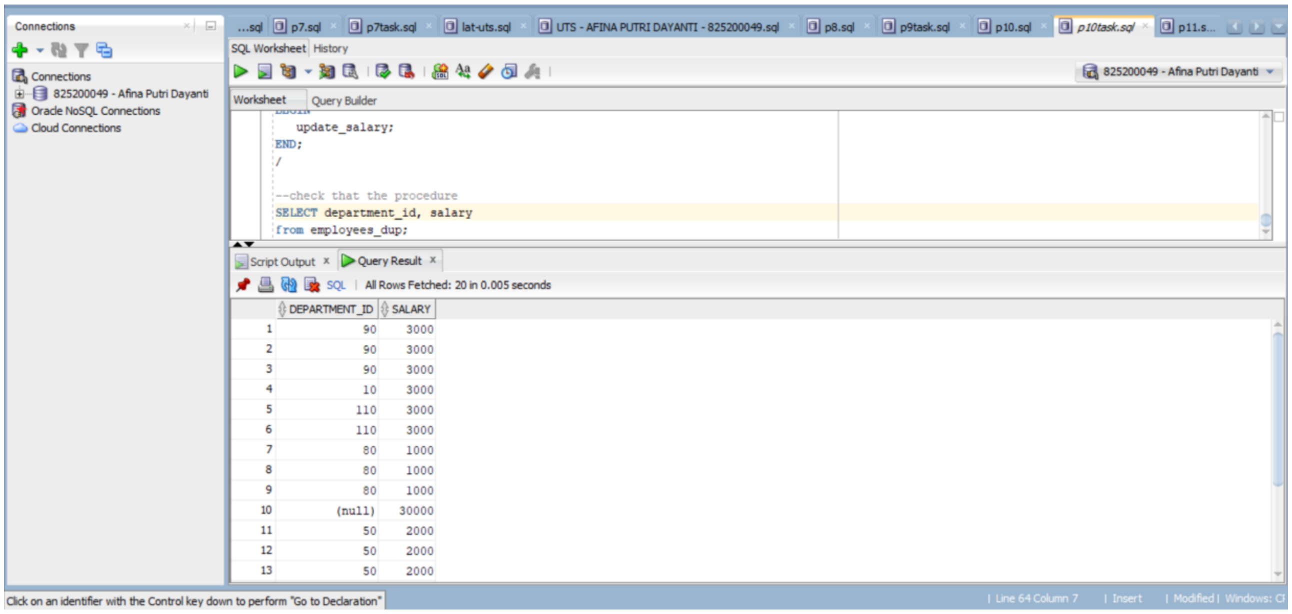
```
Procedure NAME_CHANGE compiled
```

Extension Exercise

1. Create, save, and execute a procedure which updates the salary of employees in employees_dup according to the following rules:
   - if the employee is in department 80, the new salary must = 1000
   - if the employee is in department 50, the new salary must = 2000
   - if the employee is in any other department, the new salary must = 3000.

   You will need to include three UPDATE statements, one for each of the above rules. In a later lesson you will learn how to avoid this.
   Execute your procedure from an anonymous block and verify that the updates have been performed correctly.
   Answer :

```
--procedure
CREATE OR REPLACE PROCEDURE update_salary IS
BEGIN
  UPDATE employees_dup
  SET salary = 1000 WHERE department_id = 80;
  UPDATE employees_dup
  SET salary = 2000 WHERE department_id = 50;
  UPDATE employees_dup
  SET salary = 3000 WHERE department_id NOT IN(80, 50);
END update_salary;
/

--main
BEGIN
  update_salary;
END;
/

--check that the procedure
SELECT department_id, salary
FROM employees_dup;
```



```
  update_salary;
END;
/

--check that the procedure
SELECT department_id, salary
from employees_dup;
```

All Rows Fetched: 20 in 0.005 seconds

```
     DEPARTMENT_ID  SALARY
 1              90    3000
 2              90    3000
 3              90    3000
 4              10    3000
 5             110    3000
 6             110    3000
 7              80    1000
 8              80    1000
 9              80    1000
10          (null)   30000
11              50    2000
12              50    2000
13              50    2000
```