

เราใช้ PL/SQL ทำอะไรกับระบบฐานข้อมูล Oracle

About PL/SQL

PL/SQL:

- Stands for “Procedural Language extension to SQL”
- Is Oracle Corporation’s standard data access language for relational databases
- Seamlessly integrates procedural constructs with SQL

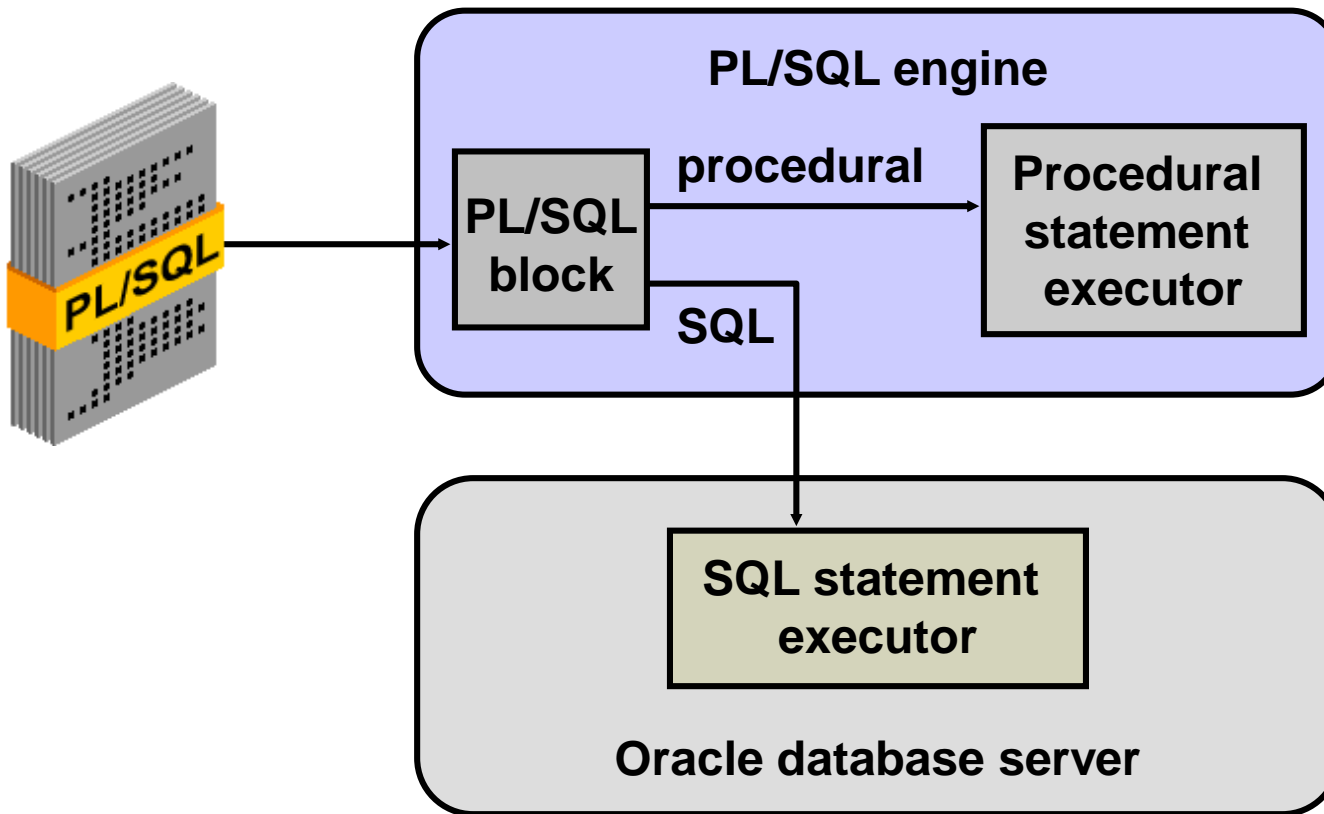


About PL/SQL

PL/SQL:

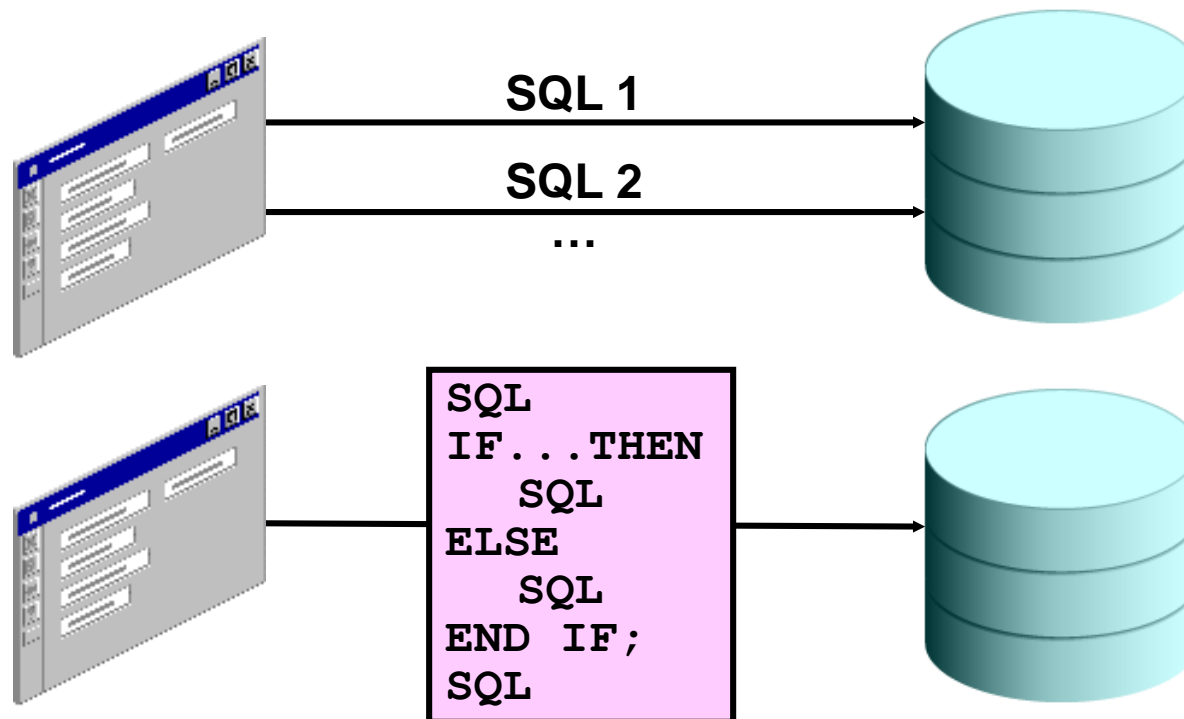
- Provides a block structure for executable units of code. Maintenance of code is made easier with such a well-defined structure.
- Provides procedural constructs such as:
 - Variables, constants, and data types
 - Control structures such as conditional statements and loops
 - Reusable program units that are written once and executed many times

PL/SQL Environment



Benefits of PL/SQL

- Integration of procedural constructs with SQL
- Improved performance



PL/SQL Block Structure

- DECLARE (optional)
 - Variables, cursors, user-defined exceptions
- BEGIN (mandatory)
 - SQL statements
 - PL/SQL statements
- EXCEPTION (optional)
 - Actions to perform when **errors** occur
- END; (mandatory)



Block Types

Anonymous

Procedure

Name block

Function

```
[DECLARE]

BEGIN
    --statements

[EXCEPTION]

END;
```

```
PROCEDURE name
IS

BEGIN
    --statements

[EXCEPTION]

END;
```

```
FUNCTION name
RETURN datatype
IS

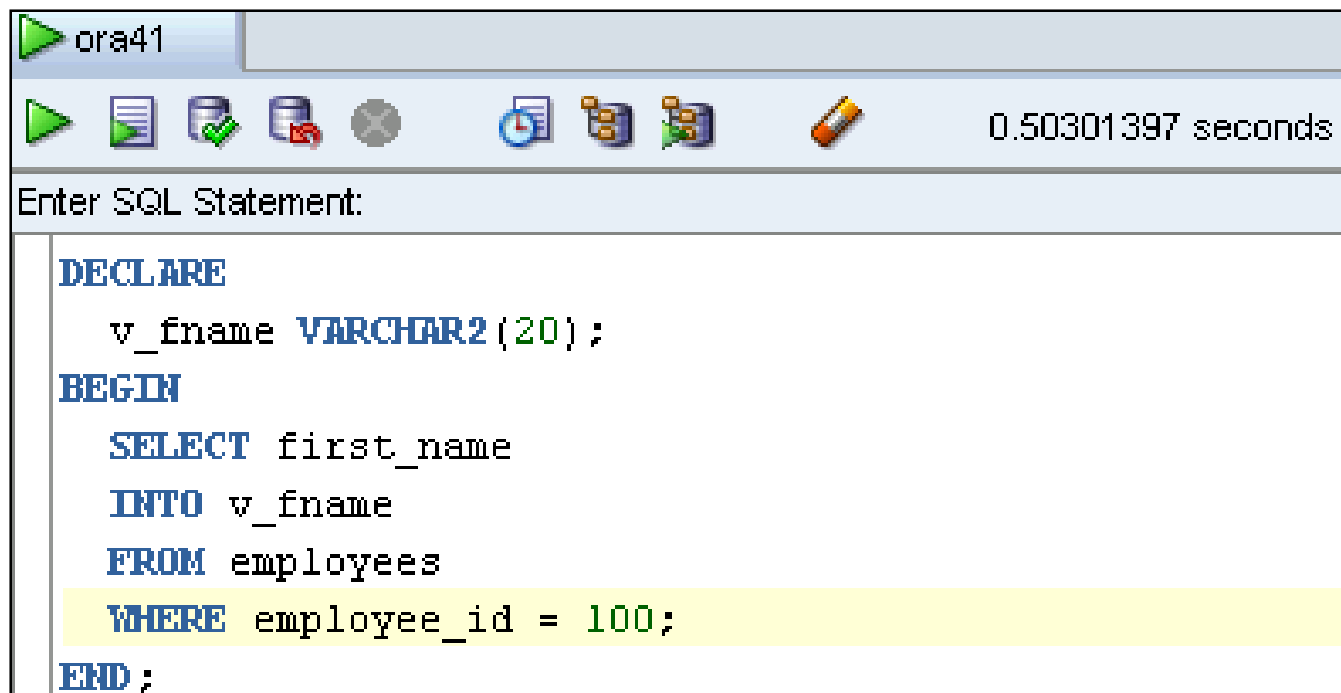
BEGIN
    --statements
    RETURN value;

[EXCEPTION]

END;
```

Create an Anonymous Block

Enter the anonymous block in the SQL Developer workspace:



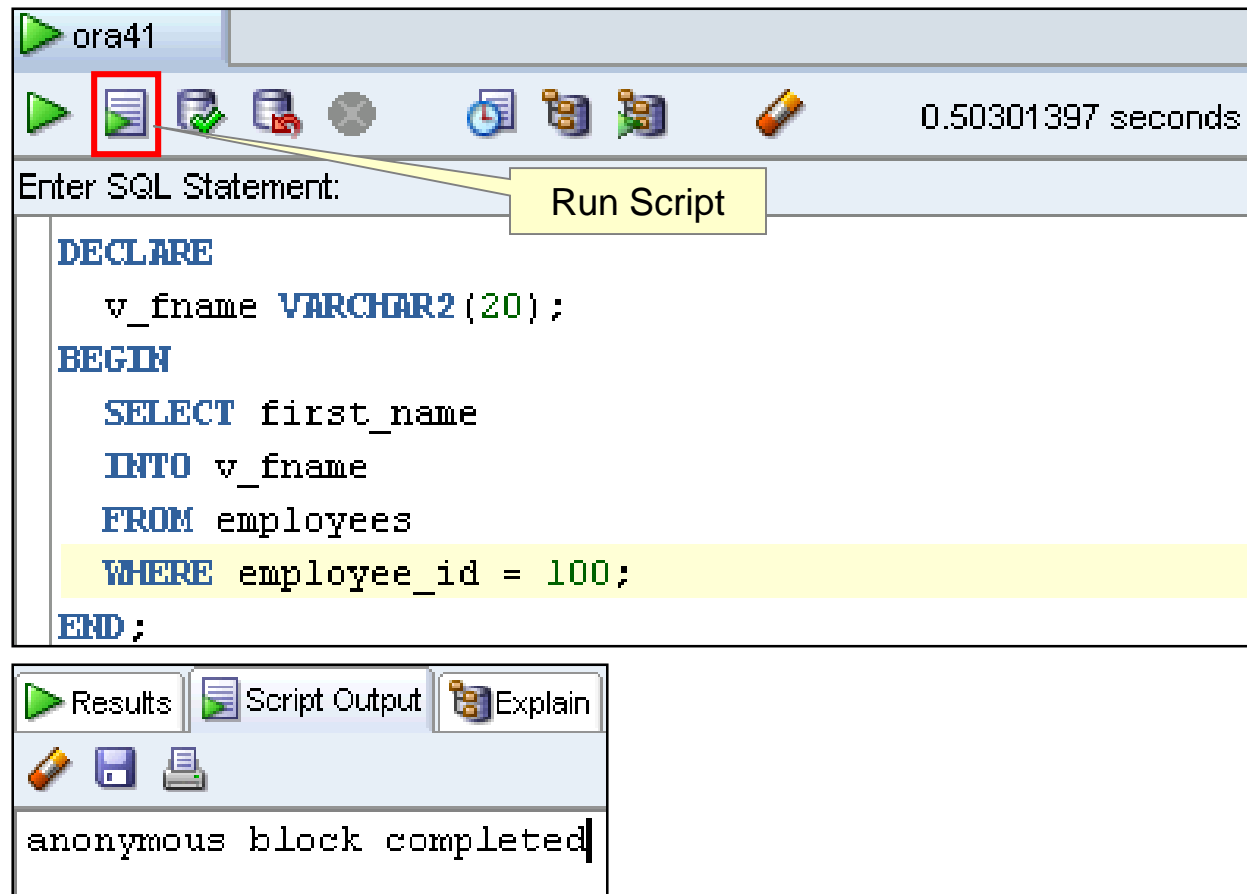
The screenshot shows the SQL Developer interface. At the top, a tab labeled 'ora41' is active. Below the tab is a toolbar with various icons for file operations, execution, and debugging. To the right of the toolbar, the execution time '0.50301397 seconds' is displayed. The main area is titled 'Enter SQL Statement:' and contains the following SQL code:

```
DECLARE
  v_fname VARCHAR2(20);
BEGIN
  SELECT first_name
  INTO v_fname
  FROM employees
  WHERE employee_id = 100;
END;
```

The 'WHERE' clause and its value are highlighted in yellow.

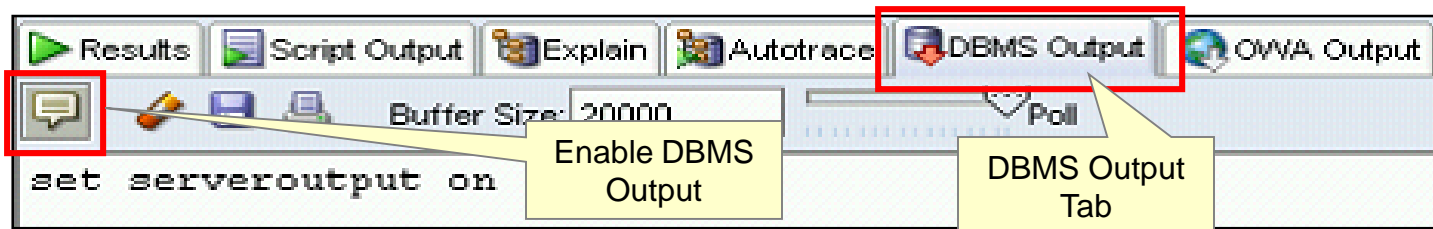
Execute an Anonymous Block

Click the Run Script button to execute the anonymous block:



Test the Output of a PL/SQL Block

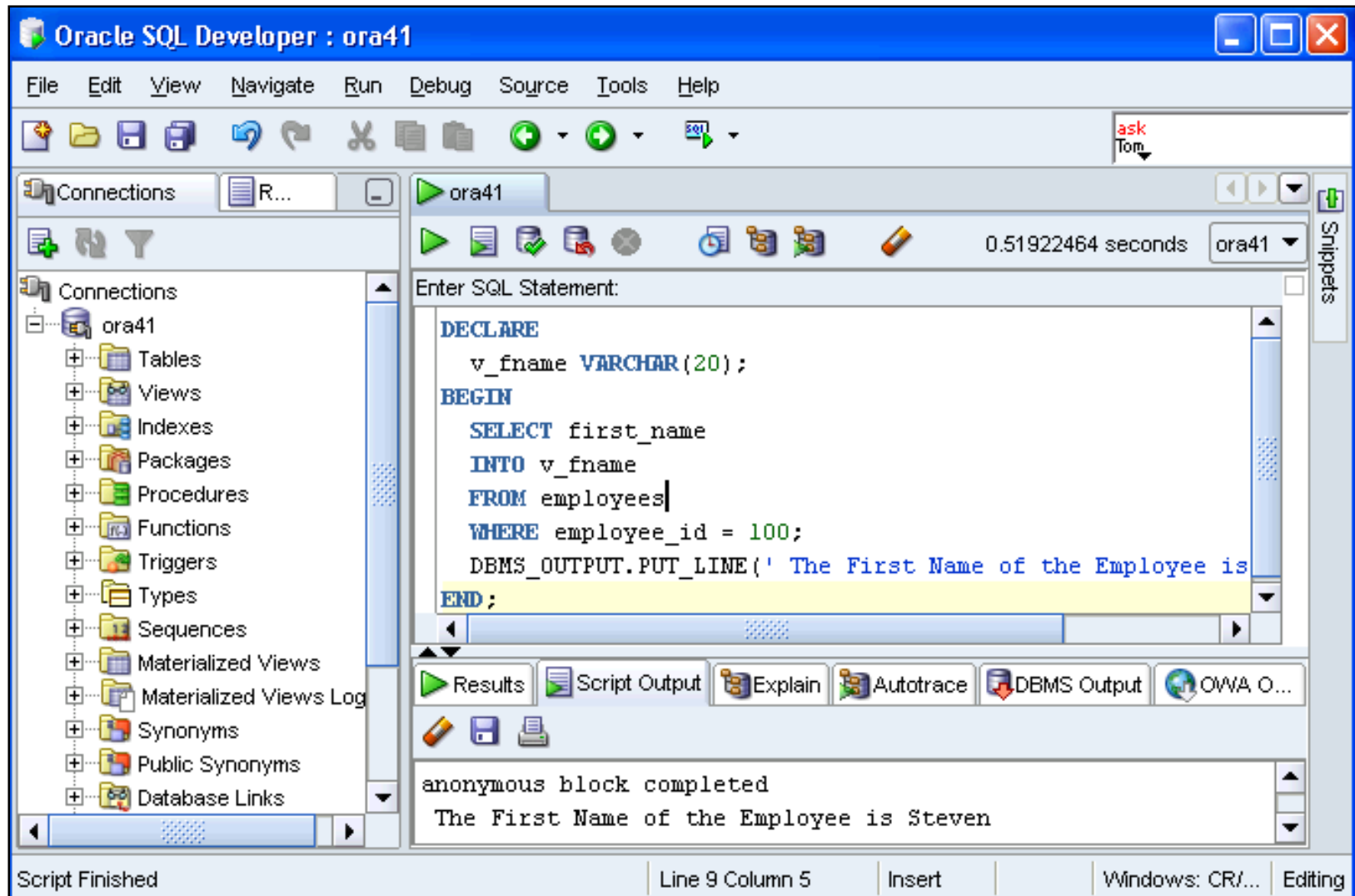
- Enable output in SQL Developer by clicking the Enable DBMS Output button on the DBMS Output tab:



- Use a predefined Oracle package and its procedure:
 - `DBMS_OUTPUT.PUT_LINE`

```
DBMS_OUTPUT.PUT_LINE(' The First Name of the  
Employee is ' || v_fname);  
...
```

Test the Output of a PL/SQL Block



โครงสร้างและตัวแปรของโปรแกรม PL/SQL

PL/SQL Block Structure

- DECLARE (optional)
 - Variables, cursors, user-defined exceptions
- BEGIN (mandatory)
 - SQL statements
 - PL/SQL statements
- EXCEPTION (optional)
 - Actions to perform when errors occur
- END; (mandatory)



Commenting Code

- Prefix single-line comments with two hyphens (--).
- Place multiple-line comments between the symbols /* and */.

Example:

```
DECLARE
...
v_annual_sal NUMBER (9,2);
BEGIN
/* Compute the annual salary based on the
   monthly salary input from the user */
v_annual_sal := monthly_sal * 12;
--The following line displays the annual salary
DBMS_OUTPUT.PUT_LINE(v_annual_sal);
END;
/
```

Operators in PL/SQL

- Logical
- Arithmetic
- Concatenation (||)
- Parentheses to control order of operations
- Exponential operator (**)
 ยกกำลัง



Same as in SQL

Operators in PL/SQL: Examples

- Increment the counter for a loop.

```
loop_count := loop_count + 1;
```

- Set the value of a Boolean flag.

```
good_sal := sal BETWEEN 50000 AND 150000;
```

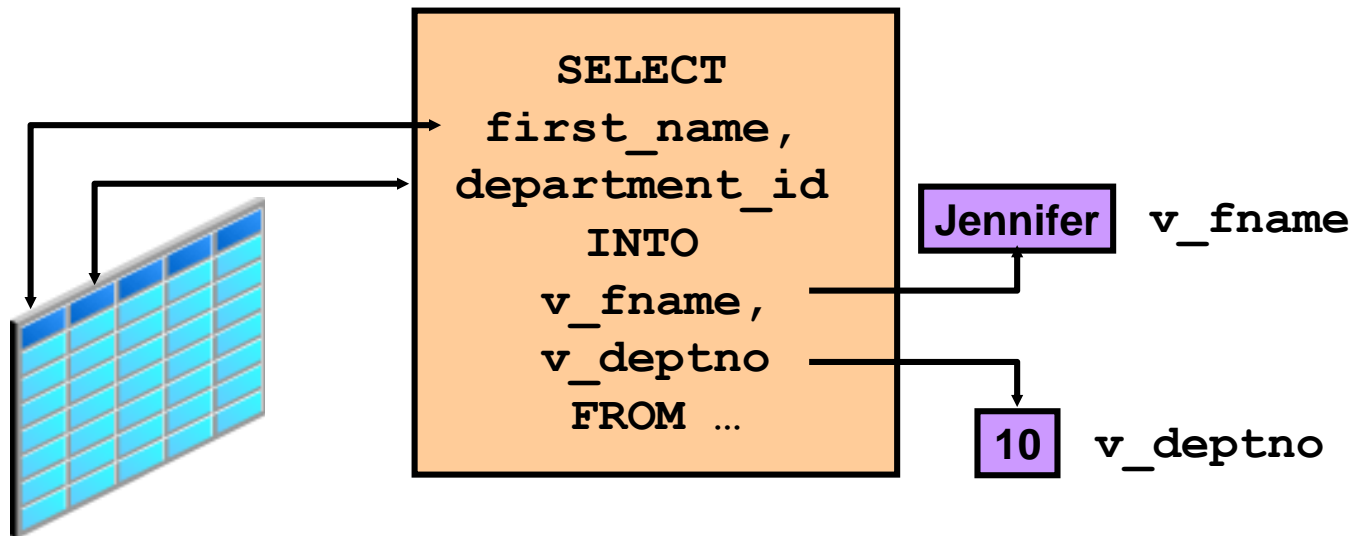
- Validate whether an employee number contains a value.

```
valid := (empno IS NOT NULL);
```


Use of Variables

Variables can be used for:

- Temporary storage of data
- Manipulation of stored values
- Reusability



Requirements for Variable Names

A variable name:

- Must start with a letter
- Can include letters or numbers
- Can include special characters (**such as \$, _, and #**)
- Must contain no more than 30 characters
- Must not include **reserved words** (เช่น คำว่า name)



Handling Variables in PL/SQL

Variables are:

- Declared and initialized in the declarative section
- Used and assigned new values in the executable section
- Passed as parameters to PL/SQL subprograms
- Used to hold the output of a PL/SQL subprogram

Scalar Data Types

- Hold a single value
- Have no internal components

TRUE

25-JAN-01

**The soul of the lazy man
desires, and he has nothing;
but the soul of the diligent
shall be made rich.**

256120.08

Atlanta

Declaring and Initializing PL/SQL Variables

Syntax:

`:=` มีความหมายเท่ากับคำว่า **DEFAULT**

```
identifier [CONSTANT] datatype [NOT NULL]  
[:= | DEFAULT expr];
```


Examples:

```
v_hiredate DATE;  
  
v_deptno    NUMBER(2) NOT NULL := 10;  
  
v_location  VARCHAR2(13) := 'Atlanta';  
  
c_comm      CONSTANT NUMBER := 1400;
```

Guidelines for Declaring PL/SQL Variables

- Avoid using column names as identifiers.

```
DECLARE
  employee_id NUMBER(6);
BEGIN
  SELECT  employee_id
  INTO    employee_id
  FROM    employees
  WHERE   last_name = 'Kochhar';
END;
/
```



อย่าตั้งชื่อให้ซ้ำกับที่ดึงมาจาก **table**

- Use the NOT NULL constraint when the variable must hold a value.

Declaring Scalar Variables

Examples:

- DECLARE
- v_emp_job VARCHAR2 (9) ;
- v_count_loop BINARY_INTEGER := 0; (จำนวนไบต์ที่เก็บน้อย)
- v_dept_sal NUMBER (9,2) := 0;
- v_orderdate DATE := SYSDATE + 7;
- c_tax_rate CONSTANT NUMBER (3,2) := 8.25;
- v_valid BOOLEAN NOT NULL := TRUE; ___(ตัวใหญ่เล็กได้)
- ...

Example:

```
• DECLARE
•     a NUMBER := 10;
•     b NUMBER := 20;
•     c NUMBER;
•     f NUMBER(5,2);
• BEGIN
•     c := a + b;
•     DBMS_OUTPUT.PUT_LINE('Value of c: ' || c);
•     f := 70.0/3.0;
•     DBMS_OUTPUT.PUT_LINE('Value of f: ' || f);
• END;
• /
```

```
Value of c: 30
Value of f: 23.333333333333333333
```


Guidelines for Declaring and Initializing PL/SQL Variables

- Follow naming conventions.
- Use meaningful identifiers for variables.
- Initialize variables designated as `NOT NULL` and `CONSTANT`.
- Initialize variables with the assignment operator `(:=)` or the `DEFAULT` keyword:

```
v_myName VARCHAR2(20) := 'John' ;
```

```
v_myName VARCHAR2(20) DEFAULT 'John' ;
```

- Declare one identifier per line for better readability and code maintenance.

Declaring and Initializing PL/SQL Variables

1

```
• DECLARE
•   v_myName VARCHAR2(20);
• BEGIN
•   DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
•   v_myName := 'John';
•   DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
• END;
• /
```

2

```
• DECLARE
•   v_myName VARCHAR2(20) := 'John';
• BEGIN
•   v_myName := 'Steven';
•   DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
• END;
• /
```

PL/SQL — CONSTANT (ค่าคงที่)

- การประกาศค่าคงที่ที่จะต้องระบุชื่อชนิดของข้อมูลและความค่าที่ต้องการ
- การประกาศใช้คำสำคัญ **CONSTANT** มันต้องมีค่าเริ่มต้นและไม่อนุญาตให้ค่าที่จะมีการเปลี่ยนแปลง ตัวอย่างเช่น:

```
PI CONSTANT NUMBER := 3.141592654;
```

- **DECLARE**
- -- constant declaration
- pi CONSTANT NUMBER := 3.141592654;
- -- other declarations
- radius NUMBER(5,2);
- dia NUMBER(5,2);
- circumference NUMBER(7, 2);
- area NUMBER(10, 2);
- **BEGIN**
- -- processing
- radius := 9.5;
- dia := radius * 2;
- circumference := 2.0 * pi * radius;
- area := pi * radius * radius;

Continue....

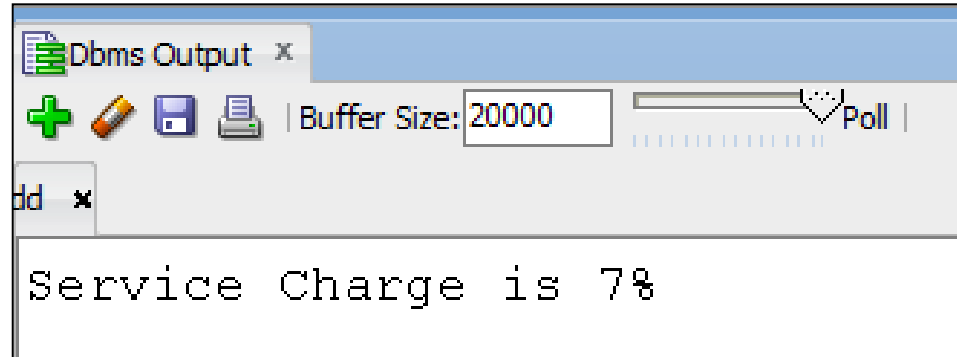
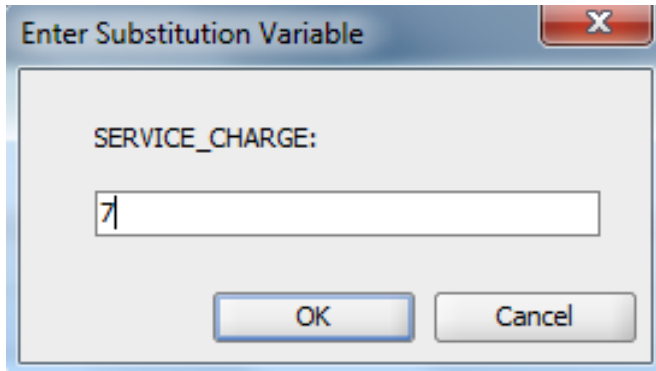
- `-- output`
- `dbms_output.put_line('Radius: ' || radius);`
- `dbms_output.put_line('Diameter: ' || dia);`
- `dbms_output.put_line('Circumference: ' || circumference);`
- `dbms_output.put_line('Area: ' || area);`
- **END;**
- `/`

```
Radius: 9.5  
Diameter: 19  
Circumference: 59.69  
Area: 283.53
```

การใช้ตัวแปรแบบ **Substitution Variable**

- ในการเขียนโปรแกรมมักจะมีการให้ผู้ใช้โปรแกรมใส่ค่าเข้ามาเป็น **Input** ของโปรแกรม เพื่อที่จะให้โปรแกรมนำไปใช้ในการทำงานของโปรแกรม
- เช่น โปรแกรมที่คิดราคาสินค้าอาจจะมีการถามที่หน้าจอว่าจะให้ส่วนลดกี่เปอร์เซ็นต์ ดังนั้นต้องมีการใช้ตัวแปรในการรับค่า **Input** จากผู้ใช้
- สำหรับการรับ **Input** จากหน้าจอเพื่อส่งให้โปรแกรมภาษา **PL/SQL** นั้นทำได้โดยผ่านทางตัวแปรที่เรียกว่า **Substitution Variable** โดยที่ตัวแปรแบบนี้จะมีการใช้เครื่องหมาย **&** นำหน้า

การใช้ตัวแปรแบบ Substitution Variable



- DECLARE**

- `service_charge number(2) := &service_charge ;`

- BEGIN**

- `DBMS_OUTPUT.PUT_LINE('Service Charge is' || service_charge
|| '%');`

- END ;**

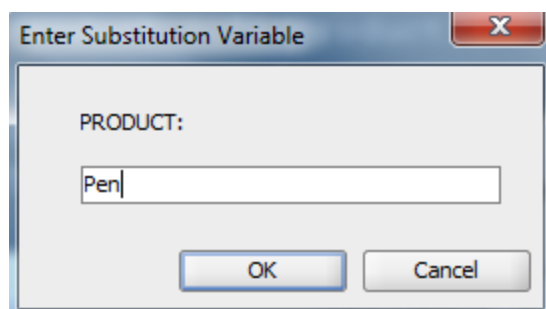
- /**

Practice: Substitution Variable

- จงเขียนโปรแกรม PL/SQL ในการแสดงข้อมูลต่อไปนี้ทางหน้าจอ
 - ชื่อสินค้า
 - จำนวนที่สั่งซื้อ
 - ราคาต่อหน่วย
 - โดยมีการกำหนดเงื่อนไขของโปรแกรมดังนี้คือ
 - ชื่อสินค้า จำนวนที่สั่งซื้อและราคาต่อหน่วยให้แสดงหน้าจอเพื่อรับข้อมูล
 - สำหรับผลลัพธ์ทางหน้าจอ ต้องการให้อยู่ในรูปแบบต่อไปนี้

Practice: Substitution Variable

Output:

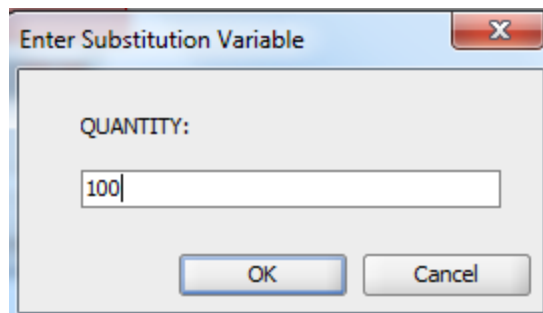


Enter Substitution Variable

PRODUCT:

Pen

OK Cancel

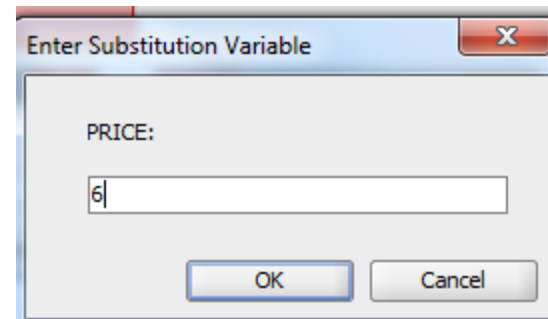


Enter Substitution Variable

QUANTITY:

100

OK Cancel

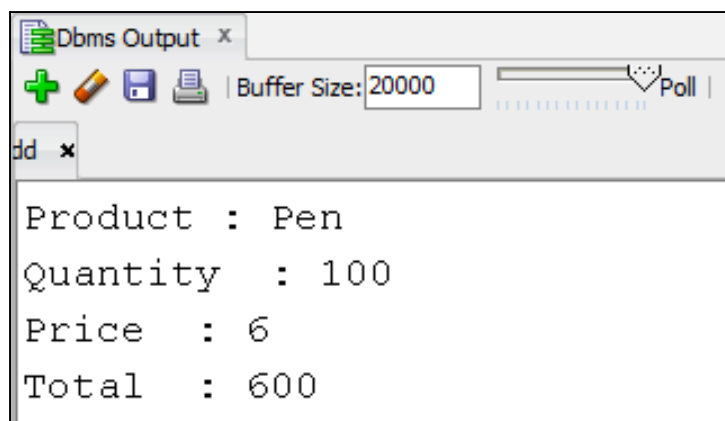


Enter Substitution Variable

PRICE:

6

OK Cancel

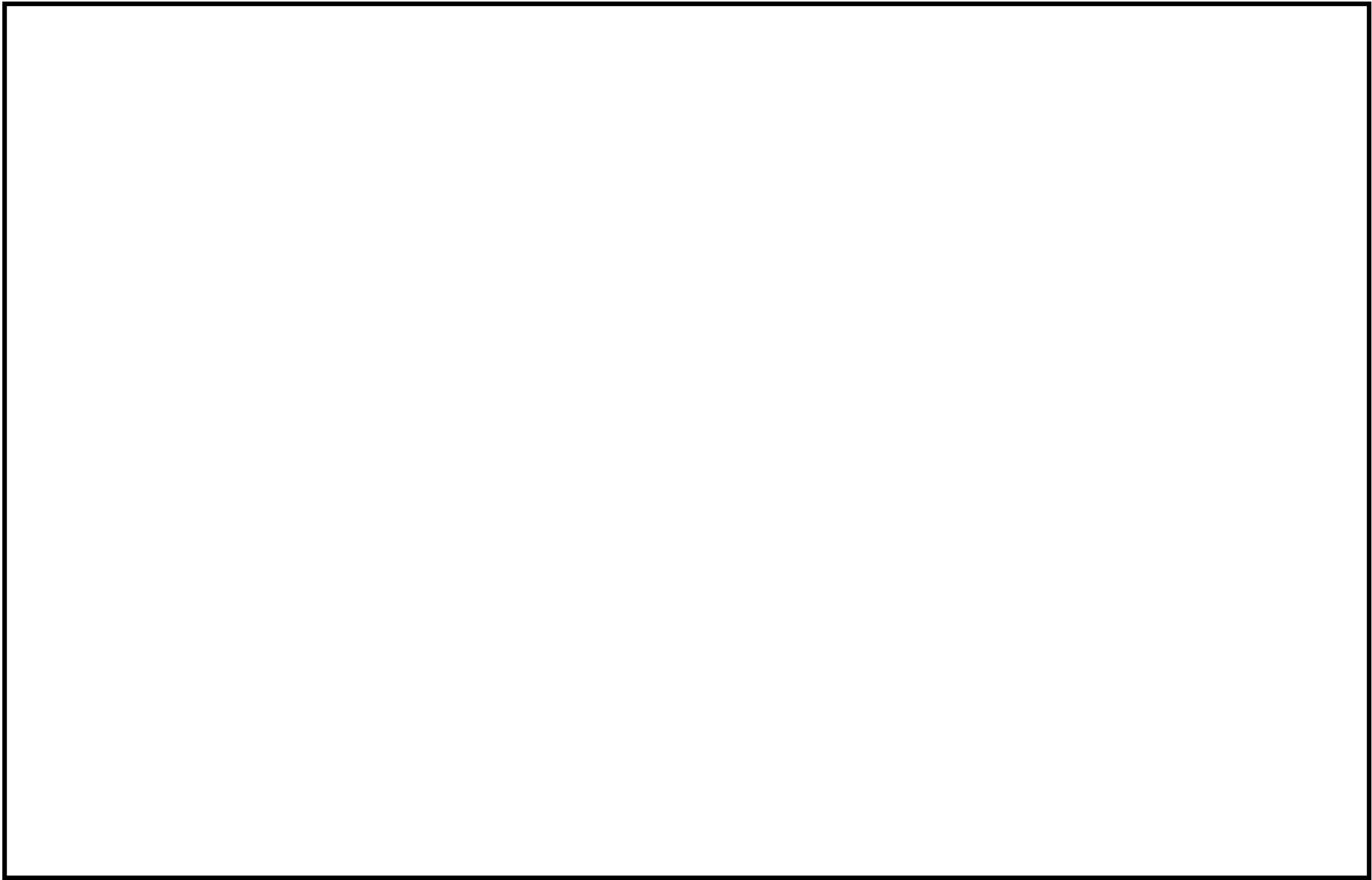


Dbms Output x

Buffer Size: 20000 Poll

dd x

```
Product : Pen
Quantity : 100
Price   : 6
Total   : 600
```



%TYPE Attribute

- Is used to declare a variable according to:
 - A database column definition
 - Another declared variable
- Is prefixed with:
 - The database table and column
 - The name of the declared variable

Declaring Variables with the %TYPE Attribute

Syntax

```
identifier      table.column_name%TYPE;
```

Examples

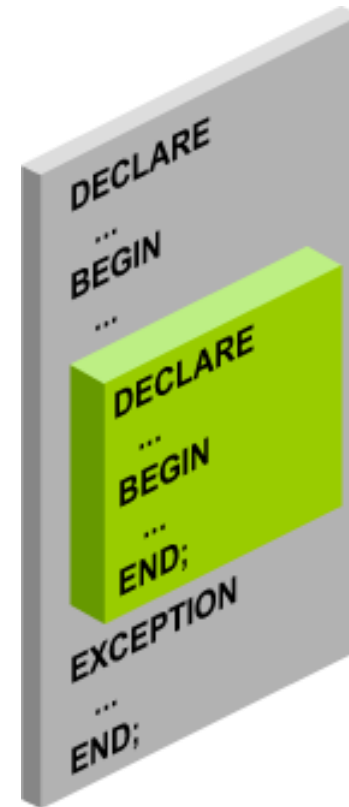
```
...  
    emp_lname      employees.last_name%TYPE;  
...
```

```
...  
    balance        NUMBER(7,2); ตัวเลขมีทศนิยม 2 ตำแหน่ง  
    min_balance    balance%TYPE := 1000;  
...
```

Nested Blocks

PL/SQL blocks can be nested.

- An executable section (`BEGIN ... END`) can contain nested blocks.
- An exception section can contain nested blocks.



ขอบเขตตัวแปรใน PL/SQL

- **PL/SQL** ช่วยในการทำงานของบล็อกคือแต่ละบล็อกโปรแกรมอาจมีบล็อกภายในอีก หากมีการประกาศตัวแปรภายในบล็อกภายในก็ไม่สามารถเข้าถึงบล็อกนอก แต่ถ้าตัวแปรมีการประกาศและเข้าถึงบล็อกด้านนอกก็ยังสามารถเข้าถึงบล็อกภายในทั้งหมดที่ซ้อนกัน มีสองประเภทของขอบเขตตัวแปร ได้แก่ :
 - **Local variables** - ตัวแปรที่ประกาศในบล็อกภายในและไม่สามารถเข้าถึงบล็อกนอก
 - **Global variables** - ตัวแปรประกาศในบล็อกนอกสุดหรือแพ็คเกจ

Variable Scope and Visibility : Example

```
• DECLARE
•   v_father_name VARCHAR2(20):='Patrick';
•   v_date_of_birth DATE:='20-Apr-1972';
• BEGIN
•   DECLARE
•     v_child_name VARCHAR2(20):='Mike';
•     v_date_of_birth DATE:='12-Dec-2002';
•     BEGIN
•       DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
1  DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
•   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
•   END;
2  DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
• END;
• /
```

- **DECLARE**
- -- Global variables
- num1 number := 95;
- num2 number := 85;
- **BEGIN**
- dbms_output.put_line('Outer Variable num1: ' || num1);
- dbms_output.put_line('Outer Variable num2: ' || num2);
- **DECLARE**
- -- Local variables
- num1 number := 195;
- num2 number := 185;
- **BEGIN**
- dbms_output.put_line('Inner Variable num1: ' || num1);
- dbms_output.put_line('Inner Variable num2: ' || num2);
- **END;**
- **END;**
- **/**

```
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185
```


เริ่มพสําน SQL กับภาษา PL/SQL เพื่อจัดการข้อมูล

SQL Statements in PL/SQL

- Retrieve a row from the database by using the `SELECT` command.
- Make changes to rows in the database by using DML commands.
- Control a transaction with the `COMMIT`, `ROLLBACK`, or `SAVEPOINT` command.

SELECT Statements in PL/SQL

Retrieve data from the database with a `SELECT` statement.

Syntax:

```
SELECT  select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
[WHERE  condition];
```

SELECT Statements in PL/SQL

- The INTO clause is required.
- Queries must return only one row.

Example:

```
• DECLARE
•   v_fname VARCHAR2(25);
• BEGIN
•   SELECT first_name INTO v_fname
•   FROM employees WHERE employee_id=200;
•   DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
• END;
• /
```

Retrieving Data in PL/SQL

Retrieve `hire_date` and `salary` for the specified employee is 100 and declaring variables with the `%TYPE` attribute.

```
Hire Date= 17-JUN-87  
Salary=    24,000.00
```

Practice: Retrieving Data in PL/SQL

Return the sum of the salaries for all the employees in the department is 60.

```
The sum of salary is 28,800.00
```

Naming Conventions

- Use a naming convention to avoid ambiguity in the `WHERE` clause.
- Avoid using database column names as identifiers.
- Syntax errors can arise because PL/SQL checks the database first for a column in the table.
- The names of local variables and formal parameters take precedence over the names of database *tables*.
- The names of database table *columns* take precedence over the names of local variables.

Naming Conventions

```
• DECLARE
•   hire_date      employees.hire_date%TYPE;
•   sysdate        hire_date%TYPE;
•   employee_id    employees.employee_id%TYPE := 176;
• BEGIN
•   SELECT   hire_date, sysdate
•   INTO     hire_date, sysdate
•   FROM     employees
•   WHERE    employee_id = employee_id;      ✗
• END;
• /
```

ผลลัพธ์หลายrecord

Error report:

ORA-01422: exact fetch returns more than requested number of rows

ORA-06512: at line 6

01422. 00000 - "exact fetch returns more than requested number of rows"

*Cause: The number specified in exact fetch is less than the rows returned.

*Action: Rewrite the query or change number of rows requested

การจัดการข้อผิดพลาดด้วย **Exception Section**

Exception Section

- ส่วนจัดการข้อผิดพลาดเป็นส่วนสุดท้ายของ **Block** ซึ่งอยู่หลังส่วน **Executable Section** โดยที่ส่วนจัดการข้อผิดพลาดนี้เป็นส่วนที่จะมีหรือไม่มีก็ได้ แต่ว่าในโปรแกรม **PL/SQL** ที่ดีแล้วควรมีส่วนนี้อยู่ด้วย เช่น ในโปรแกรม **PL/SQL** มีการใช้คำสั่ง **SELECT** เพื่อไปดึงข้อมูลจากระบบฐานข้อมูลแต่ไม่สามารถหาข้อมูลที่ต้องการได้ แบบนี้ในโปรแกรม **PL/SQL** ก็ถือว่ามีเกิด **Exception** ขึ้นในการทำงานโดยมีสาเหตุจากคำสั่ง **SELECT** ของโปรแกรม

Exception Section

- ตัวอย่างของวิธีการเขียน Exception Section เบื้องต้น

```
•BEGIN
• SELECT FIRST_NAME, LAST_NAME, SALARY INTO M_FIRST_NAME,
  M_LAST_NAME, M_SALARY
• FROM EMPLOYEES WHERE EMPLOYEE_ID=205;
• DBMS_OUTPUT.PUT_LINE('NAME IS' || M_FIRST_NAME);
• DBMS_OUTPUT.PUT_LINE('LAST NAME IS ' || M_LAST_NAME);
• DBMS_OUTPUT.PUT_LINE('SALARY IS ' || SALARY);
•EXCEPTION
• WHEN NO_DATA_FOUND THEN
• DBMS_OUTPUT.PUT_LINE('INVALID EMPLOYEE ID');
•END;
```

Predefined Exception

- Syntax

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    statement1;
    statement2;

  WHEN TOO_MANY_ROWS THEN
    statement1;
  WHEN OTHERS THEN
    statement1;
    statement2;
    statement3;
END;
```

Guidelines for Trapping Exceptions

- The `EXCEPTION` keyword starts the exception-handling section.
- Several exception handlers are allowed.
- Only one handler is processed before leaving the block.
- `WHEN OTHERS` is the last clause.

Trapping Predefined Oracle Server Errors

- Reference the predefined name in the exception-handling routine.
- Sample predefined exceptions:
 - NO_DATA_FOUND
 - TOO_MANY_ROWS
 - INVALID_CURSOR
 - ZERO_DIVIDE

Exception : NO_DATA_FOUND

•DECLARE

- c_id customers.id%type := 8;
- c_name customers.name%type;
- c_addr customers.address%type;

No such customer!

•BEGIN

- SELECT name, address INTO c_name, c_addr
- FROM customers WHERE id = c_id;
- DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
- DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);

-
-
-
-
-

•END;

•/

Exception : TOO_MANY_ROWS

```
• DECLARE
•   v_lname employees.last_name%TYPE;;
• BEGIN
•   SELECT last_name INTO v_lname
•   FROM employees
•   WHERE first_name='John';
•   DBMS_OUTPUT.PUT_LINE ('John''s last name is :'  
      ||v_lname);
• END;
• /
```

Error report:

ORA-01422: exact fetch returns more than requested number of rows

ORA-06512: at line 4

01422. 00000 - "exact fetch returns more than requested number of rows"

*Cause: The number specified in exact fetch is less than the rows returned.

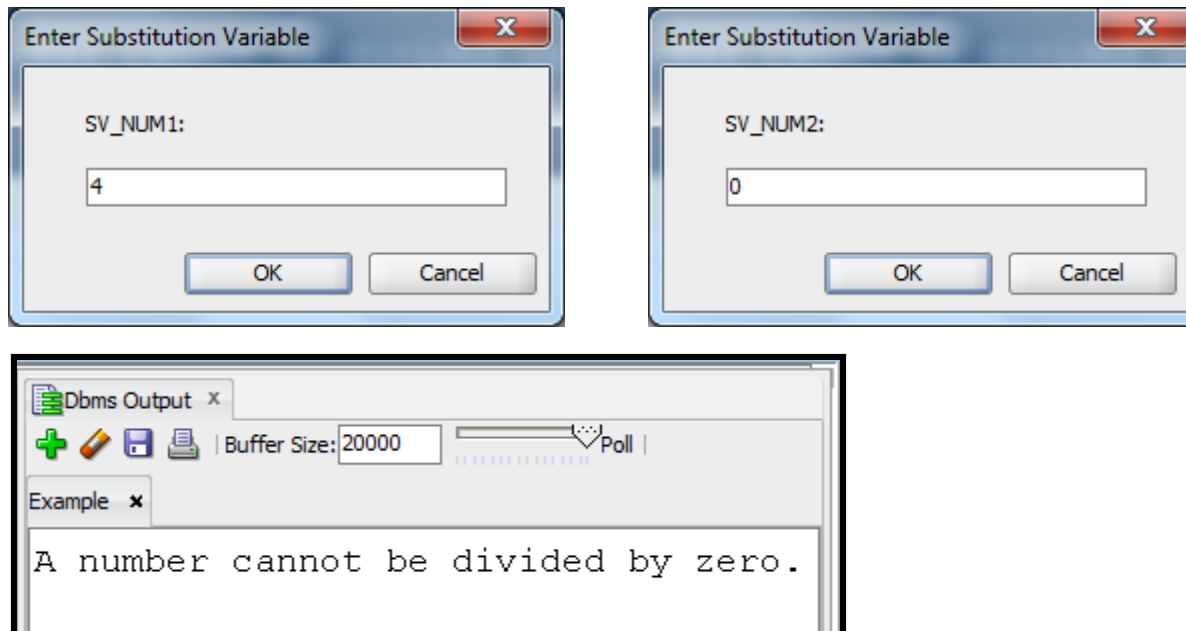
*Action: Rewrite the query or change number of rows requested

- **DECLARE**
- `v_lname employees.last_name%TYPE;;`
- **BEGIN**
- `SELECT last_name INTO v_lname`
- `FROM employees`
- `WHERE first_name='John';`
- `DBMS_OUTPUT.PUT_LINE ('John''s last name is :'`
 `||v_lname);`
-
-
-
- **END;**

Your select statement retrieved multiple rows.
Consider using a cursor.

Exception : ZERO_DIVIDE

- The section of the example in bold letters shows the exception-handling section of the block.
- When this example is executed with values of 4 and 0 for variables v_num1 and v_num2, respectively, the following output is produced:



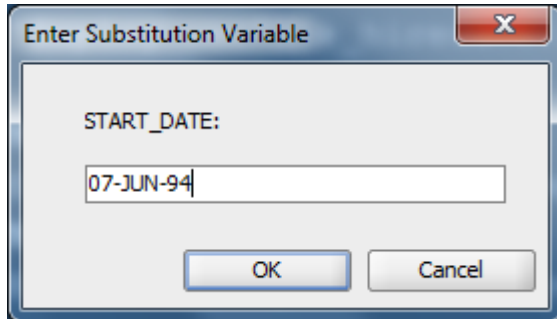
Exception : ZERO_DIVIDE

```
• DECLARE
•   v_num1 number := &sv_num1;
•   v_num2 number := &sv_num2;
•   v_result number;
• BEGIN
•   v_result := v_num1 / v_num2;
•   DBMS_OUTPUT.PUT_LINE ('v_result: ' || v_result);
•
•
•
•
• END;
• /
```

Exception Section

- This exercise contains two exceptions in the single exception handling section.
- The first exception, `NO_DATA_FOUND`, will be raised if there are no records in the `EMPLOYEE` table for a particular employee.
- The second exception, `TOO_MANY_ROWS`, will be raised if a particular employee is employ into more than one data.

Exception Section



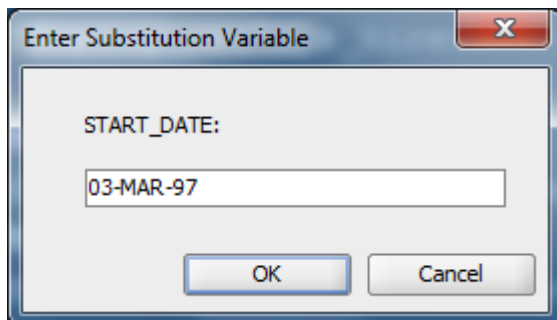
Enter Substitution Variable

START_DATE:

07-JUN-94

OK Cancel

```
Check the employee is start date  
The employee is employed into many person
```



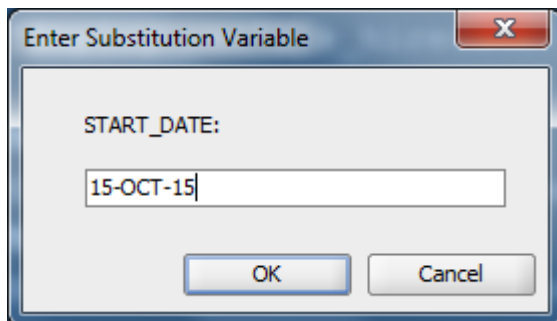
Enter Substitution Variable

START_DATE:

03-MAR-97

OK Cancel

```
Check the employee is start date  
The employee is Britney  
Employ on:03-MAR-97
```



Enter Substitution Variable

START_DATE:

15-OCT-15

OK Cancel

```
Check the employee is start date  
The employee is not employed
```

- **DECLARE**
- e_hiredate DATE := '&Start_Date';
- e_firstname employees.first_name%TYPE;;
- **BEGIN**
- DBMS_OUTPUT.PUT_LINE('Check the employee is start date');
- SELECT first_name INTO e_firstname
- FROM employees
- WHERE hire_date = e_hiredate;
- DBMS_OUTPUT.PUT_LINE('The employee is '||e_firstname);
- DBMS_OUTPUT.PUT_LINE('Employ on:'||e_hiredate);
-
-
-
-
-
-
-
- **END;**
- /

Exception Section : User Define

- PL / SQL สามารถให้ผู้ใช้เพิ่มหรือกำหนด Exception เองได้ โดยมีรูปแบบการทำงานดังนี้

```
• DECLARE
•     exception_name EXCEPTION;
• BEGIN
•     IF condition THEN
•         RAISE exception_name;
•     END IF;
• EXCEPTION
•     WHEN exception_name THEN
•         statement;
• END;
• /
```

Exception Section - User Define: Example

```
• DECLARE
•     c_id      customers.id%type := &cc_id;
•     c_name    customers.name%type;
•     c_addr    customers.address%type;
•     -- user defined exception
•     ex_invalid_id EXCEPTION;
• BEGIN
•     IF c_id <= 0 THEN
•         RAISE ex_invalid_id;
•     ELSE
•         SELECT  name, address INTO  c_name, c_addr
•         FROM    customers
•         WHERE   id = c_id;
•
•         DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
•         DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
•     END IF;
```

Continue...

ORACLE

-
-
-
-
-
-
-
-
- **END;**
- **/**

```
Enter value for cc_id: -6 (let's enter a value -6)
old 2: c_id customers.id%type := &cc_id;
new 2: c_id customers.id%type := -6;
ID must be greater than zero!
```

- จงเขียนโปรแกรม **PL/SQL** เพื่อทำการแสดง รหัสงาน ตำแหน่งงาน เงินเดือนขั้นต่ำและเงินเดือนขั้นสูง ของงานที่มีรหัสเป็น **IT_PROG** โดยมี **Output** ดังนี้

JOB ID = IT_PROG

JOB TITLE = Programmer

MIN SALARY = 4,000

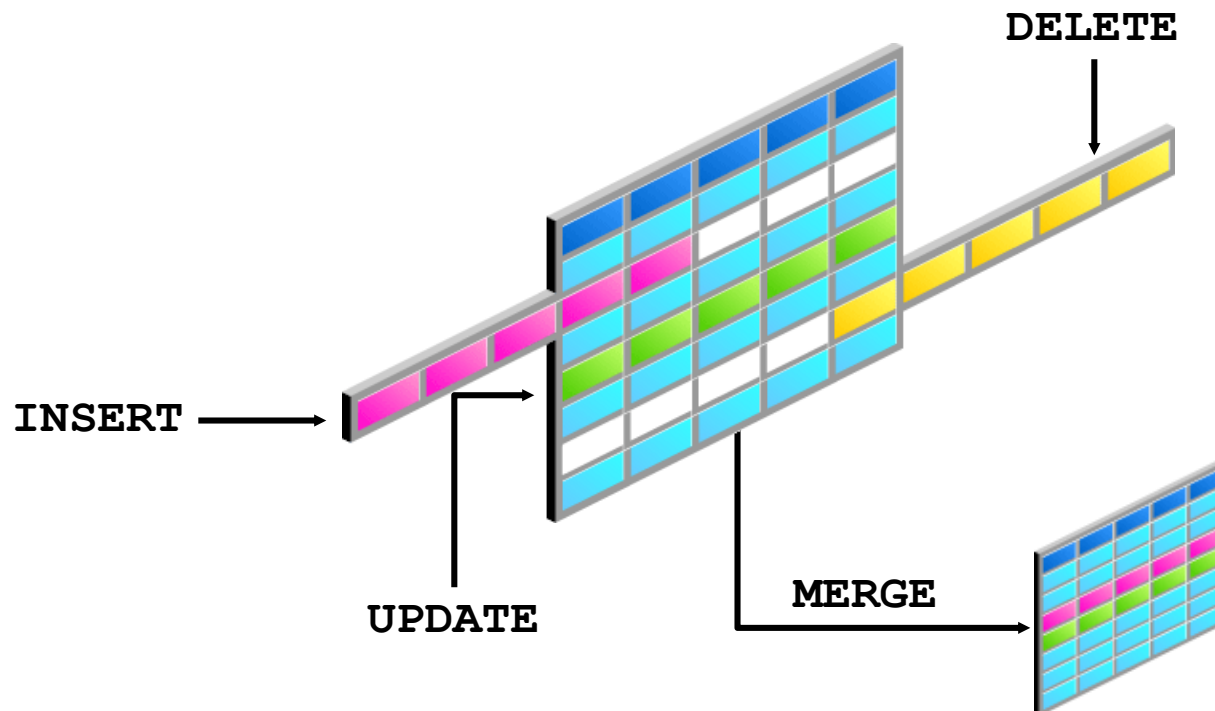
MAX SALARY = 10,000.00

Using PL/SQL to Manipulate Data

Using PL/SQL to Manipulate Data

Make changes to database tables by using DML commands:

- INSERT
- UPDATE
- DELETE



Inserting Data

Add new employee information to the EMPLOYEES table.

Example:

```
• BEGIN  
•   INSERT INTO employees  
•   VALUES (1001, 'Ruth', 'Cores', 'RCORES', '21-Feb-97',  
             'AD_ASST', 4000) ;  
• END ;  
• /
```

Inserting Data

Example:

- Use SQL Statement Create a table `JOB_BK` based on the structure of the `JOB` table. Contains `Job_id`, `Job_title`, `Min_salary` and `Max_salary`.

```
create table succeeded.
```

Inserting Data

Example:

- Use program PL/SQL insert 3 records into `JOB_BK` table below

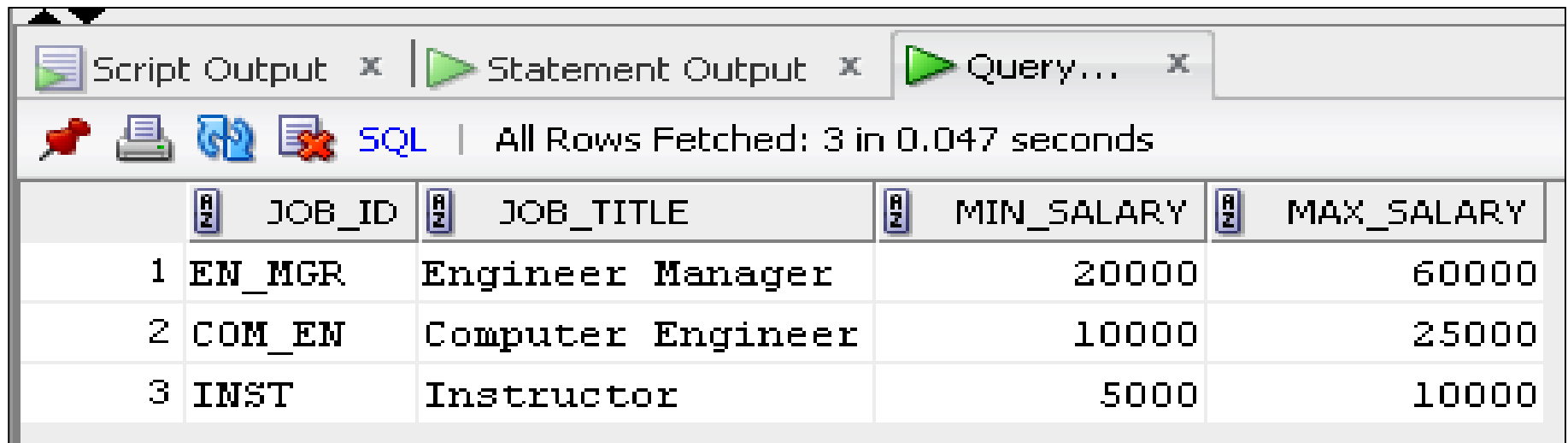
Job_id	Job_title	Min_salary	Max_salary
EN_MGR	Engineer Manager	20000	60000
COM_EN	Computer Engineer	10000	25000
INST	Instructor	5000	10000

Inserting Data



Inserting Data

• `Select * from JOB_BK;`



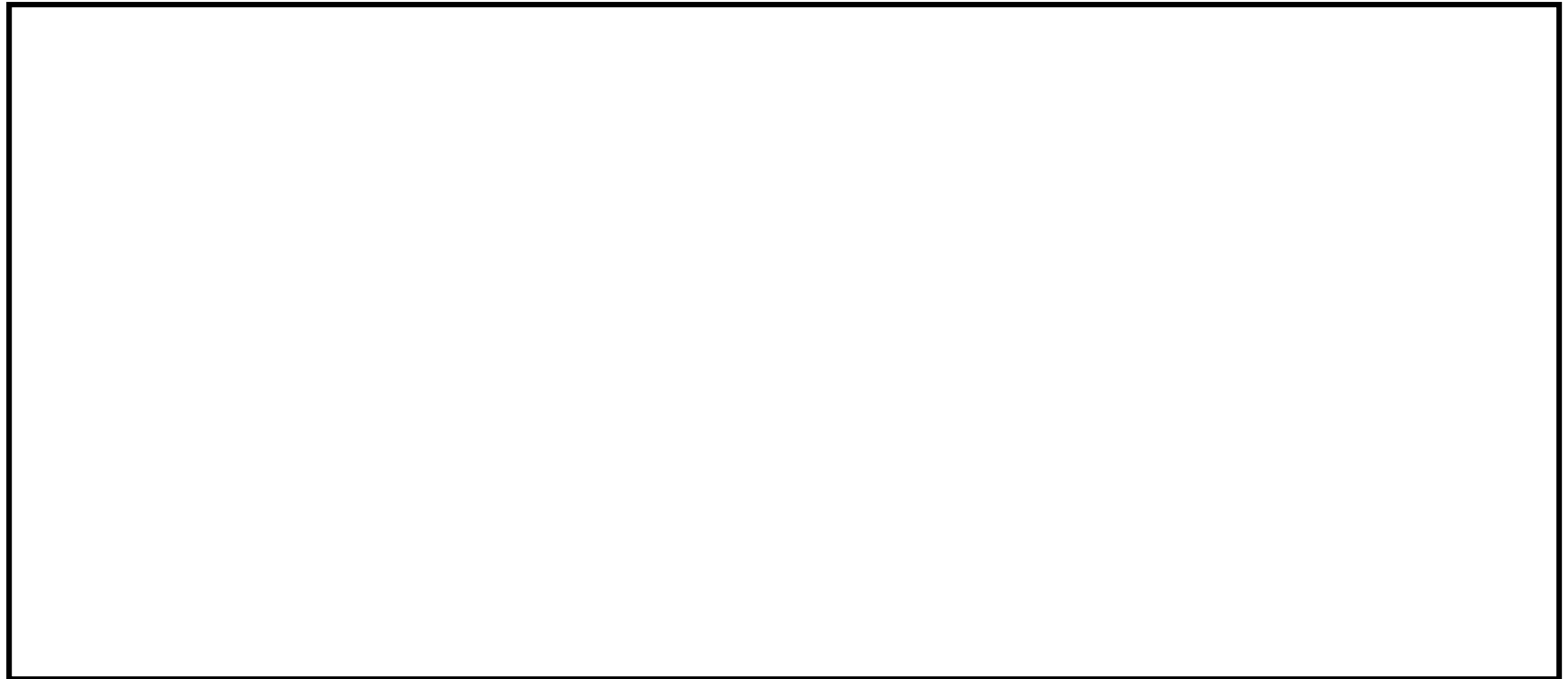
The screenshot shows a window with three tabs: 'Script Output', 'Statement Output', and 'Query...'. The 'Query...' tab is active, displaying a table of results. Above the table, it says 'SQL | All Rows Fetched: 3 in 0.047 seconds'. The table has five columns: 'JOB_ID', 'JOB_TITLE', 'MIN_SALARY', and 'MAX_SALARY'. There are three rows of data.

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	EN_MGR	Engineer Manager	20000	60000
2	COM_EN	Computer Engineer	10000	25000
3	INST	Instructor	5000	10000

Updating Data

Increase the salary of all employees who are stock clerks.

Example:

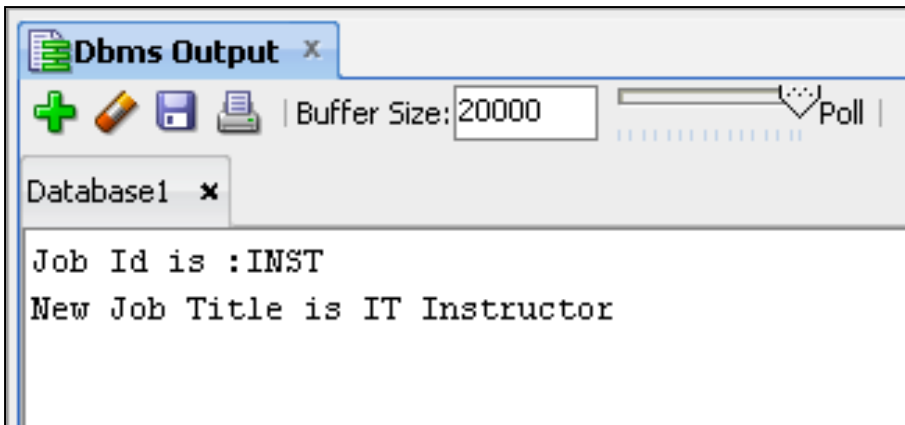


Updating Data

Example:

- Use program PL/SQL Update `JOB_BK` table โดยให้เปลี่ยนค่าของคอลัมน์ `Job_title` ให้มีค่าเป็นคำว่า `II Instructor` สำหรับ `Record` ที่มี `Job_id` เท่ากับ `INST` พร้อมทั้งแสดงค่าของ `Record` ที่มีการเปลี่ยนแปลงออกทางหน้าจอด้วย

Output:

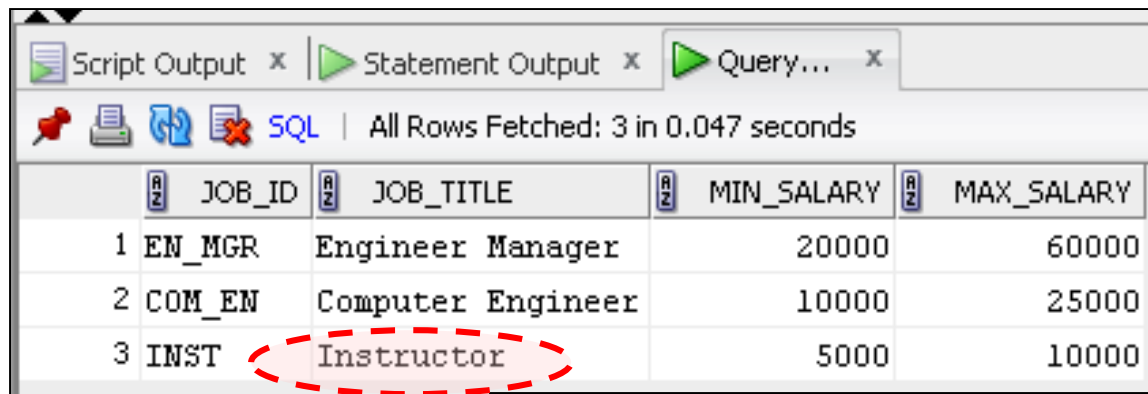


```
Dbms Output x
+ | Buffer Size: 20000 | Poll |
Database1 x
Job Id is :INST
New Job Title is IT Instructor
```

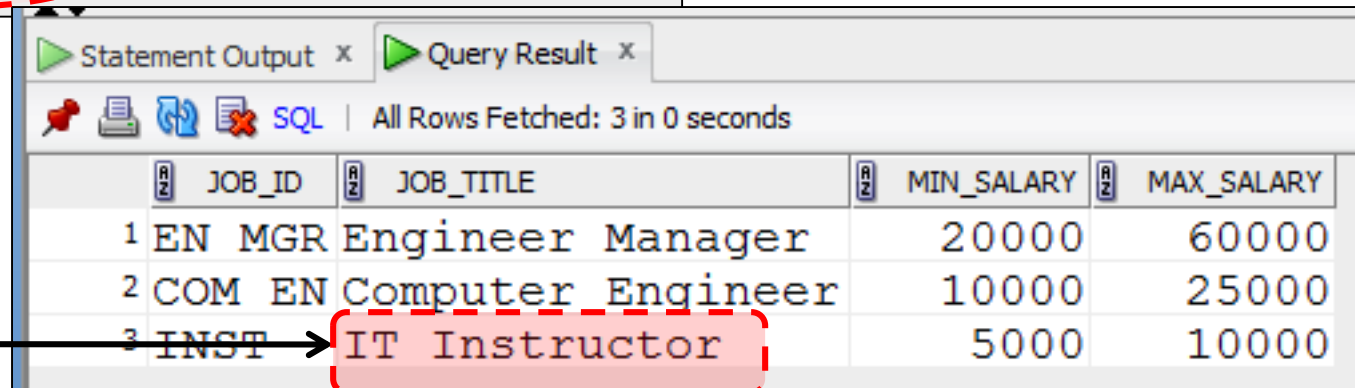


Updating Data

• `Select * from JOB_BK;`



	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	EN_MGR	Engineer Manager	20000	60000
2	COM_EN	Computer Engineer	10000	25000
3	INST	Instructor	5000	10000



	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	EN_MGR	Engineer Manager	20000	60000
2	COM_EN	Computer Engineer	10000	25000
3	INST	IT Instructor	5000	10000

Deleting Data

Delete rows that belong to department 10 from the `employees` table.

Example:

```
• DECLARE  
•   deptno    employees.department_id%TYPE := 10;  
• BEGIN  
•   DELETE FROM    employees  
•   WHERE  department_id = deptno; (เงื่อนไข ขวา ต้องไม่เท่ากับ ซ้าย)  
• END ;  
• /
```

Deleting Data

Practice:

- Use program PL/SQL delete rows that belong to Job_id =INST from the JOB_BK table

Record Deleted...

SQL Cursor

- A cursor is a pointer to the private memory area allocated by the Oracle server.
- A cursor is used to handle the result set of a `SELECT` statement.
- Using SQL cursor attributes, you can test the outcome of your SQL statements.

<code>SQL%FOUND</code>	Boolean attribute that evaluates to <code>TRUE</code> if the most recent SQL statement returned at least one row
<code>SQL%NOTFOUND</code>	Boolean attribute that evaluates to <code>TRUE</code> if the most recent SQL statement did not return even one row
<code>SQL%ROWCOUNT</code>	An integer value that represents the number of rows affected by the most recent SQL statement

SQL Cursor : Example

- Delete rows that have the specified employee ID from the `employees` table. Print the number of rows deleted.

```
• DECLARE  
•   v_rows_deleted VARCHAR2(30)  
•   v_empno employees.employee_id%TYPE := 176;  
• BEGIN  
•   DELETE FROM employees  
•   WHERE employee_id = v_empno;  
•   v_rows_deleted := (SQL%ROWCOUNT || ' row deleted.');
```

```
•   DBMS_OUTPUT.PUT_LINE (v_rows_deleted);  
• END;  
• /
```

SQL Cursor : Example

- เขียนโปรแกรม PL/SQL ดังนี้

1. ใช้คำสั่ง **SELECT** ในการดึงชื่อของพนักงานที่มีหมายเลข 206

ขึ้นมาพิมพ์บนหน้าจอ และให้ใช้ **SQL%ROWCOUNT** ในการบอกจำนวน **Record** ที่ดึงขึ้นมา

2. จากนั้นให้ใช้คำสั่ง **UPDATE** เพื่อทำการเปลี่ยนค่าเงินเดือนของพนักงานในแผนกที่ 50 ที่มีเงินเดือนน้อยกว่า 2500 ให้เป็น 3000

และให้ใช้ **SQL%ROWCOUNT** เพื่อดูว่ามีจำนวน **Record** เท่าไรที่ถูกเปลี่ยนแปลงค่าไปโดยการใช้คำสั่ง **Update**

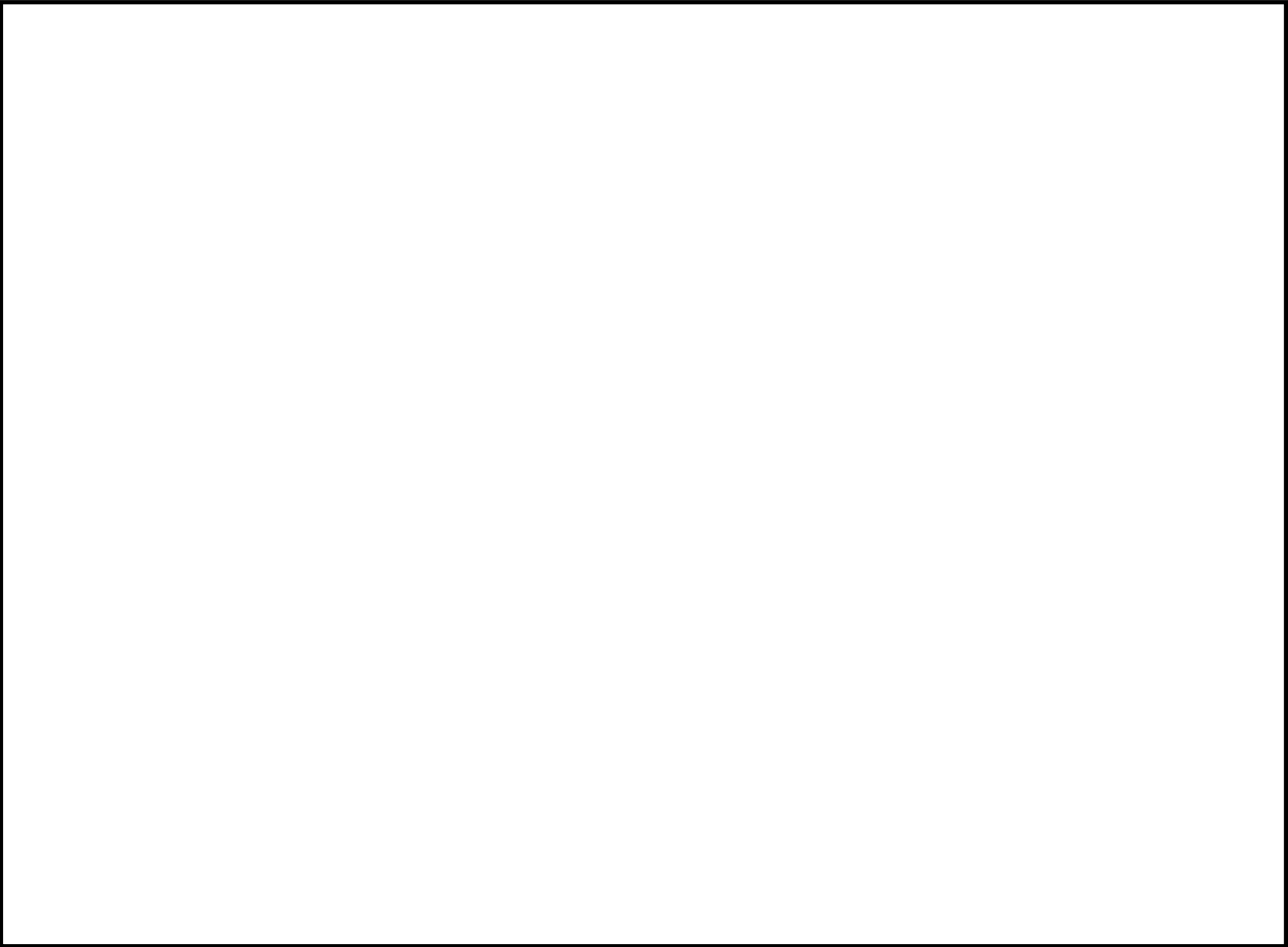
SQL Cursor : Example

1 Records Retrived
5 Records Updated

ORACLE

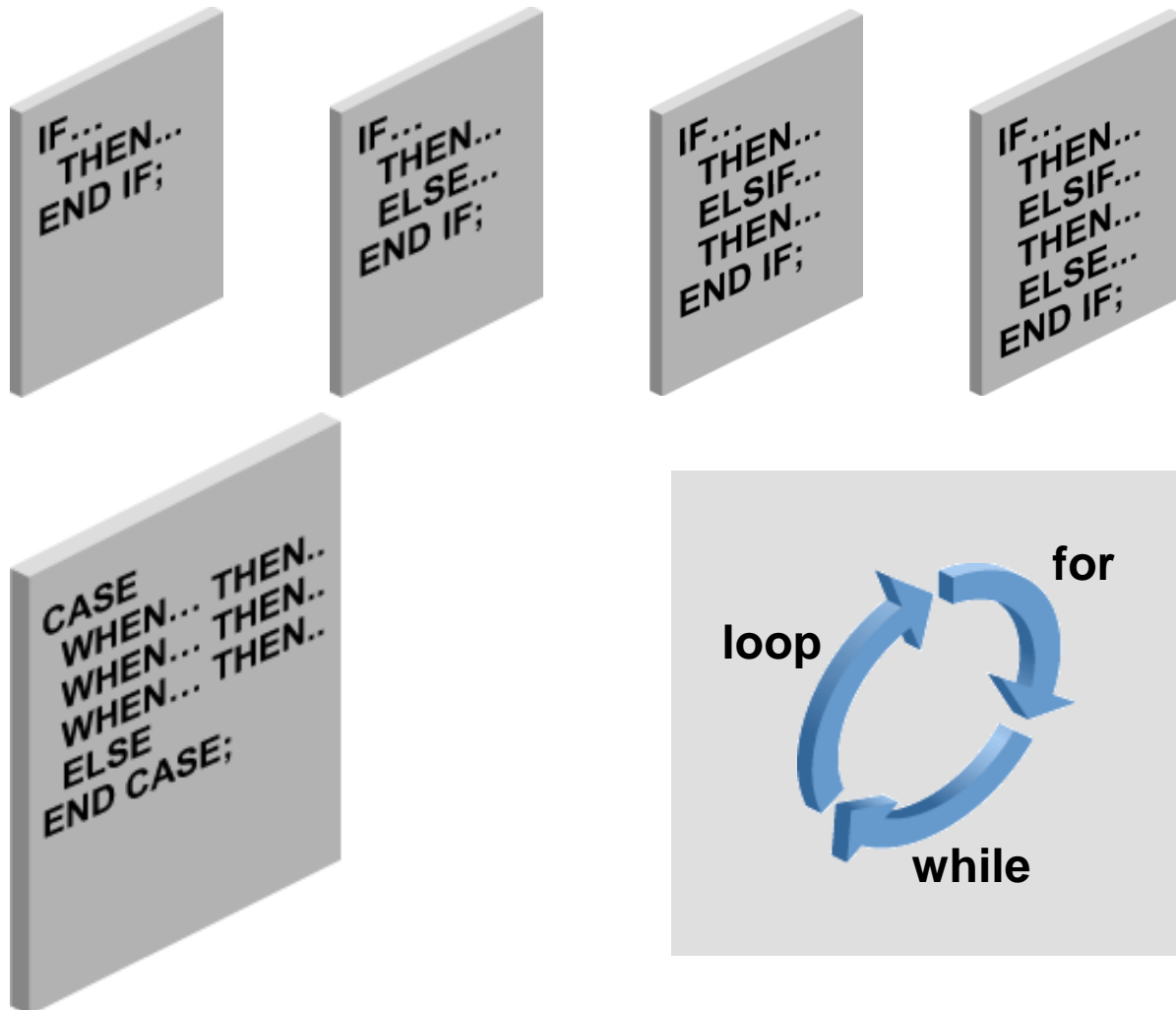
SQL Cursor : Exercise

- จากตาราง **Customer** เพิ่มเงินเดือนของลูกค้าแต่ละคน คนละ **500** บาท
- และให้ใช้ **SQL%FOUND** || **SQL%NOTFOUND** เพื่อตรวจสอบข้อมูลถ้าไม่สามารถปรับปรุงข้อมูลเงินเดือนไม่ได้ให้แสดงข้อความ "no customers selected"
- ถ้าสามารถแก้ไขได้ให้ใช้ **SQL% ROWCOUNT** เพื่อตรวจสอบว่ามีข้อมูลที่ปรับปรุงทั้งหมดกี่แถว



คำสั่งตรวจสอบเงื่อนไขและคำสั่งควบคุมการวนซ้ำ

Controlling Flow of Execution



IF Statement

Syntax:

- **IF** *condition* **THEN**
- *statements*; เป็นคำสั่ง **sql** ก็คำสั่งก็ได้ ไม่ต้องมี { }
- [**ELSIF** *condition* **THEN**
- *statements*;
- [**ELSE**
- *statements*;
- **END IF**;

Simple IF Statement

```
• DECLARE
•   v_myage  number:=31;
• BEGIN
•   IF v_myage < 11
•   THEN
•       DBMS_OUTPUT.PUT_LINE(' I am a child ');
•   END IF;
• END;
• /
```

Output:

```
anonymous block completed
```

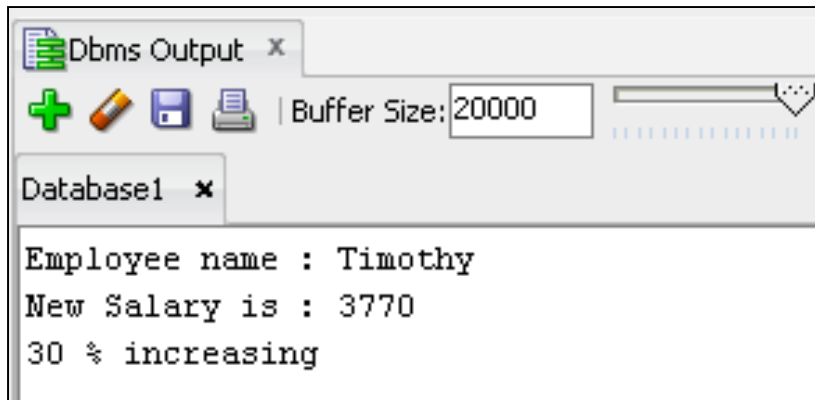
IF ELIF ELSE Clause

```
• DECLARE
•   v_myage number:=31;
• BEGIN
•   IF v_myage < 11 THEN
•       DBMS_OUTPUT.PUT_LINE(' I am a child ');
•   ELIF v_myage < 20 THEN
•       DBMS_OUTPUT.PUT_LINE(' I am young ');
•   ELIF v_myage < 30 THEN
•       DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
•   ELIF v_myage < 40 THEN
•       DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
•   ELSE
•       DBMS_OUTPUT.PUT_LINE(' I am always young ');
•   END IF;
• END;
• /
```

Output: anonymous block completed
I am in my thirties

Exercise : IF ELSIF ELSE Clause

- จงเขียนโปรแกรม PL/SQL เพื่อดึงข้อมูลชื่อ และเงินเดือน โดยต้องการดึงข้อมูลเฉพาะพนักงานที่มีรหัสเป็น 190 และตรวจสอบเงื่อนไขการทำงานดังนี้
 - ถ้าเงินเดือนพนักงานน้อยกว่า 3000 ให้ขึ้นเงินเดือน 30%
 - ถ้าเงินเดือนพนักงานมีค่าตั้งแต่ 3000 ให้ขึ้นเงินเดือน 20%



```
Dbms Output x
+  | Buffer Size: 20000
Database1 x
Employee name : Timothy
New Salary is : 3770
30 % increasing
```


CASE Expressions

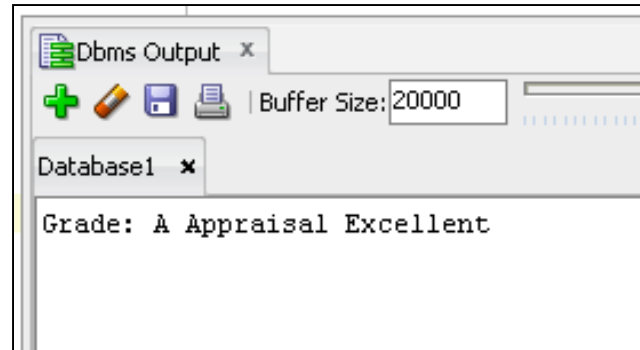
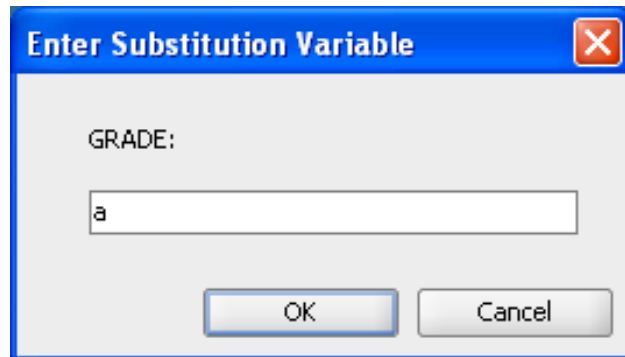
Return ค่าได้ ค่าใดค่าหนึ่ง

- A CASE expression selects a result and returns it.
- To select the result, the CASE expression uses expressions. The value returned by these expressions is used to select one of several alternatives.

```
• CASE selector
•   WHEN expression1 THEN result1
•   WHEN expression2 THEN result2
•   ...
•   WHEN expressionN THEN resultN
•   [ELSE resultN+1]
• END CASE;
• /
```

Example:CASE Expressions

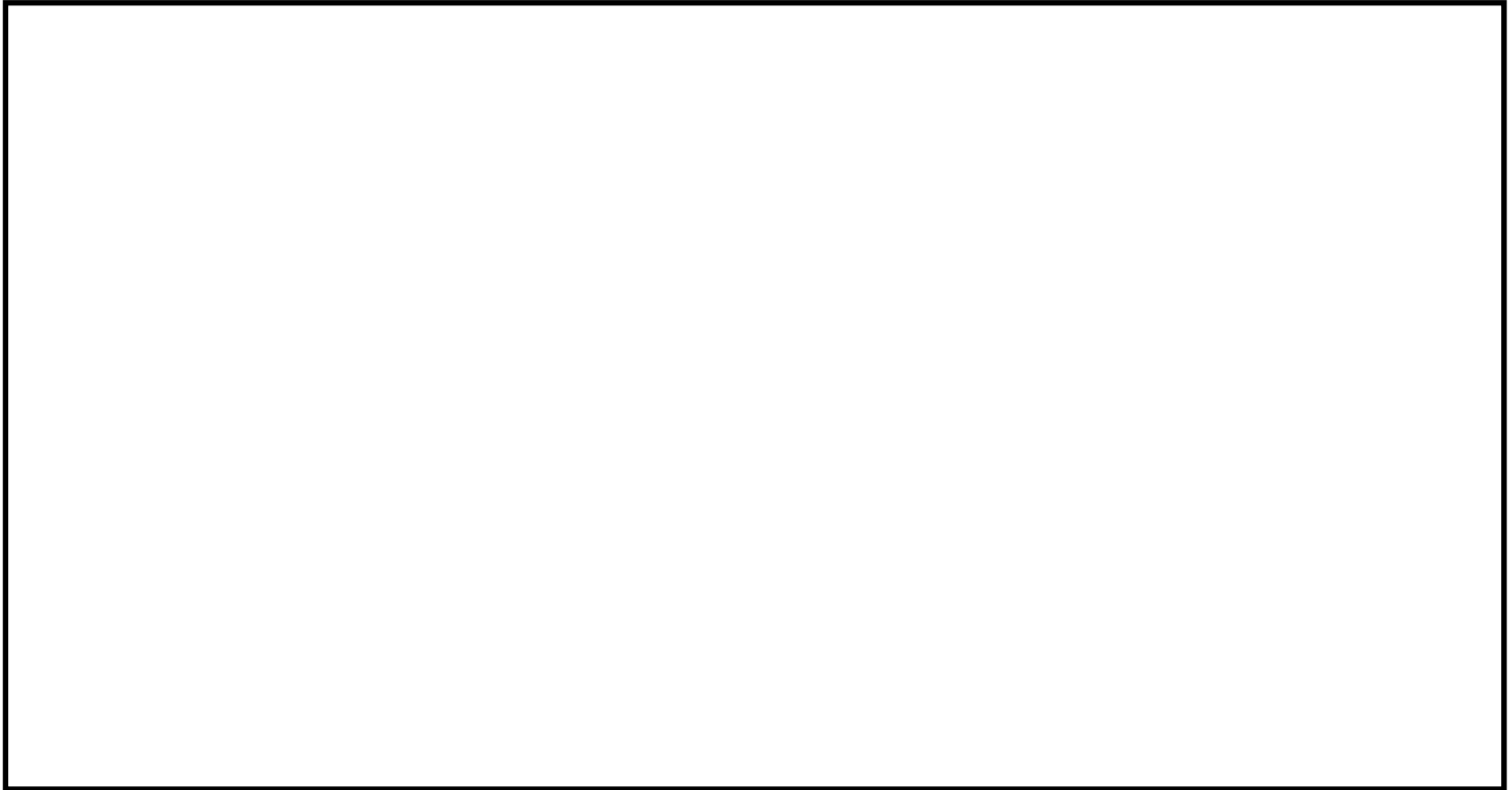
- จงเขียนโปรแกรม PL/SQL เพื่อรับเกรดและแสดงข้อความตามเงื่อนไขที่กำหนดต่อไปนี้
 - ถ้าป้อนตัวอักษรตัว A ให้แสดงข้อความ Excellent
 - ถ้าป้อนตัวอักษรตัว B ให้แสดงข้อความ Very Good
 - ถ้าป้อนตัวอักษรตัว C ให้แสดงข้อความ Good
 - ถ้าป้อนตัวอักษรตัวอื่น ๆ ให้แสดงข้อความ No such grade



CASE Expressions: Example

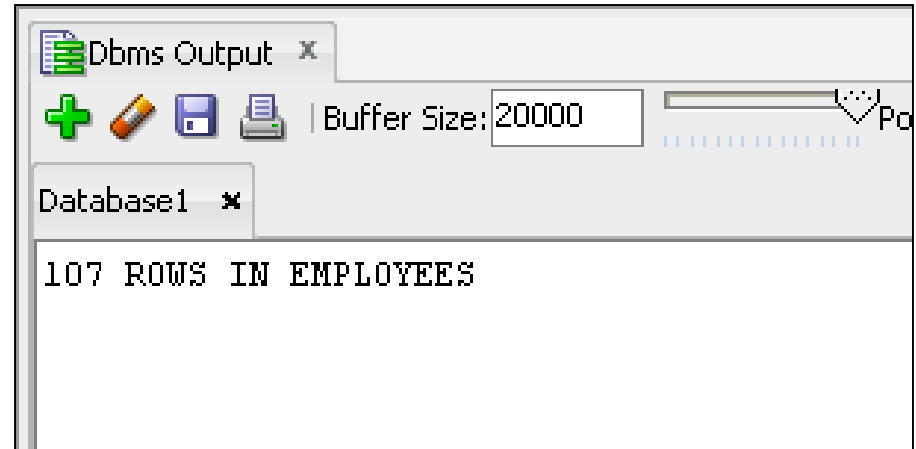
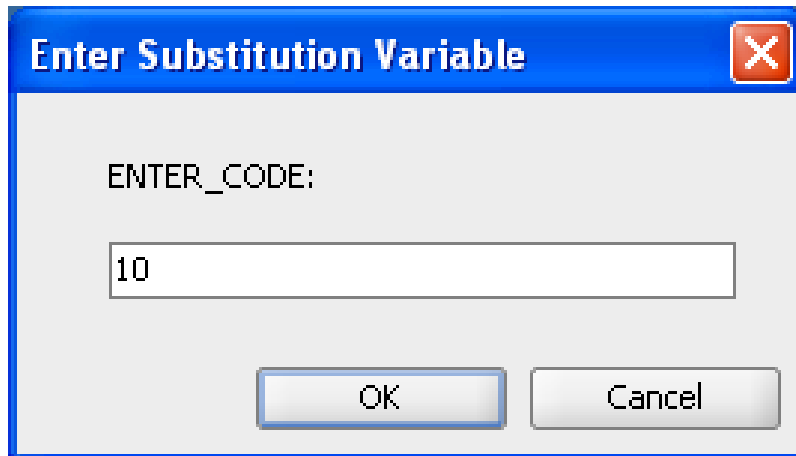


Searched CASE Expressions



Practice:CASE Expressions

- จงเขียนโปรแกรม **PL/SQL** เพื่อรับตัวเลขและแสดงข้อความตามเงื่อนไขที่กำหนดต่อไปนี้
 - ถ้าป้อนเลข 10 ให้นำจำนวนข้อมูลในตาราง **Employees**
 - ถ้าป้อนเลข 20 ให้นำจำนวนข้อมูลในตาราง **Departments**
 - ถ้าป้อนเลข 30 ให้นำจำนวนข้อมูลในตาราง **Jobs**
 - ถ้าป้อนเลขอื่น ๆ ให้แสดงข้อความว่า **Out of Number!!!**



Iterative Control: LOOP Statements

- Loops repeat a statement (or sequence of statements) multiple times.
- There are three loop types:
 - Basic loop
 - FOR loop
 - WHILE loop



Basic Loops

Syntax:

```
LOOP
  statement1;
  . . .
  EXIT [WHEN condition];
END LOOP;
```

Basic Loops : Example

- **DECLARE**

- `x number := 10;`

- **BEGIN**

- `LOOP`

- `dbms_output.put_line(x);`

- `x := x + 10;`

- `exit WHEN x > 50;`

- `END LOOP;`

- `-- after exit, control resumes here`

- `dbms_output.put_line('After Exit x is: ' || x);`

- **END;**

- `/`

10

20

30

40

50

After Exit x is: 60

WHILE Loops

Syntax:

```
WHILE condition LOOP  
    statement1;  
    statement2;  
    . . .  
END LOOP;
```

Use the `WHILE` loop to repeat statements while a condition is `TRUE`.

WHILE Loops : Example

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19

- DECLARE**

- `a number(2) := 10;`

- BEGIN**

- `WHILE a < 20 LOOP`

- `dbms_output.put_line('value of a: ' || a);`

- `a := a + 1;`

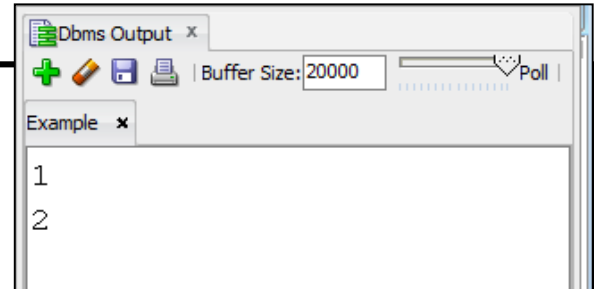
- `END LOOP;`

- END;**

- /**

WHILE Loops Example: Output?

- **DECLARE**
- X NUMBER :=1;
- Y NUMBER :=1;
- **BEGIN**
- WHILE (X<=10) LOOP
- DBMS_OUTPUT.PUT_LINE (X) ;
- X :=X+1;
- Y :=Y*3;
- EXIT WHEN Y>8;
- END LOOP;
- **END;**
- /



FOR Loops

- Use a `FOR` loop to shortcut the test for the number of iterations.
- Do not declare the counter; it is declared implicitly.

```
FOR counter IN [REVERSE] lower_bound..upper_bound LOOP
    statement1;
    statement2;
    .
    .
    .
END LOOP;
```

for i in 1..3

จะเป็น **for i in 3..1** ไม่ได้

ตัวแปร **i** มาเอง เมื่อสั่ง **for** ไม่ต้องประกาศ

ขยับค่าได้ที่ละ 1 ต้องวิ่งจากน้อยไปมากเสมอ

FOR Loops

Guidelines

- Reference the counter within the loop only; it is undefined outside the loop.
- Do not reference the counter as the target of an assignment.
- Neither loop bound should be `NULL`.

FOR Loops:Example

SQL Statement:

- DECLARE**

- `a number(2);`

- BEGIN**

- `FOR a in 10 .. 20 LOOP`

- `dbms_output.put_line('value of a: ' || a);`

- `END LOOP;`

- END;**

- /**

```
value of a: 10  
value of a: 11  
value of a: 12  
value of a: 13  
value of a: 14  
value of a: 15  
value of a: 16  
value of a: 17  
value of a: 18  
value of a: 19  
value of a: 20
```

```
value of a: 20  
value of a: 19  
value of a: 18  
value of a: 17  
value of a: 16  
value of a: 15  
value of a: 14  
value of a: 13  
value of a: 12  
value of a: 11  
value of a: 10
```

- **DECLARE**

- `a number(2);`

- **BEGIN**

- `FOR a IN REVERSE 10 .. 20 LOOP`

- `dbms_output.put_line('value of a: ' || a);`

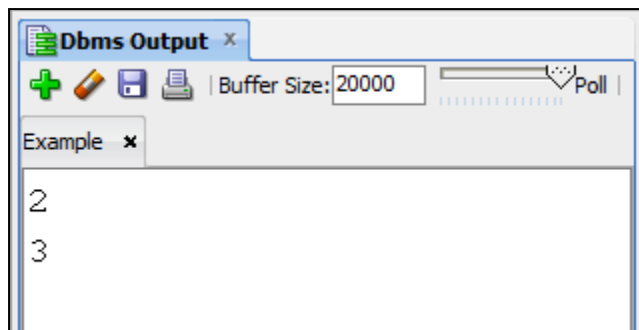
- `END LOOP;`

- **END;**

- `/`

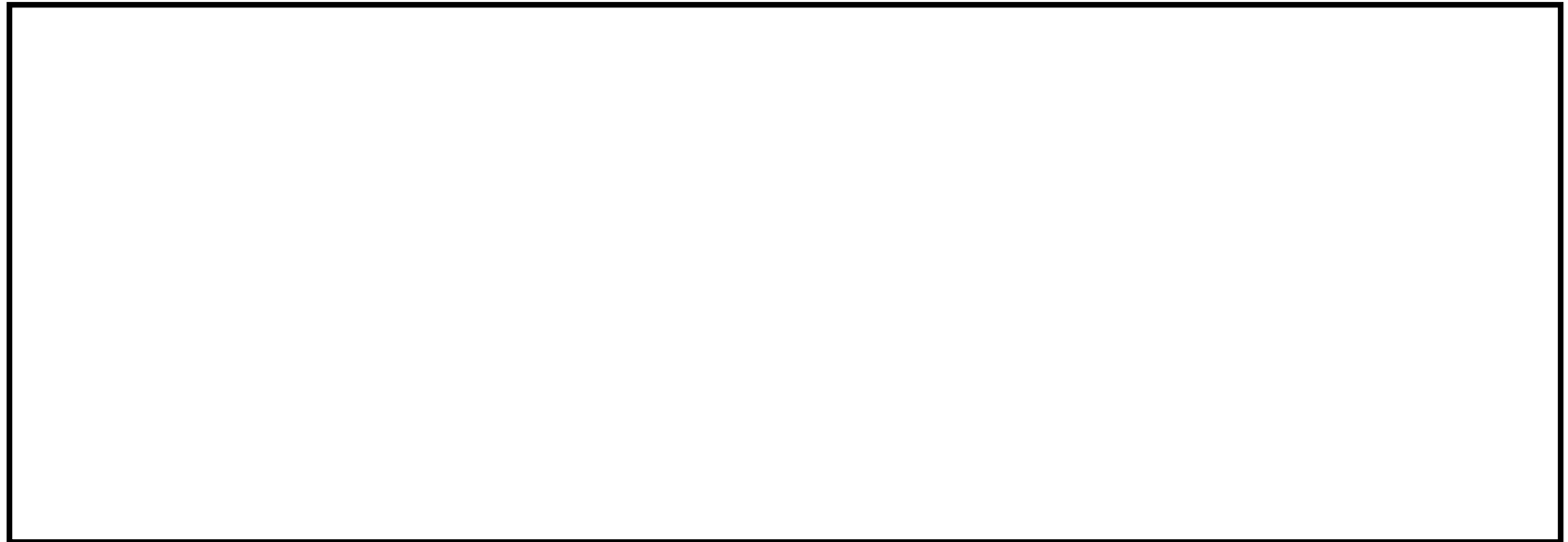
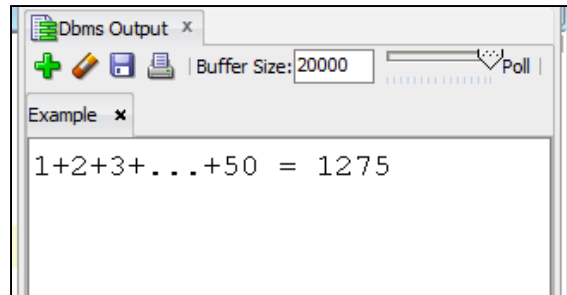
FOR Loops: Output ?

- **BEGIN**
- FOR X IN 2..10 LOOP
- DBMS_OUTPUT.PUT_LINE(X) ;
- EXIT WHEN MOD(X, 3)=0 ;
- END LOOP ;
- **END ;**
- /



LOOP Statements : Exercise

- Calculate the sum of the numbers from 1 to 50.



Practice : Loop Statement

- จงเขียนโปรแกรม PL/SQL เพื่อทำการเพิ่มข้อมูล location_id,city,country_id จำนวน 3 Record ลงในตาราง Locations โดยกำหนดให้ city เป็น Montreal ,country_id เป็น CA ส่วน location_id ให้ผู้ใช้ตรวจสอบว่า รหัสของ location_id สูงสุด ที่รหัสประเทศเป็น CA เป็นรหัสอะไรและให้เพิ่มข้อมูล ต่อท้าย เช่น รหัสสุดท้าย location_id ที่มี country_id=CA เป็น 1900 ข้อมูลที่เพิ่มต่อท้ายจะเป็น 1901 1902 และ 1903 เป็นต้น

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1901 (null)	(null)	Montreal	(null)	CA
2	1902 (null)	(null)	Montreal	(null)	CA
3	1903 (null)	(null)	Montreal	(null)	CA
4	1000 1297 Via Cola di Rie	00989	Roma	(null)	IT
5	1100 93091 Calle della Testa	10934	Venice	(null)	IT
6	1200 2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
7	1300 9450 Kamiya-cho	6823	Hiroshima	(null)	JP
8	1400 2014 Jabberwocky Rd	26192	Southlake	Texas	US
9	1500 2011 Interiors Blvd	99236	South San Francisco	California	US
10	1600 2007 Zagora St	50090	South Brunswick	New Jersey	US
11	1700 2004 Charade Rd	98199	Seattle	Washington	US
12	1800 147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA

WHILE Loops



FOR Loops

