Basic PL/SQL: Variable/Parameter Declaration

```
CREATE TABLE STUDENT(
   student_id VARCHAR2(10) NOT NULL PRIMARY KEY,
   first_name VARCHAR2(20),
   last_name  VARCHAR2(20),
   department VARCHAR2(30));
```

STUDENT

student_id	first_name	last_name	department
101	John	Smith	Computer Science
102	Robert	Hodges	Computer Science
103	Maria	Lopez	Mathematics
104	Harry	Burke	Physics
105	Annie	Nguyen	Computer Science

Basic PL/SQL: Local Variable Declaration

- Cursor Variable

```
CURSOR c_name IS
   SELECT first_name
   FROM STUDENT
   WHERE Department = 'Computer Science';
```

c_name

first_name
John
Robert
Annie

Basic PL/SQL: IF-THEN-ELSE example

```
DECLARE
  v_NumberSeats ROOMS.number_seats%TYPE;
  v_Comment VARCHAR2(35);

BEGIN
  -- Retrieve the number of seats in the room identified
  -- by ID 99999. Store the result in v_NumberSeats.

  SELECT number_seats
  INTO v_NumberSeats
  FROM ROOMS
  WHERE room_id = 99999;

  IF v_NumberSeats < 50 THEN
    v_Comment := 'Fairly small';
  ELSE
    IF v_NumberSeats < 100 THEN
       v_Comment := 'A little bigger';
    ELSE
       v_Comment := 'Lots of room';
    END IF;
  END IF;

  INSERT INTO TEMP_TABLE (char_col)
         VALUES (v_Comment);
END;
/
```

ROOMS

room_id	number_seats
10000	100
20000	60
99999	20

TEMP_TABLE

char_col

Basic PL/SQL: Loops

```
DECLARE
  v_Counter NUMBER :=1;

BEGIN
  LOOP
    -- Insert a row into TEMP_TABLE with the
    -- current value of the loop counter.

    INSERT INTO TEMP_TABLE
    VALUES (v_Counter, 'Loop index');

    v_Counter := v_Counter + 1;

    -- Exit condition - when the loop counter > 50
    -- we will break out of the loop.

    IF v_Counter > 50 THEN
      EXIT;
    END IF;

 END LOOP;
END;
/
```

TEMP_TABLE

counter	comment
1	Loop Index
…	…
50	Loop Index

Stored Procedures: Example 2

```
CREATE OR REPLACE PROCEDURE AddNewStudent (
  p_ID          STUDENTS.id%TYPE,
  p_FirstName   STUDENTS.first_name%TYPE,
  p_LastName    STUDENTS.last_name%TYPE,
  p_Major       STUDENTS.major%TYPE) AS

BEGIN
  -- Insert a new row in the students table. Use
  -- 0 for total_current_credits.
      INSERT INTO STUDENTS (ID, first_name, last_name,
                            major, total_current_credits)
      VALUES (p_ID, p_FirstName, p_LastName, p_Major, 0);
END AddNewStudent;
/
```

STUDENTS

id	first_name	last_name	major	total_current_credits

Stored Procedures: Example 3 - exception

```
CREATE PROCEDURE credit_account (acct NUMBER, credit NUMBER) AS

  old_balance NUMBER;
  new_balance NUMBER;

BEGIN
  SELECT balance INTO old_balance
  FROM ACCOUNTS
  WHERE acct_id = acct;

  new_balance := old_balance + credit;

  UPDATE ACCOUNTS
  SET balance = new_balance
  WHERE acct_id = acct;

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
         INSERT INTO ACCOUNTS (acct_id, balance)
         VALUES (acct, credit);
END credit_account;
/
```

ACCOUNTS

acct_id	acct_name	balance
1001	Kim K	200
1002	Kanye W	500
1003	Beyonce K	1000

execute credit_account ('1001', 500);

execute credit_account ('1004', 500);

Stored Procedures: Cursors

```
CREATE OR REPLACE PROCEDURE salCheck(MinSalary
number) AS

   CURSOR executive IS
      SELECT eName, eSalary
      FROM EMPLOYEE
      WHERE eSalary > MinSalary;

 BEGIN
   FOR v_cursrec IN executive LOOP

      dbms_output.put_line
         (v_cursrec.eName||' '||
              v_cursrec.eSalary);
   END LOOP;

END salCheck;
/
```

EMPLOYEE

eName	eSalary
Donnie	2000
Robert	3000
Julio	4000

execute salCheck (2000);

executive

eName	eSalary
Robert	3000
Julio	4000

Stored Procedures: Cursors

```
CREATE OR REPLACE PROCEDURE salCheck(MinSalary
number) AS

   CURSOR executive IS
      SELECT eName, eSalary
      FROM EMPLOYEE;

 BEGIN
   FOR v_cursrec IN executive LOOP

      IF (v_cursrec.eSalary > MinSalary) THEN

       dbms_output.put_line
          (v_cursrec.eName||' '||
               v_cursrec.eSalary);

      END IF;

   END LOOP;

END salCheck;
/
```

EMPLOYEE

eName	eSalary
Donnie	2000
Robert	3000
Julio	4000

execute salCheck (2000);

executive

eName	eSalary
Robert	3000
Julio	4000

Stored Procedures: Cursors

Insert into TEMP_TABLE, students name and the total credits only for students who have completed more than 300 credit points.

```
CREATE OR REPLACE PROCEDURE CheckStudentCompletion AS
  CURSOR c_student IS
      SELECT id, last_name, total_current_credits
      FROM STUDENTS;
BEGIN

FOR v_StudentRecord IN c_student LOOP
 IF (v_StudentRecord.total_current_credit > 300) THEN
          INSERT INTO TEMP_TABLE (desc_col)
          VALUES (v_StudentRecord.id || v_StudentRecord.last_name ||
                    ' final semester student!');
 END IF;
END LOOP;

END CheckStudentCompletion;
/
```

STUDENTS

id	last_name	total_current_credits
100	Doe	200
200	Wood	320
300	Nguyen	320
400	Perez	360

Stored Function

- *A function is very similar to a procedure, however, a procedure call is a PL/SQL statement by itself, while a function call is called as **part of an expression**. The RETURN statement is used to return control to the calling environment with a value.*

```
CREATE OR REPLACE FUNCTION ClassInfo (
  /* Returns 'Full' if the class is completely full,
     'Some Room' if the class is over 80% full,
     'More Room' if the class is over 60% full,
     'Lots of Room' if the class is less than 60% full, and
     'Empty' if there are no students registered. */
  p_Department classes.department%TYPE,
  p_Course     classes.course%TYPE)
  RETURN VARCHAR2 IS

  v_CurrentStudents NUMBER;
  v_MaxStudents     NUMBER;
  v_PercentFull     NUMBER;

BEGIN
        << Function Body is in the NEXT SLIDE >>
END ClassInfo;
/
```

Stored Function (ctd.)

```
BEGIN
  -- Get the current and maximum students for the requested
  -- course.
  SELECT current_students, max_students
    INTO v_CurrentStudents, v_MaxStudents
    FROM CLASSES
    WHERE department = p_Department
    AND course = p_Course;

  -- Calculate the current percentage.
  v_PercentFull := (v_CurrentStudents / v_MaxStudents) * 100;

IF v_PercentFull = 100 THEN
    RETURN 'Full';
  ELSIF v_PercentFull > 80 THEN
    RETURN 'Some Room';
  ELSIF v_PercentFull > 60 THEN
    RETURN 'More Room';
  ELSIF v_PercentFull > 0 THEN
    RETURN 'Lots of Room';
  ELSE
    RETURN 'Empty';
  END IF;
END ClassInfo;
/
```

CLASSES

Department	Course	Current_ Students	Max_ Students
CSCE	BIT	180	200
CSCE	BCS	50	50
Maths	BSc	40	60
Physics	BSc	0	20

Stored Function (ctd.)

```
SELECT department, course, classinfo(department, course)
FROM classes;
```

Department	Course	Classinfo(department, course)
CSCE	BIT	Some Room
CSCE	BCS	Full
Maths	BSc	More Room
Physics	BSc	Empty

Stored Function (ctd.)

CLASSES

Department	Course	Current_Students	Max_ Students
CSCE	BIT	180	200
CSCE	BCS	50	50
Maths	BSc	40	60
Physics	BSc	0	20

C_CLASSES

Department	Course
CSCE	BIT
CSCE	BCS
Maths	BSc
Physics	BSc