***Example 1-1 PL/SQL Block Structure***

DECLARE -- Declarative part (optional)

-- Declarations of local types, variables, & subprograms

BEGIN -- Executable part (required)

-- Statements (which can use items declared in declarative part)

[EXCEPTION -- Exception-handling part (optional)

-- Exception handlers for exceptions raised in executable part]

END;

***Example 1-2 PL/SQL Variable Declarations***

SQL> DECLARE

2 part\_number NUMBER(6); -- SQL data type

3 part\_name VARCHAR2(20); -- SQL data type

4 in\_stock BOOLEAN; -- PL/SQL-only data type

5 part\_price NUMBER(6,2); -- SQL data type

6 part\_description VARCHAR2(50); -- SQL data type

7 BEGIN

8 NULL;

9 END;

10 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-3 Assigning Values to Variables with the Assignment Operator***

SQL> DECLARE -- You can assign values here

2 wages NUMBER;

3 hours\_worked NUMBER := 40;

4 hourly\_salary NUMBER := 22.50;

5 bonus NUMBER := 150;

6 country VARCHAR2(128);

7 counter NUMBER := 0;

8 done BOOLEAN;

9 valid\_id BOOLEAN;

10 emp\_rec1 employees%ROWTYPE;

11 emp\_rec2 employees%ROWTYPE;

12 TYPE commissions IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

13 comm\_tab commissions;

14

15 BEGIN -- You can assign values here too

16 wages := (hours\_worked \* hourly\_salary) + bonus;

17 country := 'France';

18 country := UPPER('Canada');

19 done := (counter > 100);

20 valid\_id := TRUE;

21 emp\_rec1.first\_name := 'Antonio';

22 emp\_rec1.last\_name := 'Ortiz';

23 emp\_rec1 := emp\_rec2;

24 comm\_tab(5) := 20000 \* 0.15;

25 END;

26 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-4 Using SELECT INTO to Assign Values to Variables***

SQL> DECLARE

2 bonus NUMBER(8,2);

3 emp\_id NUMBER(6) := 100;

4 BEGIN

5 **SELECT salary \* 0.10 INTO bonus**

6 **FROM employees**

7 **WHERE employee\_id = emp\_id;**

8 END;

9 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-5 Assigning Values to Variables as Parameters of a Subprogram***

SQL> DECLARE

2 **new\_sal NUMBER(8,2);**

3 emp\_id NUMBER(6) := 126;

4

5 PROCEDURE adjust\_salary (

6 emp\_id NUMBER,

7 **sal IN OUT NUMBER**

8 ) IS

9 emp\_job VARCHAR2(10);

10 avg\_sal NUMBER(8,2);

11 BEGIN

12 SELECT job\_id INTO emp\_job

13 FROM employees

14 WHERE employee\_id = emp\_id;

15

16 SELECT AVG(salary) INTO avg\_sal

17 FROM employees

18 WHERE job\_id = emp\_job;

19

20 DBMS\_OUTPUT.PUT\_LINE ('The average salary for '

21 || emp\_job

22 || ' employees: '

23 || TO\_CHAR(avg\_sal)

24 );

25

26 **sal := (sal + avg\_sal)/2;**

27 END;

28

29 BEGIN

30 SELECT AVG(salary) INTO new\_sal

31 FROM employees;

32

33 DBMS\_OUTPUT.PUT\_LINE ('The average salary for all employees: '

34 || TO\_CHAR(new\_sal)

35 );

36

37 **adjust\_salary(emp\_id, new\_sal);**

38 END;

39 /

The average salary for all employees: 6461.68

The average salary for ST\_CLERK employees: 2785

PL/SQL procedure successfully completed.

SQL>

***Example 1-6 Using %ROWTYPE with an Explicit Cursor***

SQL> DECLARE

2 CURSOR **c1** IS

3 SELECT last\_name, salary, hire\_date, job\_id

4 FROM employees

5 WHERE employee\_id = 120;

6

7 **employee\_rec c1%ROWTYPE;**

8

9 BEGIN

10 OPEN c1;

11 **FETCH c1 INTO employee\_rec;**

12 DBMS\_OUTPUT.PUT\_LINE('Employee name: ' || employee\_rec.last\_name);

13 END;

14 /

Employee name: Weiss

PL/SQL procedure successfully completed.

SQL>

***Example 1-7 Using a PL/SQL Collection Type***

SQL> DECLARE

2 **TYPE staff\_list IS TABLE OF employees.employee\_id%TYPE;**

3 **staff staff\_list;**

4 lname employees.last\_name%TYPE;

5 fname employees.first\_name%TYPE;

6 BEGIN

7 **staff := staff\_list(100, 114, 115, 120, 122);**

8

9 FOR i IN staff.FIRST..staff.LAST LOOP

10 SELECT last\_name, first\_name INTO lname, fname

11 FROM employees

12 WHERE employees.employee\_id = **staff(i)**;

13

14 DBMS\_OUTPUT.PUT\_LINE (TO\_CHAR(**staff(i)**)

15 || ': '

16 || lname

17 || ', '

18 || fname

19 );

20 END LOOP;

21 END;

22 /

100: King, Steven

114: Raphaely, Den

115: Khoo, Alexander

120: Weiss, Matthew

122: Kaufling, Payam

PL/SQL procedure successfully completed.

SQL>

***Example 1-8 Declaring a Record Type***

SQL> DECLARE

2 TYPE timerec IS RECORD (

3 hours SMALLINT,

4 minutes SMALLINT

5 );

6

7 TYPE meeting\_type IS RECORD (

8 date\_held DATE,

9 duration timerec, -- nested record

10 location VARCHAR2(20),

11 purpose VARCHAR2(50)

12 );

13

14 BEGIN

15 NULL;

16 END;

17 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-9 Defining an Object Type***

SQL> CREATE TYPE bank\_account AS OBJECT (

2 acct\_number NUMBER(5),

3 balance NUMBER,

4 status VARCHAR2(10),

5

6 MEMBER PROCEDURE open

7 (SELF IN OUT NOCOPY bank\_account,

8 amount IN NUMBER),

9

10 MEMBER PROCEDURE close

11 (SELF IN OUT NOCOPY bank\_account,

12 num IN NUMBER,

13 amount OUT NUMBER),

14

15 MEMBER PROCEDURE deposit

16 (SELF IN OUT NOCOPY bank\_account,

17 num IN NUMBER,

18 amount IN NUMBER),

19

20 MEMBER PROCEDURE withdraw

21 (SELF IN OUT NOCOPY bank\_account,

22 num IN NUMBER,

23 amount IN NUMBER),

24

25 MEMBER FUNCTION curr\_bal (num IN NUMBER) RETURN NUMBER

26 );

27 /

Type created.

SQL>

***Example 1-10 Using the IF-THEN-ELSE and CASE Statement for Conditional Control***

SQL> DECLARE

2 jobid employees.job\_id%TYPE;

3 empid employees.employee\_id%TYPE := 115;

4 sal employees.salary%TYPE;

5 sal\_raise NUMBER(3,2);

6 BEGIN

7 SELECT job\_id, salary INTO jobid, sal

8 FROM employees

9 WHERE employee\_id = empid;

10

11 CASE

12 WHEN jobid = 'PU\_CLERK' THEN

13 IF sal < 3000 THEN

14 sal\_raise := .12;

15 ELSE

16 sal\_raise := .09;

17 END IF;

18

19 WHEN jobid = 'SH\_CLERK' THEN

20 IF sal < 4000 THEN

21 sal\_raise := .11;

22 ELSE

23 sal\_raise := .08;

24 END IF;

25

26 WHEN jobid = 'ST\_CLERK' THEN

27 IF sal < 3500 THEN

28 sal\_raise := .10;

29 ELSE

30 sal\_raise := .07;

31 END IF;

32

33 ELSE

34 BEGIN

35 DBMS\_OUTPUT.PUT\_LINE('No raise for this job: ' || jobid);

36 END;

37 END CASE;

38

39 UPDATE employees

40 SET salary = salary + salary \* sal\_raise

41 WHERE employee\_id = empid;

42 END;

43 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-11 Using the FOR-LOOP***

SQL> CREATE TABLE sqr\_root\_sum (

2 num NUMBER,

3 sq\_root NUMBER(6,2),

4 sqr NUMBER,

5 sum\_sqrs NUMBER

6 );

Table created.

SQL>

SQL> DECLARE

2 s PLS\_INTEGER;

3 BEGIN

4 **FOR i in 1..100 LOOP**

5 s := (i \* (i + 1) \* (2\*i +1)) / 6; -- sum of squares

6

7 INSERT INTO sqr\_root\_sum

8 VALUES (i, SQRT(i), i\*i, s );

9 **END LOOP;**

10 END;

11 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-12 Using WHILE-LOOP for Control***

SQL> CREATE TABLE temp (

2 tempid NUMBER(6),

3 tempsal NUMBER(8,2),

4 tempname VARCHAR2(25)

5 );

Table created.

SQL>

SQL> DECLARE

2 sal employees.salary%TYPE := 0;

3 mgr\_id employees.manager\_id%TYPE;

4 lname employees.last\_name%TYPE;

5 starting\_empid employees.employee\_id%TYPE := 120;

6

7 BEGIN

8 SELECT manager\_id INTO mgr\_id

9 FROM employees

10 WHERE employee\_id = starting\_empid;

11

12 **WHILE sal <= 15000 LOOP**

13 SELECT salary, manager\_id, last\_name INTO sal, mgr\_id, lname

14 FROM employees

15 WHERE employee\_id = mgr\_id;

16 **END LOOP;**

17

18 INSERT INTO temp

19 VALUES (NULL, sal, lname);

20

21 EXCEPTION

22 WHEN NO\_DATA\_FOUND THEN

23 INSERT INTO temp VALUES (NULL, NULL, 'Not found');

24 END;

25 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-13 Using the EXIT-WHEN Statement***

SQL> CREATE TABLE temp (

2 tempid NUMBER(6),

3 tempsal NUMBER(8,2),

4 tempname VARCHAR2(25)

5 );

Table created.

SQL>

SQL> DECLARE

2 total NUMBER(9) := 0;

3 counter NUMBER(6) := 0;

4 BEGIN

5 **LOOP**

6 counter := counter + 1;

7 total := total + counter \* counter;

8 **EXIT WHEN total > 25000;**

9 **END LOOP;**

10

11 DBMS\_OUTPUT.PUT\_LINE ('Counter: '

12 || TO\_CHAR(counter)

13 || ' Total: '

14 || TO\_CHAR(total)

15 );

16 END;

17 /

Counter: 42 Total: 25585

PL/SQL procedure successfully completed.

SQL>

***Example 1-14 Using the GOTO Statement***

SQL> DECLARE

2 total NUMBER(9) := 0;

3 counter NUMBER(6) := 0;

4 BEGIN

5 **<<calc\_total>>**

6 counter := counter + 1;

7 total := total + counter \* counter;

8

9 IF total > 25000 THEN

10 **GOTO print\_total;**

11 ELSE

12 **GOTO calc\_total;**

13 END IF;

14

15 **<<print\_total>>**

16 DBMS\_OUTPUT.PUT\_LINE

17 ('Counter: ' || TO\_CHAR(counter) || ' Total: ' || TO\_CHAR(total));

18 END;

19 /

Counter: 42 Total: 25585

PL/SQL procedure successfully completed.

SQL>

***Example 1-15 PL/SQL Procedure***

SQL> DECLARE

2 in\_string VARCHAR2(100) := 'Test string';

3 out\_string VARCHAR2(200);

4

5 **PROCEDURE double (**

6 **original IN VARCHAR2,**

7 **new\_string OUT VARCHAR2**

8 **) AS**

9 **BEGIN**

10 **new\_string := original || original;**

11 **END;**

12

13 BEGIN

14 DBMS\_OUTPUT.PUT\_LINE ('in\_string: ' || in\_string);

15 **double (in\_string, out\_string);**

16 DBMS\_OUTPUT.PUT\_LINE ('out\_string: ' || out\_string);

17 END;

18 /

in\_string: Test string

out\_string: Test stringTest string

PL/SQL procedure successfully completed.

SQL>

***Example 1-16 Creating a Standalone PL/SQL Procedure***

SQL> CREATE OR REPLACE PROCEDURE award\_bonus (

2 emp\_id NUMBER, bonus NUMBER) AS

3 commission REAL;

4 comm\_missing EXCEPTION;

5 BEGIN

6 SELECT commission\_pct / 100 INTO commission

7 FROM employees

8 WHERE employee\_id = emp\_id;

9

10 IF commission IS NULL THEN

11 RAISE comm\_missing;

12 ELSE

13 UPDATE employees

14 SET salary = salary + bonus\*commission

15 WHERE employee\_id = emp\_id;

16 END IF;

17 EXCEPTION

18 WHEN comm\_missing THEN

19 DBMS\_OUTPUT.PUT\_LINE

20 ('This employee does not receive a commission.');

21 commission := 0;

22 WHEN OTHERS THEN

23 NULL;

24 END award\_bonus;

25 /

Procedure created.

SQL>

***Example 1-17 Invoking a Standalone Procedure from SQL\*Plus***

SQL> -- Invoke standalone procedure with CALL statement

SQL>

SQL> CALL award\_bonus(179, 1000);

Call completed.

SQL>

SQL> -- Invoke standalone procedure from within block

SQL>

SQL> BEGIN

2 award\_bonus(179, 10000);

3 END;

4 /

PL/SQL procedure successfully completed.

SQL>

***Example 1-18 Creating a Trigger***

SQL> CREATE TABLE emp\_audit (

2 emp\_audit\_id NUMBER(6),

3 up\_date DATE,

4 new\_sal NUMBER(8,2),

5 old\_sal NUMBER(8,2)

6 );

Table created.

SQL>

SQL> **CREATE OR REPLACE TRIGGER audit\_sal**

2 **AFTER UPDATE OF salary**

3 **ON employees**

4 **FOR EACH ROW**

5 **BEGIN**

6 **INSERT INTO emp\_audit**

7 **VALUES(:old.employee\_id, SYSDATE, :new.salary, :old.salary);**

8 **END;**

9 /

Trigger created.

SQL>

***Example 1-19 Creating a Package and Package Body***

SQL> **-- Package specification:**

SQL>

SQL> CREATE OR REPLACE PACKAGE emp\_actions AS

2

3 PROCEDURE hire\_employee (

4 employee\_id NUMBER,

5 last\_name VARCHAR2,

6 first\_name VARCHAR2,

7 email VARCHAR2,

8 phone\_number VARCHAR2,

9 hire\_date DATE,

10 job\_id VARCHAR2,

11 salary NUMBER,

12 commission\_pct NUMBER,

13 manager\_id NUMBER,

14 department\_id NUMBER

15 );

16

17 PROCEDURE fire\_employee (emp\_id NUMBER);

18

19 FUNCTION num\_above\_salary (emp\_id NUMBER) RETURN NUMBER;

20 END emp\_actions;

21 /

Package created.

SQL> **-- Package body:**

SQL>

SQL> CREATE OR REPLACE PACKAGE BODY emp\_actions AS

2

3 -- Code for procedure hire\_employee:

4

5 PROCEDURE hire\_employee (

6 employee\_id NUMBER,

7 last\_name VARCHAR2,

8 first\_name VARCHAR2,

9 email VARCHAR2,

10 phone\_number VARCHAR2,

11 hire\_date DATE,

12 job\_id VARCHAR2,

13 salary NUMBER,

14 commission\_pct NUMBER,

15 manager\_id NUMBER,

16 department\_id NUMBER

17 ) IS

18 BEGIN

19 INSERT INTO employees

20 VALUES (employee\_id,

21 last\_name,

22 first\_name,

23 email,

24 phone\_number,

25 hire\_date,

26 job\_id,

27 salary,

28 commission\_pct,

29 manager\_id,

30 department\_id);

31 END hire\_employee;

32

33 -- Code for procedure fire\_employee:

34

35 PROCEDURE fire\_employee (emp\_id NUMBER) IS

36 BEGIN

37 DELETE FROM employees

38 WHERE employee\_id = emp\_id;

39 END fire\_employee;

40

41 -- Code for function num\_above\_salary:

42

43 FUNCTION num\_above\_salary (emp\_id NUMBER) RETURN NUMBER IS

44 emp\_sal NUMBER(8,2);

45 num\_count NUMBER;

46 BEGIN

47 SELECT salary INTO emp\_sal

48 FROM employees

49 WHERE employee\_id = emp\_id;

50

51 SELECT COUNT(\*) INTO num\_count

52 FROM employees

53 WHERE salary > emp\_sal;

54

55 RETURN num\_count;

56 END num\_above\_salary;

57 END emp\_actions;

58 /

Package body created.

SQL>

***Example 1-20 Invoking a Procedure in a Package***

SQL> CALL **emp\_actions.hire\_employee** (300, 'Belden', 'Enrique',

2 'EBELDEN', '555.111.2222',

3 '31-AUG-04', 'AC\_MGR', 9000,

4 .1, 101, 110);

Call completed.

SQL> BEGIN

2 DBMS\_OUTPUT.PUT\_LINE

3 ('Number of employees with higher salary: ' ||

4 TO\_CHAR(emp\_actions.num\_above\_salary(120)));

5

6 **emp\_actions.fire\_employee**(300);

7 END;

8 /

Number of employees with higher salary: 34

PL/SQL procedure successfully completed.

SQL>

***Example 1-21 Processing Query Results in a LOOP***

SQL> BEGIN

2 FOR someone IN (SELECT \* FROM employees WHERE employee\_id < 120)

3 LOOP

4 DBMS\_OUTPUT.PUT\_LINE('First name = ' || someone.first\_name ||

5 ', Last name = ' || someone.last\_name);

6 END LOOP;

7 END;

8 /

First name = Steven, Last name = King

First name = Neena, Last name = Kochhar

First name = Lex, Last name = De Haan

First name = Alexander, Last name = Hunold

First name = Bruce, Last name = Ernst

First name = David, Last name = Austin

First name = Valli, Last name = Pataballa

First name = Diana, Last name = Lorentz

First name = Nancy, Last name = Greenberg

First name = Daniel, Last name = Faviet

First name = John, Last name = Chen

First name = Ismael, Last name = Sciarra

First name = Jose Manuel, Last name = Urman

First name = Luis, Last name = Popp

First name = Den, Last name = Raphaely

First name = Alexander, Last name = Khoo

First name = Shelli, Last name = Baida

First name = Sigal, Last name = Tobias

First name = Guy, Last name = Himuro

First name = Karen, Last name = Colmenares

PL/SQL procedure successfully completed.

SQL>