**Week 10: PL/SQL Advanced Solution**

This time we are again going to use HR database for answering following questions

**Procedure**

1. Modify the following to a procedure. Execute and invoke the procedure. Then Drop the procedure.

DECLARE

v\_dept\_name departments.department\_name%TYPE:= 'Education';

v\_dept\_id NUMBER;

v\_max\_deptno NUMBER;

BEGIN

SELECT MAX(department\_id) INTO v\_max\_deptno FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is : ' ||

v\_max\_deptno);

v\_dept\_id := 10 + v\_max\_deptno;

INSERT INTO departments (department\_id, department\_name, location\_id)

VALUES (v\_dept\_id,v\_dept\_name, NULL);

DBMS\_OUTPUT.PUT\_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);

END;

/

1. Modify the procedure created above so that it takes a parameter (v\_name). Print this parameter with ‘Hello’. Create an anonymous block to invoke greet procedure.

CREATE PROCEDURE greet IS

v\_today DATE:=SYSDATE;

v\_tomorrow v\_today%TYPE;

BEGIN

v\_tomorrow:=v\_today +1;

DBMS\_OUTPUT.PUT\_LINE(' Hello World ');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS : '|| v\_today);

DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS : ' || v\_tomorrow);

END;

1. Create, compile and invoke a procedure called ADD\_JOB to insert a new job into the JOBS table. Provide ID and job title using two parameters. Check the procedure using IT\_DBA as Job ID and DATABASE ADMIN as job title.

CREATE OR REPLACE PROCEDURE add\_job (

p\_jobid jobs.job\_id%TYPE,

p\_jobtitle jobs.job\_title%TYPE) IS

BEGIN

INSERT INTO jobs (job\_id, job\_title)

VALUES (p\_jobid, p\_jobtitle);

COMMIT;

END add\_job;

EXECUTE add\_job ('IT\_DBA', 'Database Administrator')

SELECT \* FROM jobs WHERE job\_id = 'IT\_DBA';

1. Create a procedure called UPD\_JOB to insert a new job into the JOBS table. Provide ID and job title using two parameters. Handle exception when job id not found.

CREATE OR REPLACE PROCEDURE upd\_job(

p\_jobid IN jobs.job\_id%TYPE,

p\_jobtitle IN jobs.job\_title%TYPE) IS

BEGIN

UPDATE jobs

SET job\_title = p\_jobtitle

WHERE job\_id = p\_jobid;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202, 'No job updated.');

END IF;

END upd\_job;

/

1. Create a procedure that returns a value from the SALARY and JOB\_ID columns for a specified employee ID. Execute the procedure using host variables with two OUT parameters-one for salary and other for job ID.

CREATE OR REPLACE PROCEDURE get\_employee

(p\_empid IN employees.employee\_id%TYPE,

p\_sal OUT employees.salary%TYPE,

p\_job OUT employees.job\_id%TYPE) IS

BEGIN

SELECT salary, job\_id

INTO p\_sal, p\_job

FROM employees

WHERE employee\_id = p\_empid;

END get\_employee;

**Function**

1. Create a function GET\_JOB that take jobid as parameter and return a job title.

CREATE OR REPLACE FUNCTION get\_job (p\_jobid IN jobs.job\_id%type)

RETURN jobs.job\_title%type IS

v\_title jobs.job\_title%type;

BEGIN

SELECT job\_title

INTO v\_title

FROM jobs

WHERE job\_id = p\_jobid;

RETURN v\_title;

END get\_job;

1. Create a function that return annual salary of an employee and take salary and commission as input using following formula:

[Salary\*12+commission=annual salary]

CREATE OR REPLACE FUNCTION get\_annual\_comp(

p\_sal IN employees.salary%TYPE,

p\_comm IN employees.commission\_pct%TYPE)

RETURN NUMBER IS

BEGIN

RETURN (NVL(p\_sal,0) \* 12 + (NVL(p\_comm,0) \* nvl(p\_sal,0) \* 12));

END get\_annual\_comp;

/

1. Use the above function in a select statement against Employees table for employees in department 30.

SELECT employee\_id, last\_name,

get\_annual\_comp(salary,commission\_pct) "Annual Compensation"

FROM employees

WHERE department\_id=30

/

1. Create a function that validate a particular department id and return true if department exist otherwise false. Create a procedure that add an employee to Employees table. Row should be only added if function for validation return true, otherwise alert the user with an appropriate message. Call the procedure with the name “Jane Harris” in department 18 and email jaharris.

CREATE OR REPLACE FUNCTION valid\_deptid(

p\_deptid IN departments.department\_id%TYPE)

RETURN BOOLEAN IS

v\_dummy PLS\_INTEGER;

BEGIN

SELECT 1

INTO v\_dummy

FROM departments

WHERE department\_id = p\_deptid;

RETURN TRUE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END valid\_deptid;

CREATE OR REPLACE PROCEDURE add\_employee(

p\_first\_name employees.first\_name%TYPE,

p\_last\_name employees.last\_name%TYPE,

p\_email employees.email%TYPE,

p\_job employees.job\_id%TYPE DEFAULT 'SA\_REP',

p\_mgr employees.manager\_id%TYPE DEFAULT 145,

p\_sal employees.salary%TYPE DEFAULT 1000,

p\_comm employees.commission\_pct%TYPE DEFAULT 0,

p\_deptid employees.department\_id%TYPE DEFAULT 30) IS

BEGIN

IF valid\_deptid(p\_deptid) THEN

INSERT INTO employees(employee\_id, first\_name, last\_name, email,

job\_id, manager\_id, hire\_date, salary, commission\_pct, department\_id)

VALUES (employees\_seq.NEXTVAL, p\_first\_name, p\_last\_name, p\_email,

p\_job, p\_mgr, TRUNC(SYSDATE), p\_sal, p\_comm, p\_deptid);

ELSE

RAISE\_APPLICATION\_ERROR (-20204, 'Invalid department ID. Try again.');

END IF;

END add\_employee;

/

EXECUTE add\_employee('Jane', 'Harris', 'JAHARRIS', p\_deptid=> 15)