PL/SQL and Triggers Exercise

Slide10.sql

If you have a table called employees in your database, Drop it.
Redirect the output by typing in: SET SERVEROUTPUT ON;

Then set up the following table

```
CREATE TABLE employees
(
employee_id     NUMBER(6),
salary          NUMBER(8,2),
job_id          VARCHAR(8),
PRIMARY KEY (employee_id)
);

INSERT INTO employees VALUES (100, 200.50, 'PU_CLERK');
INSERT INTO employees VALUES (115, 200.50, 'PU_CLERK');
INSERT INTO employees VALUES (200, 400.50, 'PU_CLERK');
```

Now enter the following PL/SQL statement. Remember to include the / which will execute it.

```
DECLARE
  bonus   NUMBER(8,2);
BEGIN
  SELECT salary * 0.10 INTO bonus
   FROM employees
     WHERE employee_id = 100;
  DBMS_OUTPUT.PUT_LINE('The bonus for employee 100 is: ' || bonus);
END;
```

The procedure is in the buffer, but it won't execute until you enter the forward slash.

```
/
```

Slide11.sql

This procedure also uses the employee table.

```
DECLARE
   job_var        employees.job_id%TYPE;
   sal_var        employees.salary%TYPE;
   sal_raise      NUMBER(3,2);

BEGIN
  SELECT job_id, salary INTO job_var, sal_var
    FROM employees
      WHERE employee_id = 115;

  CASE
    WHEN job_var = 'PU_CLERK' THEN
      IF sal_var < 3000 THEN
        sal_raise := 0.12;
      ELSE
        sal_raise := 0.09;
      END IF;

    ELSE
      BEGIN
        DBMS_OUTPUT.PUT_LINE('No raise for this job: ' || job_var);
      END;
   END CASE;

   UPDATE employees
     SET salary = salary + salary * sal_raise
       WHERE employee_id = 115;
END;
```

Don't forget to enter the forward slash to execute the procedure /

Check that the value of salary for employee 115 has increased by entering a select query.

Slide12.sql

Then set up the following table

```
CREATE TABLE sqr_root_sum
 (
  num       NUMBER,
  sq_root   NUMBER(6,2),
  sqr       NUMBER,
  sum_sqrs  NUMBER
);
```

Now enter the following PL/SQL statement.

```
DECLARE
   s_var  PLS_INTEGER;
BEGIN
  FOR i in 1..100
    LOOP
      s_var := (i * (i + 1) * (2*i +1)) / 6;  -- sum of squares

      INSERT INTO sqr_root_sum VALUES (i, SQRT(i), i*i, s_var );
    END LOOP;
END;
```

Remember to include the / which will execute it.

Enter a select query to see the contents of the table.

Slide13.sql

This example doesn't use a database table. It demonstrates the use of a named PL/SQL procedure.

```
DECLARE
  in_var   INTEGER(3) := 25;
  out_var  INTEGER(3);

  PROCEDURE double ( original IN INTEGER, new_var OUT INTEGER )
  AS
  BEGIN
    new_var := original + original;
  END;

BEGIN
  DBMS_OUTPUT.PUT_LINE ('in_string: ' || in_var);
  double (in_var, out_var);
  DBMS_OUTPUT.PUT_LINE ('out_string: ' || out_var);
END;
```

Slide17.sql

```
CREATE OR REPLACE TRIGGER Print_salary_changes
  BEFORE DELETE OR INSERT OR UPDATE ON employees
  FOR EACH ROW
DECLARE
    sal_diff    NUMBER;
BEGIN
    sal_diff  := :NEW.salary  - :OLD.salary;
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT('Old salary: ' || :OLD.salary);
    DBMS_OUTPUT.PUT('  New salary: ' || :NEW.salary);
    DBMS_OUTPUT.PUT_LINE ('  Difference ' || sal_diff);
END;
/
```

Change all of the salaries and observe the output:

```
UPDATE employees SET salary = salary * 4;
```

Slide18.sql

Set up the following table

```
CREATE TABLE emp_audit
(
  emp_audit_id  NUMBER(6),
  up_date       DATE,
  new_sal       NUMBER(8,2),
  old_sal       NUMBER(8,2)
);
```

We will now set up a trigger on the table employees.

```
CREATE TRIGGER audit_sal
  AFTER UPDATE OF salary
    ON employees
      FOR EACH ROW
BEGIN
  INSERT INTO emp_audit
    VALUES(:old.employee_id, SYSDATE, :new.salary, :old.salary);
END;
/
```

Now make a couple of changes to the employees table.

```
UPDATE employees
     SET salary = salary * 2  WHERE employee_id = 115;

UPDATE employees
     SET salary = salary * 4 WHERE employee_id = 200;
```

Now enter this query to display contents of the table emp_audit

```
SELECT
   emp_audit_id, new_sal, old_sal,
                 TO_CHAR( up_date, 'DD/MM/YYYY HH:MM:SS') AS timestamp
FROM emp_audit;
```

Slide28.sql

This example also makes use of the employees table. Delete the records and insert a new one.

```
DELETE FROM employees;

INSERT INTO employees VALUES (44, 300.00, 'PU_CLERK');
```

Now set up a trigger on the table employees.

```
CREATE OR REPLACE TRIGGER Salary_check
     BEFORE INSERT OR UPDATE OF Salary ON employees
FOR EACH ROW
DECLARE
  Minsal               NUMBER(8,2) := 100.00;
  Maxsal               NUMBER(8,2) := 500.00;
  Salary_out_of_range  EXCEPTION;

BEGIN
  IF (:NEW.Salary < Minsal OR :NEW.Salary > Maxsal) THEN
           RAISE Salary_out_of_range;
  END IF;

EXCEPTION
  WHEN Salary_out_of_range THEN
      Raise_application_error (-20300, 'Salary ' ||TO_CHAR(:NEW.Salary)
             ||' out of range for employee '||TO_CHAR(:NEW.Employee_id));

  WHEN NO_DATA_FOUND THEN
      Raise_application_error(-20322,
        'Invalid Job Classification ' ||:NEW.Job_id);
  END;
/
```

Now make the following changes:

```
UPDATE employees SET Salary = 1000.00;
```