DBS 501 NSA – Assignment 2

**Group #3: 72/80**

**Ting-Yeh Lin 141726182**

**Yajian Yu 125884171**

**Qianwen Sun 103335188**

**Xiuqiang Li 140242181**

**Date:**

**Jun 22nd, 2021**

* **Q1**

**Code:**

SET SERVEROUTPUT ON;

SET VERIFY OFF;

CREATE OR REPLACE PROCEDURE modify\_sal(

v\_dep\_id employees.department\_id%TYPE

)

IS

v\_avg employees.salary%TYPE;

v\_increased\_amount employees.salary%TYPE;

v\_count NUMBER := 0;

v\_rows NUMBER;

CURSOR c1 IS

SELECT e.salary, e.first\_name, e.last\_name, e.employee\_id

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

unfound EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO v\_rows

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_rows > 0 THEN

SELECT AVG(e.salary) INTO v\_avg

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_avg > 0 THEN

FOR i IN c1 LOOP

EXIT WHEN c1%NOTFOUND;

IF i.salary < v\_avg THEN

v\_increased\_amount := v\_avg - i.salary;

UPDATE employees

SET salary = v\_avg

WHERE employee\_id = i.employee\_id;

DBMS\_OUTPUT.PUT\_LINE('Employee ' || i.first\_name ||' ' || i.last\_name ||' just got an increase of $' || v\_increased\_amount);

v\_count := v\_count + 1;

END IF;

END LOOP;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No salary was modified in Department: ' || v\_dep\_id);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Total # of employees who received salary increase is: ' || v\_count);

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('This Department is EMPTY: ' || v\_dep\_id);

END IF;

ELSE

RAISE unfound;

END IF;

EXCEPTION

WHEN unfound THEN

DBMS\_OUTPUT.PUT\_LINE('This Department Id is invalid: ' || v\_dep\_id);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('There are some other errors. ');

END modify\_sal;

/

EXECUTE modify\_sal(99);

EXECUTE modify\_sal(190);

EXECUTE modify\_sal(10);

EXECUTE modify\_sal(110);

ROLLBACK;

EXECUTE modify\_sal(60);

ROLLBACK;

**-- Cursor needs FOR UPDATE OF salary clause**

**-- Logic of IF .. ELSE IF .. IF .. is convoluted, should use ELSIF form**

**-- UPDATE needs WHERE CURRENT OF clause**

**Output:**

Procedure MODIFY\_SAL compiled

1):

This Department ID is invalid: 99

PL/SQL procedure successfully completed.

2):

This Department is EMPTY: 190

PL/SQL procedure successfully completed.

3):

No salary was modified in Department: 10

PL/SQL procedure successfully completed.

4):

Employee William Gietz just got an increase of $1850

Total # of employees who received salary increase is: 1

PL/SQL procedure successfully completed.

Rollback complete.

5):

Employee David Austin just got an increase of $960

Employee Valli Pataballa just got an increase of $960

Employee Diana Lorentz just got an increase of $1560

Total # of employees who received salary increase is: 3

PL/SQL procedure successfully completed.

Rollback complete.

* **Q2:**

**Code:**

CREATE OR REPLACE FUNCTION Total\_Cost (

v\_st\_ID student.student\_id%TYPE

)

RETURN NUMBER

IS

v\_count NUMBER;

v\_cost NUMBER;

BEGIN

SELECT COUNT(\*), NVL(SUM(c.cost),0) INTO v\_count, v\_cost

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.student\_id = v\_st\_ID;

IF v\_count = 0 THEN

v\_cost := -1;

END IF;

RETURN v\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_cost := -1;

RETURN v\_cost;

END Total\_Cost;

/

SET AUTOPRINT ON

VARIABLE COST NUMBER;

EXECUTE :COST := Total\_Cost(194);

EXECUTE :COST := Total\_Cost(294);

EXECUTE :COST := Total\_Cost(494);

/

**-- It is Not neded to JOIN with Students table**

**-- Also, neither JOIN is Left Join, just Inner Joins**

**Output:**

Function TOTAL\_COST compiled

PL/SQL procedure successfully completed.

COST

----

1195

PL/SQL procedure successfully completed.

COST

-

0

PL/SQL procedure successfully completed.

COST

--

-1

* **Q3:**

**Code:**

SET SERVEROUTPUT ON;

SET VERIFY OFF;

CREATE OR REPLACE PACKAGE My\_pack AS

PROCEDURE modify\_sal(v\_dep\_id employees.department\_id%TYPE);

FUNCTION Total\_Cost(v\_st\_ID student.student\_id%TYPE)

RETURN NUMBER;

END My\_pack ;

/

CREATE OR REPLACE PACKAGE BODY My\_pack AS

PROCEDURE modify\_sal(

v\_dep\_id employees.department\_id%TYPE

)

IS

v\_avg employees.salary%TYPE;

v\_increased\_amount employees.salary%TYPE;

v\_count NUMBER := 0;

v\_rows NUMBER;

CURSOR c1

IS

SELECT e.salary, e.first\_name, e.last\_name, e.employee\_id

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

unfound EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO v\_rows

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_rows > 0 THEN

SELECT AVG(e.salary) INTO v\_avg

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_avg > 0 THEN

FOR i IN c1 LOOP

EXIT WHEN c1%NOTFOUND;

IF i.salary < v\_avg THEN

v\_increased\_amount := v\_avg - i.salary;

UPDATE employees

SET salary = v\_avg

WHERE employee\_id = i.employee\_id;

DBMS\_OUTPUT.PUT\_LINE('Employee ' || i.first\_name ||' ' || i.last\_name ||' just got an increase of $' || v\_increased\_amount);

v\_count := v\_count + 1;

END IF;

END LOOP;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No salary was modified in Department: ' || v\_dep\_id);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Total # of employees who received salary increase is: ' || v\_count);

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('This Department is EMPTY: ' || v\_dep\_id);

END IF;

ELSE

RAISE unfound;

END IF;

EXCEPTION

WHEN unfound THEN

DBMS\_OUTPUT.PUT\_LINE('This Department Id is invalid: ' || v\_dep\_id);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('There are some other errors. ');

END modify\_sal;

FUNCTION Total\_Cost(

v\_st\_ID student.student\_id%TYPE

)

RETURN NUMBER

IS

v\_count NUMBER;

v\_cost NUMBER;

BEGIN

SELECT COUNT(\*), NVL(SUM(c.cost),0) INTO v\_count, v\_cost

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.student\_id = v\_st\_ID;

IF v\_count = 0 THEN

v\_cost := -1;

END IF;

RETURN v\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_cost := -1;

RETURN v\_cost;

END Total\_Cost;

END My\_pack;

/

SET AUTOPRINT ON

VARIABLE COST NUMBER;

EXECUTE :COST := My\_pack.Total\_Cost(194);

EXECUTE :COST := My\_pack.Total\_Cost(294);

EXECUTE :COST := My\_pack.Total\_Cost(494);

**Output:**

Package MY\_PACK compiled

Package Body MY\_PACK compiled

PL/SQL procedure successfully completed.

COST

----

1195

PL/SQL procedure successfully completed.

COST

-

0

PL/SQL procedure successfully completed.

COST

--

-1

* **Q4:**

**Code:**

CREATE OR REPLACE PACKAGE My\_pack

IS

PROCEDURE modify\_sal(v\_dep\_id employees.department\_id%TYPE);

FUNCTION Total\_Cost(v\_st\_ID student.student\_id%TYPE)

RETURN NUMBER;

FUNCTION Total\_Cost(v\_fname student.first\_name%TYPE,v\_lname student.last\_name%TYPE)

RETURN NUMBER;

FUNCTION Total\_Cost(v\_zip student.zip%TYPE)

RETURN NUMBER;

END My\_pack;

/

CREATE OR REPLACE PACKAGE BODY My\_pack

IS

PROCEDURE modify\_sal(

v\_dep\_id employees.department\_id%TYPE

)

IS

v\_avg employees.salary%TYPE;

v\_increased\_amount employees.salary%TYPE;

v\_count NUMBER := 0;

v\_rows NUMBER;

CURSOR c1

IS

SELECT e.salary, e.first\_name, e.last\_name, e.employee\_id

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

unfound EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO v\_rows

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_rows > 0 THEN

SELECT AVG(e.salary) INTO v\_avg

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id = v\_dep\_id;

IF v\_avg > 0 THEN

FOR i IN c1 LOOP

EXIT WHEN c1%NOTFOUND;

IF i.salary < v\_avg THEN

v\_increased\_amount := v\_avg - i.salary;

UPDATE employees

SET salary = v\_avg

WHERE employee\_id = i.employee\_id;

DBMS\_OUTPUT.PUT\_LINE('Employee ' || i.first\_name ||' ' || i.last\_name ||' just got an increase of $' || v\_increased\_amount);

v\_count := v\_count + 1;

END IF;

END LOOP;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No salary was modified in Department: ' || v\_dep\_id);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Total # of employees who received salary increase is: ' || v\_count);

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('This Department is EMPTY: ' || v\_dep\_id);

END IF;

ELSE

RAISE unfound;

END IF;

EXCEPTION

WHEN unfound THEN

DBMS\_OUTPUT.PUT\_LINE('This Department Id is invalid: ' || v\_dep\_id);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('There are some other errors. ');

END modify\_sal;

FUNCTION Total\_Cost(

v\_st\_ID student.student\_id%TYPE

)

RETURN NUMBER

IS

v\_count NUMBER;

v\_cost NUMBER;

BEGIN

SELECT COUNT(\*), NVL(SUM(c.cost),0) INTO v\_count, v\_cost

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.student\_id = v\_st\_ID;

IF v\_count = 0 THEN

v\_cost := -1;

END IF;

RETURN v\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_cost := -1;

RETURN v\_cost;

END Total\_Cost;

FUNCTION Total\_Cost(

v\_fname student.first\_name%TYPE,

v\_lname student.last\_name%TYPE

)

RETURN NUMBER

IS

v\_count NUMBER;

v\_cost NUMBER;

BEGIN

SELECT COUNT(\*), NVL(SUM(c.cost),0) INTO v\_count, v\_cost

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.first\_name = INITCAP(v\_fname) AND s.last\_name = INITCAP(v\_lname);

IF v\_count = 0 THEN

v\_cost := -1;

END IF;

RETURN v\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_cost := -1;

RETURN v\_cost;

END Total\_Cost;

FUNCTION Total\_Cost(

v\_zip student.zip%TYPE

)

RETURN NUMBER

IS

v\_count NUMBER;

v\_cost course.cost%TYPE := 0;

CURSOR c1 IS

SELECT SUM(c.cost) cost

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.zip = v\_zip;

BEGIN

SELECT COUNT(\*) INTO v\_count

FROM student s

LEFT JOIN enrollment e ON s.student\_id = e.student\_id

LEFT JOIN section se ON e.section\_id = se.section\_id

LEFT JOIN course c ON se.course\_no = c.course\_no

WHERE s.zip = v\_zip;

IF v\_count > 0 THEN

FOR i IN c1 LOOP

v\_cost := v\_cost + i.cost;

END LOOP;

ELSE

v\_cost := -1;

END IF;

RETURN v\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_cost := -1;

RETURN v\_cost;

END Total\_Cost;

END My\_pack;

/

SHOW ERRORS;

VARIABLE COST NUMBER

**-- SELECT with Multiple Joins is Not needed, just Open cursor and test for %NOTFOUND (and figure out is the zip area empty)**

EXECUTE :COST := My\_pack.Total\_Cost('VERONA','GRANT');

PRINT COST

EXECUTE :COST := My\_pack.Total\_Cost('YVONNE','WINNICKI');

PRINT COST

EXECUTE :COST := My\_pack.Total\_Cost('PETER','PAN');

PRINT COST

EXECUTE :COST := My\_pack.Total\_Cost('07044');

PRINT COST

EXECUTE :COST := My\_pack.Total\_Cost('11209');

PRINT COST

EXECUTE :COST := My\_pack.Total\_Cost('11111');

PRINT COST

**Output:**

Package MY\_PACK compiled

Package Body MY\_PACK compiled

PL/SQL procedure successfully completed.

COST

----------

1195

PL/SQL procedure successfully completed.

COST

----------

0

PL/SQL procedure successfully completed.

COST

----------

-1

PL/SQL procedure successfully completed.

COST

----------

1195

PL/SQL procedure successfully completed.

COST

----------

7070

PL/SQL procedure successfully completed.

COST

----------

-1