**DBS501: ASSIGNMENT 2**

**TO: NEBOJSA CONKIC**

**BY: ANTHONY NGUYEN (101727154)**

**DATE: Jan. 2, 18**

**QUESTION 1:**

--create procedure modify\_sal with input param department\_id

CREATE OR REPLACE PROCEDURE modify\_sal(p\_department\_id NUMBER) IS

v\_theSelectedDepartment NUMBER(20);

v\_num\_emp NUMBER(20);

v\_avg\_sal NUMBER(20);

v\_count NUMBER(20) := 0;

v\_sal\_calc NUMBER(20);

empty\_dept EXCEPTION;

BEGIN

--store selected department

SELECT department\_id INTO v\_theSelectedDepartment FROM departments

WHERE department\_id = p\_department\_id;

--count total num of employees in selected department

SELECT COUNT(\*) INTO v\_num\_emp FROM employees

WHERE department\_id = p\_department\_id;

--exception if department\_id is invalid

IF v\_num\_emp = 0 THEN

RAISE empty\_dept;

END IF;

--determine average salary from the selected department

SELECT AVG(salary) INTO v\_avg\_sal FROM employees

WHERE department\_id = p\_department\_id;

--Cursor For Loop to scan each employees's salary

For i IN (SELECT \* FROM employees WHERE department\_id = p\_department\_id FOR UPDATE NOWAIT)

--check if it is smaller than average amount earned in the department

LOOP

--if YES, adjust his/her salary to that average amount

IF i.salary < v\_avg\_sal THEN

v\_sal\_calc := v\_avg\_sal - i.salary;

UPDATE employees

SET salary = v\_avg\_sal

WHERE employee\_id = i.employee\_id;

--count # of employees who received a salary increase

v\_count := v\_count + 1;

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

END IF;

END LOOP;

--print results of the for cursor loop

IF v\_count > 0 THEN

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

ELSE

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

END IF;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

WHEN empty\_dept THEN

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

END modify\_sal;

*--TEST CASES; used the same code but changed the numbers to match the desired values (99,190,10,110,60)--*

SET SERVEROUTPUT ON;

SET VERIFY OFF;

EXECUTE modify\_sal(10);

ROLLBACK;

No salary was modified in Department: 10

This Department Id is invalid: 99

This Department is EMPTY: 190

No salary was modified in Department: 10

Employee William Gietzjust got an increase of $1850

Total # of employees who received salary increase is: 1

Employee David Austinjust got an increase of $960

Employee Valli Pataballajust got an increase of $960

Employee Diana Lorentzjust got an increase of $1560

Total # of employees who received salary increase is: 3

**QUESTION 2:**

--create function called total\_cost that takes a student id

--and returns the total cost for all enrolled courses

CREATE OR REPLACE FUNCTION Total\_Cost (p\_student\_id IN VARCHAR2)

RETURN NUMBER IS v\_total\_costs NUMBER;

v\_student\_id NUMBER;

v\_num\_courses NUMBER;

e\_invalid EXCEPTION;

BEGIN

--store the selected student

SELECT student\_id INTO v\_student\_id

FROM student

WHERE student\_id = p\_student\_id;

--count and store the number of enrolled courses for specific student

SELECT COUNT(student\_id) INTO v\_num\_courses

FROM enrollment

WHERE student\_id = p\_student\_id;

--test function for valid and invalid inputs

IF v\_num\_courses = 0 THEN

RAISE e\_invalid;

ELSE

--calculate the grand total of enrolled courses for selected student

SELECT SUM(c.cost) AS INTO v\_total\_costs

FROM enrollment e

INNER JOIN section S ON e.section\_id = s.section\_id

INNER JOIN course c ON s.course\_no = c.course\_no

WHERE e.student\_id = p\_student\_id;

--return the grand total

RETURN v\_total\_costs;

END IF;

--exception if invalid id(return -1) or NO courses(return 0)

EXCEPTION

WHEN e\_invalid THEN RETURN 0;

WHEN NO\_DATA\_FOUND THEN RETURN -1;

END Total\_Cost;

*--TEST CASES; used the same code but changed the numbers to match the desired values (194, 294, 494)--*

SET SERVEROUTPUT ON

SET VERIFY OFF

ACCEPT b\_student\_id NUMBER PROMPT ‘Please enter a Student ID: ‘;

SELECT Total\_Cost(:b\_student\_id) AS COST FROM DUAL;

COST

1195

COST

0

COST

-1

**QUESTION 3:**

**A)**

CREATE OR REPLACE PACKAGE My\_pack AS

PROCEDURE modify\_sal(p\_department\_id NUMBER);

FUNCTION Total\_Cost (p\_student\_id VARCHAR2) RETURN NUMBER;

END MY\_PACK;

**B)**

CREATE OR REPLACE PACKAGE BODY my\_pack AS

PROCEDURE modify\_sal (p\_department\_id NUMBER) IS

v\_count NUMBER(20) := 0;

v\_sal\_avg NUMBER(20);

v\_dept NUMBER(20);

v\_dept\_emp NUMBER(20);

v\_calculation NUMBER(20);

invalid\_dept EXCEPTION;

BEGIN

SELECT department\_id INTO v\_dept FROM departments

WHERE department\_id = p\_department\_id;

SELECT COUNT(\*)

INTO v\_dept\_emp FROM employees

WHERE department\_id = p\_department\_id;

IF v\_dept\_emp = 0 THEN

RAISE invalid\_dept;

END IF;

SELECT AVG(salary) INTO v\_sal\_avg FROM employees

WHERE department\_id = p\_department\_id;

FOR i IN ( SELECT \* FROM employees

WHERE department\_id = p\_department\_id

FOR UPDATE NOWAIT )

LOOP

IF i.salary < v\_sal\_avg THEN

v\_calculation := v\_sal\_avg - i.salary;

UPDATE employees

SET salary = v\_sal\_avg

WHERE employee\_id = i.employee\_id;

v\_count := v\_count + 1;

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

END IF;

END LOOP;

IF v\_count > 0 THEN

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

ELSE

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

END IF;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

WHEN invalid\_dept THEN

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

END modify\_sal;

FUNCTION TOTAL\_COST (p\_student\_id IN VARCHAR2)

RETURN NUMBER IS v\_total\_cost NUMBER;

v\_student\_id NUMBER;

v\_courses NUMBER;

e\_invalid EXCEPTION;

BEGIN

SELECT student\_id INTO v\_student\_id FROM student

WHERE student\_id = p\_student\_id;

SELECT COUNT(student\_id) INTO v\_courses FROM enrollment

WHERE student\_id = p\_student\_id;

IF v\_courses = 0 THEN

RAISE e\_invalid;

ELSE

SELECT SUM(C.COST) AS INTO v\_total\_cost FROM enrollment e

INNER JOIN section s ON e.section\_id = s.section\_id

INNER JOIN course c ON s.course\_no = c.course\_no

WHERE e.student\_id = p\_student\_id;

RETURN v\_total\_cost;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN -1;

WHEN e\_invalid THEN RETURN 0;

END total\_cost;

END my\_pack;

**C)**

SET SERVEROUTPUT ON

SET VERIFY OFF

SELECT My\_Pack.Total\_Cost(:student\_id) AS COST FROM DUAL;

**QUESTION 4:**

--Overload package with two new variations

CREATE OR REPLACE PACKAGE My\_pack IS

--1st will accept two input param(first, last names) and return cost

FUNCTION Total\_Cost(p\_firstname VARCHAR2, p\_lastname VARCHAR2)

RETURN NUMBER;

--2nd will accept one input param(zip code) and return cost for all students in zip

FUNCTION Total\_Cost(p\_zip VARCHAR2)

RETURN NUMBER;

END my\_pack;

/

CREATE OR REPLACE PACKAGE BODY My\_pack IS

--1st function

FUNCTION TOTAL\_COST(p\_firstname VARCHAR2, p\_lastname VARCHAR2)

RETURN NUMBER

IS

v\_student NUMBER;

v\_stud\_enroll NUMBER;

v\_total\_cost NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_student FROM student

WHERE UPPER(first\_name) = UPPER(p\_firstname) AND UPPER(last\_name) = UPPER(p\_lastname);

IF v\_student = 0 THEN

RETURN -1;

ELSE

SELECT COUNT(\*) INTO v\_stud\_enroll FROM enrollment

WHERE student\_id = (SELECT student\_id FROM STUDENT

WHERE UPPER(first\_name) = UPPER(p\_firstname) AND UPPER(last\_name) = UPPER(p\_lastname));

IF v\_stud\_enroll = 0 THEN

RETURN 0;

ELSE

SELECT SUM(c.cost) INTO v\_total\_cost FROM enrollment e INNER JOIN section s ON s.section\_id = e.section\_id INNER JOIN course c ON c.course\_no = s.course\_no

WHERE e.student\_id = (SELECT student\_id FROM student WHERE UPPER(first\_name) = UPPER(p\_firstname) AND UPPER(last\_name) = UPPER(p\_lastname));

RETURN v\_total\_cost;

END IF;

END IF;

END Total\_Cost;

--2nd function

FUNCTION Total\_Cost(p\_zip VARCHAR2)

RETURN NUMBER

IS

v\_zip NUMBER;

v\_cost\_stud NUMBER;

v\_total\_cost NUMBER := 0;

CURSOR c\_student\_cursor IS

SELECT \* FROM student

WHERE UPPER(zip) = UPPER(p\_zip);

BEGIN

SELECT COUNT(\*) INTO v\_zip FROM ZIPCODE WHERE zip = p\_zip;

IF v\_zip = 0 THEN

RETURN -1;

ELSE

FOR stud IN c\_student\_cursor

LOOP

v\_cost\_stud := Total\_Cost(stud.first\_name, stud.last\_name);

v\_total\_cost := v\_total\_cost + v\_cost\_stud;

END LOOP;

RETURN v\_total\_cost;

END IF;

END Total\_Cost;

END My\_pack;

/

**QUESTION 5:**

--procedure mod\_grade accepts two input param(course num, grade)

--finds all students in all sections of course and modifies final grade

--to provided value

CREATE OR REPLACE PROCEDURE mod\_grade(p\_courseno NUMBER,p\_grade NUMBER)

IS

v\_course NUMBER;

v\_total\_enrolled NUMBER;

v\_counter NUMBER := 0;

--cursor to find all students in all sections of a course

CURSOR c\_student IS

SELECT \* FROM enrollment

WHERE section\_id IN (SELECT section\_id FROM section

WHERE course\_no = p\_courseno);

BEGIN

--count courses

SELECT COUNT(\*) INTO v\_course FROM course

WHERE course\_no = p\_courseno;

--check if invalid course

IF v\_course = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('This Course Number is invalid: ' || p\_courseno);

ELSE

--check if invalid grade

IF p\_grade > 100 OR p\_grade < 0 THEN

DBMS\_OUTPUT.PUT\_LINE('This Grade invalid: ' || p\_grade ||' It must be between 0 and 100.');

ELSE

--store number of enrolled students with specific course num

SELECT COUNT(\*) INTO v\_total\_enrolled FROM enrollment

WHERE section\_id IN (SELECT section\_id FROM section

WHERE course\_no = p\_courseno);

--provide warning if nobody is enrolled in course

IF v\_total\_enrolled = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('This Course has NOBODY enrolled so far: ' || p\_courseno);

ELSE

--update grade to newly desired mark

FOR i IN c\_student

LOOP

UPDATE enrollment SET final\_grade = p\_grade

WHERE student\_id = i.student\_id AND section\_id = i.section\_id;

--count number of records being updated

v\_counter := v\_counter + 1;

END LOOP;

--print total number of records updated

DBMS\_OUTPUT.PUT\_LINE('Total # of grades changed to ' || p\_grade ||' for course number ' || p\_courseno || ' is ' || v\_counter);

END IF;

END IF;

END IF;

END MOD\_GRADE;

*--TEST CASES; used the same code but changed the numbers to match the desired values (130,75)(144,75)(99,75)(130,101)--*

SET SERVEROUTPUT ON;

SET VERIFY OFF;

EXECUTE mod\_grade(130,75);

ROLLBACK;

Total # of grades changed to 75 for course number 130 is 8

This Course has NOBODY enrolled so far: 144

This Course Number is invalid: 99

This Grade invalid: 101 It must be between 0 and 100.