**Update student procedure**

CREATE OR REPLACE PROCEDURE UPDATE\_STUDENT (

p\_student\_id IN NUMBER,

p\_student\_fname IN VARCHAR2,

p\_student\_lname IN VARCHAR2,

p\_email IN VARCHAR2,

p\_department\_id IN NUMBER

) IS

BEGIN

UPDATE CASE.STUDENTS

SET STUDENT\_FNAME = p\_student\_fname,

STUDENT\_LNAME = p\_student\_lname,

EMAIL = p\_email,

DEPARTMENT\_ID = p\_department\_id

WHERE STUDENT\_ID = p\_student\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END UPDATE\_STUDENT;

--procedure call

select \* from students ;

DECLARE

v\_student\_id NUMBER := 1;

v\_student\_fname VARCHAR2(50) := 'malak';

v\_student\_lname VARCHAR2(50) := 'sherif';

v\_email VARCHAR2(50) := 'malak@gamil';

v\_department\_id NUMBER := 1;

BEGIN

UPDATE\_STUDENT(

p\_student\_id => v\_student\_id,

p\_student\_fname => v\_student\_fname,

p\_student\_lname => v\_student\_lname,

p\_email => v\_email,

p\_department\_id => v\_department\_id

);

END;

select \* from students ;

**Student GBA function**

CREATE OR REPLACE FUNCTION CASE.caseGBA(

vStudentId IN NUMBER

) RETURN NUMBER

AS

vTotalScore NUMBER := 0;

vTotalMaxGrade NUMBER := 0;

vGPA NUMBER;

BEGIN

SELECT

NVL(SUM(g.student\_scored\_grade), 0),

NVL(SUM(c.max\_grade), 0)

INTO vTotalScore, vTotalMaxGrade

FROM grades g

JOIN courses c ON g.course\_id = c.course\_id

WHERE g.student\_id = vStudentId;

vGPA := NVL((vTotalScore / NULLIF(vTotalMaxGrade, 0)) \* 4.0, NULL);

vGPA := ROUND(vGPA, 2);

DBMS\_OUTPUT.PUT\_LINE('GPA calculated successfully.');

RETURN vGPA;

END caseGBA;

**Courses GBA function**

CREATE OR REPLACE FUNCTION CASE.calculate\_course\_gba(

v\_course\_id IN NUMBER

) RETURN NUMBER

AS

v\_avg\_score NUMBER := 0;

v\_avg\_max\_grade NUMBER := 0;

v\_course\_gba NUMBER;

BEGIN

SELECT

NVL(AVG(g.student\_scored\_grade), 0),

NVL(AVG(c.max\_grade), 0)

INTO v\_avg\_score, v\_avg\_max\_grade

FROM grades g

JOIN courses c ON g.course\_id = c.course\_id

WHERE g.course\_id = v\_course\_id;

v\_course\_gba := NVL((v\_avg\_score / NULLIF(v\_avg\_max\_grade, 0)) \* 4.0, NULL);

v\_course\_gba := ROUND(v\_course\_gba, 2);

DBMS\_OUTPUT.PUT\_LINE('Course GBA calculated successfully.');

RETURN v\_course\_gba;

END calculate\_course\_gba;

**Grades trigger**

CREATE OR REPLACE TRIGGER trg\_check\_student\_scored\_grade

BEFORE INSERT OR UPDATE ON grades

FOR EACH ROW

DECLARE

v\_max\_grade NUMBER;

BEGIN

SELECT max\_grade INTO v\_max\_grade

FROM courses

WHERE course\_id = :NEW.course\_id;

IF :NEW.student\_scored\_grade > v\_max\_grade THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Student Scored Grade cannot exceed Max Grade for the course.');

END IF;

END;

SHOW ERRORS;

IF v\_avg\_max\_grade > 0 THEN

v\_course\_gba := (v\_avg\_score / v\_avg\_max\_grade) \* 4.0;

ELSE

v\_course\_gba := NULL;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Course GBA calculated successfully.');

RETURN v\_course\_gba;

END calculate\_course\_gba;