

CREATE TABLE COURSES

(

COURSE\_ID NUMBER,

COURSE\_NAME VARCHAR2(100 BYTE) NOT NULL,

DESCRIPTION VARCHAR2(4000 BYTE),

MAX\_GRADE NUMBER NOT NULL,

DEPARTMENT\_ID NUMBER NOT NULL,

DURATION VARCHAR2(100 BYTE)

)

LOGGING

NOCOMPRESS

NOCACHE

NOPARALLEL

MONITORING;

CREATE TABLE DEPARTMENTS

(

DEPARTMENT\_ID NUMBER,

DEPARTMENT\_NAME VARCHAR2(50 BYTE) NOT NULL

)

LOGGING

NOCOMPRESS

NOCACHE

NOPARALLEL

MONITORING;

CREATE TABLE GRADES

(

COURSE\_ID NUMBER,

STUDENT\_ID NUMBER,

STUDENT\_SCORED\_GRADE NUMBER,

ENROLL\_DATE DATE NOT NULL

)

LOGGING

NOCOMPRESS

NOCACHE

NOPARALLEL

MONITORING;

CREATE TABLE STUDENTS

(

STUDENT\_ID NUMBER,

STUDENT\_FNAME VARCHAR2(50 BYTE) NOT NULL,

STUDENT\_LNAME VARCHAR2(50 BYTE) NOT NULL,

EMAIL VARCHAR2(50 BYTE),

DEPARTMENT\_ID NUMBER NOT NULL

)

LOGGING

NOCOMPRESS

NOCACHE

NOPARALLEL

MONITORING;

CREATE UNIQUE INDEX PK\_GRADE ON GRADES

(COURSE\_ID, STUDENT\_ID, ENROLL\_DATE)

LOGGING

NOPARALLEL;

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;

SHOW ERRORS;

CREATE OR REPLACE FUNCTION 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;

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;

/

SHOW ERRORS;

CREATE OR REPLACE FUNCTION 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);

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

RETURN vGPA;

END caseGBA;

/

SHOW ERRORS;

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;

ALTER TABLE COURSES ADD (

PRIMARY KEY

(COURSE\_ID));

ALTER TABLE DEPARTMENTS ADD (

PRIMARY KEY

(DEPARTMENT\_ID));

ALTER TABLE GRADES ADD (

CONSTRAINT PK\_GRADE

PRIMARY KEY

(COURSE\_ID, STUDENT\_ID, ENROLL\_DATE));

ALTER TABLE STUDENTS ADD (

PRIMARY KEY

(STUDENT\_ID));

ALTER TABLE COURSES ADD (

CONSTRAINT FK\_DEPT\_COURSE

FOREIGN KEY (DEPARTMENT\_ID)

REFERENCES DEPARTMENTS (DEPARTMENT\_ID));

ALTER TABLE GRADES ADD (

CONSTRAINT FK\_COURSE\_GRADE

FOREIGN KEY (COURSE\_ID)

REFERENCES COURSES (COURSE\_ID),

CONSTRAINT FK\_STD\_GRADE

FOREIGN KEY (STUDENT\_ID)

REFERENCES STUDENTS (STUDENT\_ID));

ALTER TABLE STUDENTS ADD (

CONSTRAINT FK\_DEPT\_STD1

FOREIGN KEY (DEPARTMENT\_ID)

REFERENCES DEPARTMENTS (DEPARTMENT\_ID));