DATABASE SYSTEMS

Mini Project

Title:

University Management System

Group 26

Names	ID Numbers	Contribution
Suchitra Sahu	2021A8PS2210H	20%
Suchetana Mukherjee	2021AAPS0759H	20%
Ashrita Vaka Naidu	2021A3PS3061H	20%
Shaurya Agarwal	2021AAPS2047H	20%
Aasheel Dave	2021A4PS3094H	20%

TABLES

1. Table: Students

```
SELECT * FROM Students;
```

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
1	John	Doe	20	1
2	Joe	Smith	22	2
3	Bob	Johnson	21	1
4	Alice	Williams	23	2
5	Charlie	Brown	19	3

Download CSV

5 rows selected.

2. Table: Courses

```
SELECT * FROM Courses;
```

COURSEID	COURSENAME	DEPARTMENTID
101	Advanced Mathematics	4
102	Physics	2
103	Computer Science	3
104	History	1
105	Biology	2

Download CSV

5 rows selected.

3. Table: Grades

```
1  SELECT * FROM Grades;
2
```

GRADEID	STUDENTID	COURSEID	GRADE
1	1	101	9.5
2	2	102	9
3	3	103	7.5
4	4	104	8
5	5	105	9.5

Download CSV

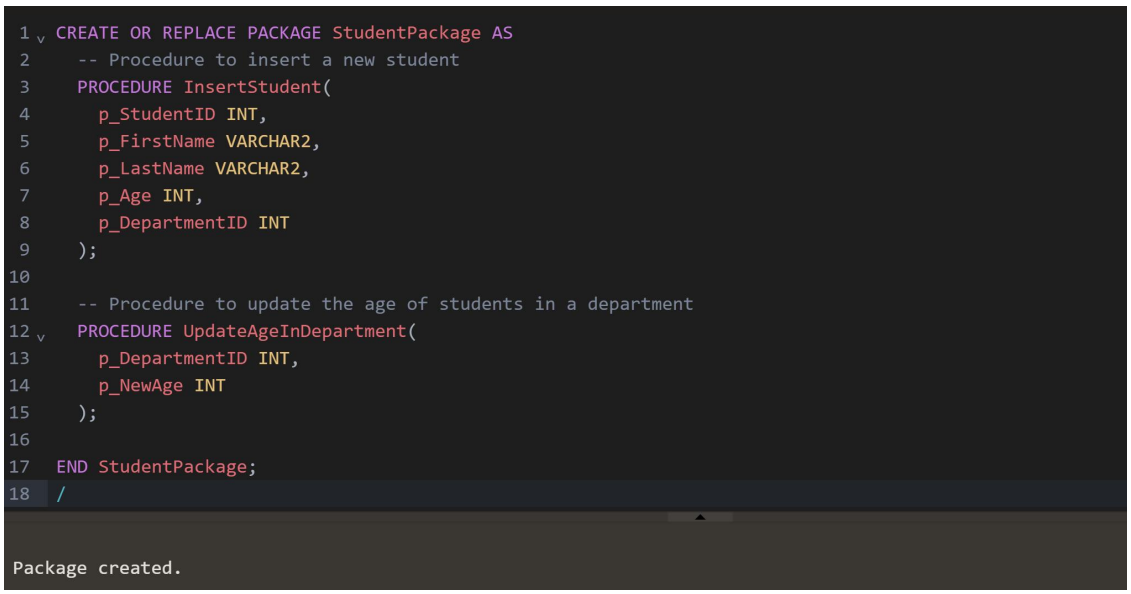
5 rows selected.

Creation of Package:

```
CREATE OR REPLACE PACKAGE StudentPackage AS
    -- Procedure to insert a new student
  PROCEDURE InsertStudent(
      p_StudentID INT,
      p_FirstName VARCHAR2,
      p_LastName VARCHAR2,
      p_Age INT,
      p_DepartmentID INT
  );

  -- Procedure to update the age of students in a department
  PROCEDURE UpdateAgeInDepartment(
      p_DepartmentID INT,
      p_NewAge INT
  );

END StudentPackage;
/
```



CREATION OF 2 PROCEDURES:

Procedure 1: InsertStudent

```
-- Create a package body
CREATE OR REPLACE PACKAGE BODY StudentPackage AS
  -- Procedure to insert a new student   PROCEDURE 1
  PROCEDURE InsertStudent(
    p_StudentID INT,
```

```
    p_FirstName VARCHAR2,
    p_LastName VARCHAR2,
    p_Age INT,
    p_DepartmentID INT
  ) IS
  BEGIN
    -- Insert new student record
    INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
    VALUES (p_StudentID, p_FirstName, p_LastName, p_Age, p_DepartmentID);

    -- Commit the transaction
    COMMIT;
  EXCEPTION
    -- Handle exceptions
    WHEN OTHERS THEN
      -- Rollback the transaction on error
      ROLLBACK;
      -- Raise an application-specific error
      RAISE_APPLICATION_ERROR(-20001, 'Error inserting student. ' ||
SQLERRM);
  END InsertStudent;
```

Procedure 2: UpdateAgeInDepartment

```
  -- Procedure to update the age of students in a department PROCEDURE 2
  PROCEDURE UpdateAgeInDepartment(
    p_DepartmentID INT,
    p_NewAge INT
  ) IS
  BEGIN
    -- Update the age of students in the specified department
    UPDATE Students
    SET Age = p_NewAge
    WHERE DepartmentID = p_DepartmentID;

    -- Commit the transaction
    COMMIT;
  EXCEPTION
    -- Handle exceptions
    WHEN OTHERS THEN
      -- Rollback the transaction on error
      ROLLBACK;
      -- Raise an application-specific error
      RAISE_APPLICATION_ERROR(-20002, 'Error updating age in department. ' ||
SQLERRM);
  END UpdateAgeInDepartment;

END StudentPackage /
```

Output of Package Body Creation along with 2 Procedures:

```
1
2   -- Create a package body
3   CREATE OR REPLACE PACKAGE BODY StudentPackage AS
4     -- Procedure to insert a new student
5     PROCEDURE InsertStudent(
6       p_StudentID INT,
7       p_FirstName VARCHAR2,
8       p_LastName VARCHAR2,
9       p_Age INT,
10      p_DepartmentID INT
11    ) IS
12    BEGIN
13      -- Insert new student record
14      INSERT INTO Students (StudentID, FirstName, LastName, Age, DepartmentID)
15      VALUES (p_StudentID, p_FirstName, p_LastName, p_Age, p_DepartmentID);
16
17      -- Commit the transaction
18      COMMIT;
19    EXCEPTION
20      -- Handle exceptions
21      WHEN OTHERS THEN
22        -- Rollback the transaction on error
23        ROLLBACK;
24        -- Raise an application-specific error
25        RAISE_APPLICATION_ERROR(-20001, 'Error inserting student. ' || SQLERRM);
23          RAISE_APPLICATION_ERROR(-20001, 'Error inserting student. ' || SQLERRM);
24    END InsertStudent;
25    -- Procedure to update the age of students in a department
26    PROCEDURE UpdateAgeInDepartment(
27      p_DepartmentID INT,
28      p_NewAge INT  ) IS
29    BEGIN
30      -- Update the age of students in the specified department
31      UPDATE Students
32      SET Age = p_NewAge
33      WHERE DepartmentID = p_DepartmentID;
34      -- Commit the transaction
35      COMMIT;
36    EXCEPTION
37      -- Handle exceptions
38      WHEN OTHERS THEN
39        -- Rollback the transaction on error
40        ROLLBACK;
41        -- Raise an application-specific error
42        RAISE_APPLICATION_ERROR(-20002, 'Error updating age in department. ' || SQLERRM);
43    END UpdateAgeInDepartment;
44  END StudentPackage;


Package Body created.
```

Please note all the exceptions here which will be resulted in our outputs later.

Creation of TRIGGERS:

A. DML Operation - INSERT:

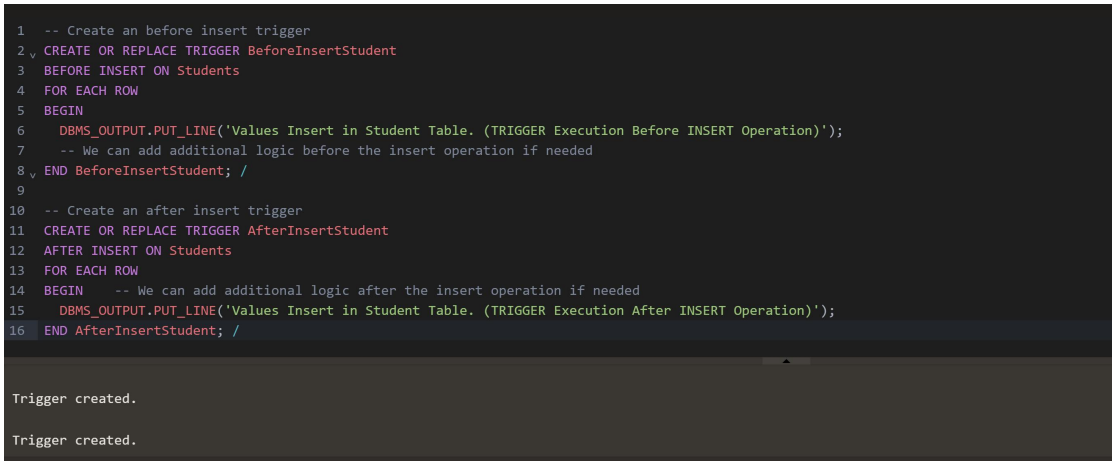
1. Before Insert Trigger

```
-- Create an before insert trigger
CREATE OR REPLACE TRIGGER BeforeInsertStudent
BEFORE INSERT ON Students
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Values Insert in Student Table. (TRIGGER
Execution Before INSERT Operation)');
  -- We can add additional logic before the insert operation if needed
END BeforeInsertStudent;
/
```

2. After Insert Trigger

```
-- Create an after insert trigger
CREATE OR REPLACE TRIGGER AfterInsertStudent
AFTER INSERT ON Students
FOR EACH ROW
BEGIN
  -- We can add additional logic after the insert operation if needed
  DBMS_OUTPUT.PUT_LINE('Values Insert in Student Table. (TRIGGER
Execution After INSERT Operation)');
END AfterInsertStudent;
/
```

Output TRIGGERS -INSERT :



B. DML Operation - UPDATE:

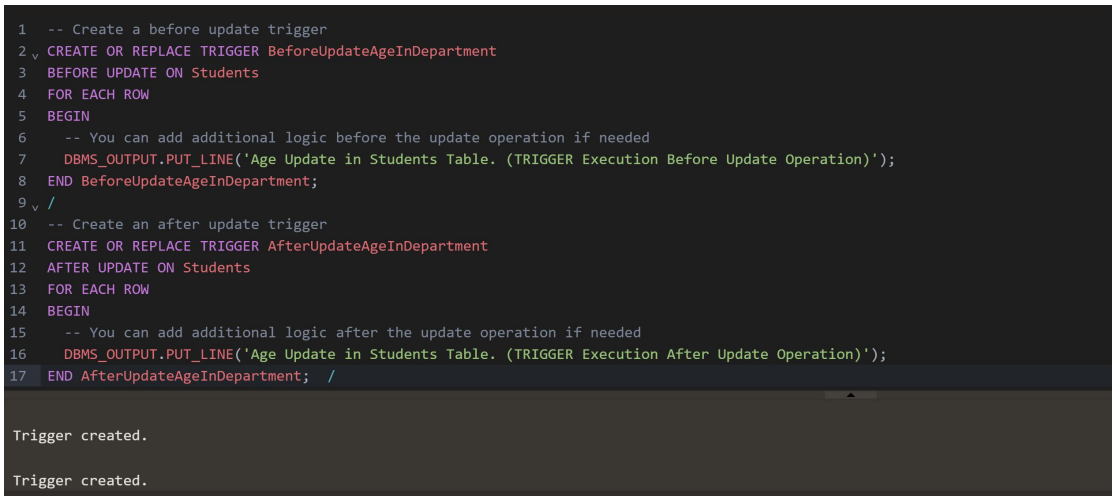
1. Before Update Trigger

```
-- Create a before update trigger
CREATE OR REPLACE TRIGGER BeforeUpdateAgeInDepartment
BEFORE UPDATE ON Students
FOR EACH ROW
BEGIN
  -- You can add additional logic before the update operation if needed
  DBMS_OUTPUT.PUT_LINE('Age Update in Students Table. (TRIGGER
Execution Before Update Operation)');
END BeforeUpdateAgeInDepartment;
/
```

2. After Update Trigger

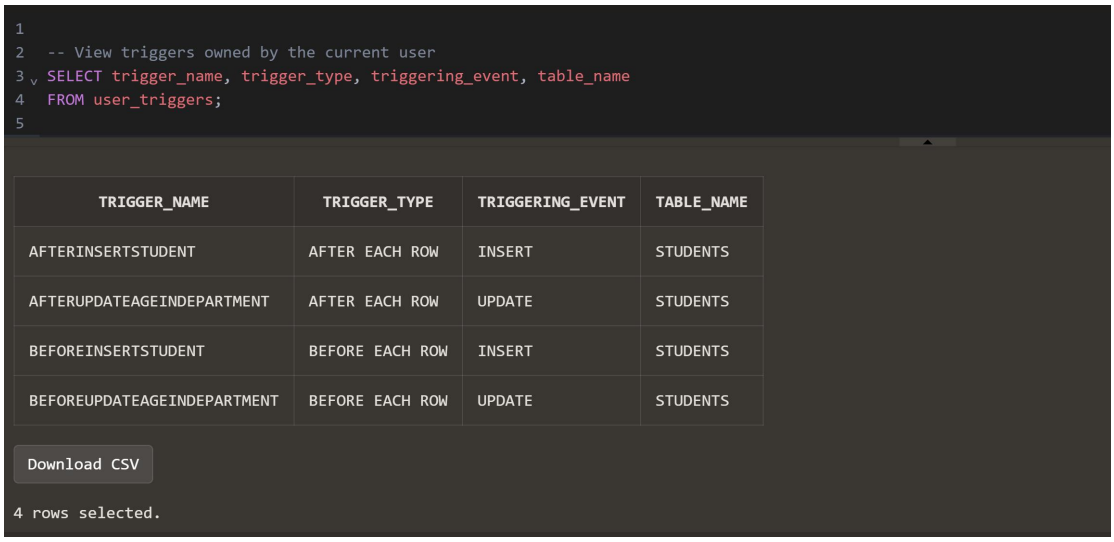
```
-- Create an after update trigger
CREATE OR REPLACE TRIGGER AfterUpdateAgeInDepartment
AFTER UPDATE ON Students
FOR EACH ROW
BEGIN
  -- We can add additional logic after the update operation if needed
  DBMS_OUTPUT.PUT_LINE('Age Update in Students Table. (TRIGGER
Execution After Update Operation)');
END AfterUpdateAgeInDepartment;
/
```

Output TRIGGERS -UPDATE:



Code to View the Triggers and their Outputs:

```
-- View triggers owned by the current user
SELECT trigger_name, trigger_type, triggering_event, table_name
FROM user_triggers;
```



```
1
2  -- View triggers owned by the current user
3  SELECT trigger_name, trigger_type, triggering_event, table_name
4  FROM user_triggers;
5
```

TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_NAME
AFTERINSERTSTUDENT	AFTER EACH ROW	INSERT	STUDENTS
AFTERUPDATEAGEINDEPARTMENT	AFTER EACH ROW	UPDATE	STUDENTS
BEFOREINSERTSTUDENT	BEFORE EACH ROW	INSERT	STUDENTS
BEFOREUPDATEAGEINDEPARTMENT	BEFORE EACH ROW	UPDATE	STUDENTS

Download CSV

4 rows selected.

BEFORE AND AFTER TRIGGER OUTPUTS :

Code for 1st Set of TRIGGERS to be executed: InsertStudent Procedure

```
BEGIN
StudentPackage.InsertStudent (6,'Paul','Miller',25,3);
StudentPackage.InsertStudent (7,'Jack','Anderson',20,1);
StudentPackage.InsertStudent (8,'Hugh','Wilson',22,2);
StudentPackage.InsertStudent (9,'Drew','Thomas',19,3);
StudentPackage.InsertStudent (10,'Luna','Moore',21,1);

END;
/
```

Output of 1ST set of TRIGGERS being executed:

```
BEGIN
StudentPackage.InsertStudent (6,'Paul','Miller',25,3);
StudentPackage.InsertStudent (7,'Jack','Anderson',20,1);
StudentPackage.InsertStudent (8,'Hugh','Wilson',22,2);
StudentPackage.InsertStudent (9,'Drew','Thomas',19,3);
StudentPackage.InsertStudent (10,'Luna','Moore',21,1);
END;
/
```

```
Statement processed.
Insert into Students Table. (TRIGGER Execution Before Insert Operation)
Insert into Students Table. (TRIGGER Execution After Insert Operation)
Insert into Students Table. (TRIGGER Execution Before Insert Operation)
Insert into Students Table. (TRIGGER Execution After Insert Operation)
Insert into Students Table. (TRIGGER Execution Before Insert Operation)
Insert into Students Table. (TRIGGER Execution After Insert Operation)
Insert into Students Table. (TRIGGER Execution Before Insert Operation)
Insert into Students Table. (TRIGGER Execution After Insert Operation)
Insert into Students Table. (TRIGGER Execution Before Insert Operation)
Insert into Students Table. (TRIGGER Execution After Insert Operation)
```

Output: AFTER TRIGGER (1st Procedure):

```
SELECT * FROM Students;
```

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
6	Paul	Miller	25	3
7	Jack	Anderson	20	1
8	Hugh	Wilson	22	2
9	Drew	Thomas	19	3
10	Luna	Moore	21	1
1	John	Doe	20	1
2	Joe	Smith	22	2
3	Bob	Johnson	21	1
4	Alice	Williams	23	2
5	Charlie	Brown	19	3

Table: Students after Insert Operation

Code for 2ND Set of TRIGGERS to be executed:

UpdateAgeInDepartment Procedure

```
--Here we are updating age of students to be 19 where the DepartmentID is 1
BEGIN
StudentPackage.UpdateAgeInDepartment(1,19);
END;
/
```

Output of 2ND set of TRIGGERS being executed:

```
BEGIN
StudentPackage.UpdateAgeInDepartment(1,19);
END;
/


Statement processed.
Age Update in Students Table. (TRIGGER
Execution Before Update Operation)
Age Update in Students Table. (TRIGGER
Execution After Update Operation)
Age Update in Students Table. (TRIGGER
Execution Before Update Operation)
Age Update in Students Table. (TRIGGER
Execution After Update Operation)
Age Update in Students Table. (TRIGGER
Execution Before Update Operation)
Age Update in Students Table. (TRIGGER
Execution After Update Operation)
Age Update in Students Table. (TRIGGER
Execution Before Update Operation)
Age Update in Students Table. (TRIGGER
Execution After Update Operation)
```

Output: AFTER TRIGGER (2nd Procedure) :

```
SELECT * FROM Students;
--Viewing the final table after all Insert and Update Operations
```

STUDENTID	FIRSTNAME	LASTNAME	AGE	DEPARTMENTID
1	John	Doe	19	1
2	Joe	Smith	22	2
3	Bob	Johnson	19	1
4	Alice	Williams	23	2
5	Charlie	Brown	19	3
6	Paul	Miller	25	3
7	Jack	Anderson	19	1
8	Hugh	Wilson	22	2
9	Drew	Thomas	19	3
10	Luna	Moore	19	1

Final Table: Students

Code GitHub Link = https://github.com/APUNJIA/DBMS-Projects-3-1.git