Student Name: Fredrick J. Sigalla

Stud ID: A00277360

Project Name: Irish Dancing Lessons - IDL

Module: Advanced Databases

Course: MSc. In Data Analytics

Date Tested: 13 May 2020

Table of Contents

Question 1. (Before Insert) – Used for Screen Cast

Due to the increased number of reported injuries from the dancers during their lessons, it has been instructed that no new lesson should be added if the material (type) of the venue floor from which the lesson is to be hosted is either unknown or includes concrete.
Write an application that can implicitly implement the new instructions.

Code – Quesiont 1

```
Clear Screen
Set Linesize 160
Set Serveroutput On Size 1000000

Create Or Replace Trigger Lesson_BI
Before Insert On Lesson
For Each Row

Declare
        V_Floor_Type Venue.Floor_Type%Type;

Begin
        Select Floor_Type Into V_Floor_Type
        From Venue
        Where Venue_ID = :New.Venue_ID;

        /*
                concrete and unknown scenarios will be treated separately to give more specific feedback
        */

        -- concrete venue floor
        If V_Floor_Type = 'Concrete' Then
                Raise_Application_Error(-20101, 'The venue floor cannot be concrete, assign the lesson to
                another venue!!!');

        -- unknown material for the venue floor
        Elsif V_Floor_Type Is Null Then
                Raise_Application_Error(-20101, 'The venue floor type must be known! Either update the
                venue details or assign the lesson to a different venue');

        End If;

Exception
        When No_Data_Found Then
                Raise_Application_Error(-20101, 'The venue does not exist!!!');

End;
/
Show err;
```

Validation Test – Question 1

```
-- reference check for the lesson to be added
Select Teacher_ID, Dancer_ID, Venue_ID, Lesson_Date
From Lesson
Where Venue_Id = 20;

-- Running validation tests
/*
	Invalid Test - Case 1: Adding a lesson whose venue is made up of concrete floor
*/
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 13, '13-OCT-19');
/*
	Invalid Test - Case 2: Adding a lesson whose venue floor type is not known
*/
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 16, '13-OCT-19');
/*
	Invalid Test - Case 3: Adding a lesson whose venue does not exist
*/
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 30, '13-OCT-19');
/*
	Valid Test - Adding lesson whose venue floor is now and it is non-concrete floor
*/
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 20, '13-OCT-19');
/*
	- Confirmatory test on the added lesson
	- Compare it to reference check to confirm the newly added lesson
*/
Select Teacher_ID, Dancer_ID, Venue_ID, Lesson_Date
From Lesson
Where Venue_Id = 20;

-- rollback
Rollback;

-- dropping the trigger
Drop Trigger Lesson_BI;
```

Snips – Question 1

```
FS_19SQL>Set Linesize 160
FS_19SQL>Set Serveroutput On Size 1000000
FS_19SQL>
FS_19SQL>Create Or Replace Trigger Lesson_BI
  2  Before Insert On Lesson
  3  For Each Row
  4
  5  Declare
  6  V_Floor_Type Venue.Floor_Type%Type;
  7
  8  Begin
  9  Select Floor_Type Into V_Floor_Type
 10  From Venue
 11  Where Venue_ID = :New.Venue_ID;
 12
 13  /*
 14  concrete and unknown scenarios will be treated separately so as to give more specific feedback
 15  */
 16
 17  -- concrete venue floor
 18  If V_Floor_Type = 'Concrete' Then
 19  Raise_Application_Error(-20101, 'The venue floor can not be concrete, assign the lesson to another venue!!!');
 20
 21  -- unknown material for the venue floor
 22  Elsif V_Floor_Type Is Null Then
 23  Raise_Application_Error(-20101, 'The venue floor type must be known! Either update the venue details or assign the lesson to a different venue');
 24
 25  End If;
 26
 27  Exception
 28  When No_Data_Found Then
 29  Raise_Application_Error(-20101, 'The venue does not exist!!!');
 30
 31  End;
 32  /

Trigger created.

FS_19SQL>Show err;
No errors.
```

```
FS_19SQL>-- reference check for the lesson to be added
FS_19SQL>Select Teacher_ID, Dancer_ID, Venue_ID, Lesson_Date
  2  From Lesson
  3  Where Venue_Id = 20;

TEACHER_ID  DANCER_ID   VENUE_ID LESSON_DA
---------- ---------- ---------- ---------
       520        777         20 26-SEP-19
       590         11         20 21-AUG-10

FS_19SQL>
FS_19SQL>-- Running validation tests
FS_19SQL>/*
FS_19SQL>Invalid Test - Case 1: Adding a lesson whose venue is made up of concrete floor
FS_19SQL>*/
FS_19SQL>Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 13, '13-OCT-19');
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 13, '13-OCT-19')
            *
ERROR at line 1:
ORA-20101: The venue floor can not be concrete, assign the lesson to another venue!!!
ORA-06512: at "STUDENT19.LESSON_BI", line 15
ORA-04088: error during execution of trigger 'STUDENT19.LESSON_BI'


FS_19SQL>/*
FS_19SQL>Invalid Test - Case 2: Adding a lesson whose venue floor type is not known
FS_19SQL>*/
FS_19SQL>Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 16, '13-OCT-19');
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 16, '13-OCT-19')
            *
ERROR at line 1:
ORA-20101: The venue floor type must be known! Either update the venue details or assign the lesson to a different venue
ORA-06512: at "STUDENT19.LESSON_BI", line 19
ORA-04088: error during execution of trigger 'STUDENT19.LESSON_BI'
```

```
FS_19SQL>/*
FS_19SQL>Invalid Test - Case 3: Adding a lesson whose venue does not exist
FS_19SQL>*/
FS_19SQL>Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 30, '13-OCT-19');
Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 30, '13-OCT-19')
            *
ERROR at line 1:
ORA-20101: The venue does not exist!!!
ORA-06512: at "STUDENT19.LESSON_BI", line 25
ORA-04088: error during execution of trigger 'STUDENT19.LESSON_BI'


FS_19SQL>/*
FS_19SQL>Valid Test - Adding lesson whose venue floor is now and it is non-concrete floor
FS_19SQL>*/
FS_19SQL>Insert Into Lesson(Teacher_ID, Dancer_ID, Venue_Id, Lesson_Date) Values(500, 77, 20, '13-OCT-19');

1 row created.

FS_19SQL>/*
FS_19SQL>- Confirmatory test on the added lesson
FS_19SQL>- Compare it to reference check to confirm the newly added lesson
FS_19SQL>*/
FS_19SQL>Select Teacher_ID, Dancer_ID, Venue_ID, Lesson_Date
  2  From Lesson
  3  Where Venue_Id = 20;

TEACHER_ID  DANCER_ID   VENUE_ID LESSON_DA
---------- ---------- ---------- ---------
       500         77         20 13-OCT-19
       520        777         20 26-SEP-19
       590         11         20 21-AUG-10

FS_19SQL>
```

```
FS_19SQL>-- rollback
FS_19SQL>Rollback;

Rollback complete.

FS_19SQL>
FS_19SQL>-- dropping the trigger
FS_19SQL>Drop Trigger Lesson_BI;

Trigger dropped.

FS_19SQL>
```

Question 2. (After Insert, Update)

Write an application that implicitly solve the following challenges on the Irish Dancing Lessons database.

a. Everytime a new teacher is added to the database, in case the specialism of the college the teacher attended is anything other than "Art and Business", then it should be updated to Art and Business.
b. When the teacher's qualification improves to Level 3, then the Teacher Fee on all the lessons he provides should be increased by 1000 Euro.
c. Every event should be logged in the table Teacher_IU_Logs by capturing the triggering event type, date and the name of the oracle user performing the operation

Code – Question 2

```
-- creating the Teacher_IU_Logs table

Create Table Teacher_IU_Logs(Event Varchar2(10), DatePerfomed Date, PerformedBy Varchar2(20));


Clear Screen

Set Linesize 160

Set Serveroutput On Size 1000000


Create Or Replace Trigger Teacher_AIU

After Insert Or Update On Teacher

For Each Row

--When (NVL(New.Qualifications, ' ') <> Old.Qualifications)


Declare

        V_Specialism College.Specialism%Type;

        V_Event_Type Varchar2(20);


Begin

        -- for inserting

        If Inserting Then

                V_Event_Type := 'Insert';


                Select Specialism Into V_Specialism

                From College

                Where College_Id = :New.College_Id;


                If V_Specialism <> 'Art and Business' Then

                        Update College
                Set Specialism = 'Arts'
                Where College_Id = :New.College_Id;
            End If;

        End If;

        -- for updating
        If Updating Then
            V_Event_Type := 'Update';

            If :New.Qualifications = 'Level 3' Then
                Update Lesson
                Set Teacher_Fee = Teacher_Fee + 1000
                Where Teacher_Id = :New.Teacher_Id;
            End If;

        End If;

        -- updating the Teacher_IU_Logs table regardless of the type of triggering event
        Insert Into Teacher_IU_Logs Values(V_Event_Type, SYSDATE, USER);

End;
/
Show err;
```

Validation Test – Question 2

```
-- Running tests


/**
        Case a: Inserting a new teacher whose college is not Arts and Business
*/
--reference check of the table to be updated
Select College_Id, Name, Speciliasm
From College
Where College_Id = 20;

-- inserting a new teacher record which triggers change on the college records
Insert Into Teacher(Teacher_Id, Fname, Sname, College_Id) Values(333, 'Noel', 'Tierney', 20);

-- confirming the changes to college table have taken place
Select College_Id, Name, Speciliasm
From College
Where College_Id = 20;

/*
        Case b: Updating teacher's qualifications to Level 3
*/
-- reference check of the table to be affected by updating teacher's qualifications
Select Teacher_id, Teacher_Fee
From Lesson
Where Teacher_Id = 590;

-- updating teacher's qualifications to Level 3
Update Teacher
Set Qualifications = 'Level 3'
Where Teacher_Id = 590;

-- confirming change's to the teacher fee on the lesson table
Select Teacher_id, Teacher_Fee
From Lesson
```

Where Teacher_Id = 590;

/*

Case c: Confirming all events have been logged into the teacher_iu_log

*/

Select * From Teacher_IU_Logs;

-- Rollback

Rollback;

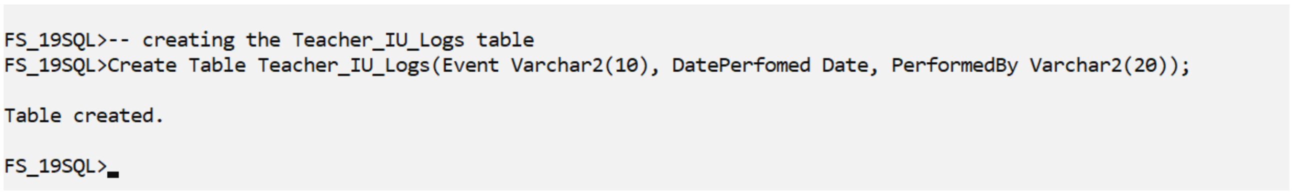
-- Dropping Teacher_IU_Logs

Drop Table Teacher_IU_Logs;

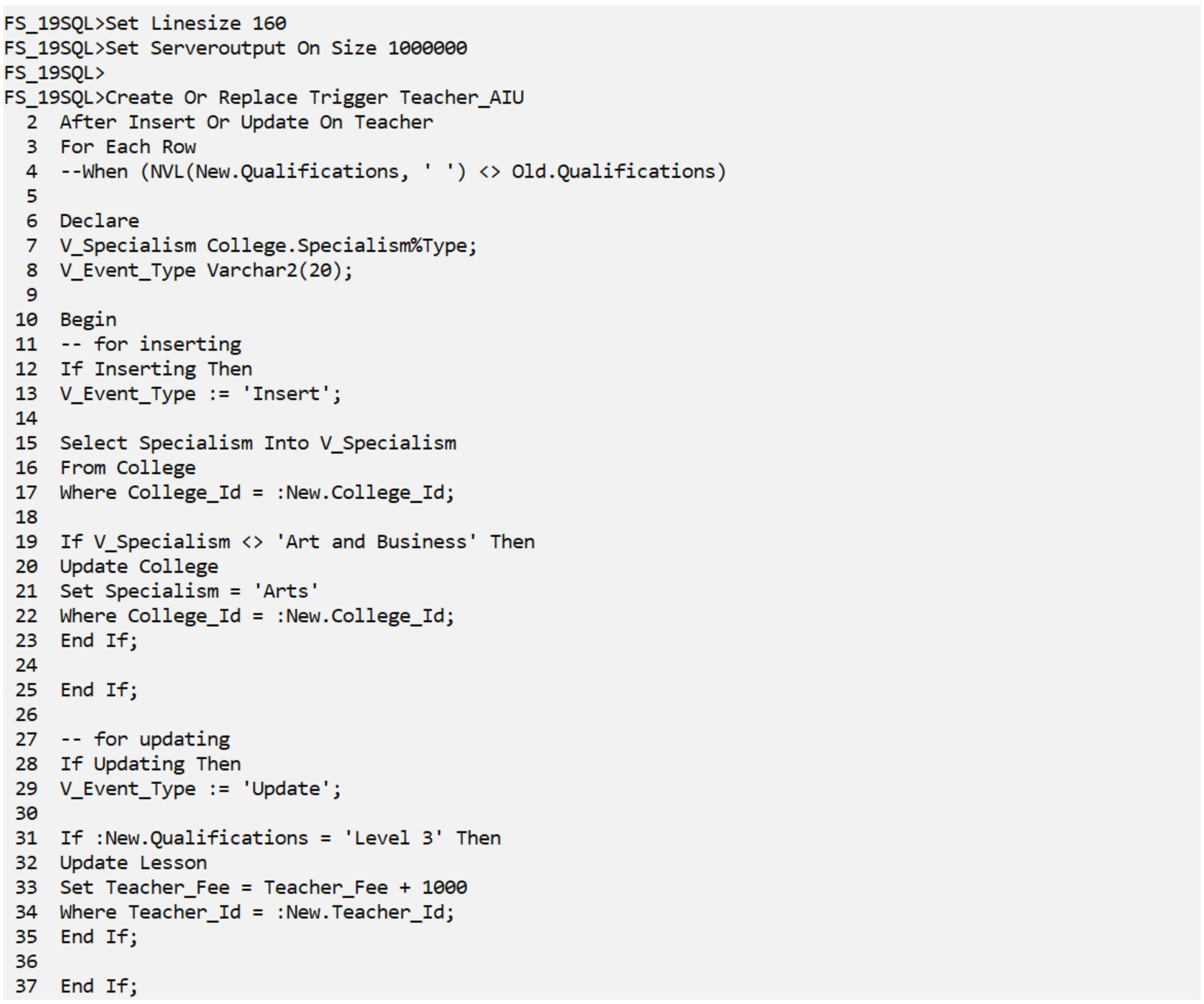
-- Dropping the trigger

Drop Trigger Teacher_AIU;

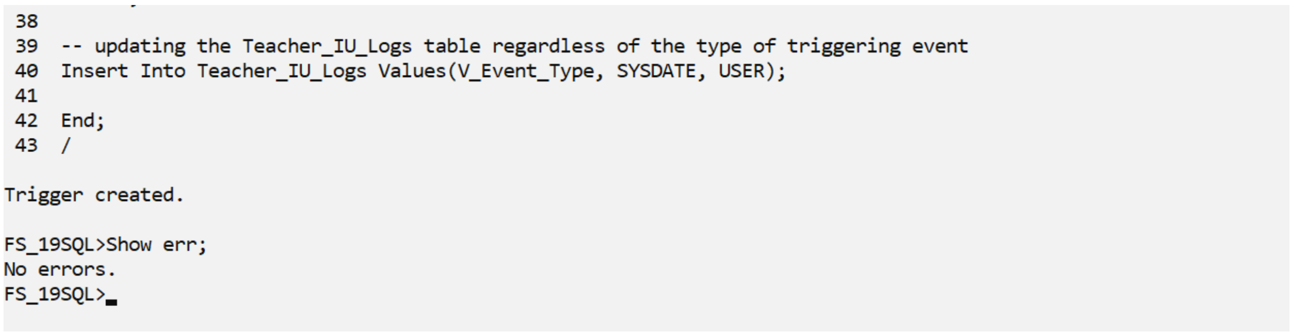
Snips – Question 2



```
FS_19SQL>-- creating the Teacher_IU_Logs table
FS_19SQL>Create Table Teacher_IU_Logs(Event Varchar2(10), DatePerfomed Date, PerformedBy Varchar2(20));

Table created.

FS_19SQL>
```



```
FS_19SQL>Set Linesize 160
FS_19SQL>Set Serveroutput On Size 1000000
FS_19SQL>
FS_19SQL>Create Or Replace Trigger Teacher_AIU
After Insert Or Update On Teacher
For Each Row
--When (NVL(New.Qualifications, ' ') <> Old.Qualifications)

Declare
V_Specialism College.Specialism%Type;
V_Event_Type Varchar2(20);

Begin
-- for inserting
If Inserting Then
V_Event_Type := 'Insert';

Select Specialism Into V_Specialism
From College
Where College_Id = :New.College_Id;

If V_Specialism <> 'Art and Business' Then
Update College
Set Specialism = 'Arts'
Where College_Id = :New.College_Id;
End If;

End If;

-- for updating
If Updating Then
V_Event_Type := 'Update';

If :New.Qualifications = 'Level 3' Then
Update Lesson
Set Teacher_Fee = Teacher_Fee + 1000
Where Teacher_Id = :New.Teacher_Id;
End If;

End If;
```



```

-- updating the Teacher_IU_Logs table regardless of the type of triggering event
Insert Into Teacher_IU_Logs Values(V_Event_Type, SYSDATE, USER);

End;
/

Trigger created.

FS_19SQL>Show err;
No errors.
FS_19SQL>
```

```
FS_19SQL>-- Running tests
FS_19SQL>
FS_19SQL>/**
FS_19SQL>Case 1: Inserting a new teacher whose college is not Arts and Business
FS_19SQL>*/
FS_19SQL>--reference check of the table to be updated
FS_19SQL>Select College_Id, Name, Specialism
  2  From College
  3  Where College_Id = 20;

COLLEGE_ID NAME                                                    SPECIALISM
---------- ----------------------------------------------------- -----------
        20 Maynooth University                                   Business

FS_19SQL>
FS_19SQL>-- inserting a new teacher record which triggers change on the college records
FS_19SQL>Insert Into Teacher(Teacher_Id, Fname, Sname, College_Id) Values(333, 'Noel', 'Tierney', 20);

1 row created.

FS_19SQL>
FS_19SQL>-- confirming the changes to college table have taken place
FS_19SQL>Select College_Id, Name, Specialism
  2  From College
  3  Where College_Id = 20;

COLLEGE_ID NAME                                                    SPECIALISM
---------- ----------------------------------------------------- -----------
        20 Maynooth University                                   Arts

FS_19SQL>
FS_19SQL>/*
FS_19SQL>Case 2: Updating teacher's qualifications to Level 3
FS_19SQL>*/
FS_19SQL>-- reference check of the table to be affected by updating teacher's qualifications
FS_19SQL>Select Teacher_id, Teacher_Fee
  2  From Lesson
  3  Where Teacher_Id = 590;
```

```
TEACHER_ID TEACHER_FEE
---------- -----------
       590        5900
       590        5900
       590        5900

FS_19SQL>
FS_19SQL>-- updating teacher's qualifications to Level 3
FS_19SQL>Update Teacher
  2  Set Qualifications = 'Level 3'
  3  Where Teacher_Id = 590;

1 row updated.

FS_19SQL>
FS_19SQL>-- confirming change's to the teacher fee on the lesson table
FS_19SQL>Select Teacher_id, Teacher_Fee
  2  From Lesson
  3  Where Teacher_Id = 590;

TEACHER_ID TEACHER_FEE
---------- -----------
       590        6900
       590        6900
       590        6900

FS_19SQL>
FS_19SQL>/*
FS_19SQL>Confirming all events have been logged into the teacher_iu_log
FS_19SQL>*/
FS_19SQL>Select * From Teacher_IU_Logs;

EVENT      DATEPERFO PERFORMEDBY
---------- --------- --------------------
Insert     13-MAY-20 STUDENT19
Update     13-MAY-20 STUDENT19

FS_19SQL>
FS_19SQL>-- Rollback
FS_19SQL>Rollback;
```

```
Rollback complete.

FS_19SQL>--Drop Teacher_IU_Logs
FS_19SQL>Drop Table Teacher_IU_Logs;

Table dropped.

FS_19SQL>
FS_19SQL>-- Dropping the trigger
FS_19SQL>Drop Trigger Teacher_AIU;

Trigger dropped.

FS_19SQL>
```

Question 3. (Before Delete, Update)

Write an application that will facilitate easy management of lessons in the database by implicity handling the following scenarios

a. All lessons taking place in venues with annual running cost more than 20000 can not be deleted from the database
b. Lesson's progress cannot be updated unless it's different from the dancer's commitment to the lesson.

Code – Question 3

```
Clear Screen

Set Linesize 160

Set Serveroutput On Size 1000000


Create Or Replace Trigger Lesson_BDU

Before Delete Or Update On Lesson

For Each Row


Declare

        V_Yr_Running_Cost Varchar2(50);

        V_Commitment Varchar2(50);


Begin

        If Deleting Then

                Select Yr_Running_Cost Into V_Yr_Running_Cost

                From Venue

                Where Venue_Id = :Old.Venue_Id;


                If V_Yr_Running_Cost > 20000 Then

                        Raise_Application_Error(-20101, 'Lessons taking place in venues with running cost
more than 20K cannot be deleted!');

                End If;


        End If;


        If Updating Then

                Select Commitment Into V_Commitment

                From Dancer

                Where Dancer_Id = :New.Dancer_Id;
```

```
                    If V_Commitment = :Old.Progress Then
                              Raise_Application_Error(-20101, 'Lesson progress cannot be updated unless its different from dancers commitment');
                    End If;

          End If;

          -- updating the log table

End;
/
Show err;
```

Validation Test – Question 3

```
-- Running tests
/*
          Invalid Test Case a: Deleting lesson taking place in venue whose annual running cost is more than 20K
*/
Delete From Lesson
Where Venue_Id = 13;

/*
          Invalid Test Case  b: Trying to update lesson progress even if it matches with dancer's commitment
*/
Update Lesson
Set Progress = 'Fair'
Where Dancer_Id = 555;

/*
          Valid Test Case a: Deleting lesson taking place in venue whose annual running cost is less than 20K
*/
Delete From Lesson
Where Venue_Id = 5;
```

/*

-- Valid Test Case b: Updating lesson progress which is different from dancer's commitment, the lesson's progres

-- is updated to match dancer's commitment

*/

Update Lesson

Set Progress = 'Fair'

Where Dancer_Id = 777; -- update to Fair

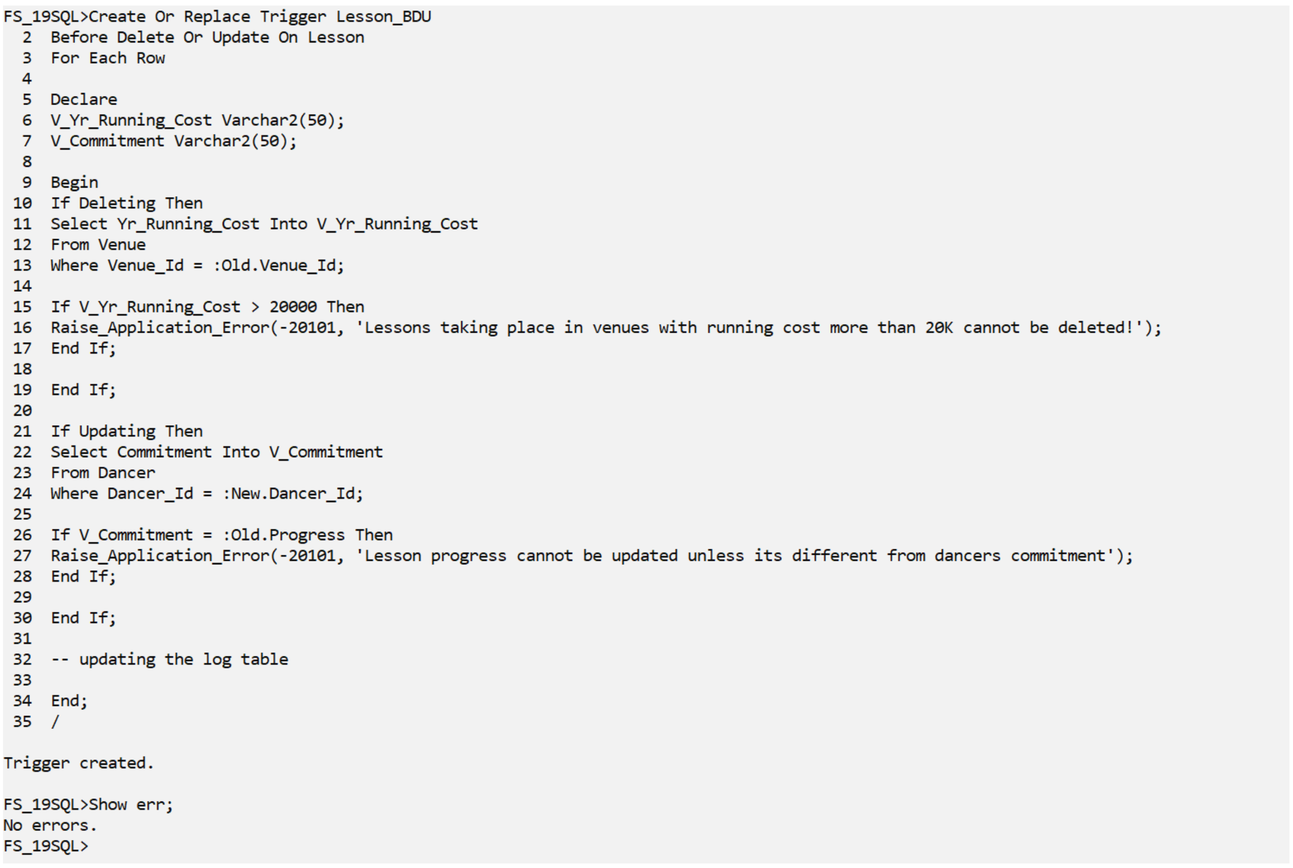
-- rollback

Rollback;

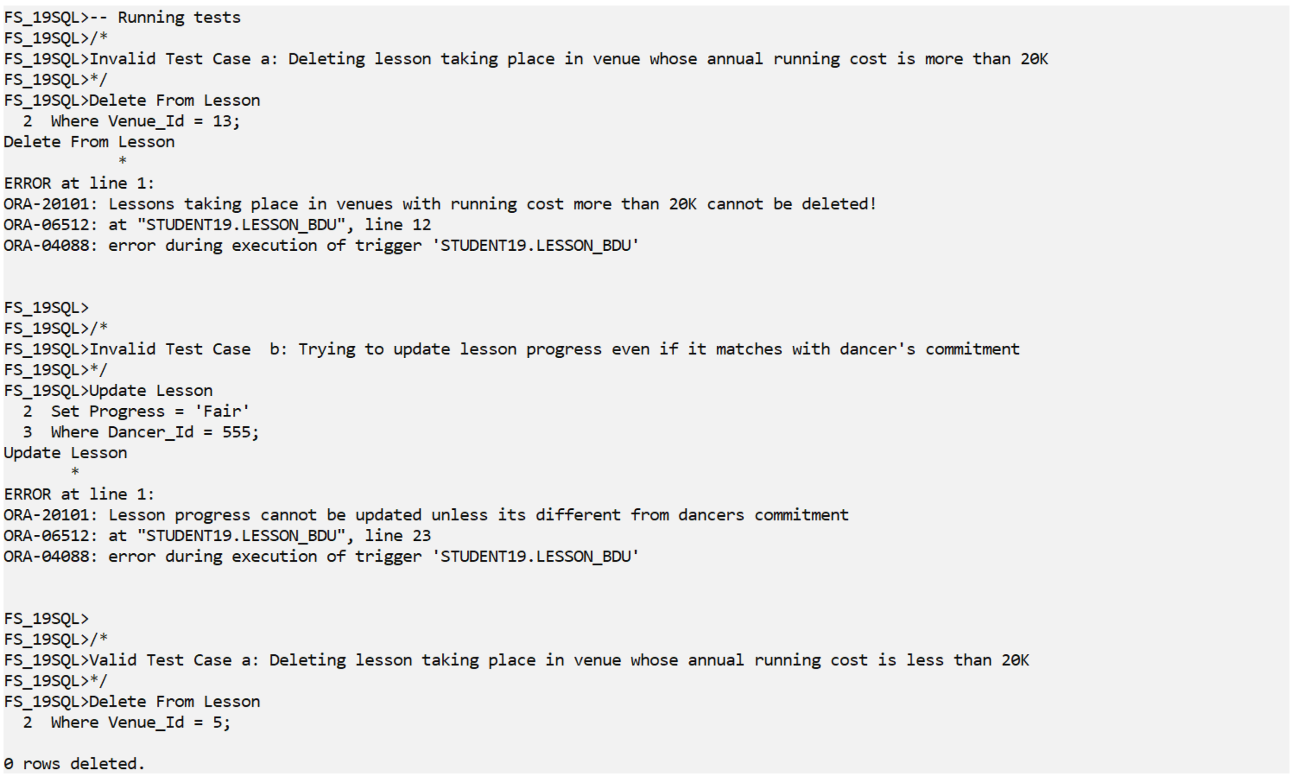
-- drop trigger

Drop Trigger Lesson_BDU;

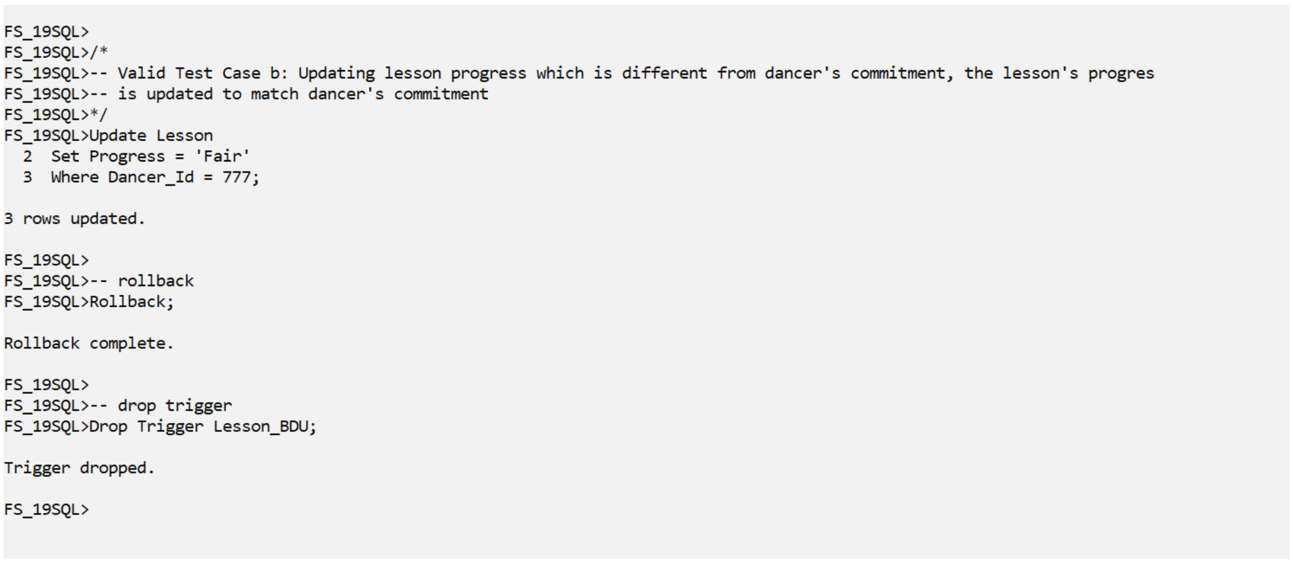
Snips – Question 3



```
FS_19SQL>Create Or Replace Trigger Lesson_BDU
Before Delete Or Update On Lesson
For Each Row

Declare
V_Yr_Running_Cost Varchar2(50);
V_Commitment Varchar2(50);

Begin
If Deleting Then
Select Yr_Running_Cost Into V_Yr_Running_Cost
From Venue
Where Venue_Id = :Old.Venue_Id;

If V_Yr_Running_Cost > 20000 Then
Raise_Application_Error(-20101, 'Lessons taking place in venues with running cost more than 20K cannot be deleted!');
End If;

End If;

If Updating Then
Select Commitment Into V_Commitment
From Dancer
Where Dancer_Id = :New.Dancer_Id;

If V_Commitment = :Old.Progress Then
Raise_Application_Error(-20101, 'Lesson progress cannot be updated unless its different from dancers commitment');
End If;

End If;

-- updating the log table

End;
/

Trigger created.

FS_19SQL>Show err;
No errors.
FS_19SQL>
```



```
FS_19SQL>-- Running tests
FS_19SQL>/*
FS_19SQL>Invalid Test Case a: Deleting lesson taking place in venue whose annual running cost is more than 20K
FS_19SQL>*/
FS_19SQL>Delete From Lesson
  2  Where Venue_Id = 13;
Delete From Lesson
            *
ERROR at line 1:
ORA-20101: Lessons taking place in venues with running cost more than 20K cannot be deleted!
ORA-06512: at "STUDENT19.LESSON_BDU", line 12
ORA-04088: error during execution of trigger 'STUDENT19.LESSON_BDU'


FS_19SQL>
FS_19SQL>/*
FS_19SQL>Invalid Test Case  b: Trying to update lesson progress even if it matches with dancer's commitment
FS_19SQL>*/
FS_19SQL>Update Lesson
  2  Set Progress = 'Fair'
  3  Where Dancer_Id = 555;
Update Lesson
       *
ERROR at line 1:
ORA-20101: Lesson progress cannot be updated unless its different from dancers commitment
ORA-06512: at "STUDENT19.LESSON_BDU", line 23
ORA-04088: error during execution of trigger 'STUDENT19.LESSON_BDU'


FS_19SQL>
FS_19SQL>/*
FS_19SQL>Valid Test Case a: Deleting lesson taking place in venue whose annual running cost is less than 20K
FS_19SQL>*/
FS_19SQL>Delete From Lesson
  2  Where Venue_Id = 5;

0 rows deleted.
```



```
FS_19SQL>
FS_19SQL>/*
FS_19SQL>-- Valid Test Case b: Updating lesson progress which is different from dancer's commitment, the lesson's progres
FS_19SQL>-- is updated to match dancer's commitment
FS_19SQL>*/
FS_19SQL>Update Lesson
  2  Set Progress = 'Fair'
  3  Where Dancer_Id = 777;

3 rows updated.

FS_19SQL>
FS_19SQL>-- rollback
FS_19SQL>Rollback;

Rollback complete.

FS_19SQL>
FS_19SQL>-- drop trigger
FS_19SQL>Drop Trigger Lesson_BDU;

Trigger dropped.

FS_19SQL>
```

Appendix

Youtube Link: https://youtu.be/5NsVROQNh2E

Reference

[1] Rosenzweig B., Silverstrova E. (2003), *Oracle PL/SQL, Second Edition*