**CP5503 Enterprise Database Systems Oracle**

**Assignment Part2 – Trigger (10%)**

**Due : End of Week 8 Prac**

Firstly, you have to setup the user SCM\_JCXXXXXX account and create the required tables. To do so, you have to use the 3 given files in ass2\_sql\_files.zip. You should perform the following steps:

1. Run **SQL Developer**

2. Connect **ORCL** by the user **system**

3. Open the script **create\_user.sql** and change **jc000000** to your jc username, for example, jc123456

4. Run the script **create\_user.sql**

5. You can now disconnect the **system** connection

6. Connect **ORCL** by the user **scm\_jcxxxxxx** (your jc username)

7. Open the script **scm\_tables.sql** and run it

Note: scm\_tables.sql creates tables with basic constraints, no additional constraints

8. Open the script **insert\_records.sql** and run it

If there is anything wrong, you can always re-run the 3 files again.

You have to login as a **SCM\_JCXXXXXX** user to write SQL and PL/SQL to define triggers, stored procedures/functions as specified in the following subsections. You should also include SQL/PLSQL statements to test the triggers and procedures/functions that you have to implement. Make sure that you test single row as well as multiple row actions. Include comments for each testing statement to indicate the trigger or the procedure/function in which it applies and the expected results of the trigger firing or procedure/function executing.

**Task 1 (5 points): TR\_change\_gender\_upper** trigger **(Hint: using upper/lower functions)**

To change the cases of **e\_gender** to upper case before inserting or updating a row in the **Employee** table.

You should produce your answer in Microsoft Word with a heading **Task1** containing:

- the required trigger (PL/SQL statements)

- testing: SQL statements to insert data into the table for testing trigger

- showing the testing results

**Hint:**

**-- the required trigger**

CREATE OR REPLACE TRIGGER TR\_change\_gender\_upper

…

…

BEGIN

:new.e\_gender := upper(:new.e\_gender);

END;

**-- testing statement(s)**

INSERT INTO Employee(…)

VALUES (…);

**-- test result(s)**

SELECT \* FROM Employee;

**Task 2 (5 points): TR\_check\_t\_type** trigger

You should create a trigger called **TR\_check\_t\_type** (**Ticket\_Type** table) to check **t\_type** before inserting or updating a row in **Ticket\_Type** table. If **t\_type** is not one of the values of “Adult”, “Concession” or “Student” then raises an application error.

You should produce your answer in Microsoft Word with a heading **Task2** containing:

- the required trigger (PL/SQL statements)

- testing: write SQL statements to insert 2 records to test the trigger as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | t\_type | price | start\_date | end\_date |
| Invalid | Worker | 12.95 | 1/12/2015 | null |
| Valid | Student | 7.95 | 1/12/2015 | null |

- showing the testing results

**Hint:**

**-- the required trigger**

CREATE OR REPLACE …

…

…

BEGIN

IF … THEN

RAISE\_APPLICATION\_ERROR(…, …);

END IF;

END;

**-- testing statement(s) and test result(s)**

…

…

…

**Task 3 (10 points):** Screening schedule.

Assume that the manager requests screening plans as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| plan\_id | movie\_id | plan\_start\_day | plan\_end\_day | plan\_min\_start\_hh24 | plan\_max\_start\_hh24 | screenings |
| 1 | 1 | 1/02/2016 | 28/02/2016 | 9 | 17 | 4 (per day) |
| 2 | 2 | 1/02/2016 | 28/02/2016 | 9 | 22 | 6 (per day) |
| 3 | 3 | 1/02/2016 | 28/02/2016 | 9 | 22 | 5 (per day) |
| 4 | 4 | 1/02/2016 | 28/02/2016 | 9 | 17 | 4 (per day) |

**Table 1. Screening Plans**

The manager wants the second movie (movie\_id = 2) to be screened in the biggest theatre and have an additional screening in another theatre at the busiest time of the day (19:30). It means that there will be 2 concurrent screenings of the second movie around the busiest time. The fourth movie is in the second biggest theatre.

(a) Provide screening information for one day, for example, the first day of the above 4 plans. Based on the information in Table 1, there are 19 screenings a day (4+6+5+4). Write your answer onto the below table (the first three screenings are done for you):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| screening\_id | plan\_id | theatre\_id | screening\_date | screening\_start\_hh24 | screening\_start\_mm60 |
| 1 | 2 | 3 |  | 9 | 30 |
| 2 | 2 | 3 |  | 12 | 0 |
| 3 | 2 | 3 |  | 14 | 30 |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |
| 10 |  |  |  |  |  |
| 11 |  |  |  |  |  |
| 12 |  |  |  |  |  |
| 13 |  |  |  |  |  |
| 14 |  |  |  |  |  |
| 15 |  |  |  |  |  |
| 16 |  |  |  |  |  |
| 17 |  |  |  |  |  |
| 18 |  |  |  |  |  |
| 19 |  |  |  |  |  |

**Table 2. Screening details on 01/02/2016**

Please make sure that there is at least a break of 30 minutes between 2 consecutive screenings in the same theatre. In addition, the movie length should be rounded up to 10s. For example, 118 → 120 (minutes), 102 → 110 (minutes).

(b) From **Table 1** and **Table 2**, insert the plan records and screening records into the database.

You should produce your answer in Microsoft Word with a heading **Task3** containing your answers for (a) and (b).

**Task 4 (10 points):**

As stated in Task 3, the plans are all 28 days therefore we have to insert the total of 28 × (4 + 6 + 5 + 4) = 532 records into the **Screening** table! In Task 3, we only insert 19 screening records for the first day. Is there a better way than manually inserting the remaining records? Give your answer to insert **the remaining records**.

You should produce your answer in Microsoft Word with a heading **Task4** containing:

- The required answer of Task 4.

**Task 5 (15 points): TR\_valid\_15\_minute\_before\_screening** trigger

This trigger should check **ticket\_date** before inserting a row in **Ticket** table. The ticket issued date (ticket\_date) cannot be greater than screening time + 15 minutes. Otherwise, the trigger should raise an application error with a meaningful message.

**Hint:**

screening time = screening\_date + (screening\_start\_hh24)/24 + (screening\_start\_mm60)/(24\*60)

You should produce your answer in Microsoft Word with a heading **Task6** containing:

- the required trigger (PL/SQL statements)

- testing: write SQL statements to insert data into the table for testing trigger. Both cases: valid and invalid records.

- showing the testing results

**Task 6 (15 points): Func\_is\_seat\_available**

Write a function called **Func\_is\_seat\_available** to check availability of a seat in a screening. This function should contain the following input parameters: **seat\_id\_p**, **screening\_id\_p**. It should return 1 if there aren’t any existing records in the **Ticket** table having the same seat number (for example, A1) and the same screening with the supplied parameters; otherwise return 0.

**Hint:** Define a cursor c; if c%found then seat is available otherwise seat is not available.

You should produce your answer in Microsoft Word with a heading **Task6** containing:

- the required function (PL/SQL statements)

- testing: write an anonymous block (PL/SQL statements) to call the function to test. Both cases: Available and not available.

- showing the testing results.

**SUBMISSION:**

Submission date: **By 11:59 pm Sunday Week 8**.   
Submit a single word file as jcnumber.docx (e.g. jc222333.docx) on LearnJCU containing **all of your answers.** You can find the submission link inside the **Assignments** folder which is located inside the **Subject Materials** folder.

**Notes on SYSDATE and date values:**

- SYSDATE: returns the current date and time.

- Strings must be converted to a proper date values – must use to\_date() function

Here is an example:

alter session set nls\_date\_format = 'DD/MM/YYYY HH24:MI:SS';

begin

dbms\_output.put\_line(to\_char(to\_date('24/01/2016 15:30:10', 'DD/MM/YYYY HH24:MI:SS')));

dbms\_output.put\_line(to\_char(to\_date('24/01/2016', 'DD/MM/YYYY')));

dbms\_output.put\_line(to\_char(sysdate, 'DD/MM/YYYY HH24:MI:SS'));

dbms\_output.put\_line(to\_char(to\_date(sysdate, 'DD/MM/YYYY HH24:MI:SS')));

dbms\_output.put\_line(to\_char(sysdate, 'DD/MM/YYYY'));

end;

-- Dbms Output --

24/01/2016 15:30:10

24/01/2016 00:00:00

23/01/2016 17:59:34

23/01/2016 17:59:34

23/01/2016