SIT103/SIT772: Database Fundamentals

8.2D: PL/SQL (Trigger, Stored Procedure and Function)

Overview

In this task, you will learn **PL/SQL to create and use trigger, stored procedure, and function**. For this exercise, you will use **ORACLE**. It is assumed that you have set up your access to the Deakin ORACLE server. If you have not done so, please refer to the unit site for instructions to set up your access first. The help resources are also included in the task resources zip file for this task.

Once you got access to your ORACLE database, check what tables are there in your database using the following command:

```
SELECT table_name FROM user_tables;
```

If there are tables called CUSTOMER and INVOICE, delete them using the following commands:

```
DROP TABLE CUSTOMER;
DROP TABLE INVOICE;
```

Now, create two tables named CUSTOMER and INVOICE, and populate records using the *'8_2D-data.sql'* script file provided in a zip file as part of the task resources. It will create the following tables for you.

FIGURE P8.16 CH08_SIMPLECO DATABASE TABLES

Database name: Ch08_SimpleCo

Table name: CUSTOMER

CUST_NUM	CUST_LNAME	CUST_FNAME	CUST_BALANCE
1000	Smith	Jeanne	1050.11
1001	Ortega	Juan	840.92

Table name: INVOICE

INV_NUM	CUST_NUM	INV_DATE	INV_AMOUNT
8000	1000	23-Mar-16	235.89
8001	1001	23-Mar-16	312.82
8002	1001	30-Mar-16	528.10
8003	1000	12-Apr-16	194.78
8004	1000	23-Apr-16	619.44

Check records you have in both CUSTOMER and INVOICE tables.

```
SELECT * FROM CUSTOMER;
SELECT * FROM INVOICE;
```

Tasks to do

1. Write a procedure named **PRC_ADD_CUSTOMER** to add a new customer to the CUSTOMER table. The procedure will take **four IN arguments** which are values of the four columns of the CUSTOMER table for the new customer to be added.

   Test the procedure by executing it to add the following new customer.

   CUST_NUM = 1002
   CUST_LNAME = Rauthor
   CUST_FNAME = Peter
   CUST_BALANCE = 0.0

   As your answer to this question, please provide **your code to create the procedure** and a **screenshot of your ORACLE SQL screen showing it works** when it is executed with the result of `SELECT * FROM CUSTOMER` before and after its successful execution.

```
SELECT * FROM CUSTOMER;
EXEC PRC_ADD_CUSTOMER(. . . .);
SELECT * FROM CUSTOMER;
```

2. Write a function named **get_invoice_count** to retrieve the number of invoices for a given customer. Your function will take CUST_NUM as argument and returns the number of invoices for the customer.

   As your answer to this question, please provide **your code to create the function** and a **screenshot of your ORACLE SQL screen showing the results of the following SQL statements calling the function**.

```
SELECT get_invoice_count(0) FROM DUAL;
SELECT get_invoice_count(1000) FROM DUAL;
SELECT get_invoice_count(1005) FROM DUAL;
```

3. Write a trigger named **TRG_UPDATE_CUST_BALANCE** to update the CUST_BALANCE in the CUSTOMER table when a new invoice is added in the INVOICE table. The value of the INV_AMOUNT column of the new invoice should be added to the respective customer's balance in the CUSTOMER table.

   Test the trigger by adding the following new invoice record in the INVOICE table.

   INV_NUM = 8005
   CUST_NUM = 1001
   INV_DATE = 27-Apr-2016

INV_AMOUNT = 225.40

It should add the INV_AMOUNT of 225.40 to the balance of the customer (CUST_NUM = 1001) in the CUSTOMER table.

[*Hint*:

This trigger is activated when a record is added, i.e., it is a row-level trigger. When a row-level trigger fired, the PL/SQL runtime system creates and populates the two **pseudo-records** called **`OLD`** and **`NEW`**.

A row-level trigger must perform action for each row. This is ensured by adding **`FOR EACH ROW`** before the `BEGIN … END` action block in the trigger definition. See the following ORACLE tutorial on Triggers for more help.

https://docs.oracle.com/database/121/TDDDG/tdddg_triggers.htm#TDDDG50000

https://docs.oracle.com/database/121/LNPLS/triggers.htm#LNPLS99888

]

As your answer to this question, please provide **your code to create the trigger** and a **screenshot of your ORACLE SQL screen showing it works** with the results of `SELECT * FROM CUSTOMER` before and after the above new invoice record is inserted in the INVOICE table.

```
SELECT * FROM CUSTOMER;
INSERT INTO INVOICE VALUES (8005,1001,'27-Apr-2016',225.40);
SELECT * FROM CUSTOMER;
```

4. Write a procedure named **PRC_ADD_INVOICE** to add a new invoice record to the INVOICE table. The procedure will take **four IN arguments** which are values of the four columns of the INVOICE table for the new invoice to be added.

Test the procedure to demonstrate it works by executing it to add the following new invoice.

INV_NUM = 8006
CUST_NUM = 1002
INV_DATE = 29-Apr-2016
INV_AMOUNT = 175.85

As your answer to this question, please provide **your code to create the procedure** and a **screenshot of your ORACLE SQL screen showing it works** when it is executed with the results of `SELECT * FROM CUSTOMER` and `SELECT * FROM INVOICE` before and after its successful execution.

```
SELECT * FROM CUSTOMER;
SELECT * FROM INVOICE;
```

```
EXEC PRC_ADD_INVOICE(. . . .);
SELECT * FROM INVOICE;
SELECT * FROM CUSTOMER;
```

Please **DO NOT FORGET** to include the results of `SELECT * FROM CUSTOMER` as well before and after executing the procedure to add the new invoice.

Submission Requirements:

Submit one PDF/WORD file with answers to all above four questions as requested.

Submission Due

The due for each task has been stated via its OnTrack task information dashboard.