Deakin University

Database Fundamentals

OnTrack Submission

PL/PSQL - Trigger, Procedure and Function

Submitted By:
Thi Thu Suong Ngo
s224757434
2025/05/23 20:03

Tutor:
Mohammad belayet Hossain

Outcome	Weight
Fundamental concepts of database	♦♦♦♦♦
Relational Database Modelling	♦♦♦♦♦
Structured Query Language (SQL)	♦♦♦♦♦
Reflection	♦♦♦♦♦
Research and critical review	♦♦♦♦♦

Through Tasks 1 to 5, I gained hands-on experience with PL/SQL in Oracle. In Task 1, I learned to create a trigger that automatically updates a customer's balance when a new invoice is inserted. In Task 2, I created a stored procedure to add a new customer using IN parameters. Task 3 extended this by creating a procedure to insert a new invoice and observing how it triggered balance updates. In Task 4, I developed a DELETE trigger that subtracts the invoice amount from a customer's balance when an invoice is removed. Finally, in Task 5, I created a function to return the number of invoices for a specific customer, applying logic with parameters and return values. These tasks helped reinforce my understanding of triggers, procedures, functions, and table manipulation in a database environment.

May 23, 2025

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

Verify the Data from `8_2D_data.sql`

SELECT * FROM CUSTOMER;

```
SQL> SELECT * FROM CUSTOMER;

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

SELECT * FROM INVOICE;

```
   INV_NUM   CUST_NUM INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8000       1000 23-MAR-22     235.89
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
```

Task 1: Trigger - TRG_UPDATE_CUST_BALANCE

- **Show Full Table before Inserting**

SELECT * FROM CUSTOMER;

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

SELECT * FROM INVOICE;

```
   INV_NUM   CUST_NUM INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8000       1000 23-MAR-22     235.89
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
```

- **Code to create the trigger**

```
CREATE OR REPLACE TRIGGER TRG_UPDATE_CUST_BALANCE
AFTER INSERT ON INVOICE
FOR EACH ROW
BEGIN
  UPDATE CUSTOMER
  SET CUST_BALANCE = CUST_BALANCE + :NEW.INV_AMOUNT
  WHERE CUST_NUM = :NEW.CUST_NUM;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER TRG_UPDATE_CUST_BALANCE
AFTER INSERT ON INVOICE
FOR EACH ROW
BEGIN
  UPDATE CUSTOMER
  SET CUST_BALANCE = CUST_BALANCE + :NEW.INV_AMOUNT
  WHERE CUST_NUM = :NEW.CUST_NUM;
END;
/  2    3    4    5    6    7    8    9

Trigger created.
```

- **Insert the Test Invoice**

```
INSERT INTO INVOICE VALUES (8005, 1001, TO_DATE('27-Apr-2022', 'DD-Mon-YYYY'), 225.40);
```

```
SQL> INSERT INTO INVOICE VALUES (8005, 1001, TO_DATE('27-Apr-2022', 'DD-Mon-YYYY'), 225.40);

1 row created.
```

- **Check full table after inserting**

```
SELECT * FROM INVOICE;
```

```
   INV_NUM   CUST_NUM INV_DATE   INV_AMOUNT
---------- ---------- --------- ----------
      8000       1000 23-MAR-22     235.89
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
      8005       1001 27-APR-22      225.4

6 rows selected.
```

SELECT * FROM CUSTOMER;

```
  CUST_NUM CUST_LNAME      CUST_FNAME      CUST_BALANCE
---------- --------------- --------------- ------------
      1000 Smith           Jeanne               1050.11
      1001 Ortega          Juan                 1066.32
```

Task 2: Procedure - PRC_ADD_CUSTOMER

- **Show CUSTOMER Table Before Insert**

SELECT * FROM CUSTOMER;

```
  CUST_NUM CUST_LNAME      CUST_FNAME      CUST_BALANCE
---------- --------------- --------------- ------------
      1000 Smith           Jeanne               1050.11
      1001 Ortega          Juan                 1066.32
```

- **Create the Procedure**

```
CREATE OR REPLACE PROCEDURE PRC_ADD_CUSTOMER (
  p_cust_num     IN NUMBER,
  p_cust_lname   IN VARCHAR2,
  p_cust_fname   IN VARCHAR2,
  p_cust_balance IN NUMBER
)
IS
BEGIN
  INSERT INTO CUSTOMER (CUST_NUM, CUST_LNAME, CUST_FNAME, CUST_BALANCE)
  VALUES (p_cust_num, p_cust_lname, p_cust_fname, p_cust_balance);
END;
/
```

```
SQL> SELECT * FROM CUSTOMER WHERE CUST_NUM = 1002;

no rows selected

SQL> CREATE OR REPLACE PROCEDURE PRC_ADD_CUSTOMER (
  2    p_cust_num     IN NUMBER,
  p_cust_lname   IN VARCHAR2,
  p_cust_fname   IN VARCHAR2,
  p_cust_balance IN NUMBER
)
IS
BEGIN
  INSERT INTO CUSTOMER (CUST_NUM, CUST_LNAME, CUST_FNAME, CUST_BALANCE)
  VALUES (p_cust_num, p_cust_lname, p_cust_fname, p_cust_balance);
END;
/
  3    4    5    6    7    8    9   10   11   12
Procedure created.
```

- **Execute the Procedure**

EXEC PRC_ADD_CUSTOMER(1002, 'Rauthor', 'Peter', 0.0);

```
SQL> EXEC PRC_ADD_CUSTOMER(1002, 'Rauthor', 'Peter', 0.0);

PL/SQL procedure successfully completed.
```

- **Show CUSTOMER Table After Insert**

SELECT * FROM CUSTOMER;

CUST_NUM	CUST_LNAME	CUST_FNAME	CUST_BALANCE
1000	Smith	Jeanne	1050.11
1001	Ortega	Juan	1066.32
1002	Rauthor	Peter	0

Task 3: Procedure - PRC_ADD_INVOICE

- **Show existing table before executing the procedure to add new invoice**

SELECT * FROM INVOICE;

```
   INV_NUM   CUST_NUM INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8000       1000 23-MAR-22     235.89
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
      8005       1001 27-APR-22      225.4

6 rows selected.
```

SELECT * FROM CUSTOMER;

```
  CUST_NUM CUST_LNAME      CUST_FNAME      CUST_BALANCE
---------- --------------- --------------- ------------
      1000 Smith           Jeanne               1050.11
      1001 Ortega          Juan                 1066.32
      1002 Rauthor         Peter                      0
```

- **Create the Procedure**

```
CREATE OR REPLACE PROCEDURE PRC_ADD_INVOICE (
  p_inv_num     IN NUMBER,
  p_cust_num    IN NUMBER,
  p_inv_date    IN DATE,
  p_inv_amount  IN NUMBER
)
IS
BEGIN
  INSERT INTO INVOICE (INV_NUM, CUST_NUM, INV_DATE, INV_AMOUNT)
  VALUES (p_inv_num, p_cust_num, p_inv_date, p_inv_amount);
END;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE PRC_ADD_INVOICE (
  p_inv_num     IN NUMBER,
  p_cust_num    IN NUMBER,
  p_inv_date    IN DATE,
  p_inv_amount  IN NUMBER
)
IS
BEGIN
  INSERT INTO INVOICE (INV_NUM, CUST_NUM, INV_DATE, INV_AMOUNT)
  VALUES (p_inv_num, p_cust_num, p_inv_date, p_inv_amount);
END;
/
  2    3    4    5    6    7    8    9   10   11   12
Procedure created.
```

- **Execute the Procedure**

EXEC PRC_ADD_INVOICE(8006, 1002, TO_DATE('29-Apr-2022', 'DD-Mon-YYYY'), 175.85);

```
SQL> EXEC PRC_ADD_INVOICE(8006, 1002, TO_DATE('29-Apr-2022', 'DD-Mon-YYYY'), 175.85);

PL/SQL procedure successfully completed.
```

- **Show all table after the procedure to add new invoice**

SELECT * FROM INVOICE;

```
   INV_NUM   CUST_NUM INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8000       1000 23-MAR-22     235.89
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
      8005       1001 27-APR-22      225.4
      8006       1002 29-APR-22     175.85

7 rows selected.
```

SELECT * FROM CUSTOMER;

```
  CUST_NUM CUST_LNAME      CUST_FNAME      CUST_BALANCE
---------- --------------- --------------- ------------
      1000 Smith           Jeanne               1050.11
      1001 Ortega          Juan                 1066.32
      1002 Rauthor         Peter                 175.85
```

Task 4: Trigger - TRG_UPDATE_CUST_BALANCE2

- **Show Tables Before Deletion**

SELECT * FROM CUSTOMER;

CUST_NUM	CUST_LNAME	CUST_FNAME	CUST_BALANCE
1000	Smith	Jeanne	1050.11
1001	Ortega	Juan	1066.32
1002	Rauthor	Peter	175.85

SELECT * FROM INVOICE;

INV_NUM	CUST_NUM	INV_DATE	INV_AMOUNT
8000	1000	23-MAR-22	235.89
8001	1001	23-MAR-22	312.82
8002	1001	30-MAR-22	526.1
8003	1000	12-APR-22	194.78
8004	1000	23-APR-22	619.44
8005	1001	27-APR-22	225.4
8006	1002	29-APR-22	175.85

7 rows selected.

- **Create the DELETE Trigger**

```
CREATE OR REPLACE TRIGGER TRG_UPDATE_CUST_BALANCE2

AFTER DELETE ON INVOICE

FOR EACH ROW

BEGIN

  UPDATE CUSTOMER

  SET CUST_BALANCE = CUST_BALANCE - :OLD.INV_AMOUNT

  WHERE CUST_NUM = :OLD.CUST_NUM;

END;
```

/

```
SQL> CREATE OR REPLACE TRIGGER TRG_UPDATE_CUST_BALANCE2
AFTER DELETE ON INVOICE
FOR EACH ROW
BEGIN
  UPDATE CUSTOMER
  SET CUST_BALANCE = CUST_BALANCE - :OLD.INV_AMOUNT
  WHERE CUST_NUM = :OLD.CUST_NUM;
END;
/  2    3    4    5    6    7    8    9

Trigger created.
```

- **Delete the Invoice**

DELETE FROM INVOICE WHERE INV_NUM = 8000;

```
SQL> DELETE FROM INVOICE WHERE INV_NUM = 8000;

1 row deleted.
```

- **Show Tables After Deletion**

SELECT * FROM INVOICE;

```
   INV_NUM   CUST_NUM INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8001       1001 23-MAR-22     312.82
      8002       1001 30-MAR-22      526.1
      8003       1000 12-APR-22     194.78
      8004       1000 23-APR-22     619.44
      8005       1001 27-APR-22      225.4
      8006       1002 29-APR-22     175.85

6 rows selected.
```

SELECT * FROM CUSTOMER;

```
  CUST_NUM CUST_LNAME      CUST_FNAME      CUST_BALANCE
---------- --------------- --------------- ------------
      1000 Smith           Jeanne                814.22
      1001 Ortega          Juan                 1066.32
      1002 Rauthor         Peter                 175.85
```

Task 5: Function - GET_INVOICE_COUNT

- **Create the Function**

```
CREATE OR REPLACE FUNCTION GET_INVOICE_COUNT (
  p_cust_num IN NUMBER
) RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM INVOICE
  WHERE CUST_NUM = p_cust_num;

  RETURN v_count;
END;
/
```

```
SQL> CREATE OR REPLACE FUNCTION GET_INVOICE_COUNT (
  p_cust_num IN NUMBER
) RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM INVOICE
  WHERE CUST_NUM = p_cust_num;

  RETURN v_count;
END;
/  2    3    4    5    6    7    8    9   10   11   12   13

Function created.
```

- **Test the Function**

```
SELECT GET_INVOICE_COUNT(1000) FROM DUAL;
```

```
GET_INVOICE_COUNT(1000)
-----------------------
                      2
```

SELECT GET_INVOICE_COUNT(1001) FROM DUAL;

```
SQL> SELECT GET_INVOICE_COUNT(1001) FROM DUAL;

GET_INVOICE_COUNT(1001)
-----------------------
                      3
```

SELECT GET_INVOICE_COUNT(1002) FROM DUAL;

```
SQL> SELECT GET_INVOICE_COUNT(1002) FROM DUAL;

GET_INVOICE_COUNT(1002)
-----------------------
                      1
```