



## CS-GY 6083 - B Principles of Database Systems

### Functions, Stored Procedures and Triggers

**Trigger:** A trigger is PL/SQL code which contains a set of SQL statements in database which is automatically executed whenever any special event occurs specific to condition defined, such as before or after of insert, delete, update etc. Triggers are used to implement complex business rules that cannot be defined using constraints.

We can define 6 types of DML triggers for the table:

AFTER INSERT: activated after data is inserted into the table.

AFTER UPDATE: activated after data in the table is modified.

AFTER DELETE: activated after data is deleted/removed from the table.

BEFORE INSERT: activated before data is inserted into the table.

BEFORE UPDATE: activated before data in the table is modified.

BEFORE DELETE: activated before data is deleted/removed from the table

**Function:** A function is PL/SQL code that accepts input parameter, and produces output based upon SQL instructions and calculation performed as coded. Function can be single row function or multi-row function (aggregate function).

**Procedure:** A procedure is also PL/SQL code and set of pre-compiled queries which can be used and shared among multiple programs. It can access and modify data in database as coded. A procedure can be thought of as a method. They can be invoked through triggers, other procedures, or applications on Java, PHP etc. All the statements of a block are passed to the database engine all at once which increases processing speed and decreases the traffic. They reduce the traffic between the database and the application. They add to code reusability, similar to how methods work in other languages such as C/C++ and Java.

### Resources:

#### Oracle

[https://docs.oracle.com/database/121/TDDDG/tdddg\\_triggers.htm#TDDDG52400](https://docs.oracle.com/database/121/TDDDG/tdddg_triggers.htm#TDDDG52400)

<https://www.oracletutorial.com/plsql-tutorial/plsql-function/>

[https://docs.oracle.com/cd/B28359\\_01/appdev.111/b28843/tdddg\\_procedures.htm](https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm)

#### MySQL

<https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html>

<https://www.sqlshack.com/learn-mysql-the-basics-of-mysql-stored-procedures/>

<https://www.mysqltutorial.org/mysql-stored-function/>

## Difference among Trigger, Function, Procedure

| Trigger                                                                     | Function                                                                                          | Procedure                                                                   |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Trigger is executed automatically on actions like update, delete, insert    | We can call a function whenever required but function cannot be executed as it is not precompiled | A procedure can be executed explicitly whenever required                    |
| Trigger cannot be called from either function or procedure                  | Function can be called from trigger and procedure                                                 | Procedure cannot be called from function. It can be called from trigger.    |
| Trigger cannot be called from other trigger                                 | Function can be called from other function                                                        | Procedure can be called from other procedure                                |
| It never returns a value on execution.                                      | It must return a value.                                                                           | It may or may not return a value on execution                               |
| Cannot be used with SQL such as SELECT, INSERT, UPDATE, DELETE, MERGE etc., | Can be used with SQL such as SELECT, INSERT, UPDATE, DELETE, MERGE etc.,                          | Cannot be used with SQL such as SELECT, INSERT, UPDATE, DELETE, MERGE etc., |
| Transactions such as COMMIT, ROLLBACK, SAVEPOINT are not allowed            | Transactions such as COMMIT, ROLLBACK, SAVEPOINT are not allowed                                  | Transactions such as COMMIT, ROLLBACK, SAVEPOINT are allowed                |

### A) FUNCTION EXAMPLE:

#### User defined function Min\_To\_Hr (Converts Minutes to Hours) : ORACLE

##### 1. Code:

```
CREATE OR REPLACE FUNCTION Min_To_Hr(min_ IN NUMBER)
RETURN VARCHAR2
IS Hr VARCHAR2(15);
BEGIN
Hr := TO_CHAR(FLOOR(min_/60)) || ' HR ' || TO_CHAR(MOD(min_, 60)) || ' MIN';
RETURN Hr;
END;
/
```

##### 2. Test Code:

```
CREATE TABLE FILM (
FILM_LEN number
);
INSERT ALL
INTO FILM(FILM_LEN) VALUES(11)
INTO FILM(FILM_LEN) VALUES(12)
INTO FILM(FILM_LEN) VALUES(13)
INTO FILM(FILM_LEN) VALUES(24)
INTO FILM(FILM_LEN) VALUES(25)
```

```

INTO FILM(FILM_LEN) VALUES(27)
INTO FILM(FILM_LEN) VALUES(62)
INTO FILM(FILM_LEN) VALUES(123)
INTO FILM(FILM_LEN) VALUES(134)
INTO FILM(FILM_LEN) VALUES(999)
SELECT * FROM dual;

```

### 3. Running result:

```
SELECT FILM_LEN as "Minutes Number", Min_To_Hr(FILM_LEN) as "Movie Time" FROM FILM;
```

| Minutes Number | Movie Time   |
|----------------|--------------|
| 11             | 0 HR 11 MIN  |
| 12             | 0 HR 12 MIN  |
| 13             | 0 HR 13 MIN  |
| 24             | 0 HR 24 MIN  |
| 25             | 0 HR 25 MIN  |
| 27             | 0 HR 27 MIN  |
| 62             | 1 HR 2 MIN   |
| 123            | 2 HR 3 MIN   |
| 134            | 2 HR 14 MIN  |
| 999            | 16 HR 39 MIN |

Download CSV  
10 rows selected.

## User defined function Min\_To\_Hr (Converts Minutes to Hours): MySQL

### 1. Code:

```

DELIMITER $$
CREATE FUNCTION Min_To_Hr(Min INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE hr VARCHAR(20);
SET hr = concat(FLOOR(Min/60), ' HR ', MOD(Min,60), ' MIN');
RETURN (hr);
END;
$$
DELIMITER ;

```

### 2. Test Code:

```

CREATE DATABASE test;
USE test;
CREATE TABLE IF NOT EXISTS `test`.`FILM` (`length` INT NOT NULL)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

INSERT INTO FILM(length) VALUES(1);
INSERT INTO FILM(length) VALUES(3);
INSERT INTO FILM(length) VALUES(27);
INSERT INTO FILM(length) VALUES(134);
INSERT INTO FILM(length) VALUES(999);

```

### 3. Running result:

```
select length as 'Minutes Number', Min_To_Hr(length) as 'Movie time' from FILM;
```

| Minutes Number | Movie time   |
|----------------|--------------|
| 1              | 0 HR 1 MIN   |
| 3              | 0 HR 3 MIN   |
| 27             | 0 HR 27 MIN  |
| 134            | 2 HR 14 MIN  |
| 999            | 16 HR 39 MIN |

5 rows in set (0.00 sec)

## B) PROCEDURE EXAMPLE:

| AP_EMP           |                      |
|------------------|----------------------|
| P *              | EMPNO NUMBER (4)     |
| *                | EFNAME VARCHAR2 (30) |
| *                | ELNAME VARCHAR2 (30) |
| *                | JOB VARCHAR2 (30)    |
|                  | MGR NUMBER (4)       |
| *                | HIREDATE DATE        |
| *                | SAL NUMBER (7,2)     |
|                  | COMM NCLOB           |
| F *              | DEPTNO NUMBER (2)    |
| AP_EMP_PK(EMPNO) |                      |

Consider above relational table and do following,

- Create this table with your initial as prefix. Create history table AP\_EMP\_SAL\_HIST (use your initial as prefix). The table AP\_EMP\_SAL\_HIST table should have following columns: EMPNO, OLDSAL, NEWSAL, and CHANGE\_DATE. The purpose of history table is to record history of salary change. Whenever the salary is updated in AP\_EMP table, it should record EMPNO, OLDSAL, NEWSAL, and CHAGE\_DATE in history table AP\_EMP\_SAL\_HIST. The OLDSAL is the original salary (before the update of the salary), NEWSQL is the new salary (after the update of the salary), CHANGE\_DATE is the date on which salary was updated, and should be default as System Date, i.e. current date. Employees may have their salary change multiple times during their employment. Define appropriate primary key of the history table.
- Research on Database Procedure (Database procedure is a PL/SQL code that takes input parameters and do activities as coded in procedure to achieve desired result). The name of the procedure should be <your initial>\_PROC\_SAL\_CHANGE. This procedure will take two input parameters, Employee Number and New Salary. The procure code will do two tasks, 1) update the salary of that employee to New Salary in AP\_EMP table 2) insert a new record in history table AP\_EMP\_SAL\_HIST with old and new salary and date of the salary change

Data of AP\_EMP (Result screenshot of SELECT \* FROM AP\_EMP)

|   |                       |
|---|-----------------------|
| 1 | SELECT * FROM AP_EMP; |
| 2 |                       |
| 3 |                       |
| 4 |                       |

| EMPNO | EFNAME    | ELNAME  | JOB       | MGR  | HIREDATE  | SAL   | COMM | DEPTNO |
|-------|-----------|---------|-----------|------|-----------|-------|------|--------|
| 7369  | CHARLES   | SMITH   | CLERK     | 7902 | 08-OCT-01 | 2400  | —    | 20     |
| 7499  | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 12-DEC-01 | 4800  | 600  | 30     |
| 7521  | MATTHEW   | WARD    | SALESMAN  | 7698 | 14-DEC-01 | 3750  | 1000 | 30     |
| 7566  | NICHOLAS  | JONES   | MANAGER   | 7839 | 22-JAN-02 | 8925  | —    | 20     |
| 7654  | CHRIS     | MARTIN  | SALESMAN  | 7698 | 20-JUL-02 | 3750  | 2800 | 30     |
| 7698  | BLAKE     | GRIFFIN | MANAGER   | 7839 | 20-FEB-02 | 8550  | —    | 30     |
| 7782  | KENT      | CLARK   | MANAGER   | 7839 | 31-MAR-02 | 7350  | —    | 10     |
| 7788  | DEVIN     | BOOKER  | ANALYST   | 7566 | 08-FEB-08 | 9000  | —    | 20     |
| 7839  | MARTIN    | KING    | PRESIDENT | —    | 08-SEP-02 | 15000 | —    | 10     |
| 7844  | WILLIAM   | TURNER  | SALESMAN  | 7698 | 30-JUN-02 | 4500  | 0    | 30     |
| 7876  | JOHN      | ADAMS   | CLERK     | 7788 | 13-MAR-08 | 3300  | —    | 20     |
| 7900  | LEBRON    | JAMES   | CLERK     | 7698 | 24-SEP-02 | 2850  | —    | 30     |
| 7902  | CHRISTIAN | FORD    | ANALYST   | 7566 | 24-SEP-02 | 9000  | —    | 20     |
| 7934  | MIKE      | MILLER  | CLERK     | 7782 | 14-NOV-02 | 3900  | —    | 10     |

Download CSV  
14 rows selected.

### Oracle version:

```
CREATE TABLE ap_emp_sal_hist (  
    empno      NUMBER(4) NOT NULL,  
    change_date DATE DEFAULT SYSDATE NOT NULL,  
    oldsal      NUMBER(7, 2) NOT NULL,  
    newsal      NUMBER(7, 2) NOT NULL  
);
```

```
ALTER TABLE ap_emp_sal_hist ADD CONSTRAINT ap_emp_sal_hist_pk PRIMARY KEY (  
empno, change_date );
```

```
CREATE OR REPLACE PROCEDURE ap_emp_sal_change (in_empno NUMBER, in_newsal  
NUMBER)  
IS duration number;  
BEGIN
```

```
    INSERT INTO ap_emp_sal_hist VALUES (in_empno, sysdate, (SELECT sal FROM ap_emp  
WHERE empno = in_empno), in_newsal);  
    UPDATE ap_emp SET sal = in_newsal WHERE empno = in_empno;
```

```
END ap_emp_sal_change;
```

```
EXECUTE ap_emp_sal_change(7782, 4600); -- Calling the procedure  
EXECUTE ap_emp_sal_change(7369, 3000); -- Calling the procedure
```

|   |                       |
|---|-----------------------|
| 1 | SELECT * FROM ap_emp; |
| 2 |                       |
| 3 |                       |
| 4 |                       |

| EMPNO | EFNAME    | ELNAME  | JOB       | MGR  | HIREDATE  | SAL   | COMM | DEPTNO |
|-------|-----------|---------|-----------|------|-----------|-------|------|--------|
| 7369  | CHARLES   | SMITH   | CLERK     | 7902 | 08-OCT-01 | 3000  | —    | 20     |
| 7499  | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 12-DEC-01 | 4800  | 600  | 30     |
| 7521  | MATTHEW   | WARD    | SALESMAN  | 7698 | 14-DEC-01 | 3750  | 1000 | 30     |
| 7566  | NICHOLAS  | JONES   | MANAGER   | 7839 | 22-JAN-02 | 8925  | —    | 20     |
| 7654  | CHRIS     | MARTIN  | SALESMAN  | 7698 | 20-JUL-02 | 3750  | 2800 | 30     |
| 7698  | BLAKE     | GRIFFIN | MANAGER   | 7839 | 20-FEB-02 | 8550  | —    | 30     |
| 7782  | KENT      | CLARK   | MANAGER   | 7839 | 31-MAR-02 | 4600  | —    | 10     |
| 7788  | DEVIN     | BOOKER  | ANALYST   | 7566 | 08-FEB-08 | 9000  | —    | 20     |
| 7839  | MARTIN    | KING    | PRESIDENT | —    | 08-SEP-02 | 15000 | —    | 10     |
| 7844  | WILLIAM   | TURNER  | SALESMAN  | 7698 | 30-JUN-02 | 4500  | 0    | 30     |
| 7876  | JOHN      | ADAMS   | CLERK     | 7788 | 13-MAR-08 | 3300  | —    | 20     |
| 7900  | LEBRON    | JAMES   | CLERK     | 7698 | 24-SEP-02 | 2850  | —    | 30     |
| 7902  | CHRISTIAN | FORD    | ANALYST   | 7566 | 24-SEP-02 | 9000  | —    | 20     |
| 7934  | MIKE      | MILLER  | CLERK     | 7782 | 14-NOV-02 | 3900  | —    | 10     |

Download CSV  
14 rows selected.

```
EXECUTE ap_emp_sal_change(7782, 5000);  
EXECUTE ap_emp_sal_change(7369, 4000);  
EXECUTE ap_emp_sal_change(7499, 5000);
```

```

1 SELECT * FROM ap_emp;
2
3
4

```

| EMPNO | EFNAME    | ELNAME  | JOB       | MGR  | HIREDATE  | SAL   | COMM | DEPTNO |
|-------|-----------|---------|-----------|------|-----------|-------|------|--------|
| 7369  | CHARLES   | SMITH   | CLERK     | 7902 | 08-OCT-01 | 4000  | —    | 20     |
| 7499  | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 12-DEC-01 | 5000  | 600  | 30     |
| 7521  | MATTHEW   | WARD    | SALESMAN  | 7698 | 14-DEC-01 | 3750  | 1000 | 30     |
| 7566  | NICHOLAS  | JONES   | MANAGER   | 7839 | 22-JAN-02 | 8925  | —    | 20     |
| 7654  | CHRIS     | MARTIN  | SALESMAN  | 7698 | 20-JUL-02 | 3750  | 2800 | 30     |
| 7698  | BLAKE     | GRIFFIN | MANAGER   | 7839 | 20-FEB-02 | 8550  | —    | 30     |
| 7782  | KENT      | CLARK   | MANAGER   | 7839 | 31-MAR-02 | 5000  | —    | 10     |
| 7788  | DEVIN     | BOOKER  | ANALYST   | 7566 | 08-FEB-08 | 9000  | —    | 20     |
| 7839  | MARTIN    | KING    | PRESIDENT | —    | 08-SEP-02 | 15000 | —    | 10     |
| 7844  | WILLIAM   | TURNER  | SALESMAN  | 7698 | 30-JUN-02 | 4500  | 0    | 30     |
| 7876  | JOHN      | ADAMS   | CLERK     | 7788 | 13-MAR-08 | 3300  | —    | 20     |
| 7900  | LEBRON    | JAMES   | CLERK     | 7698 | 24-SEP-02 | 2850  | —    | 30     |
| 7902  | CHRISTIAN | FORD    | ANALYST   | 7566 | 24-SEP-02 | 9000  | —    | 20     |
| 7934  | MIKE      | MILLER  | CLERK     | 7782 | 14-NOV-02 | 3900  | —    | 10     |

[Download CSV](#)  
 14 rows selected.

```

1 SELECT * FROM ap_emp_sal_hist;
2
3
4

```

| EMPNO | CHANGE_DATE | OLDSAL | NEWSAL |
|-------|-------------|--------|--------|
| 7782  | 26-APR-21   | 7350   | 4600   |
| 7369  | 26-APR-21   | 2400   | 3000   |
| 7782  | 26-APR-21   | 4600   | 5000   |
| 7369  | 26-APR-21   | 3000   | 4000   |
| 7499  | 26-APR-21   | 4800   | 5000   |

[Download CSV](#)  
 5 rows selected.

### MySQL Version:

```
123 • SELECT * FROM ap_emp;
```

```
124
```

```
125
```

```
126
```

100% 1:129

Result Grid

Filter Rows:

Edit:

Export/Import

| empno  | efname    | elname  | job       | mgr  | hiredate            | sal   | comm | deptno |
|--------|-----------|---------|-----------|------|---------------------|-------|------|--------|
| ▶ 7369 | CHARLES   | SMITH   | CLERK     | 7902 | 2001-10-08 00:00:00 | 2400  | NULL | 20     |
| 7499   | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 2001-12-12 00:00:00 | 4800  | 600  | 30     |
| 7521   | MATTHEW   | WARD    | SALESMAN  | 7698 | 2001-12-14 00:00:00 | 3750  | 1000 | 30     |
| 7566   | NICHOLAS  | JONES   | MANAGER   | 7839 | 2002-01-22 00:00:00 | 8925  | NULL | 20     |
| 7654   | CHRIS     | MARTIN  | SALESMAN  | 7698 | 2002-07-20 00:00:00 | 3750  | 2800 | 30     |
| 7698   | BLAKE     | GRIFFIN | MANAGER   | 7839 | 2002-02-20 00:00:00 | 8550  | NULL | 30     |
| 7782   | KENT      | CLARK   | MANAGER   | 7839 | 2002-03-31 00:00:00 | 7350  | NULL | 10     |
| 7788   | DEVIN     | BOOKER  | ANALYST   | 7566 | 2008-02-08 00:00:00 | 9000  | NULL | 20     |
| 7839   | MARTIN    | KING    | PRESIDENT | NULL | 2002-09-08 00:00:00 | 15000 | NULL | 10     |
| 7844   | WILLIAM   | TURNER  | SALESMAN  | 7698 | 2002-06-30 00:00:00 | 4500  | 0    | 30     |
| 7876   | JOHN      | ADAMS   | CLERK     | 7788 | 2008-03-13 00:00:00 | 3300  | NULL | 20     |
| 7900   | LEBRON    | JAMES   | CLERK     | 7698 | 2002-09-24 00:00:00 | 2850  | NULL | 30     |
| 7902   | CHRISTIAN | FORD    | ANALYST   | 7566 | 2002-09-24 00:00:00 | 9000  | NULL | 20     |
| 7934   | MIKE      | MILLER  | CLERK     | 7782 | 2002-11-14 00:00:00 | 3900  | NULL | 10     |
| NULL   | NULL      | NULL    | NULL      | NULL | NULL                | NULL  | NULL | NULL   |

```
CREATE TABLE ap_emp_sal_hist (  
    empno      INT NOT NULL,  
    change_date DATETIME DEFAULT NOW() NOT NULL,  
    oldsal     INT NOT NULL,  
    newsal     INT NOT NULL  
);
```

```
ALTER TABLE ap_emp_sal_hist ADD CONSTRAINT ap_emp_sal_hist_pk PRIMARY KEY (  
empno, change_date );
```

```
DELIMITER //
```

```
CREATE PROCEDURE ap_emp_sal_change (IN in_empno INT, IN in_newsalsal INT)  
BEGIN
```

```
    INSERT INTO ap_emp_sal_hist VALUES (in_empno, sysdate, (SELECT sal FROM ap_emp  
WHERE empno = in_empno), in_newsalsal);
```

```
    UPDATE ap_emp SET sal = in_newsalsal WHERE empno = in_empno;
```

```
END; //
```

```
DELIMITER ;
```



CALL ap\_emp\_sal\_change(7782, 4600); -- calling the procedure

CALL ap\_emp\_sal\_change(7369, 3000); -- calling the procedure

```
151 • SELECT * FROM ap_emp;
```

```
152
```

```
153
```

```
154
```

100% 1:154

Result Grid

Filter Rows: 

Q Search

Edit:

Export/Import

| empno | efname    | elname  | job       | mgr  | hiredate            | sal   | comm | deptno |
|-------|-----------|---------|-----------|------|---------------------|-------|------|--------|
| 7369  | CHARLES   | SMITH   | CLERK     | 7902 | 2001-10-08 00:00:00 | 3000  | NULL | 20     |
| 7499  | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 2001-12-12 00:00:00 | 4800  | 600  | 30     |
| 7521  | MATTHEW   | WARD    | SALESMAN  | 7698 | 2001-12-14 00:00:00 | 3750  | 1000 | 30     |
| 7566  | NICHOLAS  | JONES   | MANAGER   | 7839 | 2002-01-22 00:00:00 | 8925  | NULL | 20     |
| 7654  | CHRIS     | MARTIN  | SALESMAN  | 7698 | 2002-07-20 00:00:00 | 3750  | 2800 | 30     |
| 7698  | BLAKE     | GRIFFIN | MANAGER   | 7839 | 2002-02-20 00:00:00 | 8550  | NULL | 30     |
| 7782  | KENT      | CLARK   | MANAGER   | 7839 | 2002-03-31 00:00:00 | 4600  | NULL | 10     |
| 7788  | DEVIN     | BOOKER  | ANALYST   | 7566 | 2008-02-08 00:00:00 | 9000  | NULL | 20     |
| 7839  | MARTIN    | KING    | PRESIDENT | NULL | 2002-09-08 00:00:00 | 15000 | NULL | 10     |
| 7844  | WILLIAM   | TURNER  | SALESMAN  | 7698 | 2002-06-30 00:00:00 | 4500  | 0    | 30     |
| 7876  | JOHN      | ADAMS   | CLERK     | 7788 | 2008-03-13 00:00:00 | 3300  | NULL | 20     |
| 7900  | LEBRON    | JAMES   | CLERK     | 7698 | 2002-09-24 00:00:00 | 2850  | NULL | 30     |
| 7902  | CHRISTIAN | FORD    | ANALYST   | 7566 | 2002-09-24 00:00:00 | 9000  | NULL | 20     |
| 7934  | MIKE      | MILLER  | CLERK     | 7782 | 2002-11-14 00:00:00 | 3900  | NULL | 10     |
| NULL  | NULL      | NULL    | NULL      | NULL | NULL                | NULL  | NULL | NULL   |

CALL ap\_emp\_sal\_change(7782, 5000);

CALL ap\_emp\_sal\_change(7369, 4000);

CALL ap\_emp\_sal\_change(7499, 5000);

```
155 • SELECT * FROM ap_emp;
```

```
156
```

```
157
```

```
158
```

100%

1:158

Result Grid

Filter Rows:

Search

Edit:

Export/Import

| empno | efname    | elname  | job       | mgr  | hiredate            | sal   | comm | deptno |
|-------|-----------|---------|-----------|------|---------------------|-------|------|--------|
| 7369  | CHARLES   | SMITH   | CLERK     | 7902 | 2001-10-08 00:00:00 | 4000  | NULL | 20     |
| 7499  | GRAYSON   | ALLEN   | SALESMAN  | 7698 | 2001-12-12 00:00:00 | 5000  | 600  | 30     |
| 7521  | MATTHEW   | WARD    | SALESMAN  | 7698 | 2001-12-14 00:00:00 | 3750  | 1000 | 30     |
| 7566  | NICHOLAS  | JONES   | MANAGER   | 7839 | 2002-01-22 00:00:00 | 8925  | NULL | 20     |
| 7654  | CHRIS     | MARTIN  | SALESMAN  | 7698 | 2002-07-20 00:00:00 | 3750  | 2800 | 30     |
| 7698  | BLAKE     | GRIFFIN | MANAGER   | 7839 | 2002-02-20 00:00:00 | 8550  | NULL | 30     |
| 7782  | KENT      | CLARK   | MANAGER   | 7839 | 2002-03-31 00:00:00 | 5000  | NULL | 10     |
| 7788  | DEVIN     | BOOKER  | ANALYST   | 7566 | 2008-02-08 00:00:00 | 9000  | NULL | 20     |
| 7839  | MARTIN    | KING    | PRESIDENT | NULL | 2002-09-08 00:00:00 | 15000 | NULL | 10     |
| 7844  | WILLIAM   | TURNER  | SALESMAN  | 7698 | 2002-06-30 00:00:00 | 4500  | 0    | 30     |
| 7876  | JOHN      | ADAMS   | CLERK     | 7788 | 2008-03-13 00:00:00 | 3300  | NULL | 20     |
| 7900  | LEBRON    | JAMES   | CLERK     | 7698 | 2002-09-24 00:00:00 | 2850  | NULL | 30     |
| 7902  | CHRISTIAN | FORD    | ANALYST   | 7566 | 2002-09-24 00:00:00 | 9000  | NULL | 20     |
| 7934  | MIKE      | MILLER  | CLERK     | 7782 | 2002-11-14 00:00:00 | 3900  | NULL | 10     |
| NULL  | NULL      | NULL    | NULL      | NULL | NULL                | NULL  | NULL | NULL   |






158 • **SELECT \* FROM** ap\_emp\_sal\_hist;

159

160

161

100%  1:162

Result Grid   Filter Rows:

|   | empno | change_date         | oldsal | newsal |  |
|---|-------|---------------------|--------|--------|--|
| ▶ | 7369  | 2021-04-25 22:41:53 | 2400   | 3000   |  |
|   | 7369  | 2021-04-25 22:44:35 | 3000   | 4000   |  |
|   | 7499  | 2021-04-25 22:44:35 | 4800   | 5000   |  |
|   | 7782  | 2021-04-25 22:41:53 | 7350   | 4600   |  |
|   | 7782  | 2021-04-25 22:44:35 | 4600   | 5000   |  |
|   | NULL  | NULL                | NULL   | NULL   |  |

### C) TRIGGER EXAMPLE:

**Example of a trigger to keep history of salary changes of employees.**

--Oracle

```
CREATE TABLE ap_sal_history
(empno NUMBER (4),
oldsal NUMBER (7,2),
newsal NUMBER (7,2),
last_update_dt DATE
);
```

```
ALTER TABLE ap_sal_history ADD CONSTRAINT pk_ap_sal_history PRIMARY KEY (empno, last_update_dt);
```

--Trigger code in Oracle

```
CREATE OR REPLACE TRIGGER TU_AP_EMP_SAL
AFTER UPDATE OF sal
ON ap_emp
FOR EACH ROW
DECLARE
    oldval number(7,2);
    newval number(7,2);
    eno number(4);
BEGIN
    SELECT :old.sal, :new.sal, :new.empno into oldval,newval,eno from dual;
    INSERT INTO ap_sal_history(empno,oldsal,newsal,last_update_dt) values (eno,oldval,newval,sysdate);
END;
/
```

--Test: Update salary of employees as follow and then review salary history table

```
update ap_emp
set sal=sal+200 where empno=7369;
```

```

update ap_emp
set sal=sal+100 where deptno=20;

update ap_emp
set sal=sal+200 where deptno=30;

select empno,oldsal,newsal,to_char(last_update_dt,'mm/dd/yyyy hh24:mi:ss') from ap_sal_history
order by empno,last_update_dt;

```

#### **--MySQL**

```

use oltpdb;
CREATE TABLE ap_sal_history
(empno smallint ,
oldsal decimal(7,2),
newsal decimal (7,2),
last_update_dt datetime
);

ALTER TABLE ap_sal_history ADD CONSTRAINT pk_ap_sal_history PRIMARY KEY (empno,last_update_dt);

```

#### **--Trigger code in MySQL**

```

DELIMITER $$
CREATE TRIGGER TU_AP_EMP_SAL
AFTER UPDATE ON ap_emp FOR EACH ROW
BEGIN
    IF old.sal<>new.sal THEN
        INSERT INTO ap_sal_history(empno,oldsal,newsal,last_update_dt) values (new.empno,old.sal, new.sal,current_timestamp() ) ;
    END IF;
END$$
DELIMITER ;

```

#### **--Test: Update salary of employees as follow and then review salary history table**

```

update ap_emp
set sal=sal+200 where empno=7499;

update ap_emp
set sal=sal+300 where empno=7499;

update ap_emp
set sal=sal+200 where empno=7566;

update ap_emp
set sal=sal+300 where empno=7566

select * from ap_sal_history;

select * from ap_sal_history order by empno,last_update_dt;

```