Database Programming with SQL
12-3: DEFAULT Values, MERGE, and Multi-Table Inserts
Practice Activities

Objectives

- Understand when to specify a DEFAULT value
- Construct and execute a MERGE statement
- Construct and execute DML statements using SUBQUERIES
- Construct and execute multi-table inserts

Try It / Solve It

1. When would you want a DEFAULT value?

-If no value is given while row creation and I want the field to take some predefined value. For example there may be a created on column, and I want that when a row is created, it gets filled up with current time.

2. Currently, the Global Foods F_PROMOTIONAL_MENUS table START_DATE column does not have SYSDATE set as DEFAULT. Your manager has decided she would like to be able to set the starting date of promotions to the current day for some entries. This will require three steps:

   a. In your schema, Make a copy of the Global Foods F_PROMOTIONAL_MENUS table using the following SQL statement:

```
CREATE TABLE copy_f_promotional_menus
AS (SELECT * FROM f_promotional_menus)
```

```
DESCRIBE f_promotional_menus;
DESCRIBE copy_f_promotional_menus;
```

   b. Alter the current START_DATE column attributes using:

```
select start_date  FROM copy_f_promotional_menus;
ALTER TABLE copy_f_promotional_menus
MODIFY(start_date DATE DEFAULT SYSDATE)
```

   c. INSERT the new information and check to verify the results.
   INSERT a new row into the copy_f_promotional_menus table for the manager's new promotion. The promotion code is 120. The name of the promotion is 'New Customer.' Enter DEFAULT for the start date and '01-Jun-2005' for the ending date. The giveaway is a 10% discount coupon. What was the correct syntax used?

```
UPDATE  copy_f_promotional_menus
SET start_date = SYSDATE
WHERE   start_date =    TO_DATE('10-Feb-2004','dd-Mon-yyyy');

INSERT INTO copy_f_promotional_menus(code,name,start_date,end_date,give_away)
VALUES('120','New Customer',DEFAULT,TO_DATE('01-Jun-2005','dd-Mon-yyyy'),' 10% discount coupon');
```

3. Allison Plumb, the event planning manager for DJs on Demand, has just given you the following list of CDs she acquired from a company going out of business. She wants a new updated list of CDs in inventory in an hour, but she doesn't want the original D_CDS table changed. Prepare an updated inventory list just for her.

   a. Assign new cd_numbers to each new CD acquired.

   b. Create a copy of the D_CDS table called manager_copy_d_cds. What was the correct syntax used?

   c. INSERT into the manager_copy_d_cds table each new CD title using an INSERT statement. Make up one example or use this data:
      20, 'Hello World Here I Am', 'Middle Earth Records', '1998'
      What was the correct syntax used?

   d. Use a merge statement to add to the manager_copy_d_cds table, the CDs from the original table. If there is a match, update the title and year. If not, insert the data from the original table. What was the correct syntax used?

```
b)CREATE TABLE manager_copy_d_cds
AS ( SELECT * FROM d_cds);
DESCRIBE d_cds;
DESCRIBE manager_copy_d_cds;
SELECT * FROM d_cds;
SELECT * FROM manager_copy_d_cds;
INSERT INTO manager_copy_d_cds(cd_number,title,producer,year)
VALUES(20,'Hello World Here I Am','Middle Earth Records','1998');
SELECT * FROM manager_copy_d_cds
MERGE INTO manager_copy_d_cds tgt USING d_cds src
ON (src.cd_number = tgt.cd_number)
WHEN MATCHED THEN UPDATE
SET  tgt.title = src.title, tgt.producer = src.producer, tgt.year =
src.year        *la final SELECT * FROM manager_copy_d_cds ;
WHEN NOT MATCHED THEN INSERT
VALUES (src.cd_number, src.title, src.producer, src.year);
```

4. Run the following 3 statements to create 3 new tables for use in a Multi-table insert statement. All 3 tables should be empty on creation, hence the WHERE 1=2 condition in the WHERE clause.

```
CREATE TABLE sal_history (employee_id, hire_date, salary)
AS SELECT employee_id, hire_date, salary
  FROM   employees
  WHERE  1=2;

CREATE TABLE mgr_history (employee_id, manager_id, salary)
AS SELECT employee_id, manager_id, salary
  FROM   employees
  WHERE  1=2;

CREATE TABLE special_sal (employee_id, salary)
AS SELECT employee_id, salary
  FROM   employees
  WHERE  1=2;
```

```
INSERT FIRST
WHEN salary > 20000 THEN
INTO special_sal
VALUES(employee_id, salary)
ELSE
INTO sal_history
VALUES(employee_id, hire_date, salary)
INTO mgr_history
VALUES(employee_id, manager_id, salary)

SELECT employee_id, salary, hire_date,
manager_id
FROM employees;
```

Once the tables exist in your account, write a Multi-Table insert statement to first select the employee_id, hire_date, salary, and manager_id of all employees. If the salary is more than 20000 insert the employee_id and salary into the special_sal table. Insert the details of employee_id, hire_date, and salary into the sal_history table. Insert the employee_id, manager_id, and salary into the mgr_history table.

You should get a message back saying 39 rows were inserted. Verify you get this message and verify you have the following number of rows in each table:

Sal_history: 19 rows
Mgr_history: 19 rows
Special_sal: 1

```
SELECT COUNT(*) FROM special_sal;
gives the response 1
SELECT COUNT(*) FROM sal_history;
gives the response 19
SELECT COUNT(*) FROM mgr_history;
 gives the response 19
```