

Database Programming with SQL

* 1. : 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.**

1. 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:
   1. 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)

**CREATE TABLE copy\_f\_promotional\_menus**

**AS ( SELECT \* FROM f\_promotional\_menus);**

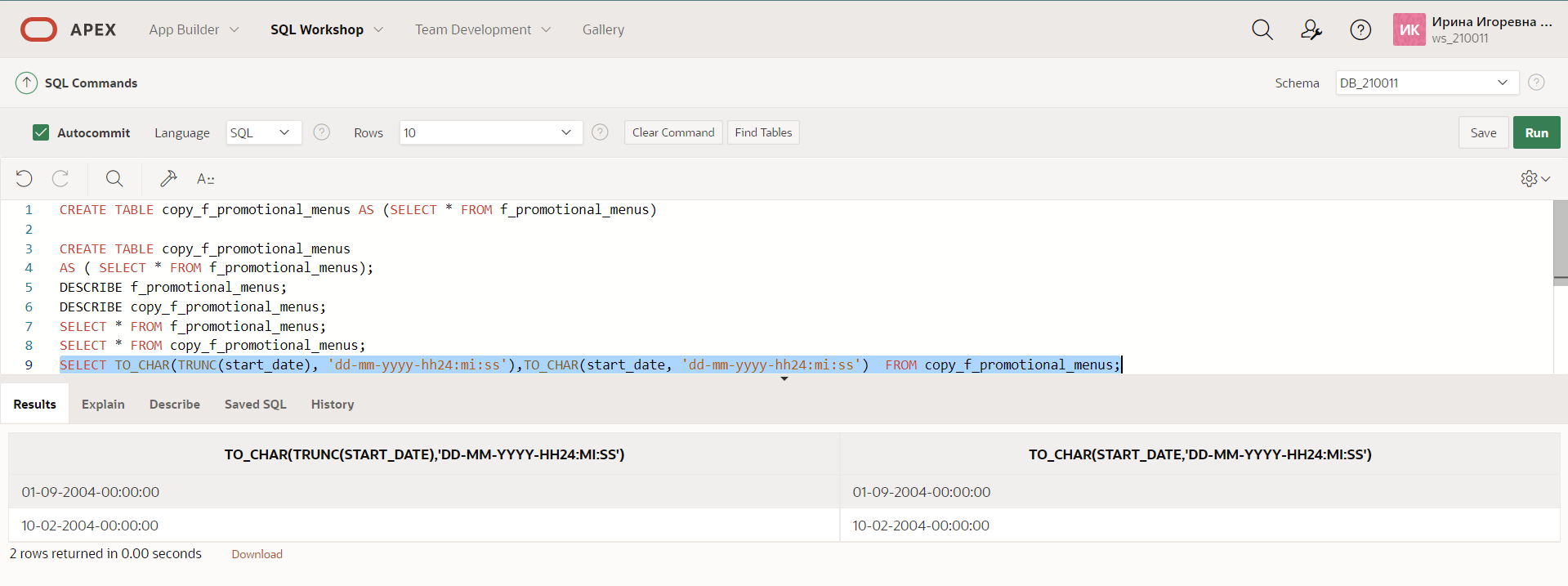
DESCRIBE f\_promotional\_menus;

DESCRIBE copy\_f\_promotional\_menus;

SELECT \* FROM f\_promotional\_menus;

SELECT \* FROM copy\_f\_promotional\_menus;

SELECT TO\_CHAR(TRUNC(start\_date), 'dd-mm-yyyy-hh24:mi:ss'),TO\_CHAR(start\_date, 'dd-mm-yyyy-hh24:mi:ss')  FROM copy\_f\_promotional\_menus;



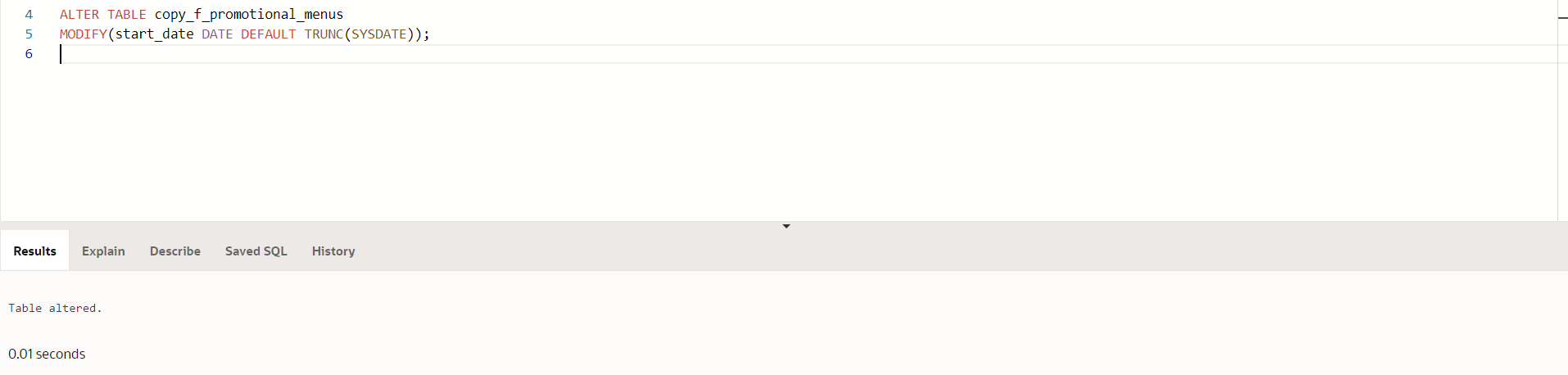
* 1. Alter the current START\_DATE column attributes using:

ALTER TABLE copy\_f\_promotional\_menus MODIFY(start\_date DATE DEFAULT SYSDATE)

**Give default vale to start\_date:**

**ALTER TABLE copy\_f\_promotional\_menus**

**MODIFY(start\_date DATE DEFAULT TRUNC(SYSDATE));**



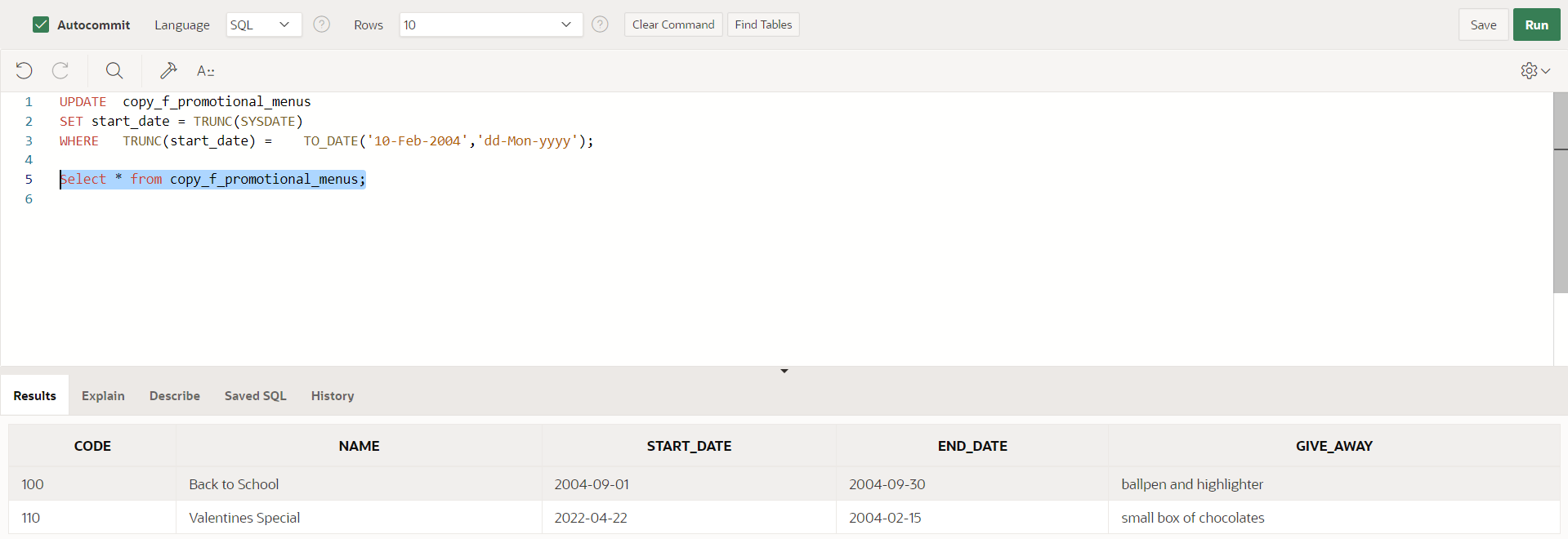
* 1. INSERT the new information and check to verify the results.

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 = TRUNC(SYSDATE)**

**WHERE   TRUNC(start\_date) =    TO\_DATE('10-Feb-2004','dd-Mon-yyyy');**



For new rows, simply skip giving any value to this field or say DEFAULT:

**INSERT INTO copy\_f\_promotional\_menus(code,name,start\_date,end\_date,give\_away)**

**VALUES('115','Back to School part 2',DEFAULT,NULL,'ballpen and highlighter again');**

or

**INSERT INTO copy\_f\_promotional\_menus(code,name,end\_date,give\_away)**

**VALUES('116','Back to School part 3',NULL,'ballpen and highlighter again2');**

**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');**

1. 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.
   1. Assign new cd\_numbers to each new CD acquired.

It seems to be, this cd\_number assignment is being done manually, I need not create a sequence for this. If the sequence had to be created, this point would have come after point b below, original table don’ have a sequence on this column.

* 1. Create a copy of the D\_CDS table called manager\_copy\_d\_cds. What was the correct syntax used?

**CREATE TABLE manager\_copy\_d\_cds**

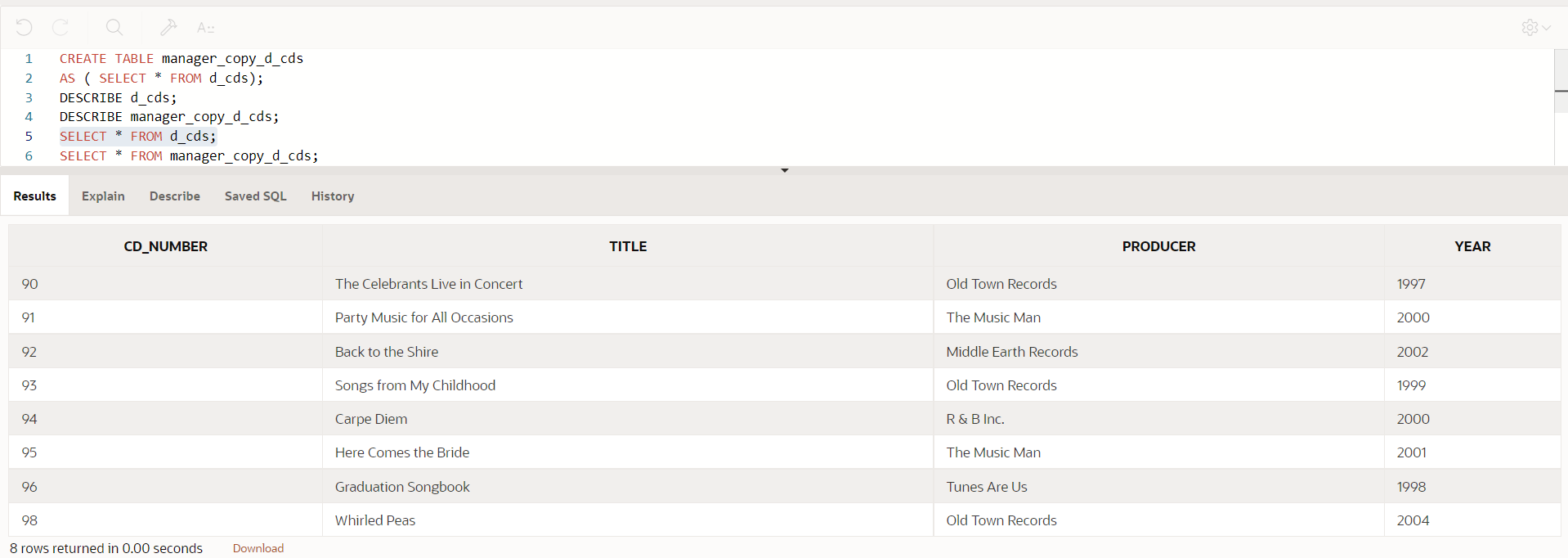
**AS ( SELECT \* FROM d\_cds);**

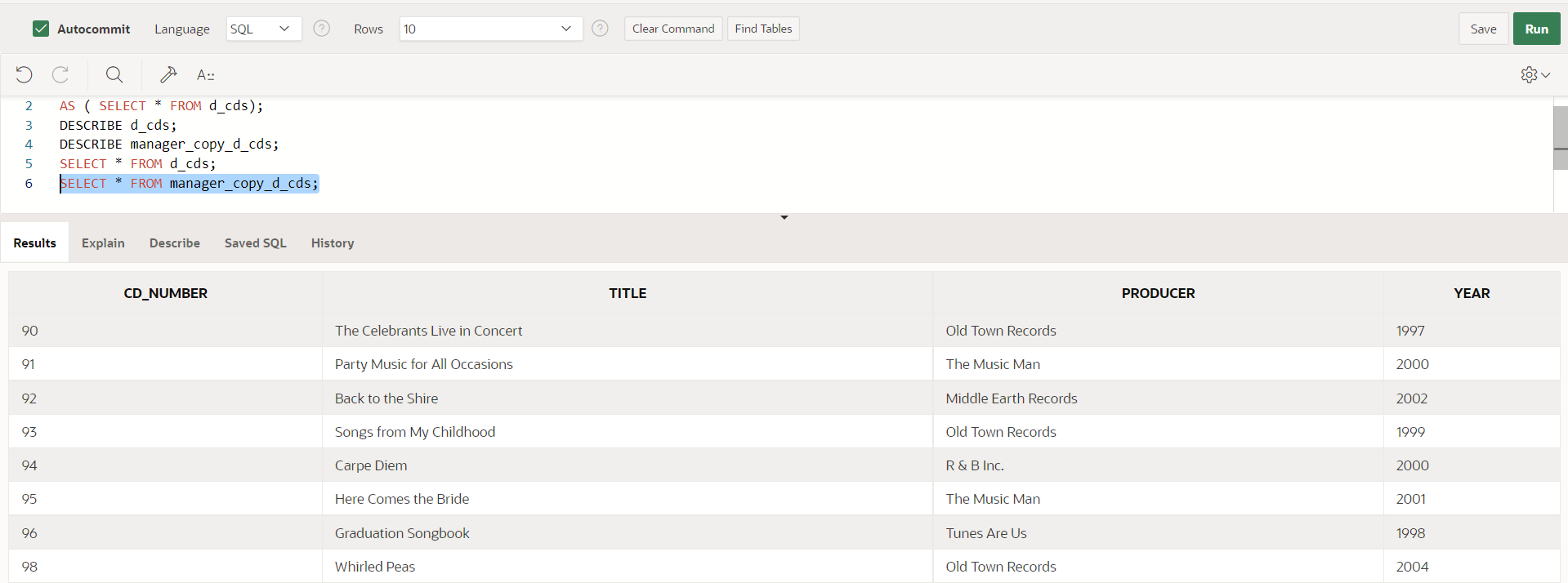
DESCRIBE d\_cds;

DESCRIBE manager\_copy\_d\_cds;

SELECT \* FROM d\_cds;

SELECT \* FROM manager\_copy\_d\_cds;





* 1. 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?

**INSERT INTO manager\_copy\_d\_cds(cd\_number,title,producer,year)**

**VALUES(20,'Hello World Here I Am','Middle Earth Records','1998');**

**INSERT INTO manager\_copy\_d\_cds(cd\_number,title,producer,year)**

**VALUES(97,'Celebrate the Day','R & B Inc.','2003');**

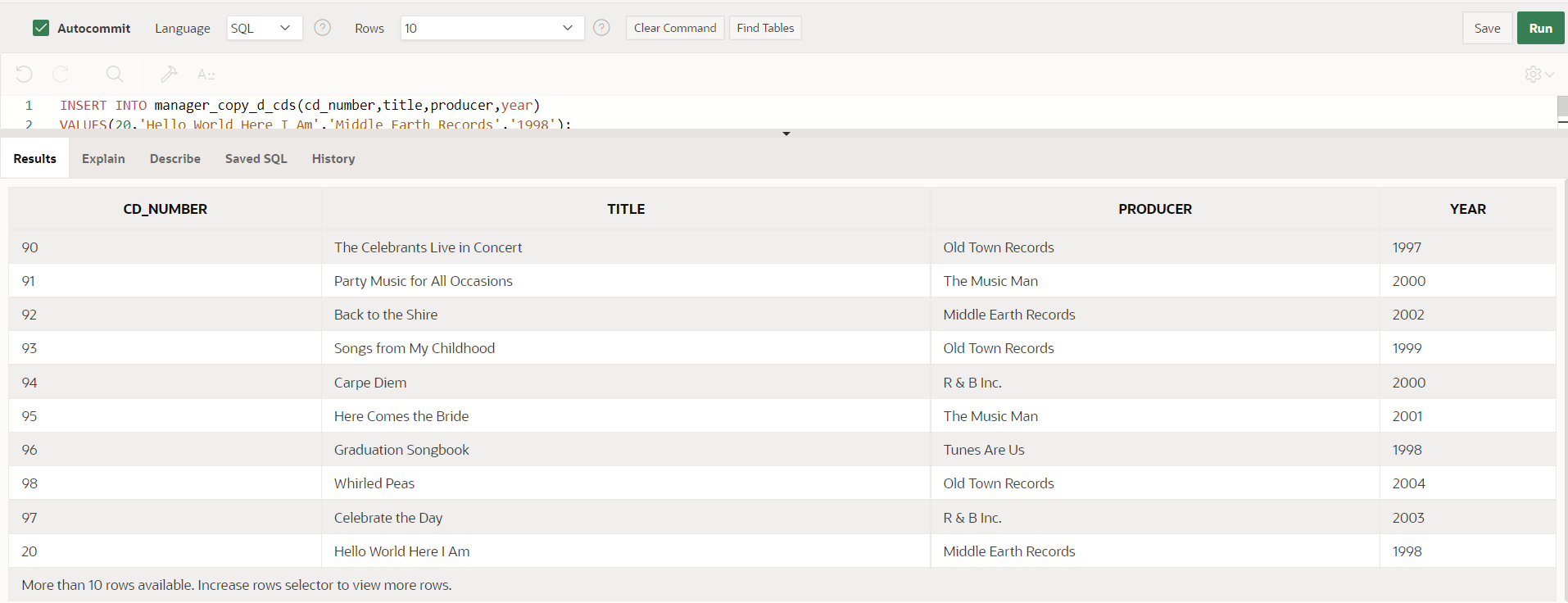
**INSERT INTO manager\_copy\_d\_cds(cd\_number,title,producer,year)**

**VALUES(99,'Party Music','Old Town Records','2004');**

**INSERT INTO manager\_copy\_d\_cds(cd\_number,title,producer,year)**

**VALUES(100,'Best of Rock and Roll','Old Town Records','2004');**

SELECT \* FROM manager\_copy\_d\_cds ;



* 1. 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?

To verify merge, first

i) I need to edit some record in manager\_copy\_d\_cds, this should get updated from d\_cds to original value.

UPDATE manager\_copy\_d\_cds

SET title = 'hkumar'

WHERE cd\_number = 90;

ii) I should delete some record in manager\_copy\_d\_cds which is present in d\_cds. This should be recreated after merge.

DELETE FROM manager\_copy\_d\_cds

WHERE cd\_number = 91;

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**

**WHEN NOT MATCHED THEN INSERT**

**VALUES (src.cd\_number, src.title, src.producer, src.year);**

If () is missing I will get: ORA-00969: missing ON keyword

SELECT \* FROM manager\_copy\_d\_cds ;

1. 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;

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

If I use FISRT / ALL no difference, since there is only one ‘WHEN’ I am using. In else there are 2 inserts.

**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;**

39 row(s) inserted.

SELECT COUNT(\*) FROM special\_sal;

1

SELECT COUNT(\*) FROM sal\_history;

19

SELECT COUNT(\*) FROM mgr\_history;

19

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.