

Database Programming with SQL

* 1. : Updating Column Values and Deleting Rows Practice Activities

# Objectives

* + - Construct and execute an UPDATE statement
    - Construct and execute a DELETE statement
    - Construct and execute a query that uses a subquery to update and delete data from a table
    - Construct and execute a query that uses a correlated subquery to update and delete from a table
    - Explain how foreign-key and primary-key integrity constraints affect UPDATE and DELETE statements
    - Explain the purpose of the FOR UPDATE Clause in a SELECT statement

# Vocabulary

Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **UPDATE** | Modifies existing rows in a table |
| **correlated subquery update** | retrieves information from one table & uses the information to update another table |
| **Integrity Constraints** | Ensures that the data adheres to a predefined set of rules |
| **correlated subquery delete** | deletes information on a linked table based on what was deleted on the other table |
| **DELETE** | Removes existing rows from a table |

# Try It / Solve It

**NOTE: Copy tables in this section do not exist**

If any change is not possible, give an explanation as to why it is not possible.

1. Monique Tuttle, the manager of Global Fast Foods, sent a memo requesting an immediate change in prices. The price for a strawberry shake will be raised from $3.59 to $3.75, and the price for fries will increase to $1.20. Make these changes to the copy\_f\_food\_items table.

CREATE TABLE copy\_f\_food\_items

AS ( SELECT \* FROM f\_food\_items);

DESCRIBE f\_food\_items;

DESCRIBE copy\_f\_food\_items;

SELECT \* FROM f\_food\_items;

SELECT \* FROM copy\_f\_food\_items;

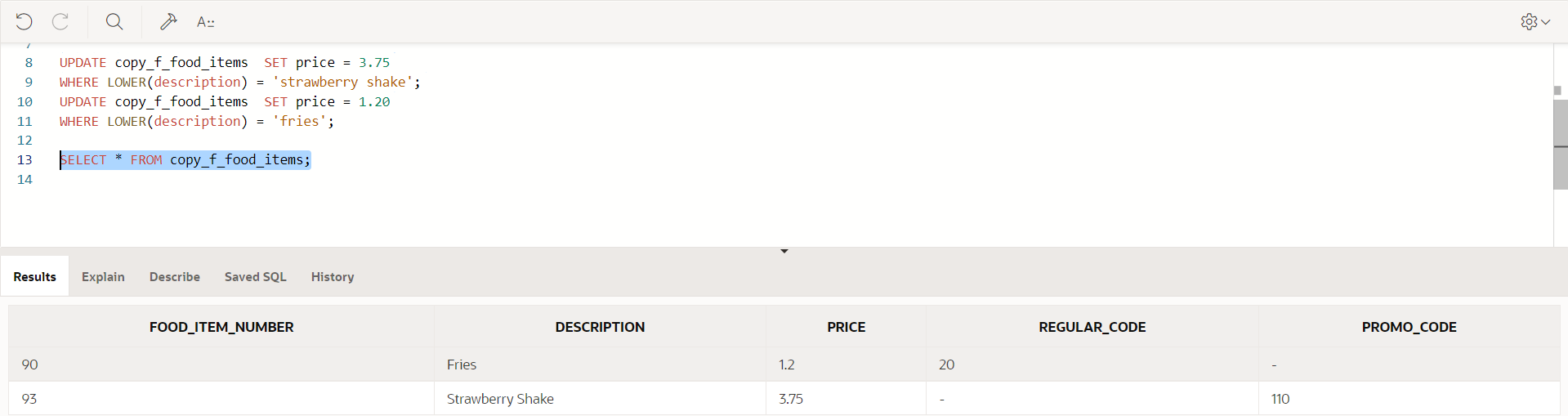
**UPDATE copy\_f\_food\_items  SET price = 3.75**

**WHERE LOWER(description) = 'strawberry shake';**

**UPDATE copy\_f\_food\_items  SET price = 1.20**

**WHERE LOWER(description) = 'fries';**

SELECT \* FROM copy\_f\_food\_items;



1. Bob Miller and Sue Doe have been outstanding employees at Global Fast Foods. Management has decided to reward them by increasing their overtime pay. Bob Miller will receive an additional

$0.75 per hour and Sue Doe will receive an additional $0.85 per hour. Update the copy\_f\_staffs table to show these new values. (Note: Bob Miller currently doesn’t get overtime pay. What function do you need to use to convert a null value to 0?)

CREATE TABLE copy\_f\_staffs

AS ( SELECT \* FROM f\_staffs);

DESCRIBE f\_staffs;

DESCRIBE copy\_f\_staffs;

SELECT \* FROM f\_staffs;

SELECT \* FROM copy\_f\_staffs;

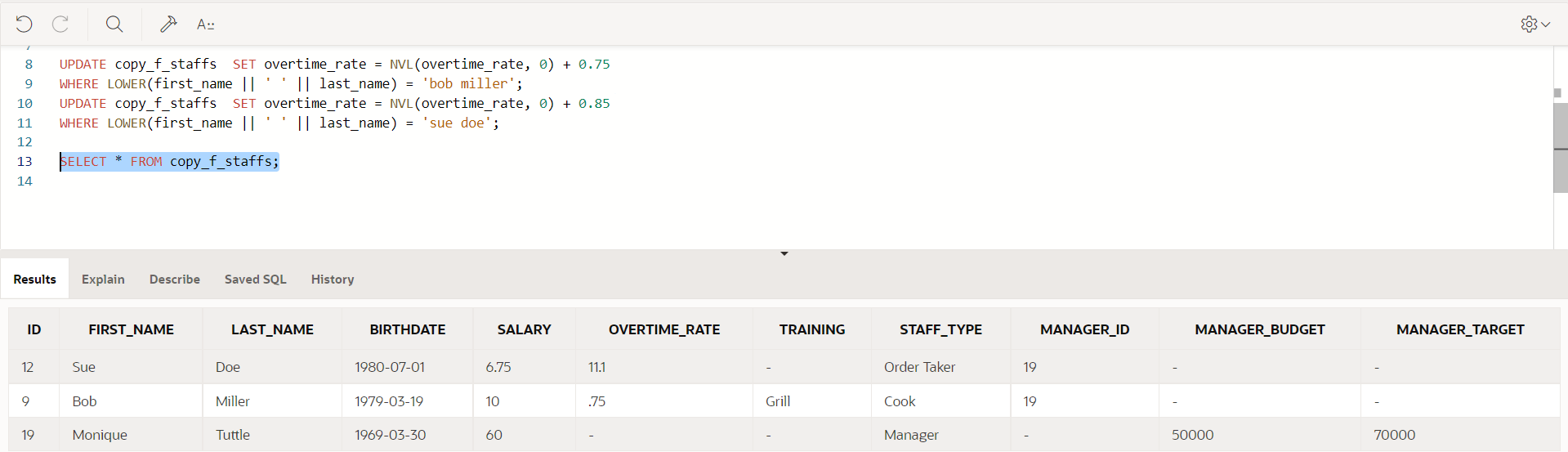
**UPDATE copy\_f\_staffs  SET overtime\_rate = NVL(overtime\_rate, 0) + 0.75**

**WHERE LOWER(first\_name || ' ' || last\_name) = 'bob miller';**

**UPDATE copy\_f\_staffs  SET overtime\_rate = NVL(overtime\_rate, 0) + 0.85**

**WHERE LOWER(first\_name || ' ' || last\_name) = 'sue doe';**

SELECT \* FROM copy\_f\_staffs;



1. Add the orders shown to the Global Fast Foods copy\_f\_orders table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ORDER\_NUMBER | ORDER\_DATE | ORDER\_TOTAL | CUST\_ID | STAFF\_ID |
| 5680 | June 12, 2004 | 159.78 | 145 | 9 |
| 5691 | 09-23-2004 | 145.98 | 225 | 12 |
| 5701 | July 4, 2004 | 229.31 | 230 | 12 |

CREATE TABLE copy\_f\_orders

AS ( SELECT \* FROM f\_orders);

DESCRIBE f\_orders;

DESCRIBE copy\_f\_orders;

SELECT \* FROM f\_orders;

SELECT \* FROM copy\_f\_orders;

**INSERT INTO copy\_f\_orders(order\_number,order\_date,order\_total,cust\_id,staff\_id)**

**VALUES(5680,TO\_DATE('June 12, 2004','fmMonth dd, yyyy'),159.78,145,9);**

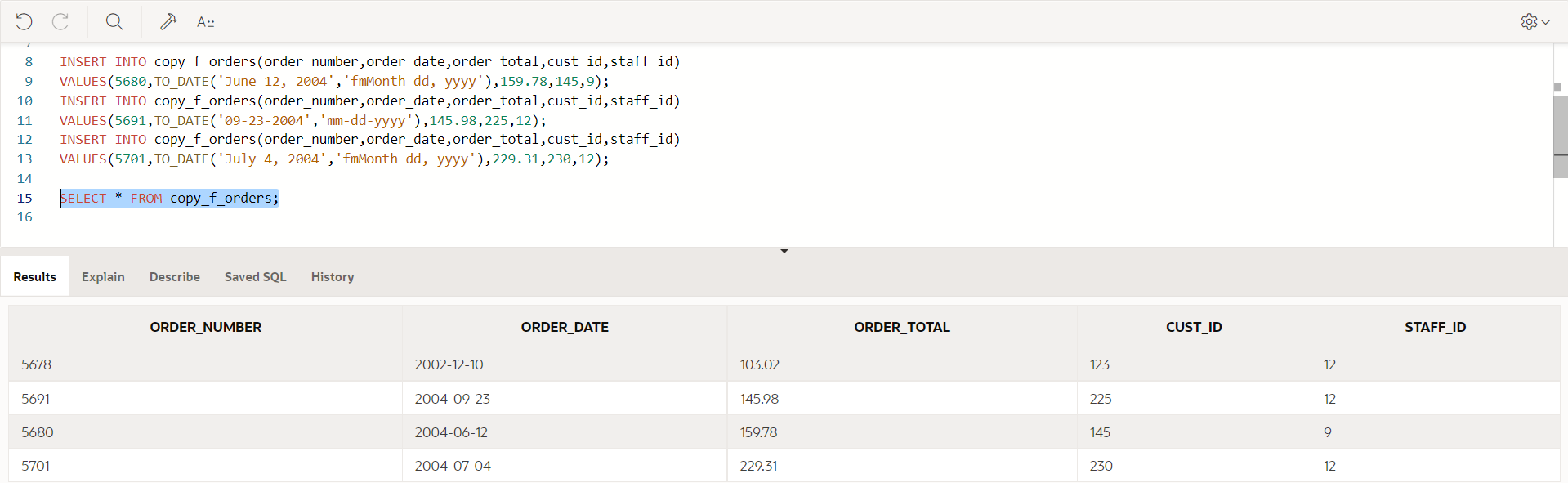
**INSERT INTO copy\_f\_orders(order\_number,order\_date,order\_total,cust\_id,staff\_id)**

**VALUES(5691,TO\_DATE('09-23-2004','mm-dd-yyyy'),145.98,225,12);**

**INSERT INTO copy\_f\_orders(order\_number,order\_date,order\_total,cust\_id,staff\_id)**

**VALUES(5701,TO\_DATE('July 4, 2004','fmMonth dd, yyyy'),229.31,230,12);**

SELECT \* FROM copy\_f\_orders;



1. Add the new customers shown below to the copy\_f\_customers table. You may already have added Katie Hernandez. Will you be able to add all these records successfully?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | FIRST\_ NAME | LAST\_ NAME | ADDRESS | CITY | STATE | ZIP | PHONE\_NUMBER |
| 145 | Katie | Hernandez | 92 Chico Way | Los Angeles | CA | 98008 | 8586667641 |
| 225 | Daniel | Spode | 1923  Silverado | Denver | CO | 80219 | 7193343523 |
| 230 | Adam | Zurn | 5 Admiral Way | Seattle | WA |  | 4258879009 |

Yes I will be able to add row, even if it has existing id, since in cloning table as mentioned below, primary key constraint is lost.

CREATE TABLE copy\_f\_customers

AS ( SELECT \* FROM f\_customers);

DESCRIBE f\_customers;

DESCRIBE copy\_f\_customers;

In copy table, zip is not nullable same as in master table, so the last row will give error while insert.

SELECT \* FROM f\_customers;

SELECT \* FROM copy\_f\_customers;

**INSERT INTO copy\_f\_customers(id,first\_name,last\_name,address,city,state,zip,phone\_number)**

**VALUES(145,'Katie','Hernandez','92 Chico Way','Los Angeles','CA',98008,'8586667641');**

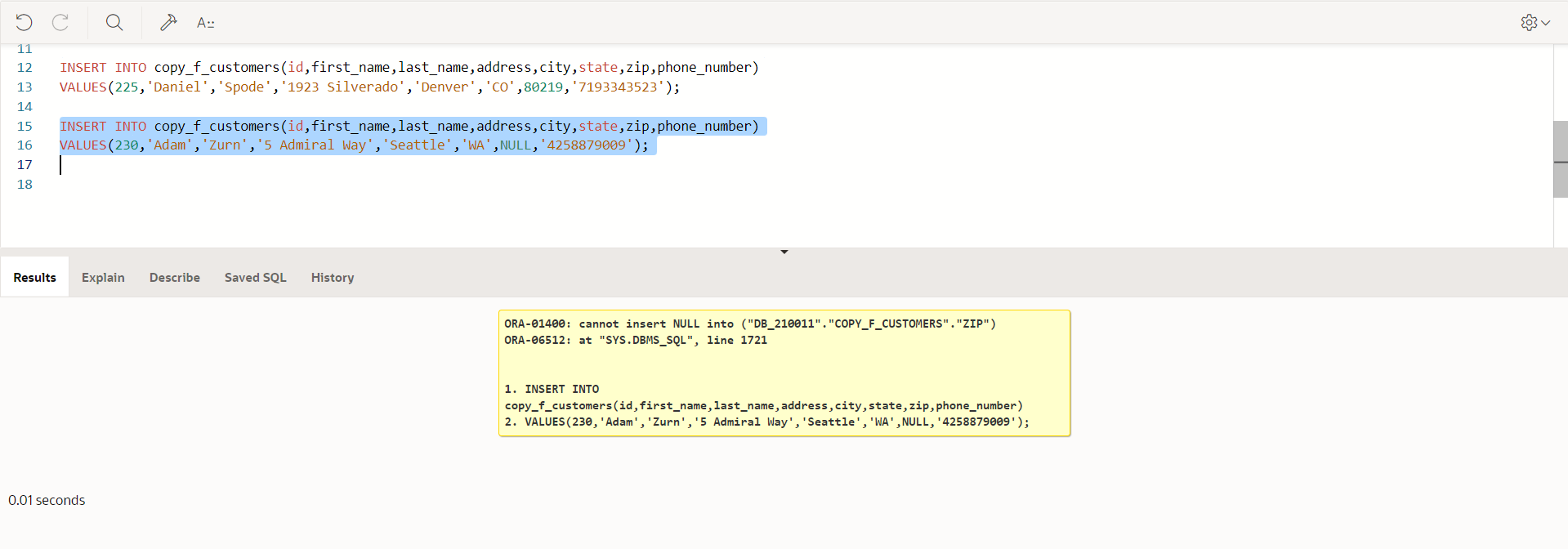
**INSERT INTO copy\_f\_customers(id,first\_name,last\_name,address,city,state,zip,phone\_number)**

**VALUES(225,'Daniel','Spode','1923 Silverado','Denver','CO',80219,'7193343523');**

**INSERT INTO copy\_f\_customers(id,first\_name,last\_name,address,city,state,zip,phone\_number)**

**VALUES(230,'Adam','Zurn','5 Admiral Way','Seattle','WA',NULL,'4258879009');**

ORA-01400: cannot insert NULL into ("HKUMAR"."COPY\_F\_CUSTOMERS"."ZIP")



1. Sue Doe has been an outstanding Global Foods staff member and has been given a salary raise. She will now be paid the same as Bob Miller. Update her record in copy\_f\_staffs.

**UPDATE copy\_f\_staffs  SET salary = (SELECT salary FROM copy\_f\_staffs WHERE LOWER(first\_name || ' ' || last\_name) = 'bob miller')**

**WHERE LOWER(first\_name || ' ' || last\_name) = 'sue doe';**

SELECT \* from copy\_f\_staffs;

1. Global Fast Foods is expanding their staff. The manager, Monique Tuttle, has hired Kai Kim. Not all information is available at this time, but add the information shown here.

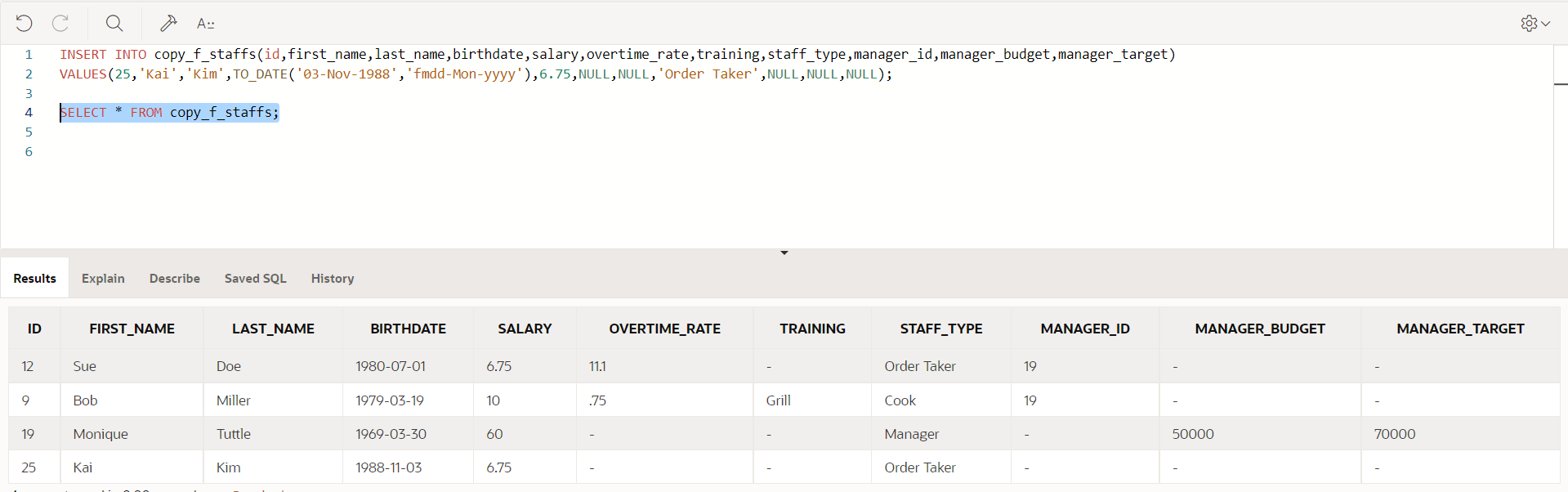
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | FIRST\_NAME | LAST\_NAME | BIRTHDATE | SALARY | STAFF\_TYPE |
| 25 | Kai | Kim | 3-Nov-1988 | 6.75 | Order Taker |

It should work since all the mandatory columns have values.

**INSERT INTO copy\_f\_staffs(id,first\_name,last\_name,birthdate,salary,overtime\_rate,training,staff\_type,manager\_id,manager\_budget,manager\_target)**

**VALUES(25,'Kai','Kim',TO\_DATE('03-Nov-1988','fmdd-Mon-yyyy'),6.75,NULL,NULL,'Order Taker',NULL,NULL,NULL);**

SELECT \* FROM copy\_f\_staffs;

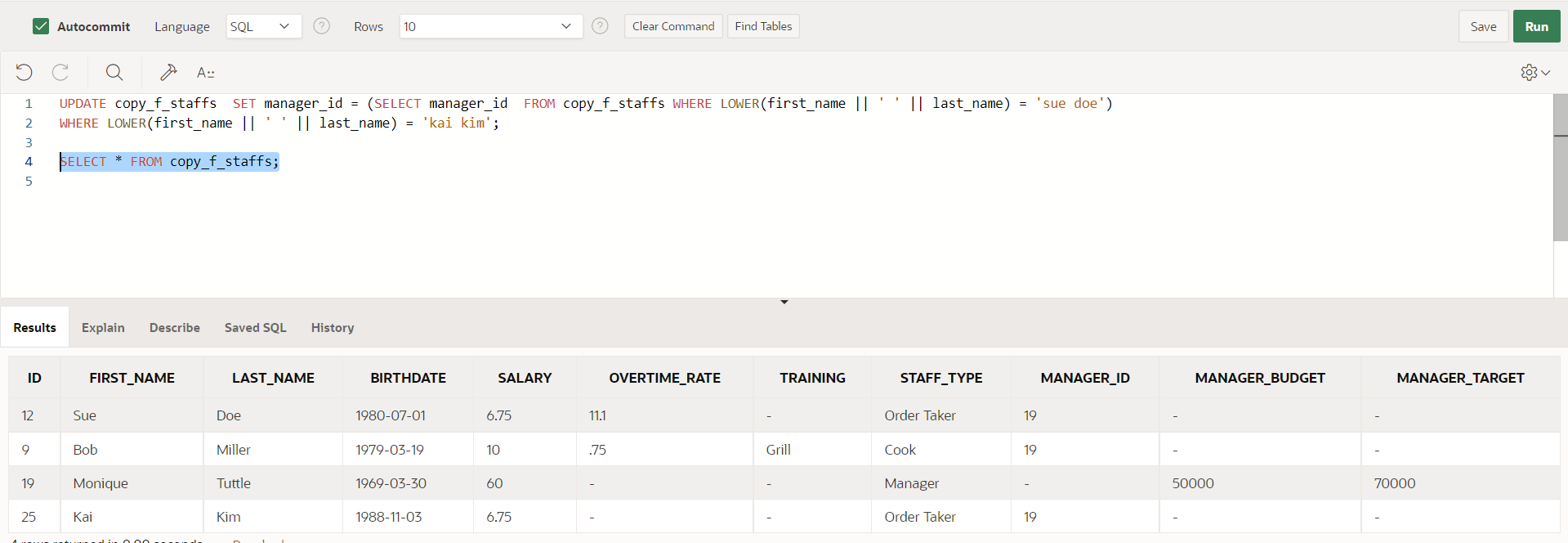


1. Now that all the information is available for Kai Kim, update his Global Fast Foods record to include the following: Kai will have the **same manager** as Sue Doe. He does not qualify for overtime. Leave the values for training, manager budget, and manager target as null.

**UPDATE copy\_f\_staffs  SET manager\_id = (SELECT manager\_id  FROM copy\_f\_staffs WHERE LOWER(first\_name || ' ' || last\_name) = 'sue doe')**

**WHERE LOWER(first\_name || ' ' || last\_name) = 'kai kim';**

SELECT \* FROM copy\_f\_staffs;



1. Execute the following SQL statement. Record your results.

DELETE from departments WHERE department\_id = 60;

**ORA-02292: integrity constraint (HKUMAR.EMP\_DEPT\_FK) violated - child record found**

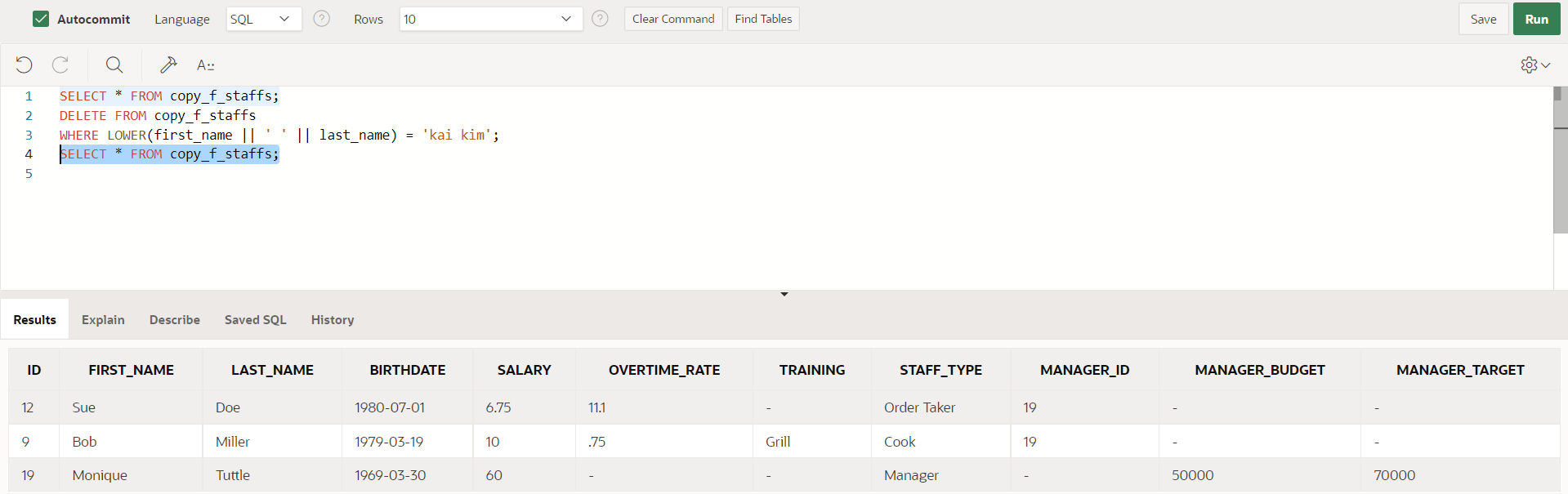
1. Kim Kai has decided to go back to college and does not have the time to work and go to school. Delete him from the Global Fast Foods staff. Verify that the change was made.

SELECT \* FROM copy\_f\_staffs;

**DELETE FROM copy\_f\_staffs**

**WHERE LOWER(first\_name || ' ' || last\_name) = 'kai kim';**

SELECT \* FROM copy\_f\_staffs;



1. Create a copy of the employees table and call it lesson7\_emp;

Once this table exists, write a correlated delete statement that will delete any employees from the lesson7\_employees table that also exist in the job\_history table.

**CREATE TABLE lesson7\_emp**

**AS ( SELECT \* FROM employees);**

DESCRIBE employees;

DESCRIBE lesson7\_emp;

SELECT \* FROM employees;

SELECT \* FROM lesson7\_emp;

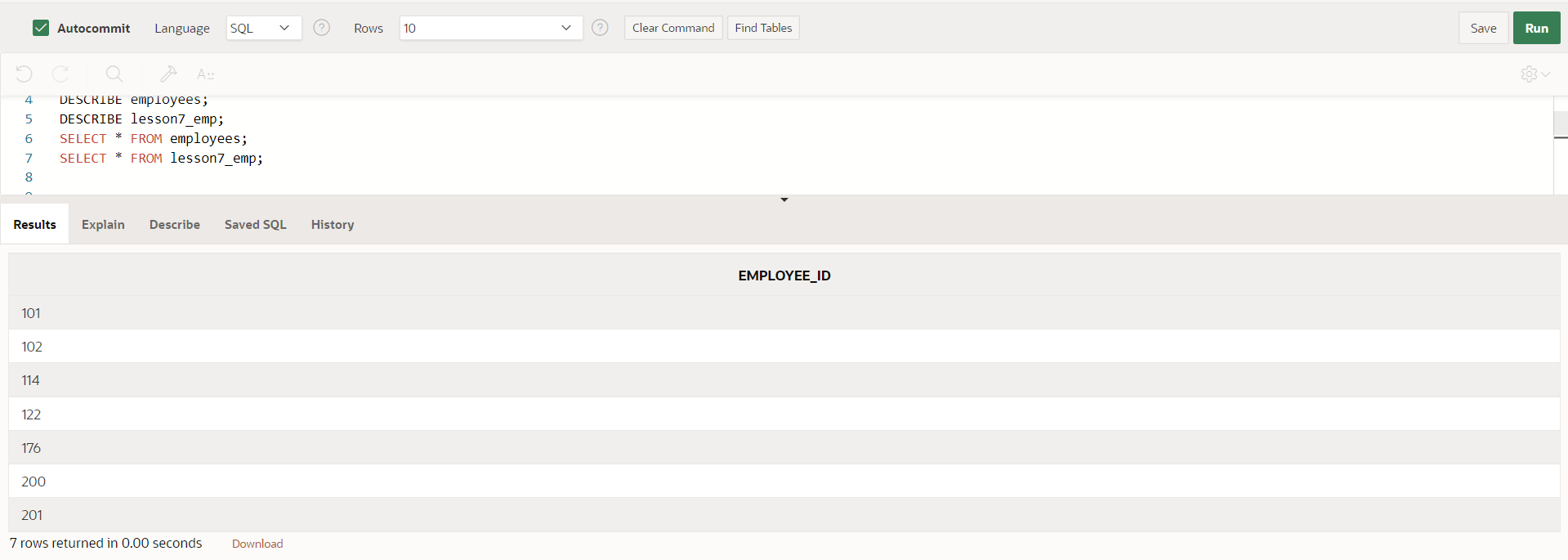
SELECT DISTINCT employee\_id FROM job\_history;

7 rows returned in 0.00 seconds

**DELETE FROM lesson7\_emp**

**WHERE employee\_id IN ( SELECT DISTINCT employee\_id FROM job\_history) ;**

**5 row(s) deleted.**



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.