**User Guide**

**Step 1:**

**Login to our schema**

Username: Team2

Password: Heinz19\_YRVJ

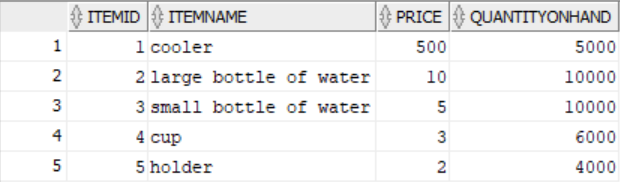
**Step 2:**

Run the script G2\_darksprings.sql file found in the folder “SQL” to create tables, views, sequences, procedures, functions, packages, jobs, roles, triggers, indexes, and to insert data into the schema.

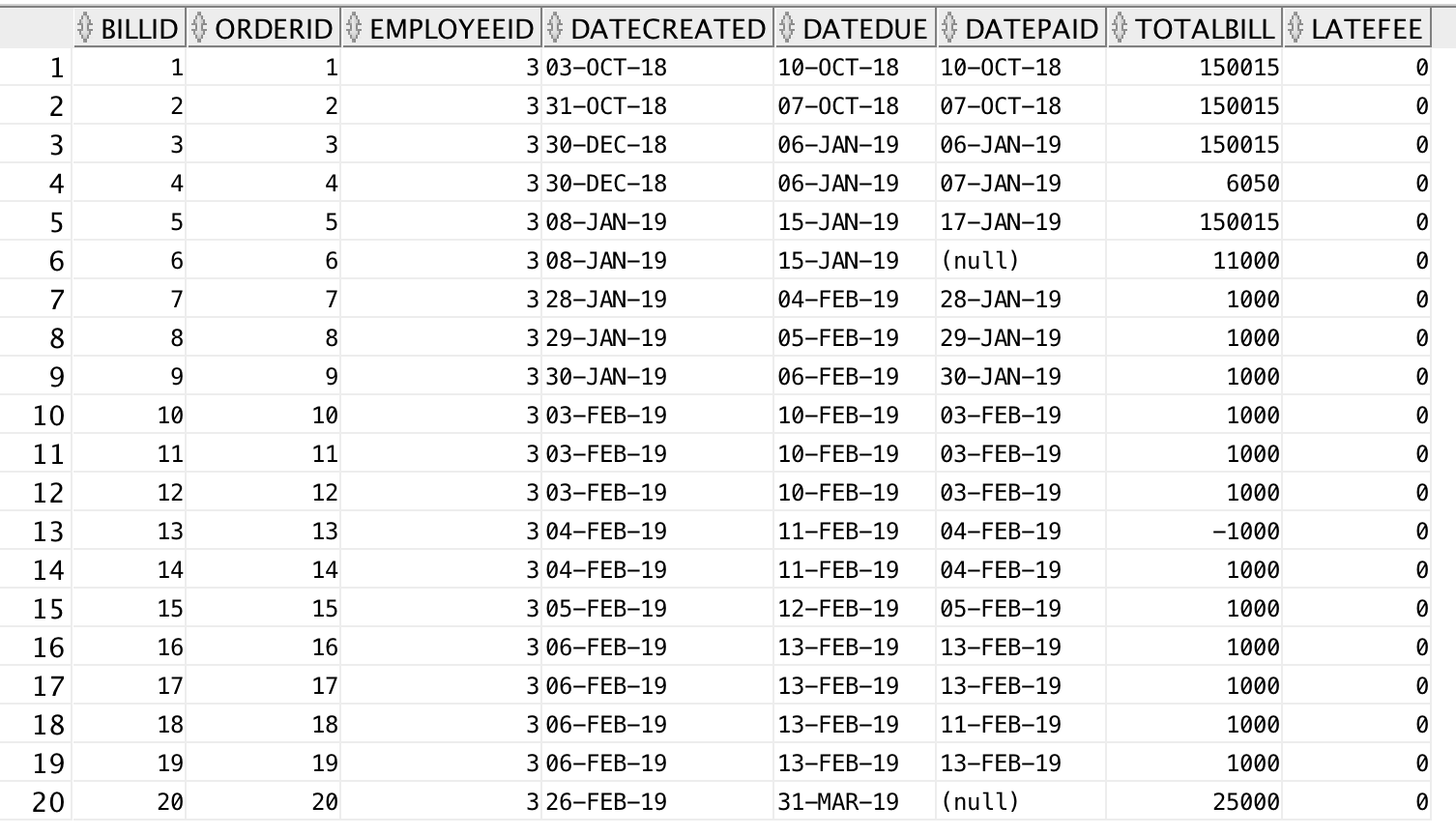
**Step 3:**

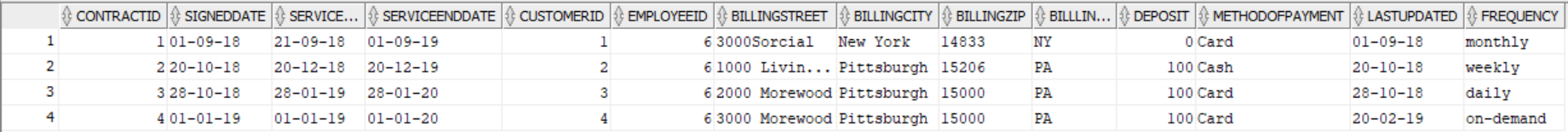
This shows the contents of all the tables in the schema

select \* from item;

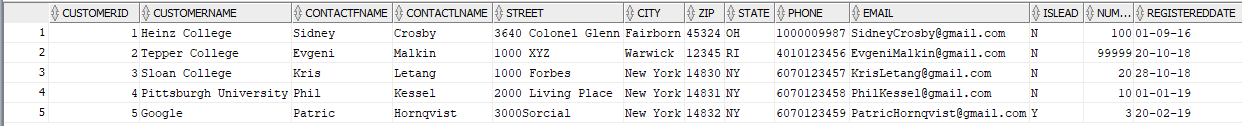
****

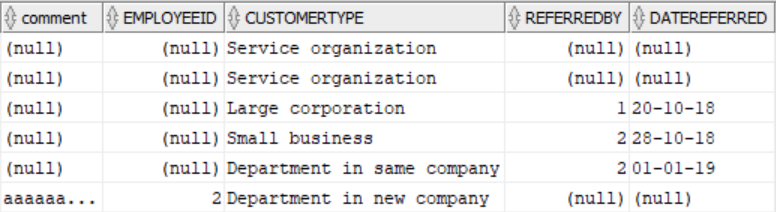
select \* from bill;

****

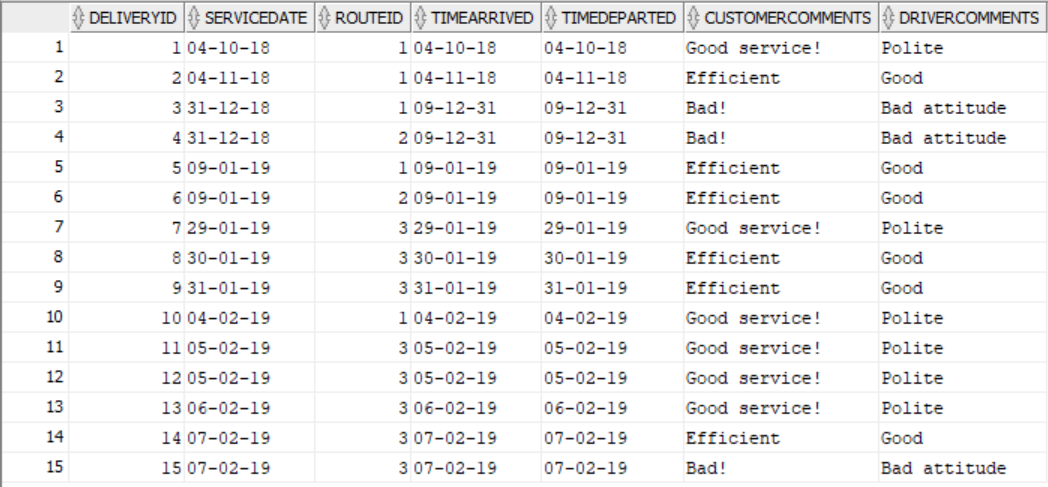
select \* from contract;****

select \* from customer;

****

****

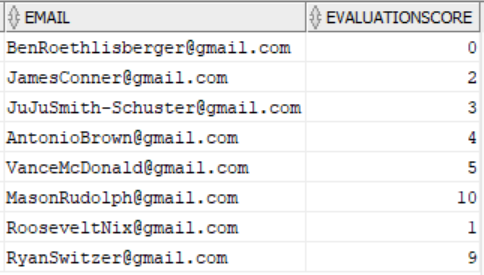
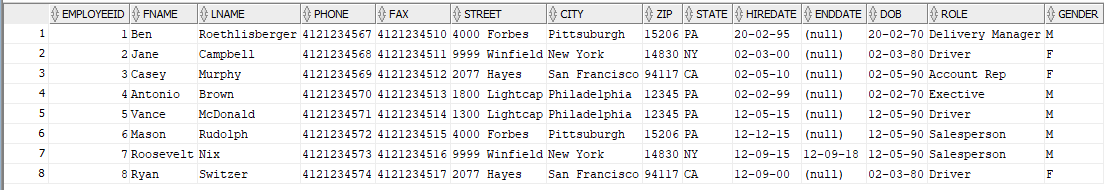
select \* from deliveryschedule;

****

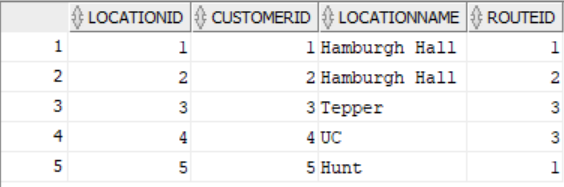
select \* from driver;

****

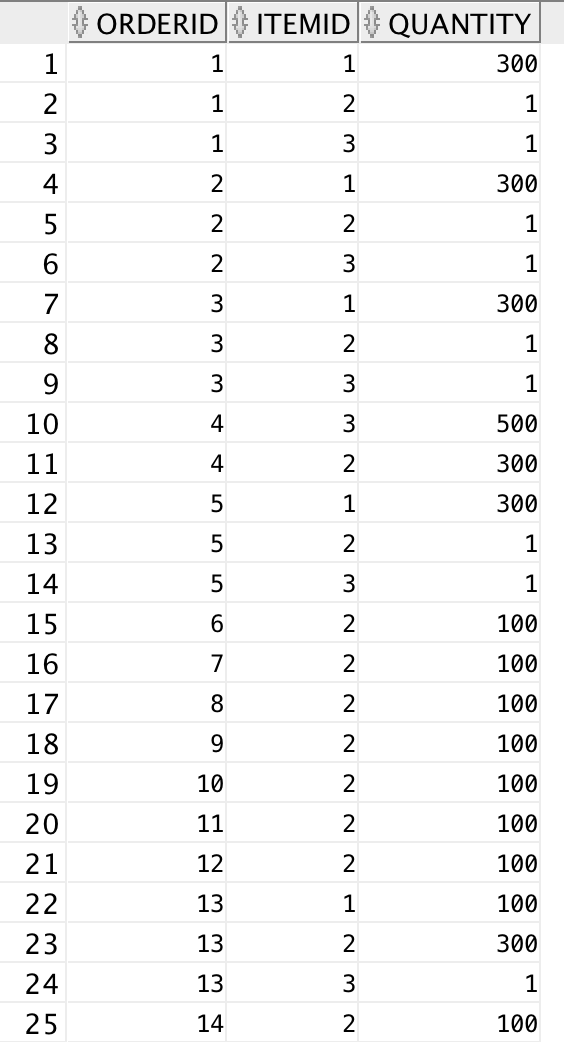
select \* from employee;

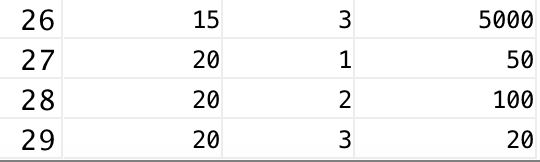
****

select \* from location;

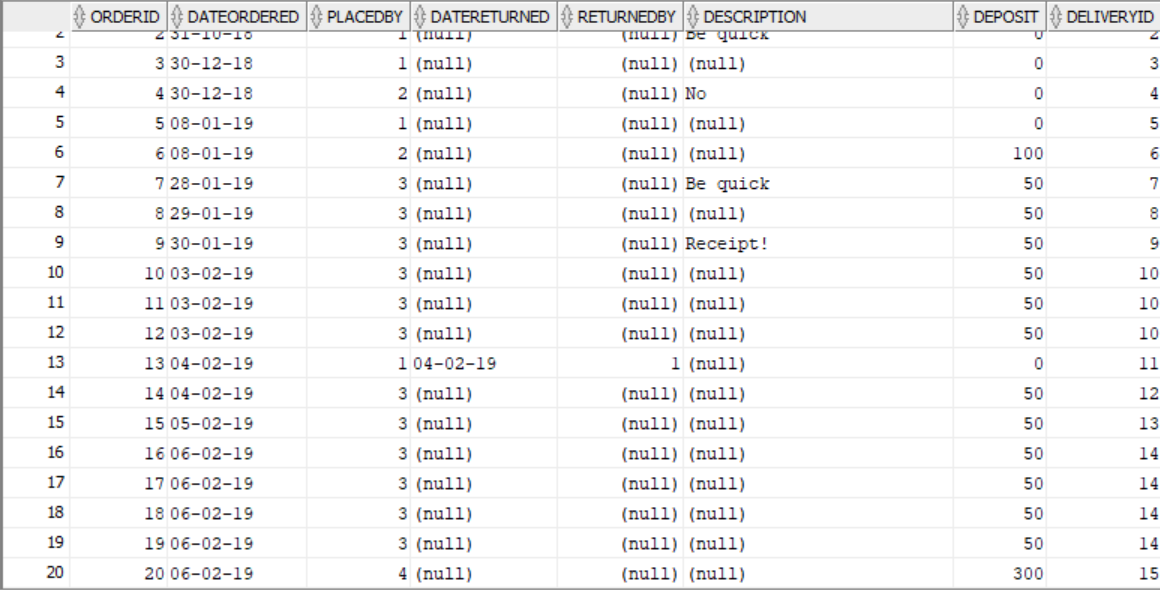
****

select \* from orderdetails;

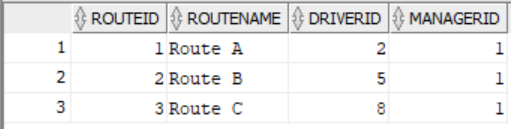
****



select \* from orders;

****

select \* from route;

****

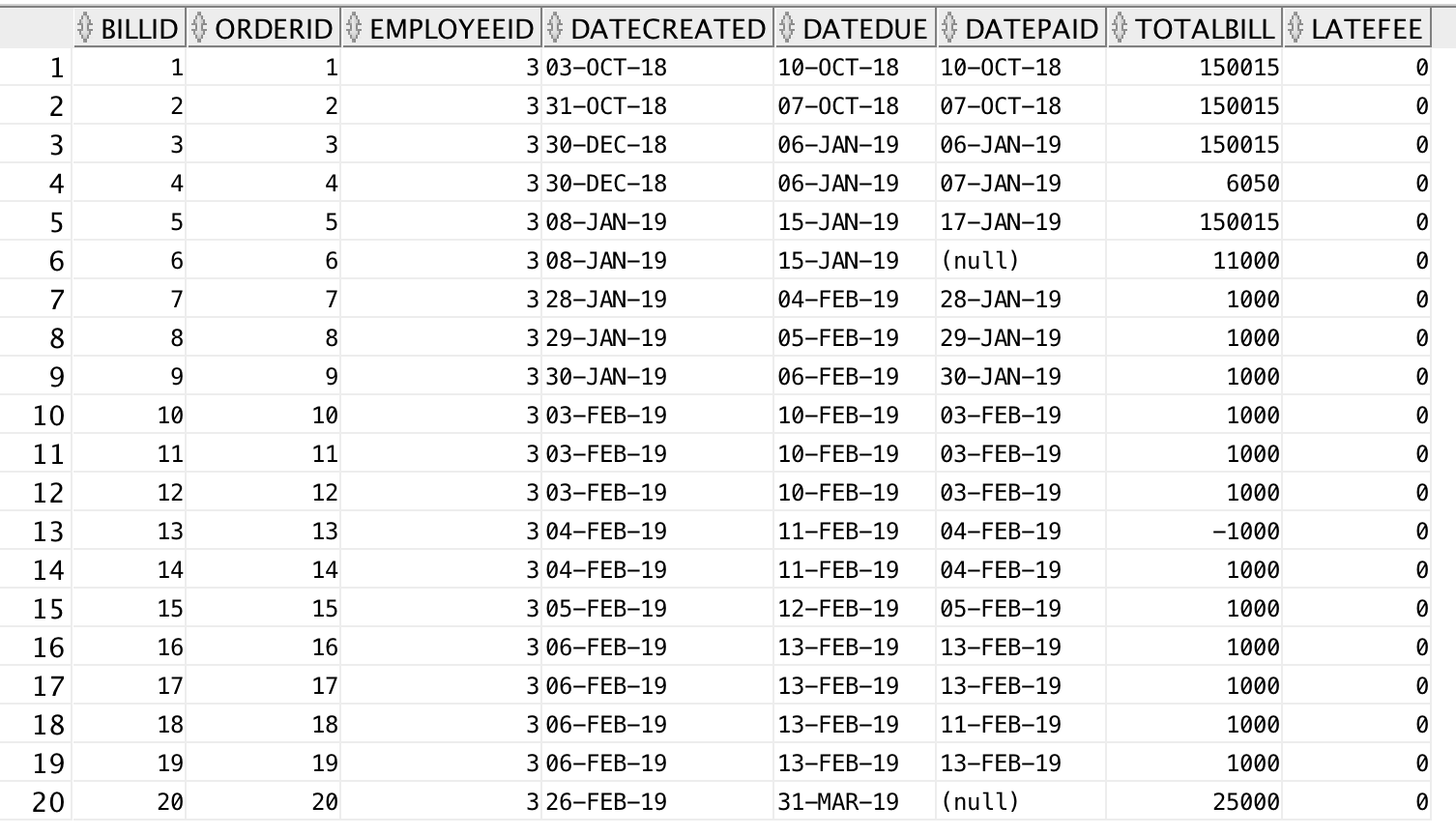
**Step 4:**

**Execute database objects:**

1. **Function**

* **Name:** calculate\_latefee
* Input: billid
* Output: amount of late fee for that bill
* Example:

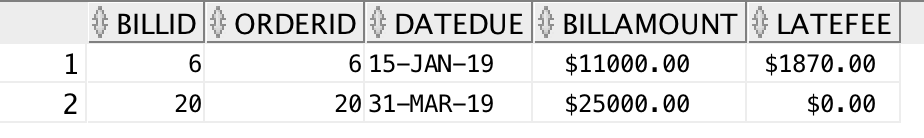
Bill Table



Query:

select billid,orderid,datedue,TO\_CHAR(totalbill,'$999990.99') AS BILLAMOUNT,TO\_CHAR(calculate\_latefee(billid),'$9990.99') as latefee from bill where billid in (select billid from bill where datepaid is null);

Result:

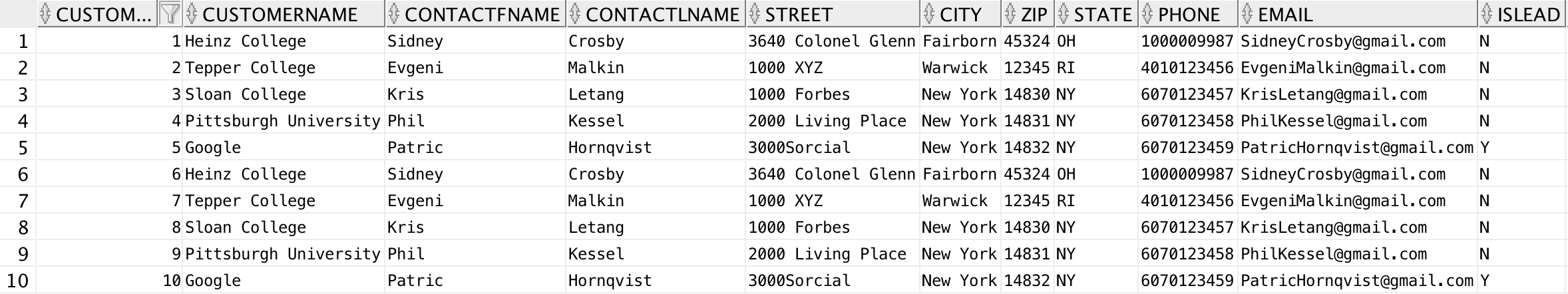


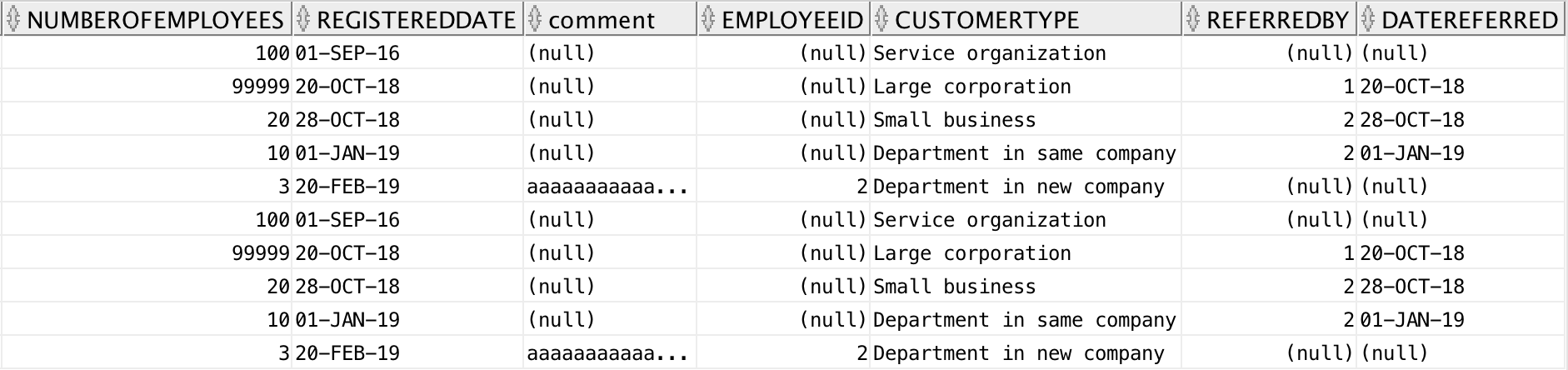
1. **View**

**2.1 customer\_view**

* Input: N/A
* Output: A table showing the customer ID, name, address, registered date, and customer type. (Does not show other personal information like the phone#, fax#, email, date of birth and gender for privacy reasons)
* Example:

Customer Table

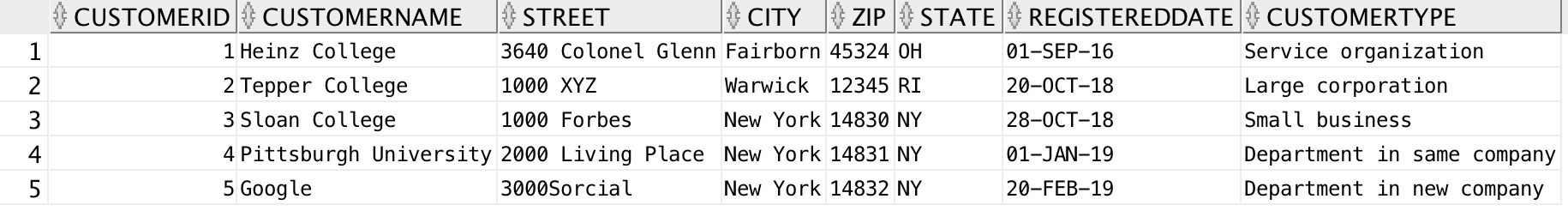
****

****

Query:

SELECT \* FROM customer\_view;

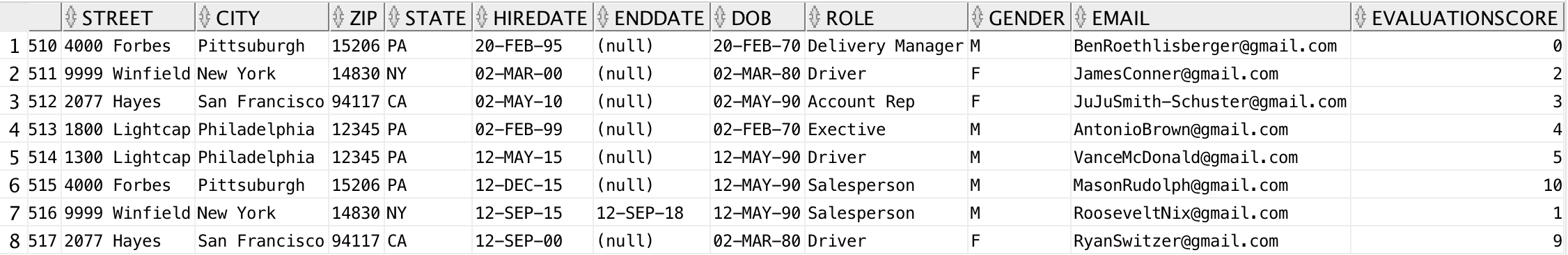
Result:

****

**2.2 employee\_view**

* Input: N/A
* output: A table showing the employee ID, name, email, and hire date. (Does not show other personal information like address, phone, fax, gender, and evaluation score for privacy reasons)
* Example:

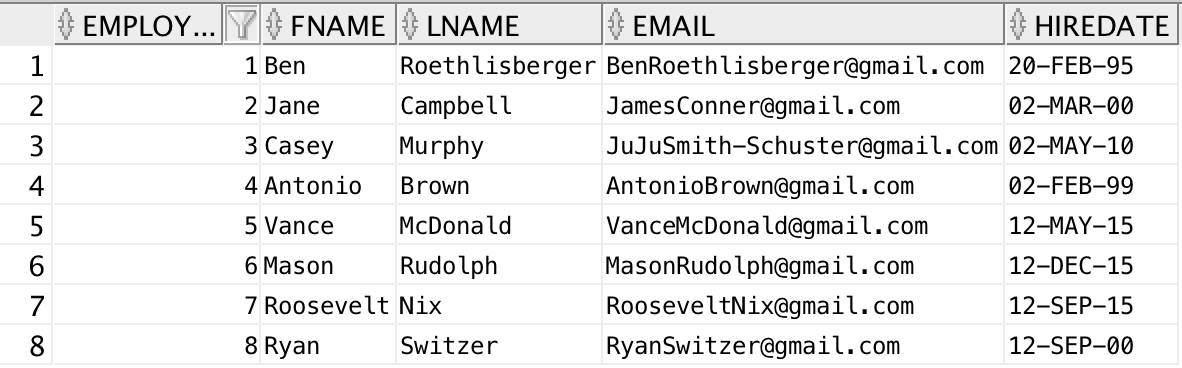
Employee Table

****

Query:

SELECT \* FROM employee\_view;

Result:

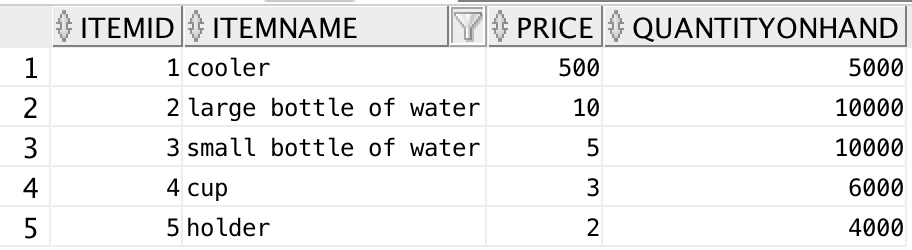
****

1. **Job**

**3.1 update\_inventory**

* Input: N/A
* output: add 5000 to quantity on hand in item table when the quantity goes below the threshold (6000)
* Example:

Item Table

****

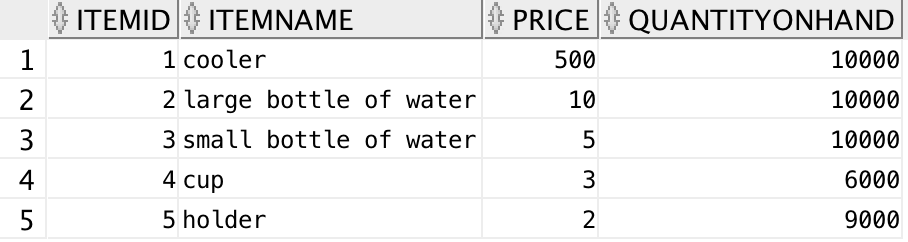
Query:

EXECUTE DBMS\_SCHEDULER.ENABLE('update\_inventory');

**Comment:** The job is scheduled to run for every 5 minutes. SO the following result would be available after a period of 5 minutes.

Result:

SELECT \* FROM item;

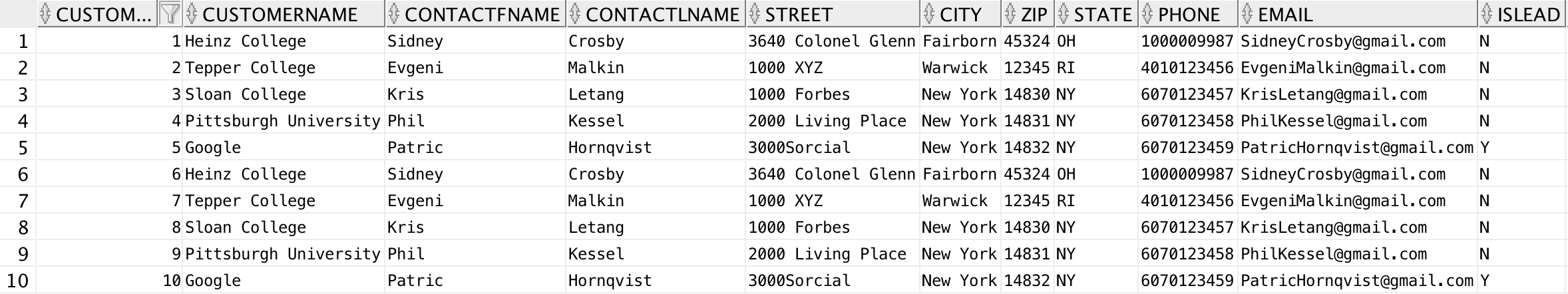


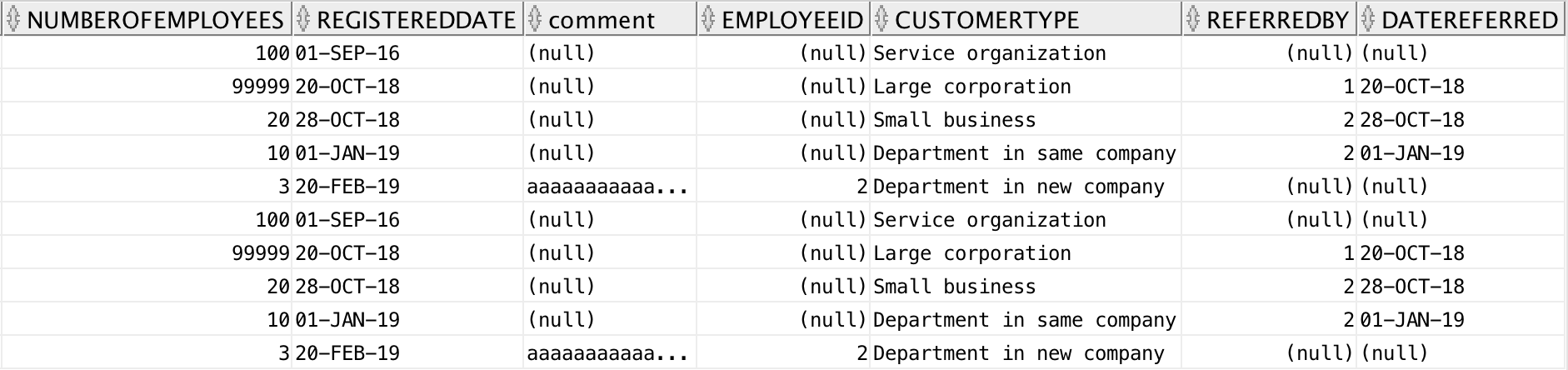
1. **Procedures**

**4.1 insert\_customer**

* **USE CASE:** adding a new customer
* Input: (customer name, contact first name, contact last name, street, city, zip, state, phone, email, islead, number of employees, registered date, comment, employee id, customer type, referred by, date referred)
* Output: a new customer record
* Example:

Customer Table

****

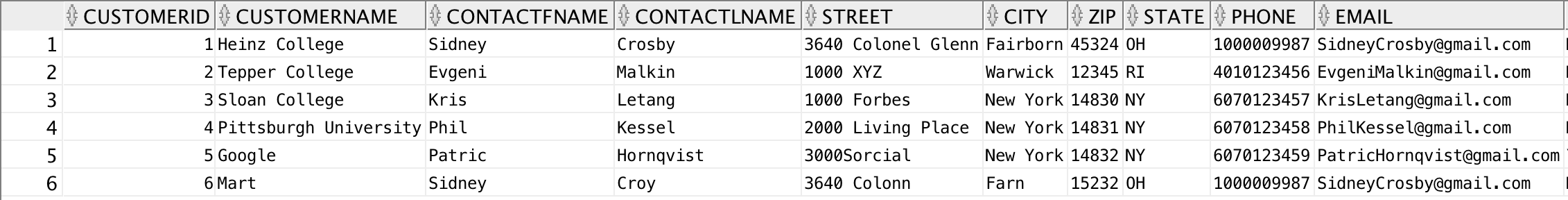
****

Query:

EXECUTE insert\_customer('Mart','Sidney','Croy','3640 Colonn','Farn','15232','OH','1000009987','SidneyCrosby@gmail.com','N',10,'20-Feb-15',NULL,NULL,'Service organization',NULL,NULL);

Result:

SELECT \* FROM customer;

****

**4.2 delivery\_schedule**

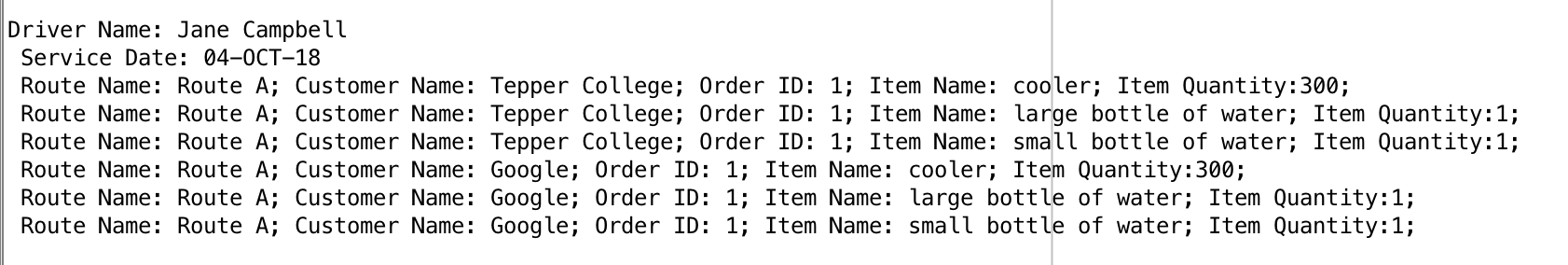
* Input: (driver’s first name, driver’s last name, service date)
* Output: a table showing the delivery schedule, including driver’s name, service date, route name, customer name, order id, item name, and item quantity
* Example:

Query:

SET SERVEROUTPUT ON

EXECUTE delivery\_schedule('Jane','Campbell','04-Oct-18');

Result:



Exception:

If input parameters are not in the tables or do not match, it will raise an exception

Try:

EXECUTE delivery\_schedule('Jan','Campbell','04-Oct-18');

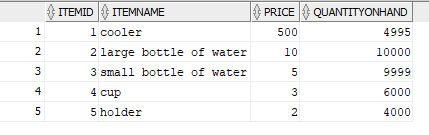
It will show:



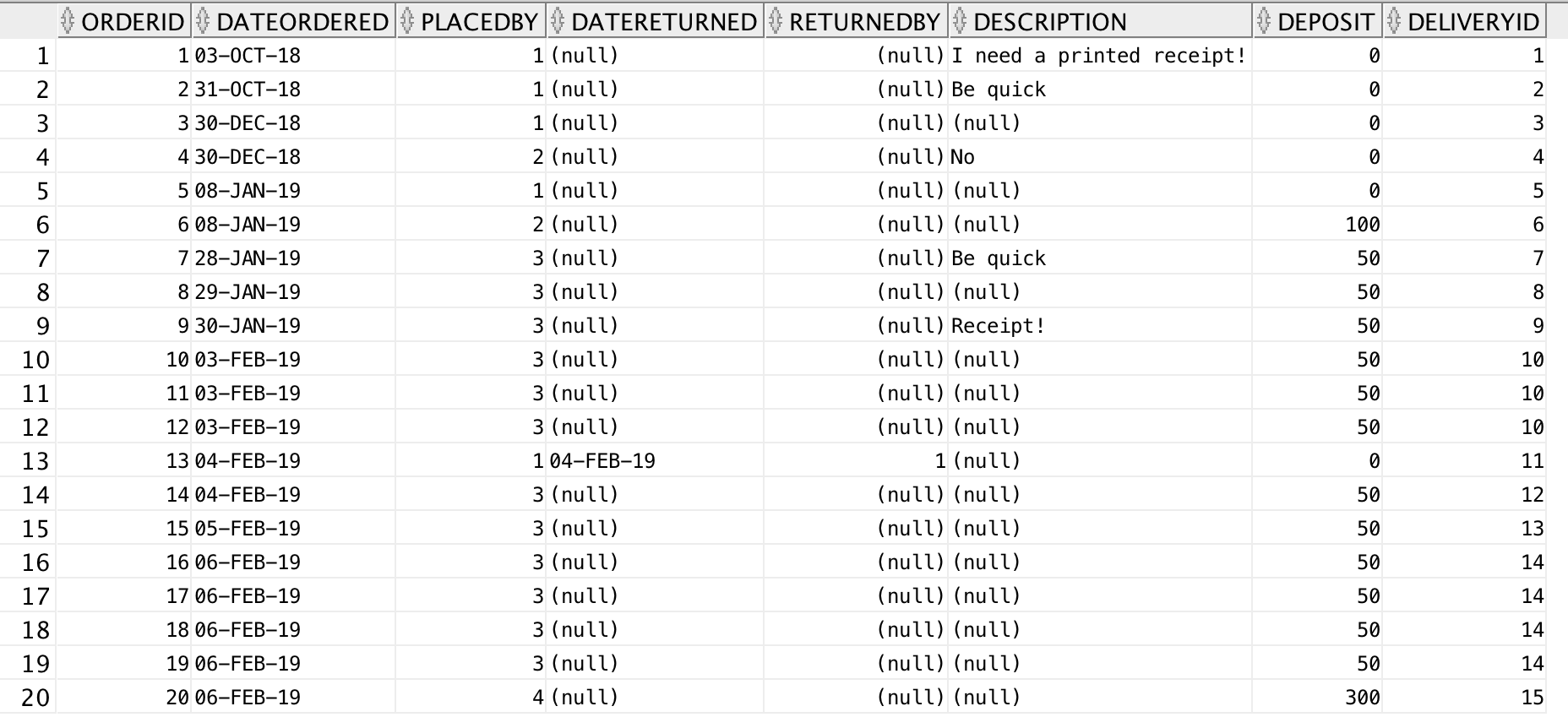
**4.3 create\_order**

* **USE CASE:** creating a new order
* Input: item1\_id, item1\_qty, item2\_id, item2\_qty, item3\_id, item3\_qty, item4\_id, item4\_qty, item5\_id, item5\_qty, customer\_id
* Output:
  + Creates order and enters data in orders and orderdetails table.
  + Updates the quantity on hand attribute in item table by the subtracting the order quantity for each ordered item.
  + Prints message on success of the order.
* Example:

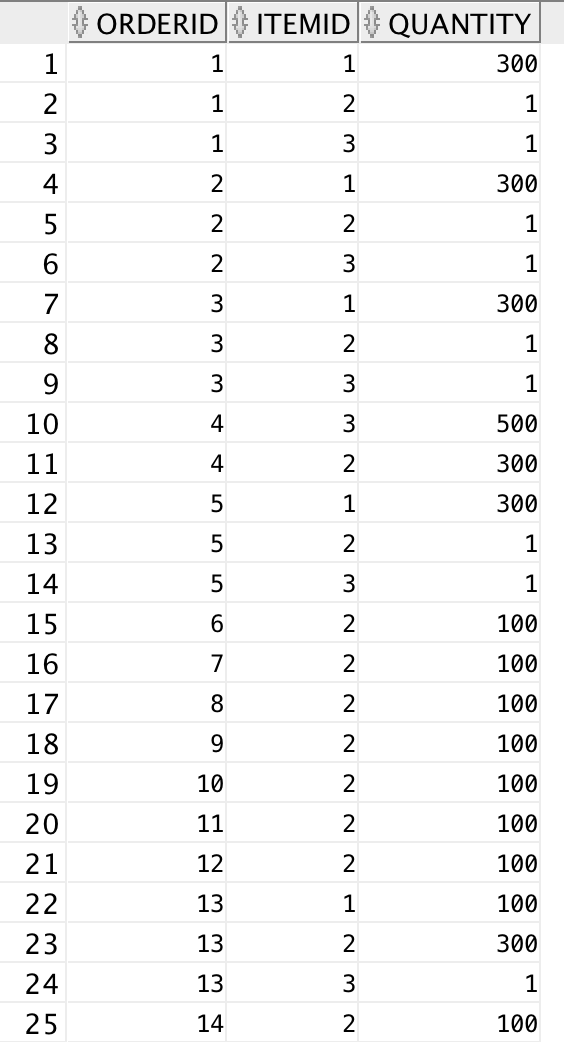
Item Table

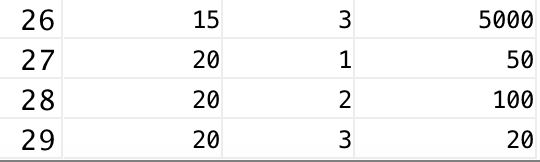


Orders Table



Order Details Table



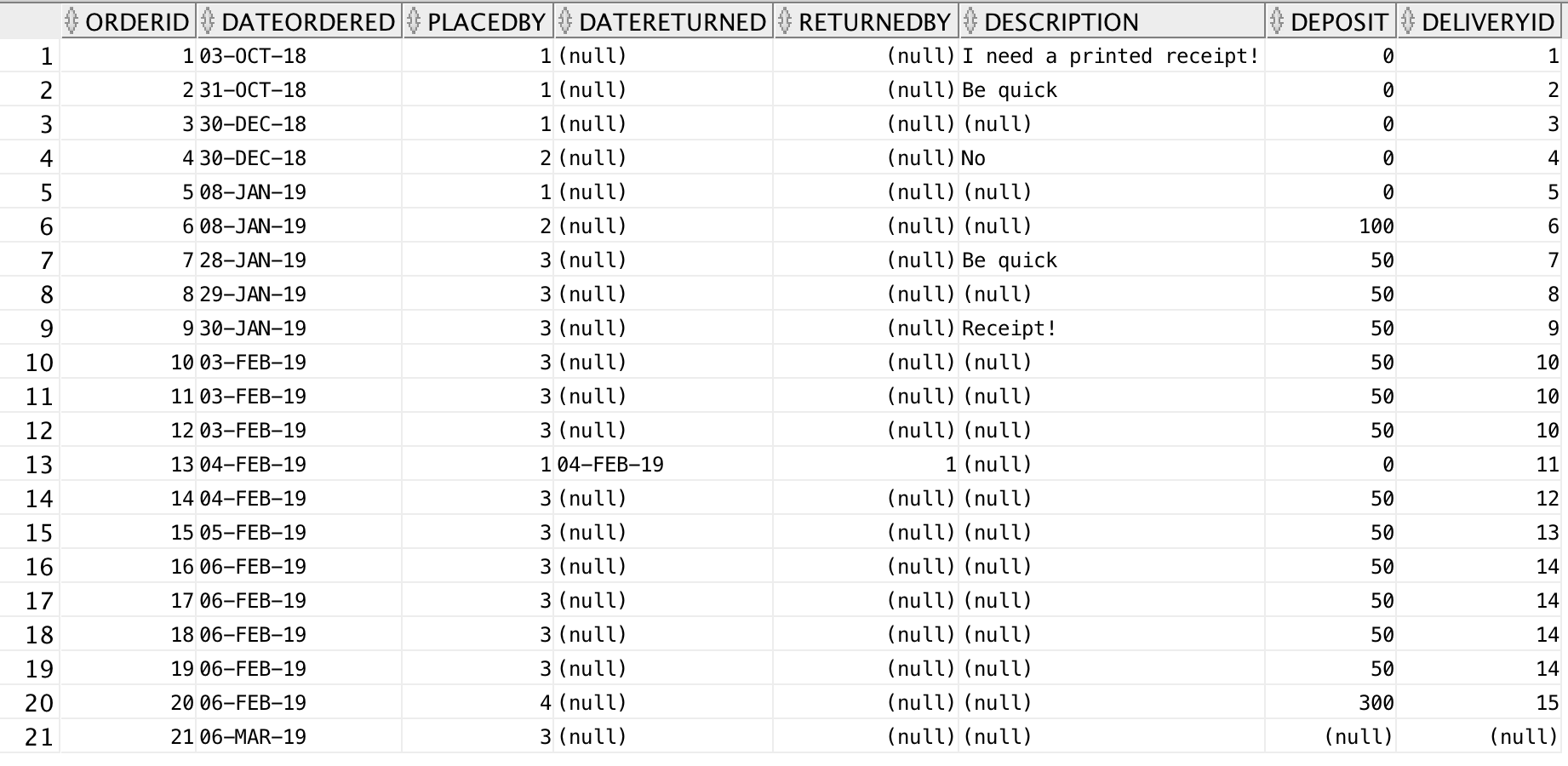


Query:

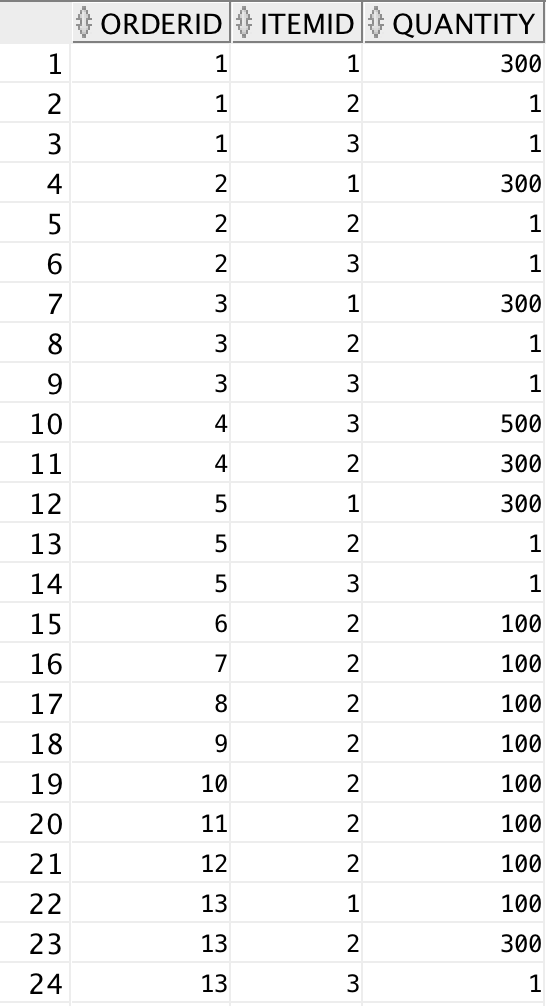
EXECUTE create\_order(1,500,2,0,3,100,4,0,5,0,3);

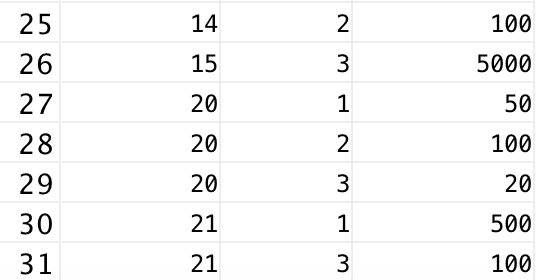
Results:

select \* from orders;

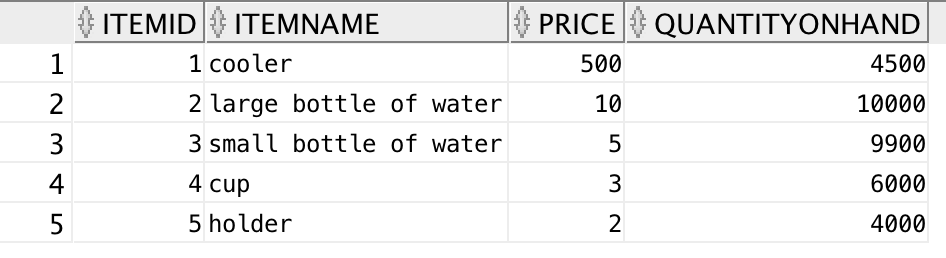
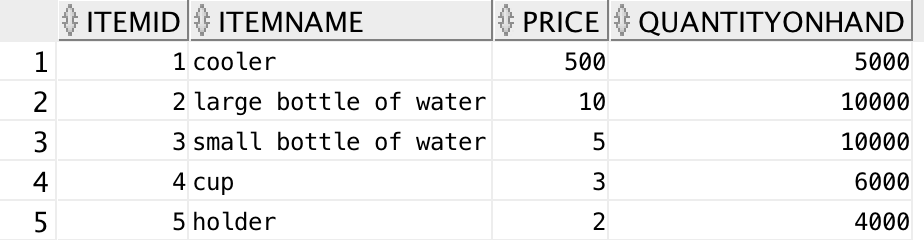


select \* from orderdetails;





select \* from item;



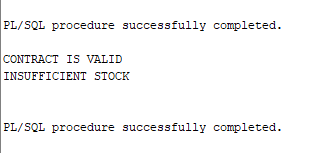
Before Order After Order

Exception:

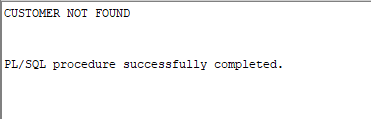
1. If the order quantity exceeds quantity on hand
2. If the contract has expired
3. If the customer does not exist

Try:

EXECUTE create\_order(1,5000,2,0,3,100,4,0,5,0,1);



EXECUTE create\_order(1,100,2,0,3,100,4,0,5,0,6);

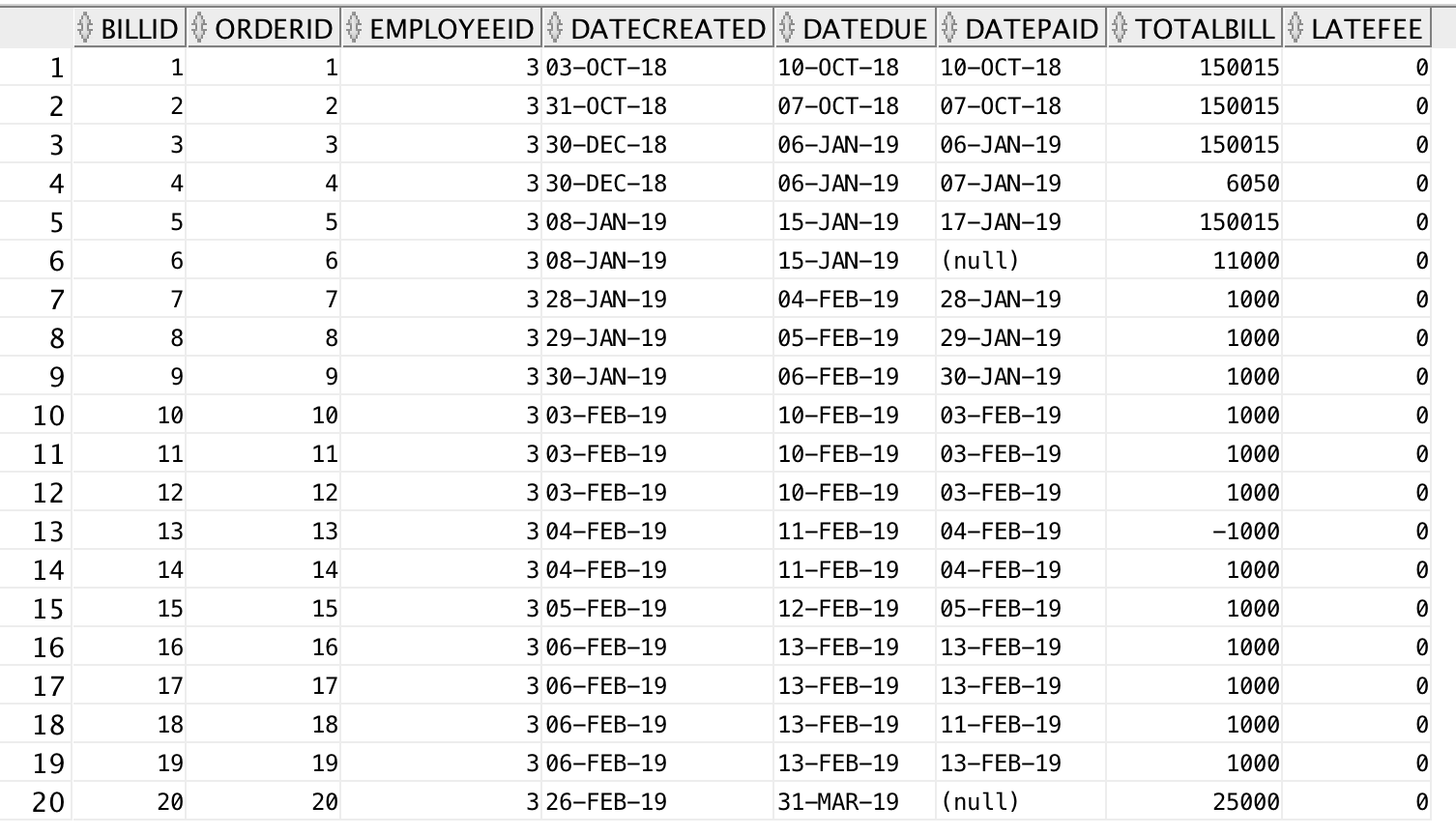


1. **Trigger**

**5.1 update\_bill\_billamount**

* Output: When a bill is updated the trigger is called which prints out the old bill amount and new bill amount and the difference between them.
* Example:

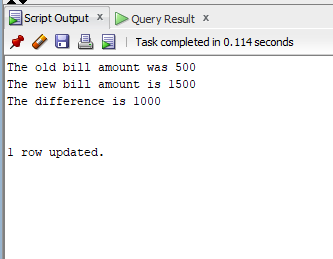
Bill Table



Query:

update bill set totalbill=1500 where billid=19;

Result:



**5.2 return\_order**

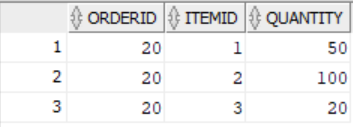
* Output: When an order is **updated** for returns such as the attributes returneddate and returnedby are updated then this trigger is called and it changes the bill amount from debit to credit if the current date hasn’t surpassed the due date and also the quantity of items in the order are added back to the items table.
* Example:

Let’s say we want to return orderid: 20

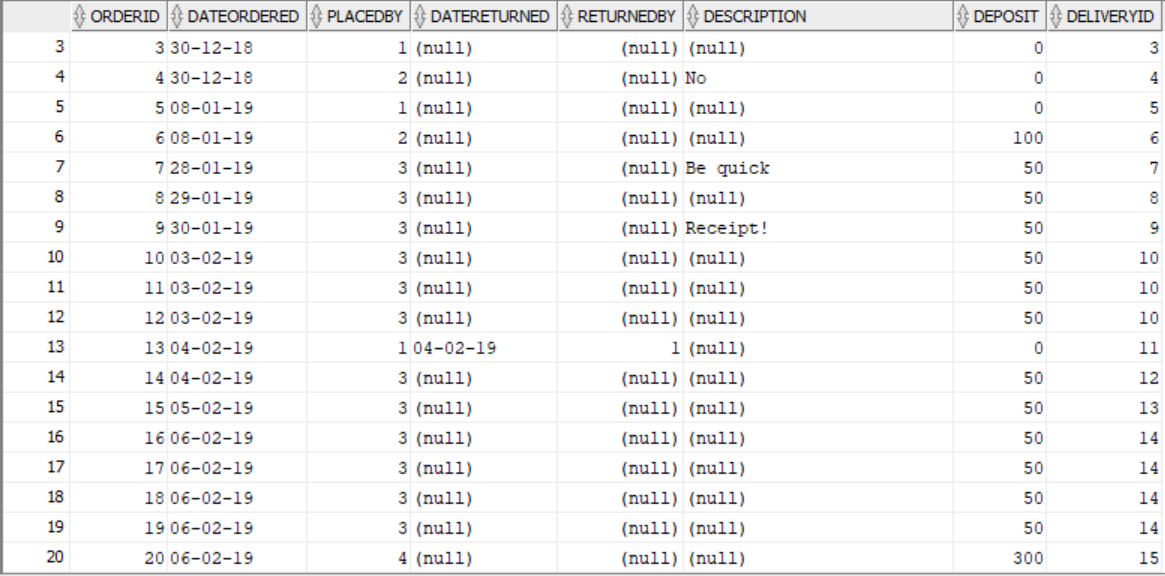
The below query shows the contents of orderid: 20.

**Before Update Tables:**

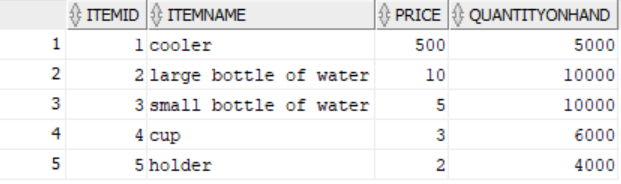
select \* from orderdetails where orderid=20;



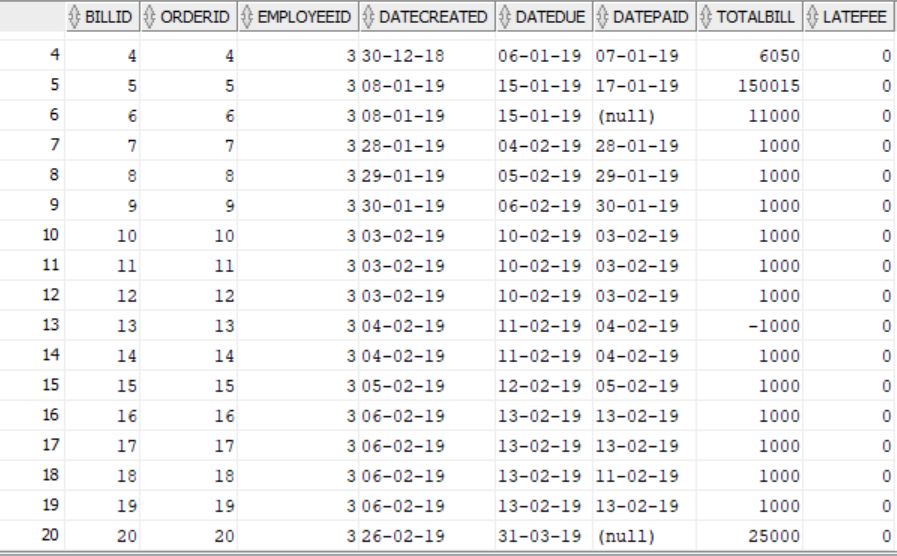
select \* from orders;



select \* from item;



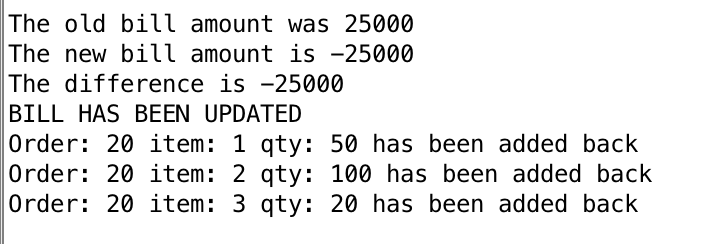
select \* from bill;



Query:

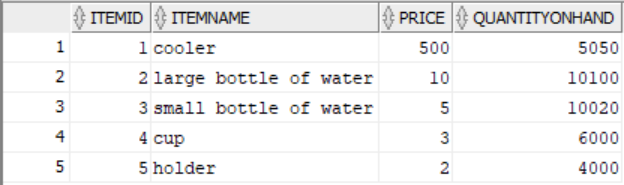
update orders set datereturned = sysdate,returnedby = 4 where orderid=20;

Result:

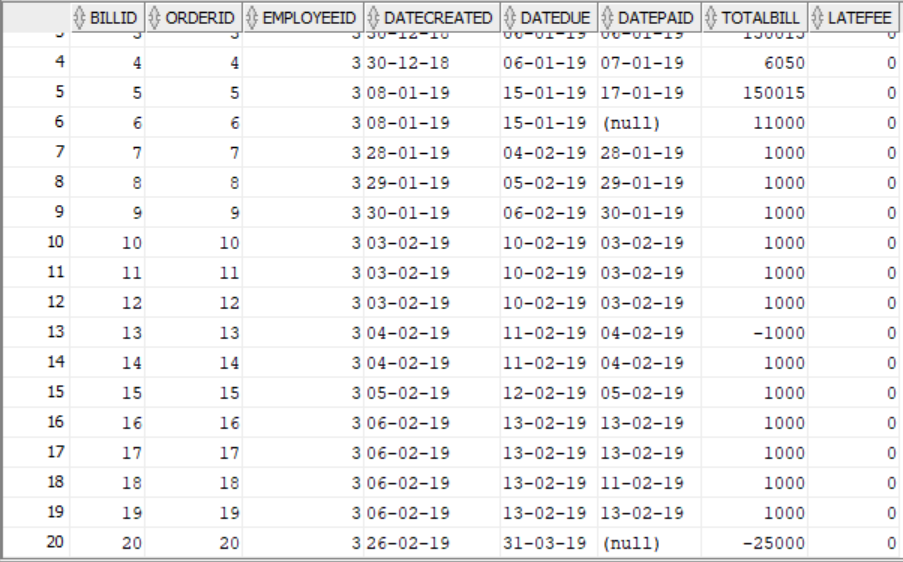


**After update:**

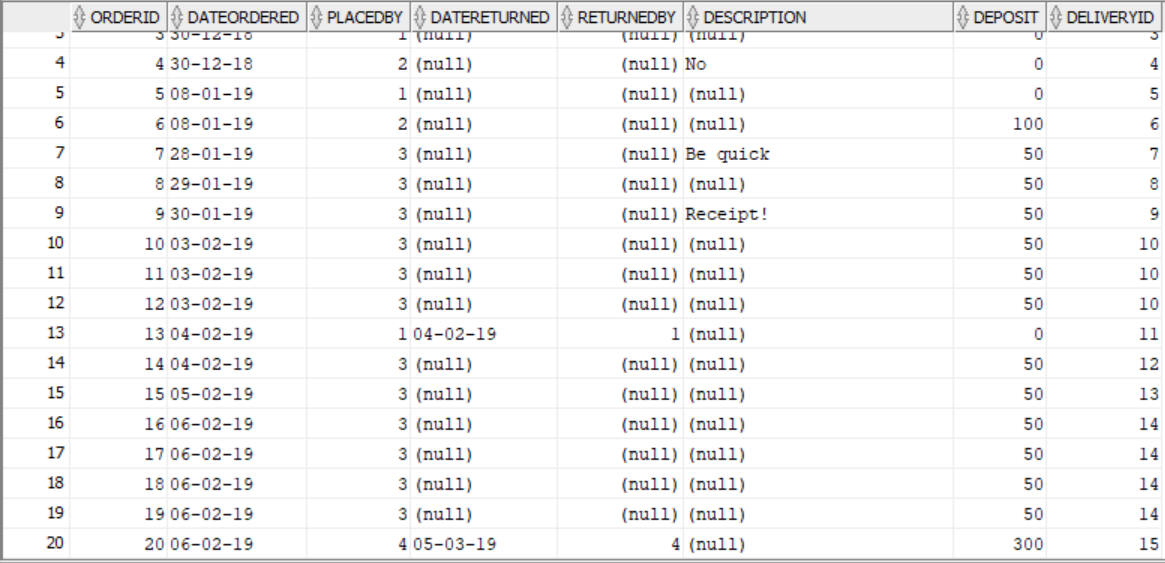
select \* from item;

****

select \* from bill;

****

select \* from orders;

****

1. **Package**

**6.1 employee\_package**

* **USE CASE**: inserting a new record for a new employee and deleting an existing record for resigned employee.

**6.1.1 Procedure: hire\_employee**

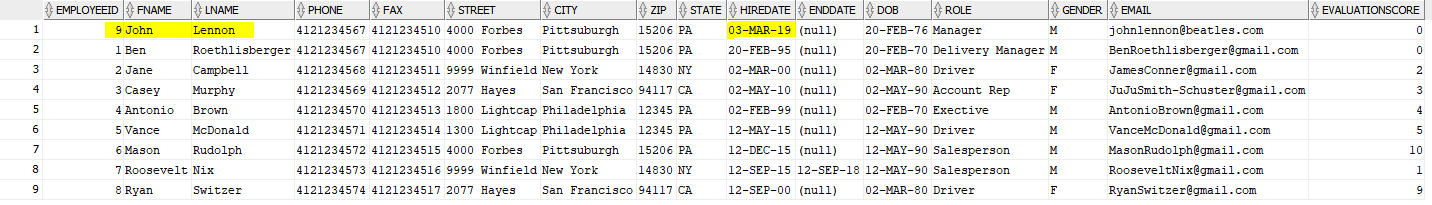
* Input: first name, last name, phone#, fax#, street, city, state, zipcode, state, date of birth, role, gender, email address
* Output: a new record in the employee table
* Example:

Query:

EXECUTE employee\_package.hire\_employee('John', 'Lennon', '4121234567','4121234510','4000 Forbes','Pittsuburgh','15206','PA','20-Feb-76','Manager','M','johnlennon@beatles.com');

Result:

SELECT \* FROM employee;



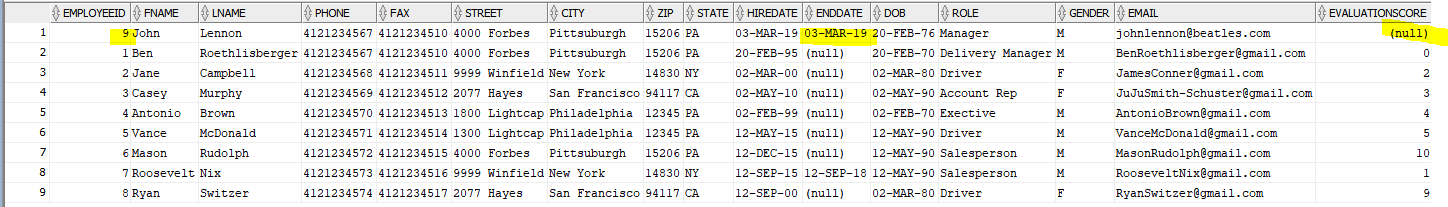
**6.1.2 Procedure: fire\_employee**

* Input: employeeID
* Output: marks the end date of the employee as of today’s date and makes the evaluation score NULL to indicate that the person has been fired/resigned
* Example:

Query:

EXECUTE employee\_package.fire\_employee(9);

SELECT \* FROM employee;

Result:

1. **Roles**

**7.1 Role for Accountant**

We create role accountant and grant access to bill table. You can change the role to test it.

CREATE ROLE ACCOUNTANT;  
GRANT SELECT,INSERT,UPDATE,DELETE ON BILL TO ACCOUNTANT;

**7.2 Role for Delivery Manager**

We create role delivery manager and grant access to route table. You can change the role to test it.

CREATE ROLE DELIVERYMANAGER;  
GRANT SELECT,INSERT,UPDATE,DELETE on ROUTE TO DELIVERYMANAGER;