# Autonomous University of Zacatecas

Academic Unit of Electrical Engineering

Academic Program of Software Engineering

Database Systems Laboratory II Practice 6 - DML

Professor:
Aldonso Becerra Sánchez

Student:
Daniel Alejandro Morales Castillo

# 1 Introduction

The Oracle DML statements are transcendental in the handling of SQL statements at the level of both administrator and database programmer, since they allow the data manipulation of database schemes regardless of the platform used to generate it. This kind of statements can provide you data treatment mechanisms during daily programmer's days.

In this practice we will review some important points about DDL declarations remembering the importance of them. Now we have studied in the theory class about transactions, inserts, updates and deletions. In practice we are going to review and complement the knowledge on the subject.

# 2 Development

## Activity 1

Write the section that describes the work developed in the following activities. Read all the choices carefully because there might be more than one correct answer. Choose all the correct answers for each question. Explain the reason for your answer.

**DESCRIBE EACH DATA MANIPULATION LANGUAGE (DML) STATEMENT**

**1. Which of the following commands can be rolled back?**

A. TRUNCATE
B. DELETE
C. UPDATE
D. MERGE
E. COMMIT
F. INSERT

My answer: B,C AND F

There are implicit and explicit transactions, the implicit transactions can not be rolled back because implicit transactions have been automatically committed or rolled back, the DDL and DCL sentences are transactions that when are executed starts and end so you can not do a ROLLBACK because the transaction is done. But the DML sentences like DELETE, UPDATE and INSERT can be rolled back because they are open until we do a COMMIT or a ROLLBACK.

**2. How can you change the primary key value of a row? (Choose the best answer.)**

A. The row must be removed with a DELETE and reentered with an INSERT.

B. This is only possible if the row is first locked with a SELECT FOR UPDATE.

C. You cannot change the primary key value.

D. Change it with a simple UPDATE statement

My answer: D

The UPDATE sentence is used to update the existing records so if you want to change a value you can use it.

**3. If an UPDATE or DELETE command has a WHERE clause that gives it a scope of several rows, what will happen if there is an error part way through execution? The command is one of several in a multistatement transaction. (Choose the best answer.)**

A. The command will skip the row that caused the error and continue.
B. Whatever work the command had done before hitting the error will be rolled back, but work done already by the transaction will remain.
C. The command will stop at the error, and the rows that have been updated or deleted will remain updated or deleted.
D. The whole transaction will be rolled back.

My answer: C
In the theory class we talked about multi statement transactions and we mentioned that if an error occurred the command will stop, the error will be rolled back and the rows that have been updated or deleted will remain updated or deleted

**INSERT ROWS INTO A TABLE**

**4. If a table T1 has four numeric columns, C1, C2, C3, and C4, which of these statements will succeed? (Choose the best answer.)**

A. insert into T1 values (1,2,3,null);
B. insert into T1 select * from T1;
C. insert into T1 values ('1','2','3','4');
D. All the statements (A, B, and C) will succeed.
E. None of the statements (A, B, or C) will succeed.

My answer: A
If the columns do not have the NOT NULL constraint that means that a null value can be used, so the statement A can be succeeded.

**5. Study the result of this SELECT statement: If you issue this statement: insert into t1 (c1,c2) values(select c1,c2 from t1); why will it fail? (Choose the best answer.)**

```
SQL> select * from t1;
C1 C2 C3 C4
---------- ---------- ---------- ----------
1 2 3 4
5 6 7 8
```

A. Because the VALUES keyword is not used with a subquery.

B. Because the subquery is not scalar: it should use MAX or MIN to generate scalar values.

C. Because the subquery returns multiple rows: it requires a WHERE clause to restrict th number of rows returned to one.

D. Because values are not provided for all the table's columns: there should be NULLs for C3 and C4.

E. It will succeed, inserting two rows with NULLs for C3 and C4.

My answer: A

You can use the word VALUES when you are inserting values in only one row, the problem sentence is for copy rows from another table, the syntax do not include the word VALUES.

**6. Consider this statement: insert into regions (region id,region name) values ((select max(region id)+1 from regions), 'Spain'); What will the result be? (Choose the best answer.)**

A. The statement will execute without error.

B. The statement will fail if the REGIONS table has a third column.

C. The statement will not succeed if the value generated for REGION ID is not unique, because REGION ID is the primary key of the REGIONS table.

D. The statement has a syntax error because you cannot use the VALUES keyword with a subquery. a subquery.

My answer: D

When you use the reserved word VALUES you must put the literal values to be inserted, you will have an error if you use a subquery.

**UPDATE ROWS IN A TABLE**

**7. You want to insert a row and then update it. What sequence of steps should you follow? (Choose the best answer.)**

A. INSERT, COMMIT, SELECT FOR UPDATE, UPDATE, COMMIT
B. INSERT, SELECT FOR UPDATE, UPDATE, COMMIT
C. INSERT, COMMIT, UPDATE, COMMIT
D. INSERT, UPDATE, COMMIT

My answer: D
You only use the COMMIT to finish a transaction, if you need to insert and update you can follow the following steps insert, update, commit.

**8. If you issue this command: update employees set salary=salary * 1.3; what will be the result? (Choose the best answer.)**
A. Every row will have SALARY incremented by 30 percent, unless SALARY was NULL.
B. There will be an error if any row has its SALARY column NULL.
C. The first row in the table will be updated.
D. The statement will fail because there is no WHERE clause to restrict the rows affected

My answer: A
Every salary will be multiplied by 1.30 but the null values will be remain it value

**DELETE ROWS FROM A TABLE**

**9. How can you delete the values from one column of every row in a table? (Choose the best answer.)**

A. Use the UPDATE command.
B. Use the DROP COLUMN command.
C. Use the DELETE COLUMN command.
D. Use the TRUNCATE COLUMN command.

My answer: A

Using the UPDATE command because you can put a null value or a ” value, delete the column can be an option but it is not so useful. INDEX.

**10. Which of these commands will remove every row in a table? (Choose one or more correct answers.)**

A. An UPDATE command, setting every column to NULL and with no WHERE clause.
B. A TRUNCATE command.
C. A DROP TABLE command.
D. A DELETE command with no WHERE clause.

My answer: B AND C

The TRUNCATE command remove all the rows leaving the table empty, the DROP TABLE delete the table with the columns, so these two options will remove every column in a table.

**CONTROL TRANSACTIONS**

**11. User JOHN updates some rows and asks user MICHAEL to log in and check the changes before he commits them. Which of the following statements is true? (Choose the best answer.)**

A. JOHN must commit the changes so that MICHAEL can see them, but only JOHN can roll them back.

B. MICHAEL will not be able to see the changes.

C. JOHN must commit the changes so that MICHAEL can see them and, if necessary, roll them back.

D. MICHAEL can see the changes but cannot alter them because JOHN will have locked the rows.

My answer: B

JOHN must commit the changes so that MICHAEL can see them, if MICHAEL try to see them before JOHN use a COMMIT MICHAEL going to see the original data without changes.

**12. User JOHN updates some rows but does not commit the changes. User MICHAEL queries the rows that JOHN updated. Which of the following statements is true? (Choose the best answer.)**

A. MICHAEL will not be able to see the rows because they will be locked.

B. MICHAEL will see the state of the state of the data as it was when JOHN last created a SAVEPOINT.

C. MICHAEL will see the old versions of the rows.

D. MICHAEL will be able to see the new values, but only if he logs in as JOHN.

My answer: C

JOHN have not done a COMMIT and he is only using UPDATE and no FOR UPDATE so the columns are not locked, MICHAEL will see the old rows

**13. Which of these commands will terminate a transaction? (Choose three correct answers.)**

A. ROLLBACK TO SAVEPOINT
B. TRUNCATE
C. ROLLBACK
D. COMMIT
E. SAVEPOINT
F. DELETE

My answer: B,C AND D

ROLLBACK and COMMIT are used to finish transactions and TRUNCATE is a transaction, so it start and finish a transaction.

## Activity 2:

Insert a dataset for the scheme SALES carried out in practice 4, activity 2. Take into account the following instructions:

a) Use sequences as possible for primary key values (using pseudo-columns CURRVAL and NEXTVAL).

b) Insert a representative set of values for each table.

c) Use SYSDATE as possible.

```
--PRACTICA6
--actividad2

INSERT INTO P4_SHOP(SHOP_ID,ADDRESS,MANAGER_SHOP)
VALUES(SEQUENCE_SHOP_ID.NEXTVAL, 'ZACATECAS ZAC','JUAN ROBLES');

INSERT INTO P4_CHANNEL(CHANNEL_ID,CNAME)
VALUES(SEQUENCE_CHANNEL_ID.NEXTVAL, 'CANAL  5 ');

INSERT INTO P4_PRODUCT(PRODUCT_ID, PNAME,SALE_PRICE, PURCHASE_PRICE, PROVIDER)
VALUES(SEQUENCE_PRODUCT_ID.NEXTVAL, 'FUNKOS','400','350','FUNKOPOP');


INSERT INTO P4_EMPLOYEE(EMP_ID,EMP_NAME,EMP_LASTN,BOSS_ID,ADDRESS,DATE_OF_BIRTH,GENDER,BENEFICIARIES)
VALUES(SEQUENCE_EMPLOYEE_ID.NEXTVAL, 'DANIEL','CASTILLO',2,'FERROCARRIL 2','15/08/1999','MALE','MEJOR EMPLEADO');
```

Salida de Script

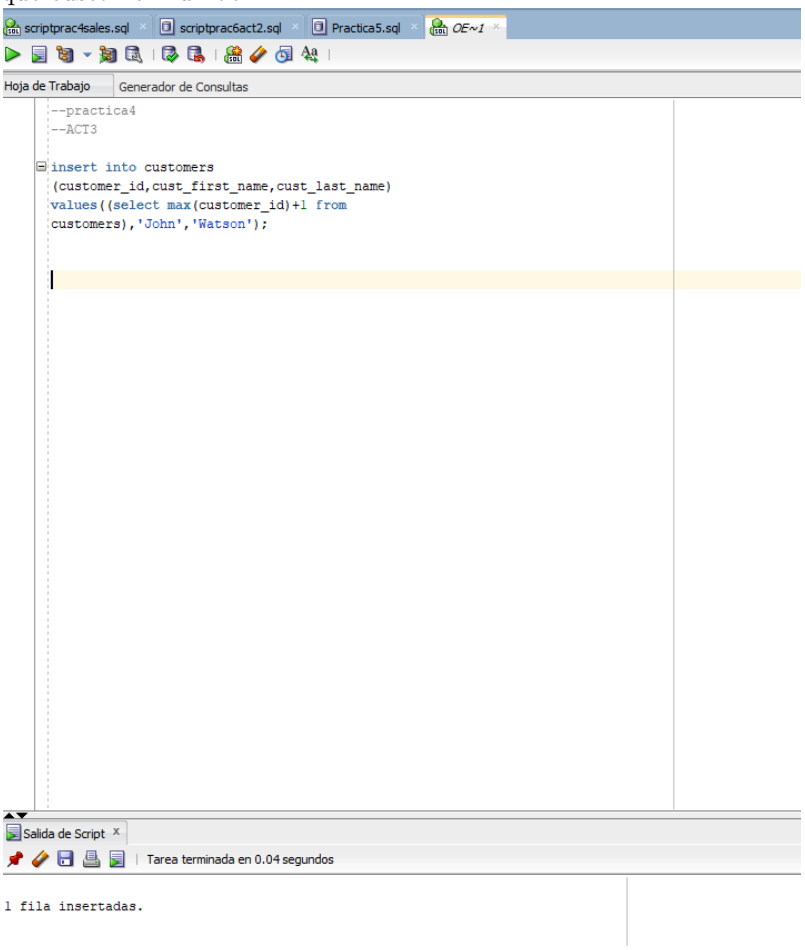
Tarea terminada en 0.171 segundos

```
          the necessary privileges.
*Action:  Ask your database administrator or designated security
          administrator to grant you the necessary privileges
Conectado.
La conexión creada por el comando de script CONNECT se ha desconectado

1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.
```

## Activity 3:

Carry out this exercise in the oe schema (copy and paste screen results). Analyze the results for each sentence.

1. Insert a customer into CUSTOMERS, using a function to generate a unique customer number:

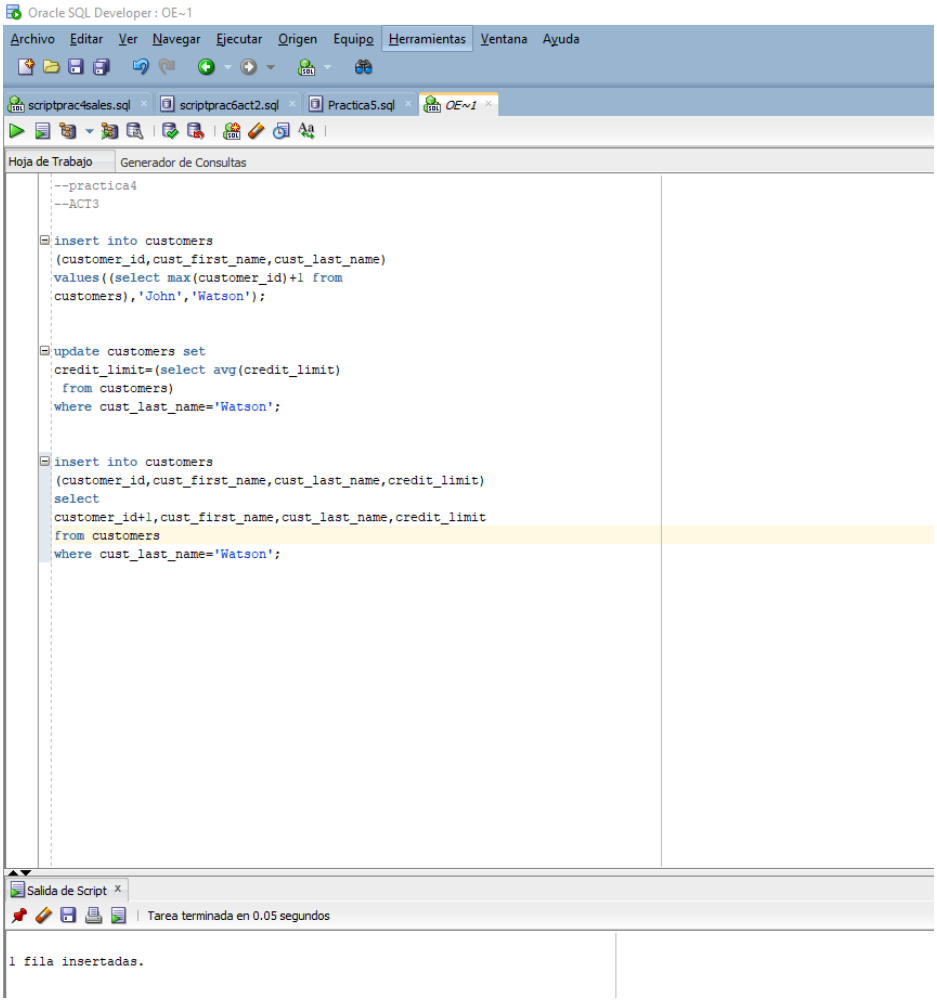


2. Give him a credit limit equal to the average credit limit:

```
--practica4
--ACT3

insert into customers
(customer_id,cust_first_name,cust_last_name)
values((select max(customer_id)+1 from
customers),'John','Watson');


update customers set
credit_limit=(select avg(credit_limit)
 from customers)
where cust_last_name='Watson';
```

```
1 fila insertadas.


1 fila actualizadas.
```

3. Create another customer using the customer just created, but make sure the CUSTOMERID is unique:



4. Change the name of the second entered customer:

```sql
--practica4
--ACT3

insert into customers
(customer_id,cust_first_name,cust_last_name)
values((select max(customer_id)+1 from
customers),'John','Watson');


update customers set
credit_limit=(select avg(credit_limit)
 from customers)
where cust_last_name='Watson';


insert into customers
(customer_id,cust_first_name,cust_last_name,credit_limit)
select
customer_id+1,cust_first_name,cust_last_name,credit_limit
from customers
where cust_last_name='Watson';




update customers
set cust_last_name='Ramklass',cust_first_name='Roopesh'
where customer_id=(select max(customer_id) from
customers);
```

```
1 fila actualizadas.
```

5. Commit this transaction:

scriptprac4sales.sql | scriptprac6act2.sql | Practica5.sql | OE~1

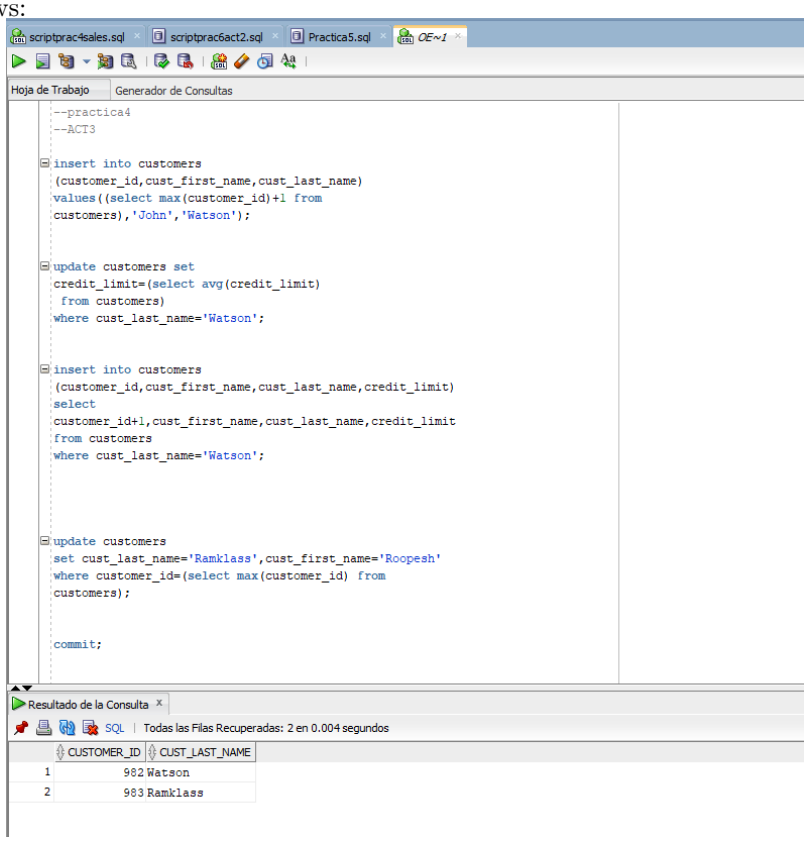
Hoja de Trabajo | Generador de Consultas

```sql
--practica4
--ACT3

insert into customers
(customer_id,cust_first_name,cust_last_name)
values((select max(customer_id)+1 from
customers),'John','Watson');


update customers set
credit_limit=(select avg(credit_limit)
 from customers)
where cust_last_name='Watson';


insert into customers
(customer_id,cust_first_name,cust_last_name,credit_limit)
select
customer_id+1,cust_first_name,cust_last_name,credit_limit
from customers
where cust_last_name='Watson';




update customers
set cust_last_name='Ramklass',cust_first_name='Roopesh'
where customer_id=(select max(customer_id) from
customers);


commit;
```

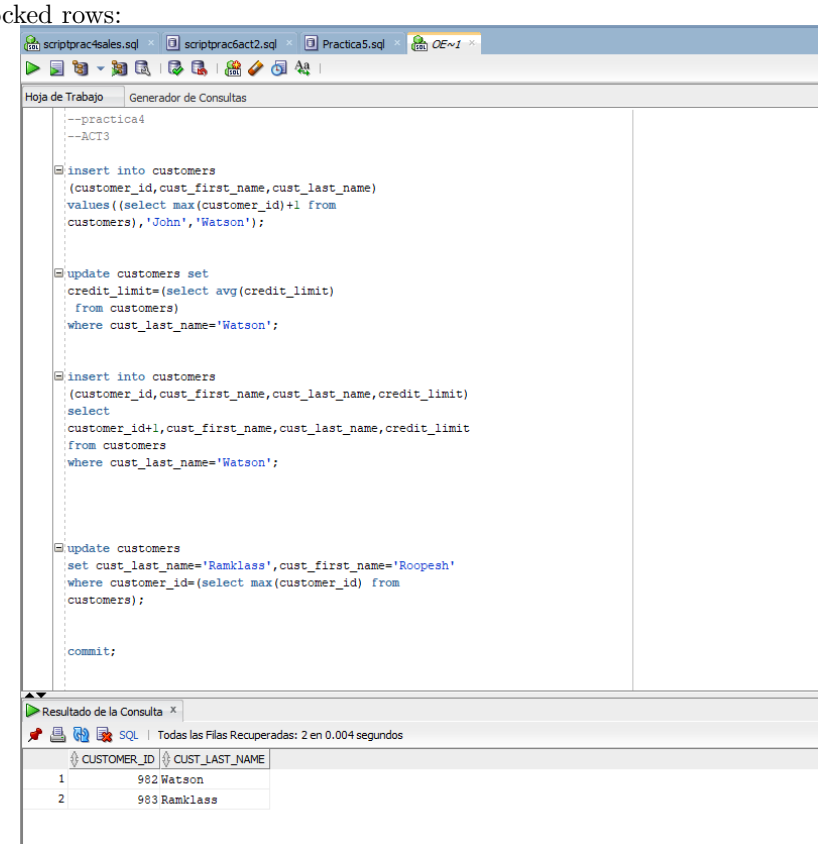
Salida de Script | Tarea terminada en 0.052 segundos

```
Confirmación terminada.
```

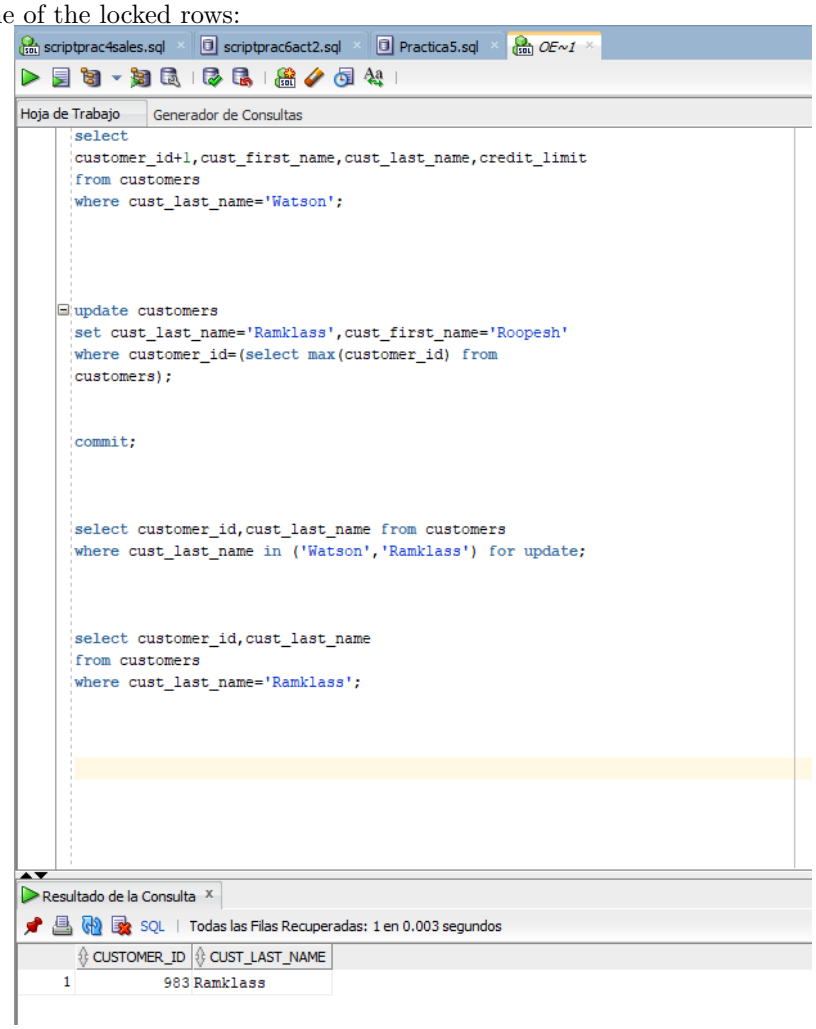
6. Determine the CUSTOMERIDs of the two new customers and lock the rows:



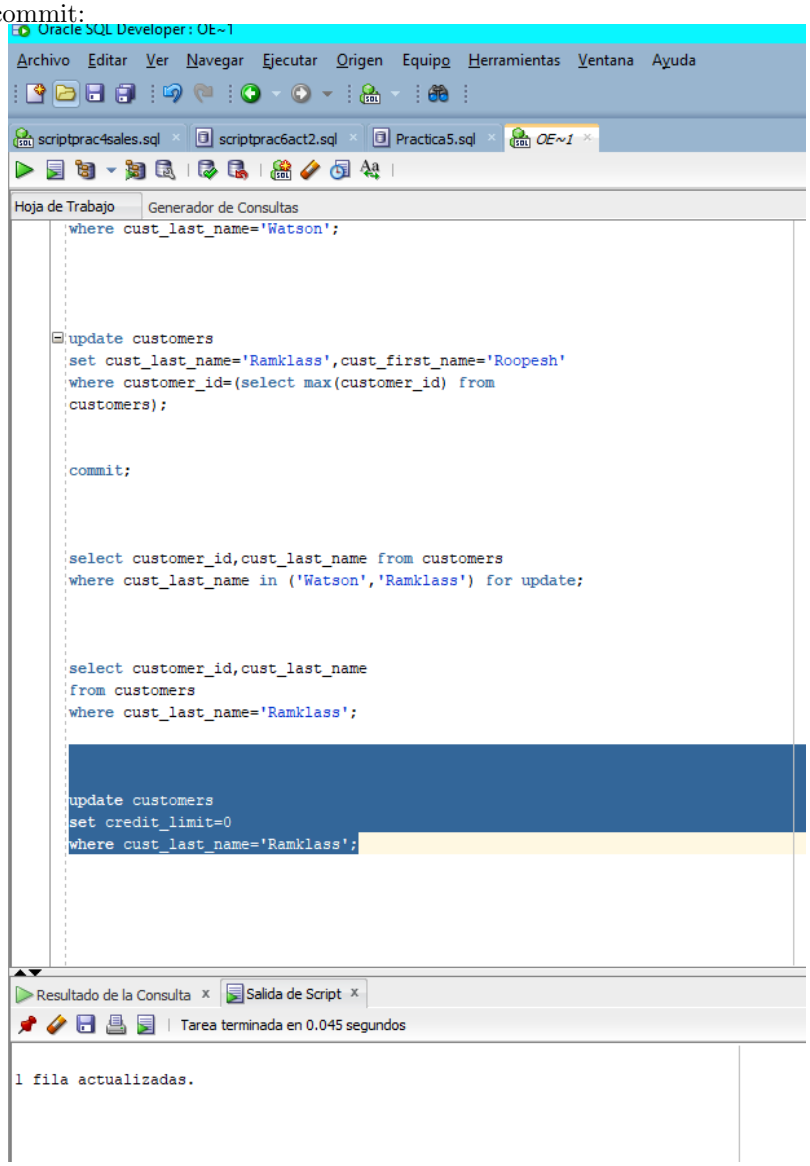
7. From another session connected to the OE schema, attempt to select the locked rows:



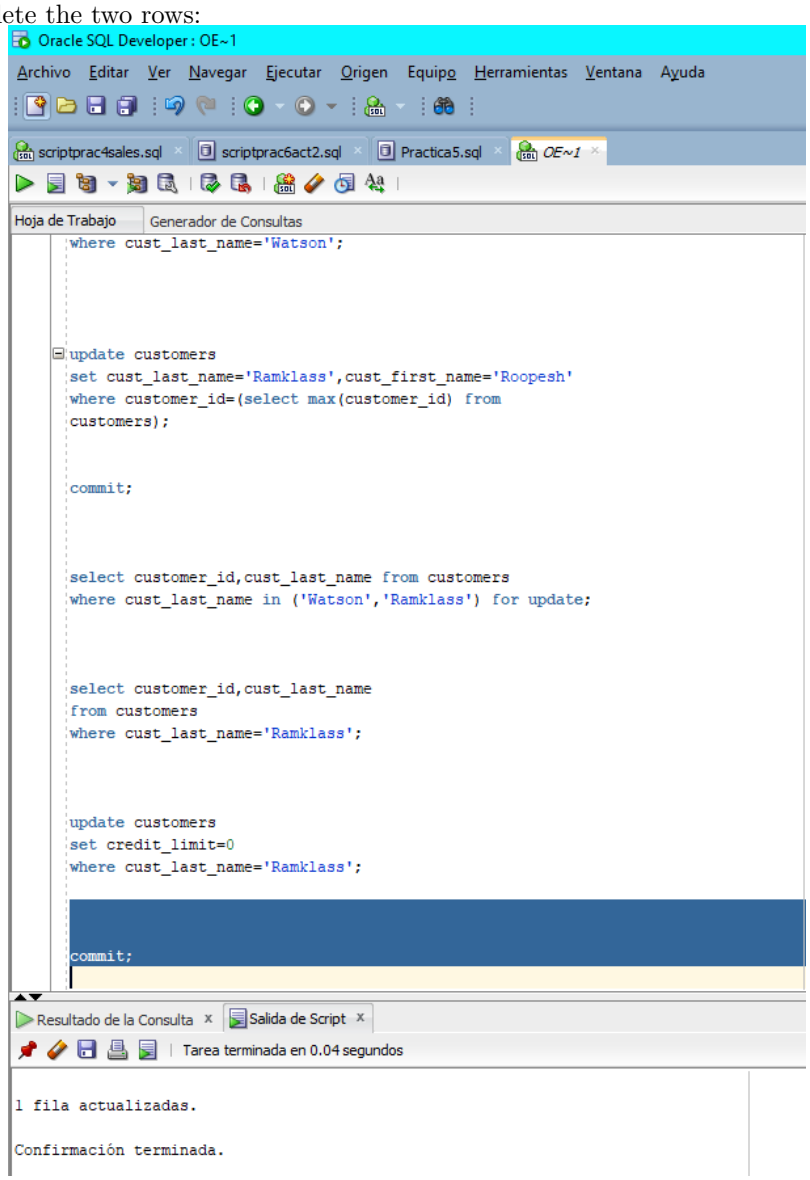
8. In this second session connected to the OE schema, attempt to update one of the locked rows:



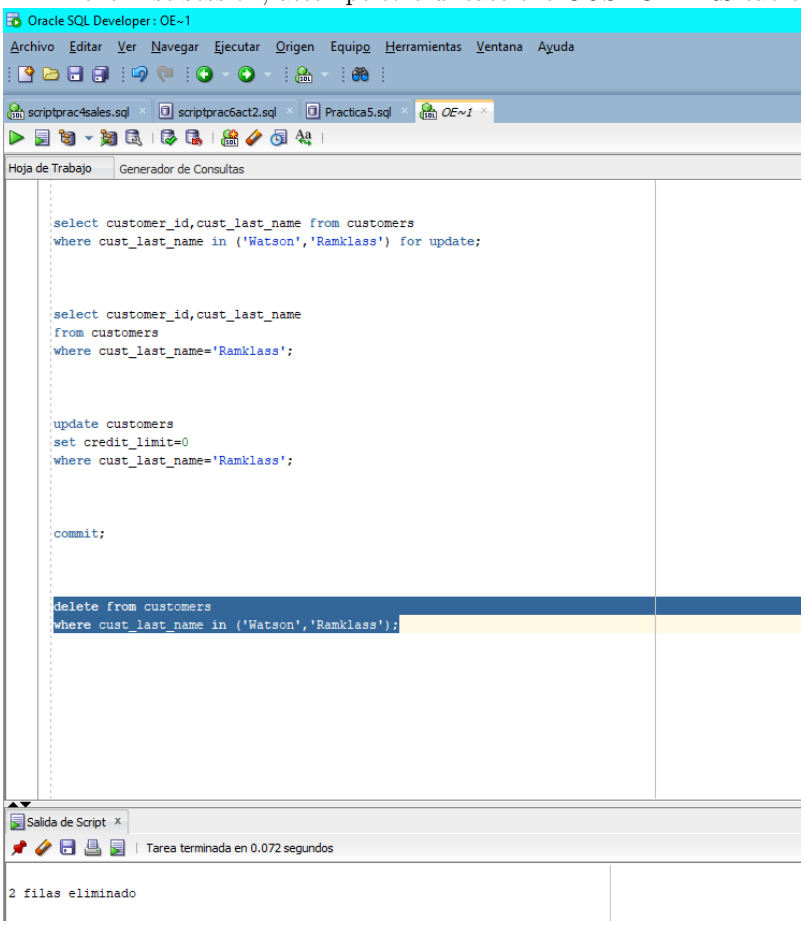
9. This command will hang. In the first session, release the locks by issuing a commit:



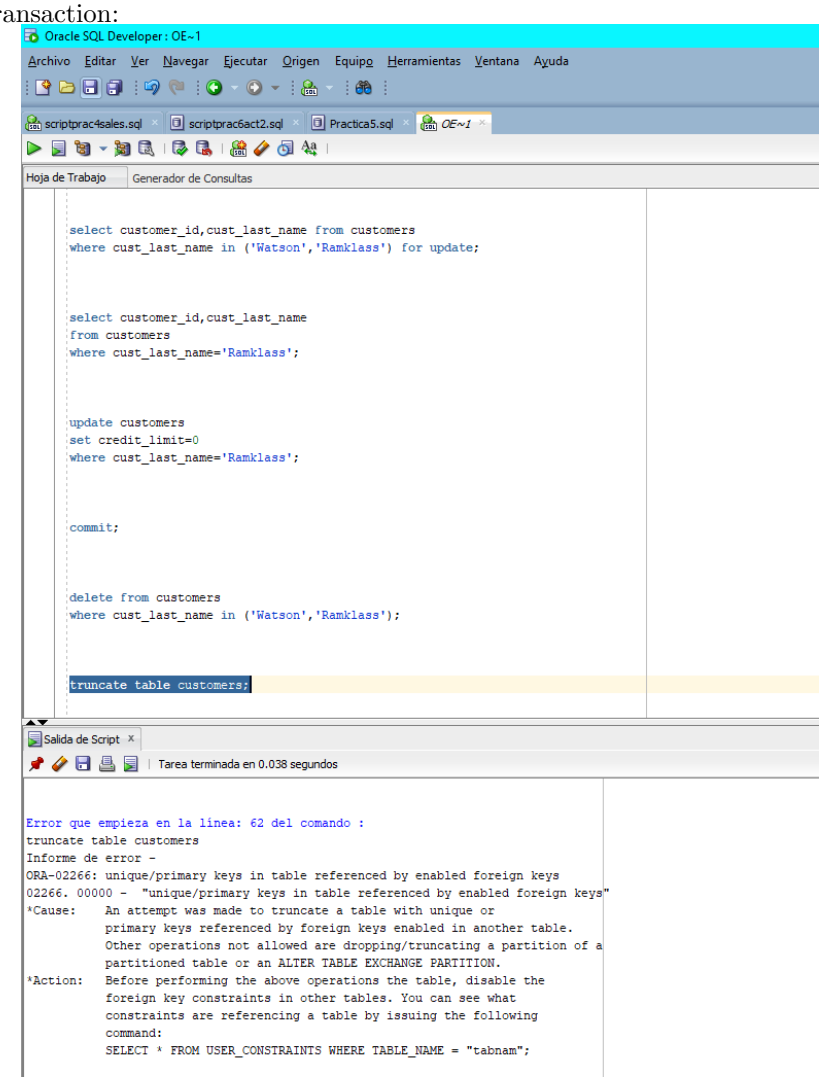
10. The second session will now complete its update. In the second session, delete the two rows:



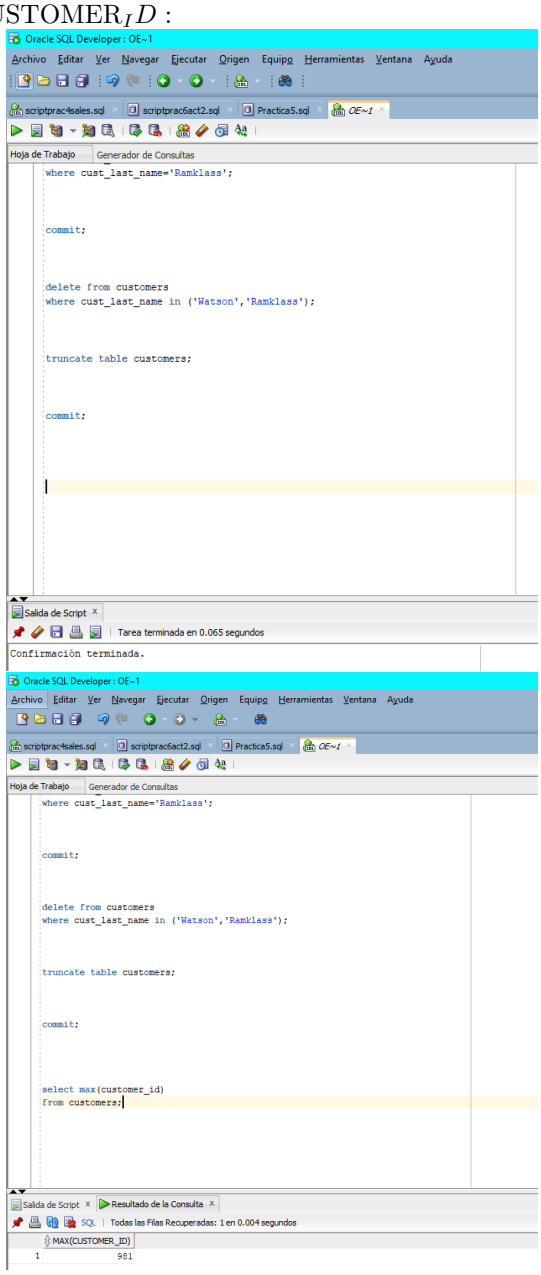
11. In the first session, attempt to truncate the CUSTOMERS table:



12. This will fail because there is a transaction in progress against the table, which will block all DDL commands. In the second session, commit the transaction:



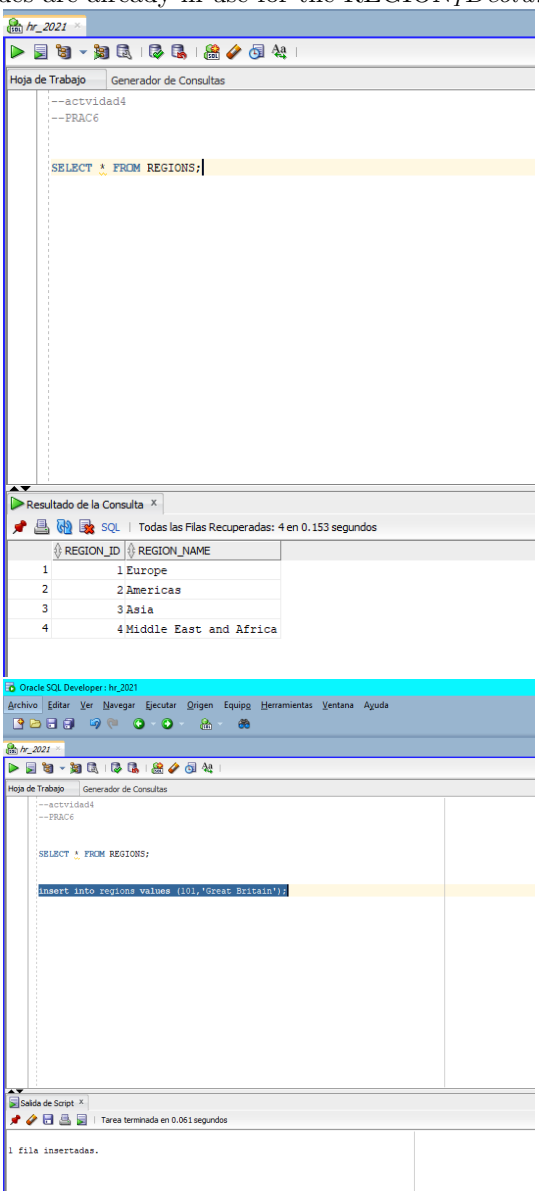
13. The CUSTOMERS table will now be back in the state it was in at the start of the exercise. Confirm this by checking the value of the highest CUSTOMER$_ID$ :



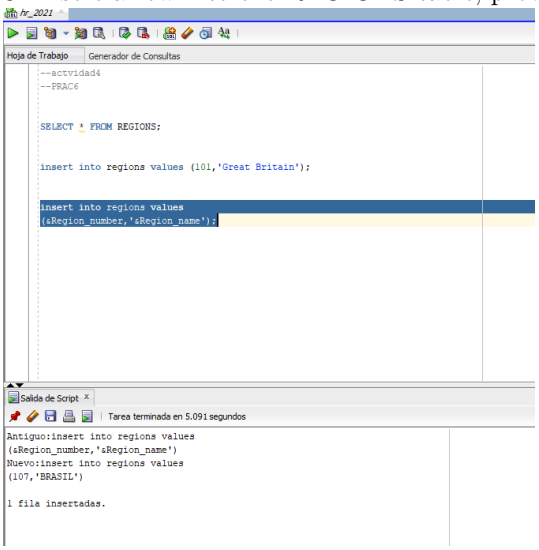
## Activity 4:

Copy and paste screen results. Analyze the results for each sentence. In this section, use various techniques to insert rows into a table. Follow the next instructions:

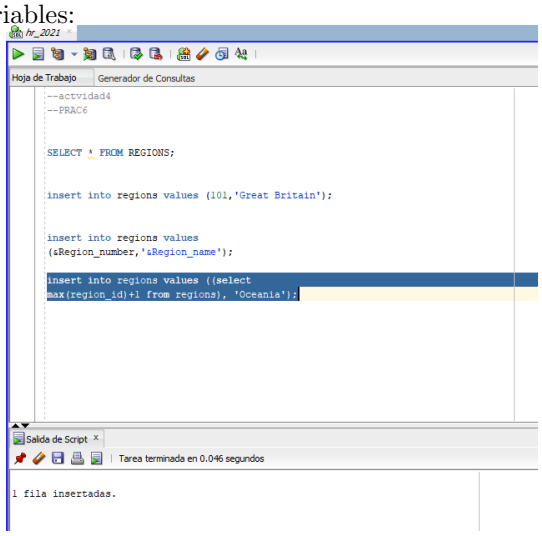
1. Connect to the HR schema. 2. Query the REGIONS table, to check what values are already in use for the REGION$_IDcolumn$ :



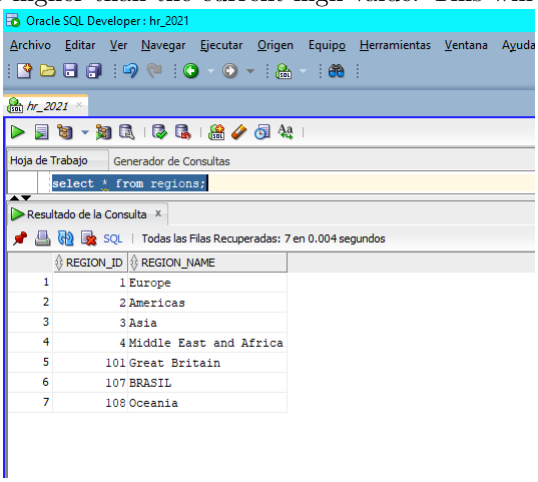
3. Insert a row into the REGIONS table, providing the values in line:




4. Insert a row into the REGIONS table, providing the values as substitution variables:

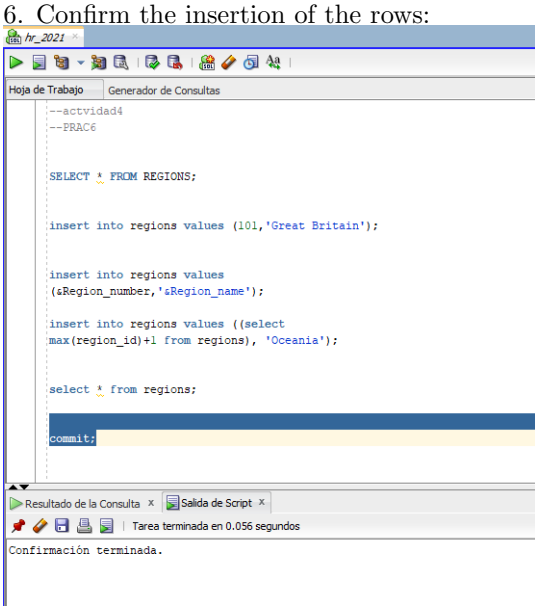


5. Insert a row into the REGIONS table, calculating the REGIONID to be one higher than the current high value. This will need a scalar subquery:




7. Commit the insertions:

6. Confirm the insertion of the rows:

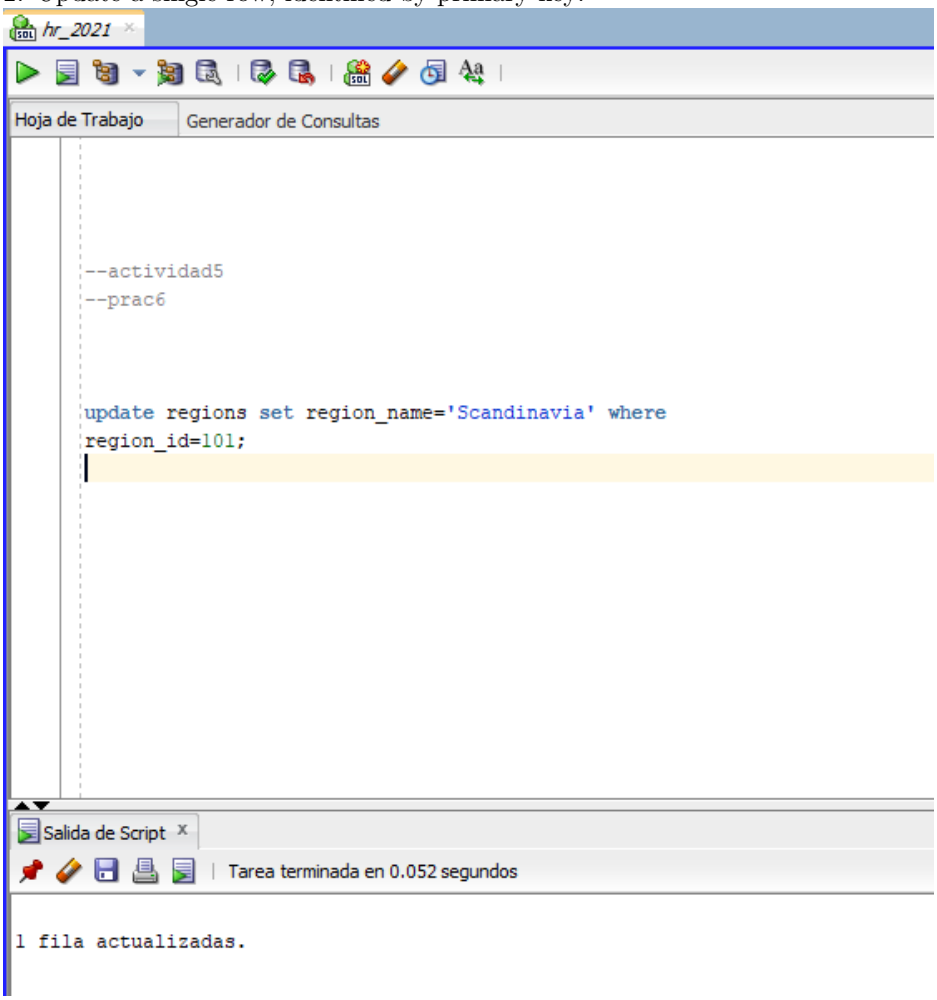


## Activity 5:

**Copy and paste screen results. Analyze the results for each sentence.**

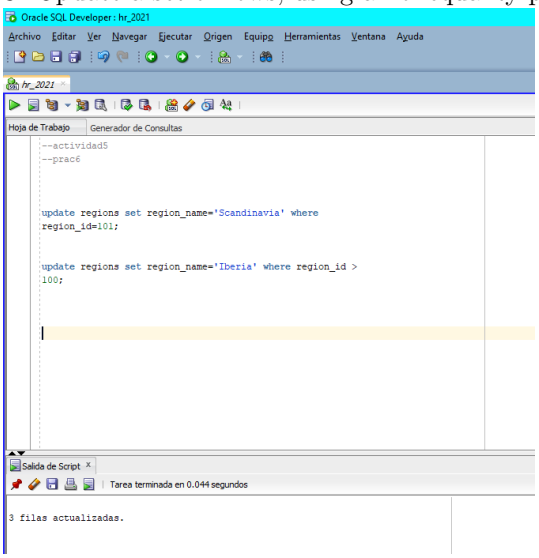
In this section, use various techniques to update rows in a table. Follow the next instructions:

1. Connect to the HR schema.
2. Update a single row, identified by primary key:



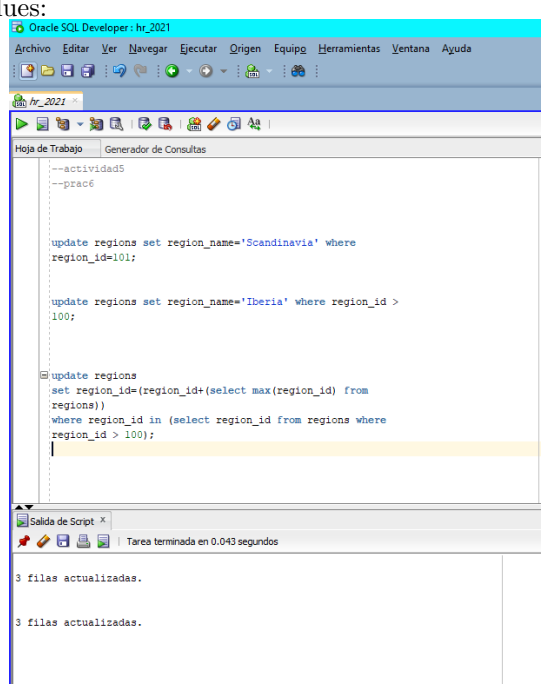

This statement should return the message “1 row updated.”

3. Update a set of rows, using a nonequality predicate:



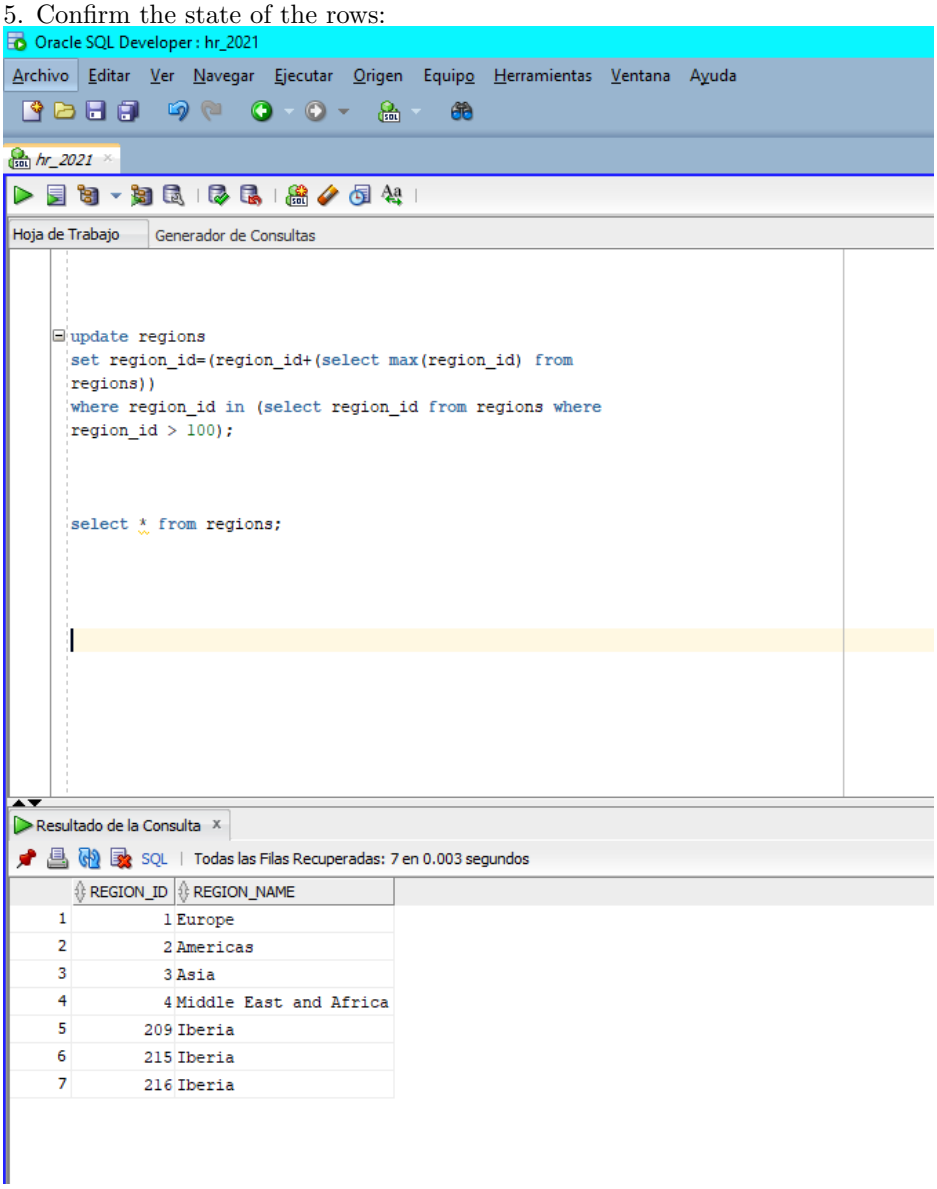

This statement should return the message "3 rows updated."

4. Update a set of rows, using subqueries to select the rows and to provide values:

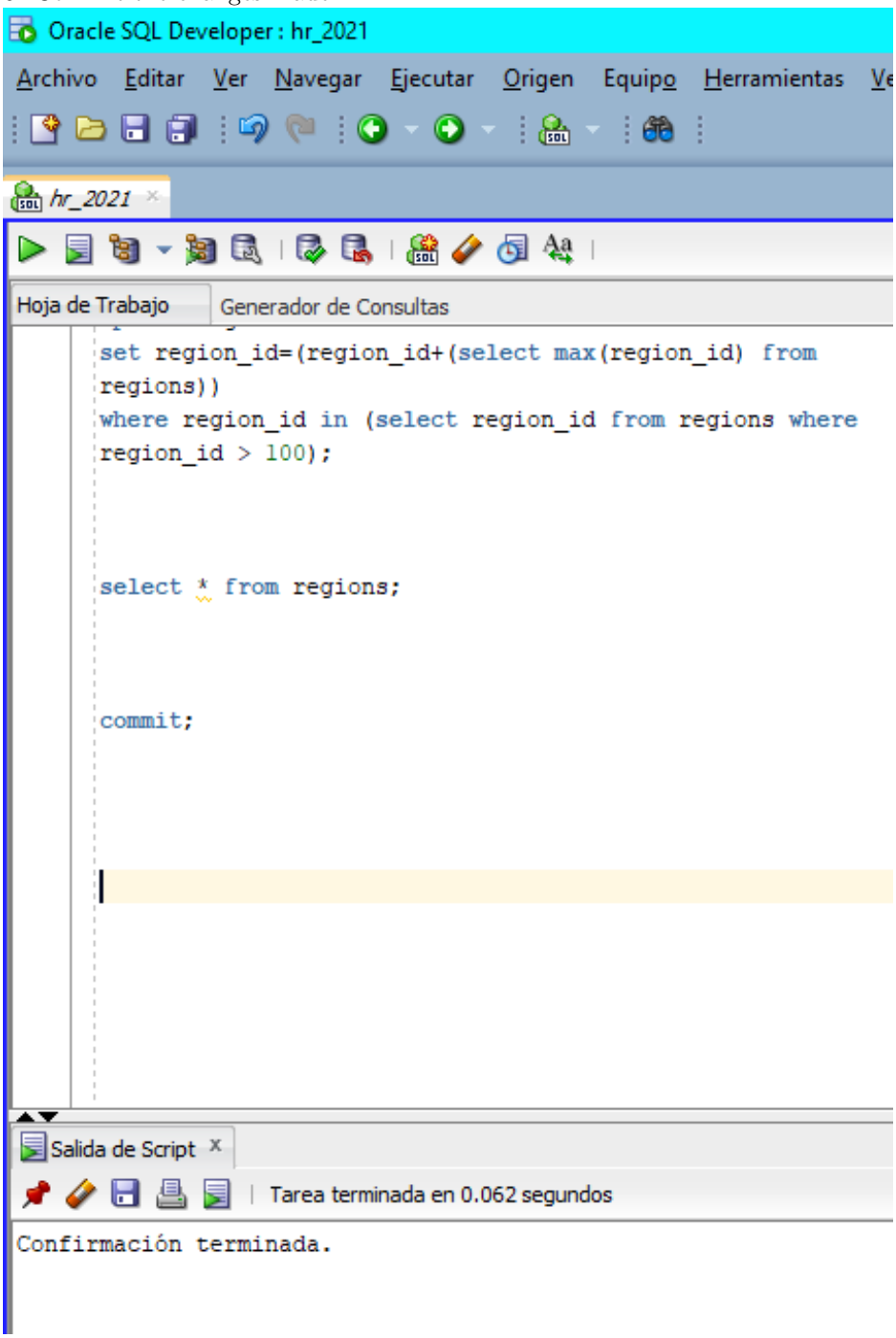



This statement should return the message "3 rows updated."

5. Confirm the state of the rows:



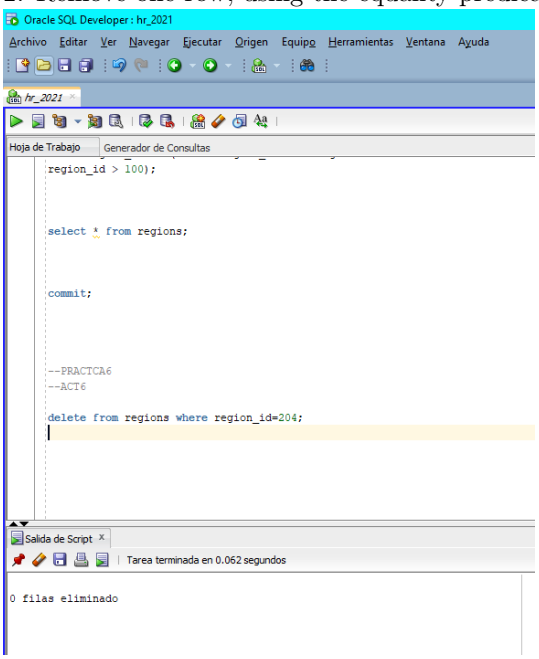
6. Commit the changes made:



## Activity 6:

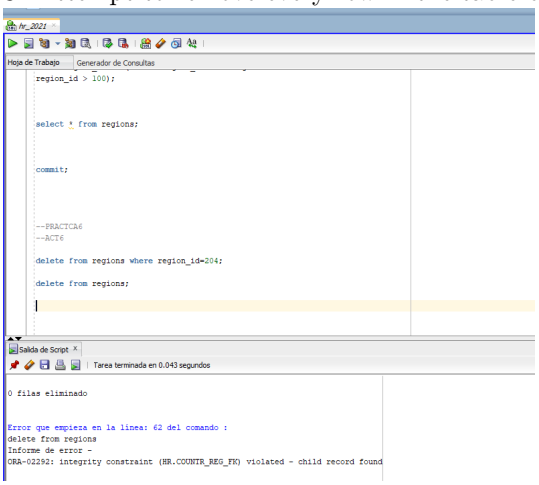
**Copy and paste screen results. Analyze the results for each sentence. In this section, use various techniques to delete rows in a table. Follow the next instructions: If not, adjust the values as necessary.**

1. Connect to the HR schema using SQL Developer.
2. Remove one row, using the equality predicate on the primary key:



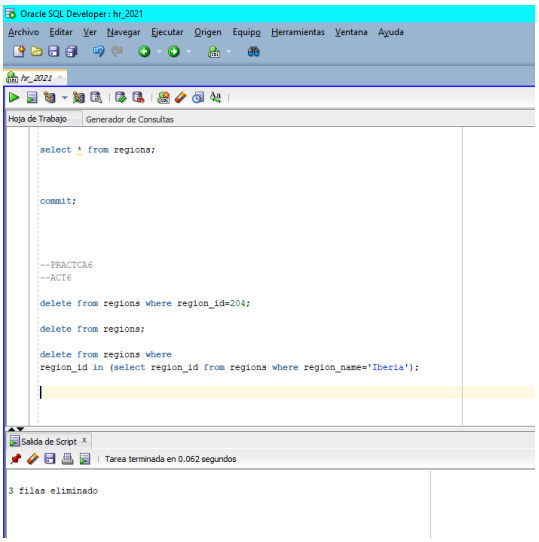

This should return the message "1 row deleted."

3. Attempt to remove every row in the table by omitting a WHERE clause:



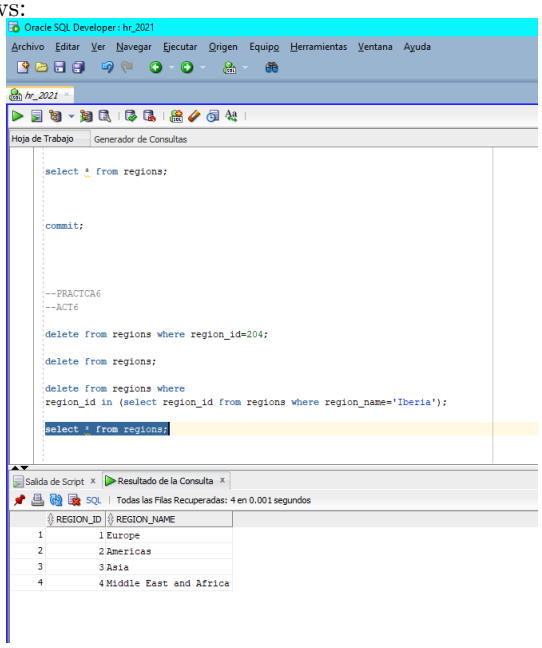
This will fail, due to a constraint violation.

4. Remove rows with the row selection based on a subquery:

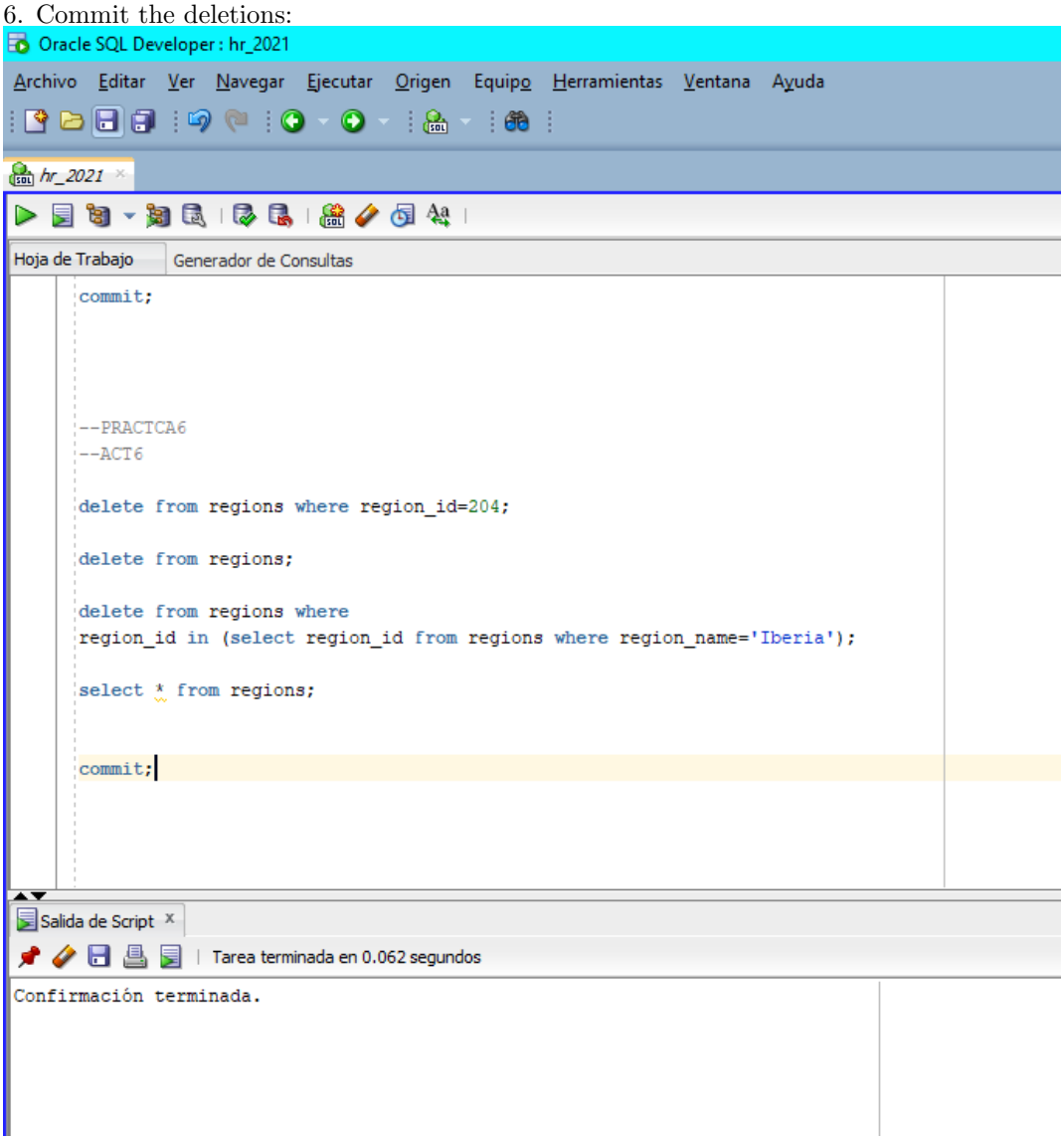



This will return the message “2 rows deleted.”

5. Confirm that the REGIONS table now contains just the original four rows:



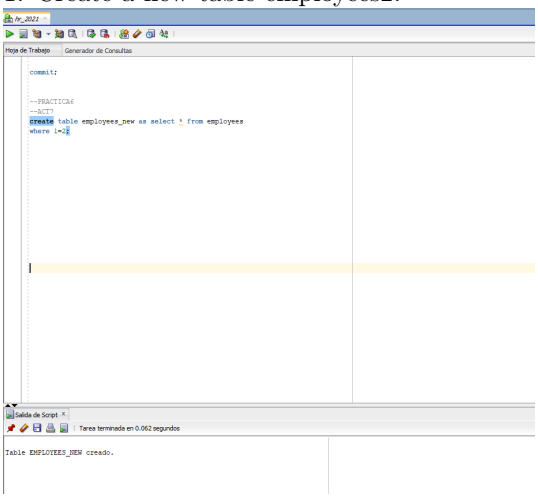
6. Commit the deletions:



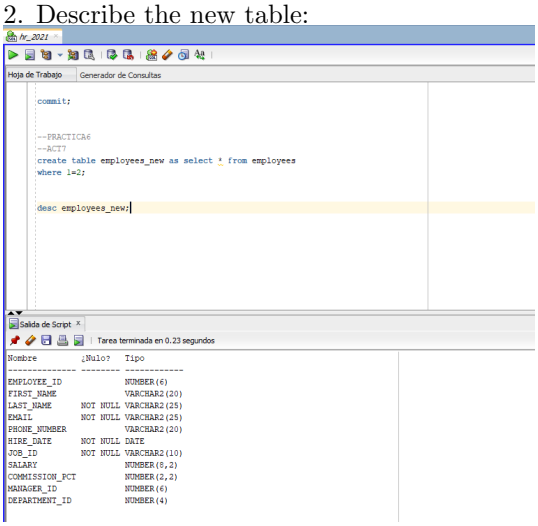
## Activity 7:

**The MERGE command is often ignored, because it does nothing that cannot be done with INSERT, UPDATE, and DELETE. It is, however, very powerful, in that with one pass through the data it can carry out all three operations. This can improve performance dramatically. Here is a simple example, do not forget to include and analyze the results:**

1. Create a new table employees2:




2. Describe the new table:



3. Insert a new row into table employees2:

```
commit;


--PRACTICA6
--ACT7
create table employees_new as select * from employees
where 1=2;


desc employees_new;


insert into employees_new select * from employees where
employee_id=100;
```

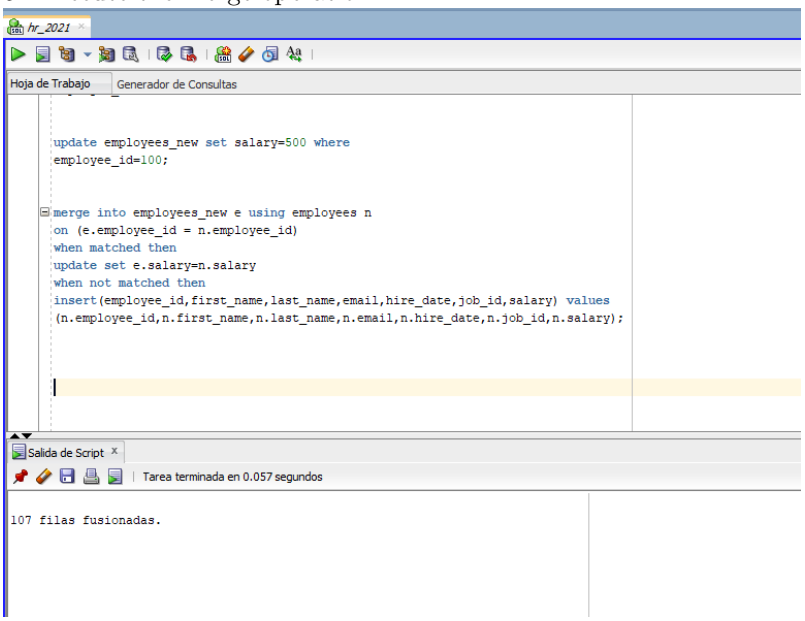
```
Salida de Script
Tarea terminada en 0.058 segundos

1 fila insertadas.
```

4. Update the salary of the inserted row:

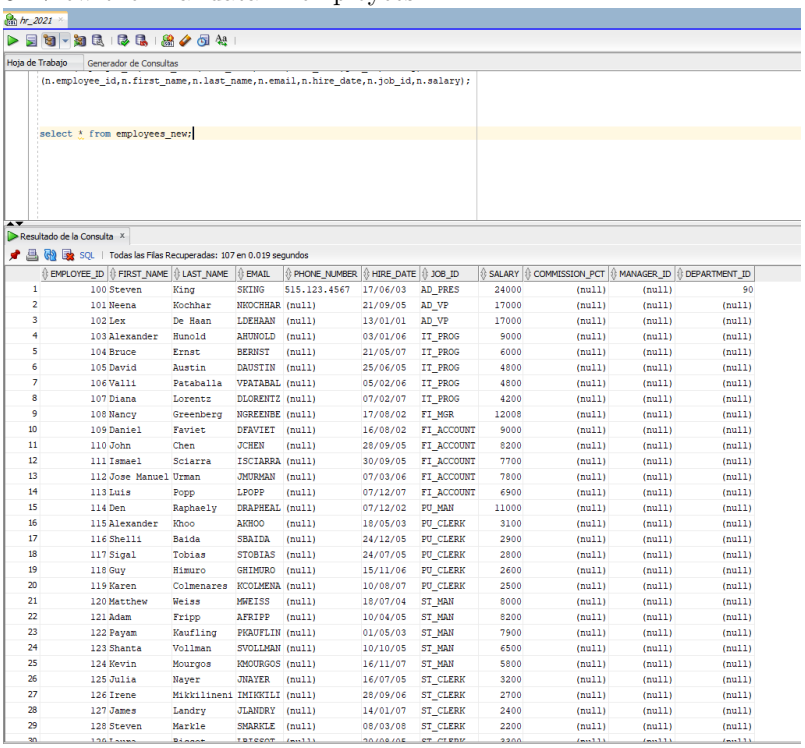
```
desc employees_new;


insert into employees_new select * from employees where
employee_id=100;


update employees_new set salary=500 where
employee_id=100;
```

```
Salida de Script
Tarea terminada en 0.059 segundos

1 fila actualizadas.
```

5. Execute the merge operation:




6. View the final data in employees2:




| | EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | COMMISSION_PCT | MANAGER_ID | DEPARTMENT_ID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | Steven | King | SKING | 515.123.4567 | 17/06/03 | AD_PRES | 24000 | (null) | (null) | 90 |
| 2 | 101 | Neena | Kochhar | NKOCHHAR | (null) | 21/09/05 | AD_VP | 17000 | (null) | (null) | (null) |
| 3 | 102 | Lex | De Haan | LDEHAAN | (null) | 13/01/01 | AD_VP | 17000 | (null) | (null) | (null) |
| 4 | 103 | Alexander | Hunold | AHUNOLD | (null) | 03/01/06 | IT_PROG | 9000 | (null) | (null) | (null) |
| 5 | 104 | Bruce | Ernst | BERNST | (null) | 21/05/07 | IT_PROG | 6000 | (null) | (null) | (null) |
| 6 | 105 | David | Austin | DAUSTIN | (null) | 25/06/05 | IT_PROG | 4800 | (null) | (null) | (null) |
| 7 | 106 | Valli | Pataballa | VPATABAL | (null) | 05/02/06 | IT_PROG | 4800 | (null) | (null) | (null) |
| 8 | 107 | Diana | Lorentz | DLORENTZ | (null) | 07/02/07 | IT_PROG | 4200 | (null) | (null) | (null) |
| 9 | 108 | Nancy | Greenberg | NGREENBE | (null) | 17/08/02 | FI_MGR | 12008 | (null) | (null) | (null) |
| 10 | 109 | Daniel | Faviet | DFAVIET | (null) | 16/08/02 | FI_ACCOUNT | 9000 | (null) | (null) | (null) |
| 11 | 110 | John | Chen | JCHEN | (null) | 28/09/05 | FI_ACCOUNT | 8200 | (null) | (null) | (null) |
| 12 | 111 | Ismael | Sciarra | ISCIARRA | (null) | 30/09/05 | FI_ACCOUNT | 7700 | (null) | (null) | (null) |
| 13 | 112 | Jose Manuel | Urman | JMURMAN | (null) | 07/03/06 | FI_ACCOUNT | 7800 | (null) | (null) | (null) |
| 14 | 113 | Luis | Popp | LPOPP | (null) | 07/12/07 | FI_ACCOUNT | 6900 | (null) | (null) | (null) |
| 15 | 114 | Den | Raphaely | DRAPHEAL | (null) | 07/12/02 | PU_MAN | 11000 | (null) | (null) | (null) |
| 16 | 115 | Alexander | Khoo | AKHOO | (null) | 18/05/03 | PU_CLERK | 3100 | (null) | (null) | (null) |
| 17 | 116 | Shelli | Baida | SBAIDA | (null) | 24/12/05 | PU_CLERK | 2900 | (null) | (null) | (null) |
| 18 | 117 | Sigal | Tobias | STOBIAS | (null) | 24/07/05 | PU_CLERK | 2800 | (null) | (null) | (null) |
| 19 | 118 | Guy | Himuro | GHIMURO | (null) | 15/11/06 | PU_CLERK | 2600 | (null) | (null) | (null) |
| 20 | 119 | Karen | Colmenares | KCOLMENA | (null) | 10/08/07 | PU_CLERK | 2500 | (null) | (null) | (null) |
| 21 | 120 | Matthew | Weiss | MWEISS | (null) | 18/07/04 | ST_MAN | 8000 | (null) | (null) | (null) |
| 22 | 121 | Adam | Fripp | AFRIPP | (null) | 10/04/05 | ST_MAN | 8200 | (null) | (null) | (null) |
| 23 | 122 | Payam | Kaufling | PKAUFLIN | (null) | 01/05/03 | ST_MAN | 7900 | (null) | (null) | (null) |
| 24 | 123 | Shanta | Vollman | SVOLLMAN | (null) | 10/10/05 | ST_MAN | 6500 | (null) | (null) | (null) |
| 25 | 124 | Kevin | Mourgos | KMOURGOS | (null) | 16/11/07 | ST_MAN | 5800 | (null) | (null) | (null) |
| 26 | 125 | Julia | Nayer | JNAYER | (null) | 16/07/05 | ST_CLERK | 3200 | (null) | (null) | (null) |
| 27 | 126 | Irene | Mikkilineni | IMIKKILI | (null) | 28/09/06 | ST_CLERK | 2700 | (null) | (null) | (null) |
| 28 | 127 | James | Landry | JLANDRY | (null) | 14/01/07 | ST_CLERK | 2400 | (null) | (null) | (null) |
| 29 | 128 | Steven | Markle | SMARKLE | (null) | 08/03/08 | ST_CLERK | 2200 | (null) | (null) | (null) |
| 30 | 129 | Laura | Bissot | LBISSOT | (null) | 20/08/05 | ST_CLERK | 3300 | (null) | (null) | (null) |

## Activity 8:

**Propose a solution to the following scenarios:**

a) Transactions, like constraints, are business rules: a technique whereby the database can enforce rules developed by business analysts. If the "logical unit of work" is huge, such as an accounting suite period rollover, should this actually be implemented as one transaction?

Answer: It could be implemented by one transaction because you can do all the work required and check it before finish the transaction with the possibility of roll back the work and later commit it when the work is done.

b) Being able to do DML operations, look at the result, then roll back and try them again can be very useful. But is it really a good idea?

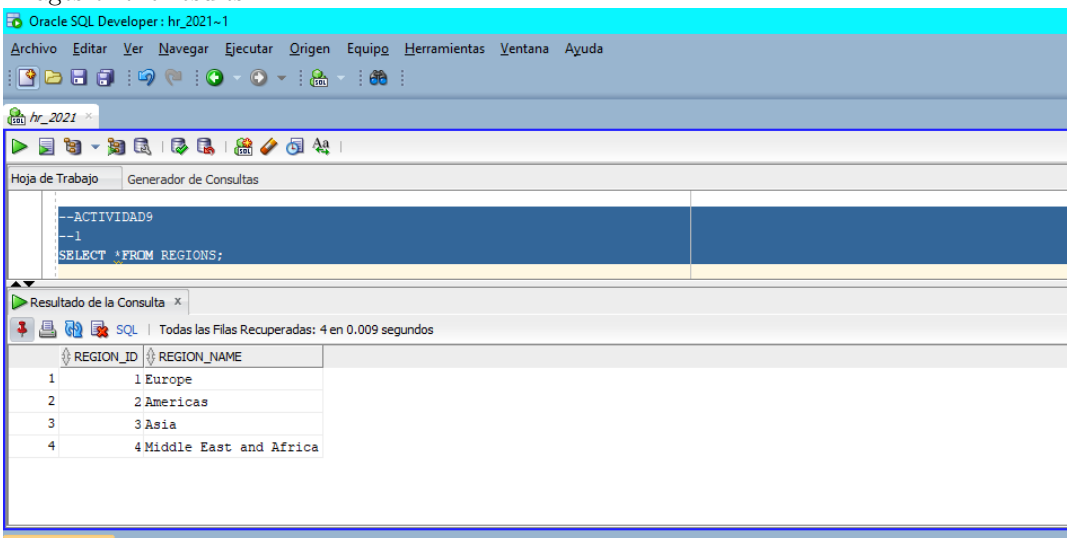
Answer: I think it depends of the quantity of work to do, imagine roll back and do again a huge work, maybe you can use save points to do the development of the work more efficient.

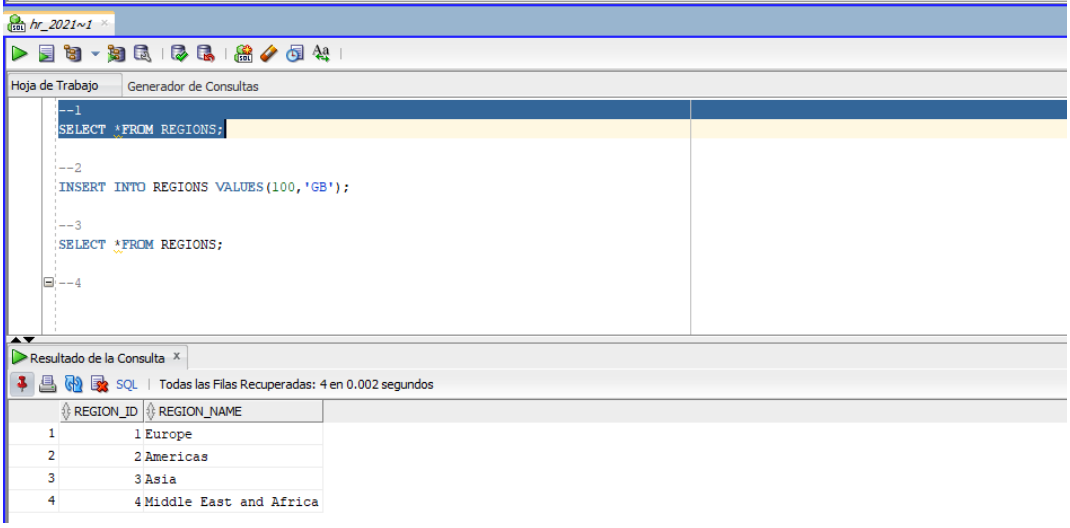
## Activity 9:

**In this exercise, demonstrate the use of transaction control statements and transaction isolation. Connect to the HR schema with two sessions concurrently. These can be two SQL Developer sessions. The following table lists steps to follow in each session.**

| Step | In your first session | In your second session |
|---|---|---|
| 1 | `select * from regions;` | `select * from regions;` |
| Both sessions see the same data. | | |
| 2 | `insert into regions values(100,'UK');` | `insert into regions values(101,'GB');` |
| 3 | `select * from regions;` | `select * from regions;` |
| Both sessions see different results: the original data, plus their own change. | | |
| 4 | `commit;` | |
| 5 | `select * from regions;` | `select * from regions;` |
| One transaction has been published to the world, the other is still visible to only one session. | | |
| 6 | `rollback;` | `rollback;` |
| 7 | `select * from regions;` | `select * from regions;` |
| The committed transaction was not reversed because it has already been committed, but the uncommitted one is now completely gone, having been terminated by rolling back the change. | | |
| 8 | `delete from regions where region_id=100;` | `delete from regions where region_id=101;` |
| 9 | `select * from regions;` | `select * from regions;` |
| Each deleted row is still visible in the session that did not delete it, until you do the following: | | |
| 10 | `commit;` | `commit;` |
| 11 | `select * from regions;` | `select * from regions;` |
| With all transactions terminated, both sessions see a consistent view of the table. | | |

Images of the results:



hr_2021

Hoja de Trabajo | Generador de Consultas

```
--2
INSERT INTO REGIONS VALUES(100,'UK');
```

Resultado de la Consulta | Salida de Script

Tarea terminada en 0.025 segundos

```
Error que empieza en la línea: 116 del comando :
INSERT INTO REGIONS VALUES(100,'UK')
Informe de error -
ORA-00001: unique constraint (HR.REG_ID_PK) violated
```

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--1
SELECT *FROM REGIONS;

--2
INSERT INTO REGIONS VALUES(100,'GB');

--3
SELECT *FROM REGIONS;

--4
```

Resultado de la Consulta | Salida de Script

Tarea terminada en 0.039 segundos

```
Error que empieza en la línea: 5 del comando :
INSERT INTO REGIONS VALUES(100,'GB')
Informe de error -
ORA-00001: unique constraint (HR.REG_ID_PK) violated
```

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--3
SELECT *FROM REGIONS;

--4
```

Resultado de la Consulta | Salida de Script | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 5 en 0.004 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--1
SELECT *FROM REGIONS;

--2
INSERT INTO REGIONS VALUES(101,'GB');

--3
SELECT *FROM REGIONS;

--4
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 6 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--4
COMMIT;
```

Resultado de la Consulta | Salida de Script | Resultado de la Consulta 1 | Resultado de la Consulta 2

Tarea terminada en 0.048 segundos

```
Error que empieza en la línea: 116 del comando :
INSERT INTO REGIONS VALUES(100,'UK')
Informe de error -
ORA-00001: unique constraint (HR.REG_ID_PK) violated

>>Query Run In:Resultado de la Consulta 2
Confirmación terminada.
Confirmación terminada.
```

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--3
SELECT *FROM REGIONS;

--4


--5
SELECT *FROM REGIONS;

--6
ROLLBACK;
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 6 en 0.001 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--5
SELECT *FROM REGIONS;
```

Resultado de la Consulta | Salida de Script | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Resultado de

SQL | Todas las Filas Recuperadas: 6 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--3
SELECT *FROM REGIONS;

--4


--5
SELECT *FROM REGIONS;

--6
ROLLBACK;
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 6 en 0.001 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--6
ROLLBACK;

--7
SELECT *FROM REGIONS;

--8
DELETE FROM REGIONS WHERE REGION_ID=100;

--9
SELECT *FROM REGIONS;


--10
COMMIT;
```

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--5
SELECT *FROM REGIONS;

--6
ROLLBACK;

--7
SELECT *FROM REGIONS;

--8
DELETE FROM REGIONS WHERE REGION_ID=101;
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

Tarea terminada en 0.039 segundos

```
1 fila insertadas.

Rollback terminado.
```

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--7
SELECT *FROM REGIONS;

--8
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 6 en 0.003 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--5
SELECT *FROM REGIONS;

--6
ROLLBACK;

--7
SELECT *FROM REGIONS;

--8
DELETE FROM REGIONS WHERE REGION_ID=101;
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 6 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |
| 5 | 100 | UK |
| 6 | 101 | GB |

Oracle SQL Developer : hr_2021~1

Archivo Editar Ver Navegar Ejecutar Origen Equipo Herramientas Ventana Ayuda

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--8
DELETE FROM REGIONS WHERE REGION_ID=100;

--9
SELECT *FROM REGIONS;
```

Salida de Script

Tarea terminada en 0.044 segundos

```
1 fila eliminado
```

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--7
SELECT *FROM REGIONS;

--8
DELETE FROM REGIONS WHERE REGION_ID=101;

--9
SELECT *FROM REGIONS;


--10
COMMIT;
```

Salida de Script

Tarea terminada en 0.047 segundos

```
1 fila eliminado
```

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--9
SELECT *FROM REGIONS;

--10
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 4 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--8
DELETE FROM REGIONS WHERE REGION_ID=101;

--9
SELECT *FROM REGIONS;

--10
COMMIT;

--11
SELECT *FROM REGIONS;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 4 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--10
COMMIT;
```

Salida de Script

Tarea terminada en 0.037 segundos

```
Confirmación terminada.
```

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--8
DELETE FROM REGIONS WHERE REGION_ID=101;

--9
SELECT *FROM REGIONS;

--10
COMMIT;

--11
SELECT *FROM REGIONS;
```

Salida de Script

Tarea terminada en 0.044 segundos

```
Confirmación terminada.
```

hr_2021

Hoja de Trabajo | Generador de Consultas

```
--11
SELECT *FROM REGIONS;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 4 en 0.002 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |

hr_2021~1

Hoja de Trabajo | Generador de Consultas

```
--8
DELETE FROM REGIONS WHERE REGION_ID=101;

--9
SELECT *FROM REGIONS;


--10
COMMIT;

--11
SELECT *FROM REGIONS;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 4 en 0.001 segundos

| | REGION_ID | REGION_NAME |
|---|---|---|
| 1 | 1 | Europe |
| 2 | 2 | Americas |
| 3 | 3 | Asia |
| 4 | 4 | Middle East and Africa |

## Activity 10:

**Write the section that describes the Work developed in the following activities (capture an image for each statement output)**

1. Create the DDL sentence to build the MYEMPLOYEE table used in this activity (see item 2). Save this sentences in script sql

Hoja de Trabajo | Generador de Consultas

```sql
CREATE TABLE MY_EMPLOYEE(
ID_EMPLOYEE NUMBER(4) CONSTRAINT  EMP_NN NOT NULL,
LAST_NAME VARCHAR2(25),
FIRST_NAME  VARCHAR2(25),
USER_ID VARCHAR(8),
SALARY NUMBER(9,2)
);



DESCRIBE MY_EMPLOYEE;


INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;

--ACT10-PARTE10
UPDATE MY_EMPLOYEE
  SET LAST_NAME='Drexler'
  WHERE ID_EMPLOYEE=3;
--11
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

Tarea terminada en 0.066 segundos

```
Table MY_EMPLOYEE creado.

Nombre      ¿Nulo?   Tipo
----------- -------- ------------
ID_EMPLOYEE NOT NULL NUMBER(4)
LAST_NAME            VARCHAR2(25)
FIRST_NAME           VARCHAR2(25)
USER_ID              VARCHAR2(8)
SALARY               NUMBER(9,2)
```

2. Describe the structure of the MYEMPLOYEE table to identify the column names.

3. Create an INSERT statement to add the first row of data to the MYEMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause

lab_06_01.sql | scriptprac6act2.sql | hr_2021

Hoja de Trabajo | Generador de Consultas

```sql
CREATE TABLE MY_EMPLOYEE(
ID_EMPLOYEE NUMBER(4) CONSTRAINT  EMP_NN NOT NULL,
LAST_NAME VARCHAR2(25),
FIRST_NAME  VARCHAR2(25),
USER_ID VARCHAR(8),
SALARY NUMBER(9,2)
);



DESCRIBE MY_EMPLOYEE;


INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);
```

Salida de Script

Tarea terminada en 0.047 segundos

```
1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.
```

4. Populate the MYEMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause

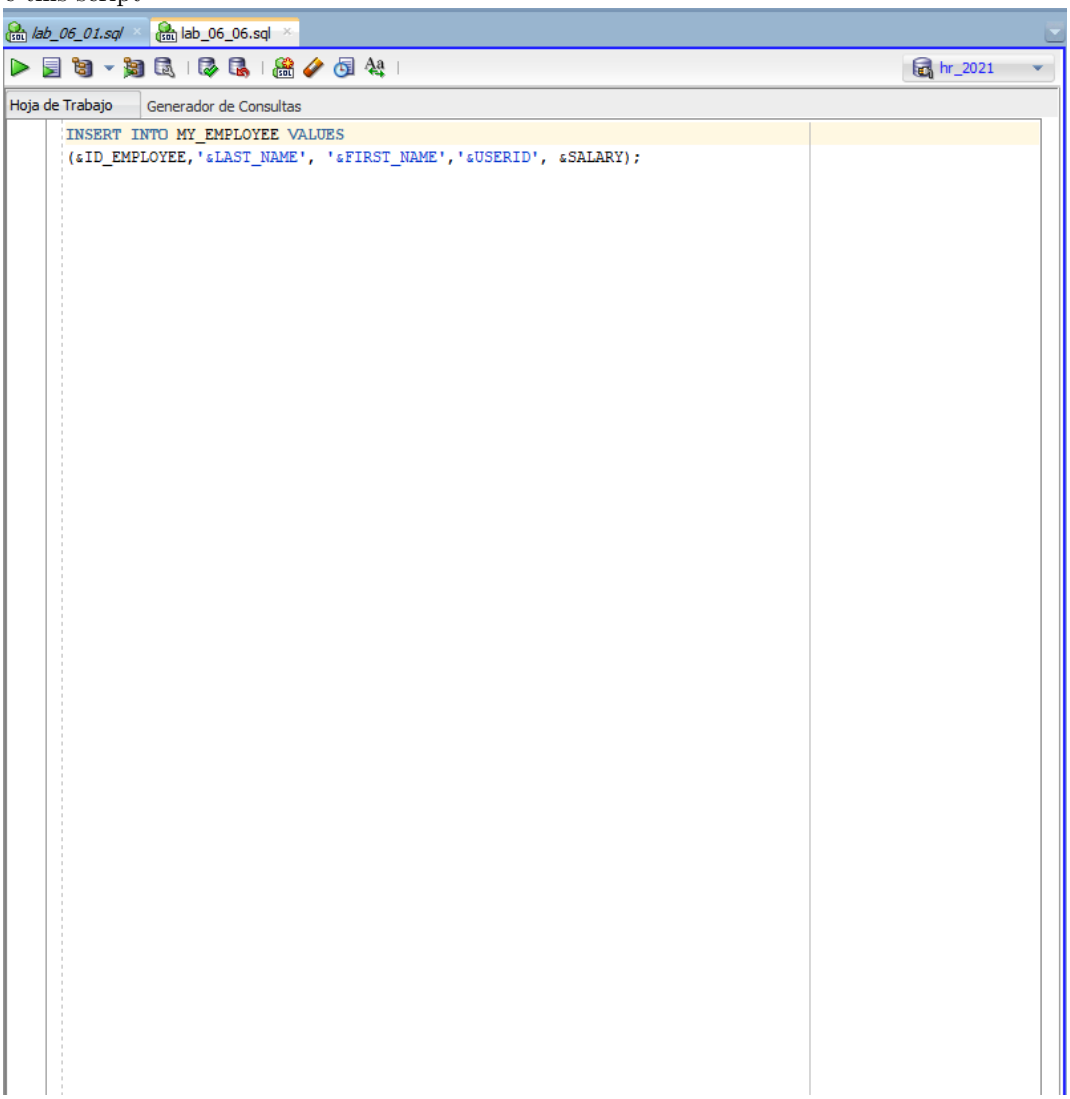
5. Confirm your addition to the table

lab_06_01.sql | scriptprac6act2.sql

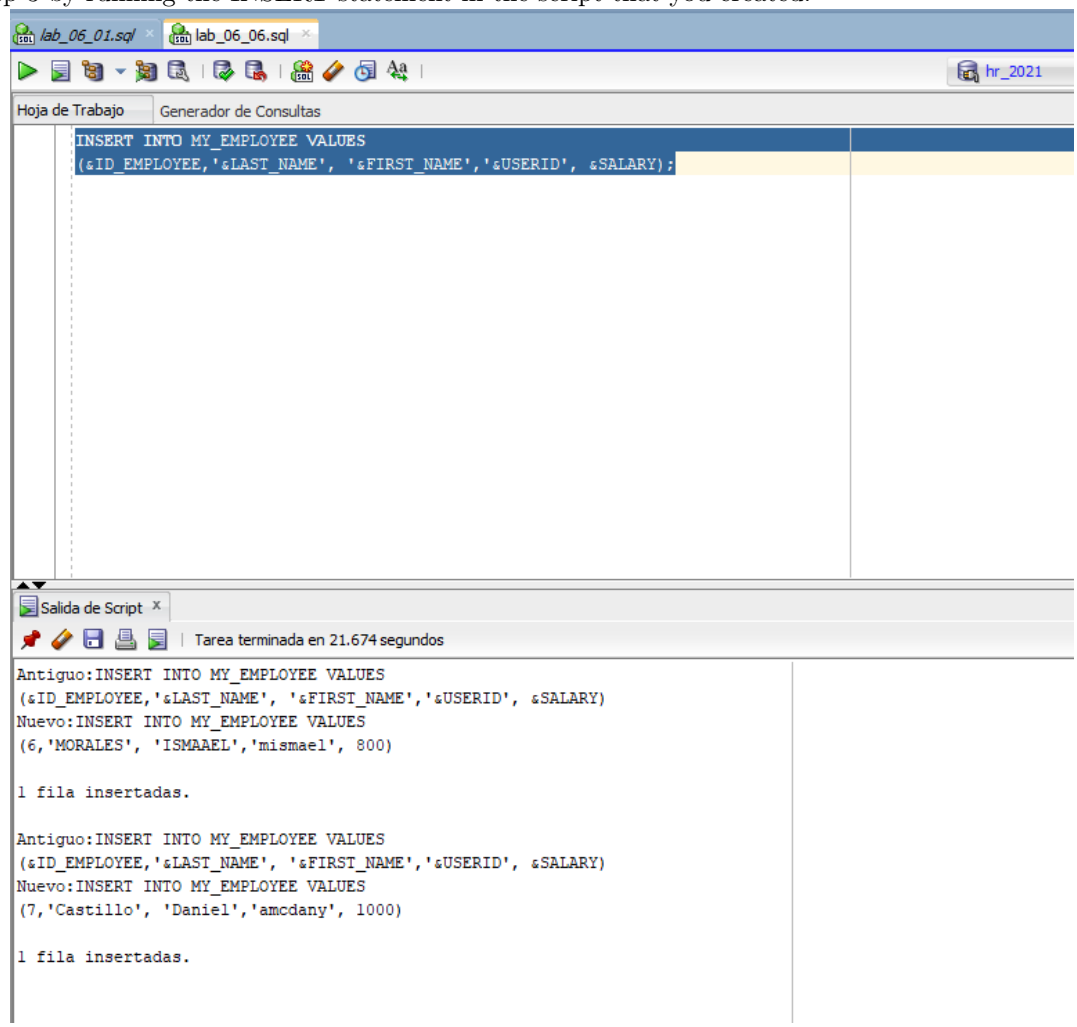
Hoja de Trabajo | Generador de Consultas

```sql
CREATE TABLE MY_EMPLOYEE(
ID_EMPLOYEE NUMBER(4) CONSTRAINT  EMP_NN NOT NULL,
LAST_NAME VARCHAR2(25),
FIRST_NAME  VARCHAR2(25),
USER_ID VARCHAR(8),
SALARY NUMBER(9,2)
);



DESCRIBE MY_EMPLOYEE;


INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;
```

6. Write an INSERT statement in a dynamic reusable script file to load the remaining rows into the MYEMPLOYEE table. The script should prompt for all the columns (ID, LASTNAME, FIRSTNAME, USERID, and SALARY). Save this script



7. Populate the table with the next two rows of the sample data listed in step 3 by running the INSERT statement in the script that you created.



8. Confirm your additions to the table.
9. Make the data additions permanent

lab_06_01.sql | lab_06_06.sql

Hoja de Trabajo | Generador de Consultas

```
INSERT INTO MY_EMPLOYEE VALUES
(&ID_EMPLOYEE,'&LAST_NAME', '&FIRST_NAME','&USERID', &SALARY);


commit;
```

Salida de Script

Tarea terminada en 0.029 segundos

```
Antiguo:INSERT INTO MY_EMPLOYEE VALUES
(&ID_EMPLOYEE,'&LAST_NAME', '&FIRST_NAME','&USERID', &SALARY)
Nuevo:INSERT INTO MY_EMPLOYEE VALUES
(6,'MORALES', 'ISMAAEL','mismael', 800)

1 fila insertadas.

Antiguo:INSERT INTO MY_EMPLOYEE VALUES
(&ID_EMPLOYEE,'&LAST_NAME', '&FIRST_NAME','&USERID', &SALARY)
Nuevo:INSERT INTO MY_EMPLOYEE VALUES
(7,'Castillo', 'Daniel','amcdany', 1000)

1 fila insertadas.

Confirmación terminada.
```

Update and delete data in the MYEMPLOYEE table.
10. Change the last name of employee 3 to Drexler.

lab_06_01.sql | lab_06_06.sql

Hoja de Trabajo | Generador de Consultas

```sql
CREATE TABLE MY_EMPLOYEE(
ID_EMPLOYEE NUMBER(4) CONSTRAINT  EMP_NN NOT NULL,
LAST_NAME VARCHAR2(25),
FIRST_NAME  VARCHAR2(25),
USER_ID VARCHAR(8),
SALARY NUMBER(9,2)
);



DESCRIBE MY_EMPLOYEE;


INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;

--ACT10-PARTE10
UPDATE MY_EMPLOYEE
  SET LAST_NAME='Drexler'
  WHERE ID_EMPLOYEE=3;
```

Salida de Script | Tarea terminada en 0.033 segundos

```
1 fila actualizadas.
```

11. Change the salary to 1,000 for all employees who have a salary less than 900.

lab_06_01.sql | lab_06_06.sql

Hoja de Trabajo | Generador de Consultas

```sql
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;

--ACT10-PARTE10
UPDATE MY_EMPLOYEE
  SET LAST_NAME='Drexler'
  WHERE ID_EMPLOYEE=3;
--11
UPDATE MY_EMPLOYEE set SALARY=1000 WHERE SALARY <
900;
--12
SELECT * FROM MY_EMPLOYEE;



DROP TABLE MY_EMPLOYEE;
```

Salida de Script | Tarea terminada en 0.04 segundos

```
1 fila insertadas.


1 fila insertadas.

Confirmación terminada.

1 fila actualizadas.


4 filas actualizadas.
```

12. Verify your changes to the table.

lab_06_01.sql | lab_06_06.sql

Hoja de Trabajo | Generador de Consultas

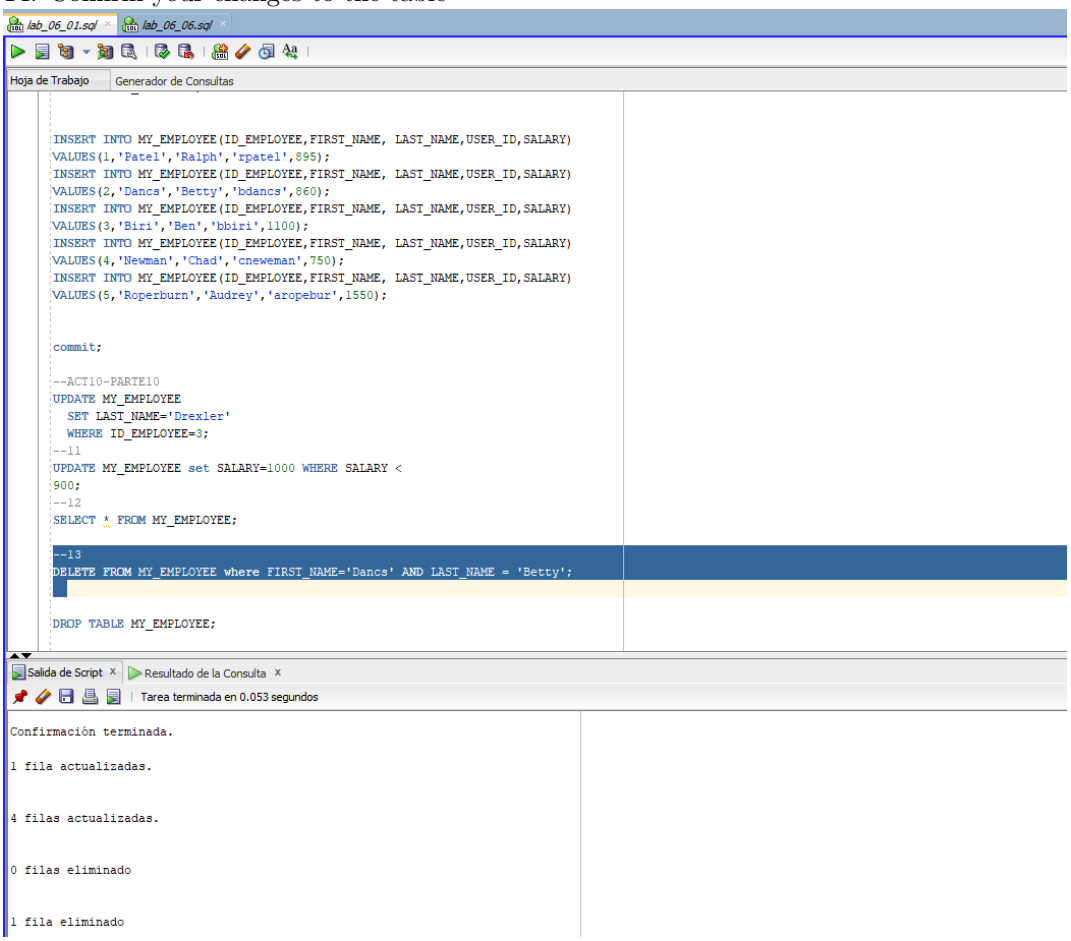
```sql
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(1,'Patel','Ralph','rpatel',895);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(2,'Dancs','Betty','bdancs',860);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(3,'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(4,'Newman','Chad','cneweman',750);
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;

--ACT10-PARTE10
UPDATE MY_EMPLOYEE
  SET LAST_NAME='Drexler'
  WHERE ID_EMPLOYEE=3;
--11
UPDATE MY_EMPLOYEE set SALARY=1000 WHERE SALARY <
900;
--12
SELECT * FROM MY_EMPLOYEE;


DROP TABLE MY_EMPLOYEE;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 7 en 0.003 segundos

| | ID_EMPLOYEE | LAST_NAME | FIRST_NAME | USER_ID | SALARY |
|---|---|---|---|---|---|
| 1 | 1 | Ralph | Patel | rpatel | 1000 |
| 2 | 2 | Betty | Dancs | bdancs | 1000 |
| 3 | 3 | Drexler | Biri | bbiri | 1100 |
| 4 | 4 | Chad | Newman | cneweman | 1000 |
| 5 | 5 | Audrey | Roperburn | aropebur | 1550 |
| 6 | 6 | MORALES | ISMAEL | mismael | 1000 |
| 7 | 7 | Castillo | Daniel | amcdany | 1000 |

13. Delete Betty Dancs from the MYEMPLOYEE table.
14. Confirm your changes to the table



15. Commit all pending changes.

```
INSERT INTO MY_EMPLOYEE(ID_EMPLOYEE,FIRST_NAME, LAST_NAME,USER_ID,SALARY)
VALUES(5,'Roperburn','Audrey','aropebur',1550);


commit;

--ACT10-PARTE10
UPDATE MY_EMPLOYEE
  SET LAST_NAME='Drexler'
  WHERE ID_EMPLOYEE=3;
--11
UPDATE MY_EMPLOYEE set SALARY=1000 WHERE SALARY <
900;
--12
SELECT * FROM MY_EMPLOYEE;

--13
DELETE FROM MY_EMPLOYEE where FIRST_NAME='Dancs' AND LAST_NAME = 'Betty';


COMMIT;


--15
SELECT * FROM MY_EMPLOYEE;


DROP TABLE MY_EMPLOYEE;
```

Todas las Filas Recuperadas: 6 en 0.001 segundos

| | ID_EMPLOYEE | LAST_NAME | FIRST_NAME | USER_ID | SALARY |
|---|---|---|---|---|---|
| 1 | 1 | Ralph | Patel | rpatel | 1000 |
| 2 | 3 | Drexler | Biri | bbiri | 1100 |
| 3 | 4 | Chad | Newman | cneweman | 1000 |
| 4 | 5 | Audrey | Roperburn | aropebur | 1550 |
| 5 | 6 | MORALES | ISMAEL | mismael | 1000 |
| 6 | 7 | Castillo | Daniel | amcdany | 1000 |

# 3 PRE-EVALUATION

Practices pre-Assessment for Database Systems Laboratory II Pre-Assessment
PRACTICE 6 carried out by student

1 COMPLIES WITH THE REQUESTED FUNCTIONALITY
YES

4 HAS THE CORRECT INDENTATION
YES

6 HAS AN EASY WAY TO ACCESS THE PROVIDED FILES
YES

7 HAS A REPORT WITH IDC FORMAT
YES

8 REPORT INFORMATION IS FREE OF SPELLING ERRORS
YES

9 DELIVERED IN TIME AND FORM
YES

10 IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)
YES,90 percent

# 4 Conclusion

This practice was very productive because i reviewed past topics like create tables and use sequences, i learned about the merge command and use two different sessions to watch how the transactions work

I also worked with the inserts and updates commands, I made modifications in tables with some specifications, I was really able to learn a lot and improve on some things that raised doubts.

It was a very long practice, but it helps us to learn those concepts that in theory can be left with doubts or not very understandable