Practice 6

Practice name	DML
Academic Program	Software Engineering
Subject name	Laboratory of Database Systems II
Unit	I. SQL.
Professor	Aldonso Becerra Sánchez
Due date	**September 22, 2022**
Due date with penalty	**September 23, 2022**
Elaboration date	**September 20, 2022**

Practice objective	Apply Oracle DML statements for several context situations.
Estimated time of completion	5 hours
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.

Reference 1:

1. Oracle Database 11g: SQL Fundamentals.

Reference 2:

2. Oracle Database SQL Language Reference 11g.

Reference 3:

Initial Activity:

Read the whole practice before start it.

Write the corresponding report, starting with the **Introduction** section.

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

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.

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.

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.

5. Study the result of this SELECT statement:

```
SQL> select * from t1;

C1          C2       C3       C4
```

```
---------- ---------- ---------- ----------
1          2          3          4
5          6          7          8
```

If you issue this statement:

insert into t1 (c1,c2) values(select c1,c2 from t1);

why will it fail? (Choose the best answer.)

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.

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.

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

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.

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.

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.

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.

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.

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

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.

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:

```
insert into customers
(customer_id,cust_first_name,cust_last_name)

values((select max(customer_id)+1 from
customers),'John','Watson');
```

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

```
update customers set

credit_limit=(select avg(credit_limit)

              from customers)

where cust_last_name='Watson';
```

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

```
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';
```

4. Change the name of the second entered customer:

```
update customers

set cust_last_name='Ramklass',cust_first_name='Roopesh'

where customer_id=(select max(customer_id) from
customers);
```

5. Commit this transaction:

```
commit;
```

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

```
select customer_id,cust_last_name from customers
where cust_last_name in (‘Watson’,‘Ramklass’) for update;
```

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

```
select customer_id,cust_last_name
from customers
where cust_last_name='Ramklass';
```

What happened?

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

```
update customers
set credit_limit=0
where cust_last_name='Ramklass';
```

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

```
commit;
```

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

```
delete from customers
where cust_last_name in ('Watson','Ramklass');
```

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

```
truncate table customers;
```

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:

```
commit;
```

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:

```
select max(customer_id)
from customers;
```

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_ID column:

```
select * from regions;
```

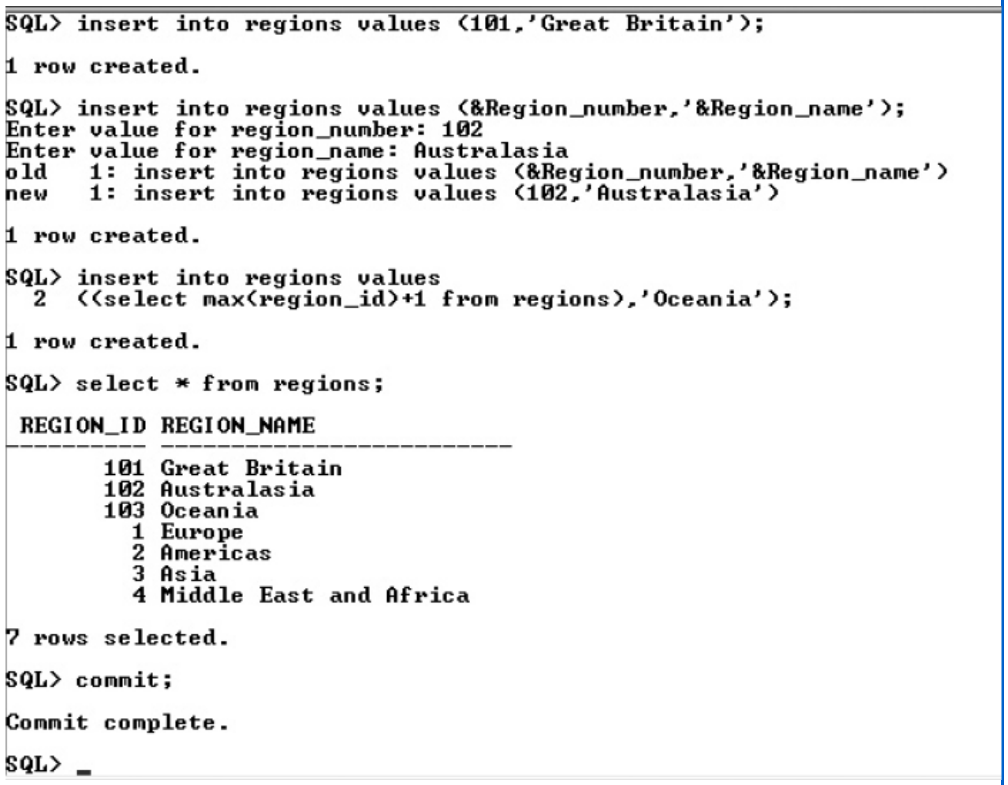
This exercise assumes that values above 100 are not in use. If they are, adjust the values suggested below to avoid primary key conflicts.

3. Insert a row into the REGIONS table, providing the values in line:
   ```
   insert into regions values (101,'Great Britain');
   ```
4. Insert a row into the REGIONS table, providing the values as substitution variables:
   ```
   insert into regions values
   (&Region_number,'&Region_name');
   ```

When prompted, give the values 102 for the number, Australasia for the name. Note the use of quotes around the string.

5. Insert a row into the REGIONS table, calculating the REGION_ID to be one higher than the current high value. This will need a scalar subquery:
   ```
   insert into regions values ((select
   max(region_id)+1 from regions), 'Oceania');
   ```
6. Confirm the insertion of the rows:
   ```
   select * from regions;
   ```
7. Commit the insertions:
   ```
   commit;
   ```

The final dataset must be identical to:



```
SQL> insert into regions values (101,'Great Britain');

1 row created.

SQL> insert into regions values (&Region_number,'&Region_name');
Enter value for region_number: 102
Enter value for region_name: Australasia
old   1: insert into regions values (&Region_number,'&Region_name')
new   1: insert into regions values (102,'Australasia')

1 row created.

SQL> insert into regions values
  2  ((select max(region_id)+1 from regions),'Oceania');

1 row created.

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
       101 Great Britain
       102 Australasia
       103 Oceania
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa

7 rows selected.

SQL> commit;

Commit complete.

SQL> _
```

NOTE: Capture an image for each statement output.

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:

```
update regions set region_name='Scandinavia' where
region_id=101;
```

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

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

```
update regions set region_name='Iberia' where region_id >
100;
```

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:

```
update regions

set region_id=(region_id+(select max(region_id) from
regions))

where region_id in (select region_id from regions where

region_id > 100);
```

This statement should return the message “3 rows updated.”

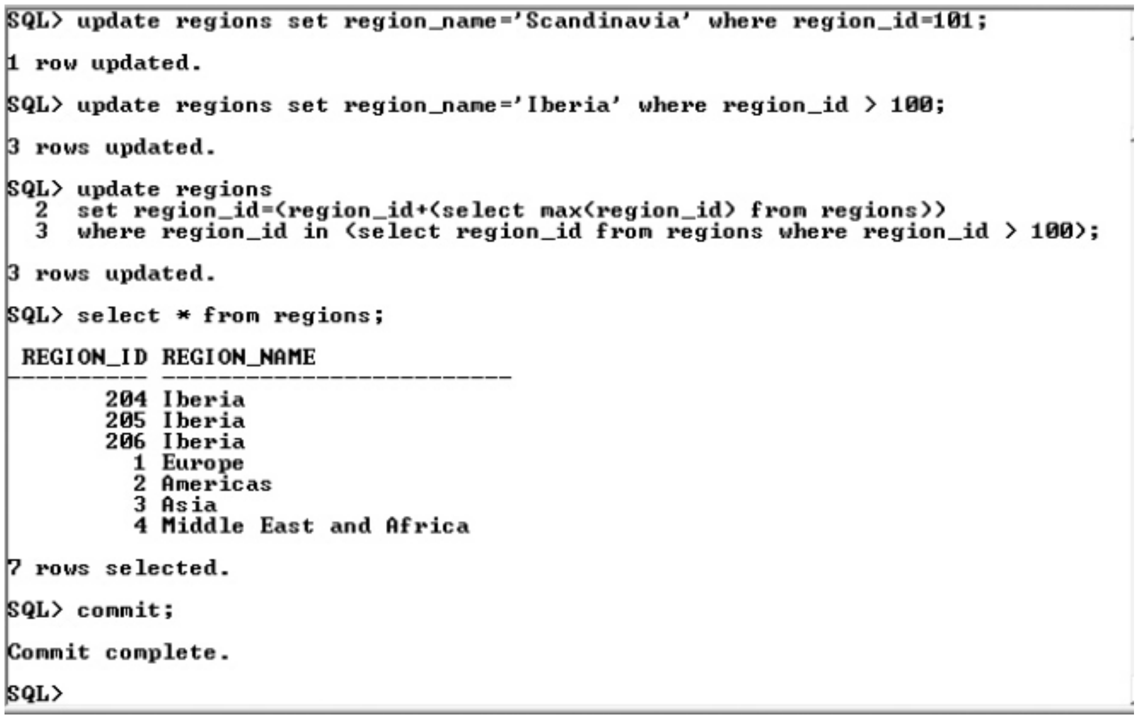
5. Confirm the state of the rows:

```
select * from regions;
```

6. Commit the changes made:

```
commit;
```

The final results must be identical to:



```
SQL> update regions set region_name='Scandinavia' where region_id=101;

1 row updated.

SQL> update regions set region_name='Iberia' where region_id > 100;

3 rows updated.

SQL> update regions
  2  set region_id=(region_id+(select max(region_id) from regions))
  3  where region_id in (select region_id from regions where region_id > 100);

3 rows updated.

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
       204 Iberia
       205 Iberia
       206 Iberia
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa

7 rows selected.

SQL> commit;

Commit complete.

SQL>
```

NOTE: Capture an image for each statement output.

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:

```
delete from regions where region_id=204;
```

This should return the message “1 row deleted.”

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

```
delete from regions;
```

This will fail, due to a constraint violation.

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

```
delete from regions where
region_id in (select region_id from regions where region_
name='Iberia');
```

This will return the message “2 rows deleted.”

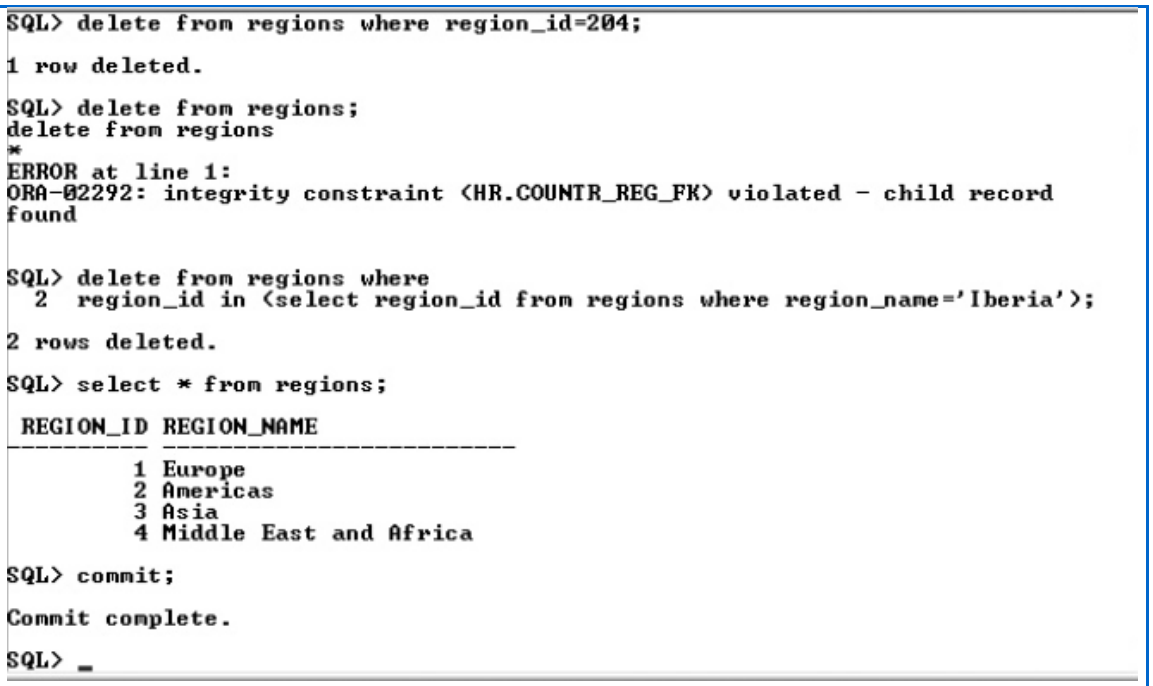
5. Confirm that the REGIONS table now contains just the original four rows:

```
select * from regions;
```

6. Commit the deletions:

```
commit;
```

The final results must be identical to:



```
SQL> delete from regions where region_id=204;

1 row deleted.

SQL> delete from regions;
delete from regions
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.COUNTR_REG_FK) violated - child record
found


SQL> delete from regions where
  2  region_id in (select region_id from regions where region_name='Iberia');

2 rows deleted.

SQL> select * from regions;

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

SQL> commit;

Commit complete.

SQL> _
```

NOTE: Capture an image for each statement output.

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:

```
   create table employees_new as select * from employees
where 1=2;
```

2. Describe the new table:
   `desc employees_new;`
3. Insert a new row into table employees2:
   `insert into employees_new select * from employees where employee_id=100;`
4. Update the salary of the inserted row:
   `update employees_new set salary=500 where employee_id=100;`
5. Execute the merge operation:

```
merge into employees_new e using employees n

     on (e.employee_id = n.employee_id)

when matched then

     update set e.salary=n.salary

when not matched then

     insert(employee_id,first_name,last_name,email,hire_da
     te,job_id,salary) values
     (n.employee_id,n.first_name,n.last_name,n.email,n.hir
     e_date,n.job_id,n.salary);
```

6. View the final data in employees2:

```
      select * from employees_new;
```

The preceding statement uses the contents of a table EMPLOYEES to update or insert rows in EMPLOYEES2. The situation could be that EMPLOYEES2 is a table of all staff, and EMPLOYEES is a table with rows for new staff and for salary changes for existing staff. The command will pass through EMPLOYEES2, and for each row, attempt to find a row in EMPLOYEES with the same EMPLOYEE_ID. If there is a row found, its SALARY column will be updated with the value of the row in NEW_EMPLOYEES. If there is not such a row, one will be inserted.

NOTE: Capture an image for each statement output.

Activity 8:

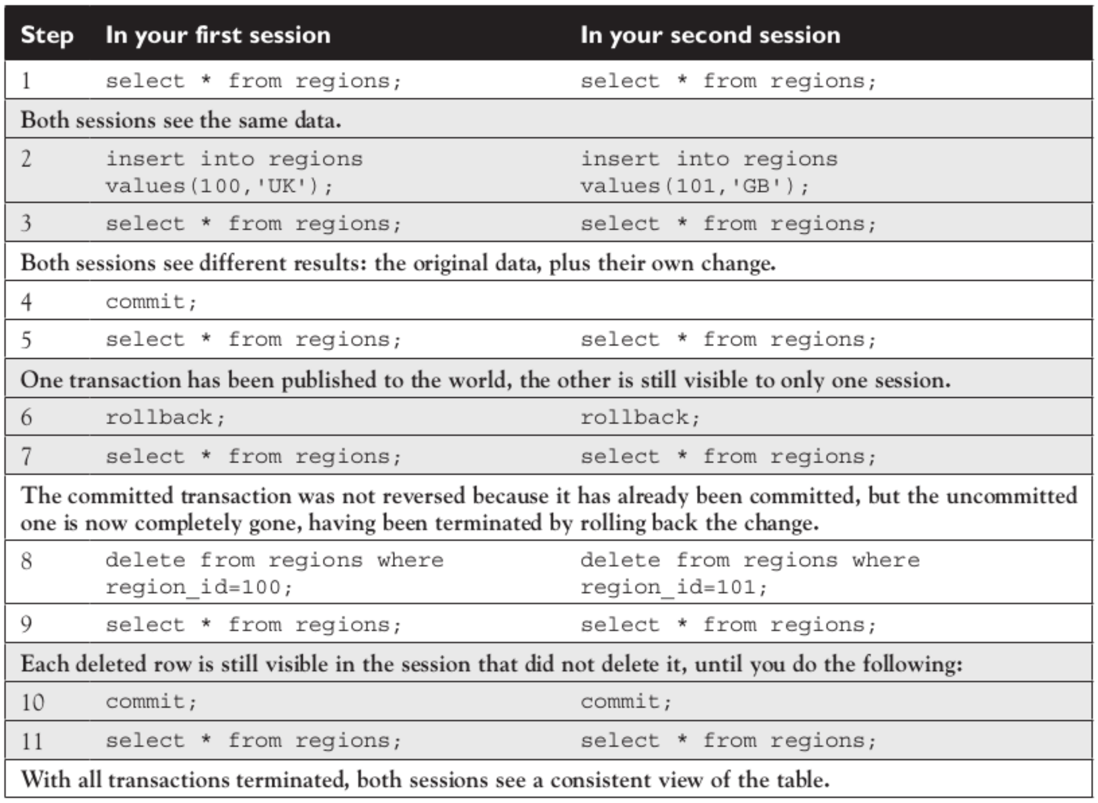
Propose a solution to the following scenarios:

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?

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?

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.		

Include an image of the results.

Activity 10:

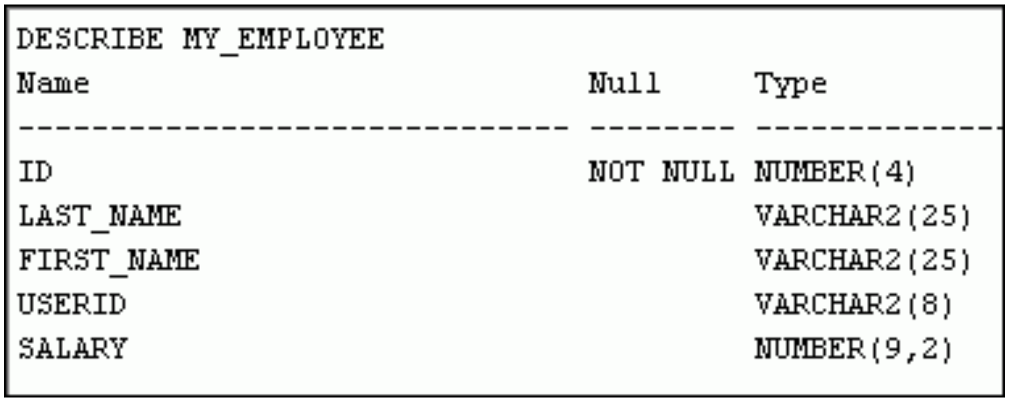
Write the section that describes the **Work developed** in the following activities (capture an image for each statement output).

The HR department wants you to create SQL statements to insert, update, and delete employee data. As a prototype, you use the `MY_EMPLOYEE` table before giving the statements to the HR department.

Note: For all the DML statements, use the Run Script icon (or press [F5]) to execute the query. This way you get to see the feedback messages on the Script Output tab page. For SELECT queries, continue to use the Execute Statement icon or press [F9] to get the formatted output on the Results tab page.

Insert data into the `MY_EMPLOYEE` table.

1. Create the DDL sentence to build the `MY_EMPLOYEE` table used in this activity (see item 2). Save this sentences in `lab_06_01.sql`.
2. Describe the structure of the `MY_EMPLOYEE` table to identify the column names.



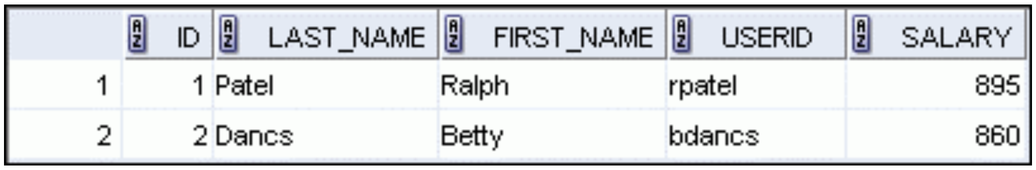
```
DESCRIBE MY_EMPLOYEE
Name                           Null     Type
------------------------------ -------- -------------
ID                             NOT NULL NUMBER(4)
LAST_NAME                               VARCHAR2(25)
FIRST_NAME                              VARCHAR2(25)
USERID                                  VARCHAR2(8)
SALARY                                  NUMBER(9,2)
```

3. Create an `INSERT` statement to add *the first row* of data to the `MY_EMPLOYEE` table from the following sample data. Do not list the columns in the `INSERT` clause. *Do not enter all rows yet.*

ID	**LAST_NAME**	**FIRST_NAME**	**USERID**	**SALARY**
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

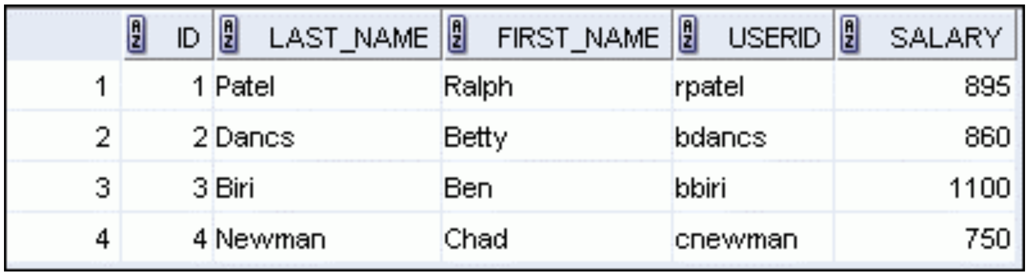
4. Populate the `MY_EMPLOYEE` table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the `INSERT` clause.

5. Confirm your addition to the table.



	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860

6. Write an `INSERT` statement in a dynamic reusable script file to load the remaining rows into the `MY_EMPLOYEE` table. The script should prompt for all the columns (`ID`, `LAST_NAME`, `FIRST_NAME`, `USERID`, and `SALARY`). Save this script to a `lab_06_06.sql` file.
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.

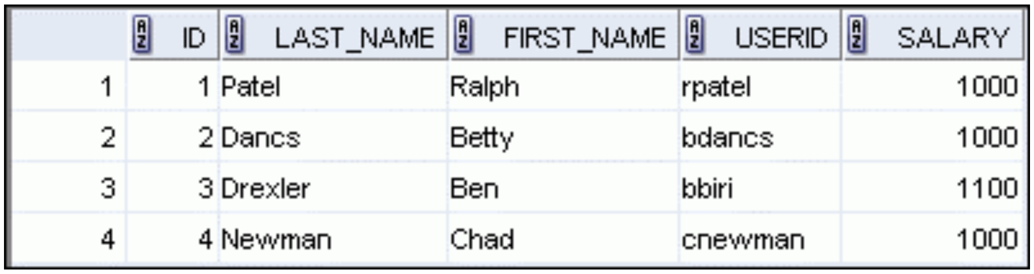


	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860
3	3	Biri	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	750

9. Make the data additions permanent.

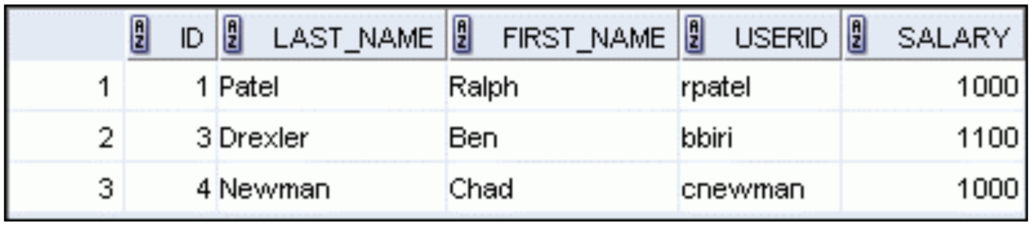
Update and delete data in the `MY_EMPLOYEE` table.

10. Change the last name of employee 3 to Drexler.
11. Change the salary to $1,000 for all employees who have a salary less than $900.
12. Verify your changes to the table.



	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	2	Dancs	Betty	bdancs	1000
3	3	Drexler	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	1000

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

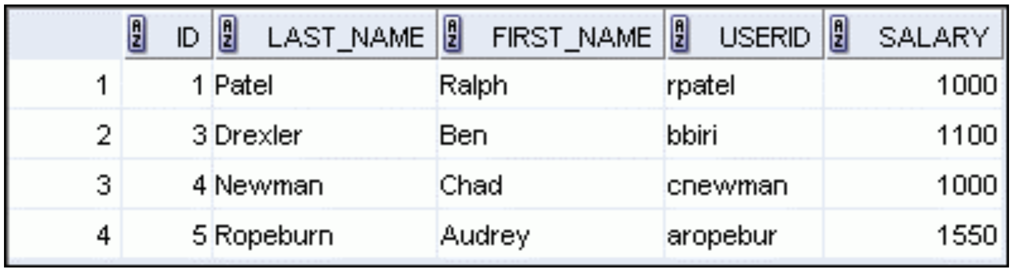


	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000

15. Commit all pending changes.

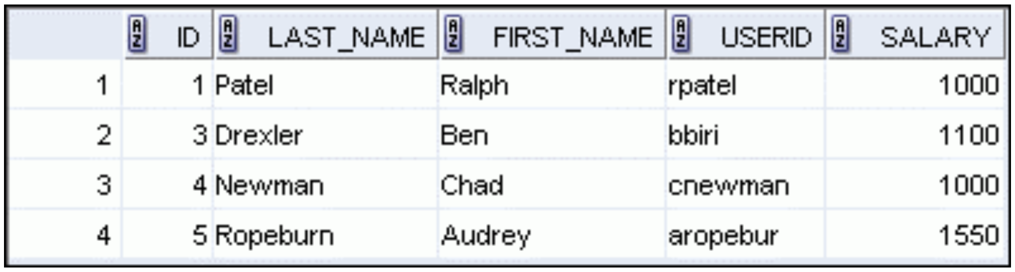
Control data transaction to the `MY_EMPLOYEE` table.

16. Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.
17. Confirm your addition to the table.



	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000
4	5	Ropeburn	Audrey	aropebur	1550

18. Mark an intermediate point in the processing of the transaction.
19. Delete all the rows from the `MY_EMPLOYEE` table.
20. Confirm that the table is empty.
21. Discard the most recent `DELETE` operation without discarding the earlier `INSERT` operation.
22. Confirm that the new row is still intact.



	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1	Patel	Ralph	rpatel	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000
4	5	Ropeburn	Audrey	aropebur	1550

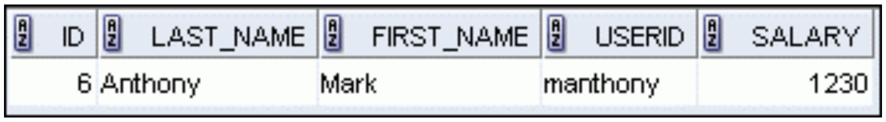
23. Make the data addition permanent.
24. Modify the `lab_06_06.sql` script such that the `USERID` is generated automatically by concatenating the first letter of the first name and the first

seven characters of the last name. The generated `USERID` must be in lowercase. Hence, the script should not prompt for the `USERID`. Save this script to a file named `lab_06_24.sql`.

25. Run the script, `lab_06_24.sql` to insert the following record:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26. Confirm that the new row was added with correct `USERID`.



ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

Activity 11:

Pre-assessment section.

Final activity:

Conclusion section.

Attached file that is required for this task (optional):

e-mail: a7donso@gmail.com