-- DBS211 Lab 06

-- Victor Krenzel

-- 102446176

-- 11/16/2022

--SET AUTOCOMMIT ON;

-- Part A (Transactions)

-- 1. List the 4 ways that we know that a transaction can be started

-- SET TRANSACTION READ WRITE;

-- BEGIN TRANSACTION;

-- COMMIT;

-- BEGIN;

-- 2. Using SQL, create an empty table, that is the same as the employees table, and name it newEmployees.

```
CREATE TABLE newEmployees
  AS (SELECT * FROM employees);
```

-- 3. Execute the following commands.

-- SET AUTCOMMIT OFF;

-- SET TRANSACTION READ WRITE;

```
SET AUTOCOMMIT OFF;
SET TRANSACTION READ WRITE;
```

-- >> Transaction READ succeeded.

-- 4. Write an INSERT statement to populate the newEmployees table with the rows of the sample data.

```
--    Insert the NULL value for the reportsTo column. (Write a single INSERT statement to insert all the rows)

INSERT ALL

   INTO newEmployees VALUES (100, 'Patel', 'Ralph', 22333, 'rpatel@mail.com', 1, NULL, 'Sales Rep')

   INTO newEmployees VALUES (101, 'Denis', 'Betty', 33444, 'bdenis@mail.com', 4, NULL, 'Sales Rep')

   INTO newEmployees VALUES (102, 'Biri', 'Ben', 44555, 'bbirir@mail.com', 3, NULL, 'Sales Rep')

   INTO newEmployees VALUES (103, 'Newman', 'Chad', 66777, 'cnewman@mail.com', 3, NULL, 'Sales Rep')

   INTO newEmployees VALUES (104, 'Ropeburn', 'Audrey', 77888, 'aropebur@mail.com', 1, NULL, 'Sales Rep')

   SELECT * FROM dual;


-- 5. Create a query that shows all the inserted rows from the newEmployees table. How many rows are selected?

SELECT * FROM newEmployees

WHERE reportsto IS NULL

AND UPPER(jobtitle) = 'SALES REP';

-- 5 Rows selected.


-- 6. Execute the rollback command. Display all rows and columns from the newEmployees table. How many rows are selected?

ROLLBACK;


SELECT * FROM newEmployees; -- 23 Rows selected.


-- 7. Repeat Task 4. Make the insertion permanent to the table newEmployees. Display all rows and columns from the newEmployee table. How many rows are selected?

SET TRANSACTION READ WRITE; -- <== Start  a new transaction


INSERT ALL

   INTO newEmployees VALUES (100, 'Patel', 'Ralph', 22333, 'rpatel@mail.com', 1, NULL, 'Sales Rep')
```

```
   INTO newEmployees VALUES (101, 'Denis', 'Betty', 33444, 'bdenis@mail.com', 4, NULL, 'Sales Rep')

   INTO newEmployees VALUES (102, 'Biri', 'Ben', 44555, 'bbirir@mail.com', 3, NULL, 'Sales Rep')

   INTO newEmployees VALUES (103, 'Newman', 'Chad', 66777, 'cnewman@mail.com', 3, NULL, 'Sales
Rep')

   INTO newEmployees VALUES (104, 'Ropeburn', 'Audrey', 77888, 'aropebur@mail.com', 1, NULL, 'Sales
Rep')

   SELECT * FROM dual;


COMMIT; -- <== Makes the newEmployees insertion permanent


SELECT * FROM newEmployees; -- 28 rows selected.


-- 8. Write an update statement to update the value of column jobTitle to ‘unknown’ for all the
employees in the newEmployees table.

UPDATE newEmployees

SET jobtitle = 'unknown';


-- 9. Make your changes permanent.

COMMIT;


-- 10. Execute the rollback command.

--        a. Display all employees from the newEmployees table whose job title is ‘unknown’. How many
rows are still updated?

--        b. Was the rollback command effective?

--        c. What was the difference between the result of the rollback execution from Task 6 and the
result of the rollback execution of this task?

ROLLBACK;


-- a.

SELECT * FROM newEmployees
```

```
WHERE LOWER(jobtitle) = 'unknown';

-- All the rows are still updated (28 rows)


-- b.

-- No the rollback was not effective.


-- c.

-- The difference is that this rollback is out of scope (task 8-9), meaning it rolledback nothing because this scope is empty.


-- 11. Begin a new transaction and then create a statement to delete to employees from the newEmployees table

SET TRANSACTION READ WRITE; -- <== Begins a new transaction


DELETE FROM newEmployees; -- <== Deletes all employees from newEmployees table.


-- 12. Create a VIEW, called vwNewEmps, that queries all the records in the newEmployees table sorted by last name and then by first name.

CREATE VIEW vwNewEmps AS

SELECT *

FROM newEmployees

ORDER BY lastname, firstname ASC;


-- 13. Perform a rollback to undo the deletion of the employees

--      a. How many employees are now in the newEmployees table?

--      b. Was the rollback effective and why?

ROLLBACK;


-- a.

SELECT * FROM newEmployees;
```

-- 0 employees

-- b.

-- No, the rollback was not effictive as it only rolled back the view creation command.

-- 14. Begin a new transaction and rerun the data insertion from Task 4 (copy the code down to Task 14 andrun it)

SET TRANSACTION READ WRITE;

INSERT ALL

INTO newEmployees VALUES (100, 'Patel', 'Ralph', 22333, 'rpatel@mail.com', 1, NULL, 'Sales Rep')

INTO newEmployees VALUES (101, 'Denis', 'Betty', 33444, 'bdenis@mail.com', 4, NULL, 'Sales Rep')

INTO newEmployees VALUES (102, 'Biri', 'Ben', 44555, 'bbirir@mail.com', 3, NULL, 'Sales Rep')

INTO newEmployees VALUES (103, 'Newman', 'Chad', 66777, 'cnewman@mail.com', 3, NULL, 'Sales Rep')

INTO newEmployees VALUES (104, 'Ropeburn', 'Audrey', 77888, 'aropebur@mail.com', 1, NULL, 'Sales Rep')

SELECT * FROM dual;

-- 15. Set a Savepoint, called insertion, after inserting the data

SAVEPOINT insertion;

-- 16. Rerun the update statement from Task 8 and run a query to view the data (copy the code down and run it again)

UPDATE newEmployees

SET jobtitle = 'unknown';

-- 17. Rollback the transaction to the Savepoint created in task 15 above and run a query to view the data. What does the data look like (i.e. describe what happened?

ROLLBACK TO insertion;

```
SELECT * FROM newEmployees;

-- All the employees report to no one (reportsto = null)


-- 18. Use the basic for of the rollback statement and again view the data. Describe what the results look like and what happened.

ROLLBACK;


SELECT * FROM newEmployees;

-- 0 rows fetched, the rollback was to before the insert all statement which is the beginning of the transaction.


-- 19.  Write a statement that denies all access to the newemployees table for all public users

REVOKE ALL ON newEmployees FROM public;


-- 20. Write a statement that allows a classmate (use their database login) read only access to the newemployees table.

-- IDK any classmate database logins so i'll just use the word classmate

GRANT READ, WRITE ON newEmployees TO classmate;


-- 21. Write a statement that allows the same classmate to modify (insert, update and delete) the data of the newemployees table

GRANT INSERT, UPDATE, DELETE ON newEmployees TO classmate;


-- 22. Write a statement the denies all access to the newemployees table for the same classmate.

REVOKE ALL ON newEmployees FROM classmate;


-- 23. Write statements to permanently remove the view and table created for this lab

DROP TABLE newEmployees;

DROP VIEW vwNewEmps;
```