Lab 06 – Transactions and Security

# Objectives:

The purpose of this lab is to introduce the student to both transactions and security. In the real-world, databases tasks often involve multiple steps and if any step in the middle fails, the procedure is a failure. This lap walks the student through a couple transactions and lets them learn how various steps have varying consequences that they need to be aware of.

By the end of this lab, the student will be able to:

* Describe the steps of a transaction, how a transaction begins and ends and walk through live scenarios of a variety of transactions
* Understand and act appropriately on what needs to be done in the case of transaction failure
* Grant and revoke permissions to and from other users and public users from the database

# Submission:

***Your submission will be a single WORD file with the query and result screenshot from Oracle SQL developer***

Make sure every SQL statement terminates with a semicolon.

* You will use following data to complete the given tasks:
* ***SET TRANSACTION READ WRITE*** starts a new transaction.
* ***COMMIT*** commits the current transaction, making its changes permanent.
* ***SAVEPOINT <name>***  sets a pointer to a location that can be rolled back to.
* ***ROLLBACK*** rolls back the current transaction, canceling its changes.
* ***SET autocommit*** disables or enables the default ***autocommit*** mode for the current session.

**Tasks:**

It is very important that these tasks/questions be performed in the order presented here for maximum learning.

## PART A - Transactions

1. Execute the following commands.

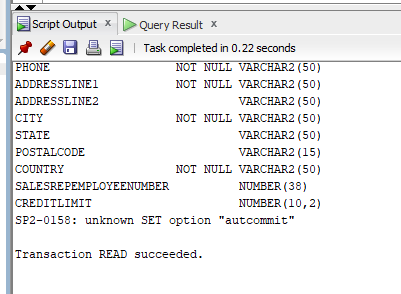
SET AUTCOMMIT OFF;  
 SET TRANSACTION READ WRITE;

Using SQL, create an **empty** table, that is the same as the RETAILCUSTOMERS table, and name it

***newCustomers***.

**SET AUTCOMMIT OFF;**

**SET TRANSACTION READ WRITE;**



**CREATE TABLE newCustomers AS**

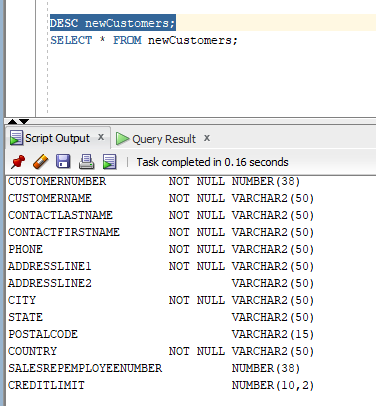
**SELECT \***

**FROM retailcustomers**

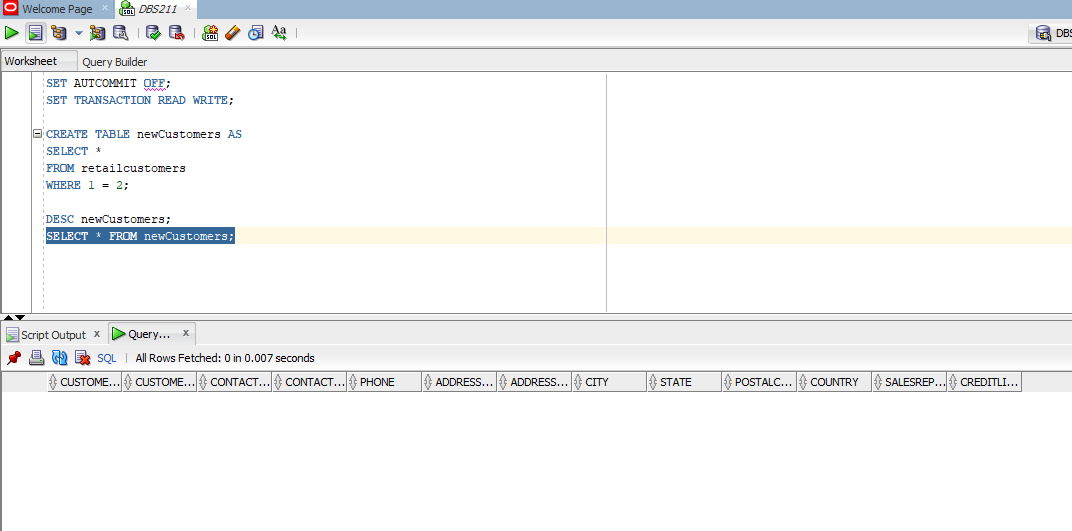
**WHERE 1 = 2;**

**TO describe table:-**

**DESC newCustomers;**



**SELECT \* FROM newCustomers;**



1. Write an INSERT statement to populate the **newCustomers** table with the rows of the sample data. (Write a single INSERT statement to insert all the rows, combine firstname and last name to get full name during insert)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **customerNumber** | **contactLastName** | **contactFirstName** | **Phone** | **addressLine1** | **city** | **country** |
| 100 | Patel | Ralph | 2233355555 | 10 SenecaWay | Paris | France |
| 101 | Denis | Betty | 3344455555 | 110 SenecaWay | Chicago | USA |
| 102 | Biri | Ben | 44555445544 | 13000 SenecaWay | Toronto | Canada |
| 103 | Newman | Chad | 66777332233 | 12 SenecaWay | Mexico city | Mexico |
| 104 | Ropeburn | Audrey | 7788811212 | 15000 SenecaWay | Havana | Cuba |
| 105 | Lucy | Preston | 45555511111 | 12 SenecaWay | Charlotte | USA |

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES (100, 'Patel', 'Ralph', '2233355555', '10 SenecaWay', 'Paris', 'France', 'Ralph Patel');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(101, 'Denis', 'Betty', '3344455555', '110 SenecaWay', 'Chicago', 'USA', 'Betty Denis');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(102, 'Biri', 'Ben', '44555445544', '13000 SenecaWay', 'Toronto', 'Canada', 'Ben Biri');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

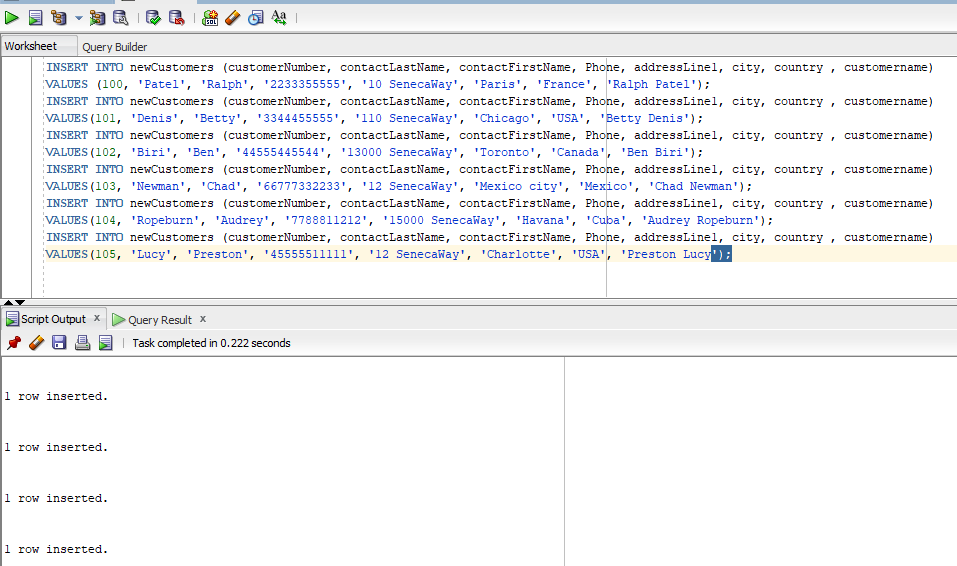
**VALUES(103, 'Newman', 'Chad', '66777332233', '12 SenecaWay', 'Mexico city', 'Mexico', 'Chad Newman');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(104, 'Ropeburn', 'Audrey', '7788811212', '15000 SenecaWay', 'Havana', 'Cuba', 'Audrey Ropeburn');**

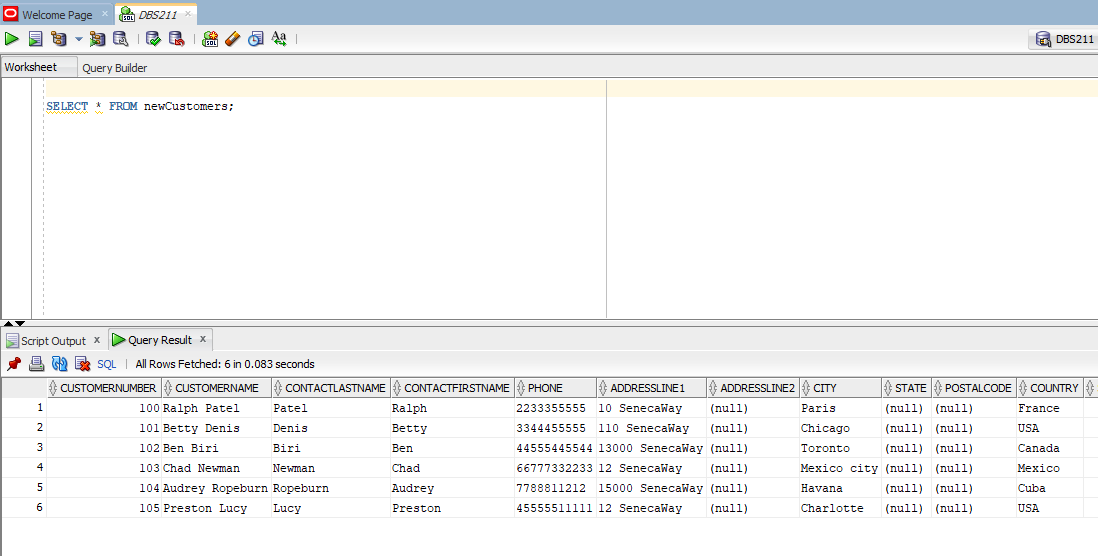
**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(105, 'Lucy', 'Preston', '45555511111', '12 SenecaWay', 'Charlotte', 'USA', 'Preston Lucy');**

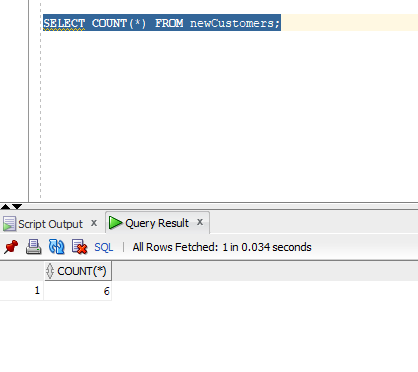


1. Create a query that shows all the inserted rows from the newCustomers table. How many rows are selected?

**SELECT \* FROM newCustomers;**

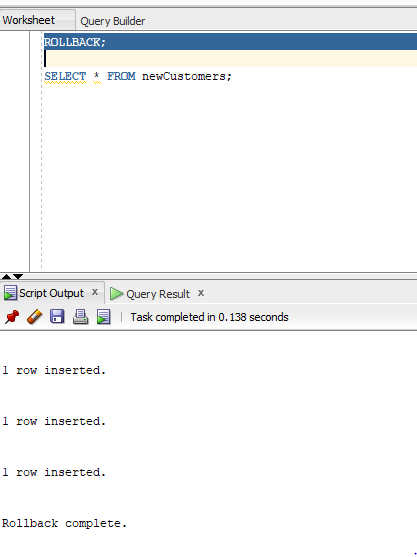


**SELECT COUNT(\*) FROM newCustomers;**

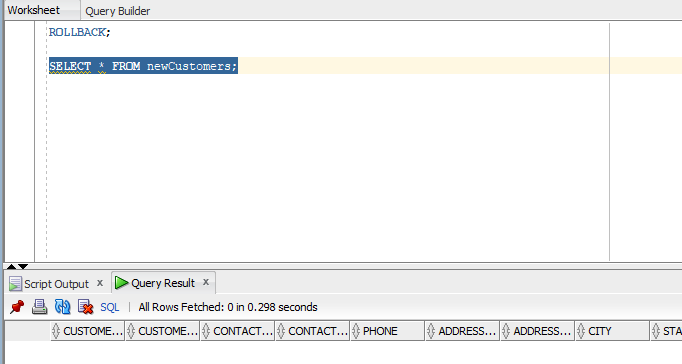


1. Execute the rollback command. Display all rows and columns from the newCustomers table. How many rows are selected?

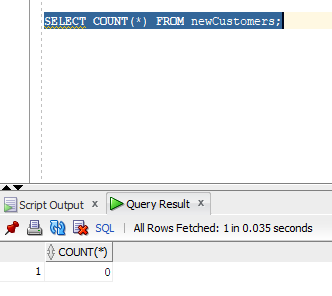
ROLLBACK;



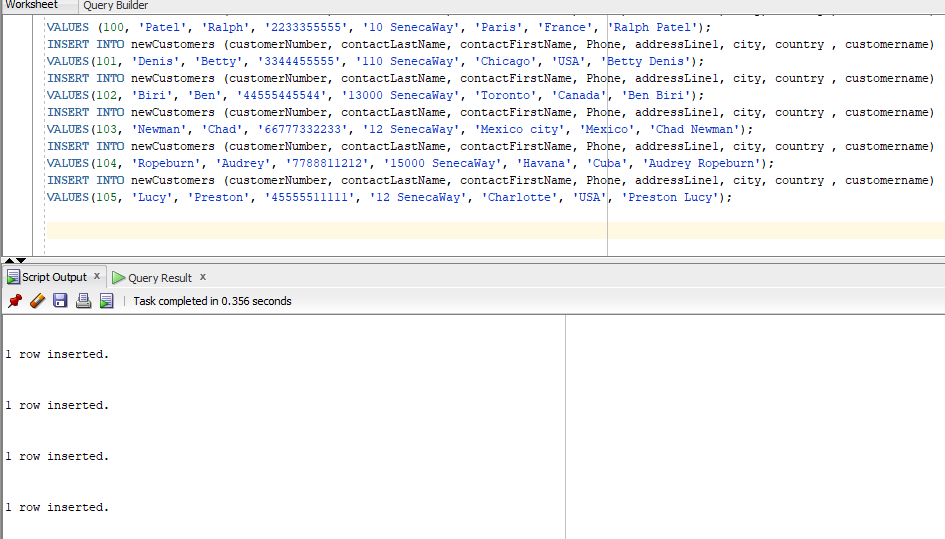
**SELECT \* FROM newCustomers;**



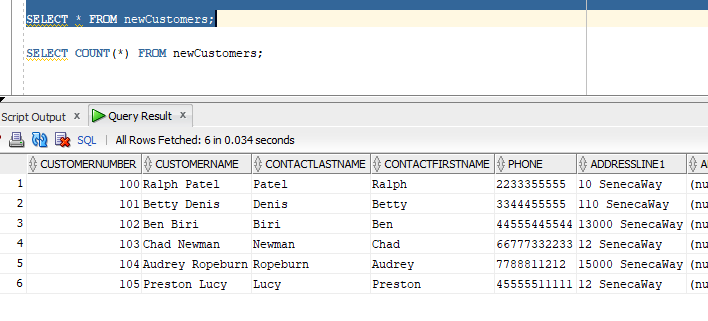
**SELECT COUNT(\*) FROM newCustomers;**



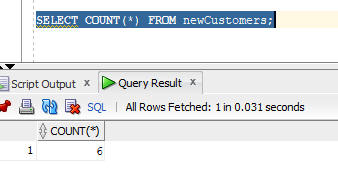
1. Repeat Question 2. Make the insertion permanent to the table newCustomers. Display all rows and columns from the newCustomers table. How many rows are selected?



**SELECT \* FROM newCustomers;**



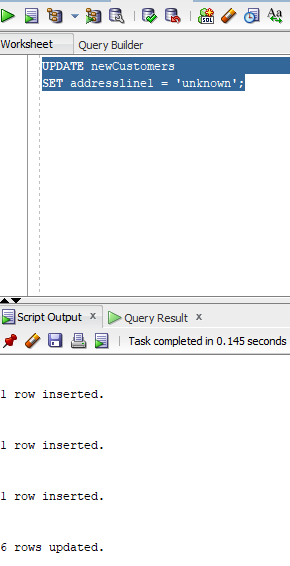
**SELECT COUNT(\*) FROM newCustomers;**



1. Write an update statement to update the value of column addressLine1 to ‘unknown’ for all the customers in the newCustomers table.

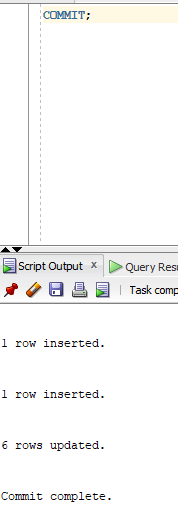
**UPDATE newCustomers**

**SET addressline1 = 'unknown';**



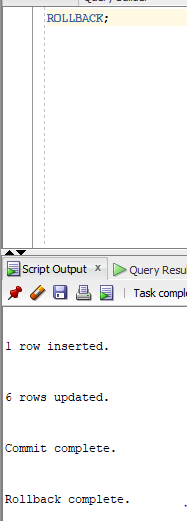
1. Make your changes permanent.

**COMMIT;**



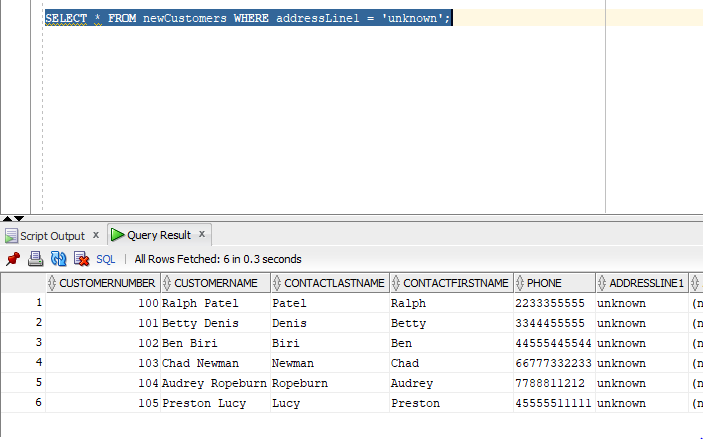
1. Execute the rollback command.

**ROLLBACK;**

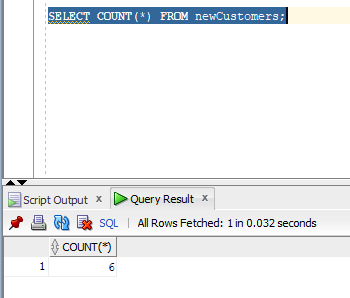


* 1. Display all customers from the newCustomers table whose address is ‘unknown’. How many rows are still updated?

**SELECT \* FROM newCustomers WHERE addressLine1 = 'unknown';**



**SELECT COUNT(\*) FROM newCustomers;**



* 1. Was the rollback command effective?

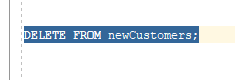
**Yes, the ROLLBACK command was effective. It undid the changes made to the "newCustomers" table during the transaction, which included setting the value of the "addressLine1" column to 'unknown' for all rows.**

* 1. What was the difference between the result of the rollback execution from Question 6 and the result of the rollback execution of this task?

**The result of the rollback execution from this task is different from the result of the rollback execution in Question 6 because in this task, we updated all the rows in the "newCustomers" table before rolling back the changes. As a result, after rolling back the changes, the "newCustomers" table should still contain all the rows that were inserted, whereas in Question 6, the "newCustomers" table would be empty after the rollback since no records were inserted before the transaction was started.**

1. Begin a new transaction and then create a statement to delete the customers from the newCustomers table

**DELETE FROM newCustomers;**

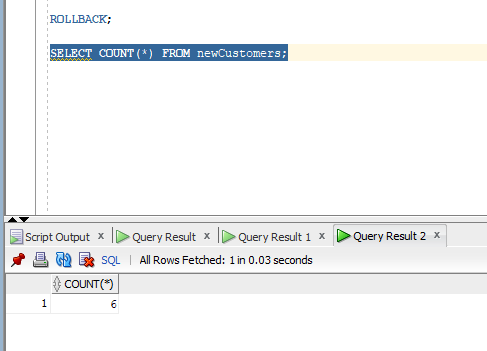




1. Perform a rollback to undo the deletion of the customers
   1. How many customers are now in the newCustomers table?

**ROLLBACK;**

**SELECT COUNT(\*) FROM newCustomers;**



* 1. Was the rollback effective and why?

**Yes, the rollback was effective in undoing the deletion of the customers. The ROLLBACK command undid all the changes made to the "newCustomers" table during the transaction, which included the deletion of all rows. After the rollback, the "newCustomers" table should contain the same rows that were present in the table before the transaction was started.**

1. Begin a new transaction and rerun the data insertion from Question 2 (copy the code down to Question 11 and run it)

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES (100, 'Patel', 'Ralph', '2233355555', '10 SenecaWay', 'Paris', 'France', 'Ralph Patel');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(101, 'Denis', 'Betty', '3344455555', '110 SenecaWay', 'Chicago', 'USA', 'Betty Denis');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(102, 'Biri', 'Ben', '44555445544', '13000 SenecaWay', 'Toronto', 'Canada', 'Ben Biri');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

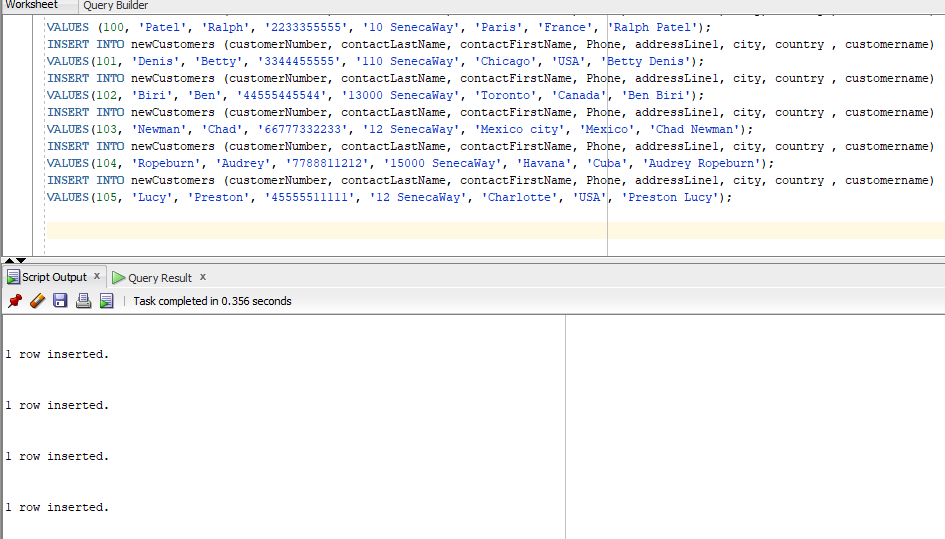
**VALUES(103, 'Newman', 'Chad', '66777332233', '12 SenecaWay', 'Mexico city', 'Mexico', 'Chad Newman');**

**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

**VALUES(104, 'Ropeburn', 'Audrey', '7788811212', '15000 SenecaWay', 'Havana', 'Cuba', 'Audrey Ropeburn');**

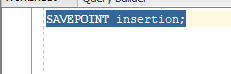
**INSERT INTO newCustomers (customerNumber, contactLastName, contactFirstName, Phone, addressLine1, city, country , customername)**

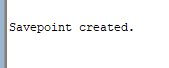
**VALUES(105, 'Lucy', 'Preston', '45555511111', '12 SenecaWay', 'Charlotte', 'USA', 'Preston Lucy');**



1. Set a Savepoint, called ***insertion***, after inserting the data

**SAVEPOINT insertion;**



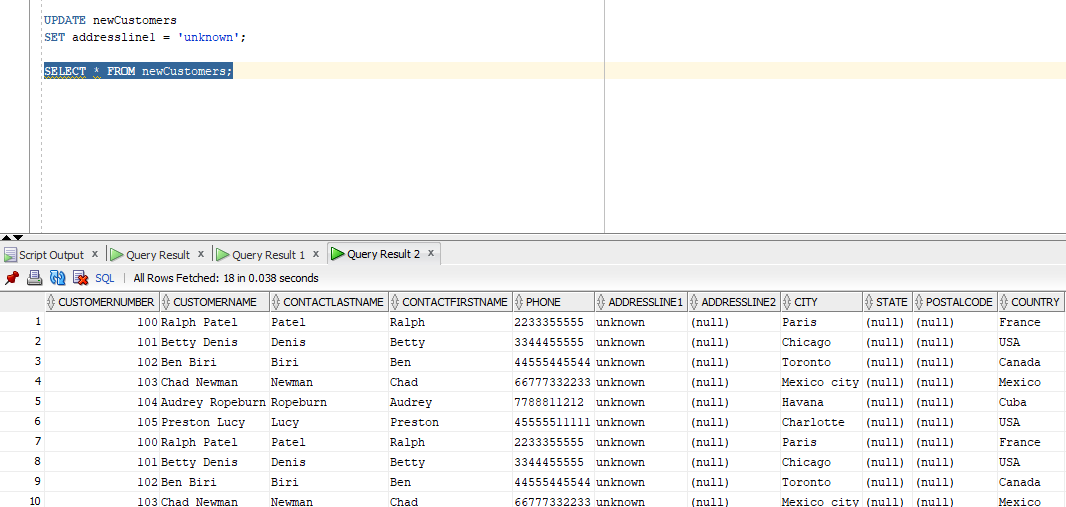


1. Rerun the update statement from Question 6 and run a query to view the data (copy the code down and run it again)

**UPDATE newCustomers**

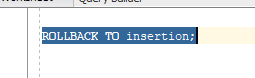
**SET addressline1 = 'unknown';**

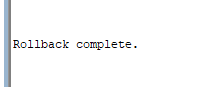
**SELECT \* FROM newCustomers;**



1. Rollback the transaction to the Savepoint created in Question 12 above and run a query to view the data.  
   What does the data look like (i.e. describe what happened?

**ROLLBACK TO insertion;**





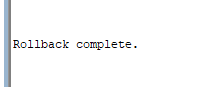
**The result of the query should show the original values of the "addressLine1" column for all customers in the "newCustomers" table. The rollback command effectively undid the update statement from Question 6, and the data in the table has been restored to its original state at the time the "insertion" savepoint was set.**

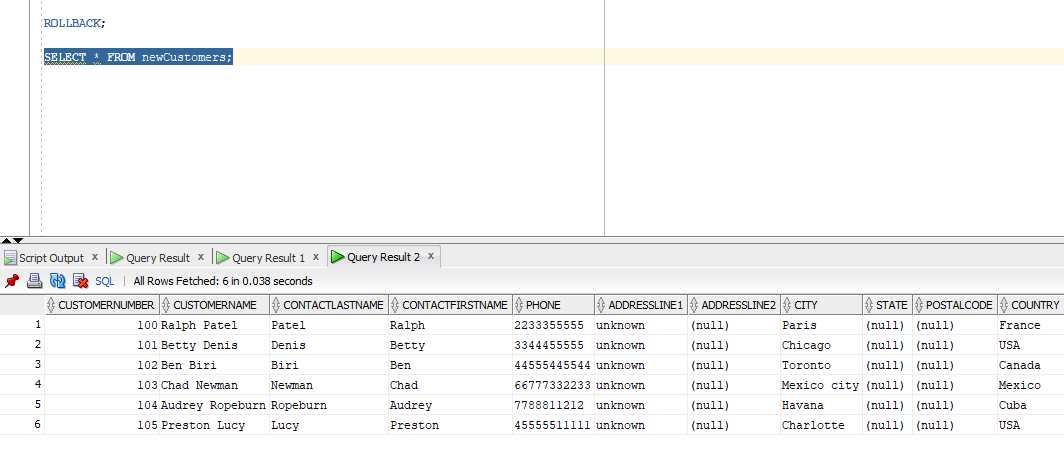
1. Use the rollback statement and again view the data. Describe what the results look like and what happened.

**ROLLBACK;**

**SELECT \* FROM newCustomers;**







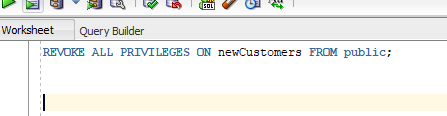
**The result should show the original state of the table before any changes were made during the transaction. This means that all data that was inserted and any updates that were made after the beginning of the transaction will be completely undone, and the "newCustomers" table will be restored to its original state.**

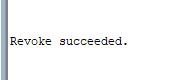
**In summary, using the ROLLBACK statement will effectively undo all changes made during the transaction, and restore the state of the database to its original state before the transaction began.**

## Part B - Permissions

1. Write a statement that denies all access to the newCustomers table for all public users

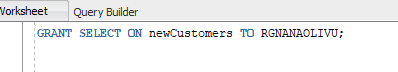
**REVOKE ALL PRIVILEGES ON newCustomers FROM public;**

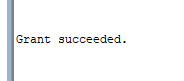




1. Write a statement that allows a person ‘RGNANAOLIVU’ read only access to the newCustomers table.

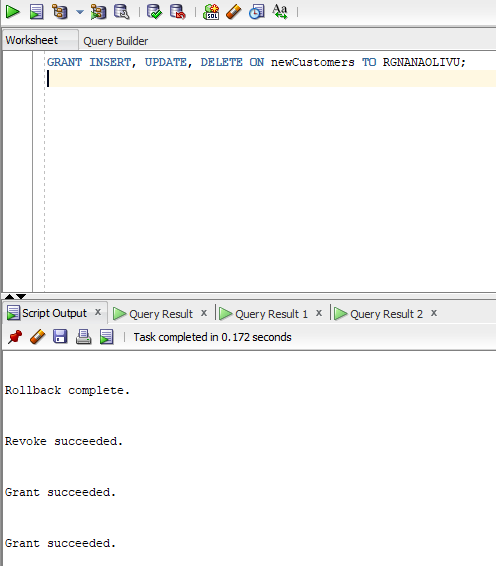
**GRANT SELECT ON newCustomers TO RGNANAOLIVU;**





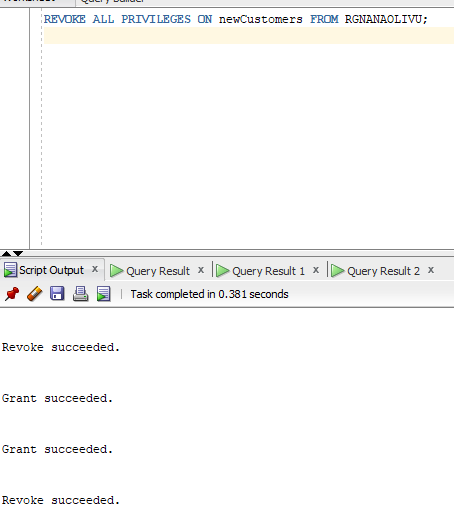
1. Write a statement that allows the same person ‘RGNANAOLIVU’ to modify (insert, update and delete) the data of the newCustomers table.

**GRANT INSERT, UPDATE, DELETE ON newCustomers TO RGNANAOLIVU;**



1. Write a statement the denies all access to the newCustomers table for the same person ‘RGNANAOLIVU’.

**REVOKE ALL PRIVILEGES ON newCustomers FROM RGNANAOLIVU;**

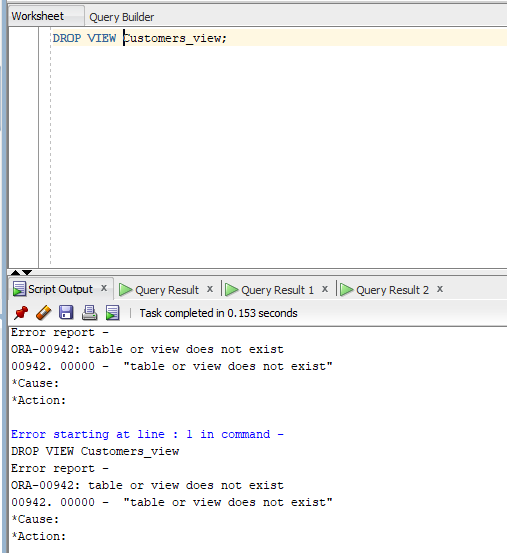


## Part C – Clean up

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

**DROP VIEW Customers\_view;**

**(No view exists so nothing dropped error is displayed)**



**DROP TABLE newCustomers;**

