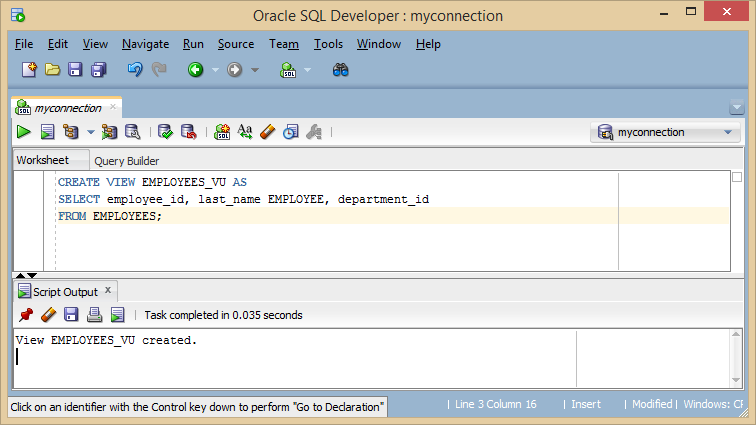
Part 1

1) The staff in the HR department wants to hide some of the data in the EMPLOYEES table. Create a view called EMPLOYEES\_VU based on the employee numbers, employee last names, and department numbers from the EMPLOYEES table. The heading for the employee name should be EMPLOYEE.

**CREATE VIEW EMPLOYEES\_VU AS**

**SELECT employee\_id, last\_name EMPLOYEE, department\_id**

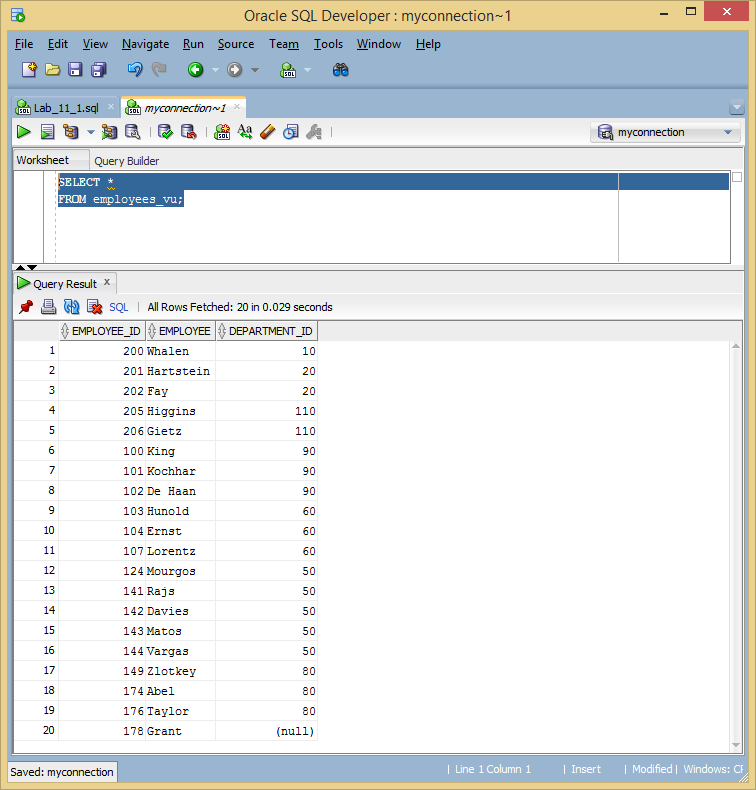
**FROM EMPLOYEES;**



2) Confirm that the view works. Display the contents of the EMPLOYEES\_VU view.

**SELECT \***

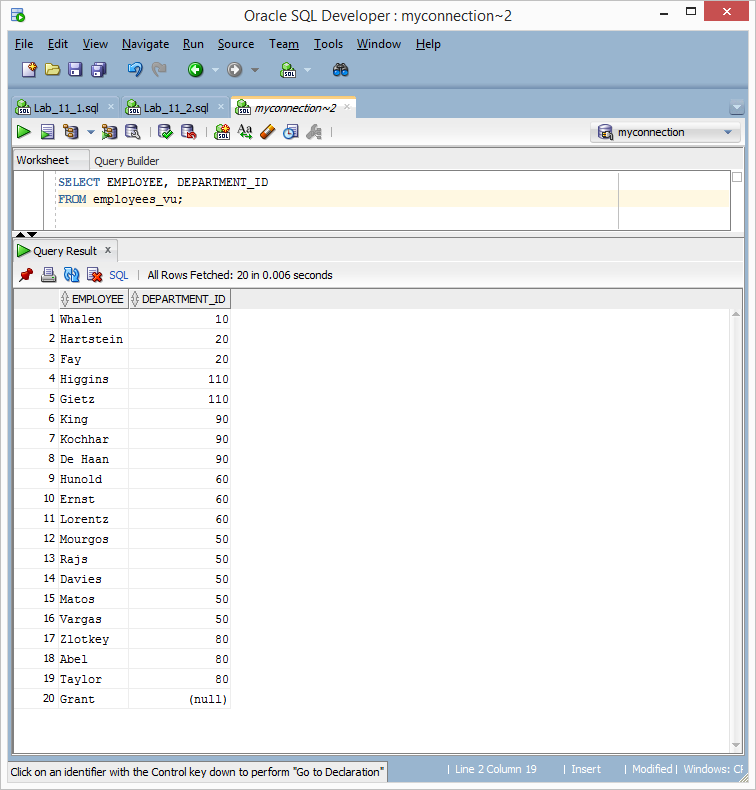
**FROM employees\_vu;**



3) Using your EMPLOYEES\_VU view, write a query for the HR department to display all employee names and department numbers.

**SELECT EMPLOYEE, DEPARTMENT\_ID**

**FROM employees\_vu;**



4) Department 50 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. You have been asked to label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.

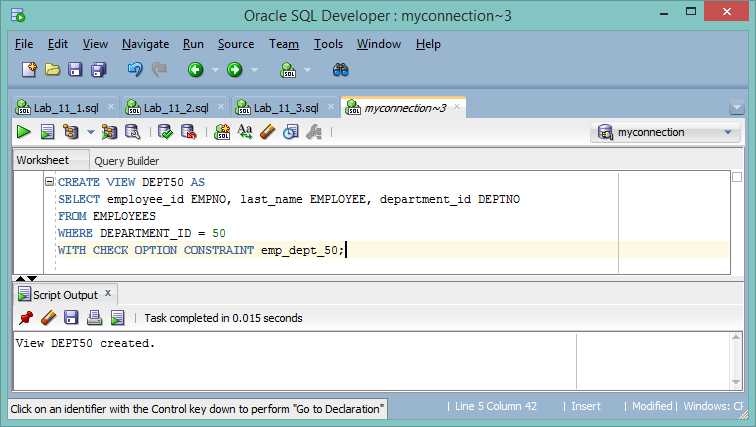
**CREATE VIEW DEPT50 AS**

**SELECT employee\_id EMPNO, last\_name EMPLOYEE, department\_id DEPTNO**

**FROM EMPLOYEES**

**WHERE DEPARTMENT\_ID = 50**

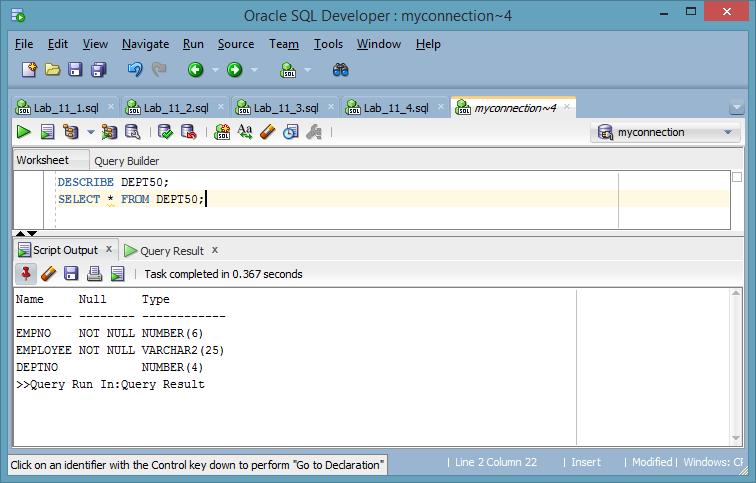
**WITH CHECK OPTION CONSTRAINT emp\_dept\_50;**

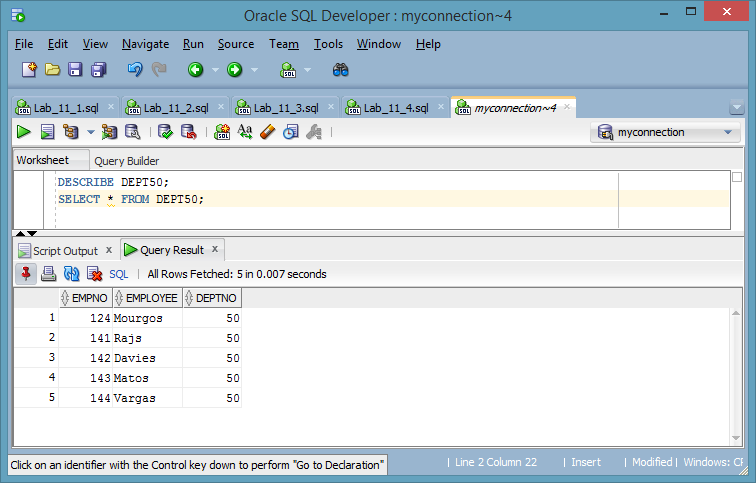


5) Display the structure and contents of the DEPT50 view.

**DESCRIBE DEPT50;**

**SELECT \* FROM DEPT50;**



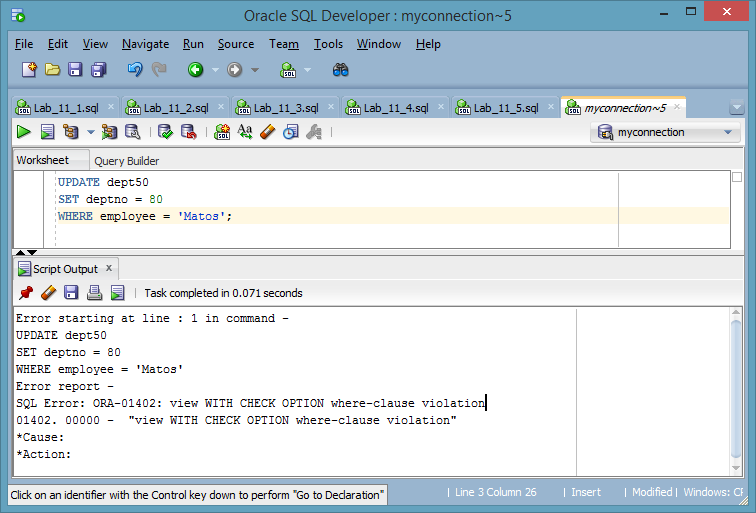


6) Test your view. Attempt to reassign Matos to department 80.

**UPDATE dept50**

**SET deptno = 80**

**WHERE employee = 'Matos';**



Part 2

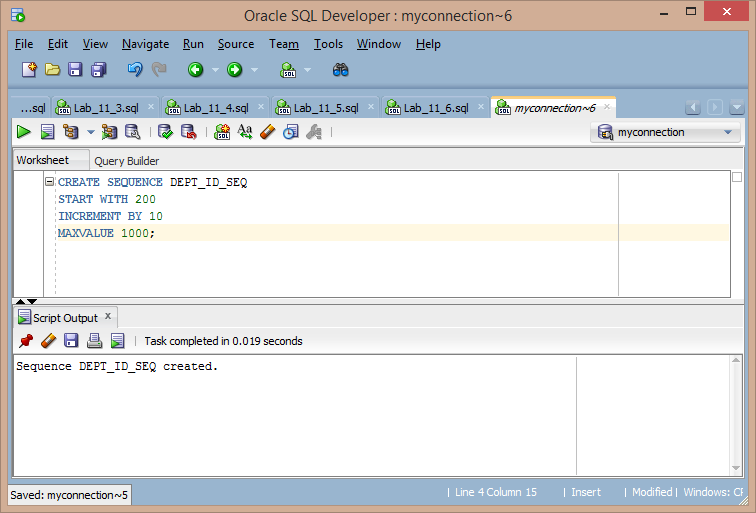
7) You need a sequence that can be used with the PRIMARY KEY column of the DEPT table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence DEPT\_ID\_SEQ.

**CREATE SEQUENCE DEPT\_ID\_SEQ**

**START WITH 200**

**INCREMENT BY 10**

**MAXVALUE 1000;**



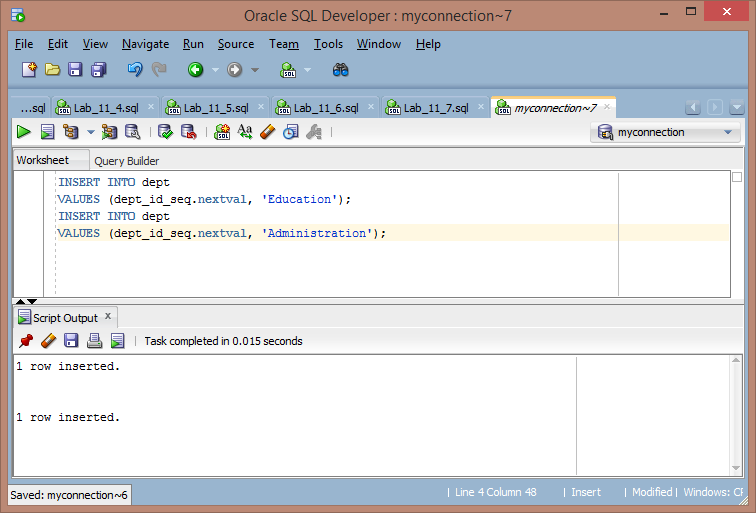
8) To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab\_11\_08.sql. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.

**INSERT INTO dept**

**VALUES (dept\_id\_seq.nextval, 'Education');**

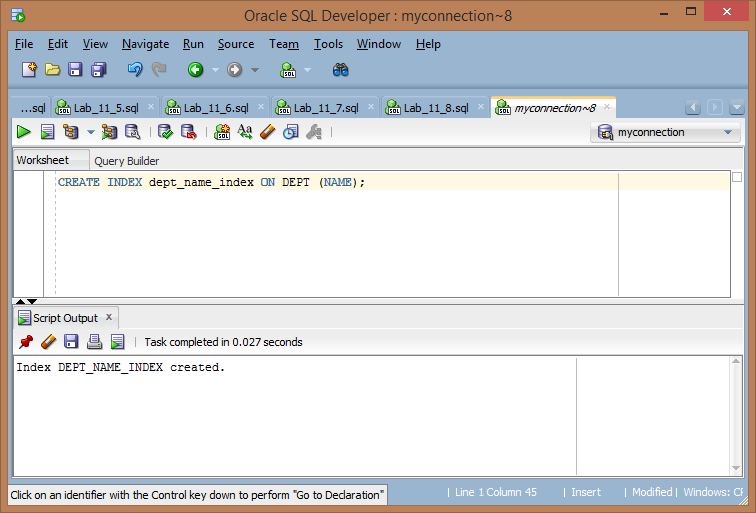
**INSERT INTO dept**

**VALUES (dept\_id\_seq.nextval, 'Administration');**

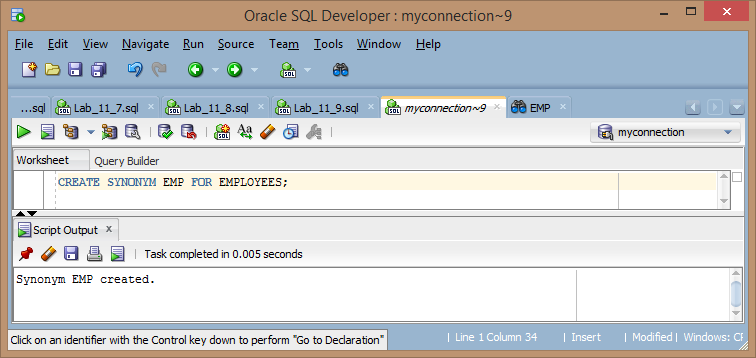


9) Create a nonunique index on the NAME column in the DEPT table.

**CREATE INDEX dept\_name\_index ON DEPT (NAME);**



10) Create a synonym for your EMPLOYEES table. Call it EMP.

**CREATE SYNONYM EMP FOR EMPLOYEES;** 

9) In this exercise, you will populate the Cookbook application tables you created in the prior assignment. Using the new tables:

* ~~Create a new sequence with a minimum value and non-maximum value~~

**CREATE SEQUENCE InventoryID**

**START WITH 1001**

**INCREMENT BY 1**

**MAXVALUE 9999;**

**CREATE SEQUENCE RecipeID**

**START WITH 1001**

**INCREMENT BY 1**

**MAXVALUE 9999;**

* ~~Create a nonunique index on two columns in your new tables~~

**CREATE INDEX Inv\_name\_index ON KITCHENINVENTORY (Inv\_name);**

**CREATE INDEX Rec\_name\_index ON COOKINGRECIPES (Rec\_name);**

* ~~Populate each of your tables with five (5) rows of data using proper insert statements~~

**INSERT INTO KITCHENINVENTORY (InventoryID, Inv\_name, Inv\_type, Inv\_quantity)**

**VALUES (InventoryID.nextval, 'Chease', 'dairy', 10);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Bread', 'starch', 20);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Butter', 'dairy', 14);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Milk', 'dairy', 0);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Tomato Soup Can', 'dairy', 1);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Can of Tuna', 'meat', 15);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Mayo', 'dairy', 7);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Ramen Noodle Pack', 'processed', 9999);**

**INSERT INTO KITCHENINVENTORY VALUES (InventoryID.nextval, 'Noodles', 'processed', 9);**

**INSERT INTO COOKINGRECIPES(RecipeID, Rec\_name, Rec\_steps)**

**VALUES (RecipeID.nextval, 'Grilled Chease',**

**'Butter bread on one side. Grill bread butter side down. Add chease to bread. Serve.');**

**INSERT INTO COOKINGRECIPES VALUES (RecipeID.nextval, 'Tomato Soup',**

**'Mix equal parts water and milk with one can of soup. Heat for 10 minutes. Serve');**

**INSERT INTO COOKINGRECIPES VALUES (RecipeID.nextval, 'Tuna Sandwitch',**

**'Mix tuna and mayo. Spread on bread. Sever');**

**INSERT INTO COOKINGRECIPES VALUES (RecipeID.nextval, 'Ramen Noodles',**

**'Put noodles in pan with 2 cups water. Heat till boil for 5 minutes. server with included seasoning');**

**INSERT INTO COOKINGRECIPES VALUES (RecipeID.nextval, 'Mac n Chease',**

**'Mix chease, milk and water. Heat till desired consitancy, add noodles. Sever');**

**INSERT INTO INGREDIENTS(RecipeID\_fk, InventoryID\_fk, Ing\_quantity)**

**VALUES(1001, 1001, 1);**

**INSERT INTO INGREDIENTS VALUES(1001, 1002, 2);**

**INSERT INTO INGREDIENTS VALUES(1001, 1003, 2);**

**INSERT INTO INGREDIENTS VALUES(1002, 1004, 1);**

**INSERT INTO INGREDIENTS VALUES(1002, 1002, 2);**

**INSERT INTO INGREDIENTS VALUES(1002, 1003, 2);**

**INSERT INTO INGREDIENTS VALUES(1003, 1002, 2);**

**INSERT INTO INGREDIENTS VALUES(1003, 1006, 2);**

**INSERT INTO INGREDIENTS VALUES(1003, 1007, 1);**

**INSERT INTO INGREDIENTS VALUES(1004, 1008, 1);**

**INSERT INTO INGREDIENTS VALUES(1005, 1001, 2);**

**INSERT INTO INGREDIENTS VALUES(1005, 1004, 1);**

**INSERT INTO INGREDIENTS VALUES(1005, 1009, 2);**

* Modify your sequence by setting a user defined cache option, justify your cache selection briefly in one or two sentences

**What does this mean?**

* ~~Write two or more final joins that display the primary data from the new tables~~

**SELECT count(i.INVENTORYID\_FK) "number of times used", i.ING\_QUANTITY,**

**count(i.INVENTORYID\_FK) \* i.ING\_QUANTITY "total needed", k.INV\_NAME , k.INV\_QUANTITY**

**FROM KITCHENINVENTORY k**

**JOIN INGREDIENTS i on k.INVENTORYID = i.INVENTORYID\_FK**

**GROUP BY k.INV\_NAME, i.ING\_QUANTITY, k.INV\_QUANTITY;**

**SELECT count(i.RECIPEID\_FK) "nuber of ingredients", c.REC\_NAME**

**FROM COOKINGRECIPES c**

**join INGREDIENTS i on c.RECIPEID = i.RECIPEID\_FK**

**GROUP BY c.RECIPEID, c.REC\_NAME;**

Capture screenshots of the SQL statements upon running them and run at least two final queries to show the kitchen inventory and recipes and how they are related. For example, appropriate joins can be used to show all of the entities.

