Lab 05 - DDL

# Objective:

The purpose of this lab is to introduce you to the DDL set of statements in SQL. By writing SQL to create tables, constraints, and views, you will have the tools needed to implement database designs that you will create later in the course. By finishing this lab, the student will be able to:

* create, modify, and drop tables based on design specifications provided,
* inserting new data into tables, update data in tables, and delete data from tables while considering referential integrity,
* enforce constraints on tables to ensure data integrity and consistency,
* create a table using the structure and data from an existing table,
* import data into a table from other tables.

# Submission:

***Your submission will be a single WORD file with the solutions provided.***

Your submission needs to include a comment header block and be commented to include the question and the solutions. Make sure every SQL statement terminates with a semicolon.

# Tasks:

Add   
SET AUTOCOMMIT ON;   
under the comment header and execute it

Consider the following table specifications:

## Part A (DDL) :

1. Create all the following tables and their given constraints:

**LAB5\_MOVIES** (movieid:int, title:varchar(35), releaseYear:int, director:int,score:decimal(3,2))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column  Name | Column  DataType | PK | Not  Null | Unique | FK | Default  Value | Validation |
| movieid | Int | ✓ |  |  |  |  |  |
| title | varchar(35) |  | ✓ |  |  |  |  |
| releaseYear | Int |  | ✓ |  |  |  |  |
| director | Int |  | ✓ |  |  |  |  |
| score | decimal(3,2) |  |  |  |  |  | < 10 and > 3 |

**CREATE TABLE LAB5\_MOVIES(**

**MOVIEID INT NOT NULL PRIMARY KEY,**

**TITLE VARCHAR(35) NOT NULL,**

**RELEASEYEAR INT NOT NULL,**

**DIRECTOR INT NOT NULL,**

**SCORE DECIMAL(3,2),**

**CONSTRAINT SCORE CHECK(SCORE BETWEEN 10 AND 3)**

**);**

Graphical user interface, text, application

Description automatically generated



**To view the table:-**



Text

Description automatically generated

**LAB5\_ACTORS** (actorid:int, firstname:varchar(20), lastname:varchar(30))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column  Name | Column  DataType | PK | Not  Null | Unique | FK | Default  Value | Validation |
| actorid | Int | ✓ |  |  |  |  |  |
| firstName | varchar(20) |  | ✓ |  |  |  |  |
| lastName | Varchar(30) |  | ✓ |  |  |  |  |

**CREATE TABLE LAB5\_ACTORS(ACTORS\_ID NUMBER PRIMARY KEY,**

**FIRSTNAME VARCHAR(20) NOT NULL,**

**LASTNAME VARCHAR(30) NOT NULL);**

Graphical user interface, text

Description automatically generated with medium confidence



**To view the table;**



Text, table, letter

Description automatically generated

**LAB5\_CASTINGS** (movieid:int, actorid:int)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column  Name | Column  DataType | PK | Not  Null | Unique | FK | Default  Value | Validation |
| movieid | Int | ✓ |  |  | ✓  (movies) |  |  |
| actorid | int | ✓ |  |  | ✓  (actors) |  |  |

**CREATE TABLE LAB5\_CASTINGS (MOVIEID NUMBER, ACTORID NUMBER,**

**CONSTRAINT PK PRIMARY KEY(MOVIEID,ACTORID),**

**CONSTRAINT FK1 FOREIGN KEY (MOVIEID) REFERENCES LAB5\_MOVIES(MOVIE\_ID),**

**CONSTRAINT FK2 FOREIGN KEY (ACTORID) REFERENCES LAB5\_ACTORS(ACTORS\_ID)**

**);**

Text

Description automatically generated



**To view the table**



Text, table, letter

Description automatically generated

**LAB5\_DIRECTORS**(directorid:int, firstname:varchar(20), lastname:varchar(30))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column  Name | Column  DataType | PK | Not  Null | Unique | FK | Default  Value | Validation |
| directorid | Int | ✓ |  |  |  |  |  |
| firstname | varchar(20) |  | ✓ |  |  |  |  |
| lastname | varchar(30) |  | ✓ |  |  |  |  |

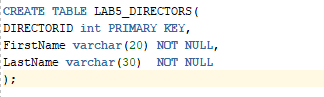
**CREATE TABLE LAB5\_DIRECTORS(**

**DIRECTORID int PRIMARY KEY,**

**FirstName varchar(20) NOT NULL,**

**LastName varchar(30) NOT NULL**

**);**





**To view the table**



Text, table, letter

Description automatically generated

1. Modify the ***movies*** table to create a foreign key constraint that refers to table ***directors***.

**ALTER TABLE LAB5\_MOVIES**

**ADD CONSTRAINT movies\_fk**

**FOREIGN KEY (DIRECTOR) REFERENCES LAB5\_DIRECTORS(DIRECTORID);**

A picture containing text

Description automatically generated



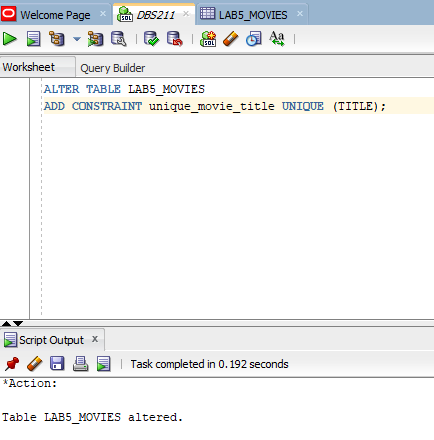
Graphical user interface, text

Description automatically generated

1. Modify the ***movies*** table to create a new constraint so the uniqueness of the movie title is guaranteed.   
   **ALTER TABLE LAB5\_MOVIES**

**ADD CONSTRAINT UNIQUE\_MOVIE**

**UNIQUE (TITLE);**



1. Write insert statements to add the following data to table ***directors*** and ***movies***.

**Director**

|  |  |  |
| --- | --- | --- |
| directorid | First name | Last name |
| 1010 | Rob | Minkoff |
| 1020 | Bill | Condon |
| 1050 | Josh | Cooley |
| 2010 | Brad | Bird |
| 3020 | Lake | Bell |

**INSERT INTO LAB5\_DIRECTORS (DIRECTORID, FIRSTNAME,LASTNAME)**

**VALUES (1010, 'Rob', 'Minkoff');**

**INSERT INTO LAB5\_DIRECTORS (DIRECTORID, FIRSTNAME,LASTNAME)**

**VALUES (1020, 'Bill', 'Condon');**

**INSERT INTO LAB5\_DIRECTORS (DIRECTORID, FIRSTNAME,LASTNAME)**

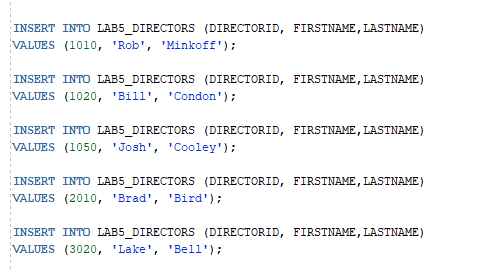
**VALUES (1050, 'Josh', 'Cooley');**

**INSERT INTO LAB5\_DIRECTORS (DIRECTORID, FIRSTNAME,LASTNAME)**

**VALUES (2010, 'Brad', 'Bird');**

**INSERT INTO LAB5\_DIRECTORS (DIRECTORID, FIRSTNAME,LASTNAME)**

**VALUES (3020, 'Lake', 'Bell');**

****

**Shape

Description automatically generated with medium confidence**

**Movies**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | title | year | director | score |
| 100 | The Lion King | 2019 | 3020 | 3.50 |
| 200 | Beauty and the Beast | 2017 | 1050 | 4.20 |
| 300 | Toy Story 4 | 2019 | 1020 | 4.50 |
| 400 | Mission Impossible | 2018 | 2010 | 5.00 |
| 500 | The Secret Life of Pets | 2016 | 1010 | 3.90 |

**INSERT INTO LAB5\_MOVIES (MOVIE\_ID, TITLE,REALEASEYEAR, DIRECTOR, SCORE)**

**VALUES (100, 'The Lion King', 2019,3020,3.50);**

**INSERT INTO LAB5\_MOVIES (MOVIE\_ID, TITLE,REALEASEYEAR, DIRECTOR, SCORE)**

**VALUES (200, 'Beauty and the Beast', 2017, 1050,4.20);**

**INSERT INTO LAB5\_MOVIES (MOVIE\_ID, TITLE,REALEASEYEAR, DIRECTOR, SCORE)**

**VALUES (300, 'Toy Story 4', 2019, 1020, 4.50);**

**INSERT INTO LAB5\_MOVIES (MOVIE\_ID, TITLE,REALEASEYEAR, DIRECTOR, SCORE)**

**VALUES (400, 'Mission Impossible', 2018,2010,5.00);**

**INSERT INTO LAB5\_MOVIES (MOVIE\_ID, TITLE,REALEASEYEAR, DIRECTOR, SCORE)**

**VALUES (500, 'The Secret Life of Pets', 2016,1010,3.90);**

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence

1. Write SQL statements to remove all above tables.   
   Is the order of tables important when removing? Why?

**DROP TABLE LAB5\_DIRECTORS CASCADE CONSTRAINTS;**

**DROP TABLE LAB5\_CASTINGS;**

**DROP TABLE LAB5\_ACTORS;**

**DROP TABLE LAB5\_MOVIES;**

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

A picture containing text

Description automatically generated

**Yes, the order of tables is important when removing them because of the foreign key constraints. If a table has a foreign key constraint that references another table, the referenced table must be removed first before the referencing table can be removed.**

**For example, if you try to remove the directors table first, and the movies table has a foreign key constraint that references the directors table, the removal of the directors table will fail because of the foreign key constraint. In this case, you would need to remove the foreign key constraint from the movies table first and then remove the directors table.**

Part B (More DML):

1. Create a new empty table (that means the table will not have any data after creating) ***employeecopy*** the same as table ***retailemployees.***  Use a single statement to create the table and insert the data at the same time (Hint use a WHERE clause that is false like 1=2)

**CREATE TABLE employeecopy AS**

**SELECT \***

**FROM retailemployees**

**WHERE 1 = 2;**

Text

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

Text

Description automatically generated

1. Modify table ***employeecopy*** and add a new column ***username*** to this table. The value of this column is not required and does not have to be unique.

**ALTER TABLE EMPLOYEECOPY**

**ADD USERNAME varchar(25);**

Text

Description automatically generated with medium confidence



Table

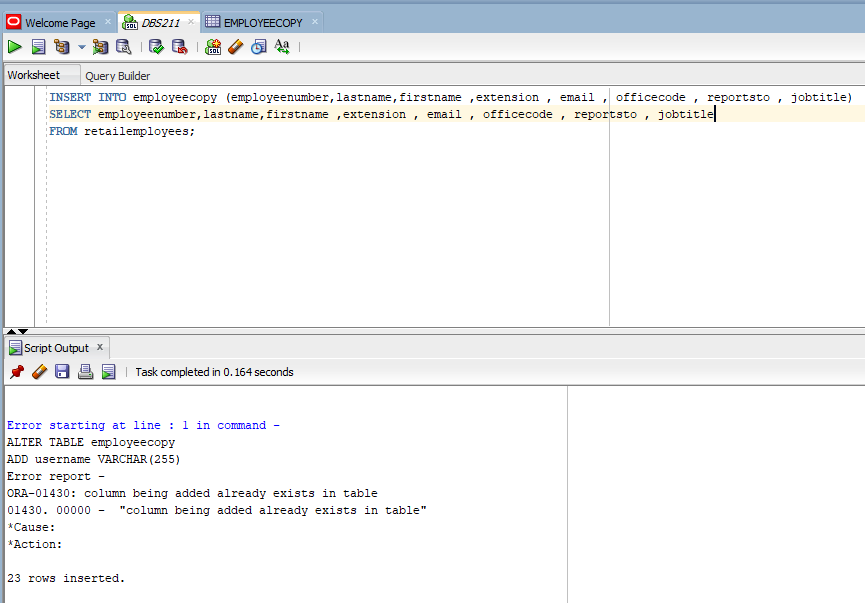
Description automatically generated

1. Re-insert all data from the ***retailemployees.***  table into your new table ***employeecopy*** using a single statement.

**INSERT INTO employeecopy(EMPLOYEENUMBER, LASTNAME, FIRSTNAME, EXTENSION, EMAIL,OFFICECODE,REPORTSTO,JOBTITLE)**

**SELECT EMPLOYEENUMBER, LASTNAME, FIRSTNAME, EXTENSION, EMAIL,OFFICECODE,REPORTSTO,JOBTITLE**

**FROM retailemployees;**



1. FROM retailemployees;In table ***employeecopy***, generate the email address for column ***username*** for each student by concatenating the employeeid and the string “@seneca.ca”. For instance, the username of employee 123 will be “123@seneca.ca’.

INSERT INTO employeecopy (username)

SELECT CONCAT(employeeid, '@seneca.ca') FROM employees;

INSERT INTO employeecopy (username)

SELECT CONCAT(employeeid, '@seneca.ca') FROM empl

**UPDATE employeecopy SET username = CONCAT (employeenumber, '@seneca.ca');**



Graphical user interface, application

Description automatically generated

**OR (2ND WAY)**

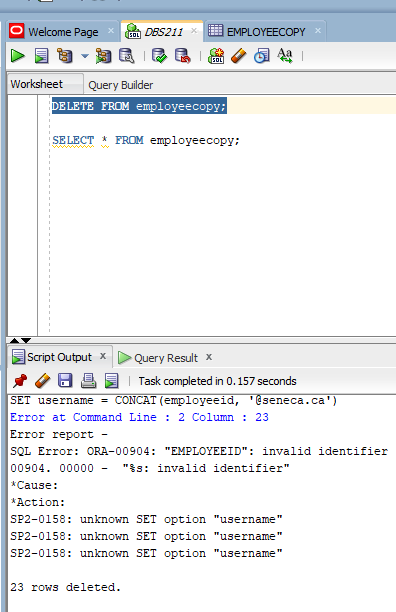
**INSERT INTO employeecopy (username)**

**SELECT CONCAT (employeeid, ‘seneca.ca’) FROM retailemployees;**

1. Delete all th e employeecopy data and display the data in the table. Does employeecopy exist? If not how can you delete table ***employeecopy***.

**DELETE FROM employeecopy;**

**SELECT \* FROM employeecopy;**



**If the employeecopy table no longer exists, you can delete it using the following statement:**

**DROP TABLE employeecopy;**

