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:

**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 |

**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) |  | ✓ |  |  |  |  |

**CASTINGS** (movieid:int, actorid:int) Composite key

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

**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) |  | ✓ |  |  |  |  |

1. Modify the ***movies*** table to create a foreign key constraint that refers to table ***directors***.
2. Modify the ***movies*** table to create a new constraint so the uniqueness of the movie title is guaranteed.
3. Write insert statements to add the following data to table ***directors*** and ***movies***. (week 3 class)

**Director**

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

**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 |

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

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)
2. 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.
3. Re-insert all data from the ***retailemployees.***  table into your new table ***employeecopy*** using a single statement.
4. 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’.
5. Delete all the employeecopy data and display the data in the table. Does employeecopy exist? If not how can you delete table ***employeecopy***.

From Lab 03-PART2– SQL – Single Table Queries will be implemented in this lab

1. Create a statement that will insert yourself as an RETAILEMPLOYEE of the company.

* Use a unique RETAILEMPLOYEE number of your choice
* Use your school email address and
* Your job title will be “Cashier”
* Office code will be 4
* You will report to RETAILEMPLOYEE 1088

1. Create a query that displays your, and only your, RETAILEMPLOYEE data
2. Create a statement to update just your job title to “Head Cashier”
3. Create a statement to insert another fictional RETAILEMPLOYEE into the database.  This RETAILEMPLOYEE will be a “Cashier” and will report to you.  Make up fake data for the other fields.
4. Create a statement to Delete yourself from the database.  Did it work?  If not, why?
5. Create a statement to delete the fake RETAILEMPLOYEE from the database and then rerun the statement to delete yourself.  Did it work?
6. Create a **single** statement that will insert both yourself and the fake RETAILEMPLOYEE at the same time.  This time the fake RETAILEMPLOYEE will report to 1088 as well.
7. Create a **single** statement to delete both yourself and the fake RETAILEMPLOYEE.
8. Create a new order in RETAILORDER table with required date Sep 22nd,2021 and order date as Sep 17th,2021. Make up the reset of the fields and then display the only the new order that you have created just now.
9. Insert a new product into product table with product name as “2020 Bugatti Veyron” and productcode as “S111\_111” and make up the rest of the fields.