# Tasks:

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

Consider the following table specifications:

## Part A (DDL):

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

**L6\_MOVIES** (movieid:int, title:varchar(35), year:int, director:int,score:decimal(3,2))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column  Name | Column  DataType | PK | Not  Null | Unique | FK | Default  Value | Validation |
| mid | Int | ✓ |  |  |  |  |  |
| title | varchar(35) |  | ✓ |  |  |  |  |
| releaseYear | Int |  | ✓ |  |  |  |  |
| director | Int |  | ✓ |  |  |  |  |
| score | decimal(3,2) |  |  |  |  |  | < 5 and > 0 |

**L6\_ACTORS** (actorid:int, name:varchar(20), lastname:varchar(30))

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

**L6\_CASTINGS** (movieid:int, actorid:int)

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

**L6\_DIRECTORS**(id:int, name: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***.

**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 ***employee2*** the same as table ***employees***. Use a single statement to create the table and insert the data at the same time.
2. Modify table ***employee2*** 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. Delete all the data in the ***employee2*** table
4. Re-insert all data from the ***employees*** table into your new table ***employee2*** using a single statement.
5. In table ***employee2***, write a SQL statement to change the first name and the last name of employee with ID ***1002*** to your name.
6. In table ***employee2***, generate the email address for column ***username*** for each student by concatenating the first character of employee’s first name and the employee’s last name. For instance, the username of employee Peter Stone will be ***pstone***. NOTE: the username is in all lower-case letters.
7. In table ***employee2***, remove all employees with office code 4.
8. Drop table ***employee2***.