**In-class Activity #2.2: Working with SQL, Part 1**

**Getting Information out of a Database**

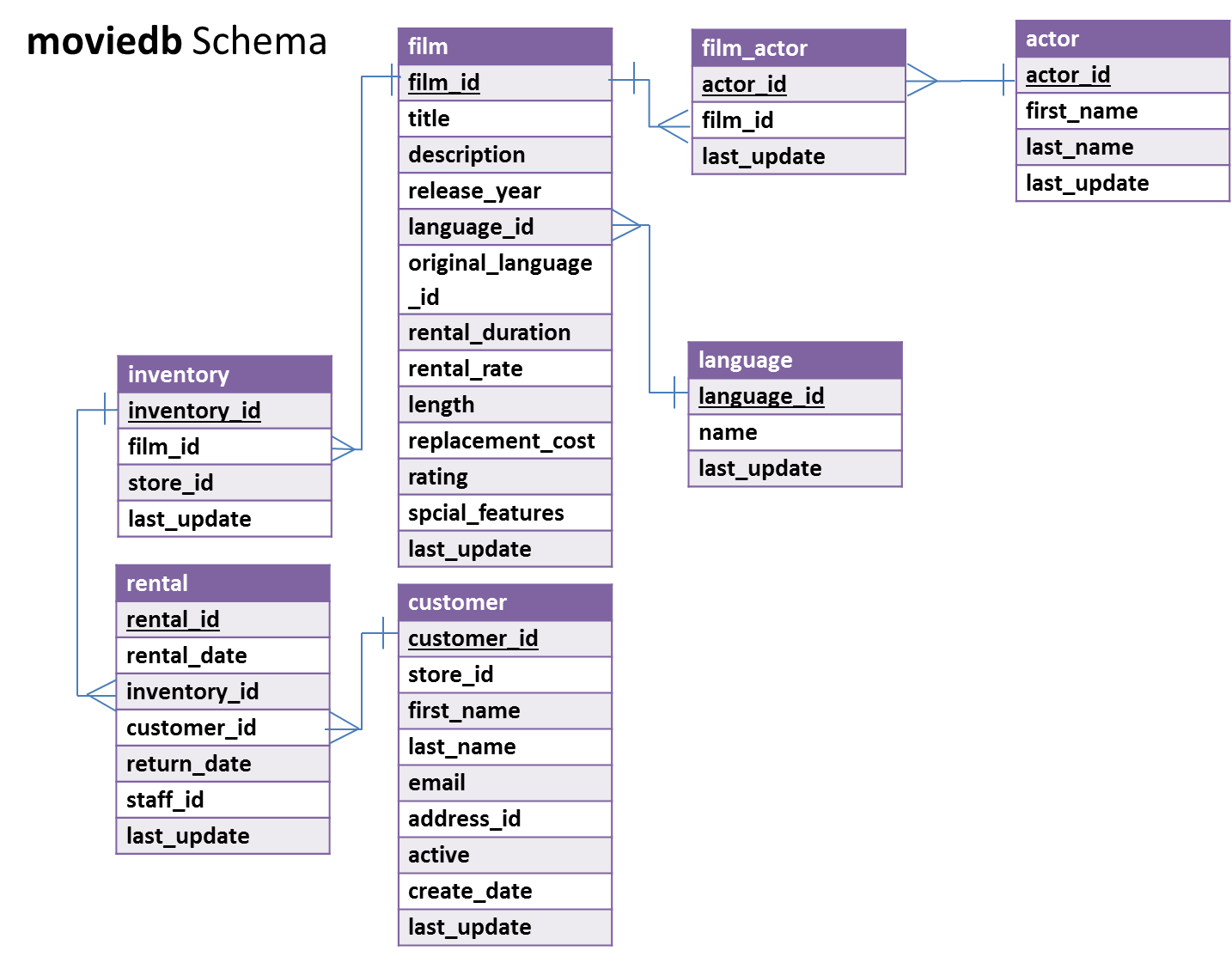
|  |
| --- |
| **Submission Instructions**  Submit your solutions for Part 2 of this exercise as a word or pdf file through **Canvas>Assignments>In-Class Activities.** Please submit your solution **within two hours after class**. |

**The moviedb schema**

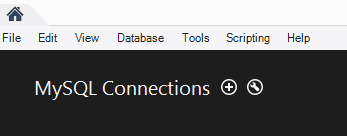
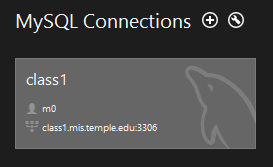
To do this exercise, you will be working with a movie rental database. The schema is simply called “**moviedb**” (not m0moviedb, m1moviedb, etc.). Spend some time looking at the schema carefully.

For example,

* In the **film** table, you can see that a film has a title, description, rating, and length (among other things).
* In the **actor** table, you can also see that an actor has a first name and a last name.
* The **film\_actor** table implements the many-to-many relationship between actor and film (i.e., a film can have more than one actor, and an actor can be in more than one film).
* There are other tables as well, and the table names/field names are quite self-explanatory.



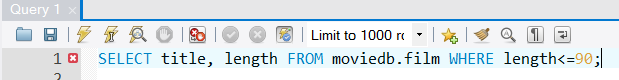
**Setting up a connection in SQL Workbench**

1. Start SQL Workbench.
2. **If you have followed the Quick Guide to MySQL Workbench to create a new connection, skip to step 4.** Otherwise, follow the instruction below to create a new connection.
3. Click on the “plus sign” next to MySQL Connections to create a new connection.  
   
4. At the “Setup New Connection” dialog, fill in the information as follows:  
     
   Connection Name: mis2502  
   Hostname: dataanalytics.temple.edu  
   Username: Your username is available on Canvas under Grades)  
     
   (If it says that “mis2502” is already taken, make the connection name something else. Any name is ok or this.)
5. Click “OK” and you’ll see the connection under “MySQL Connections”:  
     
   
6. In MySQL Workbench, open the connection to the class1 server using your username and password. Click on the “**moviedb**” schema and then the tables tab to see the list of tables.

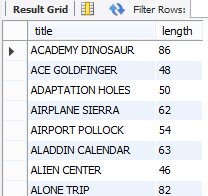
**Part 1: Try out some simple queries**

1. **A simple statement**

In the Query 1 pane, type the following (make sure you do not have any typo). And then click the Execute SQL Script button (the lightning bolt):



In the output tab, you’ll see this:



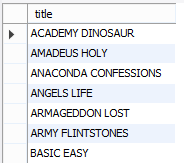
1. **A “join” statement**

Now try a **join**. Enter and run the following query:

SELECT DISTINCT film.title

FROM moviedb.actor, moviedb.film, moviedb.film\_actor  
WHERE actor.actor\_id = film\_actor.actor\_id   
AND film.film\_id = film\_actor.film\_id  
AND actor.first\_name = 'Penelope';

You’ll get the list of films starring actors with the first name of “Penelope”:



Make sure you understand what’s going on here:

|  |  |
| --- | --- |
| SELECT DISTINCT film.title | SELECTing unique film titles (no duplicates) |
| FROM moviedb.actor, moviedb.film, moviedb.film\_actor | In order to find out what films starred which actors, we need to associate actor and film. We do that by joining actor, film, and the tables in-between (in this case, that’s film\_actor). If this isn’t clear to you, check the schema on page 1 of this document. |
| WHERE actor.actor\_id = film\_actor.actor\_id AND film.film\_id = film\_actor.film\_id | We look for primary/foreign key fields in the connected tables and set them equal to each other in the WHERE clause. In this database, primary and foreign key fields will have the same names – this makes them easier to find. |
| AND actor.first\_name = 'Penelope'; | Once we join the tables, we will have all actors in all movies. We just ones the ones named “Penelope,” so we apply this last restriction. |

1. **A “subselect” statement**

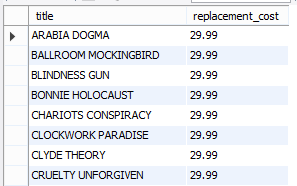
Finally, let’s do a query with a **subselect**:

SELECT title, replacement\_cost

FROM moviedb.film  
WHERE replacement\_cost = (SELECT MAX(replacement\_cost)

FROM moviedb.film);

This will return movies with the highest replacement cost (there are 53 of them), here’s the first few:



Here’s what’s going on:

|  |  |
| --- | --- |
| SELECT title, replacement\_cost  FROM moviedb.film | SELECTing the title and replacement cost of the movies from our film table |
| WHERE replacement\_cost = | Filtering based on replacement cost. We only want the movies where the cost is equal to its highest value |
| (SELECT MAX(replacement\_cost)  FROM moviedb.film) | We get the highest value with another SELECT statement that gets the MAXimum replacement cost from the film table |

**Part 2: Write queries on your own (Submit your solutions for Part 2 only to Canvas)**

Create the SQL SELECT query to answer each of the questions below. Some of the questions can be answered by querying one table; others will require joining multiple tables to get the answer.

|  |
| --- |
| For each question, you’ll need to write down:  **(1)** **the SQL query** (which you can copy and paste it from SQL Workbench), and  **(2) the answer you get from MySQL Workbench as a result of the query** (which you can copy and paste from the results). |

1. (a) What is the title and length of the longest movie in the database (assume there’s only one with the longest value and use LIMIT)?

*Display: title and length*

(b) Now assume there could be more than one movie with the longest value and use a *subselect* with the MAX() function to find the movies with the greatest length.

*Display: title and length*

1. Are R movies, on average, longer than PG movies? Prove it!

*Display: rating and average length*

(HINT: Use GROUP BY)

1. How many movies are in French?

*Display: number of movies*

HINT #1: Surround the table name `language` with back quotes.

HINT #2: The film table only contains language\_id. The name field in the language table contains the film language names (such as French, Spanish, etc.).

1. How many customers have a last name that starts with ‘A’?

*Display: number of customers*

**Hint #1:** You can use LIKE with a WHERE clause, like this:  
 SELECT *column\_name* FROM schema\_name.*table\_name* WHERE *column\_name* LIKE *value*  
 LIKE looks for close matches, not exact ones (as with the = operator)

**Hint #2:** You can use % as a wildcard value, which represents zero, one, or multiple characters.   
 LIKE '%ap%' will match any value containing “ap”; LIKE '%ap' will any value ending in “ap”

LIKE 'ap%' will match any value beginning in “ap”. For example, LIKE ‘ap%’ will match with “apple,” “application,” “apex,” etc.

1. In which films did Sandra Peck star (only return the first five)

*Display: title*

HINT: The film table contains film titles, but the actor names can only be found in the actor table. Hence, you need to find a way to join these tables.

1. How many movies has the customer Adam Gooch rented?

*Display: number of movies*

HINT: You need to use the customer table and the rental table, and join the two tables.