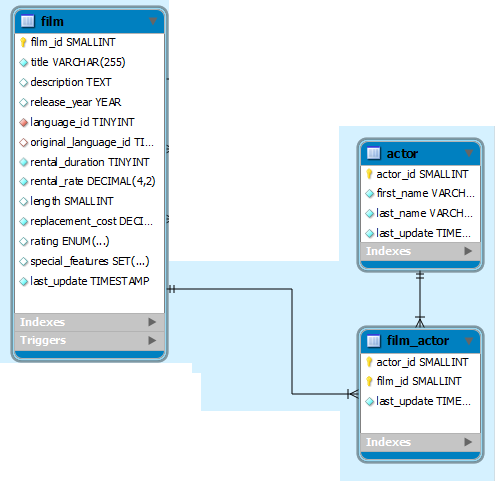
**In-Class Exercise: SQL #1  
Getting Information out of a Database**

To do this exercise, you will be working with a movie rental database. The schema for this database is provided on the accompanying document. All of the tables are in a schema called simply “moviedb” (not m0moviedb, m1moviedb, etc.). You can’t write to any of the tables – you can only use SELECT statements to read from them (so don’t worry about causing any damage).

Spend some time looking at the schema carefully. The field names are pretty self-explanatory. For example, here are three tables:

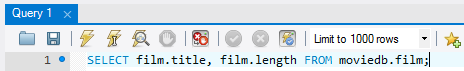


You can see that a film has a title, description, rating, and length (among other things). You can also see that an actor has a first name and a last name. The film\_actor table implements the many-to-many relationship between actor and film (i.e., a film can have more than one actor, and an actor can be in more than one film). You’ll also notice that data types are listed for each field, but they should be pretty obvious – for example, first\_name is a VARCHAR because it is a string value.

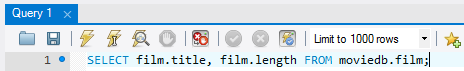
In MySQL Workbench, open the connection to the class1 server using your username and password. Click on the “moviedb” schema and then the tables tab to see the list of tables.

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

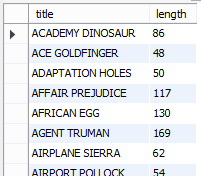
In the Query 1 pane, type the following:



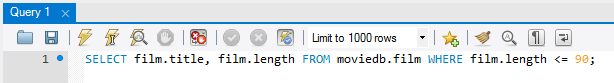
And then click the Execute SQL Script button (the lightning bolt):



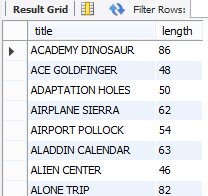
And you’ll see a list of all movie titles and their lengths (this is just the first few):



Now let’s get more sophisticated. Change the statement to read:



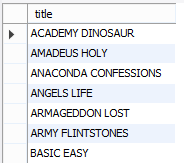
Execute the query and in the output tab you’ll see this:



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**

We’re 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.

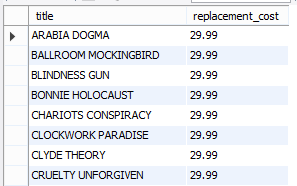
**AND actor.first\_name = 'Penelope';**

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.

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

**SELECT film.title,film.replacement\_cost FROM moviedb.film  
WHERE film.replacement\_cost = (SELECT MAX(film.replacement\_cost) FROM moviedb.film);**

This will return movies with the highest replacement cost (there are 21 of them), here’s the first few:



Here’s what’s going on:

**SELECT film.title,film.replacement\_cost FROM moviedb.film**

We’re SELECTing the title and replacement cost of the movies from our film table

**WHERE film.replacement\_cost =**

We’re filtering based on replacement cost. We only want the movies where the cost is equal to its highest value

**(SELECT MAX(film.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…**

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 write down the SQL query (which you can copy and paste it from SQL Workbench) and the answer you get as a result of the query (which you can copy and paste from the results).

1. What are the full names of actors with the first name of “Nick”?

1. What is the length and description of the film “Grit Clockwork”?
2. How many G movies are in the database?

1. 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)?

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.

.

1. Are R movies, on average, longer than PG movies? Prove it! ***(HINT: Use GROUP BY)***
2. How many movies are in English?  
   (HINT: The name field in the language table contains the film language names.)
3. In which films did Bob Fawcett star (only return the first five)
4. How many movies has the customer Melissa King rented?

1. How many customers have a first name that starts with ‘W’?

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

**Hint #2:** You can use % as a wildcard value.   
 So LIKE ‘ap%’ will match with “apple,” “application,” “apex,” etc.