

MYSQL DATABASE ADMINISTRATION

★ ★ ★ ★ ★ With Top-Rated SQL Instructor John Pauler

The background shows a screenshot of MySQL Workbench. On the left, there's a 'Customer Data' schema diagram with tables like country, city, address, and customer. In the center, there's an 'Inventory' schema diagram with tables like film, film_category, category, language, actor, and film_actor. A query editor at the bottom shows a SELECT statement:

```
1 | SELECT
2 |     customer.first_name,
3 |     customer.last_name
4 | FROM
5 |     rental
6 |     LEFT JOIN customer
7 |         ON customer.customer_id = rental.customer_id
8 | GROUP BY
9 |     customer.first_name,
10 |     customer.last_name
11 | ORDER BY
12 |     COUNT(DISTINCT rental_id) DESC;
```

On the right, there's a 'Action' sidebar with various database objects and their details.



COURSE STRUCTURE



This is a **project-based course**, for students looking for a *practical, hands-on, and highly engaging* approach to learning MySQL database administration

Additional resources include:

- ★ **Downloadable Ebook** to serve as a helpful reference when you're offline or on the go
- ★ **Quizzes & Homework Exercises** to test and reinforce key concepts, with step-by-step solutions
- ★ **Bonus Projects** to test your abilities and apply the skills developed throughout the course

COURSE OUTLINE

1

Introduction & Setup

Discuss relational databases, download MySQL Community Server and Workbench, and create the project databases

2

Creating Schemas & Tables

Create and delete schemas, tables, and columns using the Workbench Editor, and by running SQL code

3

Inserting, Updating & Deleting

Insert data into tables, and update or delete specific records based on logical criteria you specify

4

Database Design

Learn about table relationships, primary and foreign keys, and how to properly normalize a database

[MID-COURSE PROJECT]

5

Advanced DBA Concepts

Create stored procedures, triggers, and indexes, and learn how to implement unique and non-NULL constraints

6

Managing Users & Permissions

Grant access to additional users and prescribe what they can and cannot do within the database

[FINAL PROJECT]

INTRODUCING THE COURSE PROJECT

THE SITUATION

You have just been hired by a small database consulting shop. You will be working on projects for various clients to help them get their databases built and optimized for business.

THE BRIEF

As a member of the consulting team, you will be asked to perform database services for various clients.

Sometimes you will work with existing databases and tables to make small improvements, and other times you will need to set up a database for clients from the ground up.

THE OBJECTIVE

Use MySQL Workbench to:

- *Create databases and tables your clients will need to run their businesses*
- *Use your knowledge of database best practices to implement and optimize database design principles*
- *Update records, create indexes, triggers, and stored procedures, and manage users and permissions*

SETTING EXPECTATIONS

1

You'll learn MySQL Database Administration using MySQL Workbench

- *In your career, you may use other “flavors” of SQL (T-SQL, PL/SQL, PostgreSQL, etc.)*
- *Each flavor is very similar. The principles you learn here will apply universally, but syntax will vary slightly*

2

We will focus on the creation and maintenance of databases

- *We will design and create schemas and tables from scratch, and alter existing tables to improve them*

3

This course is for SQL Beginners and Data Analysts interested in DBA skills

- *This course starts with the basics; if you’re already an expert DBA, this course will not be appropriate for you*
- *If you’ve already mastered the basics, feel free to skip ahead to the parts of the course that feel relevant to you*

4

We will NOT go deep on data analysis (*that’s another course*)

- *This course will focus on beginner-level Database Administration concepts, like creating and maintaining databases*
- *We offer separate courses that focus on using SQL for data analysis and business intelligence (beginner + expert-level)*

Introduction to SQL

The screenshot displays the phpMyAdmin interface with the following sections:

- Left Sidebar:** Shows the database schema with tables like actor, address, category, city, country, customer, customer_list, film, film_actor, film_category, film_text, and inventory.
- Top Bar:** Shows "phpMyAdmin" and "Current server".
- Table Data Grid:** Shows rows from the "rental" table.
- Query Results:** Shows the results of a query to find the most frequent customer.
- Status Bar:** Shows the number of rows (599 total) and the time taken (0.0212 seconds).
- Bottom Status Bar:** Shows the MySQL connection status: "1,001 MyISAM, InnoDB, general, etc."

Table Data Grid (rental table):

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-28 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:30 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 22:54:39 | 1711 | 408 | 2005-05-28 22:12:30 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 22:54:48 | 2452 | 100 | 2005-05-28 22:12:30 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 22:54:57 | 2070 | 214 | 2005-05-28 22:12:30 | 2 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 22:54:57 | 2792 | 449 | 2005-05-28 22:12:30 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 22:54:57 | 367 | 367 | 2005-05-28 22:12:30 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 22:54:58 | 2446 | 128 | 2005-05-28 22:12:30 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

Query Results:

```
✓ Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)  
1 SELECT  
2     customer.first_name,  
3     customer.last_name,  
4     COUNT(DISTINCT rental_id) AS count  
5 FROM  
6     rental  
7     JOIN customer ON customer.customer_id = rental.customer_id  
8 GROUP BY customer.customer_id  
9 ORDER BY  
10    count DESC  
11 ORDER BY  
12    COUNT(DISTINCT rental_id) DESC
```

Bottom Status Bar:

| Table | Action | Rows | Type | Collation | Size | Overhead |
|---------------|--------|------|--------|-----------------|-----------|----------|
| actor | Browse | 140 | InnoDB | utf8_general_ci | 33.0 KiB | - |
| actor_info | Browse | 140 | InnoDB | utf8_general_ci | 44.0 KiB | - |
| address | Browse | 140 | InnoDB | utf8_general_ci | 14.0 KiB | - |
| category | Browse | 140 | InnoDB | utf8_general_ci | 14.0 KiB | - |
| city | Browse | 140 | InnoDB | utf8_general_ci | 44.0 KiB | - |
| country | Browse | 140 | InnoDB | utf8_general_ci | 14.0 KiB | - |
| customer | Browse | 140 | InnoDB | utf8_general_ci | 120.0 KiB | - |
| customer_list | Browse | 140 | InnoDB | utf8_general_ci | 120.0 KiB | - |
| film | Browse | 140 | InnoDB | utf8_general_ci | 172.0 KiB | - |
| film_actor | Browse | 140 | InnoDB | utf8_general_ci | 172.0 KiB | - |
| film_category | Browse | 140 | InnoDB | utf8_general_ci | 49.0 KiB | - |
| film_text | Browse | 140 | InnoDB | utf8_general_ci | 1.0 KiB | - |
| inventory | Browse | 140 | InnoDB | utf8_general_ci | 44.0 KiB | - |

BRIEF HISTORY OF SQL

- SQL stands for **Structured Query Language**, and was designed in the early 1970s at IBM to manipulate and retrieve data stored in a **relational database management system (RDMS)**
- There is a **universal standard** for SQL set by the International Organization for Standards and the American National Standards Institute (**ANSI**), with updates released every ~**3-5 years**
- Vendors are constantly adding new features on top of the standards, which creates different “*flavors*” of SQL (***MySQL***, ***PostgreSQL***, ***SQLite***, ***etc.***)

COMMON FLAVORS OF SQL



HEY THIS IS IMPORTANT!

These flavors of SQL are much more *similar* than they are different – all are based on the same universal standard, with slight variations in syntax.

POPULAR MySQL EDITORS



Mac, Windows, Linux



Web App (works on any OS)



Windows only



Sequel Pro

Mac only



HEY THIS IS IMPORTANT!

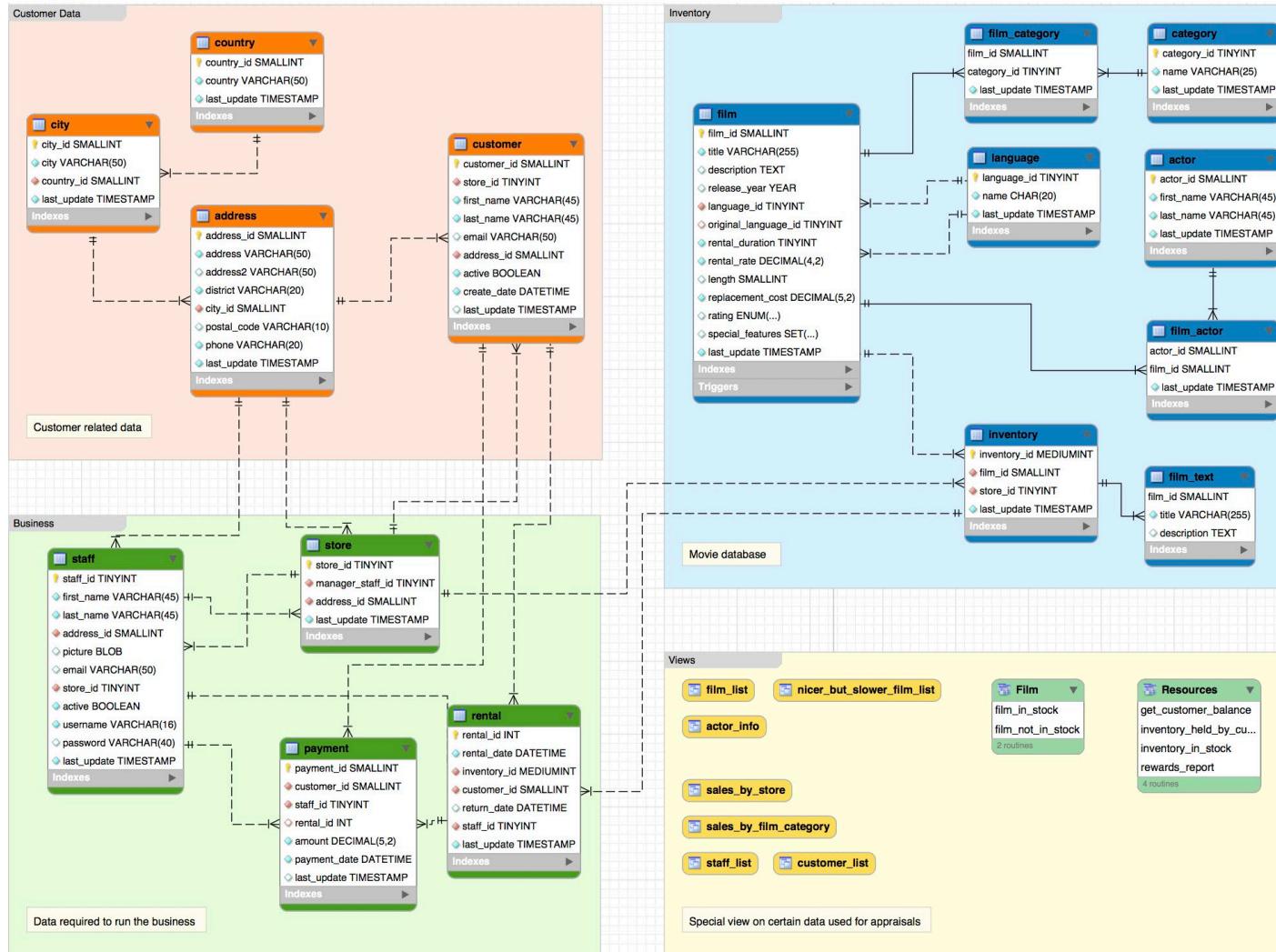
We'll be practicing MySQL using MySQL Workbench, but rather than thinking "*I learned MySQL*" or "*I can use Workbench*", you should think "***I'm learning SQL***". Period. You'll be able to use the concepts and syntax you learn here universally.

RELATIONAL DATABASES

The image shows a screenshot of the phpMyAdmin web application. On the left, there is a detailed database schema diagram with various tables like actor, address, city, country, customer, film, film_actor, film_category, inventory, and staff. In the center, there is a table named 'rental' with columns: rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update. Below it, a query results table shows rows 0-24 of data from the 'rental' table. To the right, a query editor displays an SQL query for finding the most frequent customer. At the bottom right, there is a table showing statistics for all databases.

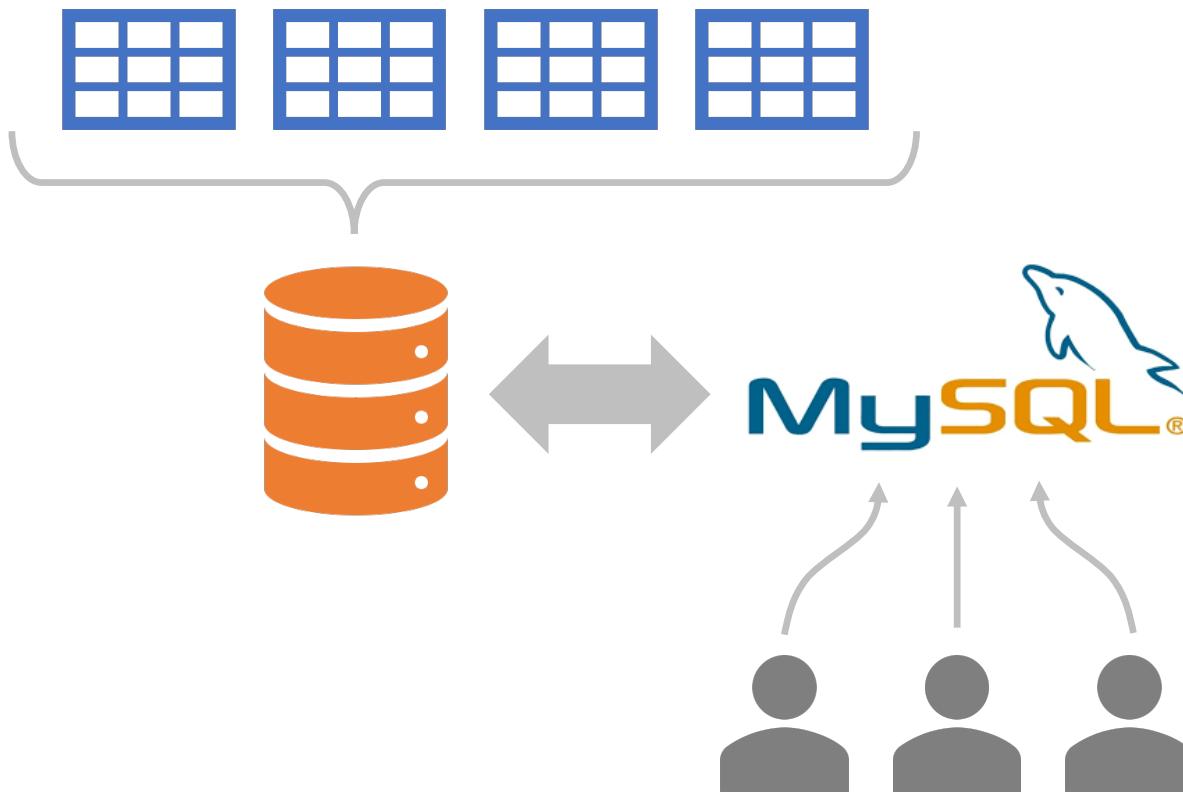
| Table | Action | Rows | Type | Collation | Size | Overhead |
|-----------------|---|-------|--------|-----------------|-----------|----------|
| actor | Browse Structure Search Insert Empty Drop | 200 | InnoDB | utf8_general_ci | 33.0 Kib | - |
| actor.info | Browse Structure Search Insert Empty Drop | - | View | - | - | - |
| address | Browse Structure Search Insert Empty Drop | 600 | InnoDB | utf8_general_ci | 94.0 Kib | - |
| city | Browse Structure Search Insert Empty Drop | 100 | InnoDB | utf8_general_ci | 16.0 Kib | - |
| country | Browse Structure Search Insert Empty Drop | 200 | InnoDB | utf8_general_ci | 44.0 Kib | - |
| customer | Browse Structure Search Insert Empty Drop | 1000 | InnoDB | utf8_general_ci | 16.0 Kib | - |
| customer_info | Browse Structure Search Insert Empty Drop | 2000 | InnoDB | utf8_general_ci | 320.0 Kib | - |
| customer_rental | Browse Structure Search Insert Empty Drop | 1000 | InnoDB | utf8_general_ci | 16.0 Kib | - |
| film | Browse Structure Search Insert Empty Drop | 1000 | InnoDB | utf8_general_ci | 172.0 Kib | - |
| film_actor | Browse Structure Search Insert Empty Drop | 1000 | InnoDB | utf8_general_ci | 172.0 Kib | - |
| film_category | Browse Structure Search Insert Empty Drop | 1000 | InnoDB | utf8_general_ci | 44.0 Kib | - |
| film_text | Browse Structure Search Insert Empty Drop | - | View | - | - | - |
| inventory | Browse Structure Search Insert Empty Drop | 1,000 | MyISAM | utf8_general_ci | 317.0 Kib | - |

A DATABASE CAN CONTAIN MANY RELATED TABLES



- The database on the left includes **16 related tables**, containing information about:
 - **Customers** (Name, Address, etc.)
 - **Business** (Staff, Rentals, etc.)
 - **Inventory** (Films, Categories, etc.)
- We'll start by using MySQL to create and modify **individual tables**, then later in the course we will get into **database design** and discuss optimal table relationships

RELATIONAL DATABASE MANAGEMENT SYSTEMS



- Relational databases consist of data stored in multiple **tables**
- **Relational database management systems (RDBMS)** serve as an interface to access and manage relational databases, using SQL
- RDBMS administrators can grant access to **users** and set specific roles and permissions

DOWNLOAD & SETUP

The image shows a screenshot of the phpMyAdmin interface. On the left, there is a database schema diagram with various tables like actor, address, category, film, film_actor, film_category, inventory, and rental. In the center, there is a query results page titled "Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)" displaying a table of rental records. On the right, there is a privilege history page showing a list of actions taken on various tables.

phpMyAdmin demo - My

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-26 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 22:54:33 | 1711 | 33 | 2005-06-01 00:00:00 | 2 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 22:54:33 | 3043 | 2459 | 2005-06-01 00:00:00 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 22:54:33 | 202 | 222 | 2005-06-01 00:00:00 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 22:54:33 | 3083 | 549 | 2005-06-01 00:00:00 | 2 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 22:54:33 | 3111 | 205 | 2005-06-01 00:00:00 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 22:54:33 | 2314 | 2465 | 2005-06-01 00:00:00 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

✓ Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)

```
1. SELECT customer.first_name, customer.last_name, COUNT(DISTINCT rental_id) AS rentals
2. FROM rental
3. LEFT JOIN customer
4. ON customer.customer_id = rental.customer_id
5. ORDER BY COUNT(DISTINCT rental_id) DESC
```

| Table | Action | Rows | Type | Collation | Size | Overhead |
|---------------|--|-------|--------|-----------------|----------|----------|
| actor | Browse > Structure > Search > Insert > Empty > Drop | 200 | InnoDB | utf8_general_ci | 33,0 kB | - |
| actor_info | Browse > Structure > Search > Insert > Empty > Drop | - | InnoDB | utf8_general_ci | - | - |
| address | Browse > Structure > Search > Insert > Empty > Drop | 400 | InnoDB | utf8_general_ci | 14,0 kB | - |
| category | Browse > Structure > Search > Insert > Empty > Drop | 100 | InnoDB | utf8_general_ci | 6,0 kB | - |
| film | Browse > Structure > Search > Insert > Empty > Drop | 100 | InnoDB | utf8_general_ci | 14,0 kB | - |
| film_actor | Browse > Structure > Search > Insert > Empty > Drop | 500 | InnoDB | utf8_general_ci | 28,0 kB | - |
| film_category | Browse > Structure > Search > Insert > Empty > Drop | 100 | InnoDB | utf8_general_ci | 14,0 kB | - |
| film_text | Browse > Structure > Search > Insert > Empty > Drop | - | InnoDB | utf8_general_ci | - | - |
| inventory | Browse > Structure > Search > Insert > Empty > Drop | 1,000 | InnoDB | utf8_general_ci | 377,0 kB | - |
| language | Browse > Structure > Search > Insert > Empty > Drop | 1,000 | InnoDB | utf8_general_ci | 377,0 kB | - |
| rental | Browse > Structure > Search > Insert > Empty > Drop | 1,000 | InnoDB | utf8_general_ci | 377,0 kB | - |
| staff | Browse > Structure > Search > Insert > Empty > Drop | 100 | InnoDB | utf8_general_ci | 6,0 kB | - |
| store | Browse > Structure > Search > Insert > Empty > Drop | - | InnoDB | utf8_general_ci | - | - |

MySQL DOWNLOAD & SETUP – OVERVIEW

Step 1

Download Community Server

This allows SQL to run on your machine

Step 2

Download MySQL Workbench

*This is the program we'll use as our RDBMS
(it's intuitive, and works across operating systems)*

Step 3

Connect Workbench to Server

*We'll get you connected to the server so you can use
Workbench to start managing your schemas*

Step 4

Review Workbench Interface

*We'll take a quick tour of the Workbench interface to get
you familiar with the layout and key components*

Step 5

Create the Databases

*We'll run a quick SQL script to build some schemas and tables
we'll be exploring throughout the course (this part is easy!)*



HEY THIS IS IMPORTANT!

If you took one of my other courses, and have already installed Community Server and MySQL Workbench, then you can skip ahead to creating the databases. No need to re-install. Whatever version you have is great.

STEP 1: COMMUNITY SERVER (MAC)



HEY THIS IS IMPORTANT!

If you took one of my other courses, and have already installed Community Server and MySQL Workbench, then you can skip ahead to creating the database. No need to re-install. Whatever version you have is great.

The screenshot shows the phpMyAdmin interface on a Mac OS X desktop. The title bar says "phpMyAdmin". The left sidebar shows a tree view of the database structure. The main area has three tabs: "rental" (selected), "query" (with a query result showing customer names and rental counts), and "structure" (listing tables like actor, address, category, etc.).

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-28 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-05-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:03:41 | 26 | 333 | 2005-05-01 23:41:34 | 1 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:03:41 | 26 | 222 | 2005-05-01 23:41:34 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:03:41 | 27 | 549 | 2005-05-01 23:41:34 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:03:41 | 36 | 262 | 2005-05-01 23:40:34 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

Query results:

```
✓ Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)  
1. SELECT  
2.     customer.first_name,  
3.     customer.last_name  
4.     COUNT(DISTINCT rental_id) AS rentals  
5.     FROM rental  
6.     JOIN store  
7.     ON rental.store_id = store.id  
8.     GROUP BY customer.first_name,  
9.             customer.last_name  
10.    ORDER BY  
11.        COUNT(DISTINCT rental_id) DESC  
12.       
```

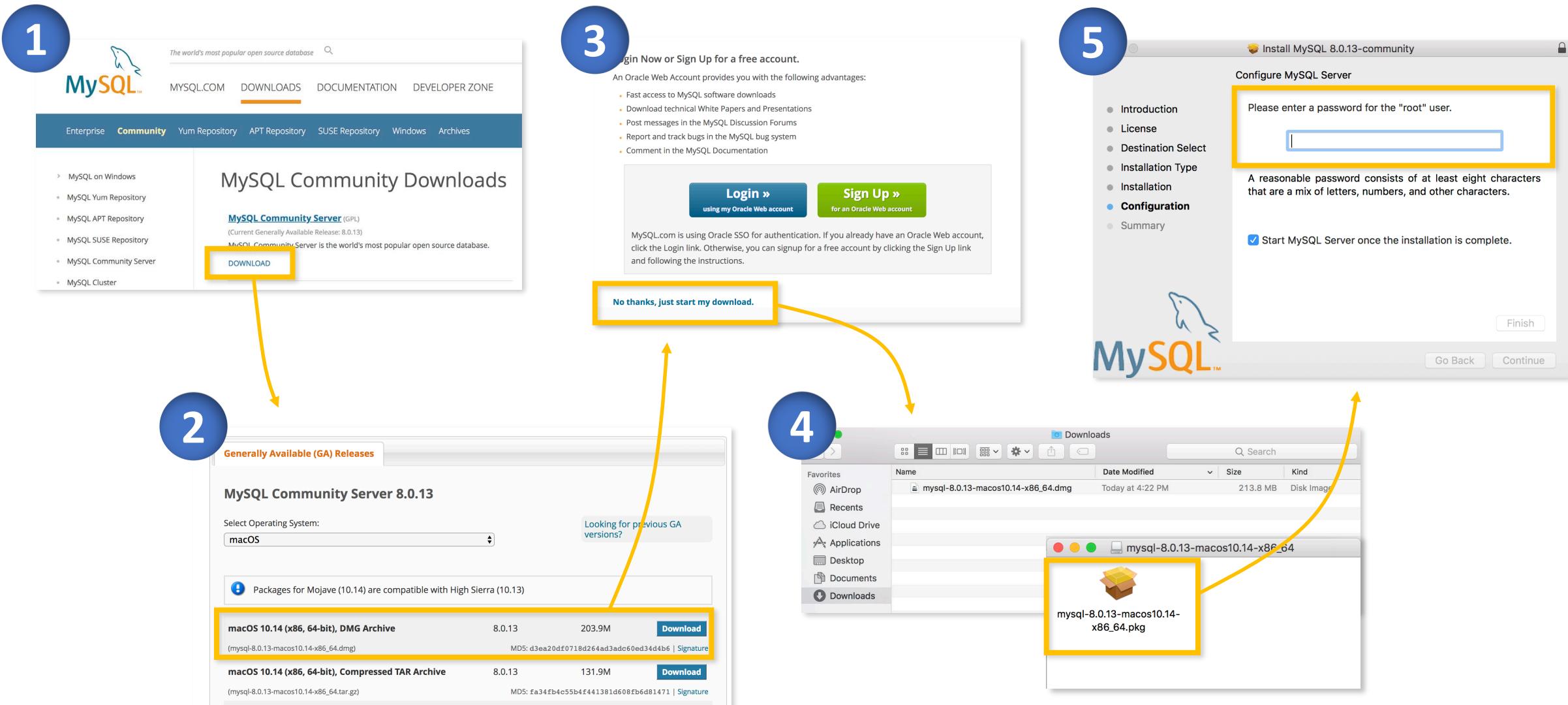
| Table | Action | Type | Collation | Size | Overhead | | | |
|------------------|--------|-----------|-----------|-------|----------|------------|----------|---|
| actor | Browse | Structure | Search | Empty | Drop | 201 InnoDB | 93.0 KiB | - |
| actor_info | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| address | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| category | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| city | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| customer | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| customer_address | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| customer_info | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| customer_payment | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| film | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| film_actor | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| film_category | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| language | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| payment | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| rental | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| store | Browse | Structure | Search | Empty | Drop | 40 InnoDB | 14.0 KiB | - |
| Table | Action | Type | Collation | Size | Overhead | | | |

MySQL COMMUNITY SERVER – MAC DOWNLOAD GUIDE



- 1 Go to <https://dev.mysql.com/downloads> and download **MySQL Community Server**
- 2 Select the **MacOS** operating system, and download the **DMG Archive** version
 - *Note: you'll likely see a later version than the one shown (just download the latest)*
- 3 No need to Login or Sign Up, just click "**No thanks, just start my download**"
- 4 Find the install file in your downloads, then double click to run the installer package
- 5 Click through each install step, leaving defaults unless you need customized settings
 - *Note: Make sure to store your **root password** somewhere, you'll need this later!*

MySQL COMMUNITY SERVER – MAC DOWNLOAD GUIDE



STEP 1: COMMUNITY SERVER (PC)

The screenshot shows the phpMyAdmin interface with the following sections:

- Left Sidebar:** Shows a detailed database schema diagram with various tables like actor, address, category, customer, film, film_copy, film_text, and inventory.
- Main Area:** A table titled "rental" showing 10 rows of data with columns: rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update.
- Query Results:** A results page showing rows 0 - 24 of 599 total, with a query took 0.0212 seconds. The query is:

```
1 SELECT
2     customer.first_name,
3     customer.last_name,
4     COUNT(DISTINCT rental_id) AS rental_count
5     FROM rental
6     LEFT JOIN customer
7     ON customer.customer_id = rental.customer_id
8     GROUP BY
9     customer.first_name,
10    customer.last_name
11 ORDER BY
12    COUNT(DISTINCT rental_id) DESC
```

- Table Structure:** A table showing the structure of the "rental" table with columns: rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update.



HEY THIS IS IMPORTANT!

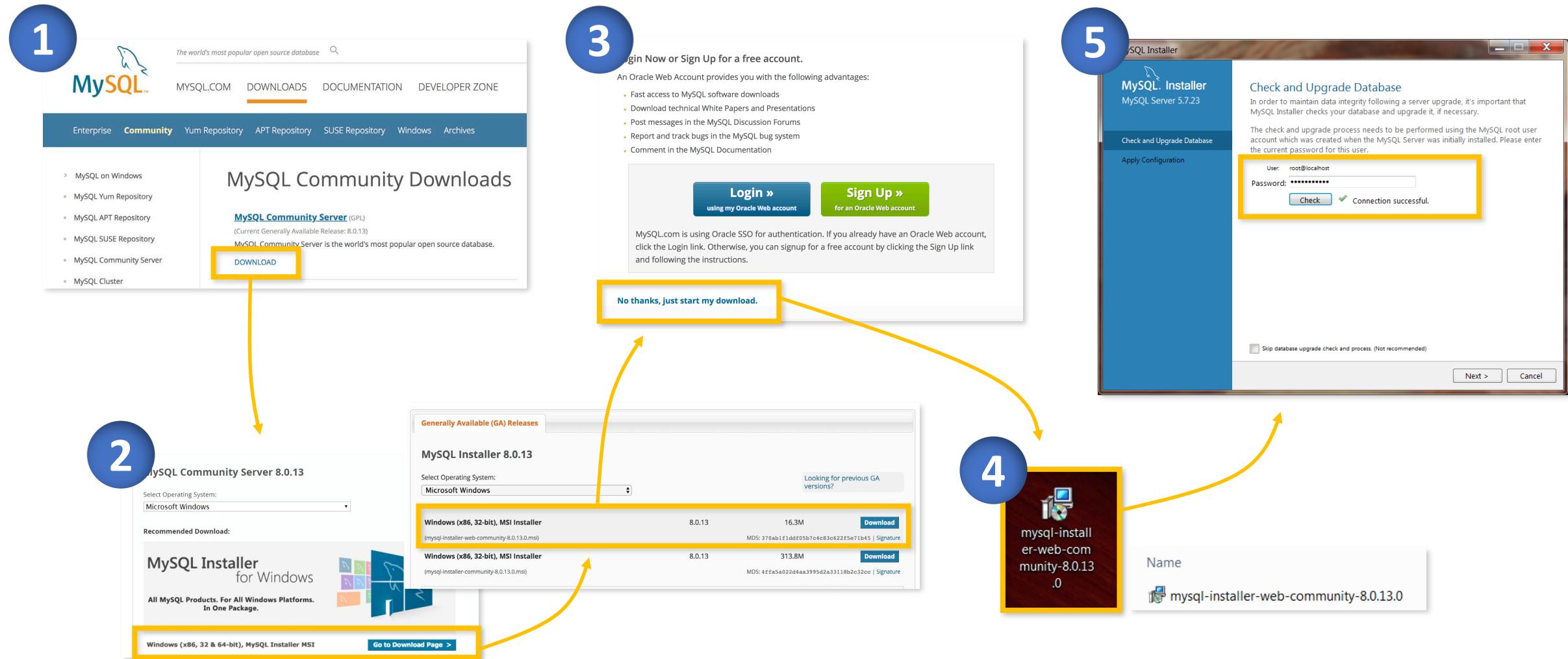
If you took one of my other courses, and have already installed Community Server and MySQL Workbench, then you can skip ahead to creating the database. No need to re-install. Whatever version you have is great.

MySQL COMMUNITY SERVER – PC DOWNLOAD GUIDE



- 1 Go to <https://dev.mysql.com/downloads> and download **MySQL Community Server**
- 2 Select the **Microsoft Windows** operating system, and the **Installer MSI** download
 - *Note: On the download page you may see two versions: select mysql-installer-web-community if you are connected to the internet, and keep in mind that you may see a later version than the one shown (just download the latest)*
- 3 No need to Login or Sign Up, just click “**No thanks, just start my download**”
- 4 Find the install file in your downloads, then double click to run the installer package
- 5 Click through each install step, leaving defaults unless you need customized settings
 - *Note: Make sure to store your **root password** somewhere, you'll need this later!*

MySQL COMMUNITY SERVER – PC DOWNLOAD GUIDE



STEP 2: MySQL WORKBENCH (MAC)



HEY THIS IS IMPORTANT!

If you took one of my other courses, and have already installed Community Server and MySQL Workbench, then you can skip ahead to creating the database. No need to re-install. Whatever version you have is great.

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view of the database schema. In the center, a query editor window displays a SELECT statement and its results. The results show 24 rows from the rental table, ordered by COUNT(DISTINCT rental_id) DESC. On the right, there's a table viewer showing the structure of the rental table.

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-26 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-06-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:04:13 | 1652 | 3 | 2005-05-30 01:43:11 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 879 | 22 | 2005-06-01 04:23:21 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:08:07 | 792 | 549 | 2005-05-25 01:57 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:11:45 | 104 | 269 | 2005-05-29 04:53 | 1 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |



MySQL WORKBENCH – MAC DOWNLOAD GUIDE

- 1 Go to <https://dev.mysql.com/downloads/workbench>, scroll down to **Generally Available (GA) Releases**, and select the **MacOS** operating system
- 2 We'll be using version **8.0.16** for this course, so you can either click "***Looking for previous GA versions?***" to search for the same one, or simply download the latest available
- 3 No need to Login or Sign Up, just click "***No thanks, just start my download***"
- 4 Find the install file in your downloads, click the MySQL Workbench logo (*with the dolphin*) and drag it into your **Applications** folder
- 5 Look for MySQL workbench in your list of applications, double click to launch, then proceed to ***Step 3: Connecting to the server***



MySQL WORKBENCH – MAC DOWNLOAD GUIDE

1

The screenshot shows the MySQL Workbench 8.0.16 download page. A yellow box highlights the 'Select Operating System' dropdown menu where 'macOS' is chosen. Another yellow box highlights the 'Looking for previous GA versions?' link.

3

The screenshot shows the MySQL login or sign-up page. It displays a list of advantages for having an Oracle Web Account and two buttons: 'Login >' and 'Sign Up >'. A yellow box highlights the 'No thanks, just start my download.' link.

5

The screenshot shows the Mac OS X Applications folder. A yellow arrow points from the 'MySQLWorkbench' icon in the Applications list to the 'Applications' folder icon at the bottom of the sidebar.

2

*Look for version **8.0.16**, or download the latest*

The screenshot shows the MySQL Workbench 6.3.10 download page. A yellow box highlights the 'Select Version' dropdown menu where '6.3.10' is chosen. Another yellow box highlights the 'Looking for the latest GA version?' link.

4

The screenshot shows the MySQL Workbench 8.0 splash screen. It includes instructions to 'Drag the MySQL Workbench icon to the Applications folder.' A yellow box highlights the MySQL Workbench icon, and another yellow box highlights the 'Applications' folder icon.

STEP 2: MySQL WORKBENCH (PC)



HEY THIS IS IMPORTANT!

If you took one of my other courses, and have already installed Community Server and MySQL Workbench, then you can skip ahead to creating the database. No need to re-install. Whatever version you have is great.

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view of a database schema with tables like actor, address, category, city, customer, film, film_actor, film_category, inventory, and rental. In the center, a query editor displays a SELECT statement and its results for rows 0-24. On the right, a table editor shows the structure and data for the 'rental' table.

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-28 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 22:55:39 | 1711 | 408 | 2005-05-21 22:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:03:47 | 2459 | 333 | 2005-06-01 14:34 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 2079 | 222 | 2005-06-02 03:23:43 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:05:29 | 2792 | 549 | 2005-05-27 05:08 | 2 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:10:53 | 2679 | 269 | 2005-05-29 22:40 | 3 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

MySQL WORKBENCH – PC DOWNLOAD GUIDE

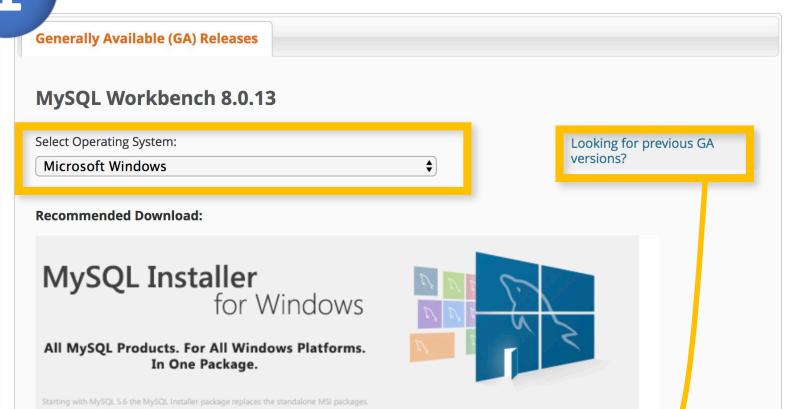


- 1 Go to <https://dev.mysql.com/downloads/workbench>, scroll down to **Generally Available (GA) Releases**, and select the **Microsoft Windows** operating system
- 2 We'll be using version **8.0.13** for this course, so you can either click "***Looking for previous GA versions?***" to search for the same one, or simply download the latest available
- 3 No need to Login or Sign Up, just click "***No thanks, just start my download***"
- 4 Find the install file in your downloads, double click to run the installation process, and stick with default settings unless you need a custom configuration
- 5 Look for MySQL workbench in your list of programs, double click to launch, then proceed to ***Step 3: Connecting to the server***
 - ***Note:*** You may see a warning if you aren't on **Windows 10+**, but most older systems (i.e. Windows 7) should be compatible

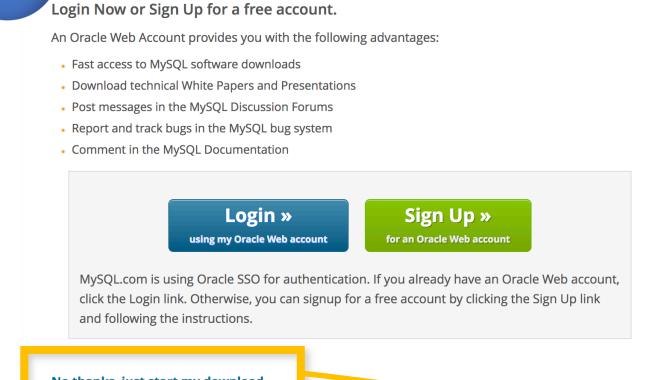


MySQL WORKBENCH – PC DOWNLOAD GUIDE

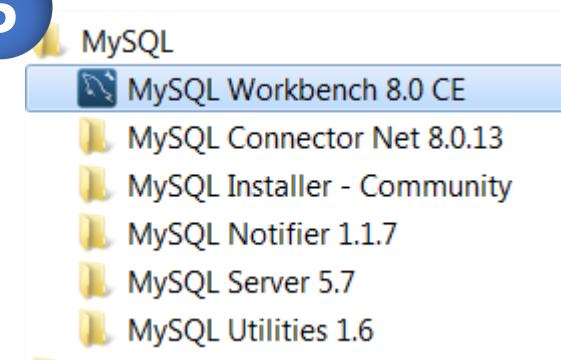
1



3

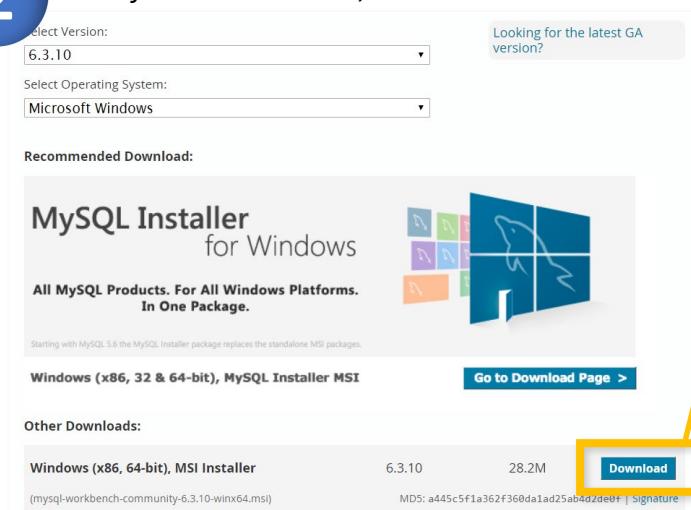


5

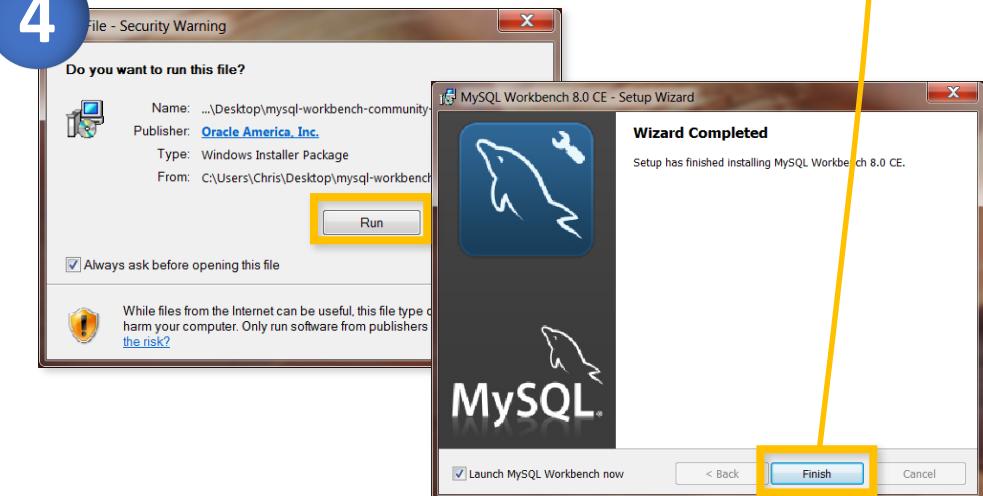


2

Look for version 8.0.13, or download the latest



4



STEP 3: CONNECTING TO THE SERVER

The screenshot shows the phpMyAdmin interface with a blue header. On the left, there's a tree view of the database schema. The main area has three panes: a top-left pane showing a table structure with columns like rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update; a middle-right pane showing a list of rows from the rental table; and a bottom-right pane showing a query editor with a SQL query and its execution results.

Table Structure:

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-28 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-06-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:04:44 | 2452 | 45 | 2005-06-01 01:21 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 207 | 222 | 2005-06-02 04:21 | 2 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:08:07 | 273 | 549 | 2005-05-27 01:07 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:11:51 | 2995 | 25 | 2005-05-29 20:53 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 128 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

Query Results:

```
✓ Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)
```

```
1. SELECT
2.     customer.first_name,
3.     customer.last_name,
4.     COUNT(DISTINCT rental_id) AS rentals
5. FROM rental
6. JOIN customer
7. ON customer.customer_id = rental.customer_id
8. GROUP BY customer_id
9. ORDER BY
10.    customer.first_name,
11.    customer.last_name
12. ORDER BY
13.    COUNT(DISTINCT rental_id) DESC
```

Table List:

| Table | Action | Rows | Type | Collation | Size | Overhead |
|------------------|---|------|--------|-----------------|----------|----------|
| actor | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 200 | InnoDB | utf8_general_ci | 93.0 kB | - |
| actor_info | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 14 | InnoDB | utf8_general_ci | 14.0 kB | - |
| address | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 100 | InnoDB | utf8_general_ci | 64.0 kB | - |
| category | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 10 | InnoDB | utf8_general_ci | 14.0 kB | - |
| country | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 200 | InnoDB | utf8_general_ci | 128.0 kB | - |
| customer | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 200 | InnoDB | utf8_general_ci | 128.0 kB | - |
| customer_address | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 100 | InnoDB | utf8_general_ci | 128.0 kB | - |
| customer_email | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 100 | InnoDB | utf8_general_ci | 128.0 kB | - |
| film | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 500 | InnoDB | utf8_general_ci | 372.0 kB | - |
| film_actor | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 500 | InnoDB | utf8_general_ci | 372.0 kB | - |
| film_category | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 500 | InnoDB | utf8_general_ci | 372.0 kB | - |
| inventory | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 500 | InnoDB | utf8_general_ci | 372.0 kB | - |
| language | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 20 | InnoDB | utf8_general_ci | 80.0 kB | - |
| rental | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 599 | InnoDB | utf8_general_ci | 317.0 kB | - |
| staff | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 10 | InnoDB | utf8_general_ci | 10.0 kB | - |
| staff_level | Browse ▾ Structure ▾ Search ▾ Insert ▾ Empty ▾ Drop | 10 | InnoDB | utf8_general_ci | 10.0 kB | - |

CONNECTING TO THE SERVER

- 1 After launching Workbench, check the **MySQL Connections** section on the welcome page
 - *If you see a connection already, right-click to **Edit Connection**, otherwise click the **plus sign (+)** to add a new one*
- 2 Name the connection “**mavenmovies**”, confirm that the Username is “**root**”, and click **OK**
- 3 Once you see the **mavenmovies** connection on your welcome screen, simply click the tile and enter your **root password** to complete the connection

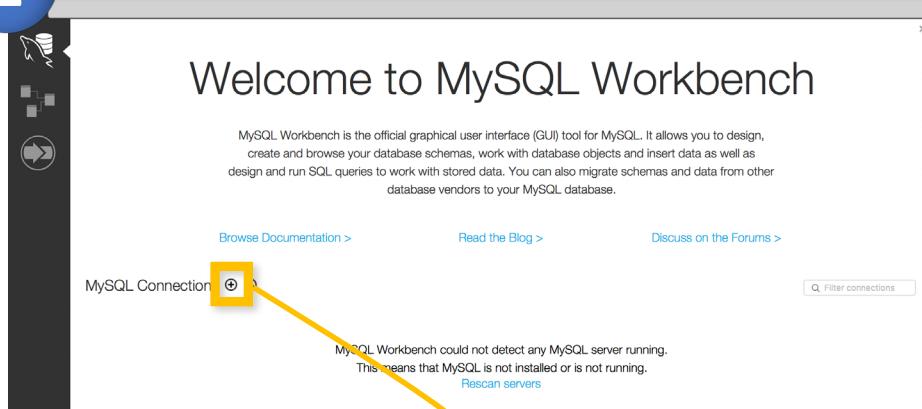


Fun Fact!

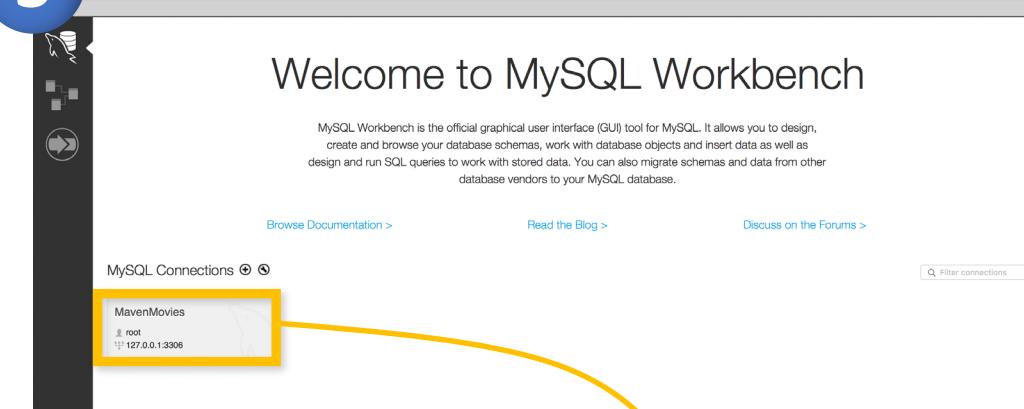
Maven Movies is the name of the database I used when I made my first course. I always name my connections ‘mavenmovies’ as tribute. **It does not matter what you name your connection. Name it anything you want!**

CONNECTING TO THE SERVER

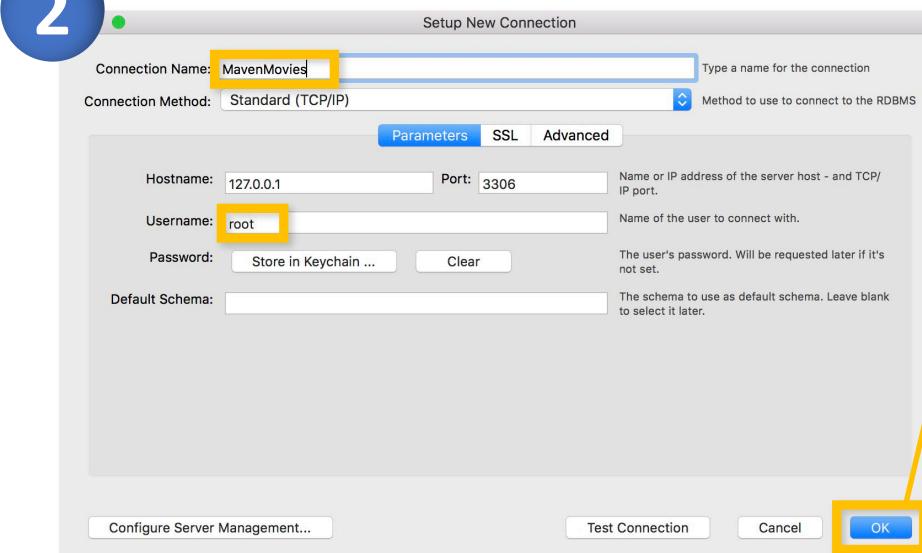
1



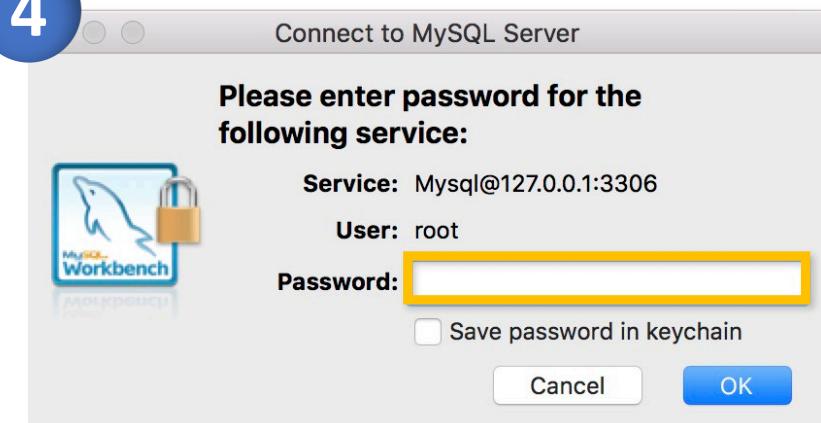
3



2



4



STEP 4: MySQL WORKBENCH INTERFACE

The screenshot displays the MySQL Workbench interface with the following components:

- Left Panel:** Shows a detailed database schema diagram with various tables like actor, address, category, customer, film, film_actor, film_category, inventory, and rental.
- Table Data Grid:** A grid showing data from the 'rental' table. The columns are: rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update. The data includes rows from May 24, 2005, to May 25, 2005.
- Query Editor:** Displays a SQL query and its results. The query is:

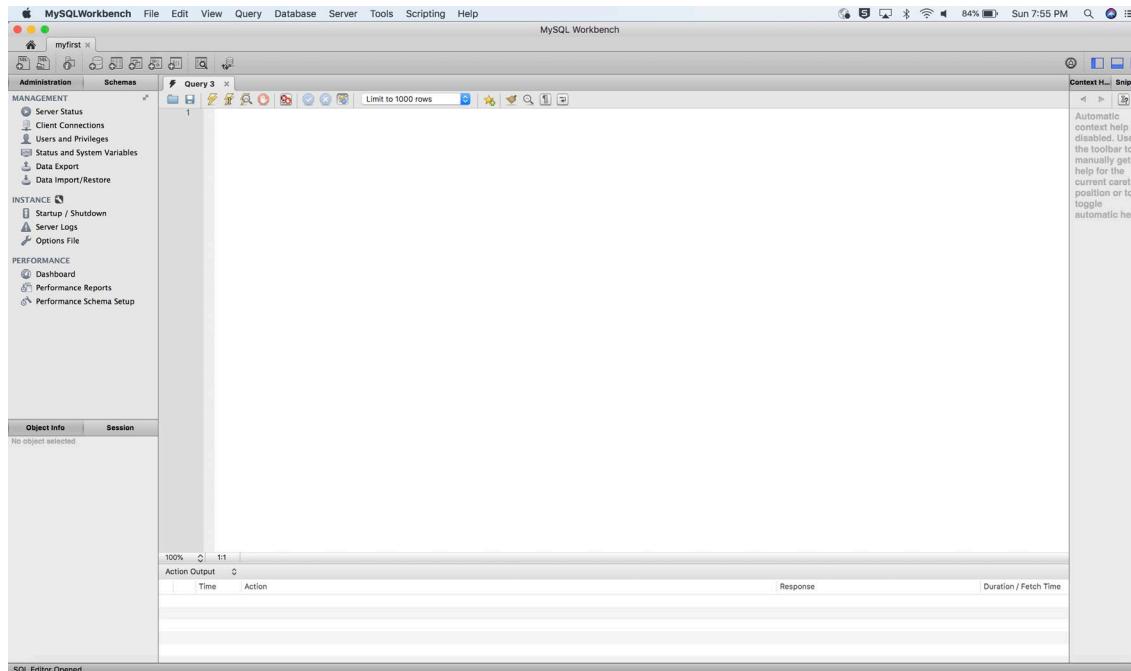
```
SELECT
    customer.first_name,
    customer.last_name
    COUNT(DISTINCT rental_id) AS rental_count
FROM
    inventory
LEFT JOIN rental
ON
    inventory.inventory_id = rental.inventory_id
GROUP BY
    customer.first_name,
    customer.last_name
ORDER BY
    COUNT(DISTINCT rental_id) DESC
```

The results show 599 total rows, with the top result being a customer with 24 rentals.
- Table List:** A list of all tables in the database, including their type, size, and overhead.

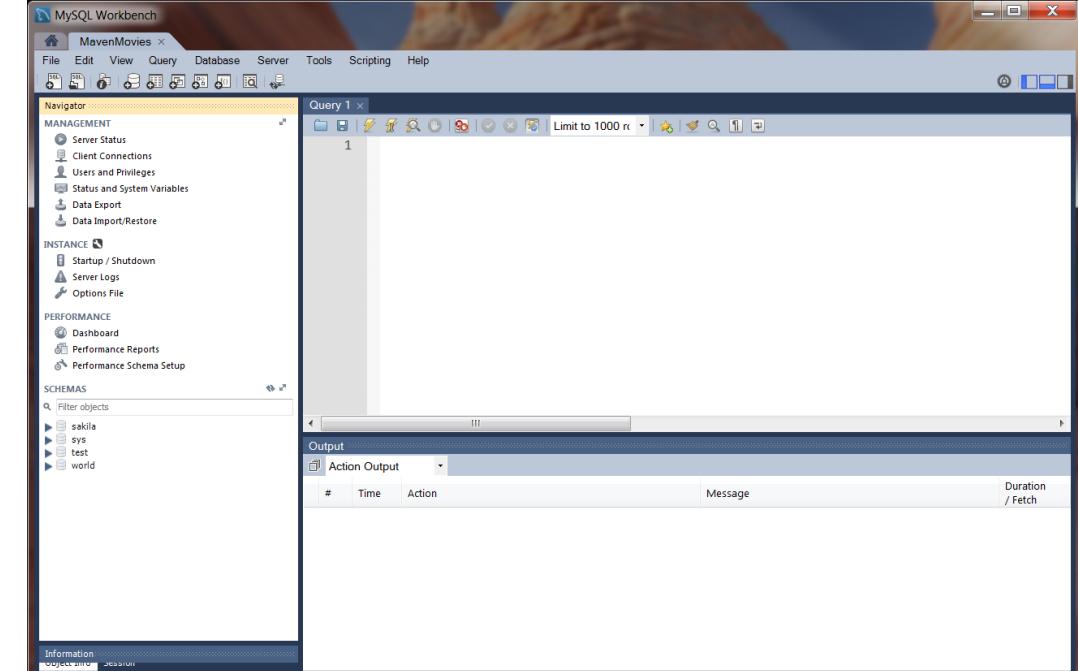
MySQL WORKBENCH INTERFACE (MAC VS. PC)



Mac interface



PC interface



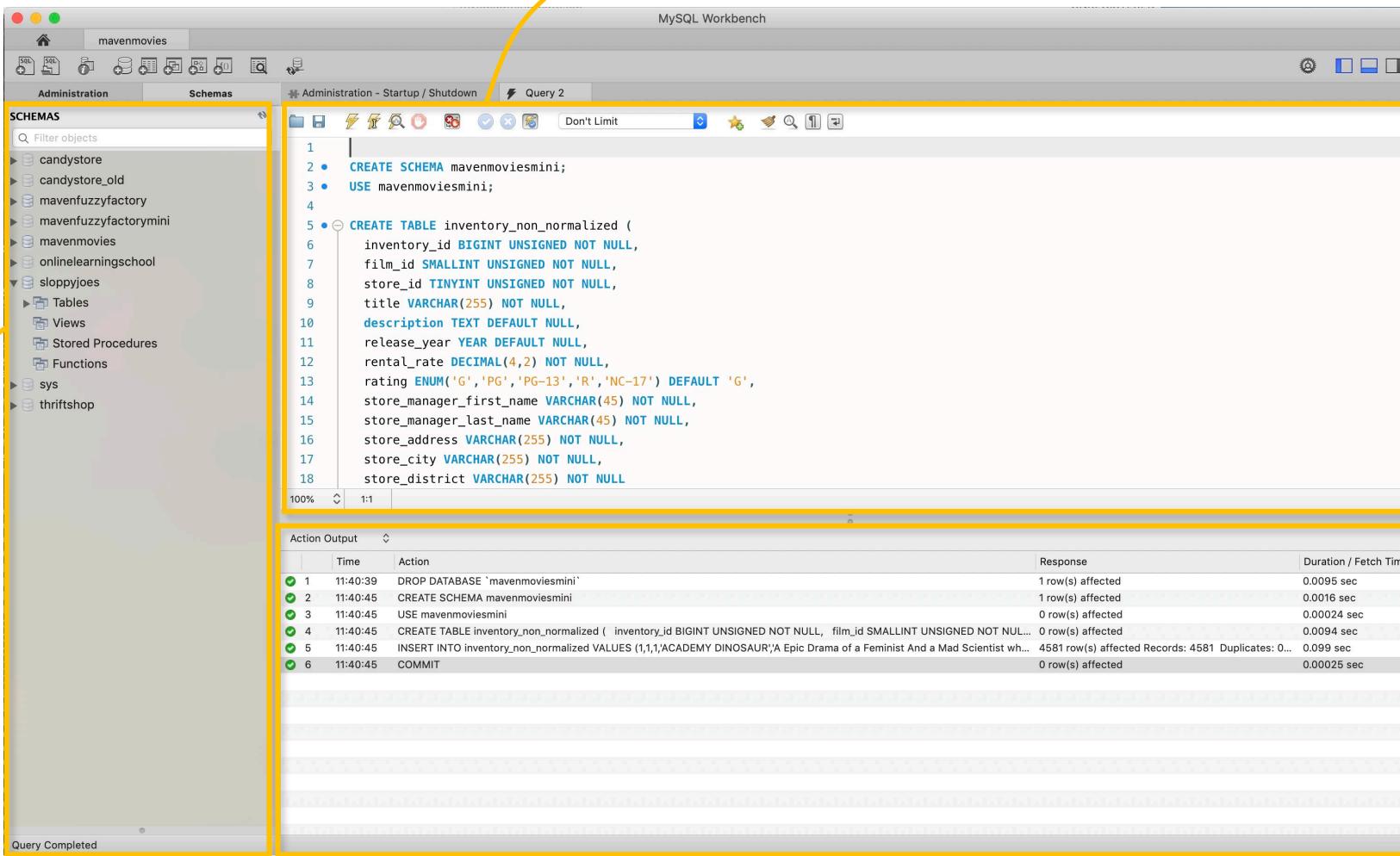
HEY THIS IS IMPORTANT!

Workbench looks slightly different on **Mac** vs. **PC**, but everything you need is found in the same place. While the course is recorded on a Mac, but you should have no problem keeping up on a PC

QUICK TOUR: THE WORKBENCH INTERFACE

Query Editor Window

This is where you write and run your code



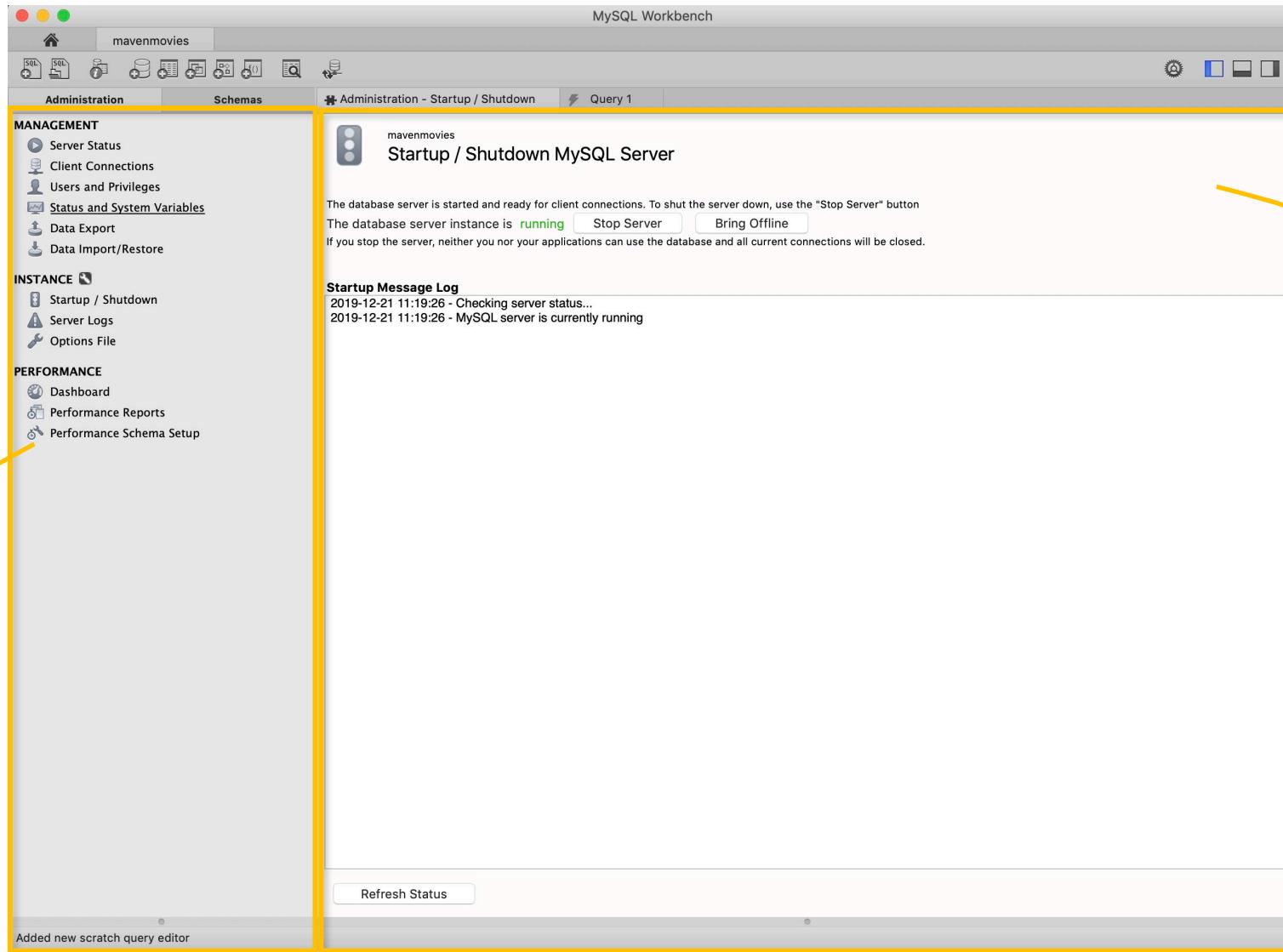
Schemas Tab

Here you can view tables and views in your schemas

Action Output

This is a summary of actions taken by the server (TIP: the Response column is great for troubleshooting!)

QUICK TOUR: THE WORKBENCH INTERFACE



Administration Tab

Here you can select the Management, Instance, and Performance tools

Tool Tabs

This is where the various Management, Instance, and Performance tools show up after they are selected in the Administration Tab.

Note: these show up as additional tabs alongside tabs you have open for SQL query editing. They co-exist in the same section of Workbench, even though their functions are different

STEP 5: CREATING THE DATABASES

The screenshot illustrates the phpMyAdmin interface, a web-based MySQL management tool. It features three main windows:

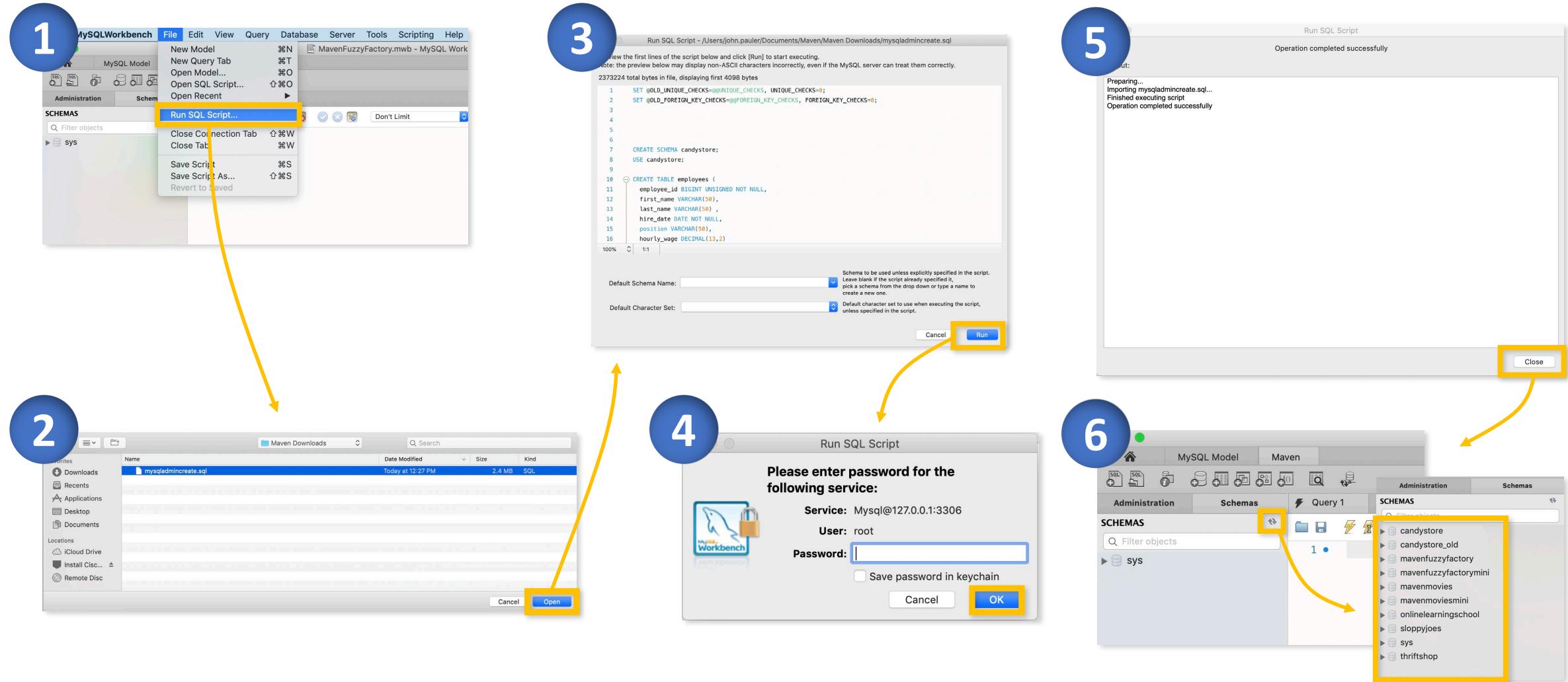
- Main Window:** Displays a table with 10 rows of rental data. The columns include rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, and last_update.
- Second Window:** Shows the results of a SQL query: "SELECT customer.first_name, customer.last_name FROM rental GROUP BY customer.customer_id ORDER BY COUNT(DISTINCT rental_id) DESC". The results show the top 12 customers based on the number of distinct rentals.
- Third Window:** Lists various databases with their sizes and overhead.

| Table | Action | Rows | Type | Collation | Size | Overhead |
|----------------------|--------|-------|--------|-----------------|---------|----------|
| actor | Browse | 200 | InnoDB | utf8_general_ci | 33.0 K | - |
| actor_info | Browse | 400 | InnoDB | utf8_general_ci | 44.0 K | - |
| address | Browse | 400 | InnoDB | utf8_general_ci | 14.0 K | - |
| category | Browse | 400 | InnoDB | utf8_general_ci | 44.0 K | - |
| city | Browse | 400 | InnoDB | utf8_general_ci | 14.0 K | - |
| country | Browse | 400 | InnoDB | utf8_general_ci | 14.0 K | - |
| customer | Browse | 500 | InnoDB | utf8_general_ci | 120.0 K | - |
| customer_address | Browse | 500 | InnoDB | utf8_general_ci | 172.0 K | - |
| customer_email | Browse | 500 | InnoDB | utf8_general_ci | 172.0 K | - |
| customer_instrument | Browse | 500 | InnoDB | utf8_general_ci | 172.0 K | - |
| employee | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| employee_info | Browse | 500 | InnoDB | utf8_general_ci | 172.0 K | - |
| genrefilm | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| language | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| languagefilm | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| rental | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| rental_instrument | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| staff | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| staff_info | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| store | Browse | 500 | InnoDB | utf8_general_ci | 49.0 K | - |
| phpMyAdmin | Browse | 1,000 | InnoDB | utf8_general_ci | 317.0 K | - |
| phpMyAdmin demo - My | Browse | 1,000 | InnoDB | utf8_general_ci | 317.0 K | - |

CREATING YOUR INITIAL SCHEMAS

- 1 In MySQL Workbench, click **File** from the top menu, then select **Run SQL Script**
- 2 Navigate to the **createmavenconsulting.sql** file provided in the course resources
 - *This code will automatically generate the databases that we'll be exploring throughout the course*
- 3 After running the code, confirm the following:
 1. *You see results in the **Action Output** window, with **green check marks** and no errors in the **Response** column*
 2. *When you refresh the **Schemas** list, you should see a number of new schemas:*
 - **candystore** & **candystore_old**
 - **mavenfuzzyfactorymini** & **mavenmoviesmini**
 - **onlinelearningschool**
 - **sloppyjoes**
 - **sys** (*this one is built-in*)
 - **thriftshop**

CREATING THE DATABASE



CREATING, ALTERING & DELETING SCHEMAS AND TABLES



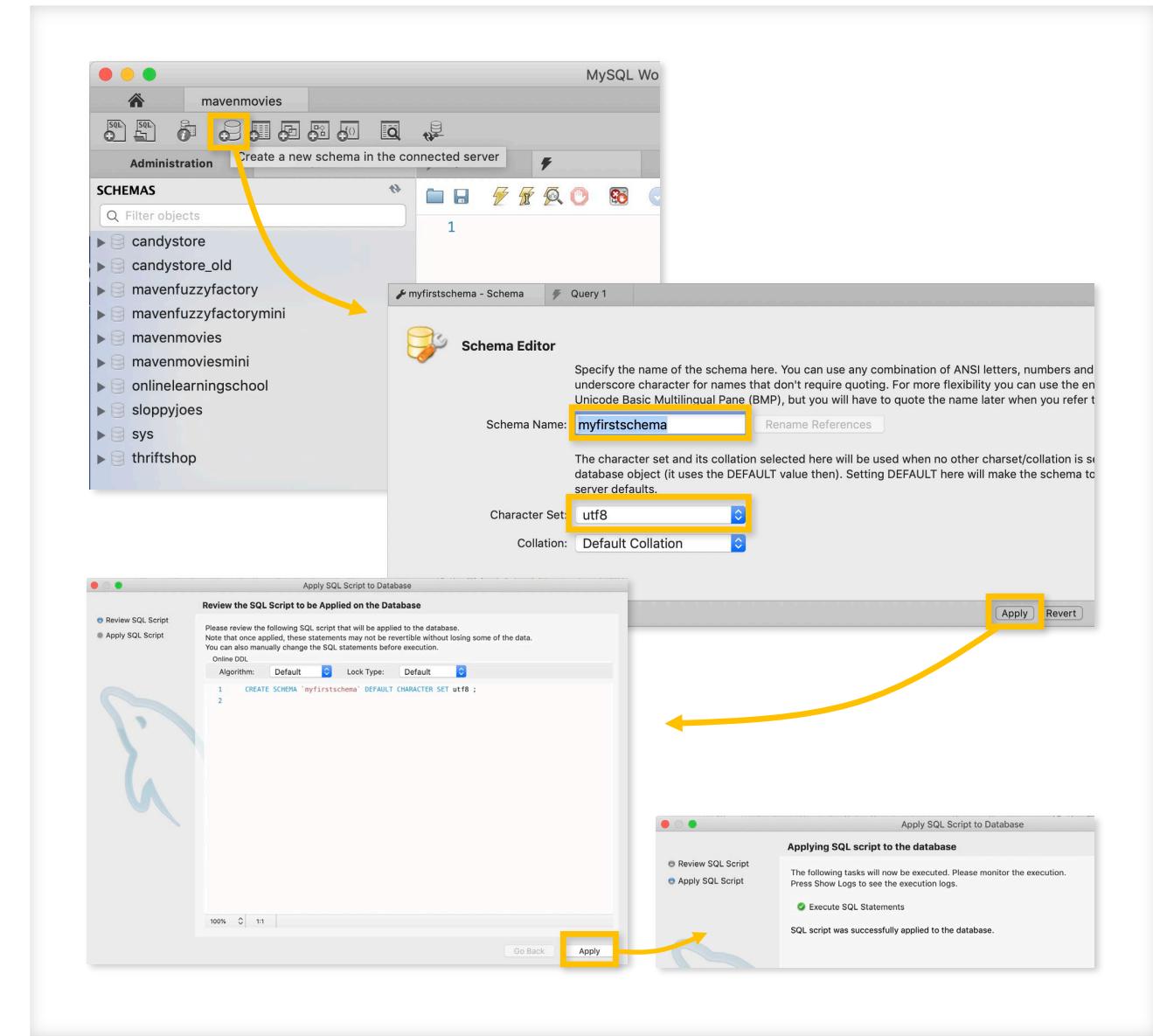
CREATE SCHEMA W/ UI

- In MySQL, the terms ‘Schema’ and ‘Database’ will be used interchangeably to discuss a collection of data tables which relate to one another

- We can create a new schema using Workbench’s user interface tools, with just a few quick clicks

- Whether we create schemas using the UI tools or using code is a matter of preference

MySQL WORKBENCH TOOLS IN ACTION:



CREATE SCHEMA W/ CODE

- We can also create a new schema using a simple SQL create statement
- When we use SQL code, we can choose the same options for our schema as when we use the UI
- Whether we create schemas using the UI tools or using code is a matter of preference

MySQL QUERY IN ACTION:

```
CREATE SCHEMA myfirstcodeschema;
```

ACTION OUTPUT:

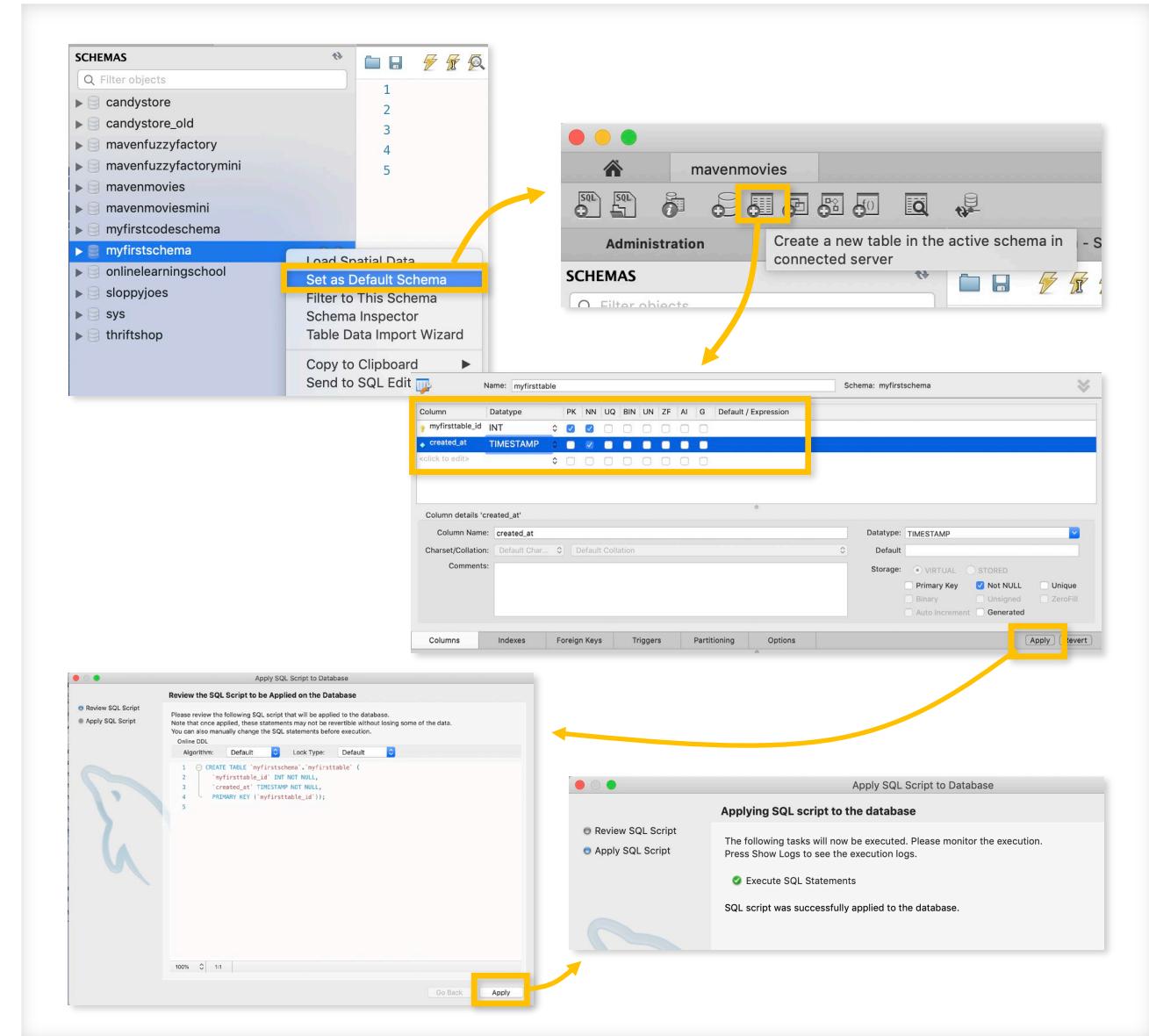
| Action Output | Time | Action |
|---------------|----------|---------------------------------|
| 1 | 14:14:25 | CREATE SCHEMA myfirstcodeschema |

CREATE TABLE W/ UI

- After our schema is created, we can begin populating our schema with tables that will contain records of data

- Just like when we created our schemas, we can create tables using Workbench's UI tools, or we can do it with code (again, this is a matter of preference)

MySQL WORKBENCH TOOLS IN ACTION:



CREATE TABLE W/ CODE

- After our schema is created, we can begin populating our schema with tables that will contain records of data

- Creating a new table with code typically involves:

- Selecting a schema to add the table to
- Giving the new table a name
- Specifying which columns to include
- Setting data formats and types

MySQL QUERY IN ACTION:

```
1
2 • USE myfirstcodeschema; -- specifying the schema to use
3
4 • CREATE TABLE myfirstcodetable (
5     myfirstcodetable_id BIGINT NOT NULL,
6     my_character_field VARCHAR(50),
7     my_text_field TEXT,
8     my_created_at TIMESTAMP
9 );
10
```

ACTION OUTPUT:

| Action Output | | | |
|---------------|----------|---|-------------------|
| | Time | Action | Response |
| 1 | 14:59:51 | USE myfirstcodeschema | 0 row(s) affected |
| 2 | 14:59:54 | CREATE TABLE myfirstcodetable (myfirstcodetable_i... | 0 row(s) affected |

COMMON MySQL DATA TYPES

| Data Type | Specifications |
|------------------|--|
| TINYINT | <i>Integer (-128 to 127)</i> |
| SMALLINT | <i>Integer (-32768 to 32767)</i> |
| MEDIUMINT | <i>Integer (-8388608 to 8388607)</i> |
| INT | <i>Integer (-2147483648 to 2147483647)</i> |
| BIGINT | <i>Integer (-9223372036854775808 to 9223372036854775807)</i> |
| FLOAT | <i>Decimal (precise to 23 digits)</i> |
| DOUBLE | <i>Decimal (23 to 53 digits)</i> |
| DECIMAL | <i>Decimal (to 65 digits – most precise)</i> |

| Data Type | Specifications |
|------------------|--|
| CHAR | <i>String (0 – 255)</i> |
| VARCHAR | <i>String (0 – 255)</i> |
| TINYTEXT | <i>String (0 – 255)</i> |
| TEXT | <i>String (0 – 65535)</i> |
| DATE | <i>YYYY-MM-DD</i> |
| DATETIME | <i>YYYY-MM-DD HH:MM:SS</i> |
| TIMESTAMP | <i>YYYYMMDDHHMMSS</i> |
| ENUM | <i>One of a number of preset options</i> |



NEW MESSAGE

From: Tom Strawberry (your first client)
Subject: Help Creating New Schema & 2 Tables

Hi there,

We're hoping that you can help us start tracking our marketing spend better. Could you please:

1. Create a new schema, called '**toms_marketing_stuff**'
2. Add a table called '**publishers**', with two columns: '**publisher_id**' (integer) and '**publisher_name**' (up to 65 chars)
3. Add another table called '**publisher_spend**' with '**publisher_id**' (integer), '**month**' (date), and **spend** (decimal)

Thanks, Tom

Reply

Forward

Helpful Tips

- Create the schema however you feel more comfortable, with SQL code or using Workbench's UI tools (**it would be good if you know how to do it both ways**)
- **Create one of the tables with code, and create the other using Workbench's UI tools** (we'll walk through both in the solution video)
- If you get stuck creating the schema or tables, **go back and watch the videos again**.
- If you get an error that says 'schema already exists' or 'table already exists', it's because you already did it.
- If you have problems with the data types, see the slide on common data types in the previous lecture. Also, **get comfortable Googling 'MySQL data types'**.

TEST YOUR SKILLS: CREATE SCHEMAS & TABLES

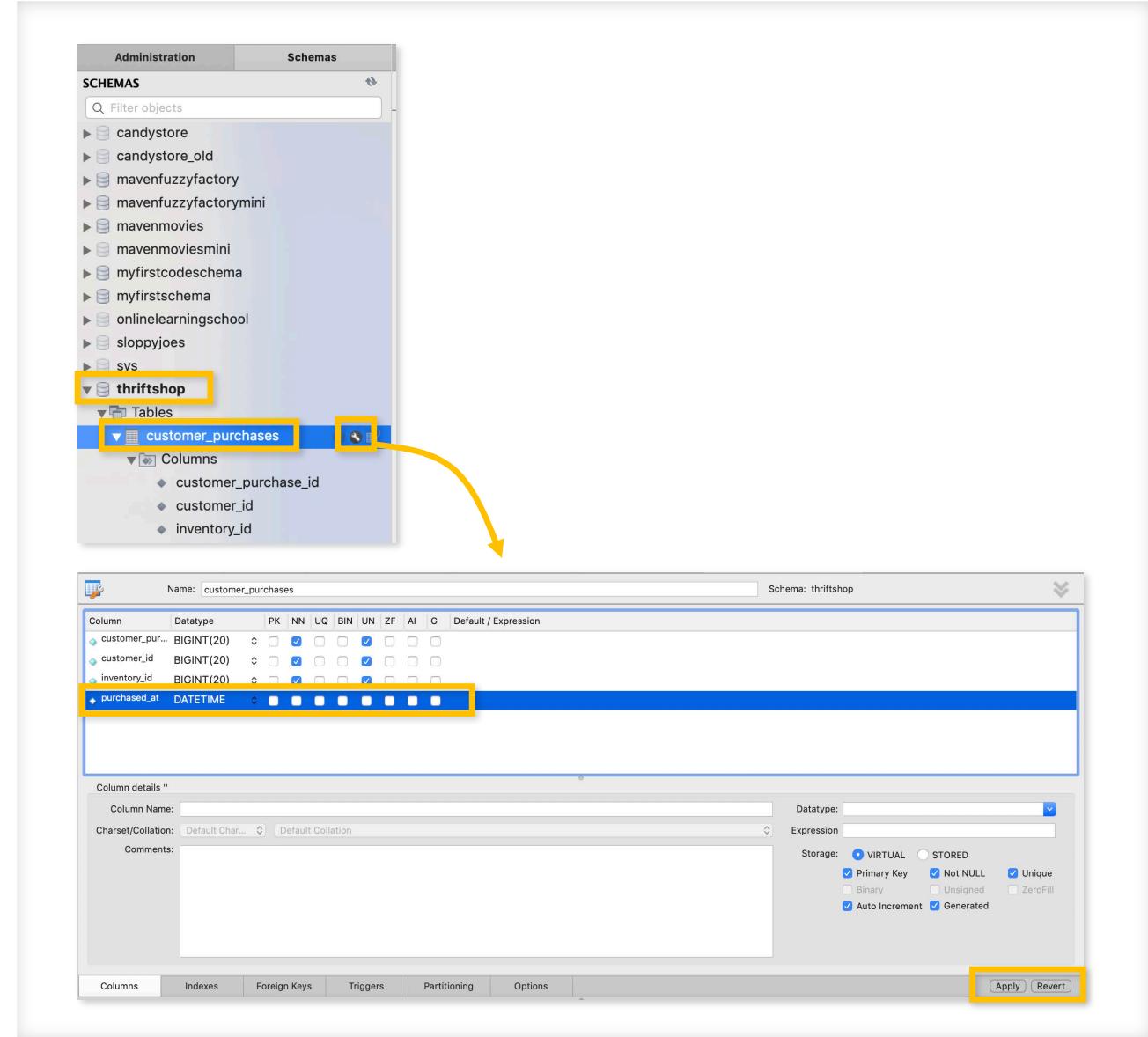
ADD & DROP COLUMNS W/ UI

- In addition to creating new tables which are fully defined, we can also add or drop columns from existing tables

- Just like when we created schemas and tables, altering tables can be done either with the UI tools, or by using SQL code

- Workbench makes it very easy to quickly define or modify the options for your columns in the UI

MySQL WORKBENCH TOOLS IN ACTION:



ALTER TABLE W/ CODE

- We can use SQL code to alter tables by adding and dropping columns

- We specify **ALTER TABLE** tablename, and then we can either **ADD** or **DROP** a column

When we **ADD** a column, we must also specify its data type

- Optionally when we **ADD** a column, we may specify where it should appear, by using **FIRST** or **AFTER**

MySQL QUERY IN ACTION:

```
SELECT * FROM customer_purchases; -- just to see what's here to start
```

```
ALTER TABLE customer_purchases  
DROP COLUMN customer_id; -- maybe this needs to get dropped for privacy
```

```
ALTER TABLE customer_purchases  
ADD COLUMN purchase_amount DECIMAL(6,2) AFTER customer_purchase_id;  
-- after is optional
```

```
SELECT * FROM customer_purchases; -- what's here at the end
```

ACTION OUTPUT:

| customer_purchase_id | purchase_amount | inventory_id | purchased_at |
|----------------------|-----------------|--------------|--------------|
| 1 | NULL | 3 | NULL |
| 2 | NULL | 2 | NULL |
| 3 | NULL | 4 | NULL |
| 4 | NULL | 7 | NULL |
| 5 | NULL | 5 | NULL |
| 6 | NULL | 1 | NULL |
| 7 | NULL | 6 | NULL |
| 8 | NULL | 8 | NULL |
| 9 | NULL | 9 | NULL |
| 10 | NULL | 3 | NULL |
| 11 | NULL | 2 | NULL |
| 12 | NULL | 6 | NULL |



NEW MESSAGE

From: **Amy Baker** (*owns the candy store*)

Subject: **Help Updating A Database Table**

Hi,

I have some basic tables set up in my ‘**candystore**’ schema, but I need some help updating some of the tables before my employees can get access and start using this more widely.

1. Can you **remove the hourly_wage column from the employees table?** (I need it hidden before I can share out)
2. Can you **add a column to the employees table called ‘avg_customer_rating’?** (decimal to one digit)

Thanks!

Reply

Forward

Helpful Tips

If you've done everything correctly, your table should go from this...

| employee_id | first_name | last_name | hire_date | position | hourly_wage |
|-------------|------------|-----------|------------|-------------------|-------------|
| 1 | Jim | John | 2018-01-15 | Manager | 35.00 |
| 2 | Wayland | Smithers | 2018-01-15 | Assistant Manager | 30.00 |
| 3 | Elizabeth | Boop | 2019-02-01 | Clerk | 20.00 |
| 4 | Margaret | Simpson | 2019-02-01 | Clerk | 22.00 |
| 5 | Lawrence | Bird | 2019-12-15 | Clerk | 16.00 |
| 6 | Santos | Halper | 2019-12-15 | Clerk | 15.00 |

To this...

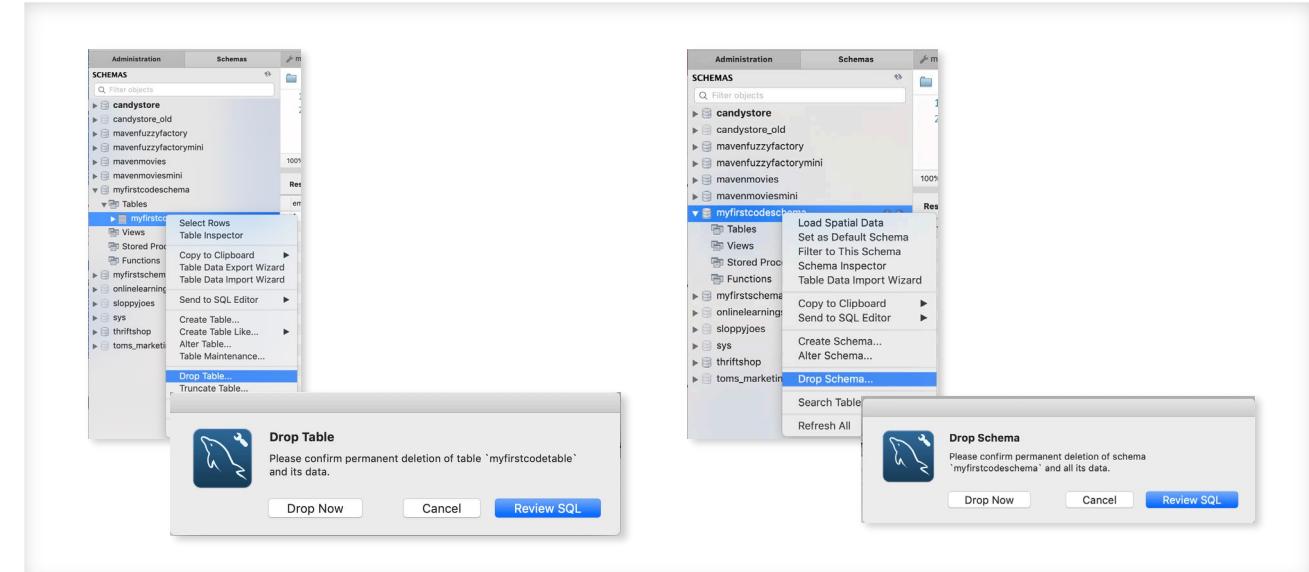
| employee_id | first_name | last_name | hire_date | position | avg_customer_rating |
|-------------|------------|-----------|------------|-------------------|---------------------|
| 1 | Jim | John | 2018-01-15 | Manager | NULL |
| 2 | Wayland | Smithers | 2018-01-15 | Assistant Manager | NULL |
| 3 | Elizabeth | Boop | 2019-02-01 | Clerk | NULL |
| 4 | Margaret | Simpson | 2019-02-01 | Clerk | NULL |
| 5 | Lawrence | Bird | 2019-12-15 | Clerk | NULL |
| 6 | Santos | Halper | 2019-12-15 | Clerk | NULL |

TEST YOUR SKILLS: ALTERING TABLES

DROP SCHEMAS & TABLES

- BE CAREFUL! Dropping schemas and tables that your company needs is a quick way to lose friends at the office 😞
- Ideally, your permissions will be set up so that most employees can access data, some can CREATE tables, and only a limited few can DROP tables or schemas
- Workbench makes it very easy to DROP schemas and tables, through the UI, or with SQL code

MySQL WORKBENCH TOOLS IN ACTION:



MySQL QUERY IN ACTION:

`USE myfirstschema;` -- sets the schema

`DROP TABLE myfirsttable;` -- drops the table

`DROP SCHEMA myfirstschema;` -- drops the schema



NEW MESSAGE

From: **Amy Baker** (*owns the candy store*)
Subject: **Need to Delete a Table and a Schema**

Hi,

We're almost ready to share our '**candystore**' schema for all employees to use. Can you please help me with the following?

1. **Drop the table 'candy_products'**. I don't want my employees knowing my margins.
2. **Drop the schema 'candystore_old'**. This is an old copy that is now out of date, so we don't need to maintain it.

Thanks!

-Amy

Reply

Forward

Warning!!



HEY THIS IS IMPORTANT!

Be extremely careful when dropping tables and columns.

DROP the wrong table or schema at work, and it could be bad news for your relationships and career.

DROP a table you'll need later in this course, and you'll need to go back to the beginning and reinstall. Just don't do it. Be careful!!

TEST YOUR SKILLS: DELETING TABLES & SCHEMAS

INSERTING, UPDATING & DELETING RECORDS

The image shows a screenshot of the phpMyAdmin interface. On the left, there's a navigation menu with icons for databases, users, privileges, and other system components. The main area has three tabs: Structure, Data, and SQL. The SQL tab is active, displaying a query results grid and a query editor.

Tables:

- actor
- actor_info
- address
- category
- city
- country
- customer
- customer_address
- customer_demographic
- customer_returns
- employee
- employee_info
- film
- film_actor
- film_category
- language
- nation
- payment
- rental
- store

Action:

- Browse
- Structure
- Search
- Insert
- Empty
- Drop
- View

Rows:

| Table | Action | Rows | Type | Collation | Size | Overhead |
|----------------------|--------|------|--------|-----------------|---------|----------|
| actor | Browse | 200 | InnoDB | utf8_general_ci | 93.0 kB | - |
| actor_info | Browse | 200 | InnoDB | utf8_general_ci | 94.0 kB | - |
| address | Browse | 200 | InnoDB | utf8_general_ci | 14.0 kB | - |
| category | Browse | 200 | InnoDB | utf8_general_ci | 14.0 kB | - |
| city | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| country | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| customer | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| customer_address | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| customer_demographic | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| customer_returns | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| employee | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| employee_info | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| film | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| film_actor | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| film_category | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| language | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| nation | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| payment | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| rental | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |
| store | Browse | 200 | InnoDB | utf8_general_ci | 23.0 kB | - |

Showing rows 0 - 24 (599 total, Query took 0.0212 seconds.)

```
1 SELECT
2   customer.first_name,
3   customer.last_name,
4   COUNT(DISTINCT rental_id) AS num_rentals
5 FROM
6   customer
7   JOIN rental ON customer.customer_id = rental.customer_id
8   GROUP BY
9     customer.last_name
10    ORDER BY
11      COUNT(DISTINCT rental_id) DESC
```

INSERTING NEW RECORDS

- MySQL makes it easy to insert additional records into a table using an **INSERT INTO** statement

- We can insert one row at a time, or we can batch insert many rows in one single SQL statement

- We may choose to specify values for every column in our **INSERT INTO** statement, or we may omit values for some columns. If we omit values, MySQL places the default value for that column

MySQL QUERY IN ACTION:

```
USE thriftshop; -- sets the schema
```

```
SELECT * FROM inventory; -- just to see what's already there
```

```
INSERT INTO inventory (inventory_id, item_name, number_in_stock) VALUES  
(10, 'wolf skin hat', 1);
```

(Another) MySQL QUERY IN ACTION:

```
-- this time we will batch insert
```

```
-- we will also not name the columns (this is optional)
```

```
INSERT INTO inventory VALUES
```

```
(11, 'fur fox skin', 1),  
(12, 'plaid button up shirt', 8),  
(13, 'flannel zebra jammies', 6);
```

UPDATING RECORDS

- We can update the values of one or more records using an **UPDATE** statement

- Our **UPDATE** statement will always require a **SET** clause, which tells the server what values to set

- In most cases, we will also use a **WHERE** clause (optional) to specify which records to update. If we do not use a **WHERE** clause, the **UPDATE** will process on all records

MySQL QUERY IN ACTION:

```
UPDATE inventory
```

```
SET number_in_stock = 0 -- these are sold out
```

```
WHERE inventory_id = 9; -- only do this for id #9
```

(ANOTHER) MySQL QUERY IN ACTION:

```
-- this time we'll do it for two records simultaneously!
```

```
UPDATE inventory
```

```
SET number_in_stock = 0 -- these are sold out
```

```
WHERE inventory_id IN(1,9); -- update 2 records at once
```



NEW MESSAGE

From: **Amy Baker** (*owns the candy store*)
Subject: **Help Updating Some Data Records**

Hi,

Exciting news! I have hired two new employees, who both started on March 15th, 2020. They are both clerks. Their names are **Charles Munger** and **William Gates**. Could you add them to the employees table for me?

Could you also update the **avg_customer_rating** column? The average ratings for employees 1-6 are **4.8**, **4.6**, **4.75**, **4.9**, **4.75**, and **4.2**, respectively. The two new clerks are both at **5.0**.

Thank you so much!

Reply

Forward

Result Preview

| employee_id | first_name | last_name | hire_date | position | avg_customer_rating |
|-------------|------------|-----------|------------|-------------------|---------------------|
| 1 | Jim | John | 2018-01-15 | Manager | 4.80 |
| 2 | Wayland | Smithers | 2018-01-15 | Assistant Manager | 4.60 |
| 3 | Elizabeth | Boop | 2019-02-01 | Clerk | 4.75 |
| 4 | Margaret | Simpson | 2019-02-01 | Clerk | 4.90 |
| 5 | Lawrence | Bird | 2019-12-15 | Clerk | 4.75 |
| 6 | Santos | Halper | 2019-12-15 | Clerk | 4.20 |
| 7 | Charles | Munger | 2020-03-15 | Clerk | 5.00 |
| 8 | William | Gates | 2020-03-15 | Clerk | 5.00 |

TEST YOUR SKILLS: INSERTING & UPDATING RECORDS

DELETE RECORDS

- MySQL allows us to target one or more records to **DELETE FROM** a given data table
- We will do this using the **DELETE FROM** statement, followed by the tablename
- As with **UPDATE** statements, we will usually want to specify which records to delete using a **WHERE** clause
- If we do not specify a **WHERE** clause, **DELETE FROM** will delete ALL records

MySQL QUERY IN ACTION:

```
-- deleting ski blankets, no longer stocked  
DELETE FROM inventory  
WHERE inventory_id = 7
```

ACTION OUTPUT:

| inventory_id | item_name | number_in_stock |
|--------------|-----------------------|-----------------|
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 12 |
| 4 | house slippers | 6 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 7 | ski blanket | 1 |
| 8 | kneeboard | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |

| inventory_id | item_name | number_in_stock |
|--------------|-----------------------|-----------------|
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 12 |
| 4 | house slippers | 6 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 7 | ski blanket | 1 |
| 8 | kneeboard | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |

TRUNCATE TABLE

- If we want to remove all records from a table but preserve the table structure, we can do that using **TRUNCATE TABLE**
- When we use **TRUNCATE TABLE**, the data is removed but the column names, data types, column order, and any constraints placed on the table are all preserved
- **TRUNCATE TABLE** is very similar to using **DELETE** without a **WHERE** clause (but there are differences)

MySQL QUERY IN ACTION:

```
USE thriftshop;  
-- just to see what's in there  
SELECT *  
FROM customers;
```

```
-- truncating the table  
TRUNCATE TABLE customers;
```

```
-- now, seeing what's there once more  
SELECT *  
FROM customers;
```

ACTION OUTPUT:

| customer_id | first_name | last_name |
|-------------|------------|-----------|
| 1 | Mack | Lewis |
| 2 | Ryan | More |
| 3 | Brooklyn | Haggerty |
| 4 | Tim | Grinnell |
| 5 | Ray | Bridwell |
| 6 | Ben | Dalton |

| customer_id | first_name | last_name |
|-------------|------------|-----------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |



DDL, DML, DQL, DCL & DTL

DATA DEFINITION LANGUAGE (DDL)

Used to create or modify the structure of a database
Examples: **CREATE, ALTER, DROP, TRUNCATE**

DATA MANIPULATION LANGUAGE (DML)

Used to add, modify, or delete data records
Examples: **INSERT, UPDATE, DELETE**

DATA QUERY LANGUAGE (DQL)

Used to query data (*often considered part of DML*)
Examples: **SELECT, SHOW, HELP**

DATA CONTROL LANGUAGE (DCL)

Used to grant and revoke permissions
Examples: **GRANT, REVOKE**

DATA TRANSACTION LANGUAGE (DTL)

Used to manage transactions
Examples: **START TRANSACTION, COMMIT, ROLLBACK**



NEW MESSAGE

From: **Amy Baker** (*owns the candy store*)

Subject: **Help Deleting Some Data**

Hi,

I have a situation I need your help with. One of my employees, **Margaret Simpson**, was recently caught generating fake customer reviews to boost her score. I would like you to...

- 1. Delete her record from the employees table**
- 2. Remove all data from the customer reviews table but preserve the table structure** (we want to start fresh)

Thank you!

-Amy

Reply

Forward

Helpful Hints

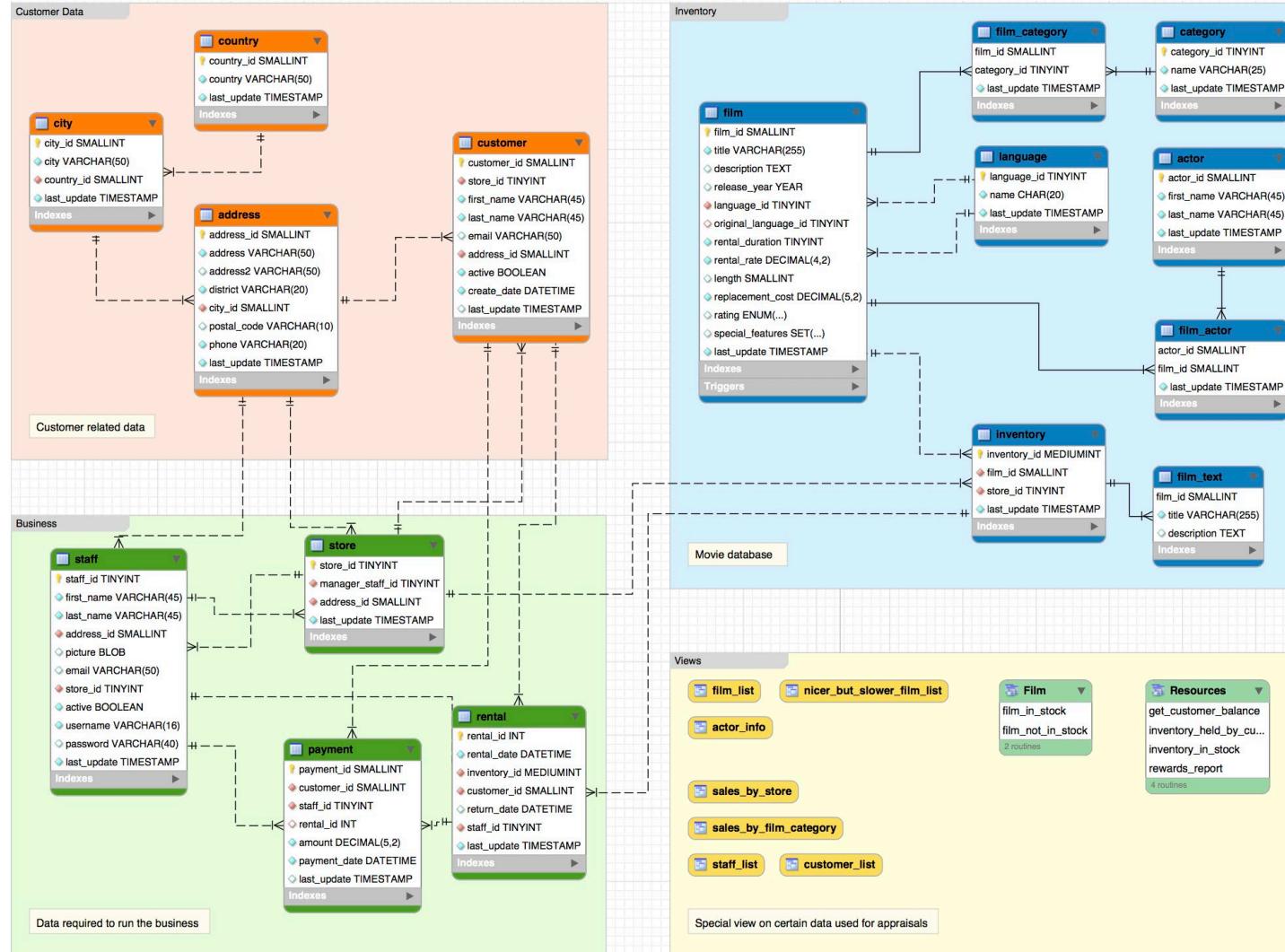
- Make sure you do not drop either of the two tables. You just want to remove records for this exercise, without impacting the table structure
- For the second part of Amy's ask, you should end up with a table containing column headers but no data records

TEST YOUR SKILLS: DELETING DATA



DATABASE DESIGN

DATABASE DESIGN IS ABOUT STRUCTURING TABLES & RELATIONSHIPS



In this case the database includes **16 related tables**, containing information about:

- **Customers** (Name, Address, etc.)
- **Business** (Staff, Rentals, etc.)
- **Inventory** (Films, Categories, etc.)

Now we are ready to start talking about optimal database design, how tables relate to each other, and how to properly normalize a database.

EACH TABLE CONTAINS ROWS & COLUMNS

| rental | |
|---------|------------------------|
| ! | rental_id INT |
| ◆ | rental_date DATETIME |
| ◆ | inventory_id MEDIUMINT |
| ◆ | customer_id SMALLINT |
| ◆ | return_date DATETIME |
| ◆ | staff_id TINYINT |
| ◆ | last_update TIMESTAMP |
| Indexes | |

Tables contain information organized into **columns (or fields)** and **rows (or records)**

In this case, our **Rental** table contains **7 columns** and **10 rows**:

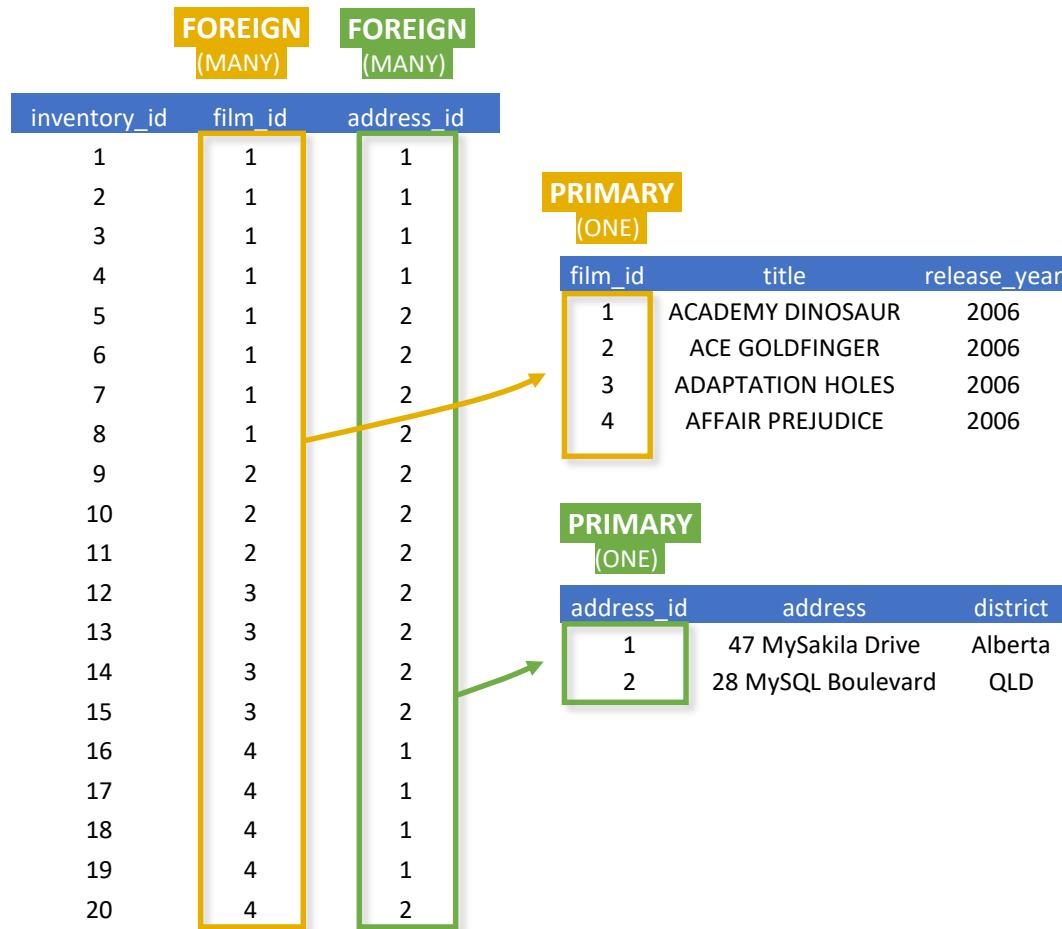
- Each **column** contains an attribute related to our film rentals (*rental/return date, customer ID, etc.*)
- Each **row** corresponds to **one specific rental** (*which film was rented, when, who rented it, etc.*)



| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-26 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-06-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:04:41 | 2452 | 333 | 2005-06-03 01:43:41 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 2079 | 222 | 2005-06-02 04:33:21 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:08:07 | 2792 | 549 | 2005-05-27 01:32:07 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:11:53 | 3995 | 269 | 2005-05-29 20:34:53 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 126 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |
| 10 | 2005-05-25 00:02:21 | 1824 | 399 | 2005-05-31 22:44:21 | 2 | 2006-02-15 21:30:53 |

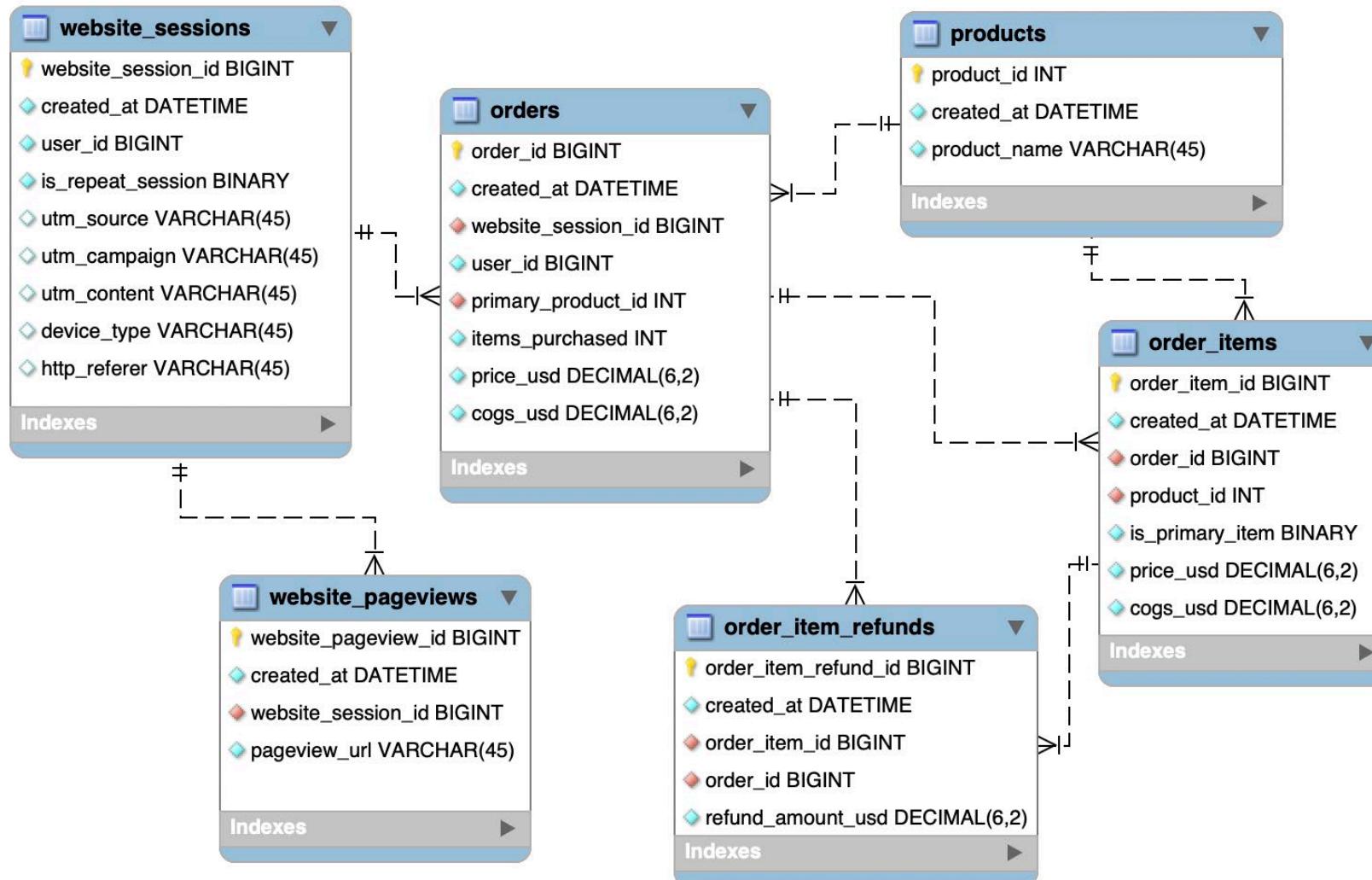
TABLE RELATIONSHIPS & CARDINALITY

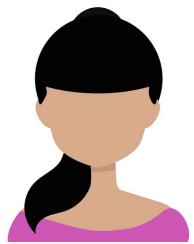
Cardinality refers to the **uniqueness of values** in a column (or attribute) of a table and is commonly used to describe how two tables relate (**one-to-one**, **one-to-many**, or **many-to-many**). For now, here are the key points to grasp:



- **Primary keys are unique**
 - They **cannot** repeat, so there is **only one** instance of each primary key value in a column
- **Foreign keys are non-unique**
 - They **can** repeat, so there may be **many** instances of each foreign key value in a column
- We can create a **one-to-many** relationship by connecting a **foreign key** in one table to a **primary key** in another

MAVEN FUZZY FACTORY DATABASE





NEW MESSAGE

From: **Emily Trapp (Owner, Online School)**

Subject: **Help Understanding My Tables**

Hey!

Would you be able to help me out? I have 3 tables in my 'onlinelearningschool' schema and would like to understand how they are related. For each table, could you please:

- 1) Tell me what the primary key is, and whether there are any foreign keys**
- 2) Explain how the tables relate to each other and what type of relationship exists (one-to-one, one-to-many, etc.)**

Thank you!

Reply

Forward

Helpful Hint

-- run these simple queries to see what's in there

```
USE onlinelearningschool;
```

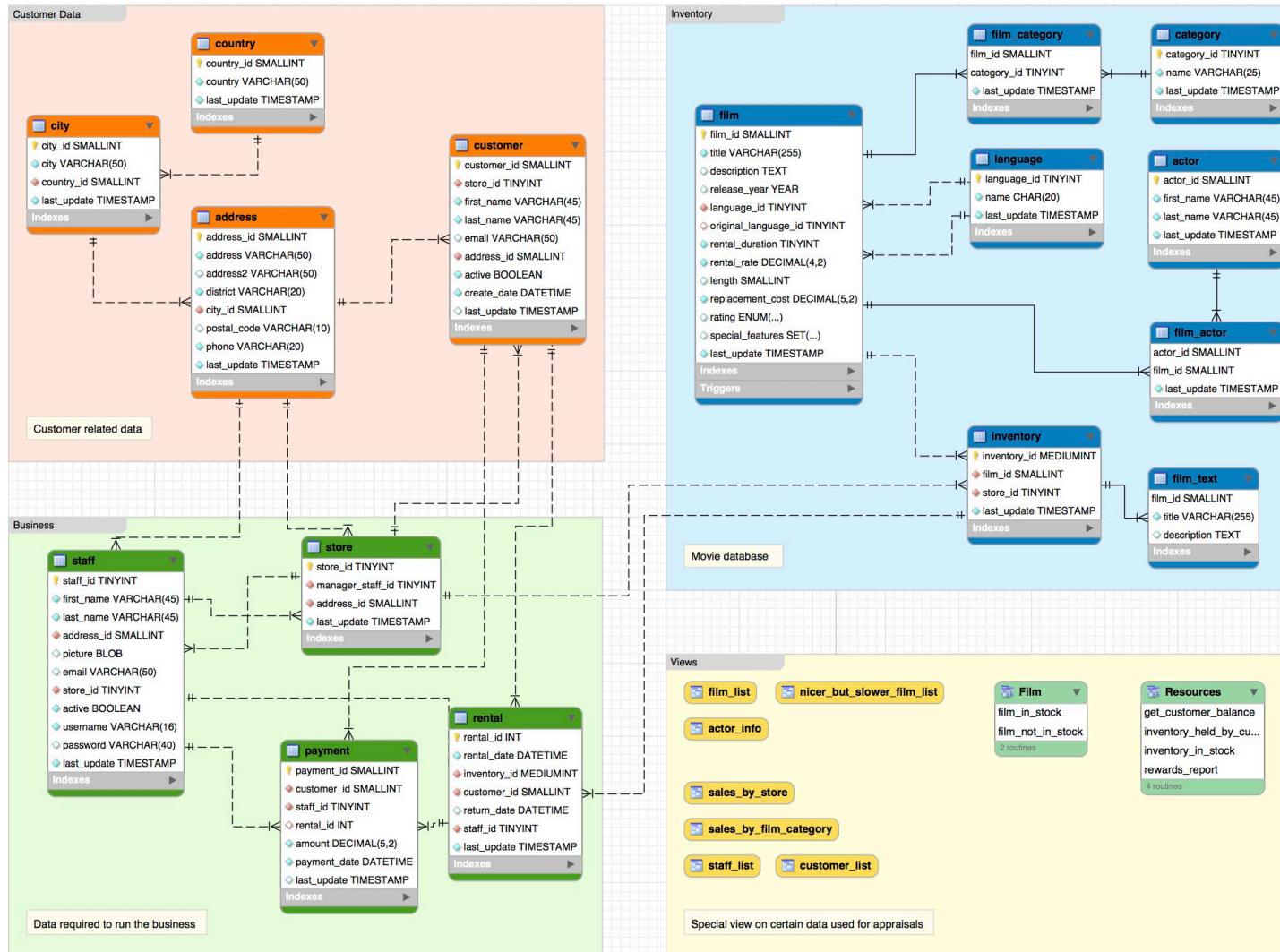
```
SELECT * FROM courses;
```

```
SELECT * FROM course_ratings;
```

```
SELECT * FROM course_ratings_summaries;
```

TEST YOUR SKILLS: UNDERSTANDING RELATIONSHIPS

A DATABASE CAN CONTAIN MANY RELATED TABLES



In this case the database includes **16 related tables**, containing information about:

- **Customers** (Name, Address, etc.)
- **Business** (Staff, Rentals, etc.)
- **Inventory** (Films, Categories, etc.)

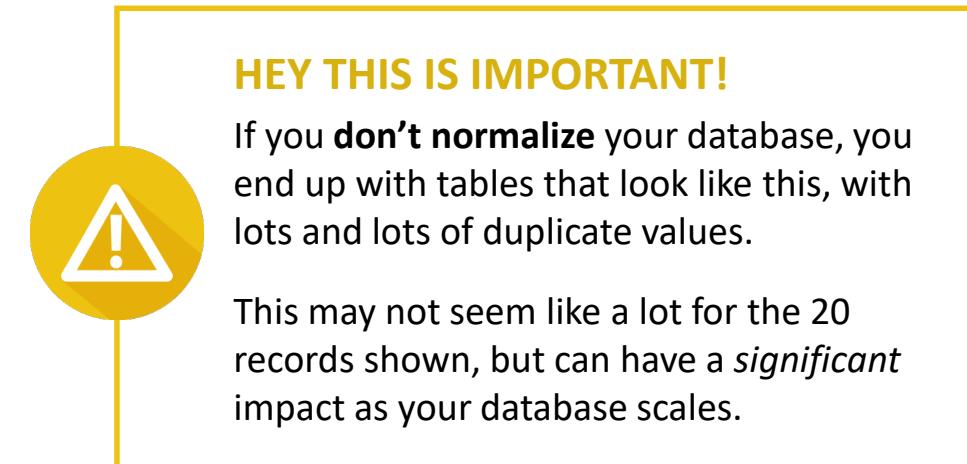
We'll start by using MySQL to create and modify **individual tables**, then later in the course we will get into **database design** and discuss optimal table relationships.

DATABASE NORMALIZATION

Normalization is the process of structuring the tables and columns in a relational database to **minimize redundancy** and **preserve data integrity**. Benefits of normalization include:

- **Eliminating duplicate data** (*this makes storage and query processing more efficient*)
- **Reducing errors and anomalies** (*restrictions around data structure help to prevent human errors*)

| inventory_id | title | release_year | store_address | store_district |
|--------------|------------------|--------------|--------------------|----------------|
| 1 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 2 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 3 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 4 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 5 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 6 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 7 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 8 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 9 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 10 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 11 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 12 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 13 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 14 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 15 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 16 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 17 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 18 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 19 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 20 | AFFAIR PREJUDICE | 2006 | 28 MySQL Boulevard | QLD |



NORMALIZATION: MULTIPLE RELATED TABLES

In practice, normalization involves breaking out data from a **single merged table** into **multiple related tables**

- Instead of storing redundant information about each store and film in a single table (*like the one on the left*), we create **new tables containing a *single record* for each unique value**, and link to those tables using a simple id



Not normalized:

| inventory_id | title | release_year | store_address | store_district |
|--------------|------------------|--------------|--------------------|----------------|
| 1 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 2 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 3 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 4 | ACADEMY DINOSAUR | 2006 | 47 MySakila Drive | Alberta |
| 5 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 6 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 7 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 8 | ACADEMY DINOSAUR | 2006 | 28 MySQL Boulevard | QLD |
| 9 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 10 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 11 | ACE GOLDFINGER | 2006 | 28 MySQL Boulevard | QLD |
| 12 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 13 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 14 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 15 | ADAPTATION HOLES | 2006 | 28 MySQL Boulevard | QLD |
| 16 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 17 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 18 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 19 | AFFAIR PREJUDICE | 2006 | 47 MySakila Drive | Alberta |
| 20 | AFFAIR PREJUDICE | 2006 | 28 MySQL Boulevard | QLD |



Normalized!

| inventory_id | film_id | address_id |
|--------------|---------|------------|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 1 |
| 4 | 1 | 1 |
| 5 | 1 | 2 |
| 6 | 1 | 2 |
| 7 | 1 | 2 |
| 8 | 1 | 2 |
| 9 | 2 | 2 |
| 10 | 2 | 2 |
| 11 | 2 | 2 |
| 12 | 3 | 2 |
| 13 | 3 | 2 |
| 14 | 3 | 2 |
| 15 | 3 | 2 |
| 16 | 4 | 1 |
| 17 | 4 | 1 |
| 18 | 4 | 1 |
| 19 | 4 | 1 |
| 20 | 4 | 2 |

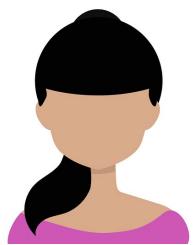
film_id title release_year

| | | |
|---|------------------|------|
| 1 | ACADEMY DINOSAUR | 2006 |
| 2 | ACE GOLDFINGER | 2006 |
| 3 | ADAPTATION HOLES | 2006 |
| 4 | AFFAIR PREJUDICE | 2006 |

address_id address district

| | | |
|---|--------------------|---------|
| 1 | 47 MySakila Drive | Alberta |
| 2 | 28 MySQL Boulevard | QLD |

We now have **single records** containing all of the information about our films and addresses – no more redundancy!



NEW MESSAGE

From: **Emily Trapp (Owner, Online School)**

Subject: **Help Understanding My Tables**

Hey!

I could use some help again. I heard about this concept of database “normalization”, but need your expertise.

Would you be able to look at my three tables and tell me if they are already normalized?

If any table is NOT normalized, **could you give me direction on how I could restructure the data to normalize my database?**

Thank you!

Reply

Forward

Helpful Hint

-- run these simple queries to see what's in there

```
USE onlinelearningschool;
```

```
SELECT * FROM courses;
```

```
SELECT * FROM course_ratings;
```

```
SELECT * FROM course_ratings_summaries;
```

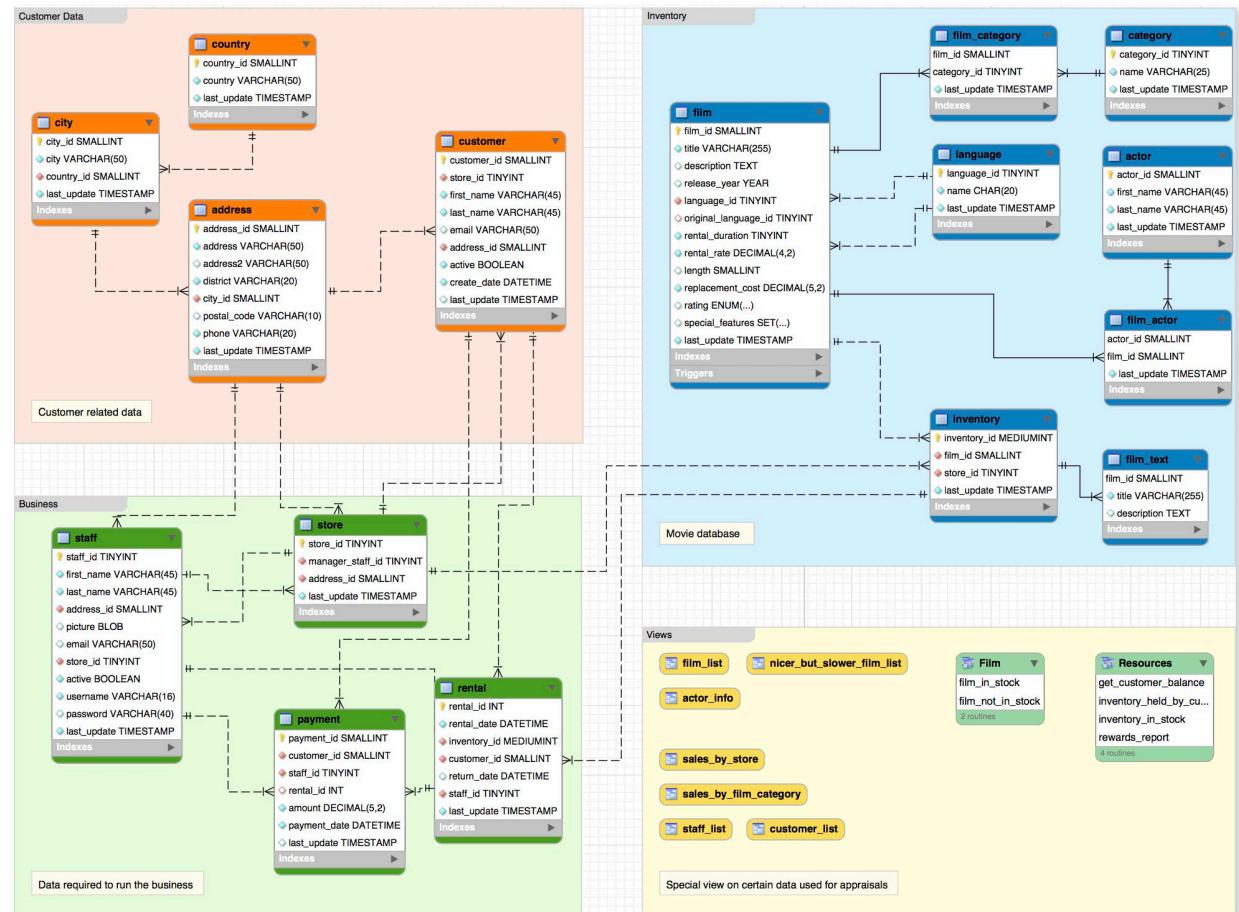
TEST YOUR SKILLS: DATABASE NORMALIZATION

ENHANCED ENTITY RELATIONSHIP (EER) MODELS

As we designing our databases, it can be helpful to create **EER diagrams** to visually model the relationships between tables and the constraints within those tables

EER diagrams can help us map out things like:

- Which tables are in the database
- Which columns exist in each table
- The data types of the various columns
- Primary and foreign keys within tables
- Relationship cardinality between tables
- Constraints on columns (i.e. Non-NULL)



MINI EXAMPLE: RELATIONSHIP DIAGRAMS

Consider the two tables shown below:

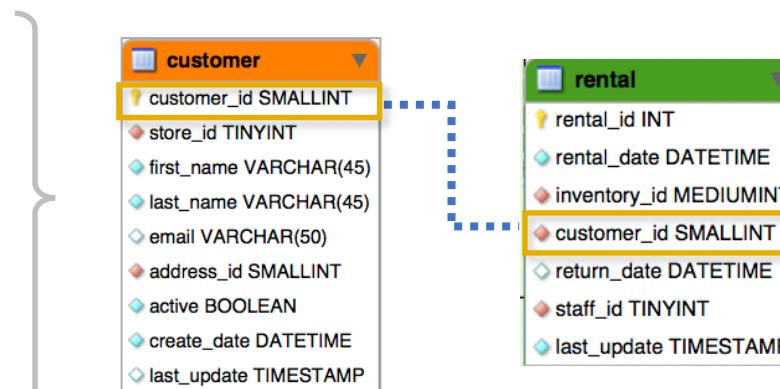
- The **Customer** table below contains details about each customer, identified by a unique **customer_id** (*the table's primary key*)
- The **Rental** table contains records of each rental, and includes a non-unique **customer_id** field since customers may rent films on multiple occasions (*this is one of the table's foreign keys*)

Customer table:

| customer_id | store_id | first_name | last_name | email | address_id | active | create_date |
|-------------|----------|------------|-----------|-------------------------------------|------------|--------|---------------------|
| 1 | 1 | MARY | SMITH | MARY.SMITH@sakilacustomer.org | 5 | 1 | 2006-02-14 22:04:36 |
| 2 | 1 | PATRICIA | JOHNSON | PATRICIA.JOHNSON@sakilacustomer.org | 6 | 1 | 2006-02-14 22:04:36 |
| 3 | 1 | LINDA | WILLIAMS | LINDA.WILLIAMS@sakilacustomer.org | 7 | 1 | 2006-02-14 22:04:36 |
| 4 | 2 | BARBARA | JONES | BARBARA.JONES@sakilacustomer.org | 8 | 1 | 2006-02-14 22:04:36 |
| 5 | 1 | ELIZABETH | BROWN | ELIZABETH.BROWN@sakilacustomer.org | 9 | 1 | 2006-02-14 22:04:36 |
| 6 | 2 | JENNIFER | DAVIS | JENNIFER.DAVIS@sakilacustomer.org | 10 | 1 | 2006-02-14 22:04:36 |
| 7 | 1 | MARIA | MILLER | MARIA.MILLER@sakilacustomer.org | 11 | 1 | 2006-02-14 22:04:36 |
| 8 | 2 | SUSAN | WILSON | SUSAN.WILSON@sakilacustomer.org | 12 | 1 | 2006-02-14 22:04:36 |
| 9 | 2 | MARGARET | MOORE | MARGARET.MOORE@sakilacustomer.org | 13 | 1 | 2006-02-14 22:04:36 |

Rental table:

| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-26 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-06-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:04:41 | 2452 | 333 | 2005-06-03 01:43:41 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 2079 | 222 | 2005-06-02 04:33:21 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:08:07 | 2792 | 549 | 2005-05-27 01:32:07 | 1 | 2006-02-15 21:30:53 |
| 7 | 2005-05-24 23:11:53 | 3995 | 269 | 2005-05-29 20:34:53 | 2 | 2006-02-15 21:30:53 |
| 8 | 2005-05-24 23:31:46 | 2346 | 239 | 2005-05-27 23:33:46 | 2 | 2006-02-15 21:30:53 |
| 9 | 2005-05-25 00:00:40 | 2580 | 126 | 2005-05-28 00:22:40 | 1 | 2006-02-15 21:30:53 |



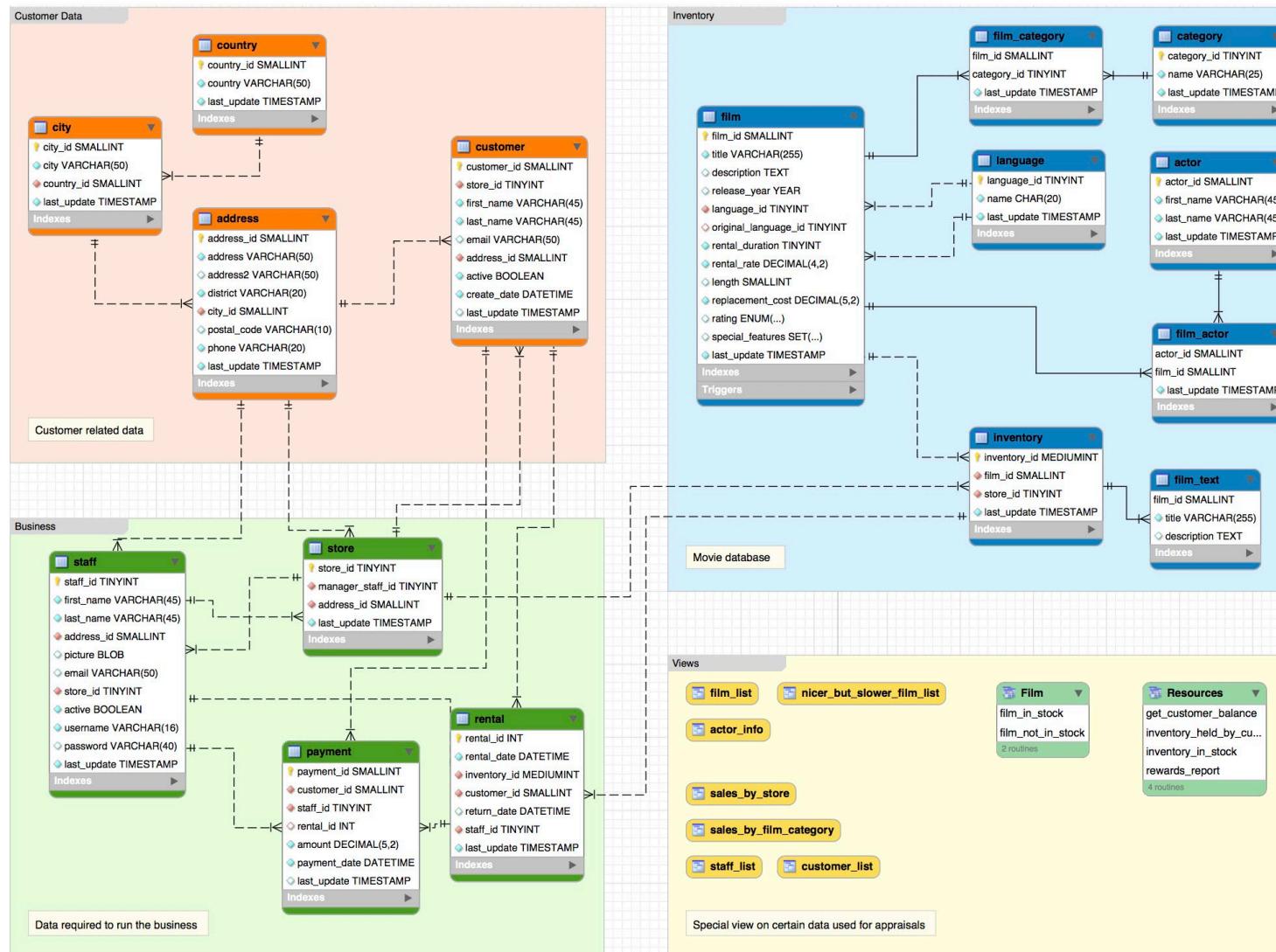
We can diagram table relationships in Workbench to understand how the records in each table relate

NOTE: This isn't required, but serves as a helpful reference

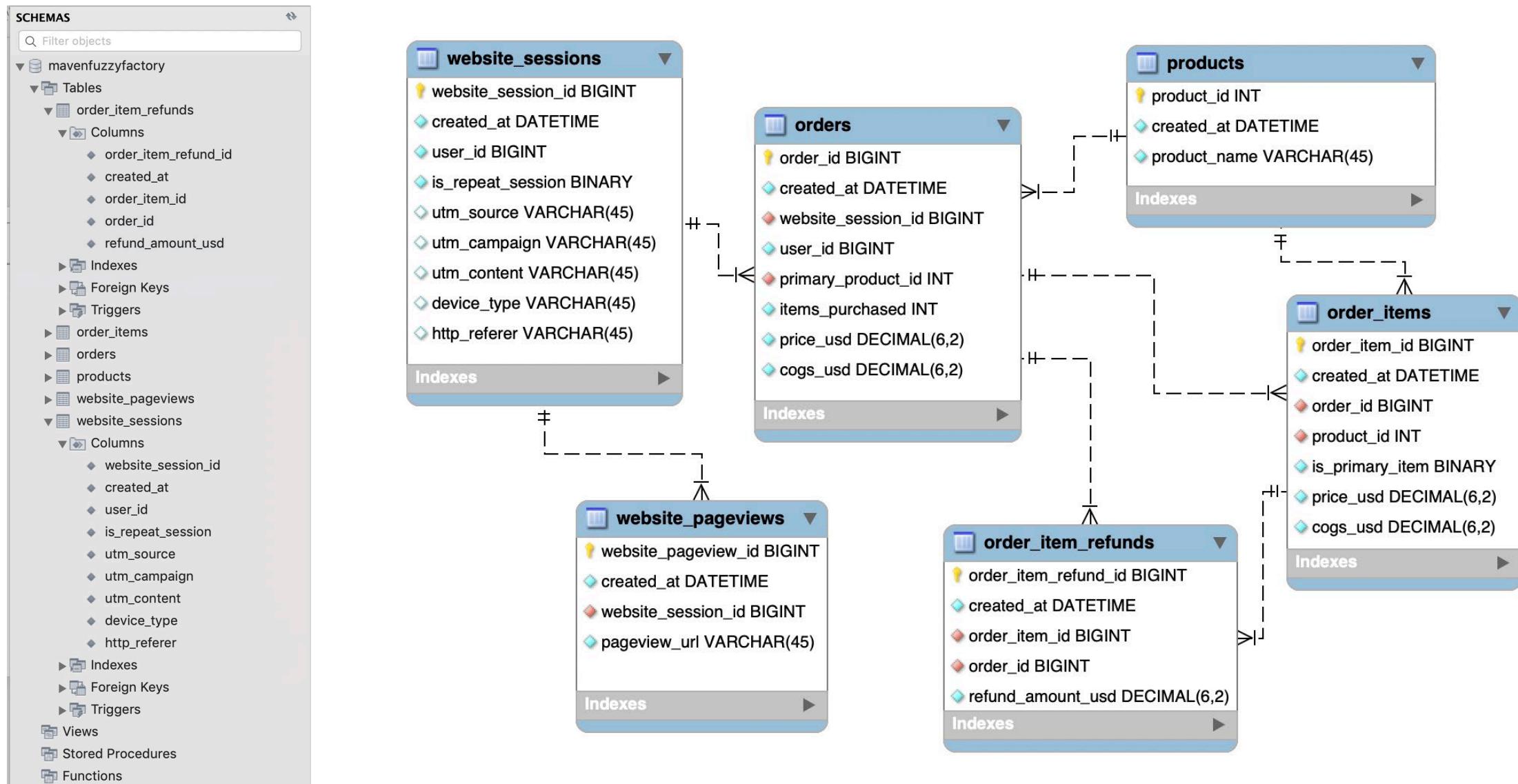
PRO TIP:

When you explore a database for the first time, **diagram your relationships** to understand your table structure

EER DIAGRAM EXAMPLE: MAVEN MOVIES DATABASE

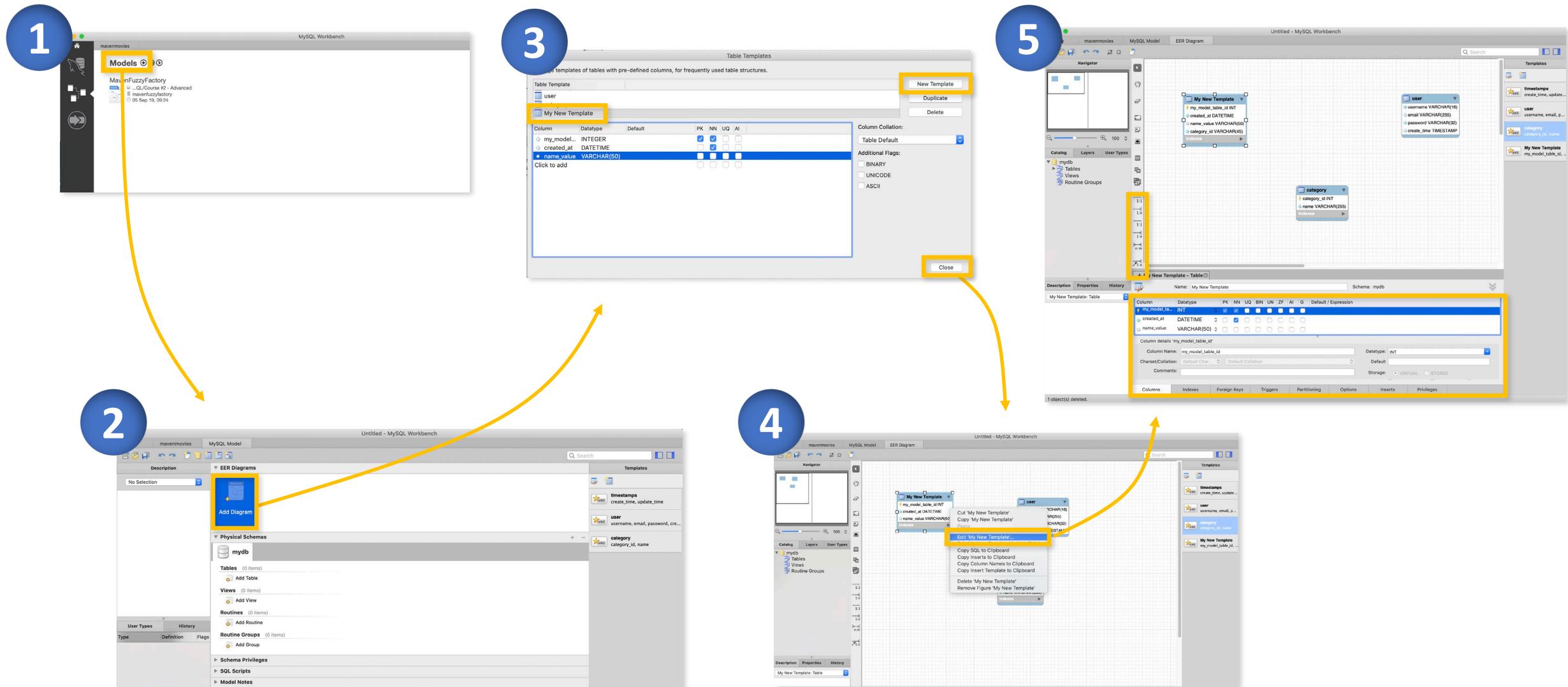


ANOTHER EXAMPLE: MAVEN FUZZY FACTORY DATABASE



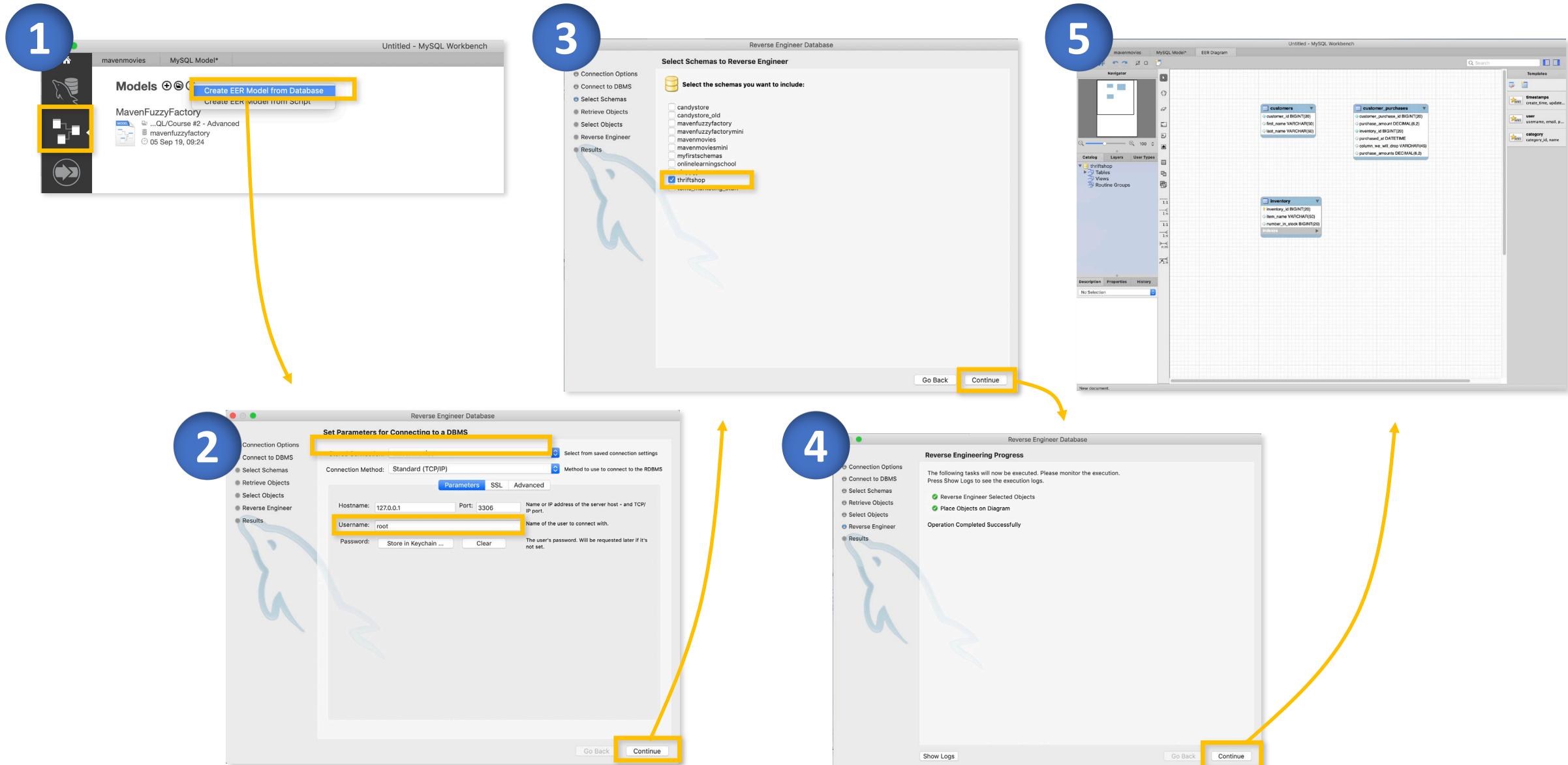


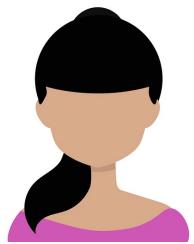
CREATE AN EER DIAGRAM FROM SCRATCH





CREATE AN EER DIAGRAM FROM EXISTING SCHEMA





NEW MESSAGE

From: **Emily Trapp (Owner, Online School)**

Subject: **Help Creating an EER Diagram**

Hey again!

I would like to have a better visual representation of the database so I can share it with our new employees as we start to scale.

Could you please **build an EER diagram of the onlinelearningschool schema?** If you could map the relationships that would be awesome.

Thank you!

-Emily

Reply

Forward

Helpful Hint

-- run these simple queries to see what's in there

```
USE onlinelearningschool;
```

```
SELECT * FROM courses;
```

```
SELECT * FROM course_ratings;
```

```
SELECT * FROM course_ratings_summaries;
```

TEST YOUR SKILLS: BUILDING EER DIAGRAMS

INTRODUCING THE MID COURSE PROJECT

THE **SITUATION**

A new client, the owner of the Maven Movies DVD rental business, has reached out to you for help restructuring their non-normalized database.

THE **OBJECTIVE**

Use Your MySQL Database Administration skills to:

Design a better set of tables to store the data in the existing schema. Explain to the owner why the current system is not optimized for scale, and how you propose to improve it. Then, **create a new schema with your ideal specifications and populate it with data.**

MID COURSE PROJECT QUESTIONS

1

Take a look at the *mavenmoviesmini* schema. What do you notice about it? How many tables are there? What does the data represent? What do you think the current schema?

~ 1:10

2

If you wanted to break out the data from the *inventory_non_normalized* table into multiple tables, how many tables do you think would be ideal? What would you name those tables?

~ 4:30

3

Based on your answer from question #2, create a new schema with the tables you think will best serve this data set. You can use SQL code or Workbench's UI tools (whichever you feel more comfortable with).

~ 8:10

MID COURSE PROJECT QUESTIONS

4

- Next, use the data from the original schema to populate the tables in your newly optimized schema**
(TIP: Revisit the video on database normalization again if you get stuck)

~ 17:45

5

- Make sure your new tables have the proper primary keys defined and that applicable foreign keys are added.**
Add any constraints you think should apply to the data as well (*unique, non-NUL*, etc.)

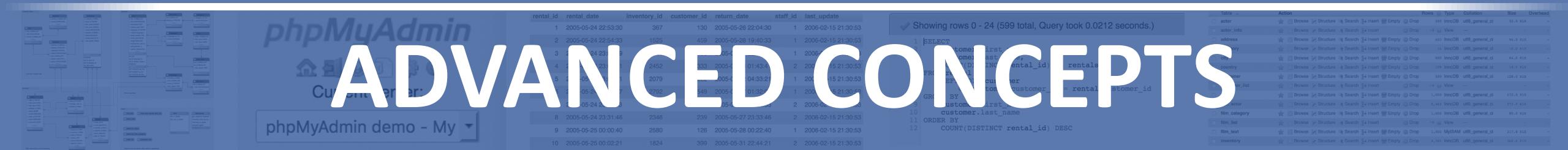
~ 24:50

6

- Finally, after doing all of this technical work, write a brief summary of what you have done, in a way that your non-technical client can understand. Communicate what you did, and why your new schema design is better.**

~ 26:05

ADVANCED CONCEPTS



USING INDEXES

From MySQL.com:

The best way to improve the performance of SELECT operations is to create indexes on one or more of the columns that are tested in the query. The index entries act like pointers to the table rows, allowing the query to quickly determine which rows match a condition in the WHERE clause, and retrieve the other column values for those rows. All MySQL data types can be indexed.

Searching Without an Index

| inventory_id | item_name | number_in_stock |
|--------------|-----------------------|-----------------|
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 12 |
| 4 | house slippers | 6 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 7 | ski blanket | 1 |
| 8 | kneeborder | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |

Using an Index

| inventory_id | item_name | number_in_stock |
|--------------|-----------------------|-----------------|
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 12 |
| 4 | house slippers | 6 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 7 | ski blanket | 1 |
| 8 | kneeborder | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |

CREATING AN INDEX

- Workbench makes it easy to manage indexes via the UI tools
- We can add, and delete indexes through the UI, or we can do it with SQL code

• Note that in MySQL, the Primary Key and any Foreign Key you create on a table is always an index automatically

MySQL Workbench in Action

The screenshot shows the MySQL Workbench interface with the 'customer_reviews' table selected. The primary key is defined with three columns: 'customer_review_id', 'employee_id', and 'customer_rating'. A green arrow points from the text 'Note that in MySQL, the Primary Key and any Foreign Key you create on a table is always an index automatically' to the 'Indexes' tab at the bottom of the table configuration window.

customer_reviews - Table

Name: customer_reviews Schema: candystore

Index Type

PRIMARY PRIMARY

customer_review_id employee_id customer_rating

Index details 'PRIMARY'

Index Columns # Order Length

customer_review_id 1 ASC

employee_id ASC

customer_rating ASC

Storage Type: Key Block Size: 0 Parser: Visibility: Comments:

Indexes

Action Output

Time Action Response Duration

1 15:34:40 SELECT * FROM category LIMIT 0, 50000 0.015 se

2 16:53:30 SELECT * FROM rental LIMIT 0, 50000 0.00050

3 17:44:54 SELECT * FROM rental LIMIT 0, 50000 0.015 se

Showing rows 0 - 24 (

1 SELECT

2 customer.first

3 customer.last

4 COUNT(DISTINCT

5 FROM rental

6 LEFT JOIN customer

7 ON customer.customer_id = rental.customer_id

8 GROUP BY

9 customer.first_name,

10 customer.last_name

11 ORDER BY

Copyright Maven Analytics, LLC

UNIQUE CONSTRAINT

- We can prescribe which columns in our tables we allow to have repeating values, and which columns must be unique

- Adding a unique constraint is easy via Workbench's UI or using SQL code

- Including a unique constraint on columns that should not repeat is a good way to ensure data integrity, and is a good best practice

MySQL Workbench in Action

The screenshot shows the MySQL Workbench interface for managing the 'customer_reviews' table in the 'candystore' schema. The main pane displays the table structure with three columns: 'customer_rev...', 'employee_id', and 'customer_rati...'. The 'customer_rev...' column is defined as BIGINT(20) and has the primary key (PK) and unique (UQ) constraints selected. The 'Indexes' tab at the bottom is highlighted with a green circle, indicating where a unique constraint would be applied.

Customer reviews data

Business

Customer review data

Movie database

Views

address

category

country

customer_reviews - Table

Name: customer_reviews Schema: candystore

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|------------------|------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|----------------------|
| customer_rev... | BIGINT(20) | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| employee_id | BIGINT(20) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| customer_rati... | BIGINT(20) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

<click to edit>

Column details 'customer_review_id'

Column Name: customer_review_id Datatype: BIGINT(20)

Charset/Collation: Default Char... Default Collation

Comments:

Storage: VIRTUAL STORED

Primary Key Not NULL Unique
 Binary Unsigned ZeroFill
 Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Action Output

Time Action Response Duration

1 15:34:40 SELECT * FROM rental LIMIT 0, 50000
16044 row(s) returned
0.015 se
2 16:53:30 SELECT * FROM category LIMIT 0, 50000
16 row(s) returned
0.00050
3 17:44:54 SELECT * FROM rental LIMIT 0, 50000
Showing rows 0 - 24 (16044 total)
SELECT COUNT(DISTINCT rental_id) AS rental_count, COUNT(DISTINCT customer_id) AS customer_count, COUNT(DISTINCT category_id) AS category_count
FROM rental
LEFT JOIN customer
ON customer.customer_id = rental.customer_id
GROUP BY
customer.first_name,
customer.last_name
ORDER BY

Time Action Response Duration / Fetch Time

*Copyright Maven Analytics, LLC

NON-NULL CONSTRAINT

- We may want certain columns to be populated with a value for every single record that ever gets written
- We can prescribe this by applying a **NON-NULL** constraint to a certain column, which requires all records to have a value in that column
- If the **NON-NULL** constraint is on a column, and someone tries to **INSERT** a record without including a value for the column, the **INSERT** will fail

MySQL Workbench in Action

The screenshot shows the MySQL Workbench interface with the 'customer_reviews' table selected. The main pane displays the table structure with three columns: customer_review_id (BIGINT(20)), employee_id (BIGINT(20)), and customer_rating (BIGINT(20)). The 'customer_review_id' column has the 'Not Null' (NN) constraint checked. A green arrow points from this checked box in the main table list to the corresponding checked box in the detailed 'Column details' panel below. The 'Customer reviews' table is defined in the 'candystore' schema.

| Column | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|------------------|------------|----|-------------------------------------|-------------------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|----------------------|
| customer_rev... | BIGINT(20) | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| employee_id | BIGINT(20) | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| customer_rati... | BIGINT(20) | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Column details 'customer_review_id'

Column Name: customer_review_id Datatype: BIGINT(20)
Charset/Collation: Default Char... Default Collation
Comments:
Storage: VIRTUAL STORED
Primary Key: Not NULL:
Binary: Unsigned:
Auto Increment: ZeroFill:

Columns Indexes Foreign Keys Triggers Partitioning Options



NEW MESSAGE

From: **Joe Tidyman** (Owner, Sloppy Joe's)
Subject: Help Optimizing Database for Scale

Hi there...

We are about to scale up our operations and open up a very large number of restaurants in a short amount of time.

Could you help me set up our database so it can scale with data integrity? I think we'll need to set up keys, indexes, and constraints to make sure the data quality is protected as we ramp up and the data sets get huge.

Can you help!?!?

-Joe

Reply

Forward

Helpful Hint

-- run these queries to see what's in there

`USE sloppyjoes;`

`SELECT * FROM customer_orders ;`

`SELECT * FROM menu_items;`

`SELECT * FROM staff;`

TEST YOUR SKILLS: INDEXES, KEYS, CONSTRAINTS

STORED PROCEDURES

- MySQL gives us the ability to store and call frequently used queries on the server. These are then referred to as ‘Procedures’, or commonly ‘Stored Procedures’

- Benefits include more efficient query writing and query performance, and ability to share complex procedures more easily between Analysts and other database users

MySQL Workbench in Action

```
USE thriftshop;

-- changing the delimiter
DELIMITER //
-- creating the procedure
CREATE PROCEDURE sp_selectAllInventory()
BEGIN
    SELECT * FROM inventory;
END //
-- changing the delimiter back to the default
DELIMITER ;

-- calling the procedure that we have created
CALL sp_selectAllInventory();

-- if we later want to DROP the procedure, we can use the this...
-- DROP PROCEDURE IF EXISTS sp_selectAllInventory();
```



NEW MESSAGE

From: **Joe Tidyman** (Owner, Sloppy Joe's)
Subject: Help Creating a Stored Procedure

Hi there...

As we scale up, I would like a very quick way to query how many orders each of our staff members has served, all-time.

I am not a database pro, so I need this to be as simple as possible. Something like the following would be ideal...

CALL sp_staffOrdersServed;

Can you help?

-Joe

Reply

Forward

Helpful Hint

```
-- HINT: this is the query you want to embed in a stored procedure
SELECT
    staff_id,
    COUNT(order_id) AS orders_served
FROM customer_orders
GROUP BY staff_id
```

TEST YOUR SKILLS: STORED PROCEDURES

TRIGGERS

- MySQL allows us to create Triggers, where we can prescribe certain actions on a table to trigger one or more other actions to occur

- We may prescribe that our triggered action occurs either BEFORE or AFTER an INSERT, UPDATE, or a DELETE

- Triggers are a very common way to make sure related tables remain in sync as they are updated over time

MySQL Workbench in Action

```
CREATE TRIGGER purchaseUpdateInventory
AFTER INSERT ON customer_purchases
FOR EACH ROW
    UPDATE inventory
        -- subtracting an item for each purchase
        SET number_in_stock = number_in_stock - 1
    WHERE inventory_id = NEW.inventory_id;

INSERT INTO customer_purchases VALUES
(13,NULL,3,NULL), -- inventory_id = 3, velour jumpsuit
(14,NULL,4,NULL) -- inventory_id = 4, house slippers
;
```

```
SELECT * FROM inventory;
```

Result Preview (SELECT * FROM inventory)



| Result Grid | | |
|--------------|-----------------------|-----------------|
| inventory_id | item_name | number_in_stock |
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 12 |
| 4 | house slippers | 6 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 8 | kneeboard | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |

| Result Grid | | |
|--------------|-----------------------|-----------------|
| inventory_id | item_name | number_in_stock |
| 1 | fur coat | 0 |
| 2 | moccassins | 4 |
| 3 | velour jumpsuit | 11 |
| 4 | house slippers | 5 |
| 5 | brown leather jacket | 3 |
| 6 | broken keyboard | 6 |
| 8 | kneeboard | 2 |
| 9 | pro wings sneakers | 0 |
| 10 | wolf skin hat | 1 |
| 11 | fur fox skin | 1 |
| 12 | plaid button up shirt | 8 |
| 13 | flannel zebra jammies | 6 |



NEW MESSAGE

From: **Joe Tidyman** (Owner, Sloppy Joe's)

Subject: Help Creating Triggers

Hi there...

Is there any way we could automatically update the number of orders served in the staff table anytime we serve another order and a record is written to the customer_orders table?

Is this even possible?

-Joe

Reply

Forward

Helpful Hint

-- use this to see the # of orders_served before updates

```
SELECT * FROM staff;
```

-- then, create your trigger

-- then, insert these 4 records into customer_orders

```
INSERT INTO customer_orders VALUES
```

```
(21, 1, 10.98),
```

```
(22, 2, 5.99),
```

```
(23, 2, 7.99),
```

```
(24, 2, 12.97);
```

-- finally, query the staff table again

-- if you did this right, orders_served has increased

```
SELECT * FROM staff
```

TEST YOUR SKILLS: TRIGGERS

SERVER MANAGEMENT

- The Administration tab gives us options to monitor and control the status of the MySQL Server we are connected to

The Server Status view gives us a quick view of our Server

- The Startup / Shutdown tool allows us to Start or Stop our database instance

MySQL Workbench in Action

This screenshot shows the MySQL Workbench interface with the 'Administration' tab selected. The main pane displays the 'Server Status' section, which includes fields for Connection Name (mavenmovies), Host, Port, Configuration File, and Running Since. Below this, there's a summary of Available Server Features and a detailed section for Server Directories. On the right, a dashboard provides real-time monitoring for various MySQL metrics.

This screenshot shows the MySQL Workbench interface with the 'Administration' tab selected. The main pane displays the 'Startup / Shutdown MySQL Server' dialog. It shows that the database server is started and ready for client connections. The 'Stop Server' button is visible if needed. Below this, the 'Startup Message Log' pane is shown, which typically displays logs related to the server's startup process.



NEW MESSAGE

From: **Joe Tidyman** (Owner, Sloppy Joe's)

Subject: Is Our Server Stuck?

Hi again...

It seems like our database might be stuck, or out of sync?

I talked with another business owner who said the server can sometimes get stuck and need to be restarted by a Database Administrator. Can you check the status of the server and restart it for us?

-Joe

Reply

Forward

Helpful Hint

```
/*
for the purpose of this exercise, pretend the server
on your machine is the server in question
*/  
  
-- check the status of your server  
  
-- IF it's stopped, start it  
-- IF it's already running, stop it, then start it  
  
/*  
NOTE: depending on the settings of your machine,  
you may need to enter both your root user password,  
AND possibly also your system password  
(the one you use to log in to your machine)  
*/
```

TEST YOUR SKILLS: SERVER MANAGEMENT

USER MANAGEMENT

- Workbench makes it easy for us to add Users and to manage their Privileges

- Users can be granted varying levels of permissions. At one end of the spectrum, you could create a User that can only SELECT, and at the other end of the spectrum is the DBA role, which carries all privileges

MySQL Workbench in Action

The screenshot shows the MySQL Workbench interface with the title "Administration - Users and Privileges". In the sidebar under "MANAGEMENT", the "Users and Privileges" item is highlighted with a green circle and an arrow pointing to the main content area. The main area displays a table titled "User Accounts" with columns "User" and "From Host". The table lists several accounts: mysql.infoschema (localhost), mysql.session (localhost), mysql.sys (localhost), newuser (%), newuser (localhost), and root (localhost). A green arrow points from the "Add Account" button at the bottom left of the table to the "Details for account newuser@%" tab, which is currently active. This tab contains fields for "Login Name" (newuser), "Authentication Type" (Standard), "Limit to Hosts Matching" (%), "Password" (redacted), "Confirm Password" (redacted), and "Expire Password".



NEW MESSAGE

From: **Joe Tidyman (Owner, Sloppy Joe's)**

Subject: **Help Creating 2 New Users**

Hi there...

Hoping you could help us out with one more thing. I would like to create two new Users who can access our database...

User: DataGuru, DBA-level privileges, Password: ChangeMe!

User: Analyst, SELECT only privileges, Password: ChangeMe!

Thank you!!

-Joe

Reply

Forward

Helpful Hint

-- REMEMBER: restrict access and permissions to just what is needed

TEST YOUR SKILLS: USER MANAGEMENT

INTRODUCING THE FINAL COURSE PROJECT

THE **SITUATION**

A new client, Bubs Glover, the owner of Bubs' Bigtime Baby Booties, has reached out to you for help building his business a database from the ground up.

THE **OBJECTIVE**

Use Your MySQL Database Administration skills to:

Design a database from scratch, which will capture information about Bubs' customers, the purchases they make, his products, and his employees.

FINAL COURSE PROJECT QUESTIONS

1

- Bubs wants you to track information on his customers (first name, last name, email), his employees (first name, last name, start date, position held), his products, and the purchases customers make (which customer, when it was purchased, for how much money). Think about how many tables you should create. Which data goes in which tables? How should the tables relate to one another?
- ~ 0:37

2

- Given the database design you came up with, use Workbench to create an EER diagram of the database. Include things like primary keys and foreign keys, and anything else you think you should have in the tables. Make sure to use reasonable data types for each column.
- ~ 4:50

3

- Create a schema called bubsbooties. Within that schema, create the tables that you have diagrammed out. Make sure they relate to each other, and that they have the same reasonable data types you selected previously.
- ~ 15:50

FINAL COURSE PROJECT QUESTIONS

4

Add any constraints you think your tables' columns should have. Think through which columns need to be unique, which ones are allowed to have NULL values, etc.

~ 24:05

5

Insert at least 3 records of data into each table. The exact values do not matter, so feel free to make them up. Just make sure that the data you insert makes sense, and that the tables tie together.

~ 28:40

6

Create two users and give them access to the database. The first user, TuckerReilly, will be the DBA, and should get full database administrator privileges. The second user, EllaBrody, is an Analyst, and only needs read access.

~ 40:10