SQL

Student Exercises

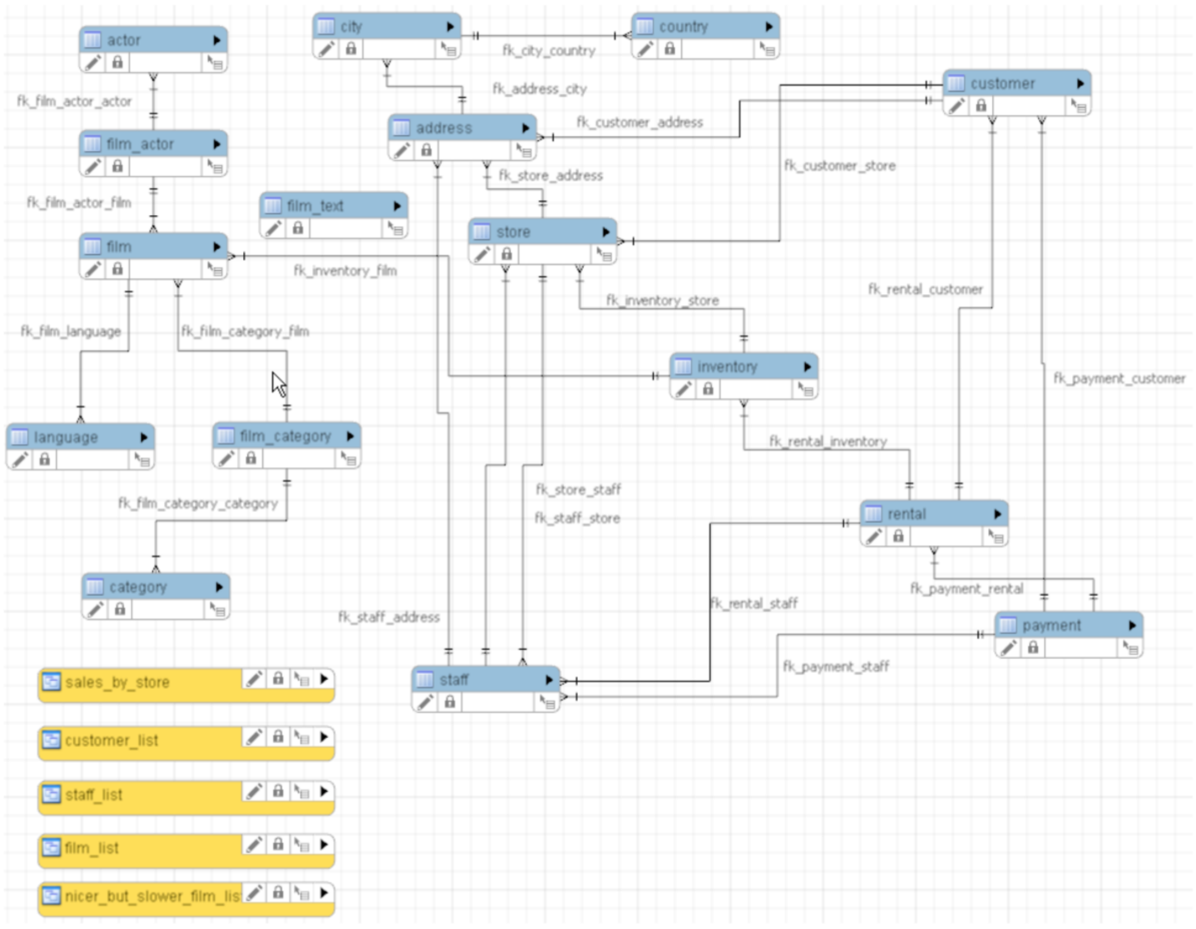
Version 6.0 ITC

Day 6

The sakila Database

- **The examples/exercises here use a sample database called sakila**

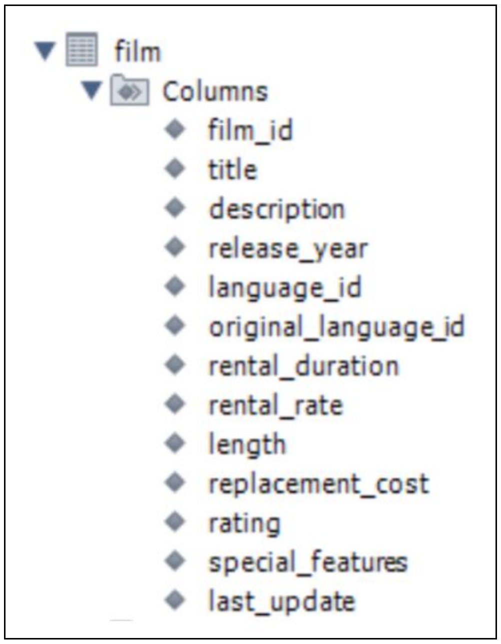
- **It is a database for a company that sells movies**
  - You can read about the schema here: https://dev.mysql.com/doc/sakila/en/



- **Early examples use the film table**
  - The film table is a list of all films that potentially might in stock in the stores

- The actual in-stock copies of each film are represented in the inventory table.

- **The film table contains the following columns:**

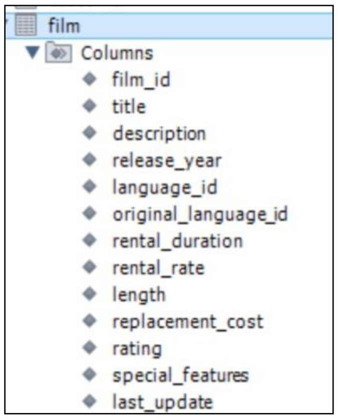


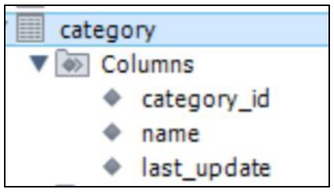
film
Columns
- film_id
- title
- description
- release_year
- language_id
- original_language_id
- rental_duration
- rental_rate
- length
- replacement_cost
- rating
- special_features
- last_update

Examples: Using SELECT

Query

SAKILA DATABASE: The **`film`** and **`category`** tables are shown below:





QUERY: What are the categories of films we carry?

APPROACH: Use a SELECT statement and list all columns from the category table.

```
SELECT *
FROM category
```

RESULTS:

```
+-------------+----------------------+---------------------+
| category_id | name                 | last_update         |
+-------------+----------------------+---------------------+
| 1           | Action               | 2006-02-15 04:46:27 |
| 2           | Anumation            | 2006-02-15 04:46:27 |
| ... there are many rows returned ...                     |
+-------------+----------------------+---------------------+
```

Query

QUERY: What films have a "PG" rating that run between 90 and 120 minutes? List the results in descending order by length. If two or more films have the same length, then list them in alphabetical order by title.

APPROACH: Use a WHERE clause and specify a condition that the rating column must be "equal to" the string "PG" and that the value in the rating column must be between 90 and 120.

```
SELECT film_id, title, rating, length
FROM film
WHERE rating = "PG" AND (length >= 90 AND length <= 120)
ORDER BY length DESC, title;
```

RESULTS:

```
+---------+----------------------+---------+---------+
| film_id | title                | rating  | length  |
+---------+----------------------+---------+---------+
| 477     | JAWVREAKER BROOKLYN  | PG      | 118     |
| 645     | OTHERS SOUP          | PG      | 118     |
| ... there are many rows that match ...             |
+---------+----------------------+---------+---------+
```

Query

QUERY: What films titles start with the word "Theory"?

APPROACH: Use LIKE with a wildcard value of "Theory%" in the query.

```
SELECT film_id, title, rating
FROM film
WHERE title LIKE "Theory%";
```

RESULTS:

```
+---------+----------------------+---------+
| film_id | title                | rating  |
+---------+----------------------+---------+
| 886     | THEORY MERMAID       | PG-13   |
+---------+----------------------+---------+
```

Query

QUERY: What films have a running length in the range 89-91 minutes?

APPROACH: Use BETWEEN to specify a range of values for length.

```
SELECT film_id, title, length
FROM film
WHERE length BETWEEN 89 AND 91
ORDER BY title;
```

RESULTS:

```
+---------+----------------------+---------+
| film_id | title                | length  |
+---------+----------------------+---------+
| 28      | ANTHEM LUKE          | 91      |
| 57      | BASIC EASY           | 90      |
| ... there are many rows that match ...   |
+---------+----------------------+---------+
```

Query

QUERY: Find all films that don't have a value for original_language_id

```
SELECT film_id, title
FROM film
WHERE original_language_id IS NULL;
```

Query

QUERY: Find all of the unique prices we rent films for

```
SELECT DISTINCT(rental_rate)
FROM film;
```

RESULTS:

```
+--------------+
| rental_rate  |
+--------------+
| 0.99         |
| 4.99         |
| 2.99         |
+--------------+
```

Exercises

EXERCISE 1

In this exercise, you will install the Northwind database and then run some simple queries against it.

We will use the Microsoft Northwind database for many of the exercises and examples in this workbook. You can find the SQL script file and install instructions for MySQL here: `https://www.aspsnippets.com/Articles/Download-and-Install-Microsoft-Northwind-Sample-database-in-MySql.aspx`

To see your new database in the Navigator window, you may have to refresh it. Right-click in the Navigator window and choose Refresh All.

Northwind is a database for a small grocery store. Take a few minutes to examine the schema. Then answer the following questions by either looking at the tables, the columns, or running a query.

NOTE: You will want to add these to a `.sql` file with comments or a txt file and save them in a GitHub repo for future reference.

You can put all SQL statements in the same script with comments in front of them and then only run the selected query by pressing the 2nd lightning bolt.

1. What is the name of the table that holds the items Northwind sells?
2. Write a query to list the product id, product name, and unit price of every product.
3. Write a query to list the product id, product name, and unit price of every product. Except this time, order then in ascending order by price.
4. What are the products that we carry where the unit price is $7.50 or less?
5. What are the products that we carry where we have at least 100 units on hand? Order them in descending order by price.
6. What are the products that we carry where we have at least 100 units on hand? Order them in descending order by price. If two or more have the same price, list those in ascending order by product name.
7. What are the products that we carry where we have no units on hand, but 1 or more units of them on backorder? Order them by product name.

8. What is the name of the table that holds the types (categories) of the items Northwind sells?

9. Write a query that lists all of the columns and all of the rows of the categories table? What is the category id of seafood?

10. Examine the Products table. How does it identify the type (category) of each item sold? Write a query to list all of the seafood items we carry.

11. What are the first and last names of all of the Northwind employees?

12. What employees have "manager" in their titles?

13. List the distinct job titles in employees.

14. What employees have a salary that is between $200 0 and $2500?

15. List all of the information about all of Northwind's suppliers.

16. Examine the Products table. How do you know what supplier supplies each product? Write a query to list all of the items that "Tokyo Traders" supplies to Northwind

Examples: Aggregate Functions

Query

QUERY: How many films are in the films table?

APPROACH: Use the COUNT() function to count the number of rows in the film table.

```
SELECT COUNT(*)
FROM film;
```

RESULTS:

```
+----------+
| COUNT(*) |
+----------+
| 1000     |
+----------+
```

Query

QUERY: How many distinct ratings are represented in the films table?

APPROACH: Use the COUNT() function combined with DISTINCT to count the number of ratings in the film table.

```
SELECT COUNT(DISTINCT(rating))
FROM film;
```

RESULTS:

```
+-------------------------+
| COUNT(DISTINCT(rating)) |
+-------------------------+
| 5                       |
+-------------------------+
```

Query

QUERY: If I wanted to watch all of the movies in the film catalog, how long would it take?

APPROACH: Use the SUM() function to add up all the length values in the films table.

```
SELECT SUM(length)
FROM film;
```

RESULTS:

```
+---------------+
| SUM(length)   |
+---------------+
| 115272        |
+---------------+
```

Query

QUERY: What is the average cost to rent a "G"-rated film?

APPROACH: Use the AVG() function to find the average value in the rental_rate column of all films whose rating is "G".

```
SELECT AVG(rental_rate)
FRPM film
WHERE rating = "G";
```

RESULTS:

```
+------------------+
| AVG(rental_rate) |
+------------------+
| 2.888876         |
+------------------+
```

Query

QUERY: How short is the shortest film? What about the longest?

APPROACH: Use the MIN() and MAX() function to examine the length.

```
SELECT MIN(length)
FROM film;
```

RESULTS:

```
+-------------+
| MIN(length) |
+-------------+
| 46          |
+-------------+
```

```
SELECT MAX(length)
FROM film;
```

RESULTS:

```
+-------------+
| MAX(length) |
+-------------+
| 185         |
+-------------+
```

For a list of other MySQL functions, see:
https://www.w3schools.com/sql/sql_ref_mysql.asp

Examples: Working with Groups

Query

QUERY: How many movies are available broken down by rating (G, PG, etc)?

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the COUNT() function to count the number of rows in each group.

```
SELECT rating, COUNT(*)
FROM film
GROUP BY rating;
```

RESULTS:

```
+----------+----------+
| rating   | COUNT(*) |
+----------+----------+
| PG       | 194      |
| G        | 178      |
| NC-17    | 210      |
| PG-13    | 223      |
| R        | 195      |
+----------+----------+
```

Query

QUERY: What is the average price to rent a movie broken down by rating (G, PG, etc)??

APPROACH: Group films by rating and then use the AVG() function to calculate the average rental_rate of rows in each group.

```
SELECT rating, avg(rental_rate)
FROM film
GROUP BY rating;
```

RESULTS:

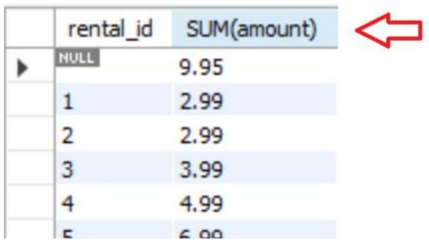
```
+----------+------------------+
| rating   | AVG(rental_rate) |
+----------+------------------+
| PG       | 3.051856         |
| G        | 2.888876         |
| NC-17    | 2.970952         |
| PG-13    | 3.034843         |
| R        | 2.9387818        |
+----------+------------------+
```

Examples: Working with AS

Query

Computed fields don't have an official name in a SQL query

```
SELECT rental_id, SUM(amount)
FROM payment
GROUP BY rental_id
ORDER BY rental_id;
```



	rental_id	SUM(amount)
▶	NULL	9.95
	1	2.99
	2	2.99
	3	3.99
	4	4.99
	5	6.99

This can be a problem if you want to use it to order the results. SQL provides the AS keyword to create an alias for the column name

```
SELECT rental_id, SUM(amount) AS total_amount
FROM payment
GROUP BY rental_id
ORDER BY rental_id;
```

Query

QUERY: What is the average price to rent a movie broken down by rating (G, PG, PG-13, etc) and displayed in ascending order by average price?

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the `AVG()` function to calculate the average rental_rate of rows in each group. Make sure to name the value returned by the `AVG()` function so that we can use it in the ORDER BY clause.

```
SELECT rating, AVG(rental_rate) AS avg_rate
FROM film
GROUP BY rating
ORDER BY avg_rating;
```

RESULTS:

```
+----------+-----------+
| rating   | avg_rate   |
+----------+-----------+
| G        | 2.888876   |
| R        | 2.938781   |
| NC-17    | 2.970952   |
| PG-13    | 3.034843   |
| PG       | 3.051856   |
+----------+-----------+
```

Query

QUERY: What is the average rating for movies broken down by rating (G, PG, PG-13, etc)? NOTE: I'm not interested in the rating if there are less than 200 films in the group.

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the COUNT() function to count the number rows in each group. Only display the groups that have at least 200 rows.

```
SELECT rating, COUNT(*)
FROM film
GROUP BY rating
HAVING COUNT(*) >= 200
ORDER BY rating;
```

RESULTS:

```
+----------+-----------+
| rating   | COUNT(*)  |
+----------+-----------+
| NC-17    | 210       |
| PG-13    | 223       |
+----------+-----------+
```

Exercises

EXERCISE 1

Continue to execute queries against the Northwind database.

Add these to your `.sql` or text file and save them in a GitHub repo

1. How many suppliers are there? Use a query!
2. What is the sum of all the employee's salaries?
3. What is the price of the cheapest item that Northwind sells?
4. What is the average price of items that Northwind sells?
5. What is the price of the most expensive item that Northwind sells?
6. What is the supplier ID of each supplier and the number of items they supply? You can answer this query by only looking at the Products table.
7. What is the category ID of each category and the average price of each item in the category? You can answer this query by only looking at the Products table.
8. For suppliers that provide at least 5 items to Northwind, what is the supplier ID of each supplier and the number of items they supply? You can answer this query by only looking at the Products table.
9. List the product id, product name, and inventory value (calculated by multiplying unit price by the number of units on hand). Sort the results in descending order by value. If two or more have the same value, order by product name.

Examples: Nested Queries

Query

QUERY: Which film(s) are the most expensive to replace?

APPROACH: Use the SQL `max()` function to find the largest replacement_cost in the film table, and then use that maximum cost in a different query to select the film(s) that have that replacement cost.

```
SELECT film_id, title, replacement_cost
FROM film
WHERE replacement_cost = (SELECT MAX(replacement_cost)
                          FROM film);
```

RESULTS:

```
+---------+-------------------------+-------------------+
| film_id | title                   | replacement_cost  |
+---------+-------------------------+-------------------+
| 34      | ARABIA DOGMA            | 29.99             |
| 52      | BALLROOM MOCKINGBIRD    | 29.99             |
| ... there were many, many rows that matched ...       |
+---------+-------------------------+-------------------+
```

Query

QUERY: Which film(s) are described as documentaries and how long do they run?

APPROACH: If we research the film_text table in the sakila database, we find it contains 3 columns named film_id, title, and description. We can run a query to find the films that have "documentary" in their descriptions. But the length of the film isn't available in film_text.

In this solution below, we keep the film_id values of the query that searches for documentaries and then use ANOTHER query against the film table to find all films in that 1st query's film_id list. Note that the where uses the keyword "in" rather than an "=" to match film_id values.

```
SELECT title, length
FROM film
WHERE film_id IN (SELECT film_id
                  FROM film_text
                  WHERE description LIKE "%documentary%");
```

RESULTS:

```
+--------------------------+--------+
| title                    | length |
+--------------------------+--------+
| AFFAIR PREJUDICE         | 117    |
| AFRICAN EGG              | 130    |
| ... many rows matched ...         |
+--------------------------+--------+
```

Exercises

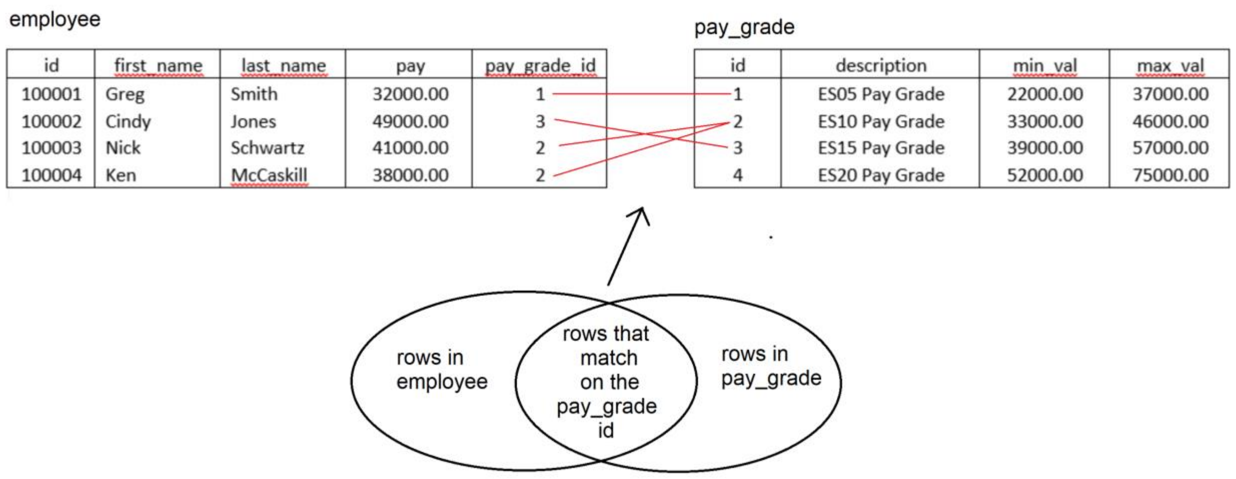
EXERCISE 1

Continue to execute queries against the Northwind database. Continue to add these to your file.

1. What is the product name(s) of the most expensive products? HINT: Find the max price in a subquery and then use that value to find products whose price equals that value.
2. What is the order id, shipping name and shipping address of all orders shipped via "Federal Shipping"? HINT: Find the shipper id of "Federal Shipping" in a subquery and then use that value to find the orders that used that shipper.
3. What are the order ids of the orders that ordered "Sasquatch Ale"? HINT: Find the product id of "Sasquatch Ale" in a subquery and then use that value to find the matching orders from the \`order details\` table. Because the \`order details\` table has a space in its name, you will need to surround it with back ticks in the FROM clause.
4. What is the name of the employee that sold order 10266?
5. What is the name of the customer that bought order 10266?

Example: Working with Joins

Query

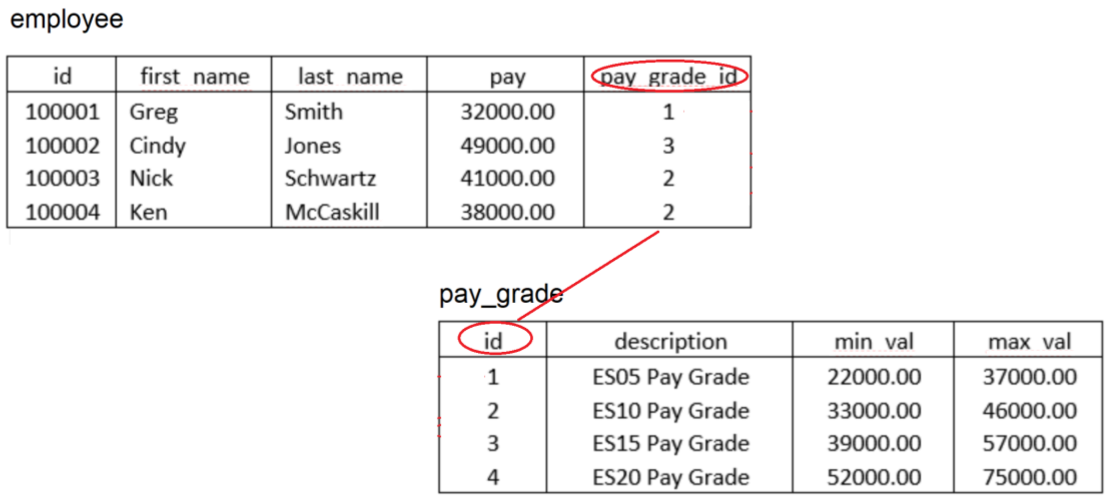

employee

id	first_name	last_name	pay	pay_grade_id
100001	Greg	Smith	32000.00	1
100002	Cindy	Jones	49000.00	3
100003	Nick	Schwartz	41000.00	2
100004	Ken	McCaskill	38000.00	2

pay_grade

id	description	min_val	max_val
1	ES05 Pay Grade	22000.00	37000.00
2	ES10 Pay Grade	33000.00	46000.00
3	ES15 Pay Grade	39000.00	57000.00
4	ES20 Pay Grade	52000.00	75000.00

QUERY: We want to list each employee, along with their pay_grade description and the min/max salary of that pay grade.


employee

id	first_name	last_name	pay	pay_grade_id
100001	Greg	Smith	32000.00	1
100002	Cindy	Jones	49000.00	3
100003	Nick	Schwartz	41000.00	2
100004	Ken	McCaskill	38000.00	2

pay_grade

id	description	min_val	max_val
1	ES05 Pay Grade	22000.00	37000.00
2	ES10 Pay Grade	33000.00	46000.00
3	ES15 Pay Grade	39000.00	57000.00
4	ES20 Pay Grade	52000.00	75000.00

APPROACH: Join the employee table to the pay_grade table and match employee.pay_grade_id to pay_grade.id

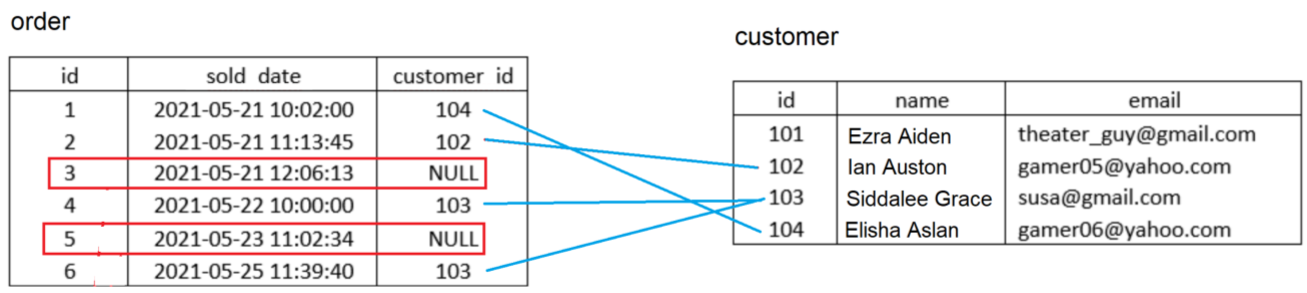
```
SELECT employee.id, first_name, last_name, description, min_val,
max_val
FROM employee
JOIN pay_grade
  ON employee.pay_grade_id = pay_grade.id;
```

RESULTS:

```
+--------+------------+-----------+---------------+-----------+-----------+
| id     | first_name | last_name | description   | min_val   | max_val   |
+--------+------------+-----------+---------------+-----------+-----------+
| 100001 | Greg       | Smith     | ES05 Pay Grade | 22000.00  | 37000.00  |
| 100002 | Cindy      | Jones     | ES15 Pay Grade | 39000.00  | 46000.00  |
| 100003 | Nick       | Schwartz  | ES10 Pay Grade | 33000.00  | 46000.00  |
| 100004 | Ken        | McCaskill | ES10 Pay Grade | 33000.00  | 46000.00  |
+--------+------------+-----------+---------------+-----------+-----------+
```

We could have added the `WHERE, ORDER BY, GROUP BY`, and `HAVING` clauses if we wanted. And although this example does a JOIN on two tables, you can JOIN as many tables as you need to by continuing to add additional JOIN clauses.

Query



order

id	sold_date	customer_id
1	2021-05-21 10:02:00	104
2	2021-05-21 11:13:45	102
3	2021-05-21 12:06:13	NULL
4	2021-05-22 10:00:00	103
5	2021-05-23 11:02:34	NULL
6	2021-05-25 11:39:40	103

customer

id	name	email
101	Ezra Aiden	theater_guy@gmail.com
102	Ian Auston	gamer05@yahoo.com
103	Siddalee Grace	susa@gmail.com
104	Elisha Aslan	gamer06@yahoo.com

orders 3 and 5 are included
in a LEFT OUTER JOIN

QUERY: What orders were sold when?

```
SELECT order.id, sold_date, name, email
FROM order
LEFT JOIN customer
       ON order.customer_id = customer.id;
```

RESULTS:

```
+-----+---------------------+------------------+-------------------+
| id  | sold_date           | name             | email             |
+-----+---------------------+------------------+-------------------+
| 1   | 2021-05-21 10:02:00 | Elisha Aslan     | gamer06@yahoo.com |
| 2   | 2021-05-21 11:13:45 | Ian Auston       | gamer05@yahoo.com |
| 3   | 2021-05-21 12:06:13 | NULL             | NULL              |
| 4   | 2021-05-22 10:00:00 | Siddalee Grace   | susa@gmail.com    |
| 5   | 2021-05-23 11:02:34 | NULL             | NULL              |
| 6   | 2021-05-25 11:39:40 | Siddalee Grace   | susa@gmail.com    |
+-----+---------------------+------------------+-------------------+
```

Exercises

EXERCISE 1

Now take a few minutes to look at this great visual diagram of the different types of joins:

```
https://www.codeproject.com/Articles/33052/Visual-
Representation-of-SQL-Joins
```

EXERCISE 2

Let's continue working with Northwind.

1. List the product id, product name, unit price and category name of all products. Order by category name and within that, by product name.
2. List the product id, product name, unit price and supplier name of all products that cost more than $75. Order by product name.
3. List the product id, product name, unit price, category name, and supplier name of every product. Order by product name.
4. What is the product name(s) and categories of the most expensive products? HINT: Find the max price in a subquery and then use that in your more complex query that joins products with categories.
5. List the order id, ship name, ship address, and shipping company name of every order that shipped to Germany.
6. List the order id, order date, ship name, ship address of all orders that ordered "Sasquatch Ale"?

Examples: Inserting, Updating and Deleting Data

Query

TASK: Add a new country to the sakila country table

STATEMENT (option 1):

```
INSERT INTO country(country_id, country, last_update)
VALUES(110, "Zimbabwe", NOW());
```

STATEMENT (option 2):

```
INSERT INTO country
VALUES(110, "Zimbabwe", NOW());
```

Query

TASK: Change the first and last name for the customer whose customer_id is 2.

```
UPDATE customer
SET first_name = 'PATTY', last_name = 'JOHNSTON'
WHERE customer_id = 2;
```

Query

TASK: Change all PATTY first names to PATRICE.

```
-- No primary key specified
SET SQL_SAFE_UPDATES=0;

UPDATE customer
SET first_name = 'PATRICE'
WHERE first_name = 'PATTY';

SET SQL_SAFE_UPDATES=1;
```

Query

TASK: Delete all references to the payment whose payment_id is 100

```
DELETE FROM payment
WHERE payment_id = 100;
```

Exercises

EXERCISE 1

Let's continue working with Northwind.

1. Add a new supplier.
2. Add a new product provided by that supplier
3. List all products and their suppliers.
4. Raise the price of your new product by 15%.
5. List the products and prices of all products from that supplier.
6. Delete the new product.
7. Delete the new supplier.
8. List all products.
9. List all suppliers.

Examples: Working with the Schema

Note: A list of MySQL data types can be found here:
    `https://dev.mysql.com/doc/refman/8.0/en/data-types.html`

Query

TASK: Create a new table to track advertised sales

```
CREATE TABLE advertisements (
    AdId int NOT NULL,
    Title varchar(50) NOT NULL,
    MagicCode varchar(9),
    PercentOff float NOT NULL,
    PRIMARY KEY(AdId)
);
```

Query

TASK: Create a new table to track advertised sales but use an auto-increment key

```
CREATE TABLE advertisements (
    AdId int NOT NULL AUTO_INCREMENT,
    Title varchar(50) NOT NULL,
    MagicCode varchar(9),
    PercentOff float NOT NULL,
    PRIMARY KEY(AdId)
);
```

Query

TASK: Drop the advertisements table

```
DROP TABLE advertisements;
```

Query

TASK: Delete the data in the advertisements table

```
TRUNCATE TABLE advertisements;
```

Query

TASK: Add an AuthorizedBy column to the advertisements table

```
ALTER TABLE advertisements
ADD COLUMN AuthorizedBy varchar(20);
```

Query

TASK: Drop the AdvertisedBy column from the advertisements table

```
ALTER TABLE advertisements
DROP COLUMN AuthorizedBy;
```

Query

TASK: Modify the MagicCode column from being a varchar(9) to a varchar(12) in the advertisements table

```
ALTER TABLE advertisements
MODIFY COLUMN MagicCode varchar(12);
```

Day 7

Additional Exercises: Querying the sakila Database

If you feel you need MORE experience practicing SQL, we have included additional exercises here. If you feel your understanding of SQL is pretty good, move on to the Node.js videos and exercises for SQL and Sequelize.

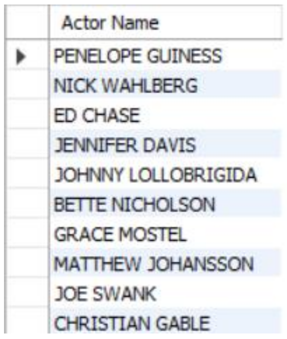
The following questions must answer by creating SQL queries that run against the sakila database. Below the query is the expected result that you should get back.

Use MySQL Workbench to test your queries. Save your queries in a new .sql or text file.

Exercises

1. Display the first and last name of each actor in a single column in upper case letters. Name the column `Actor Name`.

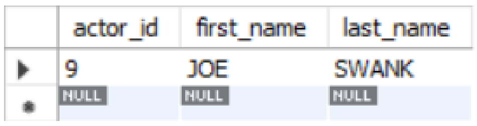
Result set



Actor Name
PENELOPE GUINESS
NICK WAHLBERG
ED CHASE
JENNIFER DAVIS
JOHNNY LOLLOBRIGIDA
BETTE NICHOLSON
GRACE MOSTEL
MATTHEW JOHANSSON
JOE SWANK
CHRISTIAN GABLE

2. You need to find the ID number, first name, and last name of an actor, of whom you know only the first name, "Joe."

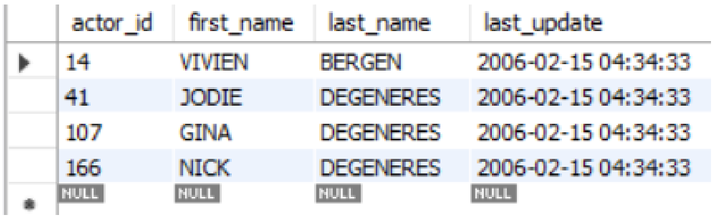
Result set



actor_id	first_name	last_name
9	JOE	SWANK
NULL	NULL	NULL

3. Find all actors whose last name contain the letters `GEN`.

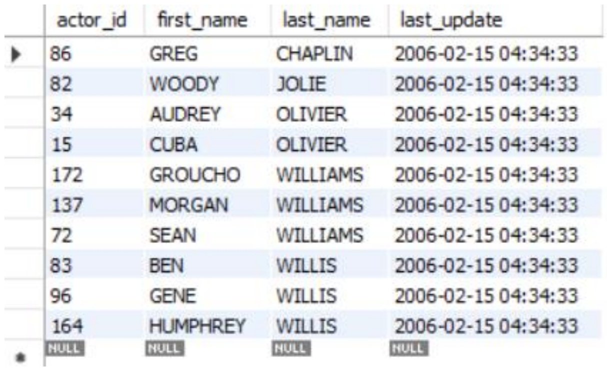
Result set



actor_id	first_name	last_name	last_update
14	VIVIEN	BERGEN	2006-02-15 04:34:33
41	JODIE	DEGENERES	2006-02-15 04:34:33
107	GINA	DEGENERES	2006-02-15 04:34:33
166	NICK	DEGENERES	2006-02-15 04:34:33
NULL	NULL	NULL	NULL

4. Find all actors whose last names contain the letters `"LI"`. This time, order the rows by last name and first name, in that order.

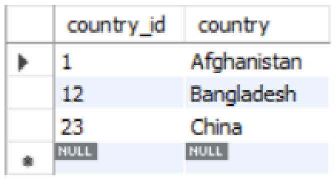
Result set



actor_id	first_name	last_name	last_update
86	GREG	CHAPLIN	2006-02-15 04:34:33
82	WOODY	JOLIE	2006-02-15 04:34:33
34	AUDREY	OLIVIER	2006-02-15 04:34:33
15	CUBA	OLIVIER	2006-02-15 04:34:33
172	GROUCHO	WILLIAMS	2006-02-15 04:34:33
137	MORGAN	WILLIAMS	2006-02-15 04:34:33
72	SEAN	WILLIAMS	2006-02-15 04:34:33
83	BEN	WILLIS	2006-02-15 04:34:33
96	GENE	WILLIS	2006-02-15 04:34:33
164	HUMPHREY	WILLIS	2006-02-15 04:34:33
NULL	NULL	NULL	NULL

5. Using `IN`, display the `country_id` and `country` columns of the following countries: Afghanistan, Bangladesh, and China.

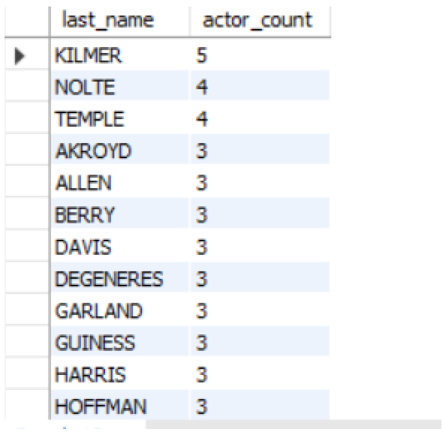
Result set



country_id	country
1	Afghanistan
12	Bangladesh
23	China
NULL	NULL

6. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors

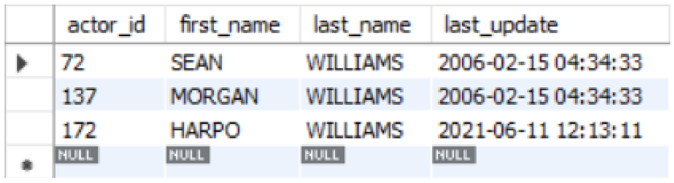
Result set



last_name	actor_count
KILMER	5
NOLTE	4
TEMPLE	4
AKROYD	3
ALLEN	3
BERRY	3
DAVIS	3
DEGENERES	3
GARLAND	3
GUINESS	3
HARRIS	3
HOFFMAN	3

7. The actor `HARPO WILLIAMS` was accidentally entered in the actor table as `GROUCHO WILLIAMS`. Write a query to fix the record, and another to verify the change.

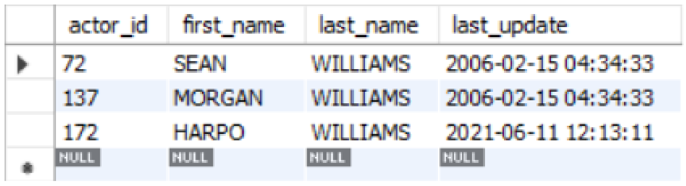
Result set



actor_id	first_name	last_name	last_update
72	SEAN	WILLIAMS	2006-02-15 04:34:33
137	MORGAN	WILLIAMS	2006-02-15 04:34:33
172	HARPO	WILLIAMS	2021-06-11 12:13:11
NULL	NULL	NULL	NULL

8. Perhaps we were too hasty in changing `GROUCHO` to `HARPO`. It turns out that `GROUCHO` was the correct name after all! In a single query, if the first name of the actor is currently `HARPO`, change it to `GROUCHO`. Then write a query to verify your change.

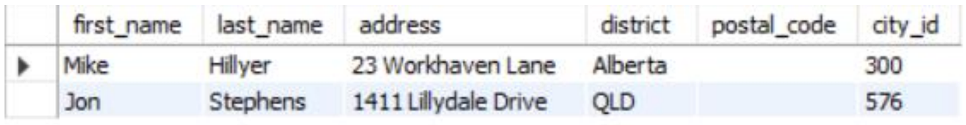
Result set



actor_id	first_name	last_name	last_update
72	SEAN	WILLIAMS	2006-02-15 04:34:33
137	MORGAN	WILLIAMS	2006-02-15 04:34:33
172	HARPO	WILLIAMS	2021-06-11 12:13:11
NULL	NULL	NULL	NULL

9. Perhaps we were too hasty in changing `GROUCHO` to `HARPO`. It turns out that `GROUCHO` was the correct name after all! In a single query, if the first name of the actor is currently `HARPO`, change it to `GROUCHO`. Then write a query to verify your change.

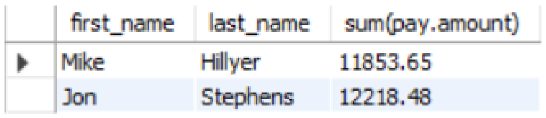
Result set



first_name	last_name	address	district	postal_code	city_id
Mike	Hillyer	23 Workhaven Lane	Alberta		300
Jon	Stephens	1411 Lillydale Drive	QLD		576

10. Use `JOIN` to display the total amount rung up by each staff member in August of 2005. Use tables `staff` and `payment`.

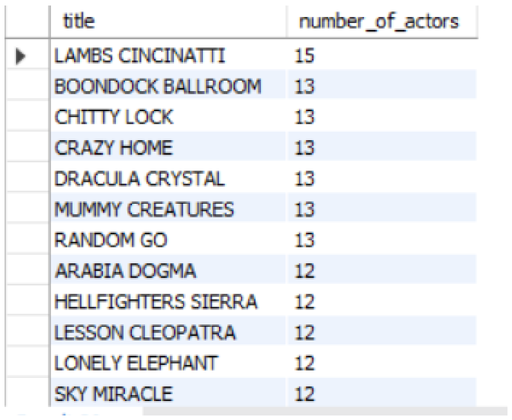
Result set



first_name	last_name	sum(pay.amount)
Mike	Hillyer	11853.65
Jon	Stephens	12218.48

11. List each film and the number of actors who are listed for that film. Use tables film_actor and film. Use inner join.

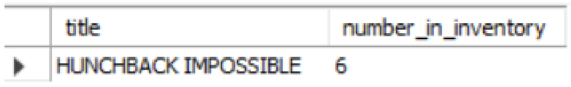
Result set



title	number_of_actors
LAMBS CINCINATTI	15
BOONDOCK BALLROOM	13
CHITTY LOCK	13
CRAZY HOME	13
DRACULA CRYSTAL	13
MUMMY CREATURES	13
RANDOM GO	13
ARABIA DOGMA	12
HELLFIGHTERS SIERRA	12
LESSON CLEOPATRA	12
LONELY ELEPHANT	12
SKY MIRACLE	12

12. How many copies of the film Hunchback Impossible exist in the inventory system?

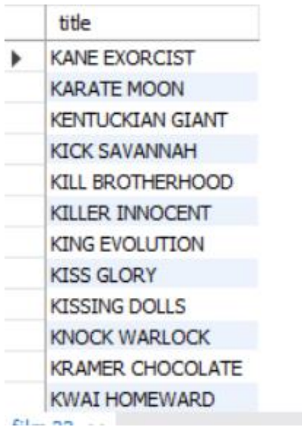
Result set



title	number_in_inventory
HUNCHBACK IMPOSSIBLE	6

13. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence, films starting with the letters K and Q have also soared in popularity. Use **subqueries** to display the titles of movies starting with the letters K and Q whose language is English.

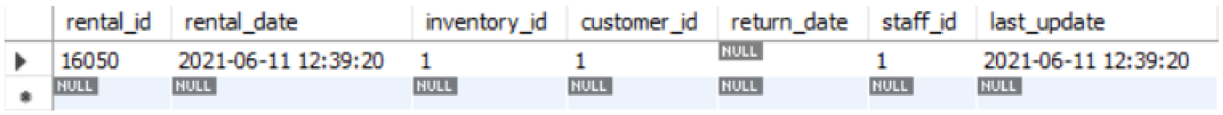
Result set



title
KANE EXORCIST
KARATE MOON
KENTUCKIAN GIANT
KICK SAVANNAH
KILL BROTHERHOOD
KILLER INNOCENT
KING EVOLUTION
KISS GLORY
KISSING DOLLS
KNOCK WARLOCK
KRAMER CHOCOLATE
KWAI HOMEWARD

14. Insert a record to represent `Mary Smith` renting the movie `‘Academy Dinosaur’` from `Mike Hillyer` at `Store 1` today. Then write a query to capture the exact row you entered into the `rental` table.

Result set (your `rental date` value will of course show the date and time you entered the record)



rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
16050	2021-06-11 12:39:20	1	1	NULL	1	2021-06-11 12:39:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Exercises: SQL Murder Mystery

If you want to test your SQL using an online game, play SQL Murder Mystery!

The SQL Murder Mystery website provides a fun, interactive opportunity to use SQL in a fun way by solving a murder mystery using your SQL skills.

Visit https://mystery.knightlab.com/ and see if YOU can solve the mystery!