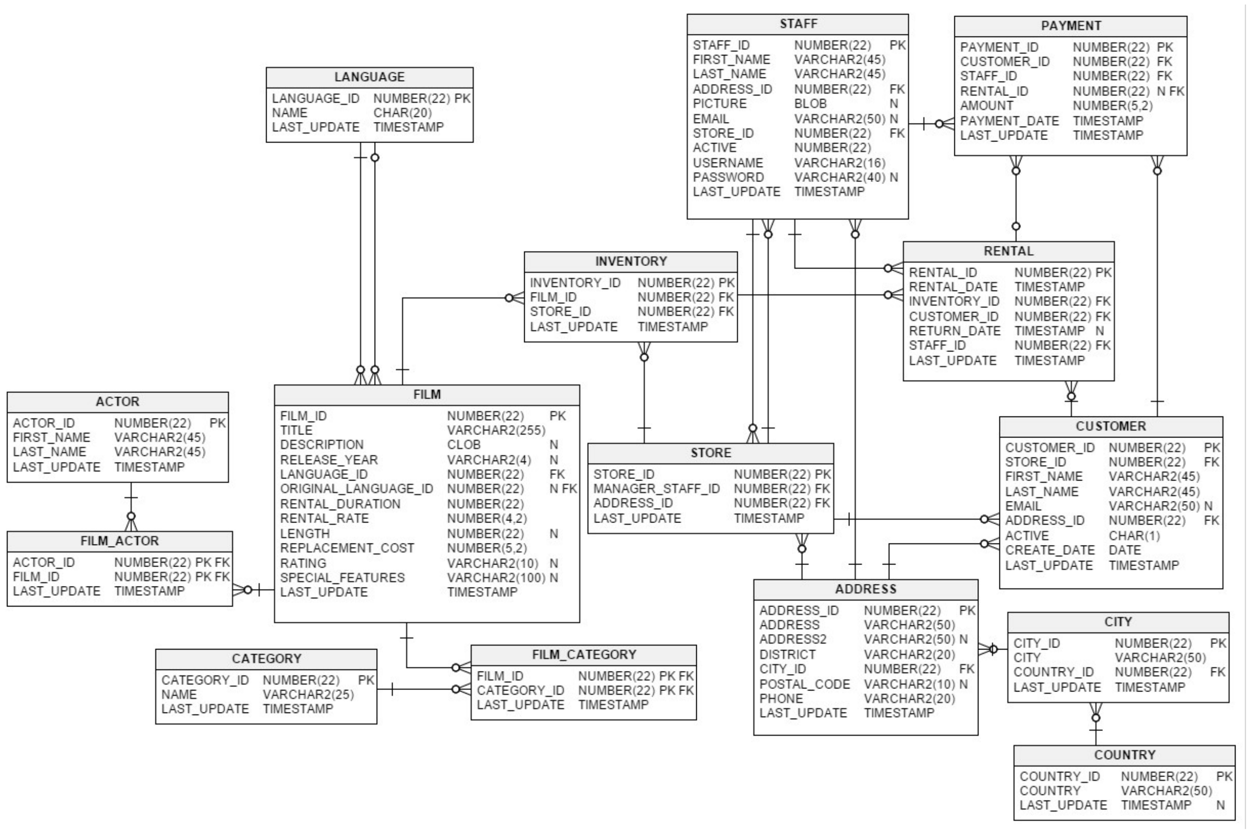
Introduction

The Sakila database is a nicely normalised schema modelling a DVD rental store, featuring things like films, actors, film-actor relationships, and a central inventory table that connects films, stores, and rentals.



Installation

Download from https://downloads.mysql.com/docs/sakila-db.zip

A downloadable archive is available in compressed **tar** file or Zip format. The archive contains three files: `sakila-schema.sql`, `sakila-data.sql`, and `sakila.mwb`.

The `sakila-schema.sql` file contains all the `CREATE` statements required to create the structure of the Sakila database including tables, views, stored procedures, and triggers.

The `sakila-data.sql` file contains the `INSERT` statements required to populate the structure created by the `sakila-schema.sql` file, along with definitions for triggers that must be created after the initial data load.

The `sakila.mwb` file is a MySQL Workbench data model that you can open within MySQL Workbench to examine the database structure

To install the Sakila sample database, follow these steps:

1. Extract the installation archive to a temporary location such as `C:\temp\` or `/tmp/`. When you unpack the archive, it creates a directory named `sakila-db` that contains the `sakila-schema.sql` and `sakila-data.sql` files.
2. Connect to the MySQL server using the **mysql** command-line client with the following command:

   ```
   $> mysql -u root -p
   ```

   Enter your password when prompted.

3. Execute the `sakila-schema.sql` script to create the database structure, and execute the `sakila-data.sql` script to populate the database structure, by using the following commands:

   ```
   mysql> SOURCE C:/temp/sakila-db/sakila-schema.sql;
   ```

   ```
   mysql> SOURCE C:/temp/sakila-db/sakila-data.sql;
   ```

   Replace the paths to the `sakila-schema.sql` and `sakila-data.sql` files with the actual paths on your system.

4. Confirm that the sample database is installed correctly. Execute the following statements. You should see output similar to that shown here.

```
mysql> USE sakila;
Database changed

mysql> SHOW FULL TABLES;
+----------------------------+------------+
| Tables_in_sakila           | Table_type |
+----------------------------+------------+
| actor                      | BASE TABLE |
| actor_info                 | VIEW       |
| address                    | BASE TABLE |
| category                   | BASE TABLE |
| city                       | BASE TABLE |
| country                    | BASE TABLE |
| customer                   | BASE TABLE |
| customer_list              | VIEW       |
| film                       | BASE TABLE |
| film_actor                 | BASE TABLE |
| film_category              | BASE TABLE |
| film_list                  | VIEW       |
| film_text                  | BASE TABLE |
| inventory                  | BASE TABLE |
| language                   | BASE TABLE |
| nicer_but_slower_film_list | VIEW       |
| payment                    | BASE TABLE |
| rental                     | BASE TABLE |
| sales_by_film_category     | VIEW       |
| sales_by_store             | VIEW       |
| staff                      | BASE TABLE |
| staff_list                 | VIEW       |
| store                      | BASE TABLE |
+----------------------------+------------+
23 rows in set (0.01 sec)
```

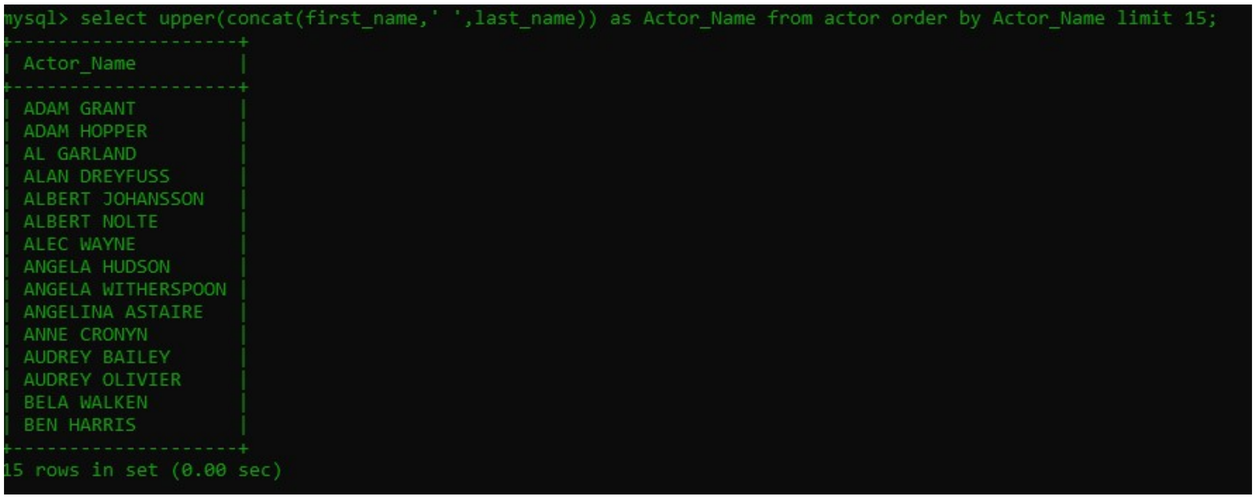
```
mysql> SELECT COUNT(*) FROM film;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM film_text;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

Tables

`https://dev.mysql.com/doc/sakila/en/sakila-structure-tables.html`

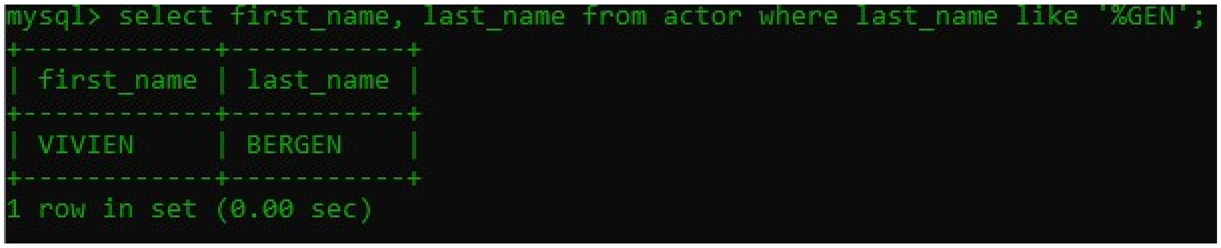
Exercises

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



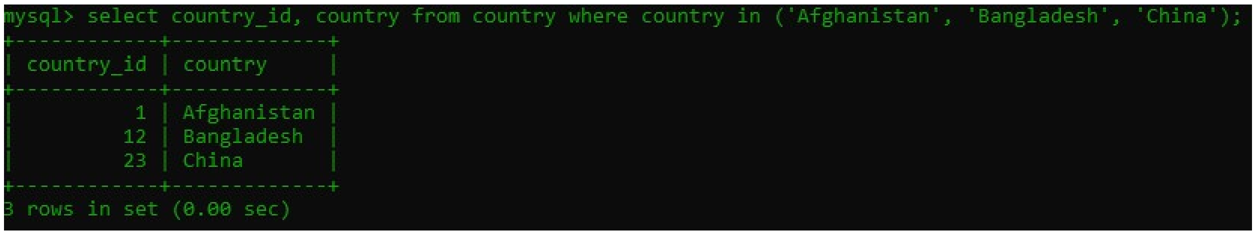
```
mysql> select upper(concat(first_name,' ',last_name)) as Actor_Name from actor order by Actor_Name limit 15;
+--------------------+
| Actor_Name         |
+--------------------+
| ADAM GRANT         |
| ADAM HOPPER        |
| AL GARLAND         |
| ALAN DREYFUSS      |
| ALBERT JOHANSSON   |
| ALBERT NOLTE       |
| ALEC WAYNE         |
| ANGELA HUDSON      |
| ANGELA WITHERSPOON |
| ANGELINA ASTAIRE   |
| ANNE CRONYN        |
| AUDREY BAILEY      |
| AUDREY OLIVIER     |
| BELA WALKEN        |
| BEN HARRIS         |
+--------------------+
15 rows in set (0.00 sec)
```

2. Find all actors whose last name contain the letters GEN:



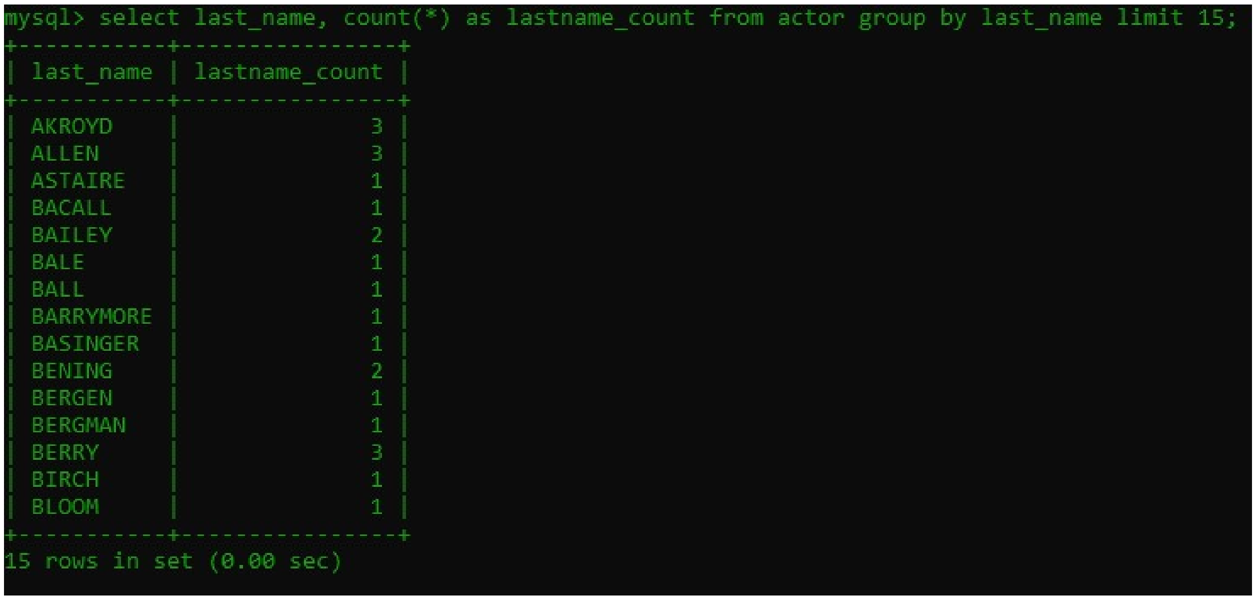
```
mysql> select first_name, last_name from actor where last_name like '%GEN';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| VIVIEN     | BERGEN    |
+------------+-----------+
1 row in set (0.00 sec)
```

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



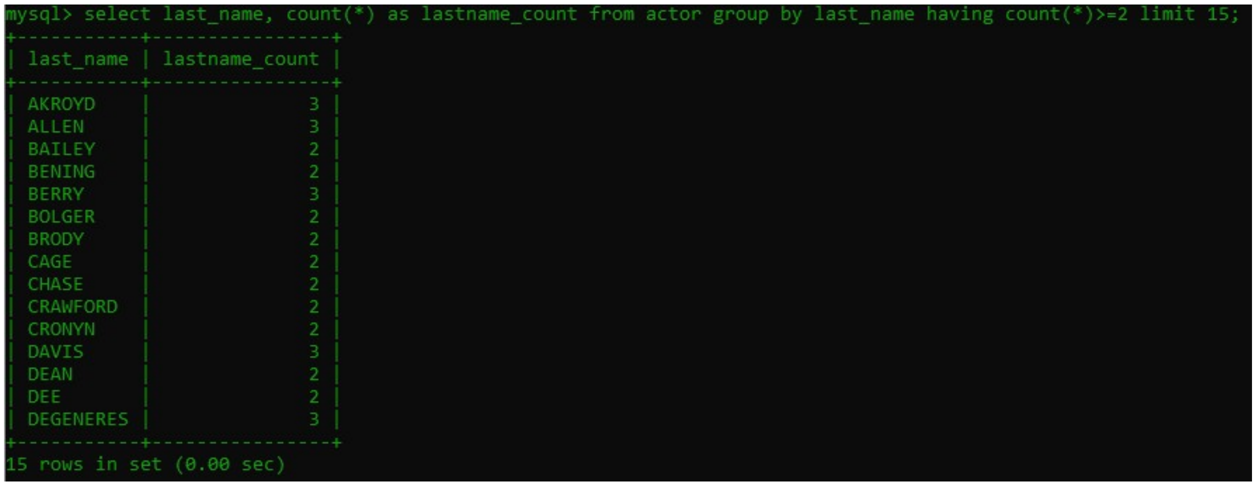
```
mysql> select country_id, country from country where country in ('Afghanistan', 'Bangladesh', 'China');
+------------+-------------+
| country_id | country     |
+------------+-------------+
|          1 | Afghanistan |
|         12 | Bangladesh  |
|         23 | China       |
+------------+-------------+
3 rows in set (0.00 sec)
```

4. List the last names of actors, as well as how many actors have that last name.



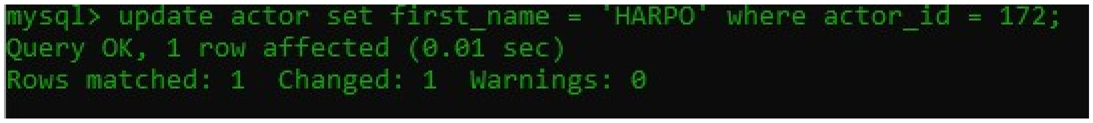
```
mysql> select last_name, count(*) as lastname_count from actor group by last_name limit 15;
+-----------+----------------+
| last_name | lastname_count |
+-----------+----------------+
| AKROYD    |              3 |
| ALLEN     |              3 |
| ASTAIRE   |              1 |
| BACALL    |              1 |
| BAILEY    |              2 |
| BALE      |              1 |
| BALL      |              1 |
| BARRYMORE |              1 |
| BASINGER  |              1 |
| BENING    |              2 |
| BERGEN    |              1 |
| BERGMAN   |              1 |
| BERRY     |              3 |
| BIRCH     |              1 |
| BLOOM     |              1 |
+-----------+----------------+
15 rows in set (0.00 sec)
```

5. 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



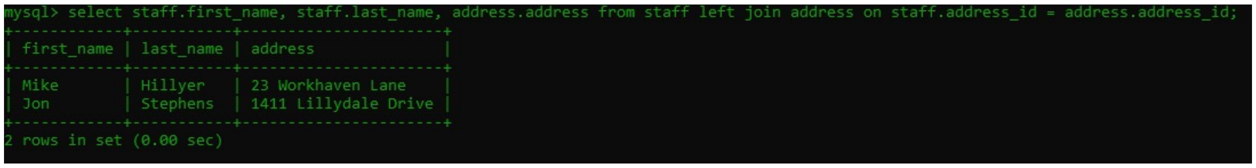
```
mysql> select last_name, count(*) as lastname_count from actor group by last_name having count(*)>=2 limit 15;
+-----------+----------------+
| last_name | lastname_count |
+-----------+----------------+
| AKROYD    |              3 |
| ALLEN     |              3 |
| BAILEY    |              2 |
| BENING    |              2 |
| BERRY     |              3 |
| BOLGER    |              2 |
| BRODY     |              2 |
| CAGE      |              2 |
| CHASE     |              2 |
| CRAWFORD  |              2 |
| CRONYN    |              2 |
| DAVIS     |              3 |
| DEAN      |              2 |
| DEE       |              2 |
| DEGENERES |              3 |
+-----------+----------------+
15 rows in set (0.00 sec)
```

6. The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS. Write a query to fix the record.



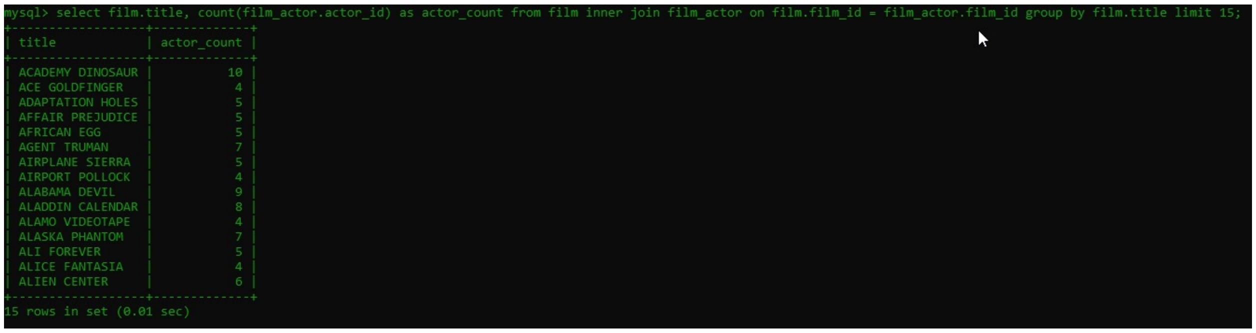
```
mysql> update actor set first_name = 'HARPO' where actor_id = 172;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

7. Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables staff and address:



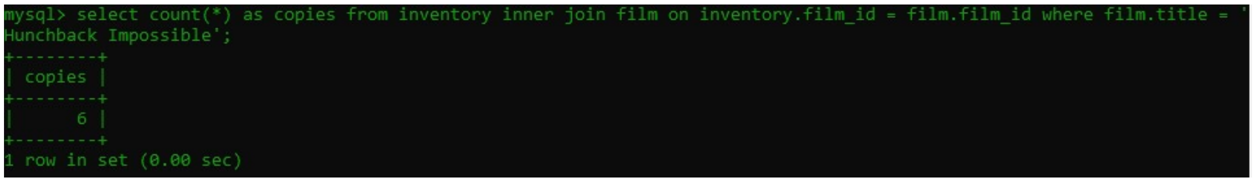
```
mysql> select staff.first_name, staff.last_name, address.address from staff left join address on staff.address_id = address.address_id;
+------------+-----------+----------------------+
| first_name | last_name | address              |
+------------+-----------+----------------------+
| Mike       | Hillyer   | 23 Workhaven Lane    |
| Jon        | Stephens  | 1411 Lillydale Drive |
+------------+-----------+----------------------+
2 rows in set (0.00 sec)
```

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



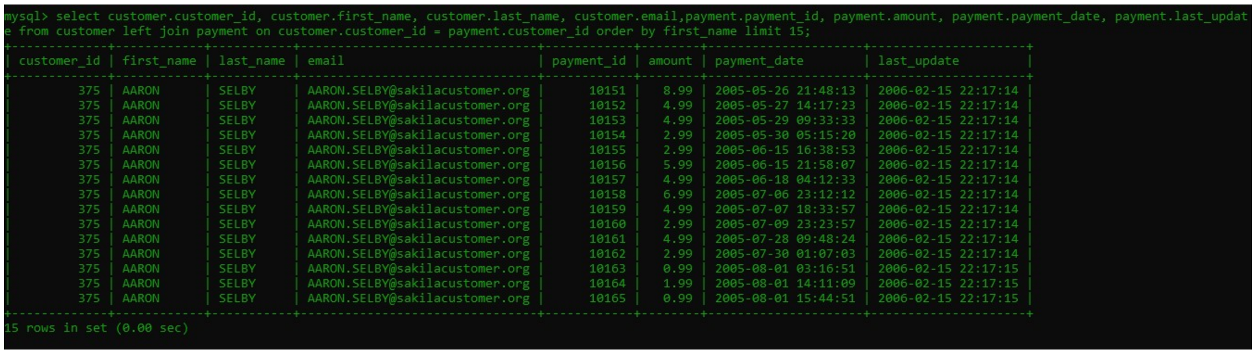
```
mysql> select film.title, count(film_actor.actor_id) as actor_count from film inner join film_actor on film.film_id = film_actor.film_id group by film.title limit 15;
+------------------+-------------+
| title            | actor_count |
+------------------+-------------+
| ACADEMY DINOSAUR |          10 |
| ACE GOLDFINGER   |           4 |
| ADAPTATION HOLES |           5 |
| AFFAIR PREJUDICE |           5 |
| AFRICAN EGG      |           5 |
| AGENT TRUMAN     |           7 |
| AIRPLANE SIERRA  |           5 |
| AIRPORT POLLOCK  |           4 |
| ALABAMA DEVIL    |           9 |
| ALADDIN CALENDAR |           8 |
| ALAMO VIDEOTAPE  |           4 |
| ALASKA PHANTOM   |           7 |
| ALI FOREVER      |           5 |
| ALICE FANTASIA   |           4 |
| ALIEN CENTER     |           6 |
+------------------+-------------+
15 rows in set (0.01 sec)
```

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



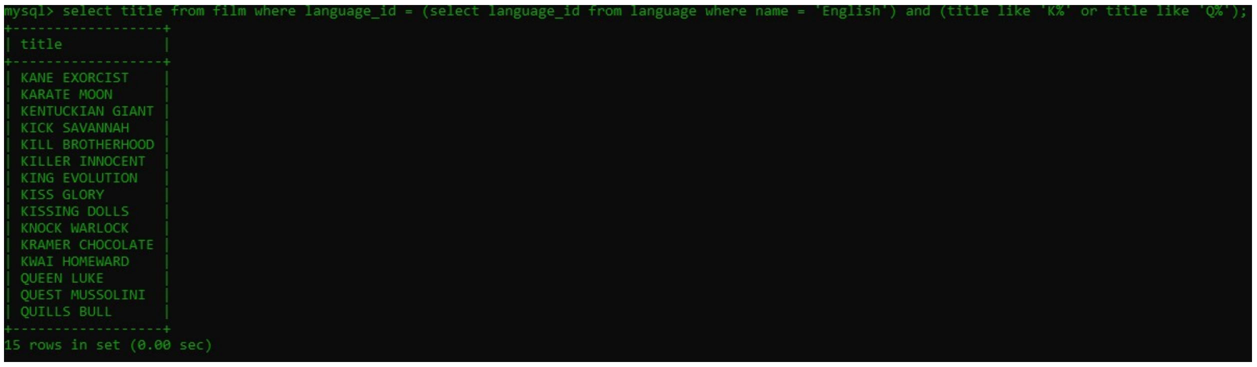
```
mysql> select count(*) as copies from inventory inner join film on inventory.film_id = film.film_id where film.title = '
Hunchback Impossible';
+--------+
| copies |
+--------+
|      6 |
+--------+
1 row in set (0.00 sec)
```

10. Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically by last name



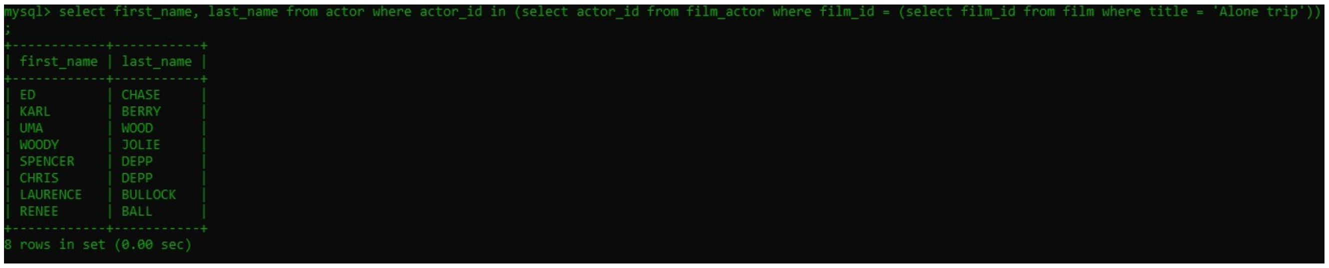
```
mysql> select customer.customer_id, customer.first_name, customer.last_name, customer.email,payment.payment_id, payment.amount, payment.payment_date, payment.last_updat
e from customer left join payment on customer.customer_id = payment.customer_id order by first_name limit 15;
+-------------+------------+-----------+----------------------------------+------------+--------+---------------------+---------------------+
| customer_id | first_name | last_name | email                            | payment_id | amount | payment_date        | last_update         |
+-------------+------------+-----------+----------------------------------+------------+--------+---------------------+---------------------+
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10151 |   8.99 | 2005-05-26 21:48:13 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10152 |   4.99 | 2005-05-27 14:17:23 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10153 |   4.99 | 2005-05-29 09:33:33 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10154 |   2.99 | 2005-05-30 05:15:20 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10155 |   2.99 | 2005-06-15 16:38:53 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10156 |   5.99 | 2005-06-15 21:58:07 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10157 |   4.99 | 2005-06-18 04:12:33 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10158 |   6.99 | 2005-07-06 23:12:12 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10159 |   4.99 | 2005-07-07 18:33:57 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10160 |   2.99 | 2005-07-09 23:23:57 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10161 |   4.99 | 2005-07-28 09:48:24 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10162 |   2.99 | 2005-07-30 01:07:03 | 2006-02-15 22:17:14 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10163 |   0.99 | 2005-08-01 03:16:51 | 2006-02-15 22:17:15 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10164 |   1.99 | 2005-08-01 14:11:09 | 2006-02-15 22:17:15 |
|         375 | AARON      | SELBY     | AARON.SELBY@sakilacustomer.org |      10165 |   0.99 | 2005-08-01 15:44:51 | 2006-02-15 22:17:15 |
+-------------+------------+-----------+----------------------------------+------------+--------+---------------------+---------------------+
15 rows in set (0.00 sec)
```

11. 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.



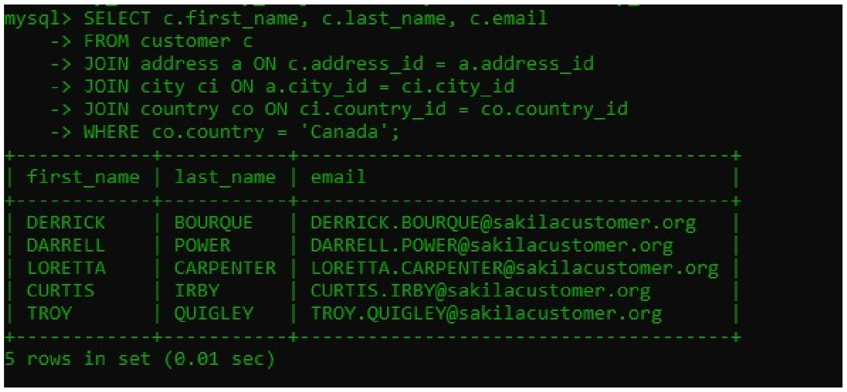
```
mysql> select title from film where language_id = (select language_id from language where name = 'English') and (title like 'K%' or title like 'Q%');
+------------------+
| title            |
+------------------+
| KANE EXORCIST    |
| KARATE MOON      |
| KENTUCKIAN GIANT |
| KICK SAVANNAH    |
| KILL BROTHERHOOD |
| KILLER INNOCENT  |
| KING EVOLUTION   |
| KISS GLORY       |
| KISSING DOLLS    |
| KNOCK WARLOCK    |
| KRAMER CHOCOLATE |
| KWAI HOMEWARD    |
| QUEEN LUKE       |
| QUEST MUSSOLINI  |
| QUILLS BULL      |
+------------------+
15 rows in set (0.00 sec)
```

12. Use subqueries to display all actors who appear in the film `Alone Trip`.



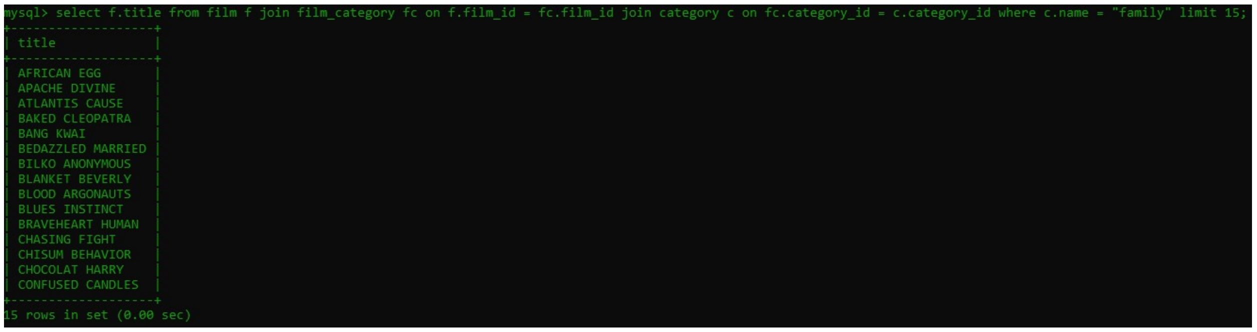
```
mysql> select first_name, last_name from actor where actor_id in (select actor_id from film_actor where film_id = (select film_id from film where title = 'Alone trip'))
;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| ED         | CHASE     |
| KARL       | BERRY     |
| UMA        | WOOD      |
| WOODY      | JOLIE     |
| SPENCER    | DEPP      |
| CHRIS      | DEPP      |
| LAURENCE   | BULLOCK   |
| RENEE      | BALL      |
+------------+-----------+
8 rows in set (0.00 sec)
```

13. You want to run an email marketing campaign in Canada, for which you will need the names and email addresses of all Canadian customers. Use joins to retrieve this information.



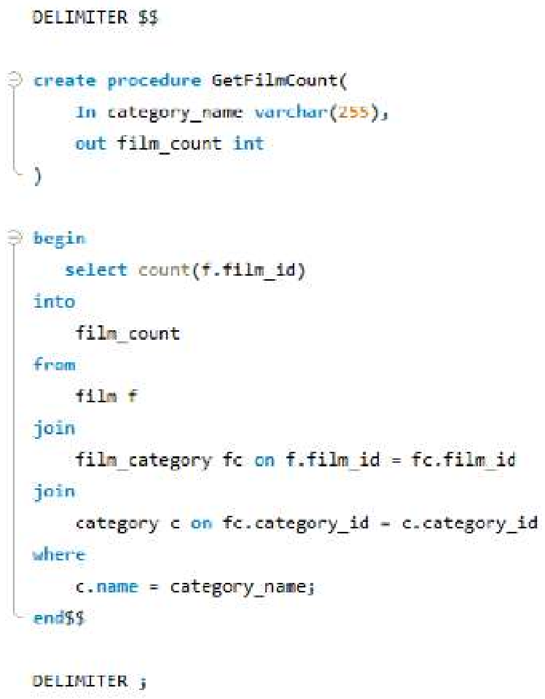
```
mysql> SELECT c.first_name, c.last_name, c.email
    -> FROM customer c
    -> JOIN address a ON c.address_id = a.address_id
    -> JOIN city ci ON a.city_id = ci.city_id
    -> JOIN country co ON ci.country_id = co.country_id
    -> WHERE co.country = 'Canada';
+------------+-----------+-------------------------------------+
| first_name | last_name | email                               |
+------------+-----------+-------------------------------------+
| DERRICK    | BOURQUE   | DERRICK.BOURQUE@sakilacustomer.org   |
| DARRELL    | POWER     | DARRELL.POWER@sakilacustomer.org     |
| LORETTA    | CARPENTER | LORETTA.CARPENTER@sakilacustomer.org |
| CURTIS     | IRBY      | CURTIS.IRBY@sakilacustomer.org       |
| TROY       | QUIGLEY   | TROY.QUIGLEY@sakilacustomer.org      |
+------------+-----------+-------------------------------------+
5 rows in set (0.01 sec)
```

14. Sales have been lagging among young families, and you wish to target all family movies for a promotion. Identify all movies categorized as famiy films.



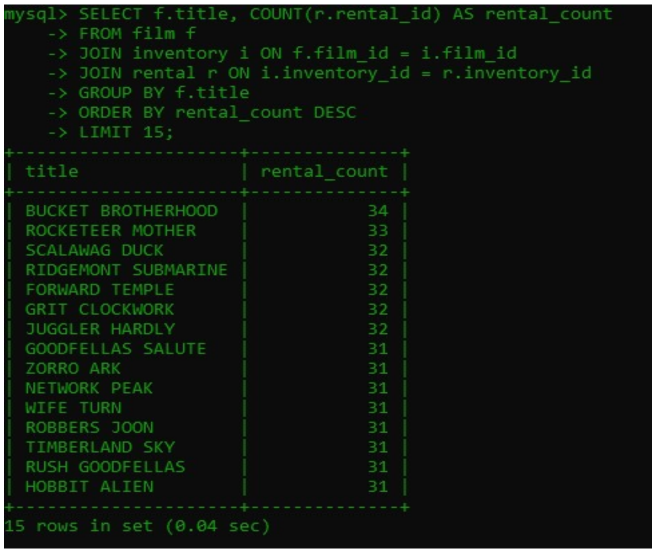
```
mysql> select f.title from film f join film_category fc on f.film_id = fc.film_id join category c on fc.category_id = c.category_id where c.name = "family" limit 15;
+-------------------+
| title             |
+-------------------+
| AFRICAN EGG       |
| APACHE DIVINE     |
| ATLANTIS CAUSE    |
| BAKED CLEOPATRA   |
| BANG KWAI         |
| BEDAZZLED MARRIED |
| BILKO ANONYMOUS   |
| BLANKET BEVERLY   |
| BLOOD ARGONAUTS   |
| BLUES INSTINCT    |
| BRAVEHEART HUMAN  |
| CHASING FIGHT     |
| CHISUM BEHAVIOR   |
| CHOCOLAT HARRY    |
| CONFUSED CANDLES  |
+-------------------+
15 rows in set (0.00 sec)
```

15. Create a Stored procedure to get the count of films in the input category (IN category_name, OUT count)



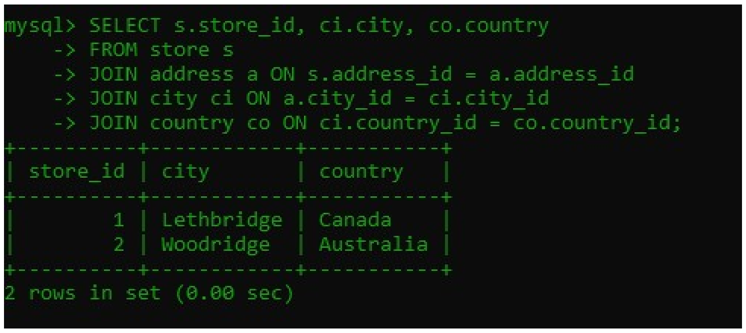
```
DELIMITER $$

create procedure GetFilmCount(
    In category_name varchar(255),
    out film_count int
)

begin
   select count(f.film_id)
into
    film_count
from
    film f
join
    film_category fc on f.film_id = fc.film_id
join
    category c on fc.category_id = c.category_id
where
    c.name = category_name;
end$$

DELIMITER ;
```

16. Display the most frequently rented movies in descending order.



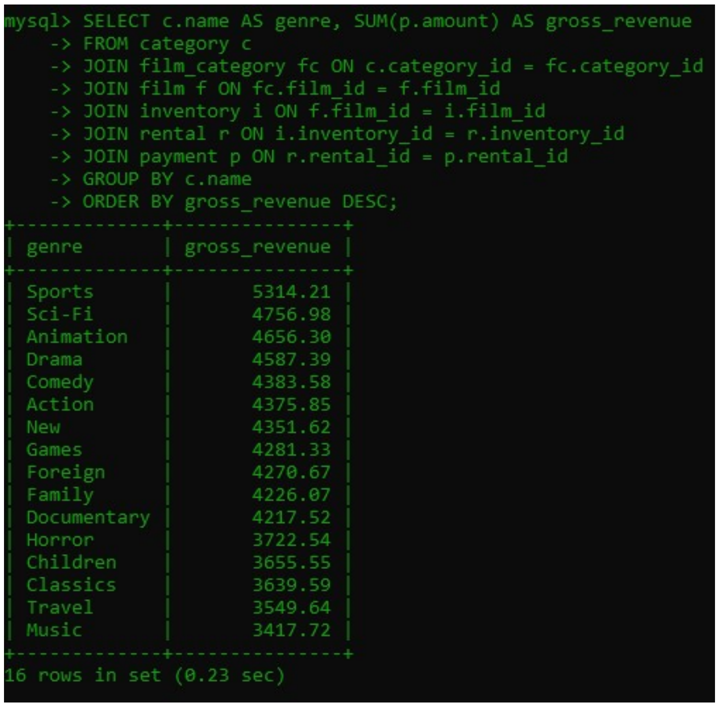
```
mysql> SELECT f.title, COUNT(r.rental_id) AS rental_count
    -> FROM film f
    -> JOIN inventory i ON f.film_id = i.film_id
    -> JOIN rental r ON i.inventory_id = r.inventory_id
    -> GROUP BY f.title
    -> ORDER BY rental_count DESC
    -> LIMIT 15;
+---------------------+--------------+
| title               | rental_count |
+---------------------+--------------+
| BUCKET BROTHERHOOD  |           34 |
| ROCKETEER MOTHER    |           33 |
| SCALAWAG DUCK       |           32 |
| RIDGEMONT SUBMARINE |           32 |
| FORWARD TEMPLE      |           32 |
| GRIT CLOCKWORK      |           32 |
| JUGGLER HARDLY      |           32 |
| GOODFELLAS SALUTE   |           31 |
| ZORRO ARK           |           31 |
| NETWORK PEAK        |           31 |
| WIFE TURN           |           31 |
| ROBBERS JOON        |           31 |
| TIMBERLAND SKY      |           31 |
| RUSH GOODFELLAS     |           31 |
| HOBBIT ALIEN        |           31 |
+---------------------+--------------+
15 rows in set (0.04 sec)
```

17. Write a query to display for each store its store ID, city, and country.



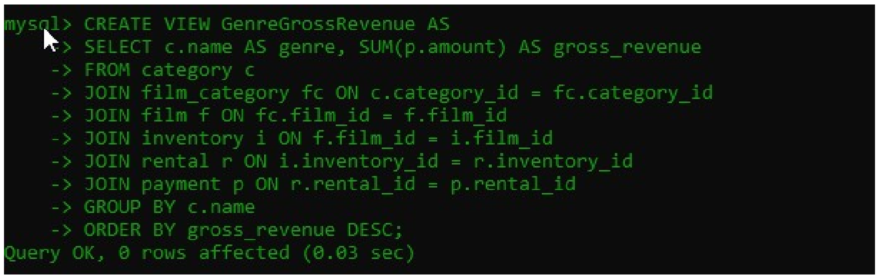
```
mysql> SELECT s.store_id, ci.city, co.country
    -> FROM store s
    -> JOIN address a ON s.address_id = a.address_id
    -> JOIN city ci ON a.city_id = ci.city_id
    -> JOIN country co ON ci.country_id = co.country_id;
+----------+------------+-----------+
| store_id | city       | country   |
+----------+------------+-----------+
|        1 | Lethbridge | Canada    |
|        2 | Woodridge  | Australia |
+----------+------------+-----------+
2 rows in set (0.00 sec)
```

18. List the genres and its gross revenue.

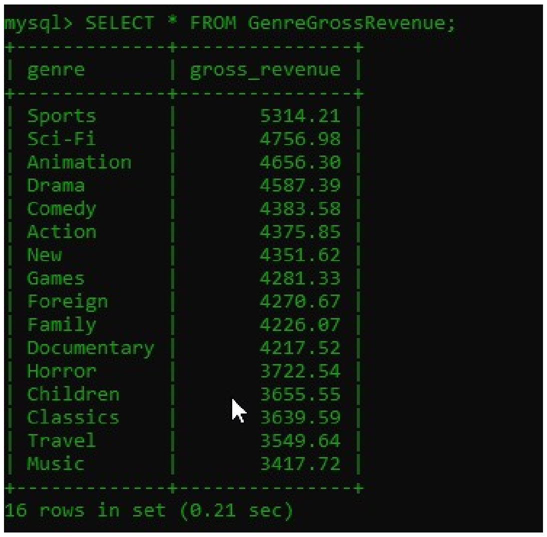


```
mysql> SELECT c.name AS genre, SUM(p.amount) AS gross_revenue
    -> FROM category c
    -> JOIN film_category fc ON c.category_id = fc.category_id
    -> JOIN film f ON fc.film_id = f.film_id
    -> JOIN inventory i ON f.film_id = i.film_id
    -> JOIN rental r ON i.inventory_id = r.inventory_id
    -> JOIN payment p ON r.rental_id = p.rental_id
    -> GROUP BY c.name
    -> ORDER BY gross_revenue DESC;
+-------------+---------------+
| genre       | gross_revenue |
+-------------+---------------+
| Sports      |       5314.21 |
| Sci-Fi      |       4756.98 |
| Animation   |       4656.30 |
| Drama       |       4587.39 |
| Comedy      |       4383.58 |
| Action      |       4375.85 |
| New         |       4351.62 |
| Games       |       4281.33 |
| Foreign     |       4270.67 |
| Family      |       4226.07 |
| Documentary |       4217.52 |
| Horror      |       3722.54 |
| Children    |       3655.55 |
| Classics    |       3639.59 |
| Travel      |       3549.64 |
| Music       |       3417.72 |
+-------------+---------------+
16 rows in set (0.23 sec)
```

19. Create a View for the above query(18)

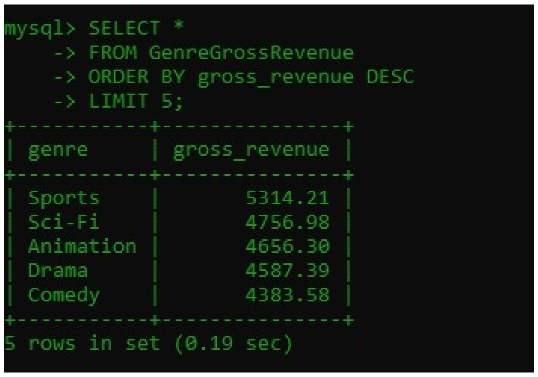


```
mysql> CREATE VIEW GenreGrossRevenue AS
    -> SELECT c.name AS genre, SUM(p.amount) AS gross_revenue
    -> FROM category c
    -> JOIN film_category fc ON c.category_id = fc.category_id
    -> JOIN film f ON fc.film_id = f.film_id
    -> JOIN inventory i ON f.film_id = i.film_id
    -> JOIN rental r ON i.inventory_id = r.inventory_id
    -> JOIN payment p ON r.rental_id = p.rental_id
    -> GROUP BY c.name
    -> ORDER BY gross_revenue DESC;
Query OK, 0 rows affected (0.03 sec)
```



```
mysql> SELECT * FROM GenreGrossRevenue;
+-------------+---------------+
| genre       | gross_revenue |
+-------------+---------------+
| Sports      |       5314.21 |
| Sci-Fi      |       4756.98 |
| Animation   |       4656.30 |
| Drama       |       4587.39 |
| Comedy      |       4383.58 |
| Action      |       4375.85 |
| New         |       4351.62 |
| Games       |       4281.33 |
| Foreign     |       4270.67 |
| Family      |       4226.07 |
| Documentary |       4217.52 |
| Horror      |       3722.54 |
| Children    |       3655.55 |
| Classics    |       3639.59 |
| Travel      |       3549.64 |
| Music       |       3417.72 |
+-------------+---------------+
16 rows in set (0.21 sec)
```

20. Select top 5 genres in gross revenue view.



```
mysql> SELECT *
    -> FROM GenreGrossRevenue
    -> ORDER BY gross_revenue DESC
    -> LIMIT 5;
+-----------+---------------+
| genre     | gross_revenue |
+-----------+---------------+
| Sports    |       5314.21 |
| Sci-Fi    |       4756.98 |
| Animation |       4656.30 |
| Drama     |       4587.39 |
| Comedy    |       4383.58 |
+-----------+---------------+
5 rows in set (0.19 sec)
```