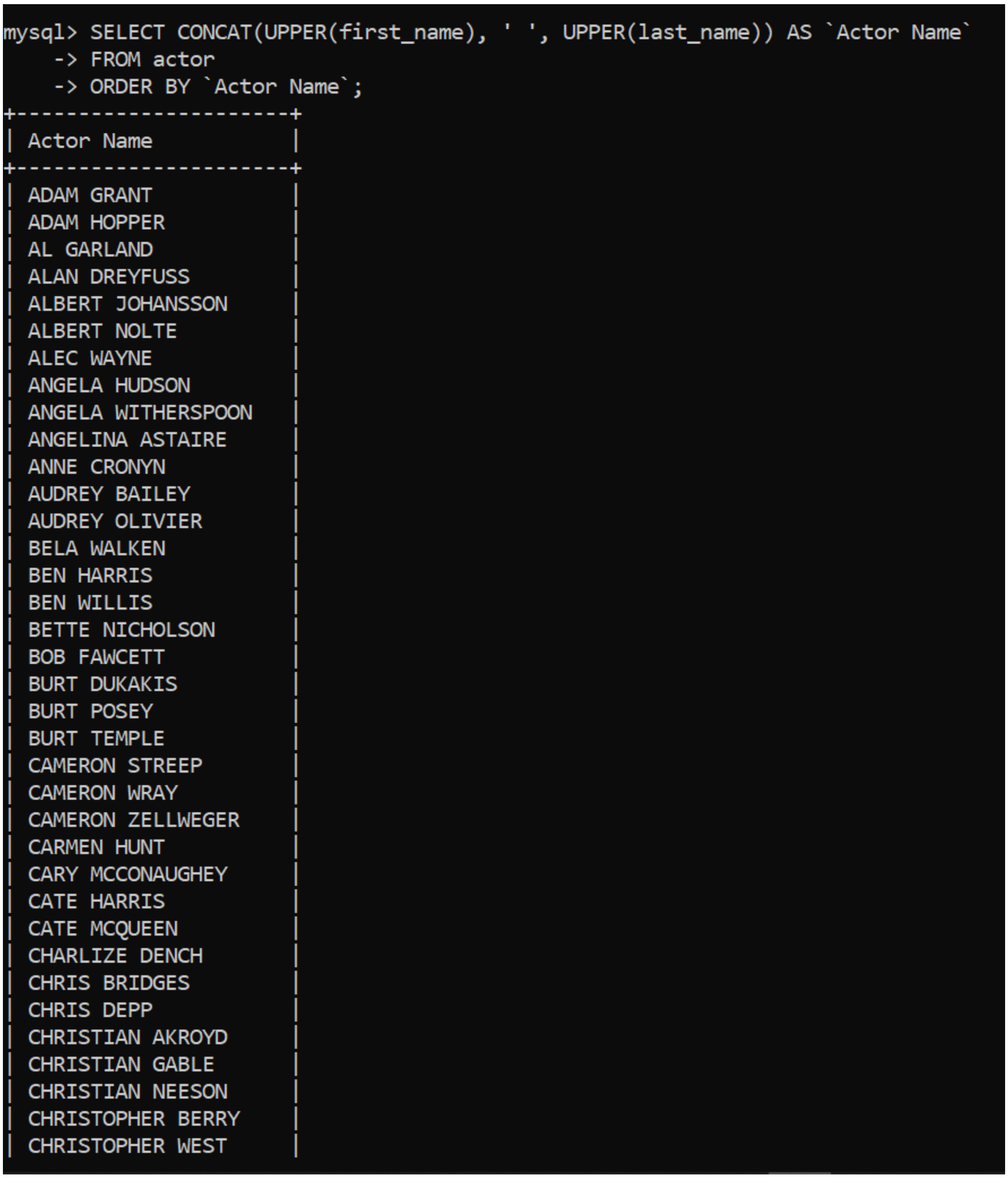
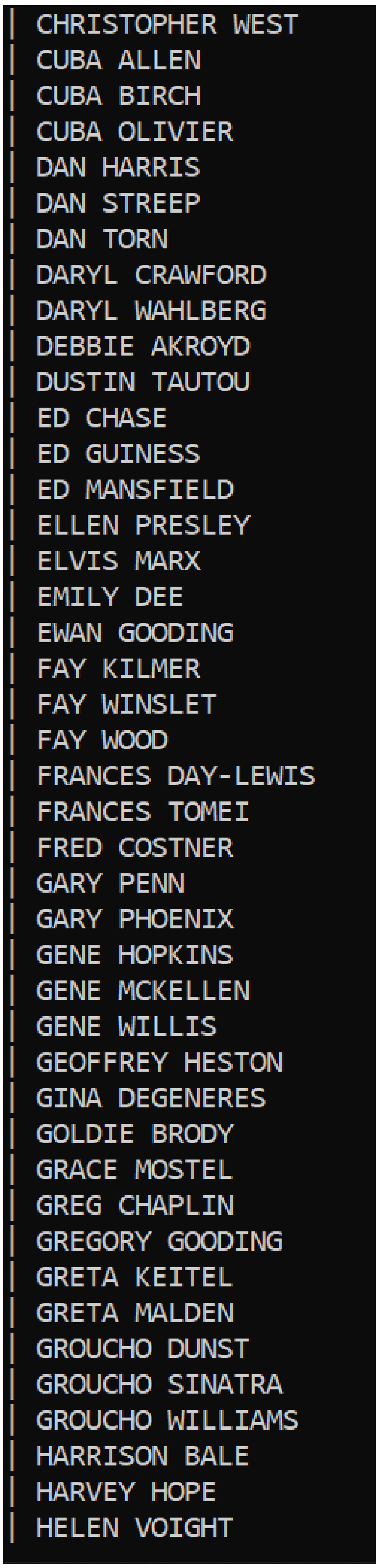
Final Assessment - Sakila db

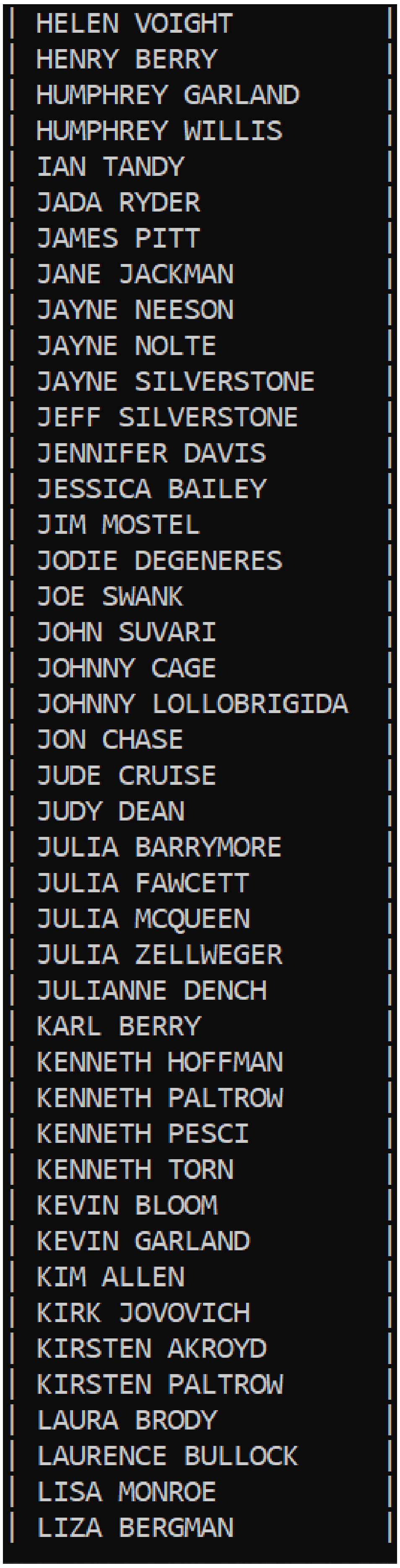
1. **Display the first and last name of each actor in a single column in upper case letters in alphabetic order. Name the column Actor Name.**



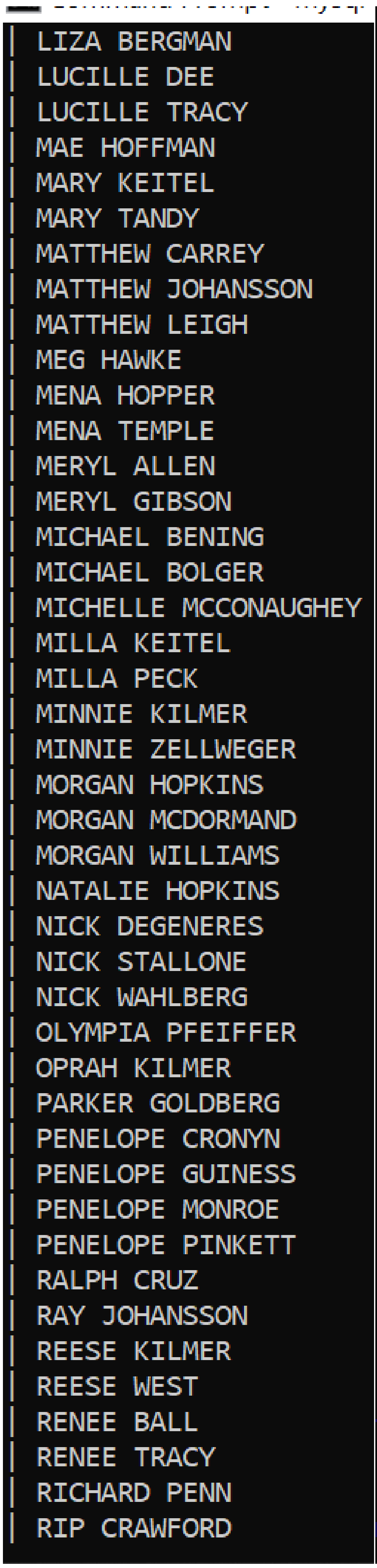
```
mysql> SELECT CONCAT(UPPER(first_name), ' ', UPPER(last_name)) AS `Actor Name`
    -> FROM actor
    -> ORDER BY `Actor Name`;
+----------------------+
| Actor Name           |
+----------------------+
| ADAM GRANT           |
| ADAM HOPPER          |
| AL GARLAND           |
| ALAN DREYFUSS        |
| ALBERT JOHANSSON     |
| ALBERT NOLTE         |
| ALEC WAYNE           |
| ANGELA HUDSON        |
| ANGELA WITHERSPOON   |
| ANGELINA ASTAIRE     |
| ANNE CRONYN          |
| AUDREY BAILEY        |
| AUDREY OLIVIER       |
| BELA WALKEN          |
| BEN HARRIS           |
| BEN WILLIS           |
| BETTE NICHOLSON      |
| BOB FAWCETT          |
| BURT DUKAKIS         |
| BURT POSEY           |
| BURT TEMPLE          |
| CAMERON STREEP       |
| CAMERON WRAY         |
| CAMERON ZELLWEGER    |
| CARMEN HUNT          |
| CARY MCCONAUGHEY     |
| CATE HARRIS          |
| CATE MCQUEEN         |
| CHARLIZE DENCH       |
| CHRIS BRIDGES        |
| CHRIS DEPP           |
| CHRISTIAN AKROYD     |
| CHRISTIAN GABLE      |
| CHRISTIAN NEESON     |
| CHRISTOPHER BERRY    |
| CHRISTOPHER WEST     |
```



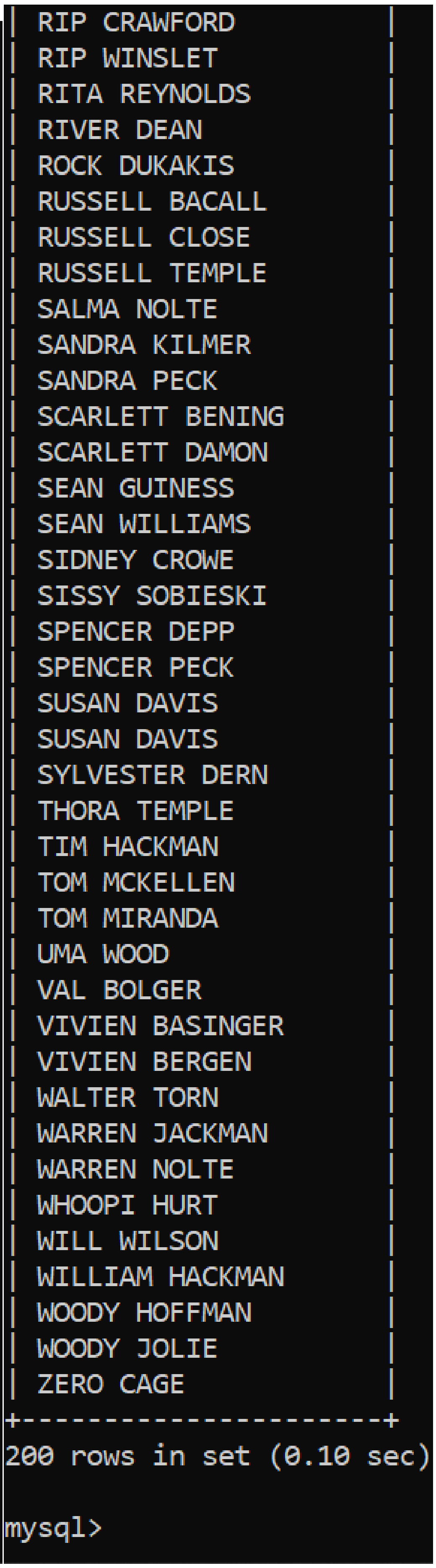
```
| CHRISTOPHER WEST
| CUBA ALLEN
| CUBA BIRCH
| CUBA OLIVIER
| DAN HARRIS
| DAN STREEP
| DAN TORN
| DARYL CRAWFORD
| DARYL WAHLBERG
| DEBBIE AKROYD
| DUSTIN TAUTOU
| ED CHASE
| ED GUINESS
| ED MANSFIELD
| ELLEN PRESLEY
| ELVIS MARX
| EMILY DEE
| EWAN GOODING
| FAY KILMER
| FAY WINSLET
| FAY WOOD
| FRANCES DAY-LEWIS
| FRANCES TOMEI
| FRED COSTNER
| GARY PENN
| GARY PHOENIX
| GENE HOPKINS
| GENE MCKELLEN
| GENE WILLIS
| GEOFFREY HESTON
| GINA DEGENERES
| GOLDIE BRODY
| GRACE MOSTEL
| GREG CHAPLIN
| GREGORY GOODING
| GRETA KEITEL
| GRETA MALDEN
| GROUCHO DUNST
| GROUCHO SINATRA
| GROUCHO WILLIAMS
| HARRISON BALE
| HARVEY HOPE
| HELEN VOIGHT
```



```
| HELEN VOIGHT          |
| HENRY BERRY           |
| HUMPHREY GARLAND      |
| HUMPHREY WILLIS       |
| IAN TANDY             |
| JADA RYDER            |
| JAMES PITT            |
| JANE JACKMAN          |
| JAYNE NEESON          |
| JAYNE NOLTE           |
| JAYNE SILVERSTONE     |
| JEFF SILVERSTONE      |
| JENNIFER DAVIS        |
| JESSICA BAILEY        |
| JIM MOSTEL            |
| JODIE DEGENERES       |
| JOE SWANK             |
| JOHN SUVARI           |
| JOHNNY CAGE           |
| JOHNNY LOLLOBRIGIDA   |
| JON CHASE             |
| JUDE CRUISE           |
| JUDY DEAN             |
| JULIA BARRYMORE       |
| JULIA FAWCETT         |
| JULIA MCQUEEN         |
| JULIA ZELLWEGER       |
| JULIANNE DENCH        |
| KARL BERRY            |
| KENNETH HOFFMAN       |
| KENNETH PALTROW       |
| KENNETH PESCI         |
| KENNETH TORN          |
| KEVIN BLOOM           |
| KEVIN GARLAND         |
| KIM ALLEN             |
| KIRK JOVOVICH         |
| KIRSTEN AKROYD        |
| KIRSTEN PALTROW       |
| LAURA BRODY           |
| LAURENCE BULLOCK      |
| LISA MONROE           |
| LIZA BERGMAN          |
```

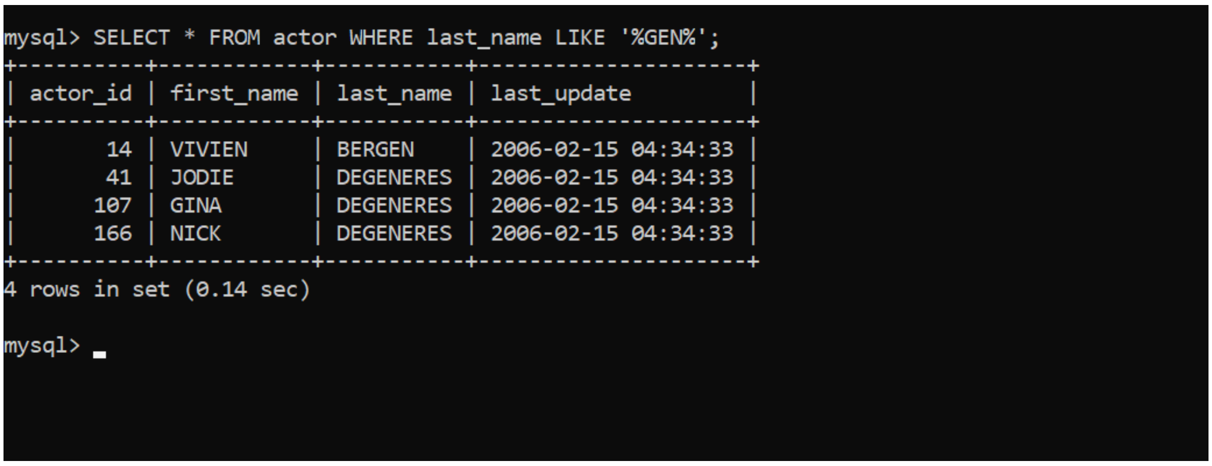


```
| LIZA BERGMAN
| LUCILLE DEE
| LUCILLE TRACY
| MAE HOFFMAN
| MARY KEITEL
| MARY TANDY
| MATTHEW CARREY
| MATTHEW JOHANSSON
| MATTHEW LEIGH
| MEG HAWKE
| MENA HOPPER
| MENA TEMPLE
| MERYL ALLEN
| MERYL GIBSON
| MICHAEL BENING
| MICHAEL BOLGER
| MICHELLE MCCONAUGHEY
| MILLA KEITEL
| MILLA PECK
| MINNIE KILMER
| MINNIE ZELLWEGER
| MORGAN HOPKINS
| MORGAN MCDORMAND
| MORGAN WILLIAMS
| NATALIE HOPKINS
| NICK DEGENERES
| NICK STALLONE
| NICK WAHLBERG
| OLYMPIA PFEIFFER
| OPRAH KILMER
| PARKER GOLDBERG
| PENELOPE CRONYN
| PENELOPE GUINESS
| PENELOPE MONROE
| PENELOPE PINKETT
| RALPH CRUZ
| RAY JOHANSSON
| REESE KILMER
| REESE WEST
| RENEE BALL
| RENEE TRACY
| RICHARD PENN
| RIP CRAWFORD
```



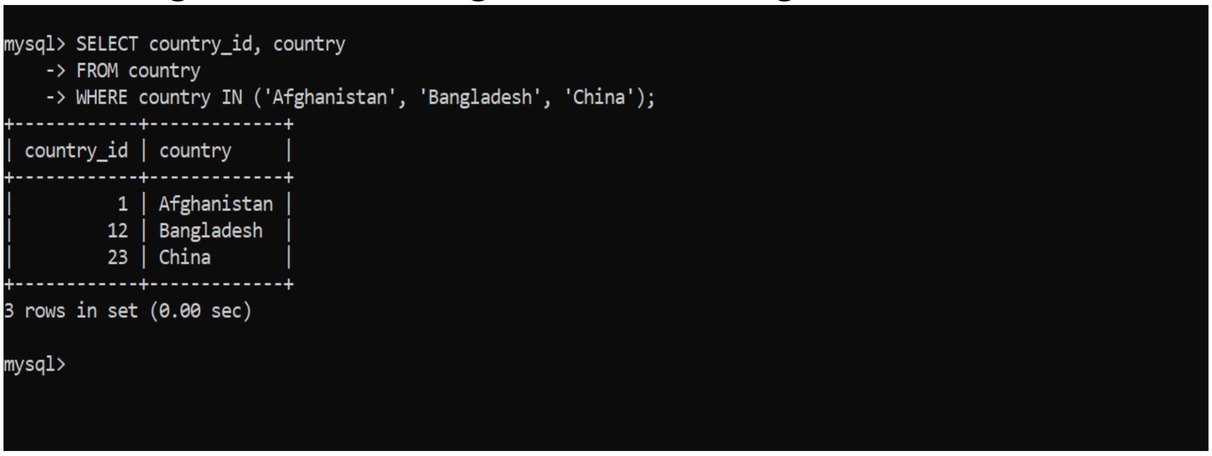
```
| RIP CRAWFORD          |
| RIP WINSLET           |
| RITA REYNOLDS         |
| RIVER DEAN            |
| ROCK DUKAKIS          |
| RUSSELL BACALL        |
| RUSSELL CLOSE         |
| RUSSELL TEMPLE        |
| SALMA NOLTE           |
| SANDRA KILMER         |
| SANDRA PECK           |
| SCARLETT BENING       |
| SCARLETT DAMON        |
| SEAN GUINESS          |
| SEAN WILLIAMS         |
| SIDNEY CROWE          |
| SISSY SOBIESKI        |
| SPENCER DEPP          |
| SPENCER PECK          |
| SUSAN DAVIS           |
| SUSAN DAVIS           |
| SYLVESTER DERN        |
| THORA TEMPLE          |
| TIM HACKMAN           |
| TOM MCKELLEN          |
| TOM MIRANDA           |
| UMA WOOD              |
| VAL BOLGER            |
| VIVIEN BASINGER       |
| VIVIEN BERGEN         |
| WALTER TORN           |
| WARREN JACKMAN        |
| WARREN NOLTE          |
| WHOOPI HURT           |
| WILL WILSON           |
| WILLIAM HACKMAN       |
| WOODY HOFFMAN         |
| WOODY JOLIE           |
| ZERO CAGE             |
+-----------------------+
200 rows in set (0.10 sec)

mysql>
```

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



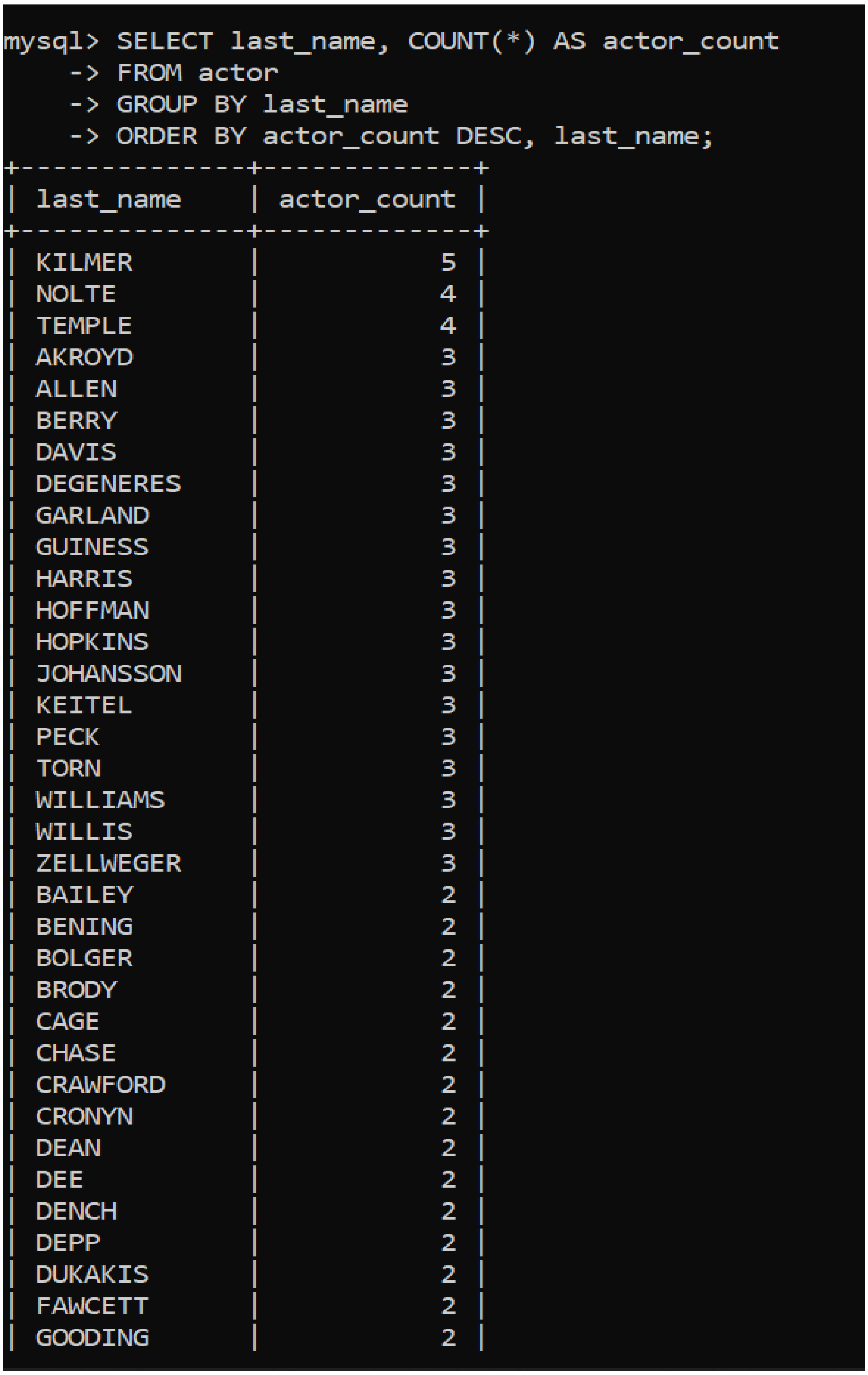
```
mysql> SELECT * FROM actor WHERE last_name LIKE '%GEN%';
+----------+------------+-----------+---------------------+
| actor_id | first_name | last_name | last_update         |
+----------+------------+-----------+---------------------+
|       14 | VIVIEN     | BERGEN    | 2006-02-15 04:34:33 |
|       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 |
+----------+------------+-----------+---------------------+
4 rows in set (0.14 sec)

mysql> _
```

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

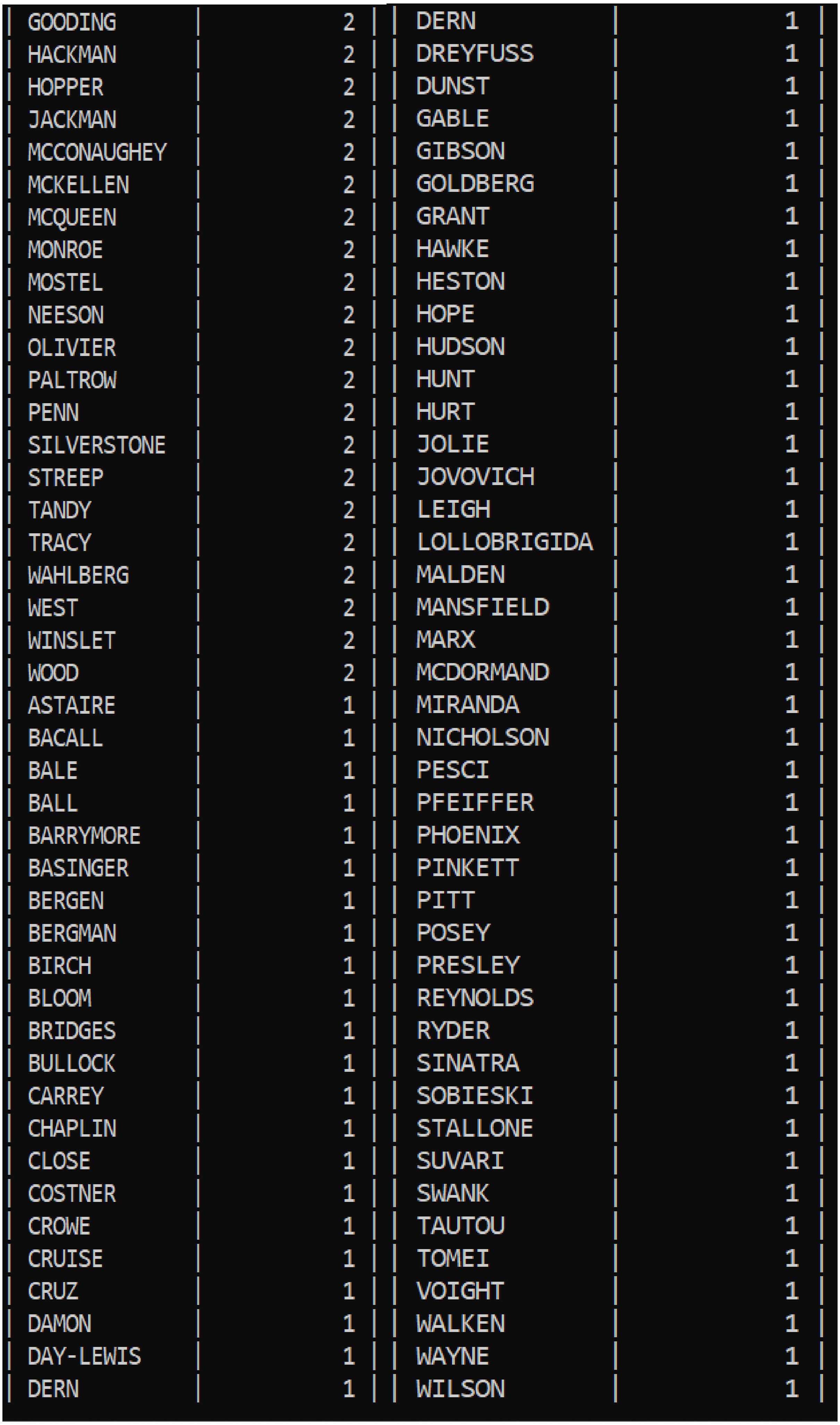


```
mysql> SELECT country_id, country
    -> FROM country
    -> WHERE country IN ('Afghanistan', 'Bangladesh', 'China');
+------------+-------------+
| country_id | country     |
+------------+-------------+
|          1 | Afghanistan |
|         12 | Bangladesh  |
|         23 | China       |
+------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

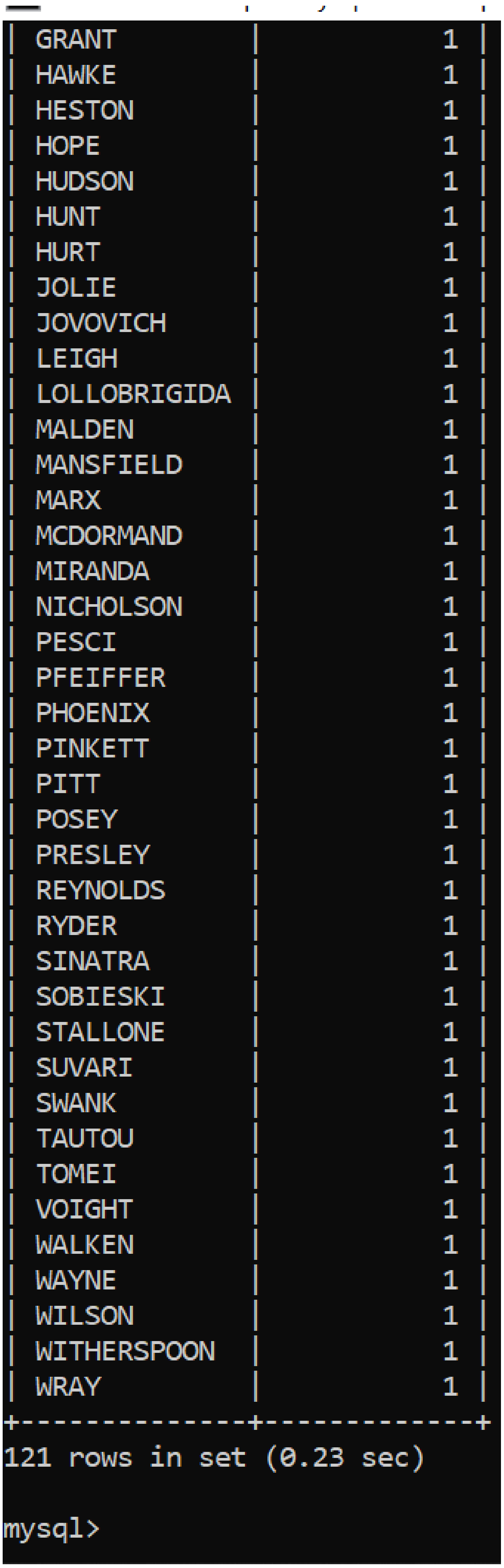
4. List the last names of actors, as well as how many actors have that last name.



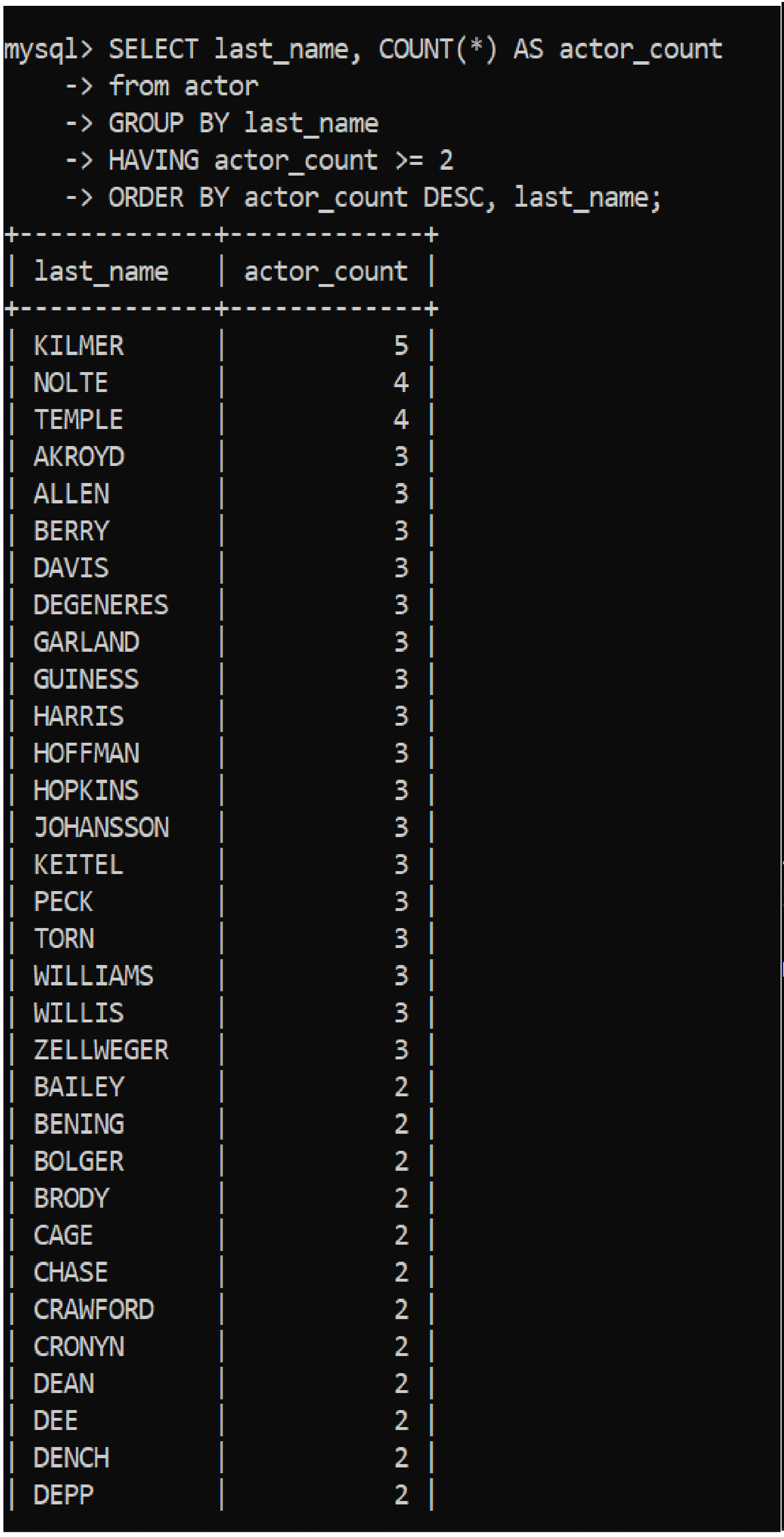
```
mysql> SELECT last_name, COUNT(*) AS actor_count
    -> FROM actor
    -> GROUP BY last_name
    -> ORDER BY actor_count DESC, last_name;
+--------------+-------------+
| last_name    | actor_count |
+--------------+-------------+
| KILMER       |           5 |
| NOLTE        |           4 |
| TEMPLE       |           4 |
| AKROYD       |           3 |
| ALLEN        |           3 |
| BERRY        |           3 |
| DAVIS        |           3 |
| DEGENERES    |           3 |
| GARLAND      |           3 |
| GUINESS      |           3 |
| HARRIS       |           3 |
| HOFFMAN      |           3 |
| HOPKINS      |           3 |
| JOHANSSON    |           3 |
| KEITEL       |           3 |
| PECK         |           3 |
| TORN         |           3 |
| WILLIAMS     |           3 |
| WILLIS       |           3 |
| ZELLWEGER    |           3 |
| BAILEY       |           2 |
| BENING       |           2 |
| BOLGER       |           2 |
| BRODY        |           2 |
| CAGE         |           2 |
| CHASE        |           2 |
| CRAWFORD     |           2 |
| CRONYN       |           2 |
| DEAN         |           2 |
| DEE          |           2 |
| DENCH        |           2 |
| DEPP         |           2 |
| DUKAKIS      |           2 |
| FAWCETT      |           2 |
| GOODING      |           2 |
```



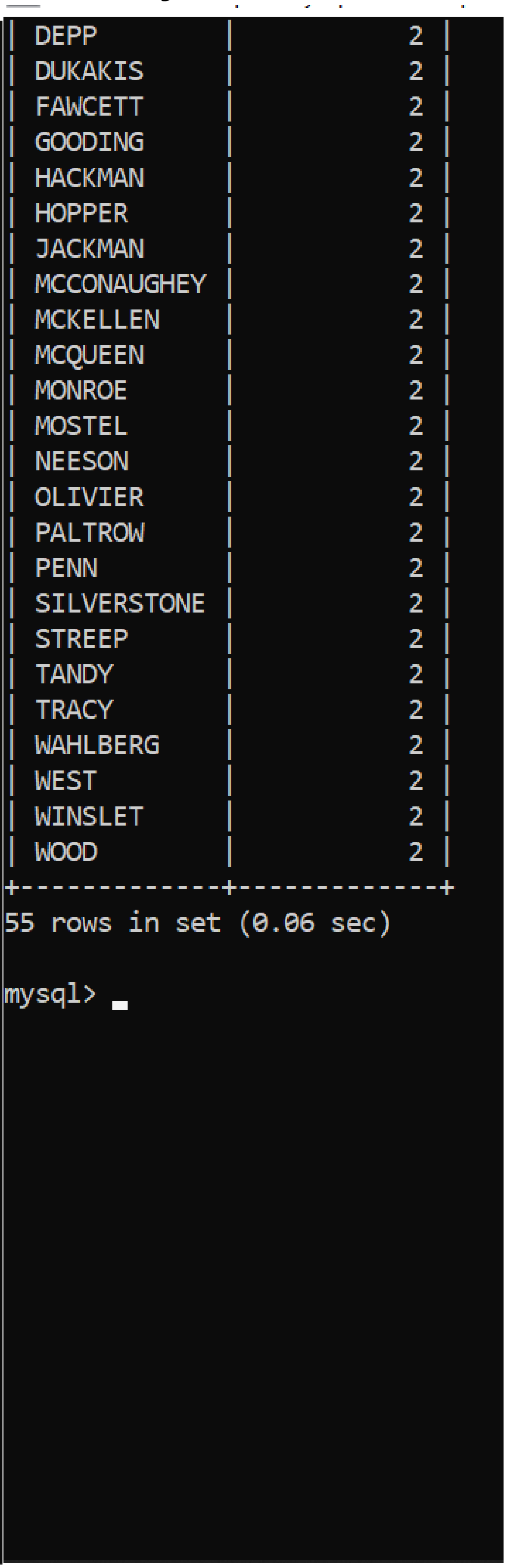
```
| GOODING      |          2 | | DERN          |          1 |
| HACKMAN      |          2 | | DREYFUSS      |          1 |
| HOPPER       |          2 | | DUNST         |          1 |
| JACKMAN      |          2 | | GABLE         |          1 |
| MCCONAUGHEY  |          2 | | GIBSON        |          1 |
| MCKELLEN     |          2 | | GOLDBERG      |          1 |
| MCQUEEN      |          2 | | GRANT         |          1 |
| MONROE       |          2 | | HAWKE         |          1 |
| MOSTEL       |          2 | | HESTON        |          1 |
| NEESON       |          2 | | HOPE          |          1 |
| OLIVIER      |          2 | | HUDSON        |          1 |
| PALTROW      |          2 | | HUNT          |          1 |
| PENN         |          2 | | HURT          |          1 |
| SILVERSTONE  |          2 | | JOLIE         |          1 |
| STREEP       |          2 | | JOVOVICH      |          1 |
| TANDY        |          2 | | LEIGH         |          1 |
| TRACY        |          2 | | LOLLOBRIGIDA  |          1 |
| WAHLBERG     |          2 | | MALDEN        |          1 |
| WEST         |          2 | | MANSFIELD     |          1 |
| WINSLET      |          2 | | MARX          |          1 |
| WOOD         |          2 | | MCDORMAND     |          1 |
| ASTAIRE      |          1 | | MIRANDA       |          1 |
| BACALL       |          1 | | NICHOLSON     |          1 |
| BALE         |          1 | | PESCI         |          1 |
| BALL         |          1 | | PFEIFFER      |          1 |
| BARRYMORE    |          1 | | PHOENIX       |          1 |
| BASINGER     |          1 | | PINKETT       |          1 |
| BERGEN       |          1 | | PITT          |          1 |
| BERGMAN      |          1 | | POSEY         |          1 |
| BIRCH        |          1 | | PRESLEY       |          1 |
| BLOOM        |          1 | | REYNOLDS      |          1 |
| BRIDGES      |          1 | | RYDER         |          1 |
| BULLOCK      |          1 | | SINATRA       |          1 |
| CARREY       |          1 | | SOBIESKI      |          1 |
| CHAPLIN      |          1 | | STALLONE      |          1 |
| CLOSE        |          1 | | SUVARI        |          1 |
| COSTNER      |          1 | | SWANK         |          1 |
| CROWE        |          1 | | TAUTOU        |          1 |
| CRUISE       |          1 | | TOMEI         |          1 |
| CRUZ         |          1 | | VOIGHT        |          1 |
| DAMON        |          1 | | WALKEN        |          1 |
| DAY-LEWIS    |          1 | | WAYNE         |          1 |
| DERN         |          1 | | WILSON        |          1 |
```


```
| GRANT        |           1 |
| HAWKE        |           1 |
| HESTON       |           1 |
| HOPE         |           1 |
| HUDSON       |           1 |
| HUNT         |           1 |
| HURT         |           1 |
| JOLIE        |           1 |
| JOVOVICH     |           1 |
| LEIGH        |           1 |
| LOLLOBRIGIDA |           1 |
| MALDEN       |           1 |
| MANSFIELD    |           1 |
| MARX         |           1 |
| MCDORMAND    |           1 |
| MIRANDA      |           1 |
| NICHOLSON    |           1 |
| PESCI        |           1 |
| PFEIFFER     |           1 |
| PHOENIX      |           1 |
| PINKETT      |           1 |
| PITT         |           1 |
| POSEY        |           1 |
| PRESLEY      |           1 |
| REYNOLDS     |           1 |
| RYDER        |           1 |
| SINATRA      |           1 |
| SOBIESKI     |           1 |
| STALLONE     |           1 |
| SUVARI       |           1 |
| SWANK        |           1 |
| TAUTOU       |           1 |
| TOMEI        |           1 |
| VOIGHT       |           1 |
| WALKEN       |           1 |
| WAYNE        |           1 |
| WILSON       |           1 |
| WITHERSPOON  |           1 |
| WRAY         |           1 |
+--------------+-------------+
121 rows in set (0.23 sec)

mysql>
```

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**

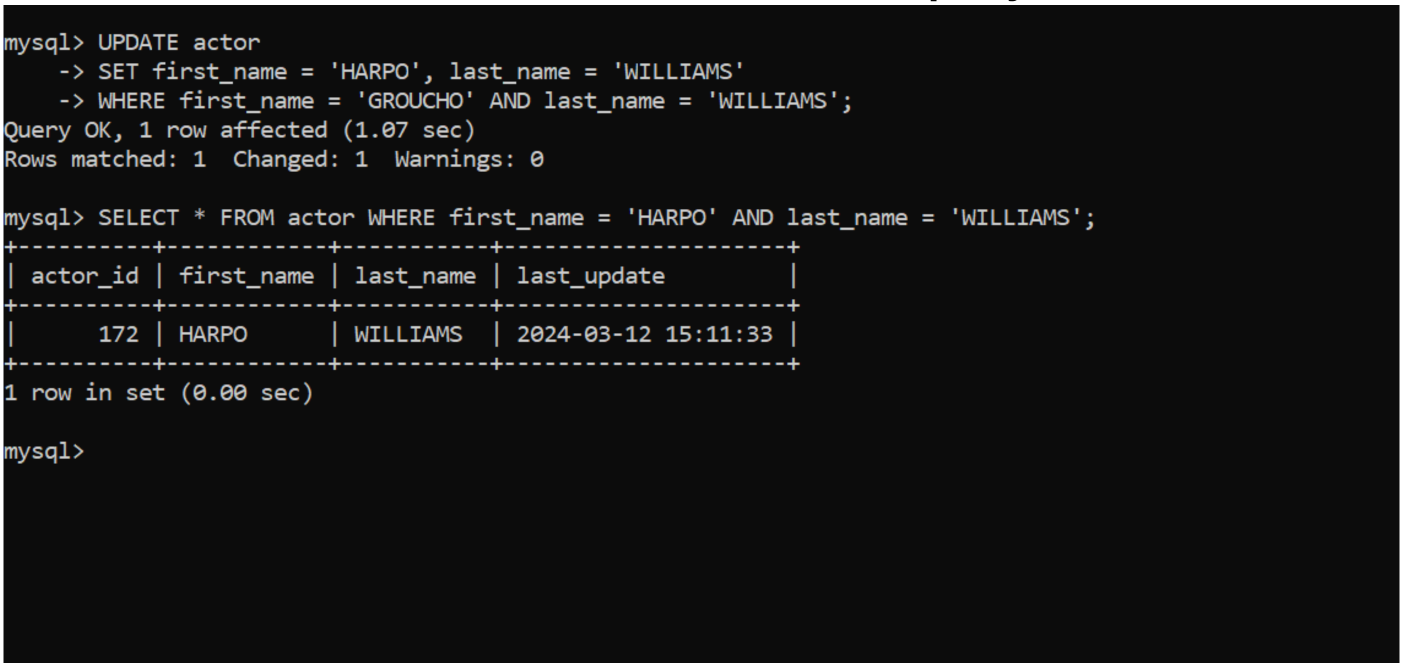


```
mysql> SELECT last_name, COUNT(*) AS actor_count
    -> from actor
    -> GROUP BY last_name
    -> HAVING actor_count >= 2
    -> ORDER BY actor_count DESC, last_name;
+-------------+-------------+
| last_name   | actor_count |
+-------------+-------------+
| KILMER      |           5 |
| NOLTE       |           4 |
| TEMPLE      |           4 |
| AKROYD      |           3 |
| ALLEN       |           3 |
| BERRY       |           3 |
| DAVIS       |           3 |
| DEGENERES   |           3 |
| GARLAND     |           3 |
| GUINESS     |           3 |
| HARRIS      |           3 |
| HOFFMAN     |           3 |
| HOPKINS     |           3 |
| JOHANSSON   |           3 |
| KEITEL      |           3 |
| PECK        |           3 |
| TORN        |           3 |
| WILLIAMS    |           3 |
| WILLIS      |           3 |
| ZELLWEGER   |           3 |
| BAILEY      |           2 |
| BENING      |           2 |
| BOLGER      |           2 |
| BRODY       |           2 |
| CAGE        |           2 |
| CHASE       |           2 |
| CRAWFORD    |           2 |
| CRONYN      |           2 |
| DEAN        |           2 |
| DEE         |           2 |
| DENCH       |           2 |
| DEPP        |           2 |
```



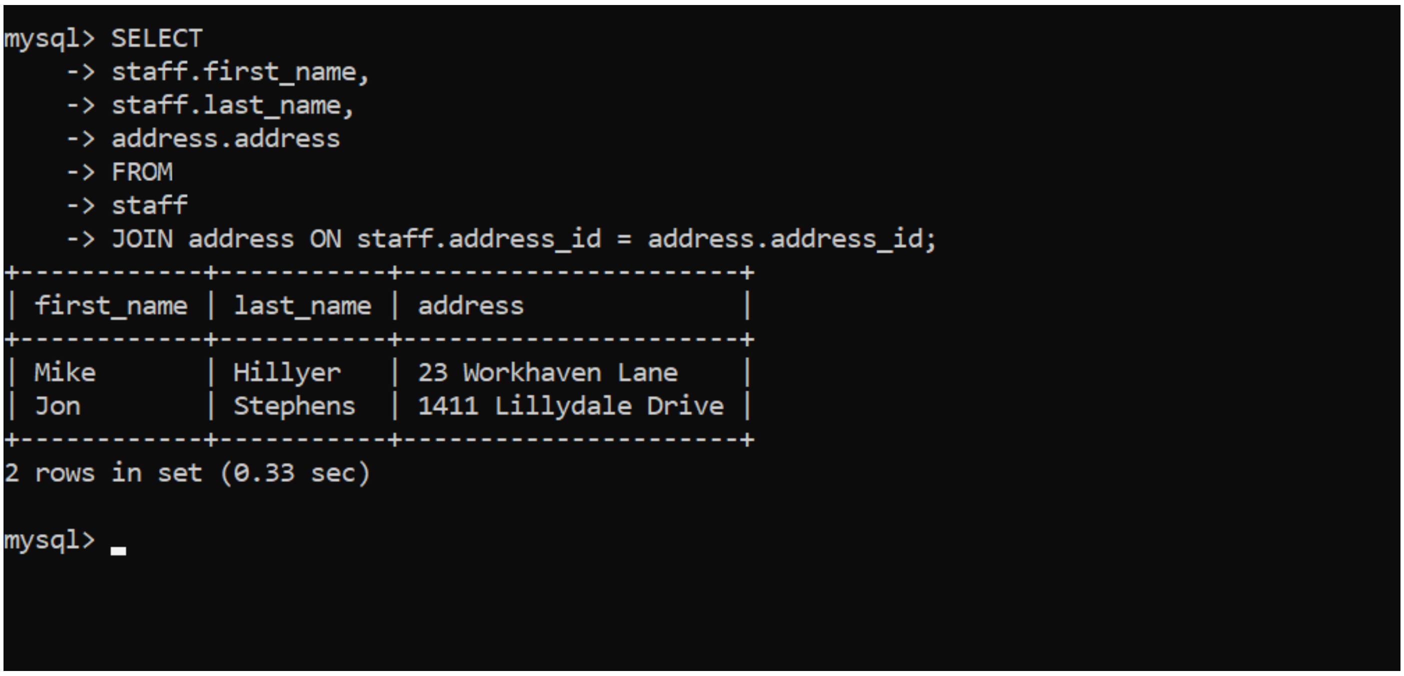
```
| DEPP        |           2 |
| DUKAKIS     |           2 |
| FAWCETT     |           2 |
| GOODING     |           2 |
| HACKMAN     |           2 |
| HOPPER      |           2 |
| JACKMAN     |           2 |
| MCCONAUGHEY |           2 |
| MCKELLEN    |           2 |
| MCQUEEN     |           2 |
| MONROE      |           2 |
| MOSTEL      |           2 |
| NEESON      |           2 |
| OLIVIER     |           2 |
| PALTROW     |           2 |
| PENN        |           2 |
| SILVERSTONE |           2 |
| STREEP      |           2 |
| TANDY       |           2 |
| TRACY       |           2 |
| WAHLBERG    |           2 |
| WEST        |           2 |
| WINSLET     |           2 |
| WOOD        |           2 |
+-------------+-------------+
55 rows in set (0.06 sec)

mysql> _
```

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



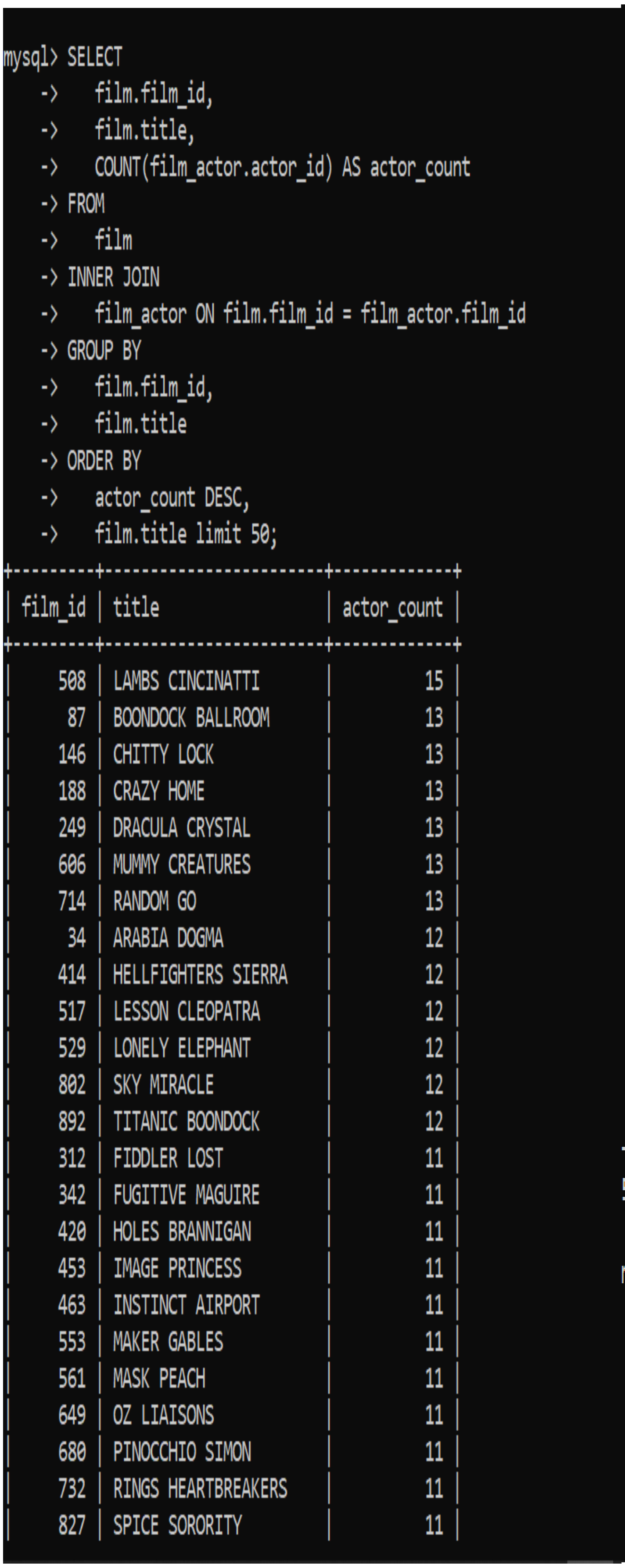
```
mysql> UPDATE actor
    -> SET first_name = 'HARPO', last_name = 'WILLIAMS'
    -> WHERE first_name = 'GROUCHO' AND last_name = 'WILLIAMS';
Query OK, 1 row affected (1.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM actor WHERE first_name = 'HARPO' AND last_name = 'WILLIAMS';
+----------+------------+-----------+---------------------+
| actor_id | first_name | last_name | last_update         |
+----------+------------+-----------+---------------------+
|      172 | HARPO      | WILLIAMS  | 2024-03-12 15:11:33 |
+----------+------------+-----------+---------------------+
1 row in set (0.00 sec)

mysql>
```

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:**

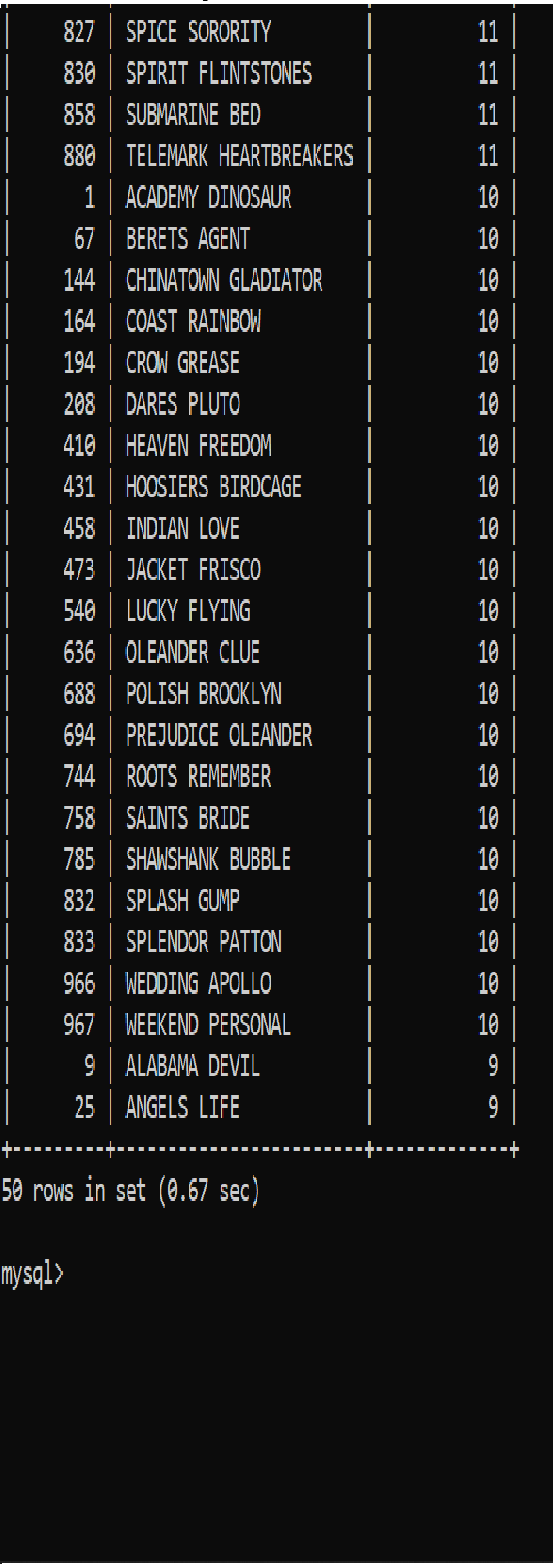


```
mysql> SELECT
    -> staff.first_name,
    -> staff.last_name,
    -> address.address
    -> FROM
    -> staff
    -> JOIN address ON staff.address_id = address.address_id;
+------------+-----------+----------------------+
| first_name | last_name | address              |
+------------+-----------+----------------------+
| Mike       | Hillyer   | 23 Workhaven Lane    |
| Jon        | Stephens  | 1411 Lillydale Drive |
+------------+-----------+----------------------+
2 rows in set (0.33 sec)

mysql>
```

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

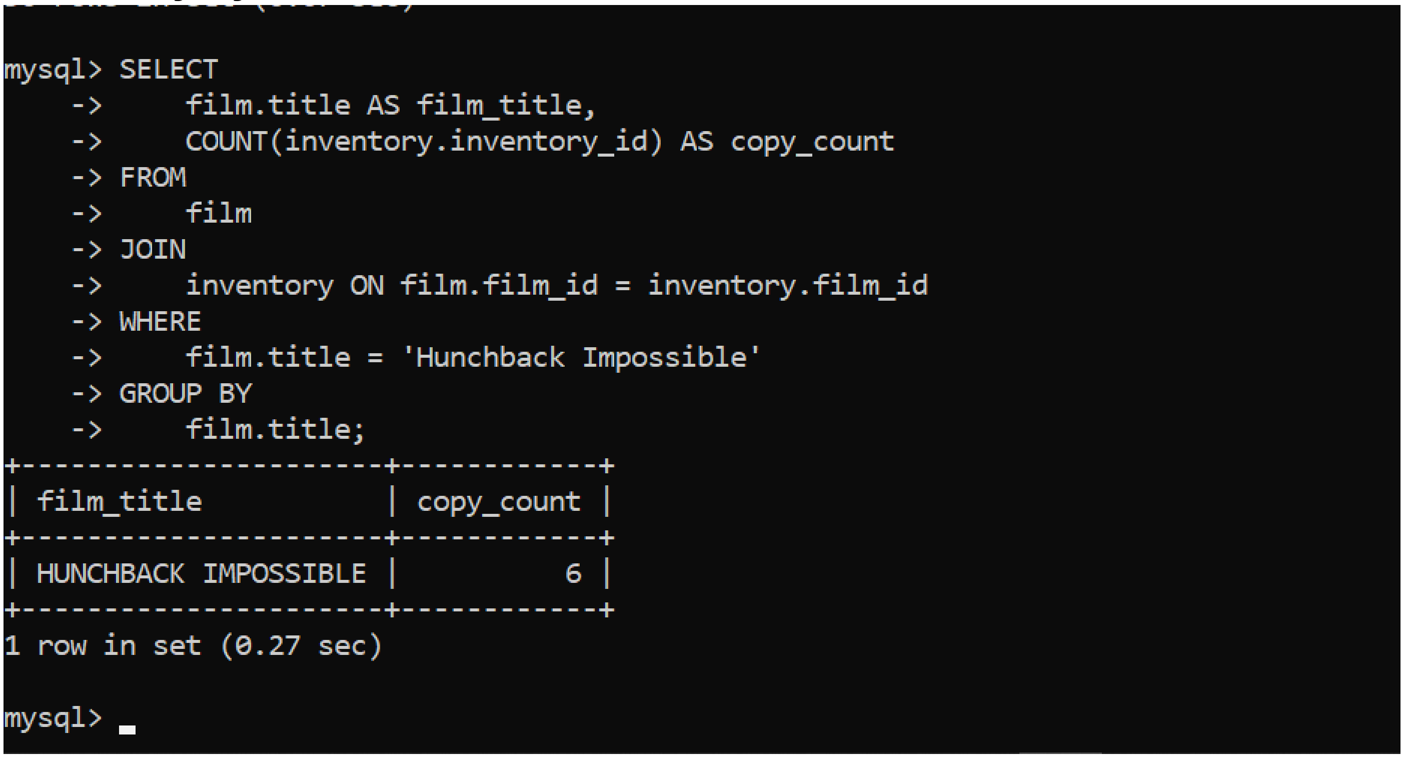


```
mysql> SELECT
    ->    film.film_id,
    ->    film.title,
    ->    COUNT(film_actor.actor_id) AS actor_count
    -> FROM
    ->    film
    -> INNER JOIN
    ->    film_actor ON film.film_id = film_actor.film_id
    -> GROUP BY
    ->    film.film_id,
    ->    film.title
    -> ORDER BY
    ->    actor_count DESC,
    ->    film.title limit 50;
+---------+----------------------+-------------+
| film_id | title                | actor_count |
+---------+----------------------+-------------+
|     508 | LAMBS CINCINATTI     |          15 |
|      87 | BOONDOCK BALLROOM    |          13 |
|     146 | CHITTY LOCK          |          13 |
|     188 | CRAZY HOME           |          13 |
|     249 | DRACULA CRYSTAL      |          13 |
|     606 | MUMMY CREATURES      |          13 |
|     714 | RANDOM GO            |          13 |
|      34 | ARABIA DOGMA         |          12 |
|     414 | HELLFIGHTERS SIERRA  |          12 |
|     517 | LESSON CLEOPATRA     |          12 |
|     529 | LONELY ELEPHANT      |          12 |
|     802 | SKY MIRACLE          |          12 |
|     892 | TITANIC BOONDOCK     |          12 |
|     312 | FIDDLER LOST         |          11 |
|     342 | FUGITIVE MAGUIRE     |          11 |
|     420 | HOLES BRANNIGAN      |          11 |
|     453 | IMAGE PRINCESS       |          11 |
|     463 | INSTINCT AIRPORT     |          11 |
|     553 | MAKER GABLES         |          11 |
|     561 | MASK PEACH           |          11 |
|     649 | OZ LIAISONS          |          11 |
|     680 | PINOCCHIO SIMON      |          11 |
|     732 | RINGS HEARTBREAKERS  |          11 |
|     827 | SPICE SORORITY       |          11 |
```



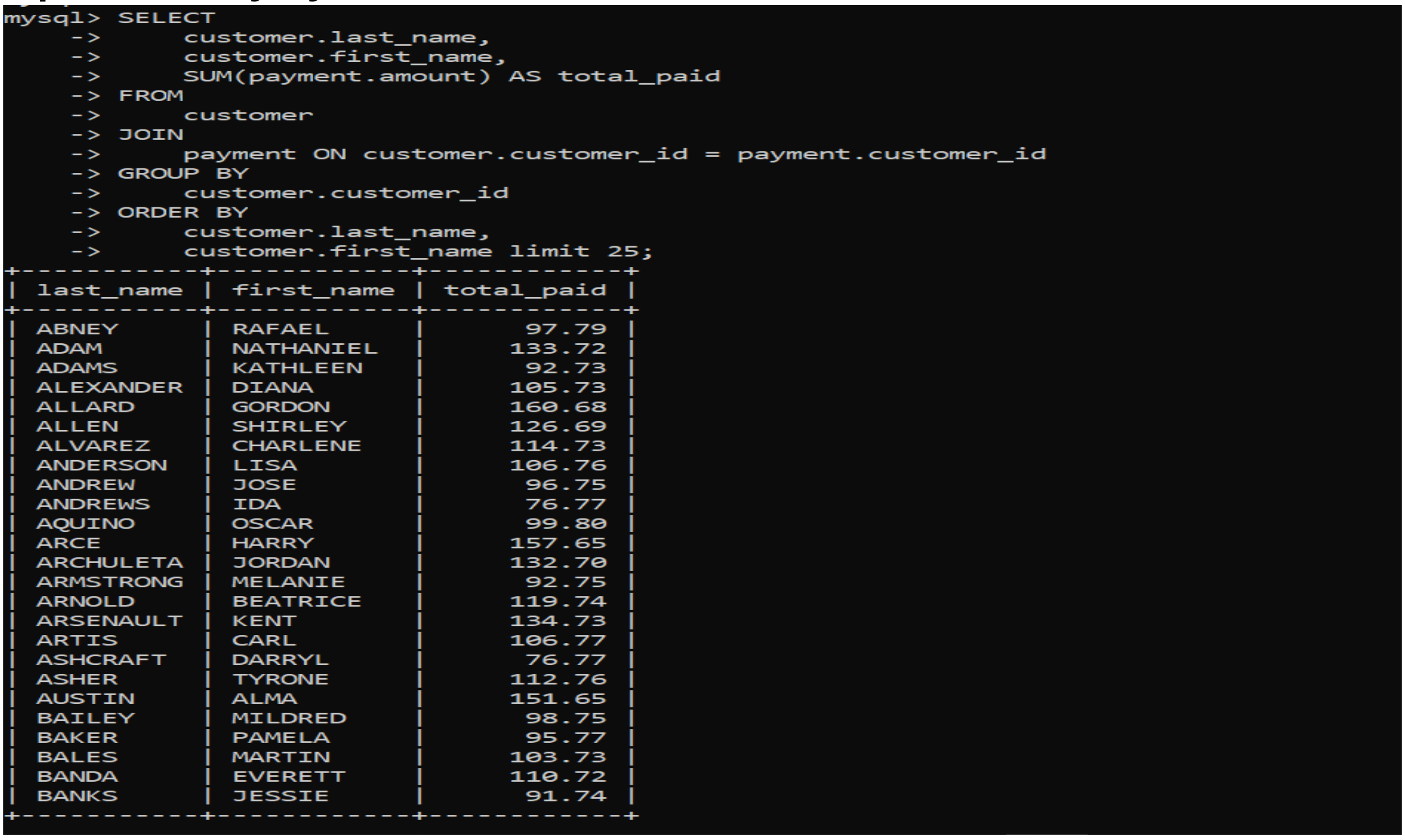
```
|     827 | SPICE SORORITY       |          11 |
|     830 | SPIRIT FLINTSTONES   |          11 |
|     858 | SUBMARINE BED        |          11 |
|     880 | TELEMARK HEARTBREAKERS |        11 |
|       1 | ACADEMY DINOSAUR     |          10 |
|      67 | BERETS AGENT         |          10 |
|     144 | CHINATOWN GLADIATOR  |          10 |
|     164 | COAST RAINBOW        |          10 |
|     194 | CROW GREASE          |          10 |
|     208 | DARES PLUTO          |          10 |
|     410 | HEAVEN FREEDOM       |          10 |
|     431 | HOOSIERS BIRDCAGE    |          10 |
|     458 | INDIAN LOVE          |          10 |
|     473 | JACKET FRISCO        |          10 |
|     540 | LUCKY FLYING         |          10 |
|     636 | OLEANDER CLUE        |          10 |
|     688 | POLISH BROOKLYN      |          10 |
|     694 | PREJUDICE OLEANDER   |          10 |
|     744 | ROOTS REMEMBER       |          10 |
|     758 | SAINTS BRIDE         |          10 |
|     785 | SHAWSHANK BUBBLE     |          10 |
|     832 | SPLASH GUMP          |          10 |
|     833 | SPLENDOR PATTON      |          10 |
|     966 | WEDDING APOLLO       |          10 |
|     967 | WEEKEND PERSONAL     |          10 |
|       9 | ALABAMA DEVIL        |           9 |
|      25 | ANGELS LIFE          |           9 |
+---------+----------------------+-------------+
50 rows in set (0.67 sec)

mysql>
```

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



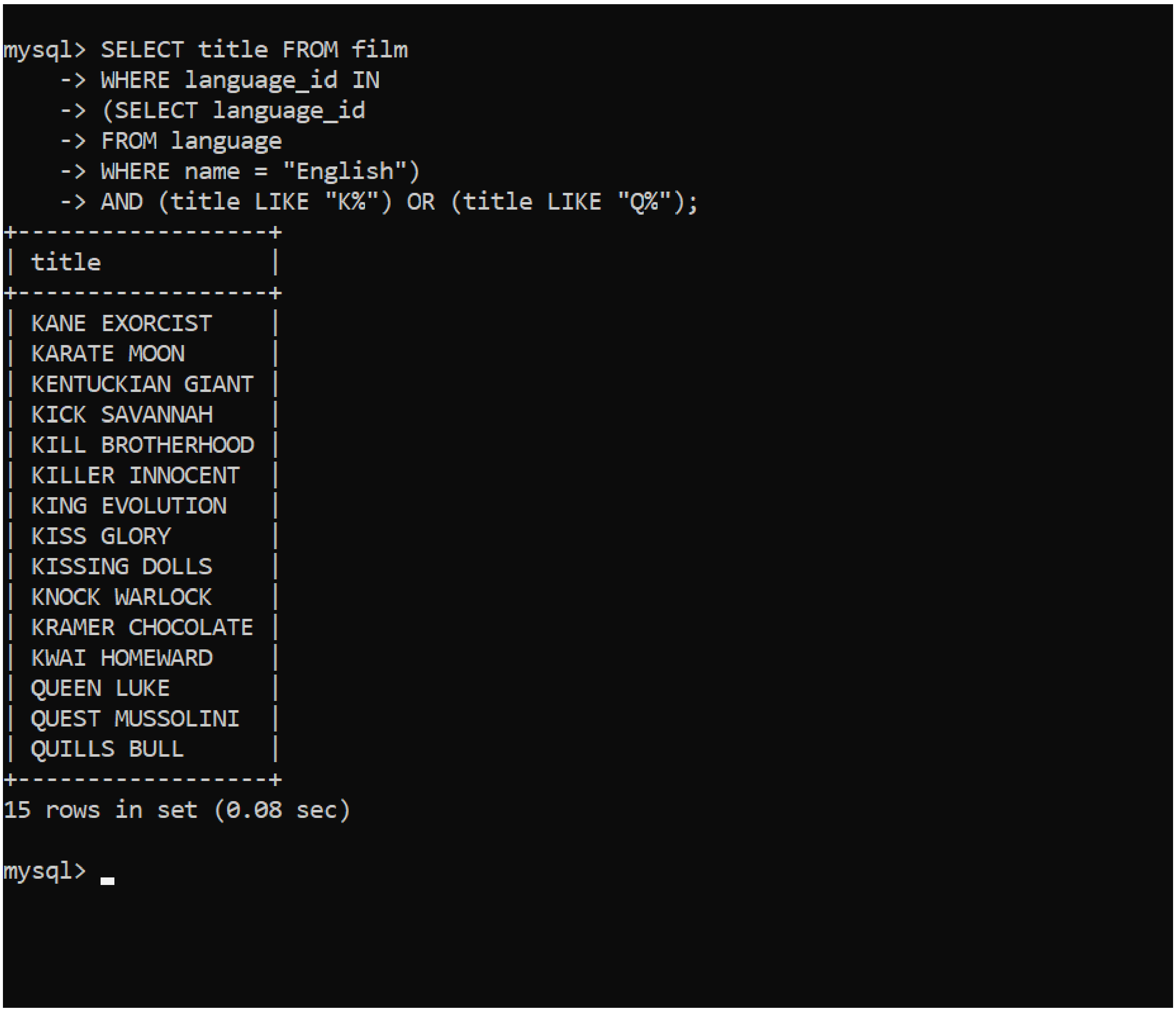
```
mysql> SELECT
    ->     film.title AS film_title,
    ->     COUNT(inventory.inventory_id) AS copy_count
    -> FROM
    ->     film
    -> JOIN
    ->     inventory ON film.film_id = inventory.film_id
    -> WHERE
    ->     film.title = 'Hunchback Impossible'
    -> GROUP BY
    ->     film.title;
+----------------------+------------+
| film_title           | copy_count |
+----------------------+------------+
| HUNCHBACK IMPOSSIBLE |          6 |
+----------------------+------------+
1 row in set (0.27 sec)

mysql> _
```

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



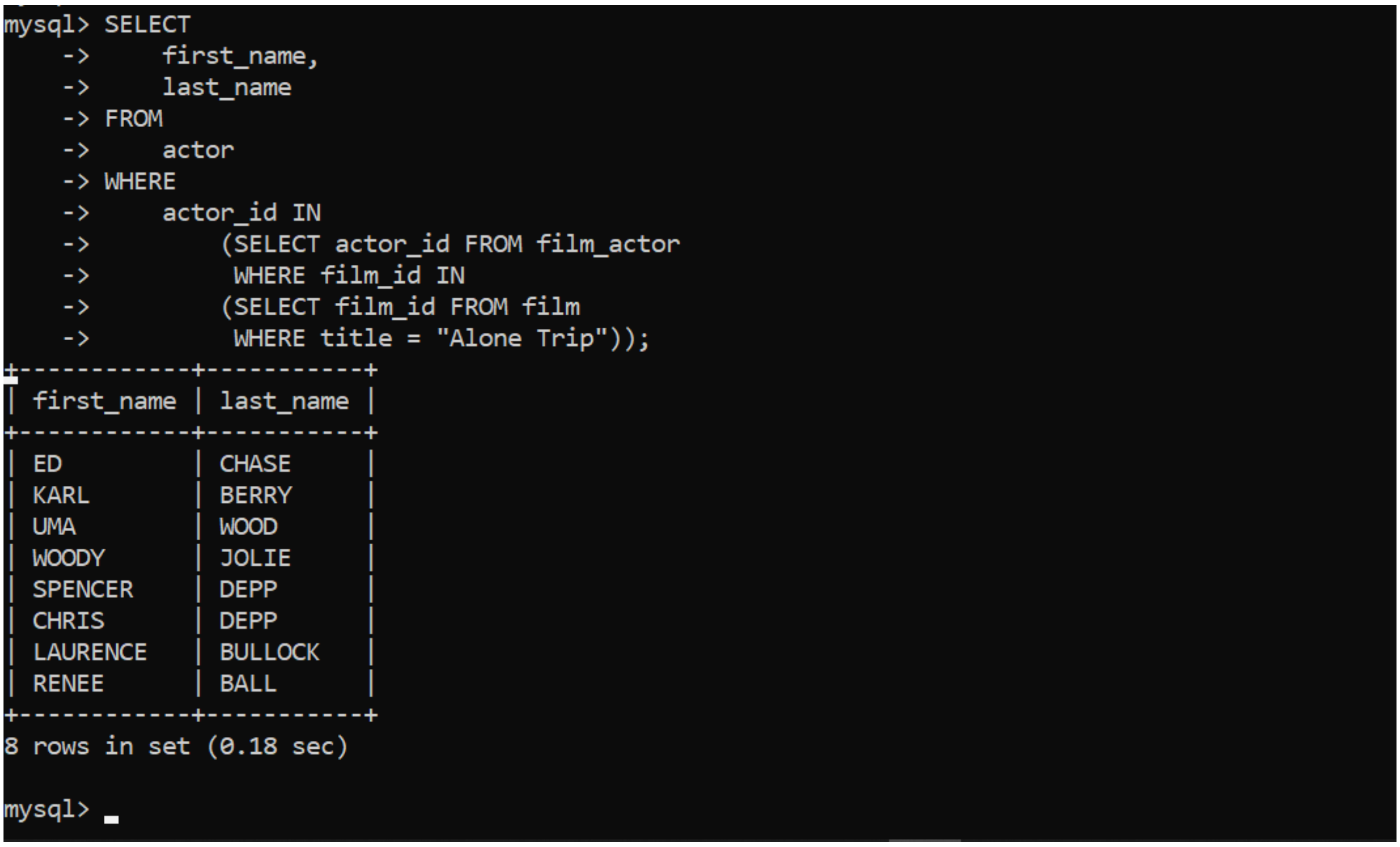
```
mysql> SELECT
    ->     customer.last_name,
    ->     customer.first_name,
    ->     SUM(payment.amount) AS total_paid
    -> FROM
    ->     customer
    -> JOIN
    ->     payment ON customer.customer_id = payment.customer_id
    -> GROUP BY
    ->     customer.customer_id
    -> ORDER BY
    ->     customer.last_name,
    ->     customer.first_name limit 25;
+-----------+------------+------------+
| last_name | first_name | total_paid |
+-----------+------------+------------+
| ABNEY     | RAFAEL     |      97.79 |
| ADAM      | NATHANIEL  |     133.72 |
| ADAMS     | KATHLEEN   |      92.73 |
| ALEXANDER | DIANA      |     105.73 |
| ALLARD    | GORDON     |     160.68 |
| ALLEN     | SHIRLEY    |     126.69 |
| ALVAREZ   | CHARLENE   |     114.73 |
| ANDERSON  | LISA       |     106.76 |
| ANDREW    | JOSE       |      96.75 |
| ANDREWS   | IDA        |      76.77 |
| AQUINO    | OSCAR      |      99.80 |
| ARCE      | HARRY      |     157.65 |
| ARCHULETA | JORDAN     |     132.70 |
| ARMSTRONG | MELANIE    |      92.75 |
| ARNOLD    | BEATRICE   |     119.74 |
| ARSENAULT | KENT       |     134.73 |
| ARTIS     | CARL       |     106.77 |
| ASHCRAFT  | DARRYL     |      76.77 |
| ASHER     | TYRONE     |     112.76 |
| AUSTIN    | ALMA       |     151.65 |
| BAILEY    | MILDRED    |      98.75 |
| BAKER     | PAMELA     |      95.77 |
| BALES     | MARTIN     |     103.73 |
| BANDA     | EVERETT    |     110.72 |
| BANKS     | JESSIE     |      91.74 |
+-----------+------------+------------+
```

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.**



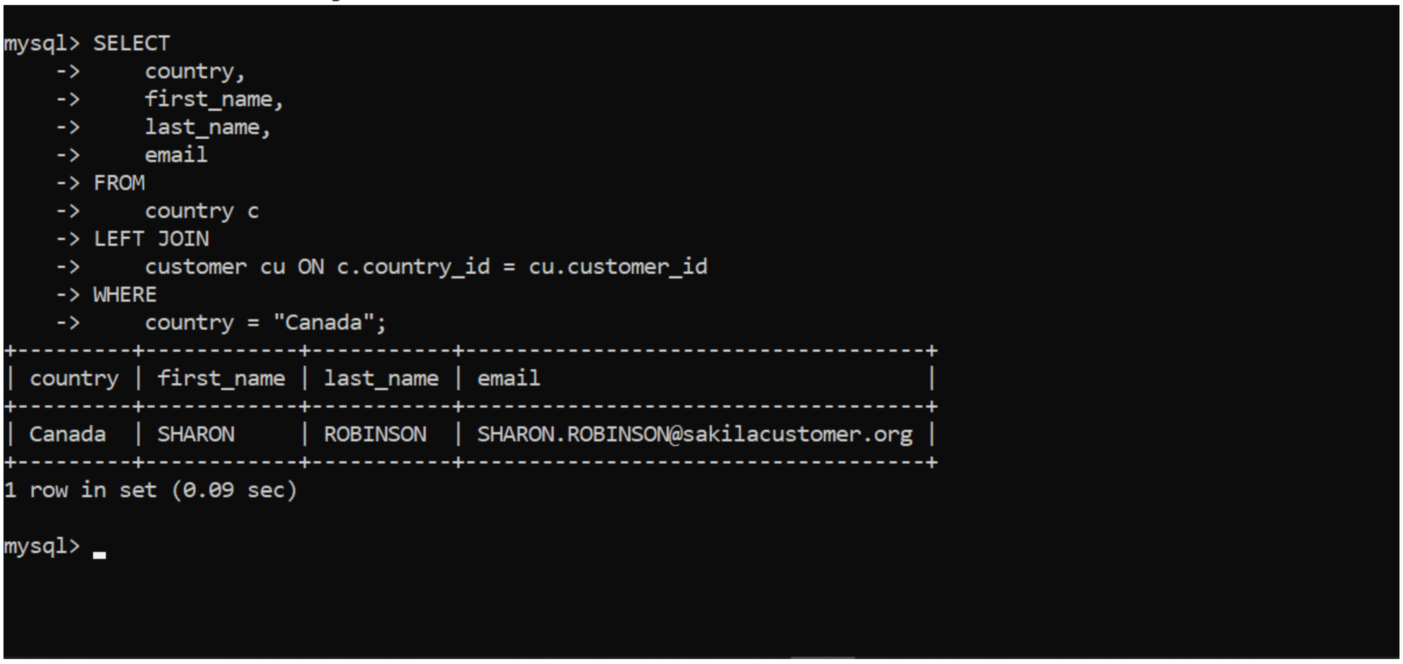
```
mysql> SELECT title FROM film
    -> WHERE language_id IN
    -> (SELECT language_id
    -> FROM language
    -> WHERE name = "English")
    -> AND (title LIKE "K%") OR (title LIKE "Q%");
+-------------------+
| title             |
+-------------------+
| KANE EXORCIST     |
| KARATE MOON       |
| 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 (0.08 sec)

mysql> _
```

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



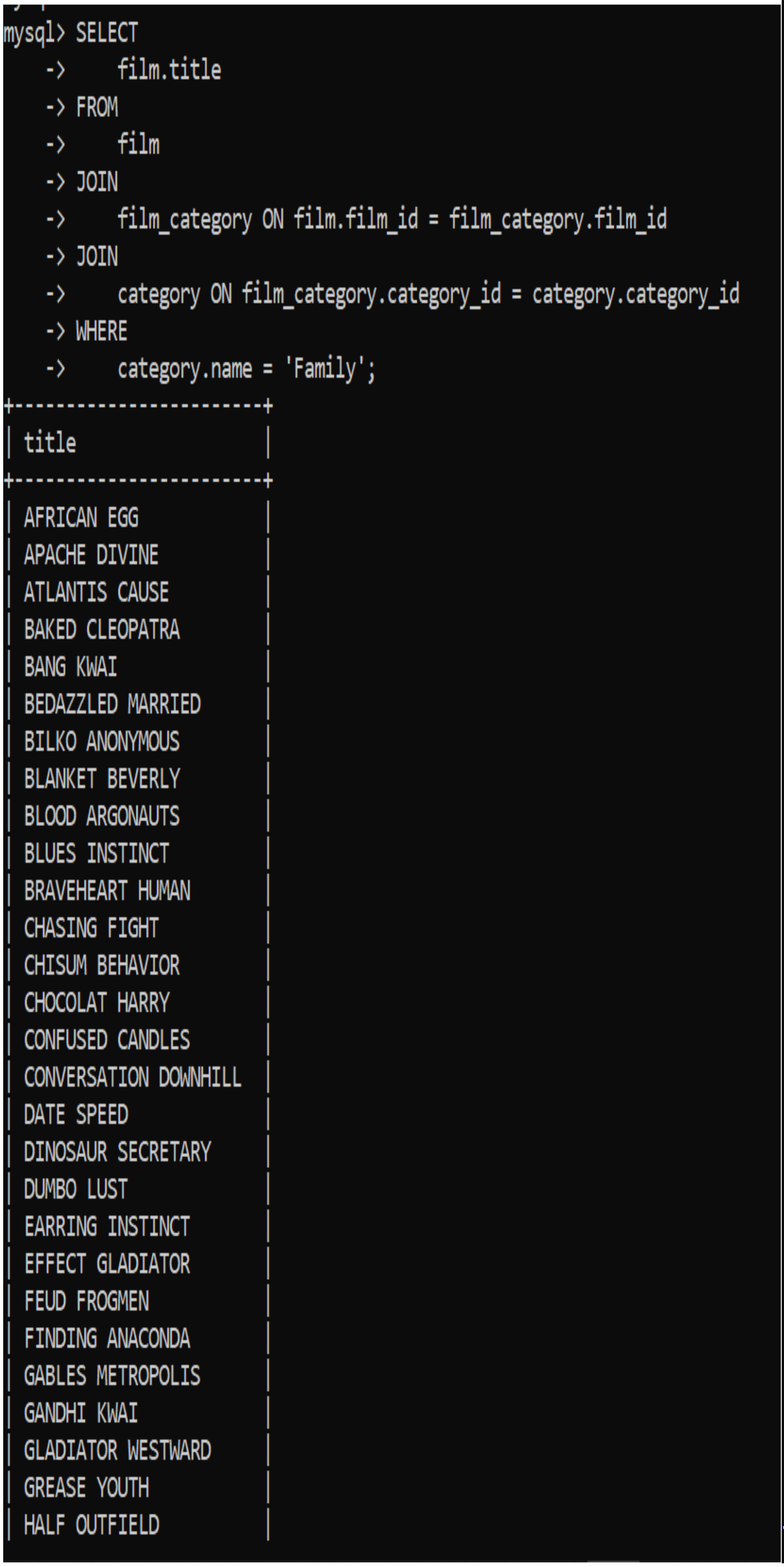
```
mysql> SELECT
    ->     first_name,
    ->     last_name
    -> FROM
    ->     actor
    -> WHERE
    ->     actor_id IN
    ->         (SELECT actor_id FROM film_actor
    ->          WHERE film_id IN
    ->         (SELECT film_id FROM film
    ->          WHERE title = "Alone Trip"));
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| ED         | CHASE     |
| KARL       | BERRY     |
| UMA        | WOOD      |
| WOODY      | JOLIE     |
| SPENCER    | DEPP      |
| CHRIS      | DEPP      |
| LAURENCE   | BULLOCK   |
| RENEE      | BALL      |
+------------+-----------+
8 rows in set (0.18 sec)

mysql> _
```

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.

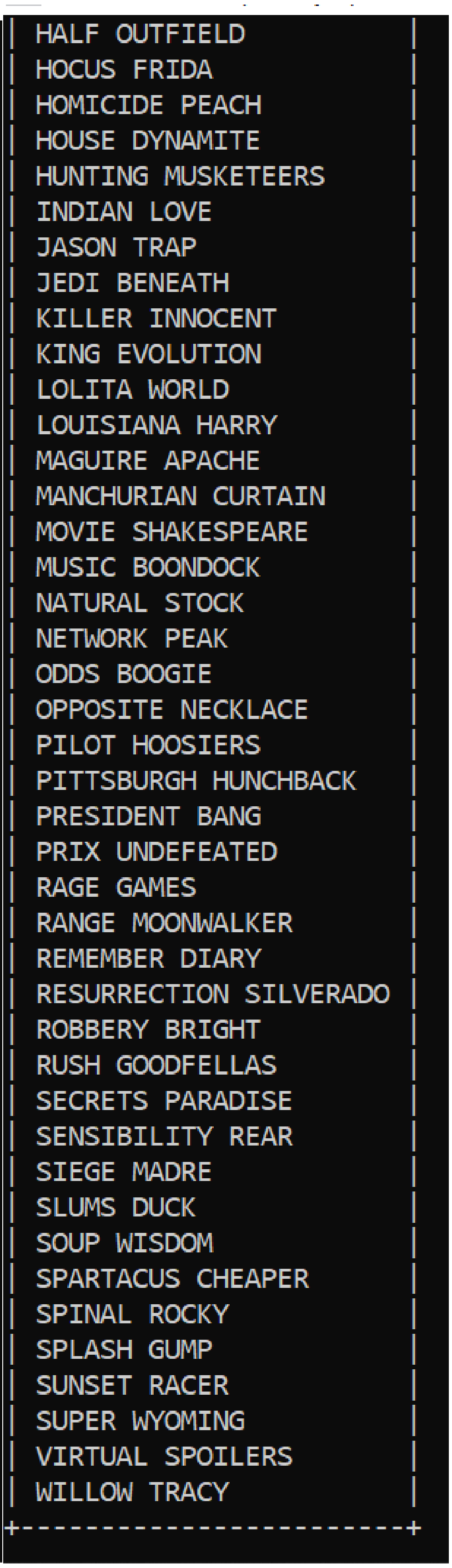


```
mysql> SELECT
    ->     country,
    ->     first_name,
    ->     last_name,
    ->     email
    -> FROM
    ->     country c
    -> LEFT JOIN
    ->     customer cu ON c.country_id = cu.customer_id
    -> WHERE
    ->     country = "Canada";
+---------+------------+-----------+-----------------------------------+
| country | first_name | last_name | email                             |
+---------+------------+-----------+-----------------------------------+
| Canada  | SHARON     | ROBINSON  | SHARON.ROBINSON@sakilacustomer.org |
+---------+------------+-----------+-----------------------------------+
1 row in set (0.09 sec)

mysql> _
```

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

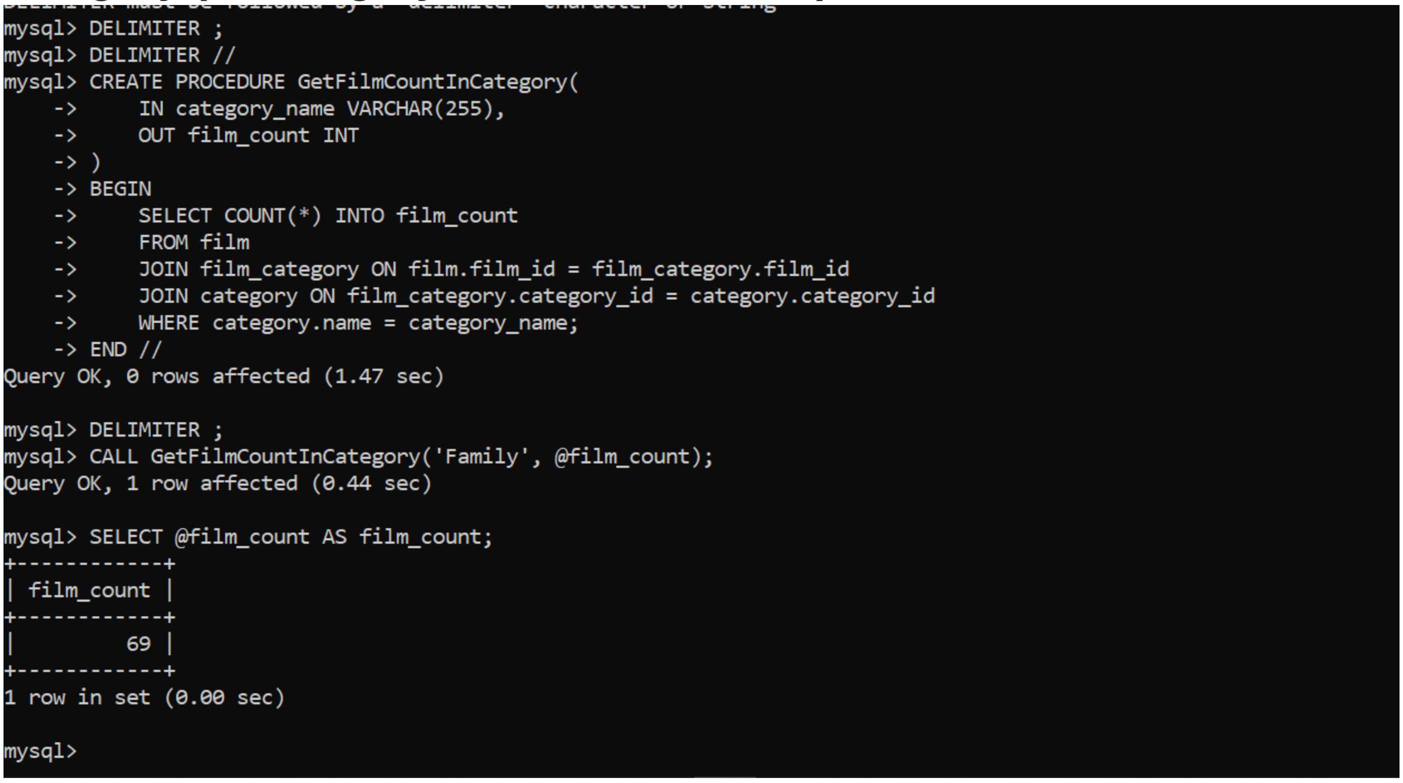


```
mysql> SELECT
    ->     film.title
    -> FROM
    ->     film
    -> JOIN
    ->     film_category ON film.film_id = film_category.film_id
    -> JOIN
    ->     category ON film_category.category_id = category.category_id
    -> WHERE
    ->     category.name = 'Family';
+-----------------------+
| 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      |
| CONVERSATION DOWNHILL |
| DATE SPEED            |
| DINOSAUR SECRETARY    |
| DUMBO LUST            |
| EARRING INSTINCT      |
| EFFECT GLADIATOR      |
| FEUD FROGMEN          |
| FINDING ANACONDA      |
| GABLES METROPOLIS     |
| GANDHI KWAI           |
| GLADIATOR WESTWARD    |
| GREASE YOUTH          |
| HALF OUTFIELD         |
```



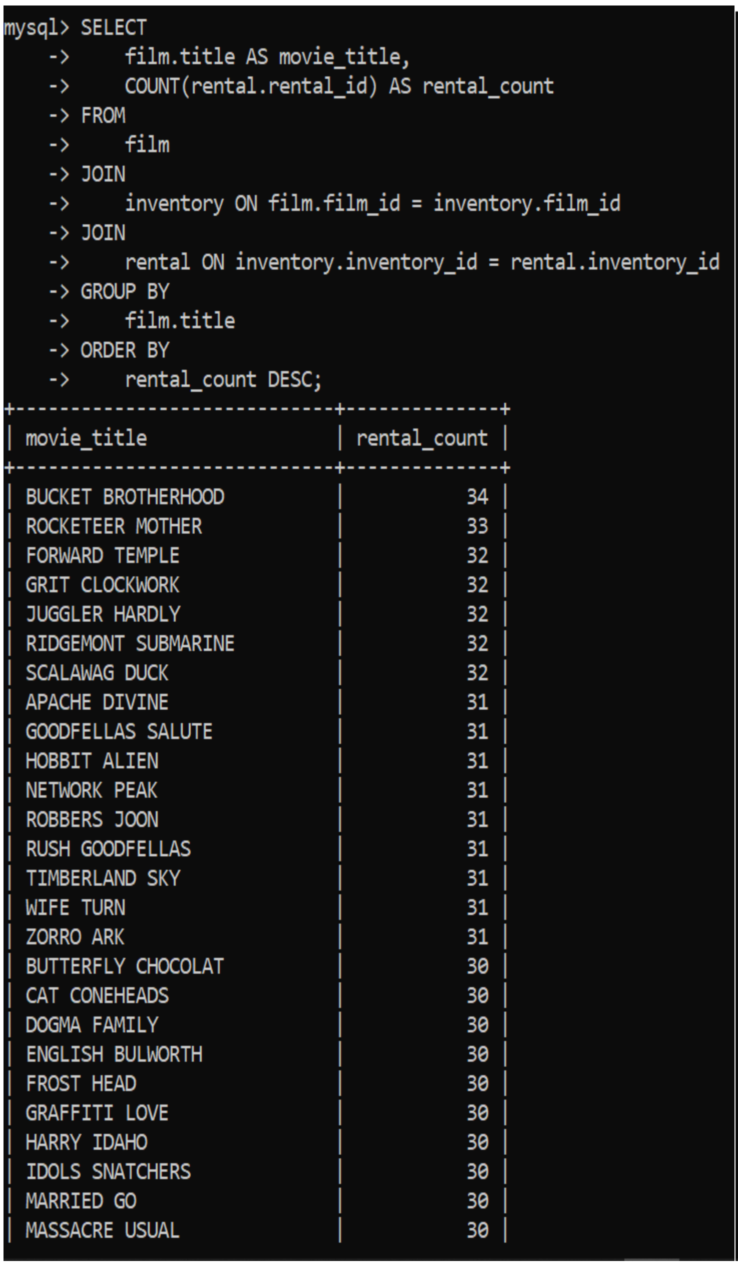
```
| HALF OUTFIELD          |
| HOCUS FRIDA            |
| HOMICIDE PEACH         |
| HOUSE DYNAMITE         |
| HUNTING MUSKETEERS     |
| INDIAN LOVE            |
| JASON TRAP             |
| JEDI BENEATH           |
| KILLER INNOCENT        |
| KING EVOLUTION         |
| LOLITA WORLD           |
| LOUISIANA HARRY        |
| MAGUIRE APACHE         |
| MANCHURIAN CURTAIN     |
| MOVIE SHAKESPEARE      |
| MUSIC BOONDOCK         |
| NATURAL STOCK          |
| NETWORK PEAK           |
| ODDS BOOGIE            |
| OPPOSITE NECKLACE      |
| PILOT HOOSIERS         |
| PITTSBURGH HUNCHBACK   |
| PRESIDENT BANG         |
| PRIX UNDEFEATED        |
| RAGE GAMES             |
| RANGE MOONWALKER       |
| REMEMBER DIARY         |
| RESURRECTION SILVERADO |
| ROBBERY BRIGHT         |
| RUSH GOODFELLAS        |
| SECRETS PARADISE       |
| SENSIBILITY REAR       |
| SIEGE MADRE            |
| SLUMS DUCK             |
| SOUP WISDOM            |
| SPARTACUS CHEAPER      |
| SPINAL ROCKY           |
| SPLASH GUMP            |
| SUNSET RACER           |
| SUPER WYOMING          |
| VIRTUAL SPOILERS       |
| WILLOW TRACY           |
+------------------------+
```

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

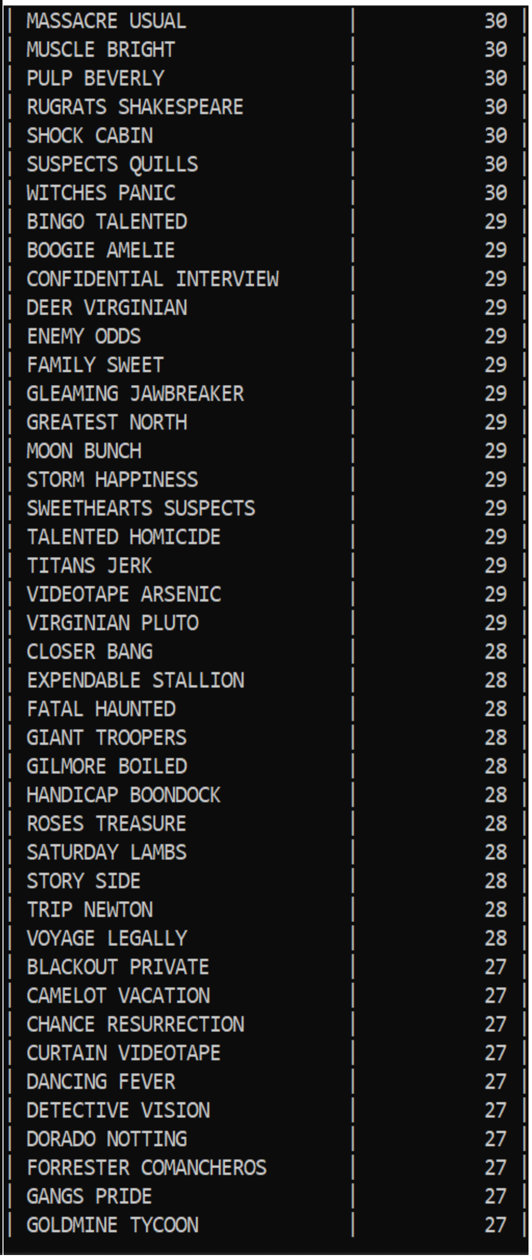


```
mysql> DELIMITER ;
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetFilmCountInCategory(
    ->     IN category_name VARCHAR(255),
    ->     OUT film_count INT
    -> )
    -> BEGIN
    ->     SELECT COUNT(*) INTO film_count
    ->     FROM film
    ->     JOIN film_category ON film.film_id = film_category.film_id
    ->     JOIN category ON film_category.category_id = category.category_id
    ->     WHERE category.name = category_name;
    -> END //
Query OK, 0 rows affected (1.47 sec)

mysql> DELIMITER ;
mysql> CALL GetFilmCountInCategory('Family', @film_count);
Query OK, 1 row affected (0.44 sec)

mysql> SELECT @film_count AS film_count;
+------------+
| film_count |
+------------+
|         69 |
+------------+
1 row in set (0.00 sec)

mysql>
```

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

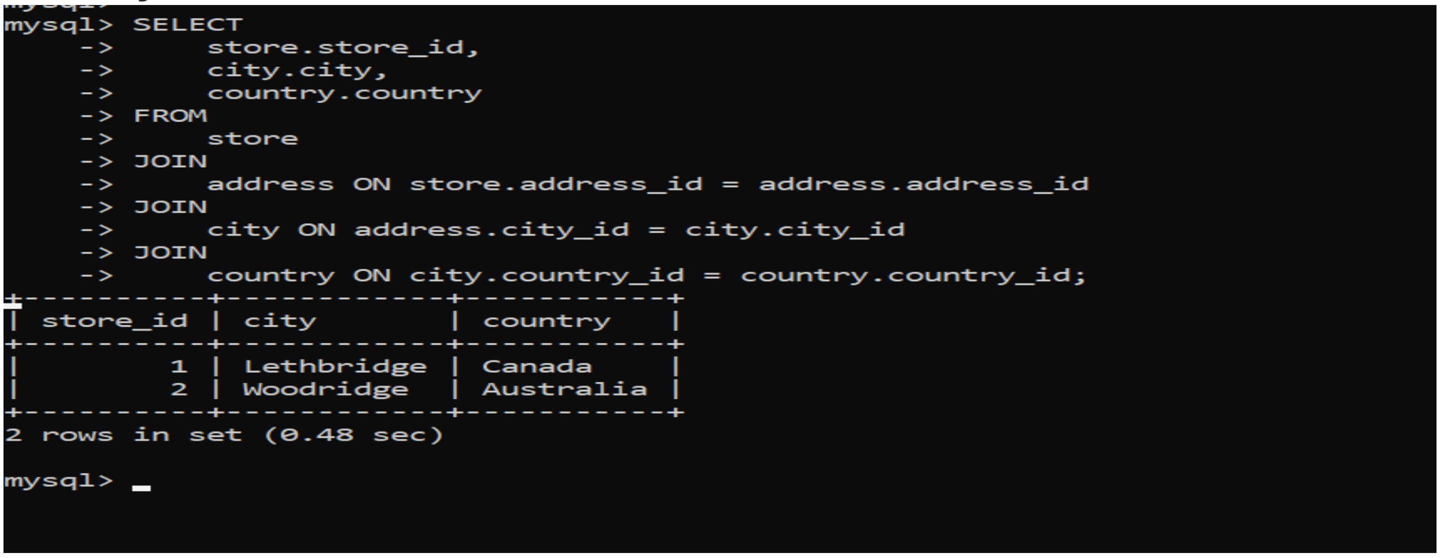


```
mysql> SELECT
    ->     film.title AS movie_title,
    ->     COUNT(rental.rental_id) AS rental_count
    -> FROM
    ->     film
    -> JOIN
    ->     inventory ON film.film_id = inventory.film_id
    -> JOIN
    ->     rental ON inventory.inventory_id = rental.inventory_id
    -> GROUP BY
    ->     film.title
    -> ORDER BY
    ->     rental_count DESC;
+---------------------------+--------------+
| movie_title               | rental_count |
+---------------------------+--------------+
| BUCKET BROTHERHOOD        |           34 |
| ROCKETEER MOTHER          |           33 |
| FORWARD TEMPLE            |           32 |
| GRIT CLOCKWORK            |           32 |
| JUGGLER HARDLY            |           32 |
| RIDGEMONT SUBMARINE       |           32 |
| SCALAWAG DUCK             |           32 |
| APACHE DIVINE             |           31 |
| GOODFELLAS SALUTE         |           31 |
| HOBBIT ALIEN              |           31 |
| NETWORK PEAK              |           31 |
| ROBBERS JOON              |           31 |
| RUSH GOODFELLAS           |           31 |
| TIMBERLAND SKY            |           31 |
| WIFE TURN                 |           31 |
| ZORRO ARK                 |           31 |
| BUTTERFLY CHOCOLAT        |           30 |
| CAT CONEHEADS             |           30 |
| DOGMA FAMILY              |           30 |
| ENGLISH BULWORTH          |           30 |
| FROST HEAD                |           30 |
| GRAFFITI LOVE             |           30 |
| HARRY IDAHO               |           30 |
| IDOLS SNATCHERS           |           30 |
| MARRIED GO                |           30 |
| MASSACRE USUAL            |           30 |
```



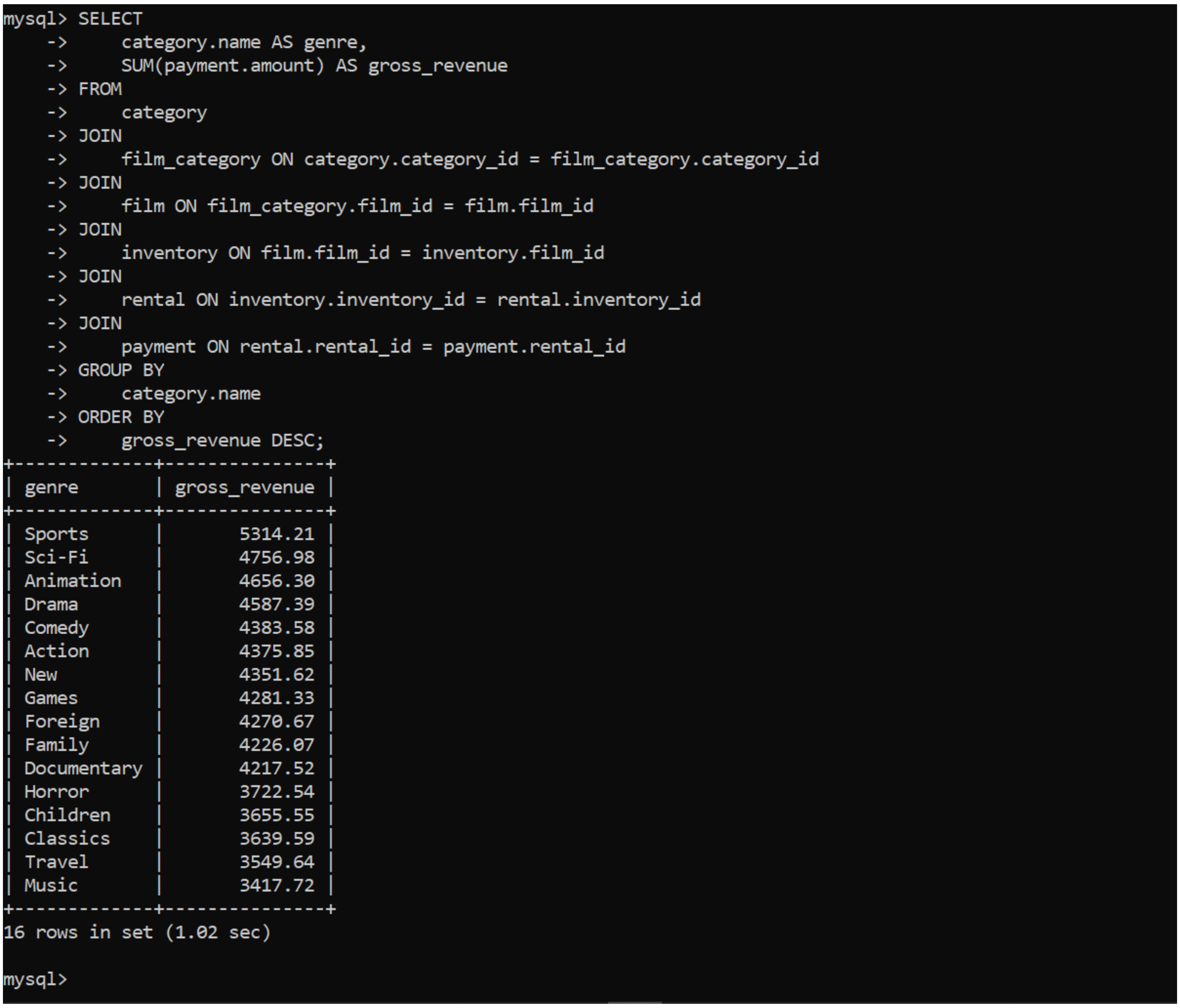
```
| MASSACRE USUAL               |           30 |
| MUSCLE BRIGHT                |           30 |
| PULP BEVERLY                 |           30 |
| RUGRATS SHAKESPEARE          |           30 |
| SHOCK CABIN                  |           30 |
| SUSPECTS QUILLS              |           30 |
| WITCHES PANIC                |           30 |
| BINGO TALENTED               |           29 |
| BOOGIE AMELIE                |           29 |
| CONFIDENTIAL INTERVIEW       |           29 |
| DEER VIRGINIAN               |           29 |
| ENEMY ODDS                   |           29 |
| FAMILY SWEET                 |           29 |
| GLEAMING JAWBREAKER          |           29 |
| GREATEST NORTH               |           29 |
| MOON BUNCH                   |           29 |
| STORM HAPPINESS              |           29 |
| SWEETHEARTS SUSPECTS         |           29 |
| TALENTED HOMICIDE            |           29 |
| TITANS JERK                  |           29 |
| VIDEOTAPE ARSENIC            |           29 |
| VIRGINIAN PLUTO              |           29 |
| CLOSER BANG                  |           28 |
| EXPENDABLE STALLION          |           28 |
| FATAL HAUNTED                |           28 |
| GIANT TROOPERS               |           28 |
| GILMORE BOILED               |           28 |
| HANDICAP BOONDOCK            |           28 |
| ROSES TREASURE               |           28 |
| SATURDAY LAMBS               |           28 |
| STORY SIDE                   |           28 |
| TRIP NEWTON                  |           28 |
| VOYAGE LEGALLY               |           28 |
| BLACKOUT PRIVATE             |           27 |
| CAMELOT VACATION             |           27 |
| CHANCE RESURRECTION          |           27 |
| CURTAIN VIDEOTAPE            |           27 |
| DANCING FEVER                |           27 |
| DETECTIVE VISION             |           27 |
| DORADO NOTTING               |           27 |
| FORRESTER COMANCHEROS        |           27 |
| GANGS PRIDE                  |           27 |
| GOLDMINE TYCOON              |           27 |
```

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



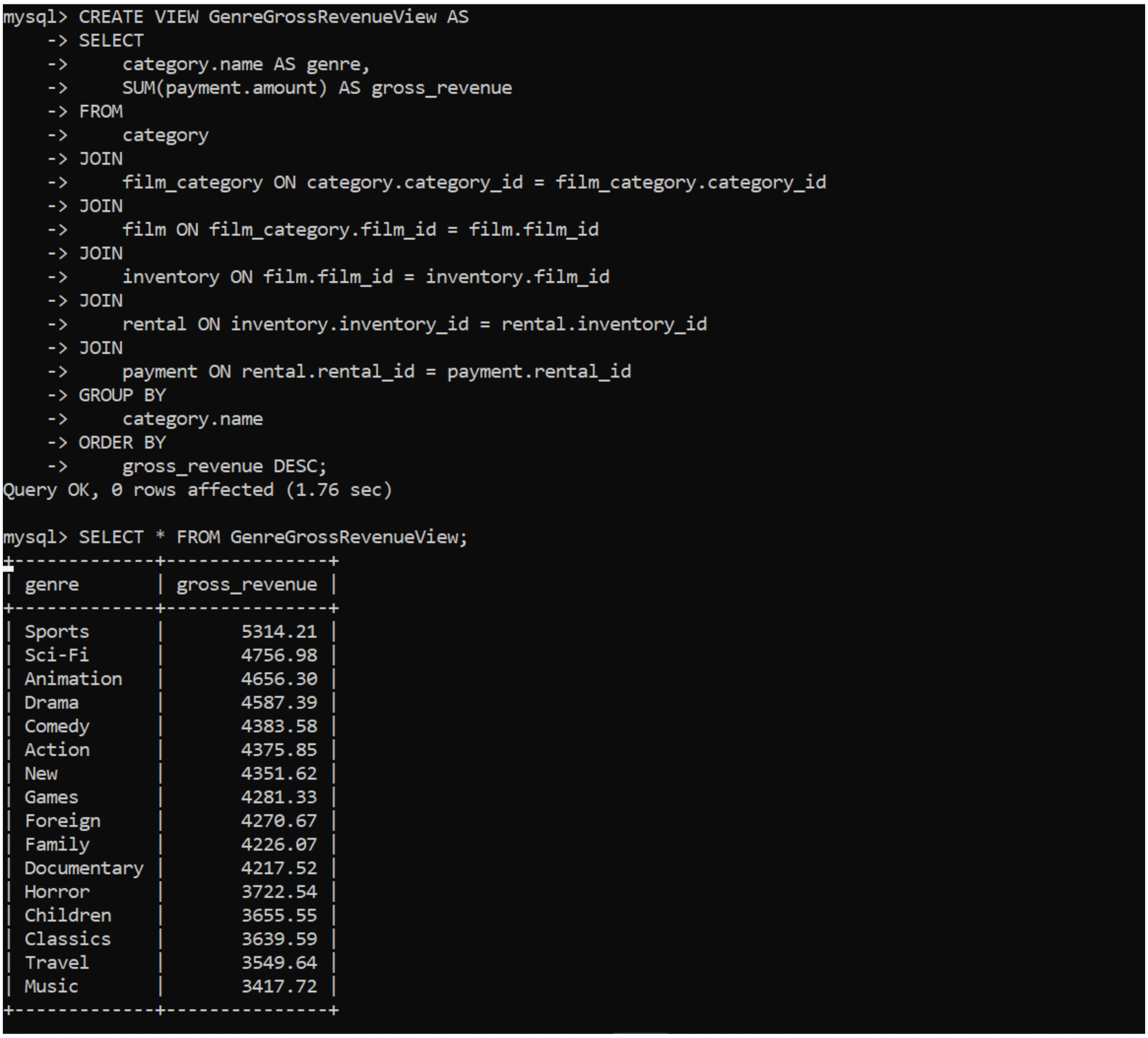
```
mysql> SELECT
    ->     store.store_id,
    ->     city.city,
    ->     country.country
    -> FROM
    ->     store
    -> JOIN
    ->     address ON store.address_id = address.address_id
    -> JOIN
    ->     city ON address.city_id = city.city_id
    -> JOIN
    ->     country ON city.country_id = country.country_id;
+----------+------------+-----------+
| store_id | city       | country   |
+----------+------------+-----------+
|        1 | Lethbridge | Canada    |
|        2 | Woodridge  | Australia |
+----------+------------+-----------+
2 rows in set (0.48 sec)

mysql> _
```

18. List the genres and its gross revenue.



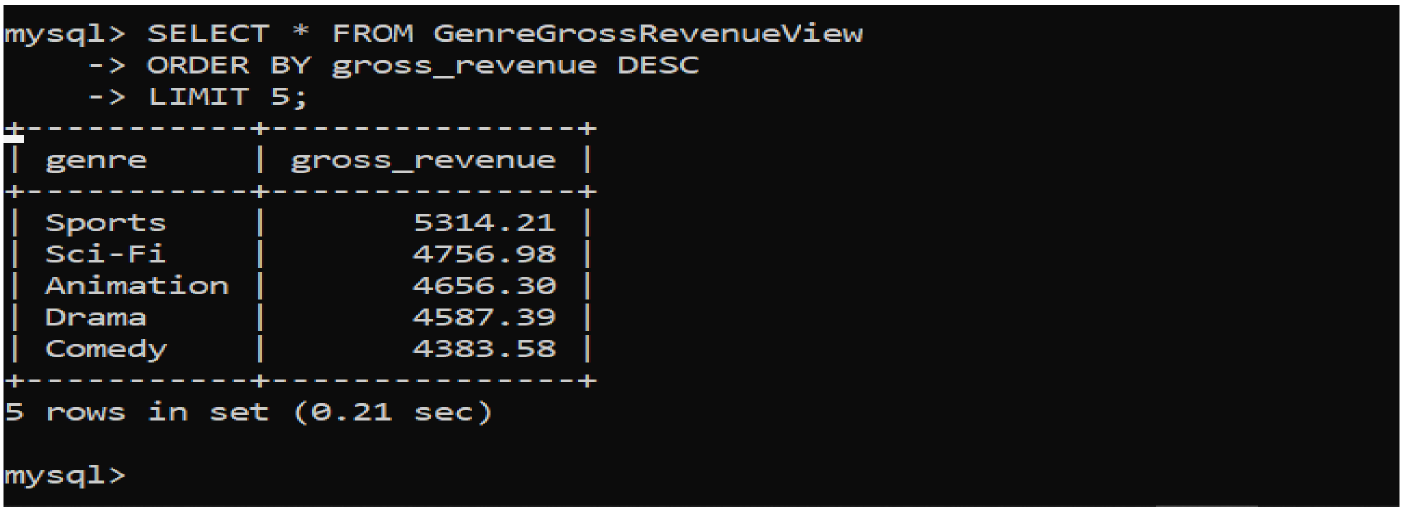
```
mysql> SELECT
    ->     category.name AS genre,
    ->     SUM(payment.amount) AS gross_revenue
    -> FROM
    ->     category
    -> JOIN
    ->     film_category ON category.category_id = film_category.category_id
    -> JOIN
    ->     film ON film_category.film_id = film.film_id
    -> JOIN
    ->     inventory ON film.film_id = inventory.film_id
    -> JOIN
    ->     rental ON inventory.inventory_id = rental.inventory_id
    -> JOIN
    ->     payment ON rental.rental_id = payment.rental_id
    -> GROUP BY
    ->     category.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 (1.02 sec)

mysql>
```

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



```
mysql> CREATE VIEW GenreGrossRevenueView AS
    -> SELECT
    ->     category.name AS genre,
    ->     SUM(payment.amount) AS gross_revenue
    -> FROM
    ->     category
    -> JOIN
    ->     film_category ON category.category_id = film_category.category_id
    -> JOIN
    ->     film ON film_category.film_id = film.film_id
    -> JOIN
    ->     inventory ON film.film_id = inventory.film_id
    -> JOIN
    ->     rental ON inventory.inventory_id = rental.inventory_id
    -> JOIN
    ->     payment ON rental.rental_id = payment.rental_id
    -> GROUP BY
    ->     category.name
    -> ORDER BY
    ->     gross_revenue DESC;
Query OK, 0 rows affected (1.76 sec)

mysql> SELECT * FROM GenreGrossRevenueView;
+-------------+---------------+
| 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 |
+-------------+---------------+
```

20. Select top 5 genres in gross revenue view.



```
mysql> SELECT * FROM GenreGrossRevenueView
    -> 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 (0.21 sec)

mysql>
```