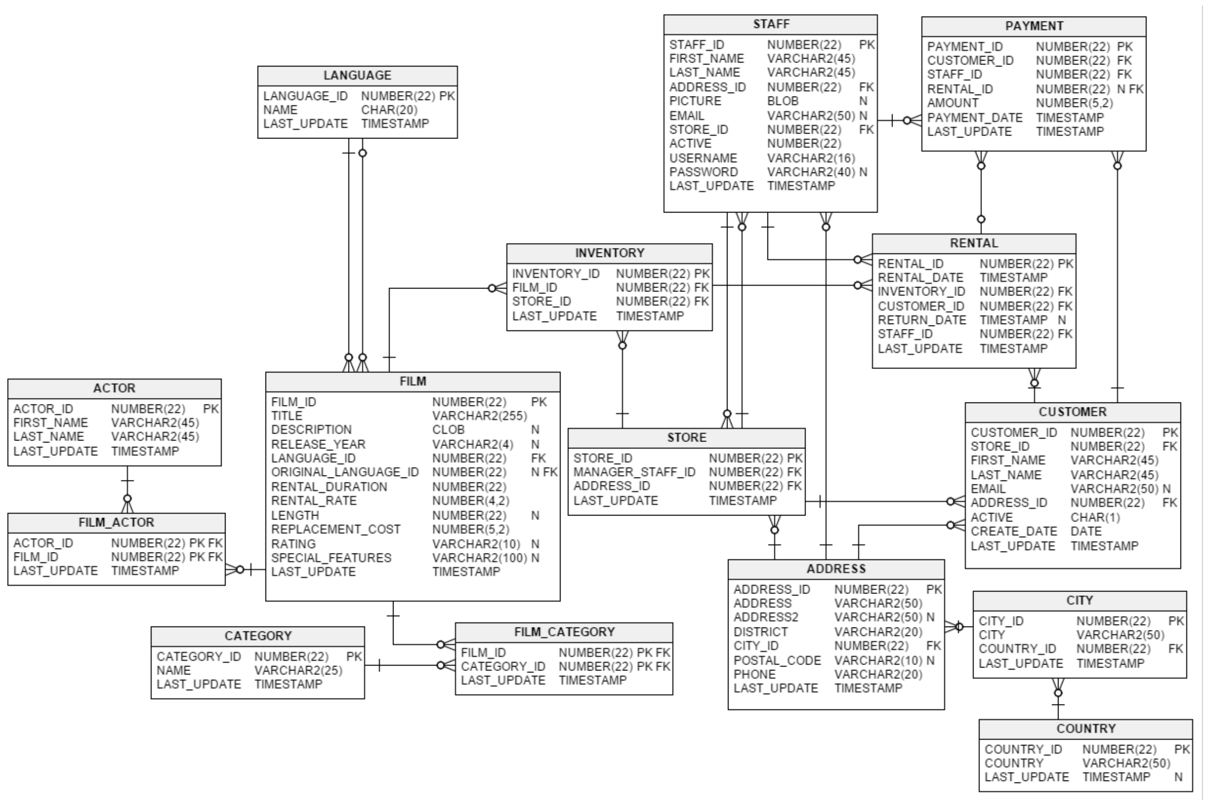
Introduction

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)
```

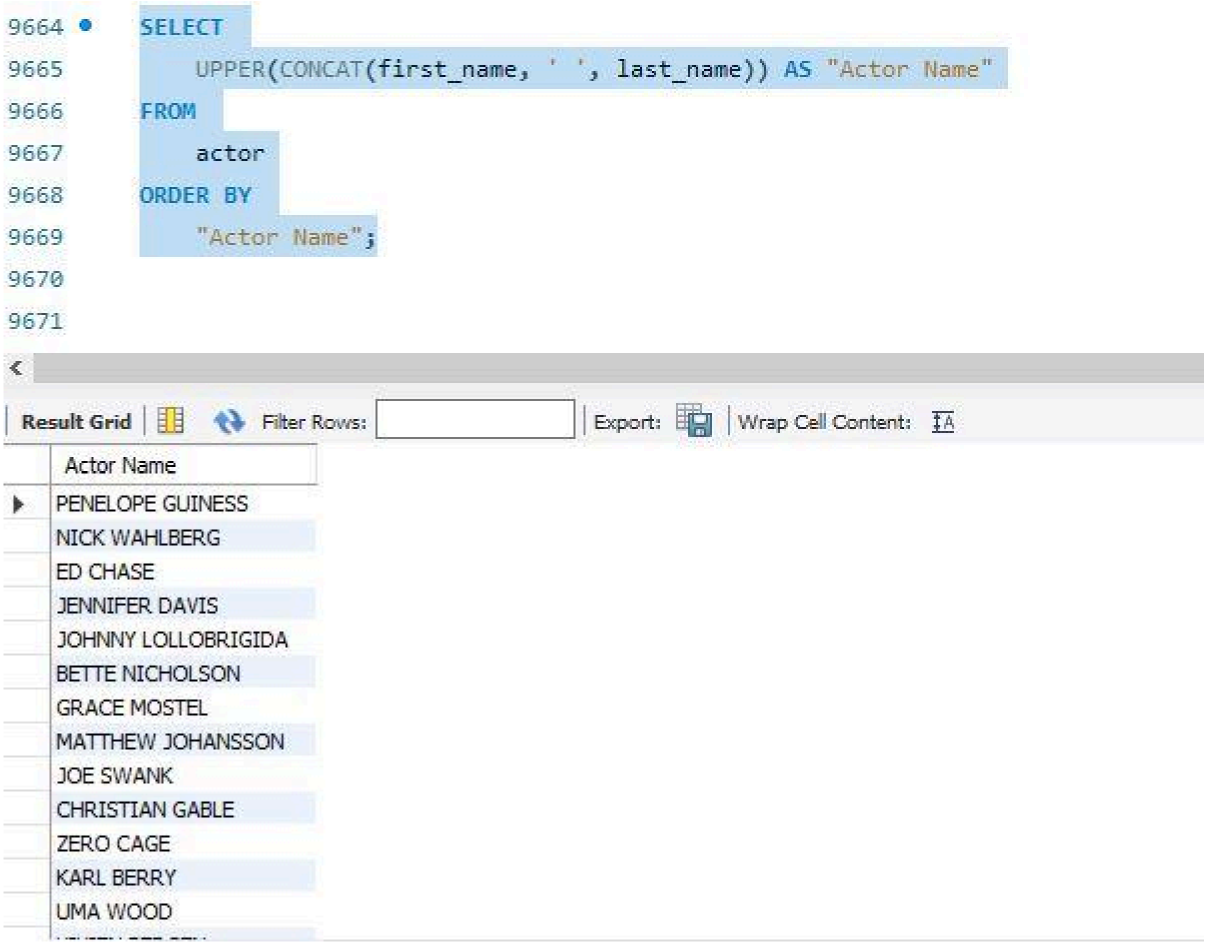
```
mysql> SELECT COUNT(*) FROM film;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

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

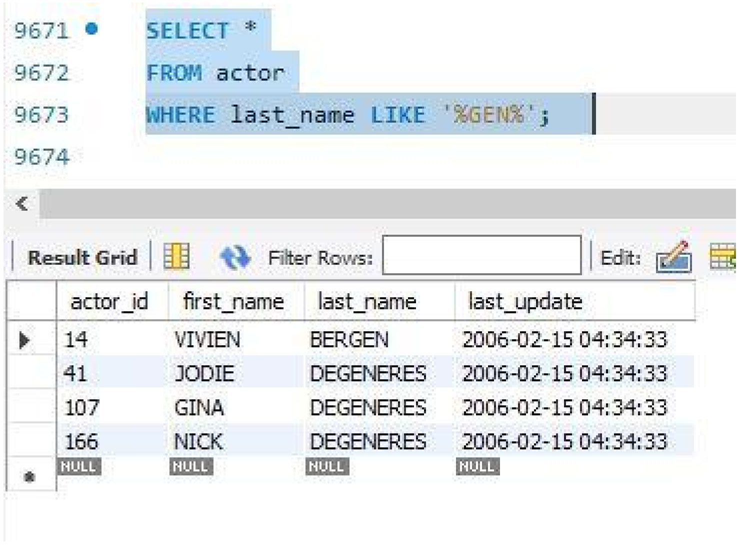
1. Display the first and last name of each actor in a single column in upper case letters in alphabetic order. Name the column Actor Name.



```
SELECT
    UPPER(CONCAT(first_name, ' ', last_name)) AS "Actor Name"
FROM
    actor
ORDER BY
    "Actor Name";
```

Actor Name
PENELOPE GUINESS
NICK WAHLBERG
ED CHASE
JENNIFER DAVIS
JOHNNY LOLLOBRIGIDA
BETTE NICHOLSON
GRACE MOSTEL
MATTHEW JOHANSSON
JOE SWANK
CHRISTIAN GABLE
ZERO CAGE
KARL BERRY
UMA WOOD

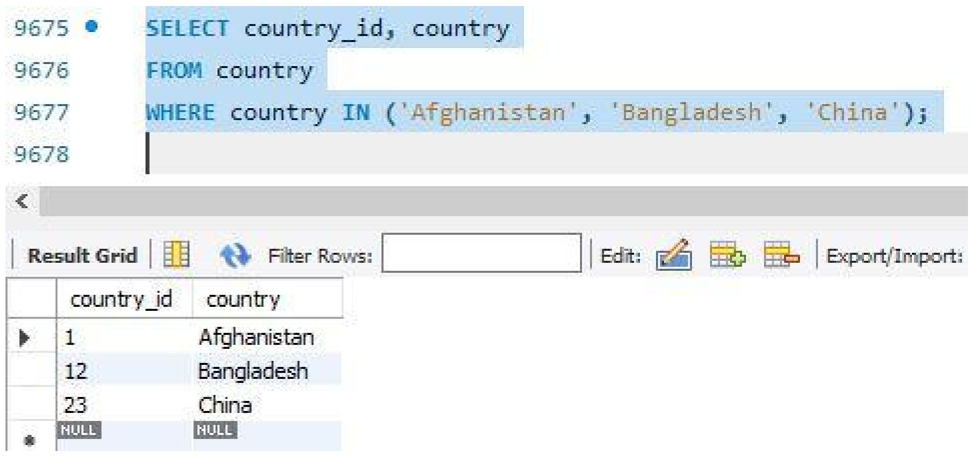
2. Find all actors whose last name contain the letters GEN:



```
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
NULL	NULL	NULL	NULL

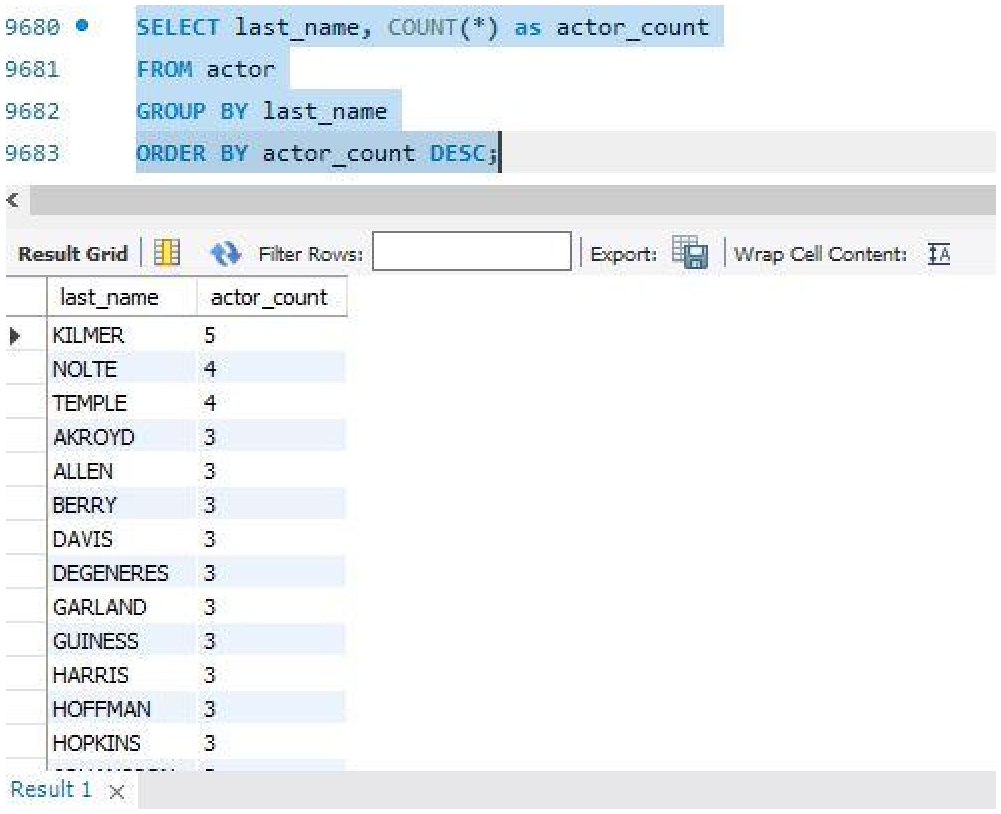
3. Using IN, display the country_id and country columns of the following countries: Afghanistan, Bangladesh, and China:



```sql
SELECT country_id, country
FROM country
WHERE country IN ('Afghanistan', 'Bangladesh', 'China');
```

country_id	country
1	Afghanistan
12	Bangladesh
23	China
NULL	NULL

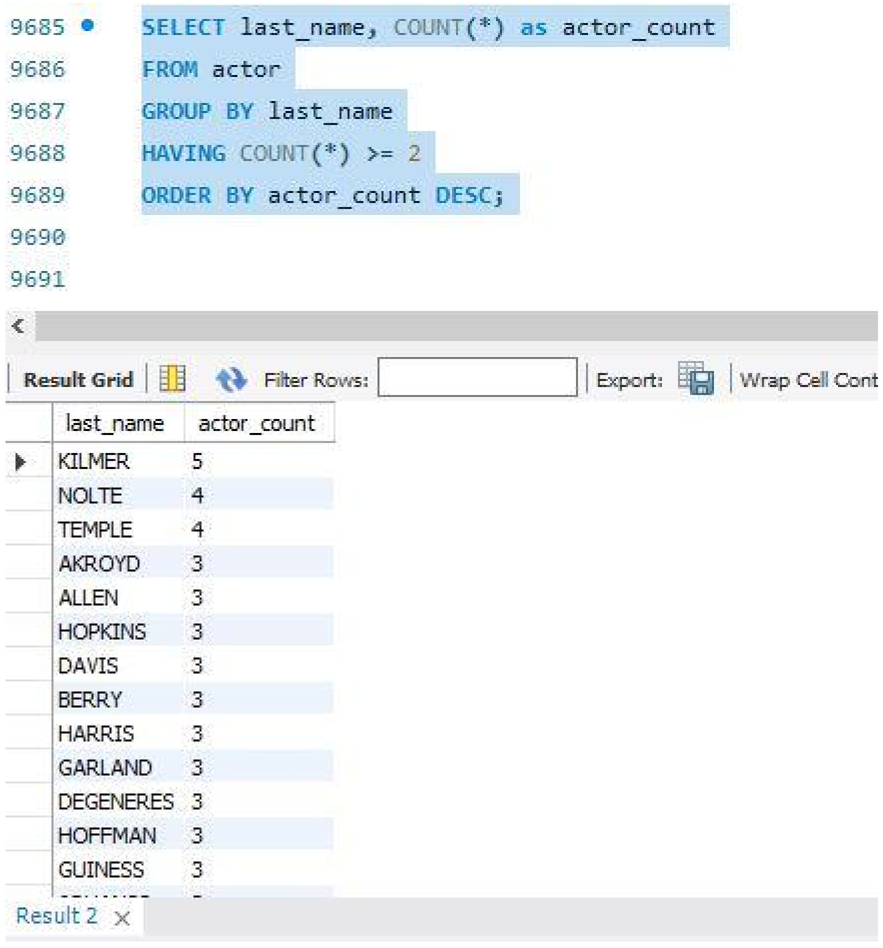
4. List the last names of actors, as well as how many actors have that last name.



```sql
SELECT last_name, COUNT(*) as actor_count
FROM actor
GROUP BY last_name
ORDER BY actor_count DESC;
```

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

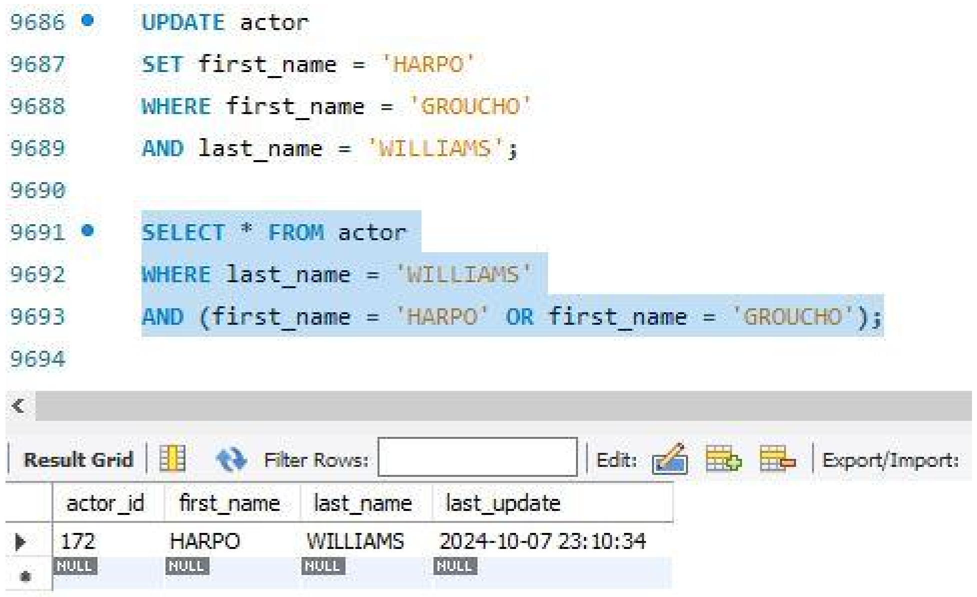
5. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors



```sql
SELECT last_name, COUNT(*) as actor_count
FROM actor
GROUP BY last_name
HAVING COUNT(*) >= 2
ORDER BY actor_count DESC;
```

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

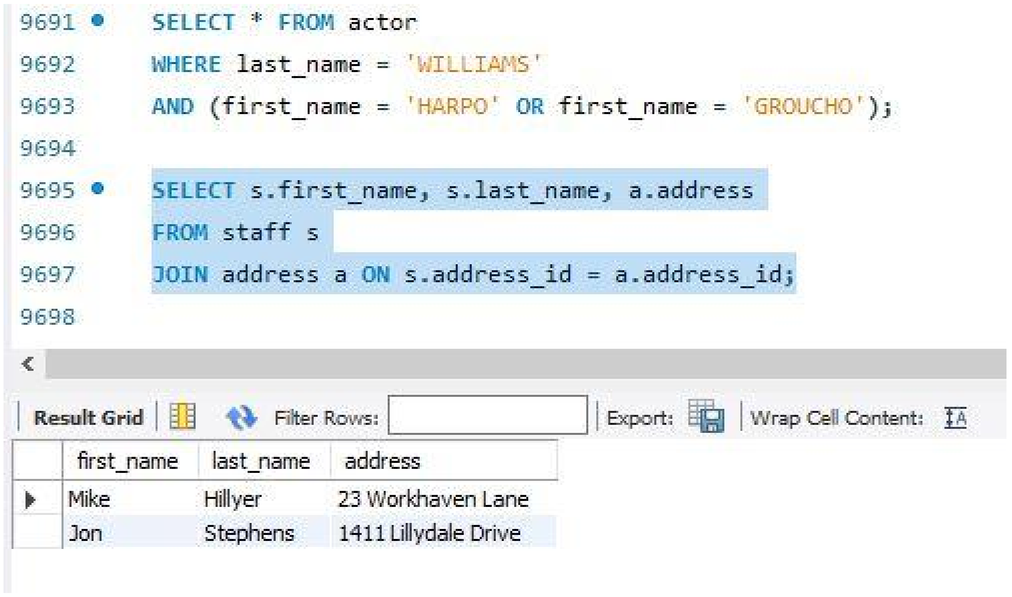
6. The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS. Write a query to fix the record.



```sql
UPDATE actor
SET first_name = 'HARPO'
WHERE first_name = 'GROUCHO'
AND last_name = 'WILLIAMS';

SELECT * FROM actor
WHERE last_name = 'WILLIAMS'
AND (first_name = 'HARPO' OR first_name = 'GROUCHO');
```

actor_id	first_name	last_name	last_update
172	HARPO	WILLIAMS	2024-10-07 23:10:34
NULL	NULL	NULL	NULL

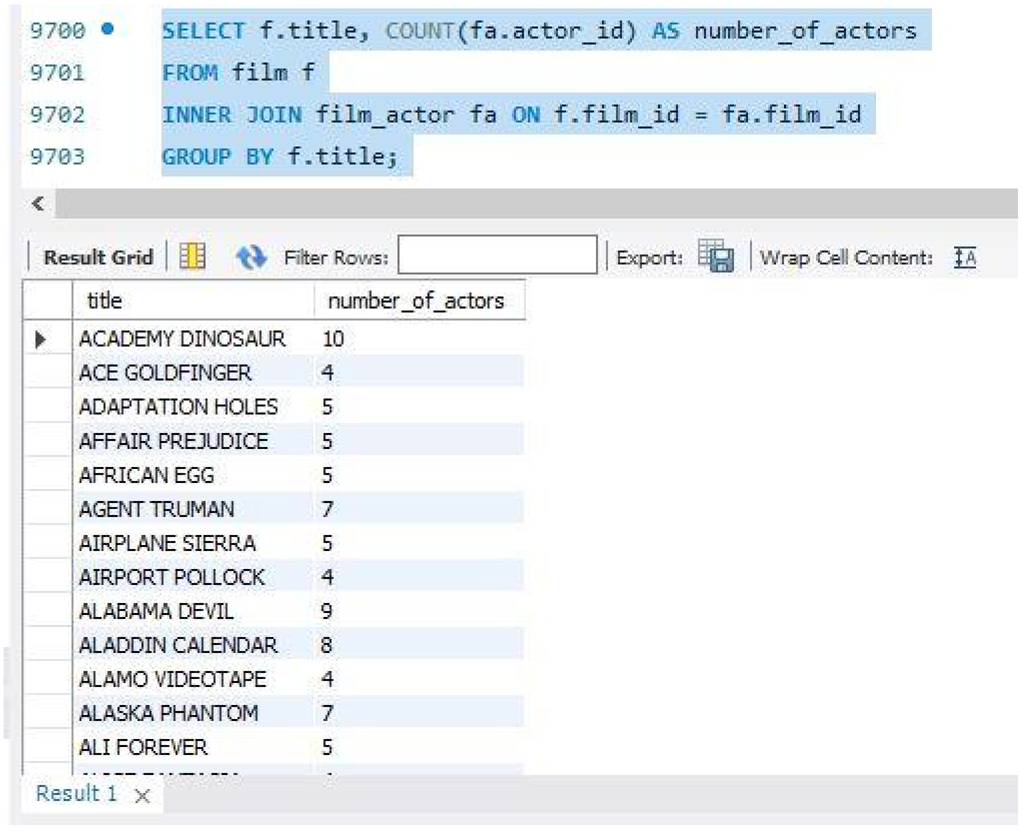
7. Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables staff and address:



```sql
SELECT * FROM actor
WHERE last_name = 'WILLIAMS'
AND (first_name = 'HARPO' OR first_name = 'GROUCHO');

SELECT s.first_name, s.last_name, a.address
FROM staff s
JOIN address a ON s.address_id = a.address_id;
```

first_name	last_name	address
Mike	Hillyer	23 Workhaven Lane
Jon	Stephens	1411 Lillydale Drive

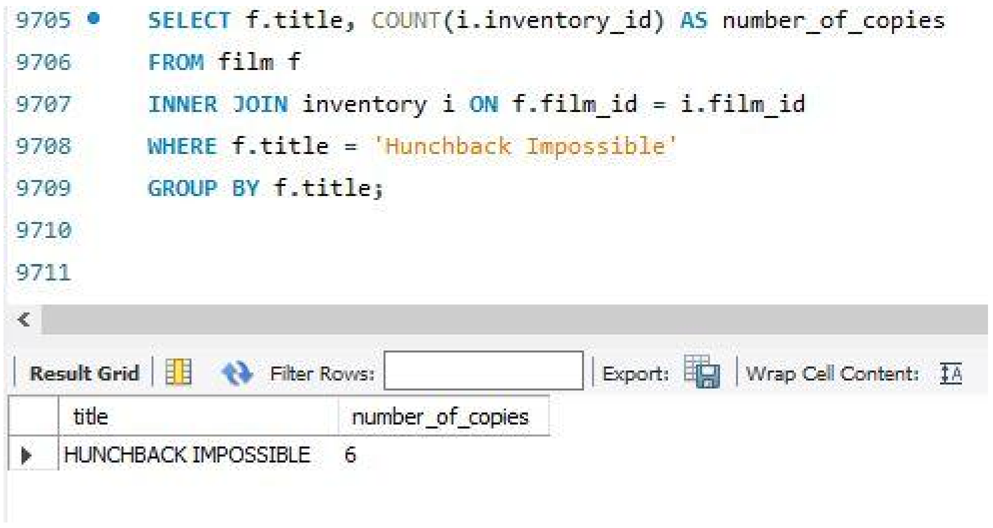
8. List each film and the number of actors who are listed for that film. Use tables film_actor and film. Use inner join.



```sql
SELECT f.title, COUNT(fa.actor_id) AS number_of_actors
FROM film f
INNER JOIN film_actor fa ON f.film_id = fa.film_id
GROUP BY f.title;
```

title	number_of_actors
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

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

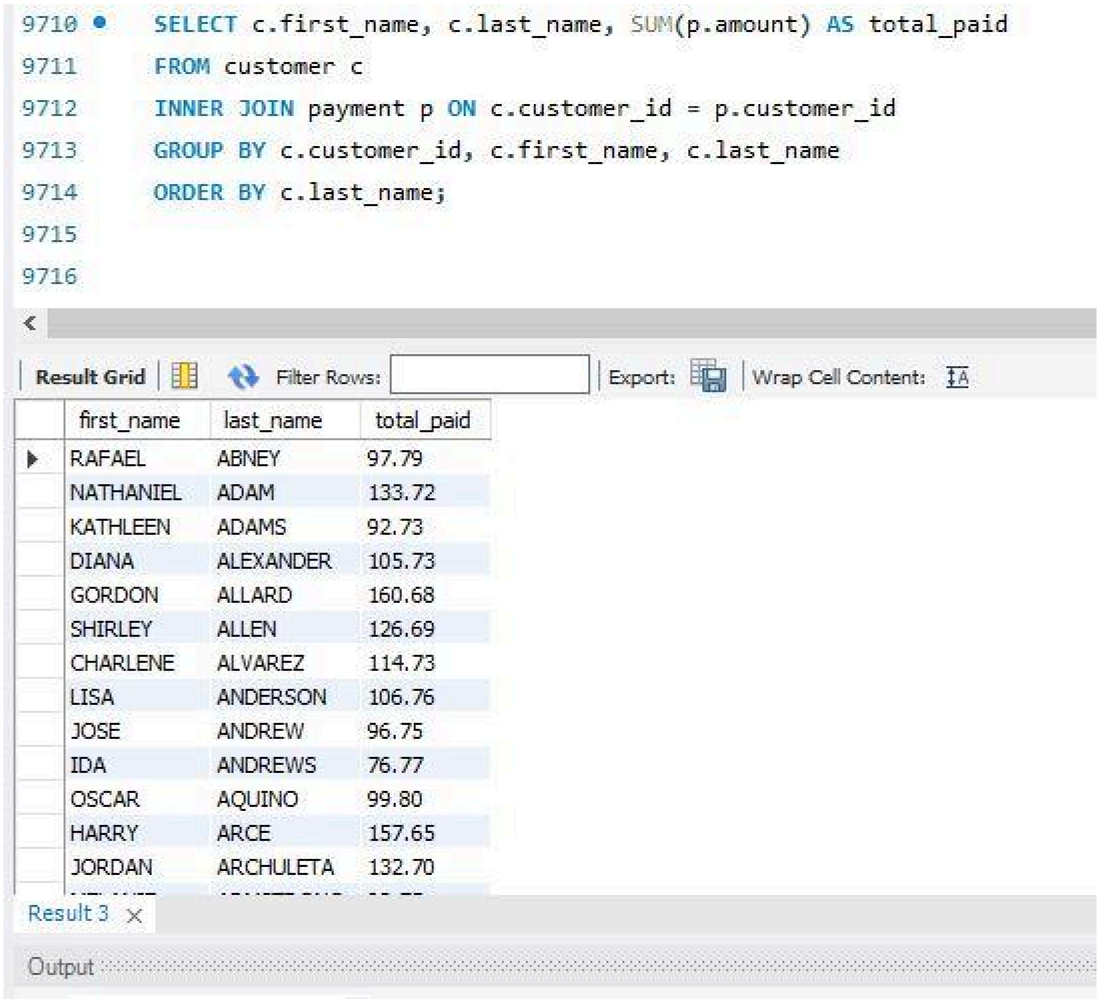


```sql
SELECT f.title, COUNT(i.inventory_id) AS number_of_copies
FROM film f
INNER JOIN inventory i ON f.film_id = i.film_id
WHERE f.title = 'Hunchback Impossible'
GROUP BY f.title;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
title                 number_of_copies
HUNCHBACK IMPOSSIBLE  6
```

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



```sql
SELECT c.first_name, c.last_name, SUM(p.amount) AS total_paid
FROM customer c
INNER JOIN payment p ON c.customer_id = p.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY c.last_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

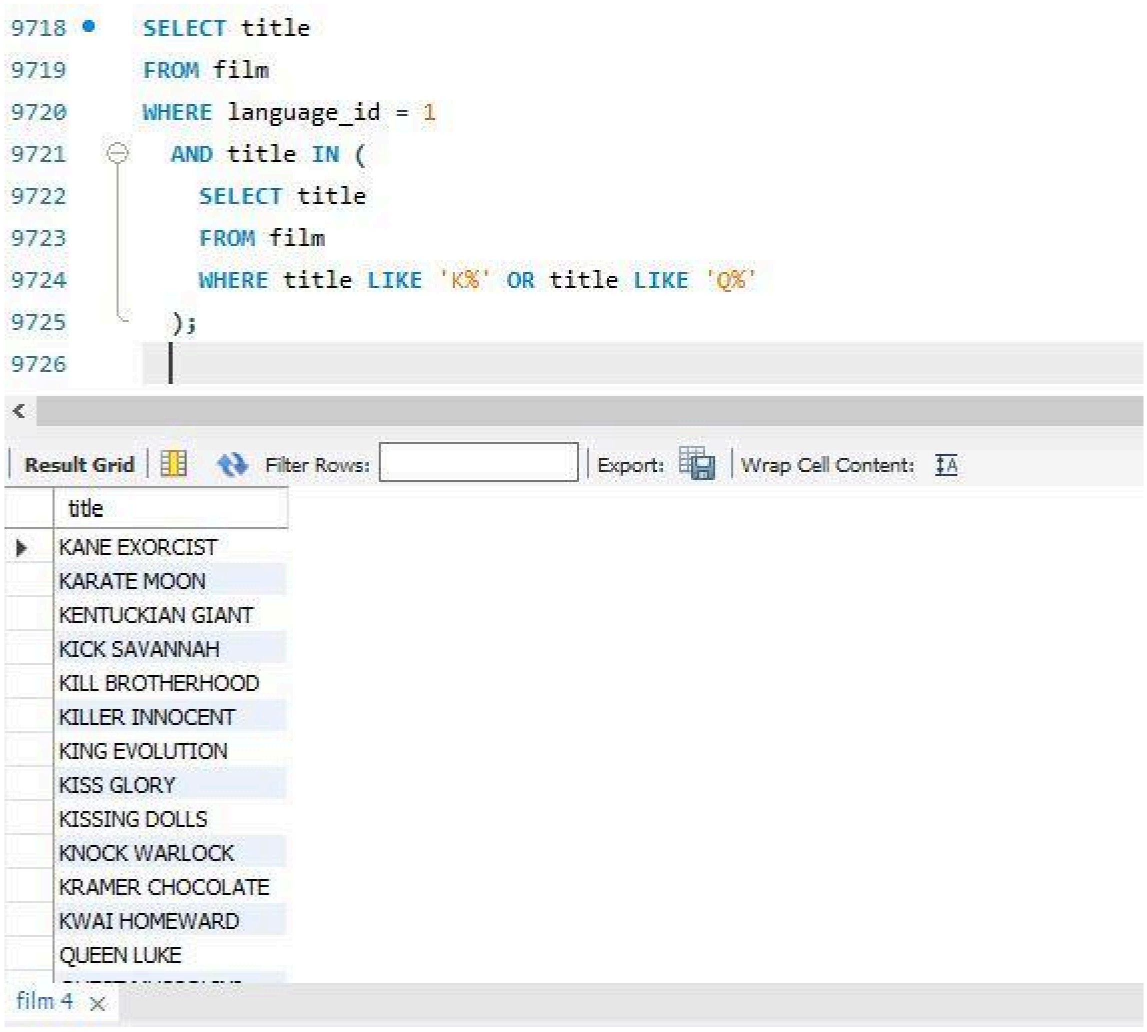
```
first_name  last_name  total_paid
RAFAEL      ABNEY      97.79
NATHANIEL   ADAM       133.72
KATHLEEN    ADAMS      92.73
DIANA       ALEXANDER  105.73
GORDON      ALLARD     160.68
SHIRLEY     ALLEN      126.69
CHARLENE    ALVAREZ    114.73
LISA        ANDERSON   106.76
JOSE        ANDREW     96.75
IDA         ANDREWS    76.77
OSCAR       AQUINO     99.80
HARRY       ARCE       157.65
JORDAN      ARCHULETA  132.70
```

Result 3 ×

Output

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.



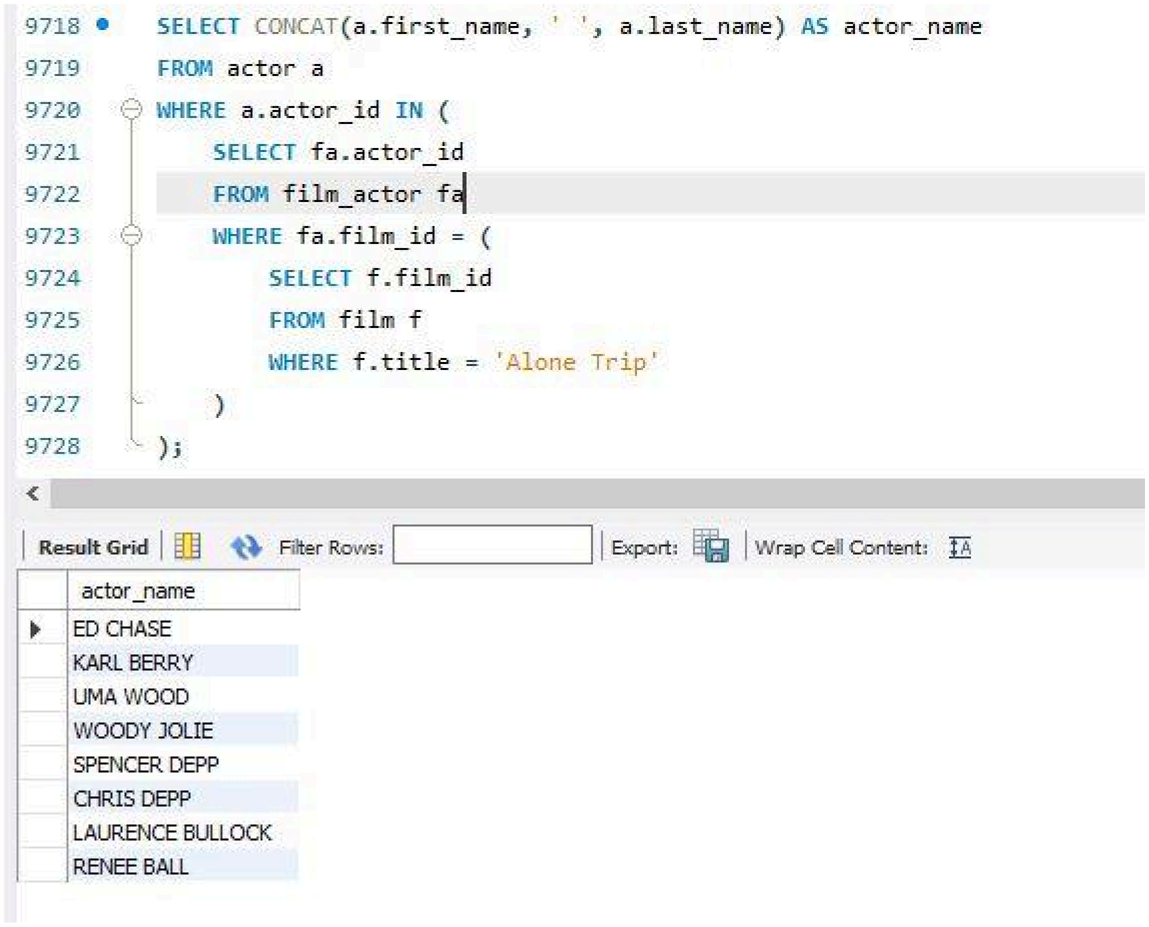
```
SELECT title
FROM film
WHERE language_id = 1
  AND title IN (
    SELECT title
    FROM film
    WHERE title LIKE 'K%' OR title LIKE 'Q%'
  );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

film 4

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

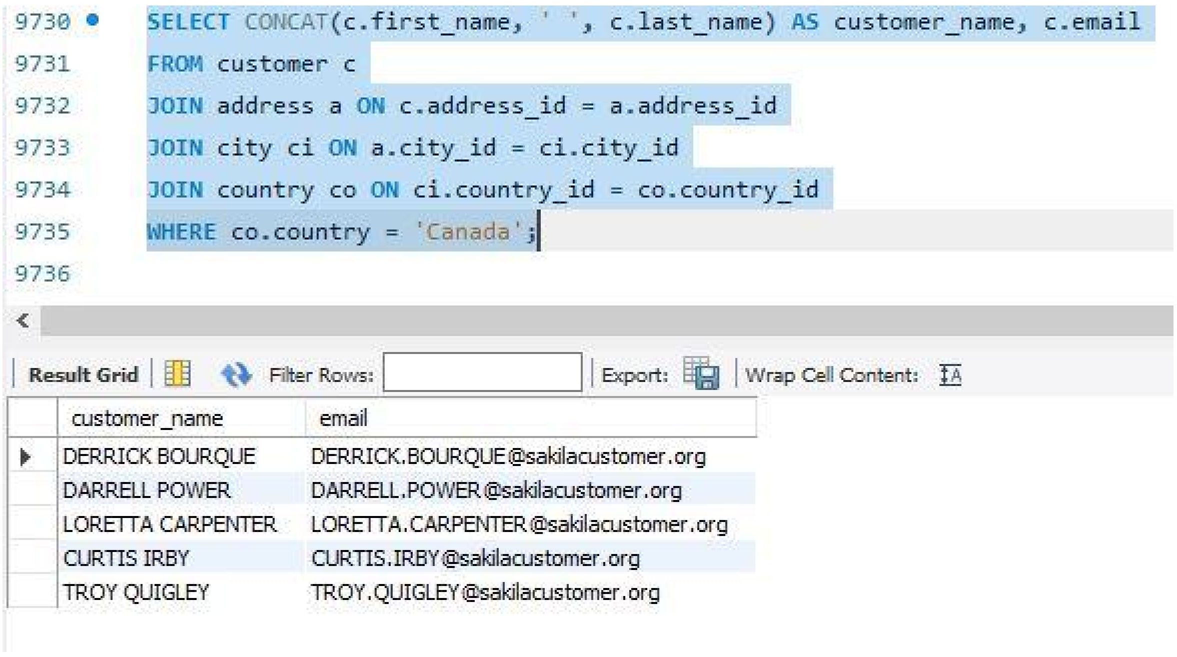


```sql
SELECT CONCAT(a.first_name, ' ', a.last_name) AS actor_name
FROM actor a
WHERE a.actor_id IN (
    SELECT fa.actor_id
    FROM film_actor fa
    WHERE fa.film_id = (
        SELECT f.film_id
        FROM film f
        WHERE f.title = 'Alone Trip'
    )
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

actor_name
ED CHASE
KARL BERRY
UMA WOOD
WOODY JOLIE
SPENCER DEPP
CHRIS DEPP
LAURENCE BULLOCK
RENEE BALL

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.

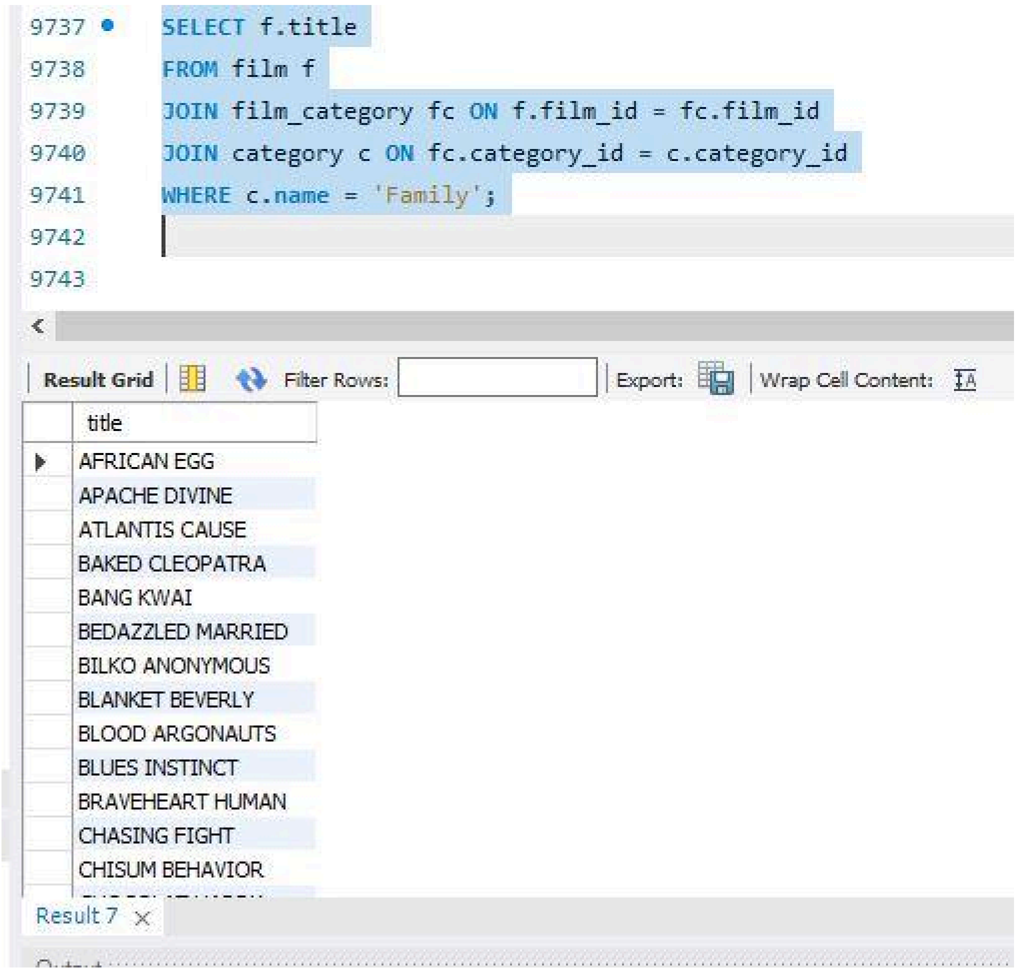


```sql
SELECT CONCAT(c.first_name, ' ', c.last_name) AS customer_name, c.email
FROM customer c
JOIN address a ON c.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country co ON ci.country_id = co.country_id
WHERE co.country = 'Canada';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_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

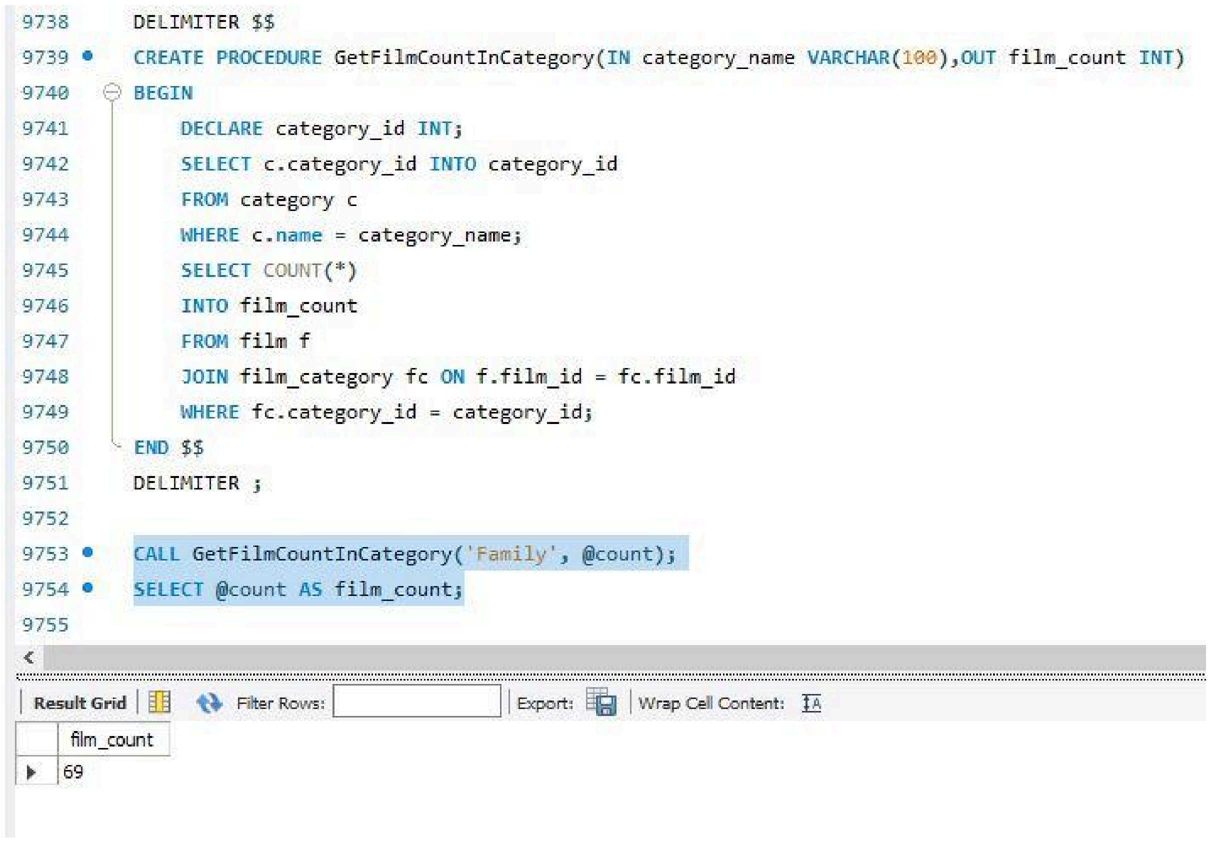
14. Sales have been lagging among young families, and you wish to target all family movies for a promotion. Identify all movies categorized as famiy films.



```sql
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';
```

```
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
```

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

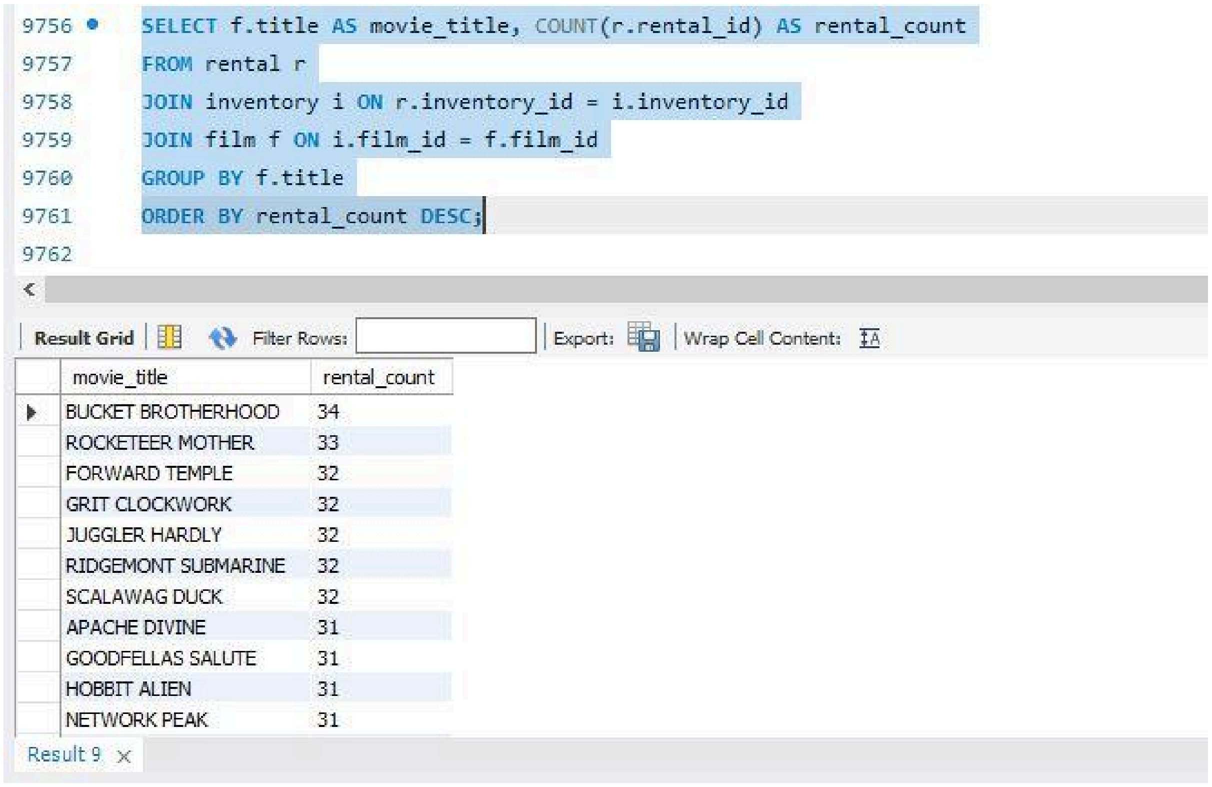


```
DELIMITER $$
CREATE PROCEDURE GetFilmCountInCategory(IN category_name VARCHAR(100),OUT film_count INT)
BEGIN
    DECLARE category_id INT;
    SELECT c.category_id INTO category_id
    FROM category c
    WHERE c.name = category_name;
    SELECT COUNT(*)
    INTO film_count
    FROM film f
    JOIN film_category fc ON f.film_id = fc.film_id
    WHERE fc.category_id = category_id;
END $$
DELIMITER ;

CALL GetFilmCountInCategory('Family', @count);
SELECT @count AS film_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

film_count
69

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



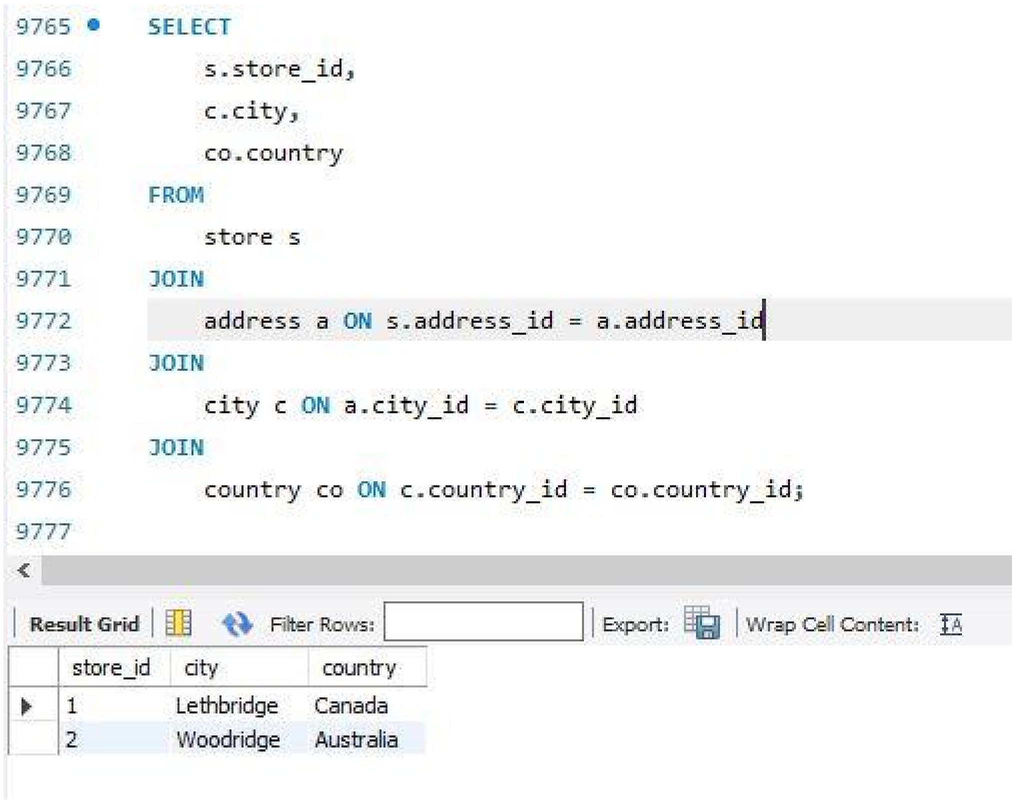
```
SELECT f.title AS movie_title, COUNT(r.rental_id) AS rental_count
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
GROUP BY f.title
ORDER BY rental_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Result 9

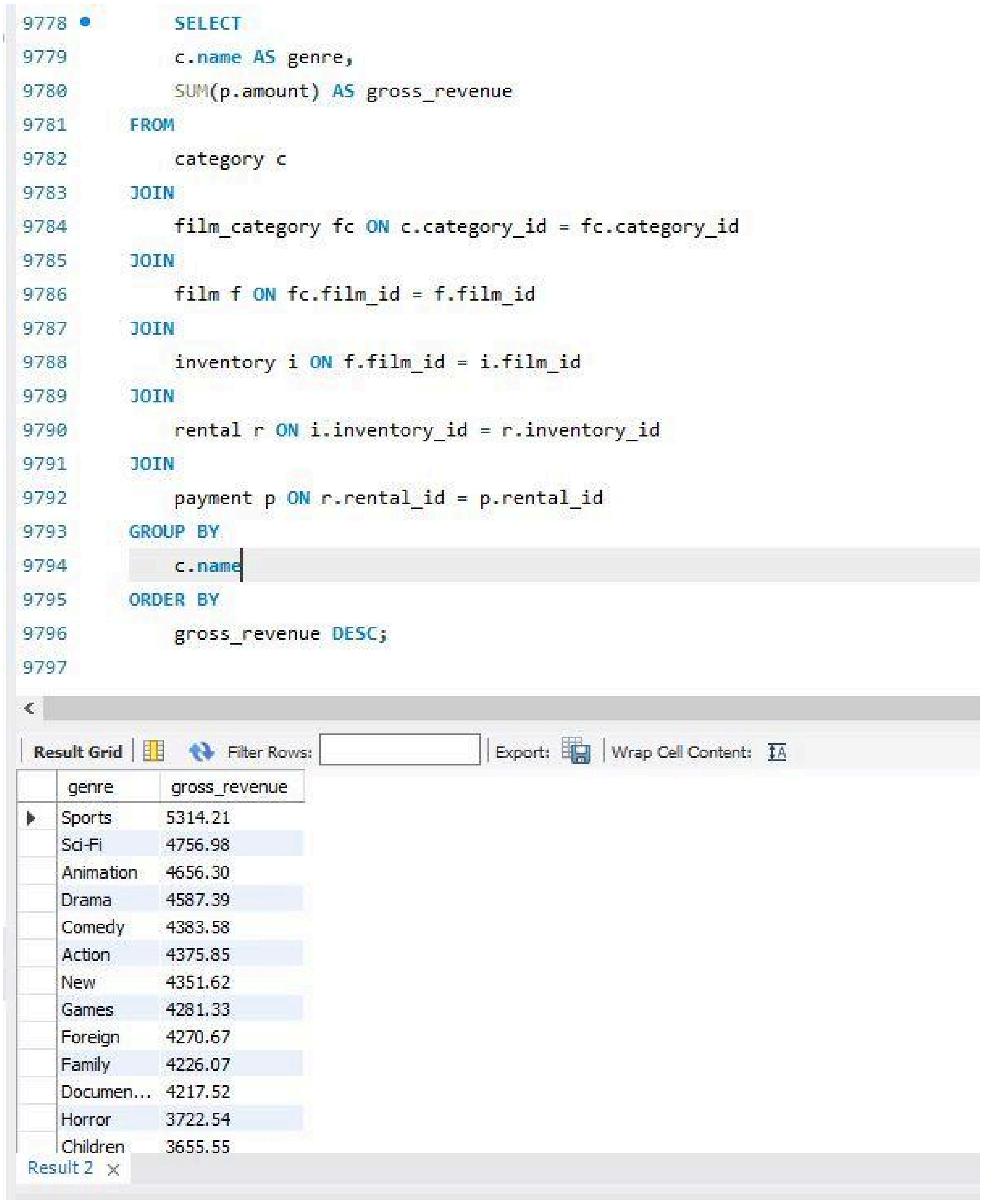
17. Write a query to display for each store its store ID, city, and country.



```
SELECT
    s.store_id,
    c.city,
    co.country
FROM
    store s
JOIN
    address a ON s.address_id = a.address_id
JOIN
    city c ON a.city_id = c.city_id
JOIN
    country co ON c.country_id = co.country_id;
```

store_id	city	country
1	Lethbridge	Canada
2	Woodridge	Australia

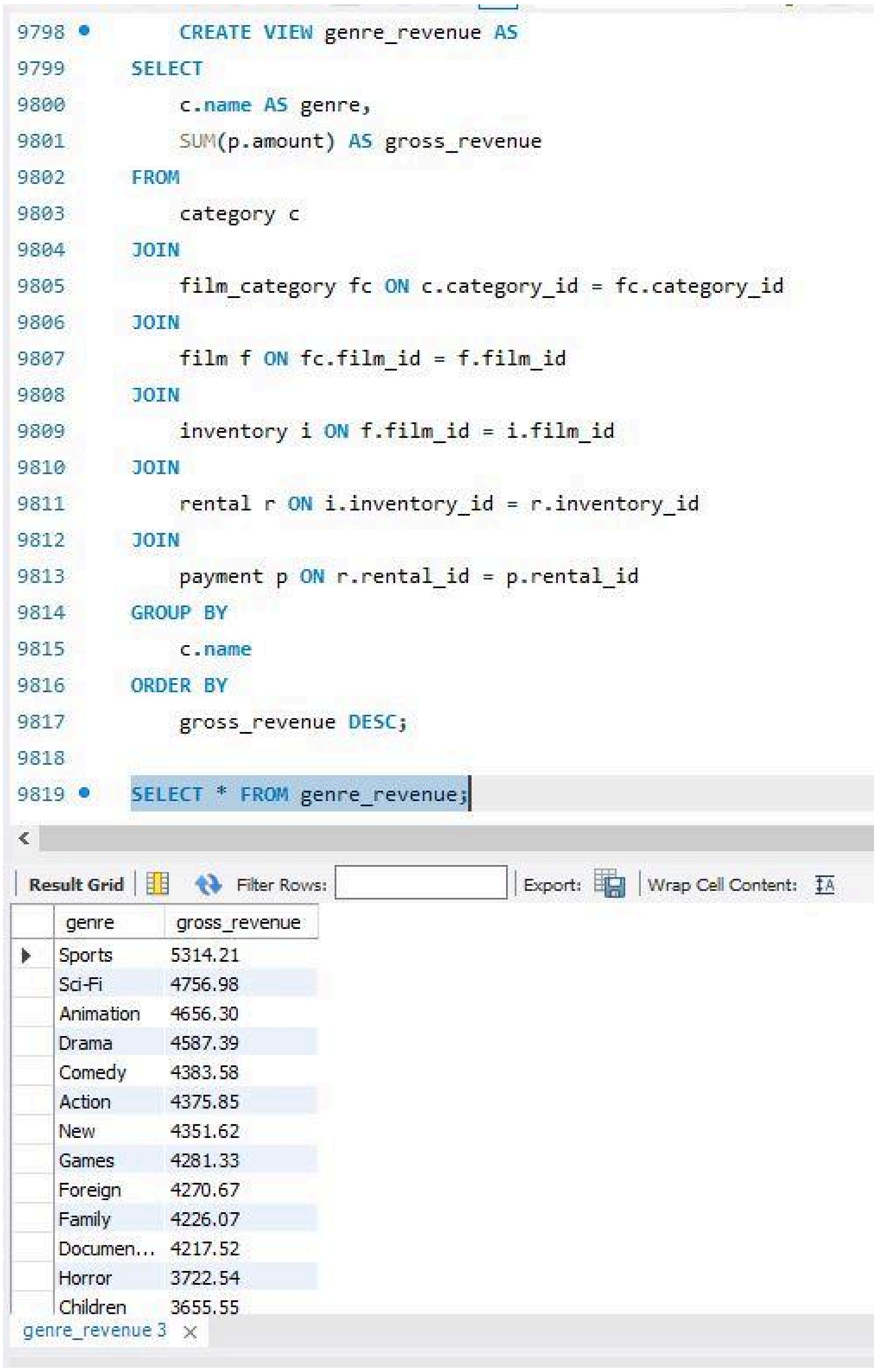
18. List the genres and its gross revenue.



```
    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
Documen...	4217.52
Horror	3722.54
Children	3655.55

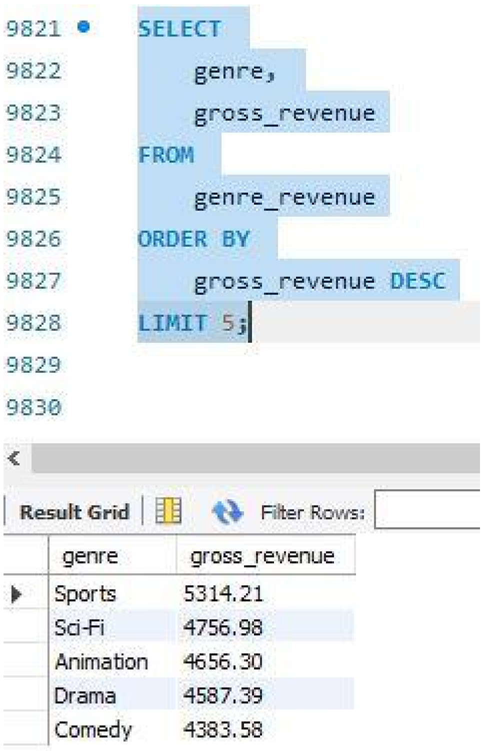
19. Create a View for the above query(18)



```
CREATE VIEW genre_revenue 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;

SELECT * FROM genre_revenue;
```

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
Documen...	4217.52
Horror	3722.54
Children	3655.55

20. Select top 5 genres in gross revenue view.



```sql
SELECT
    genre,
    gross_revenue
FROM
    genre_revenue
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