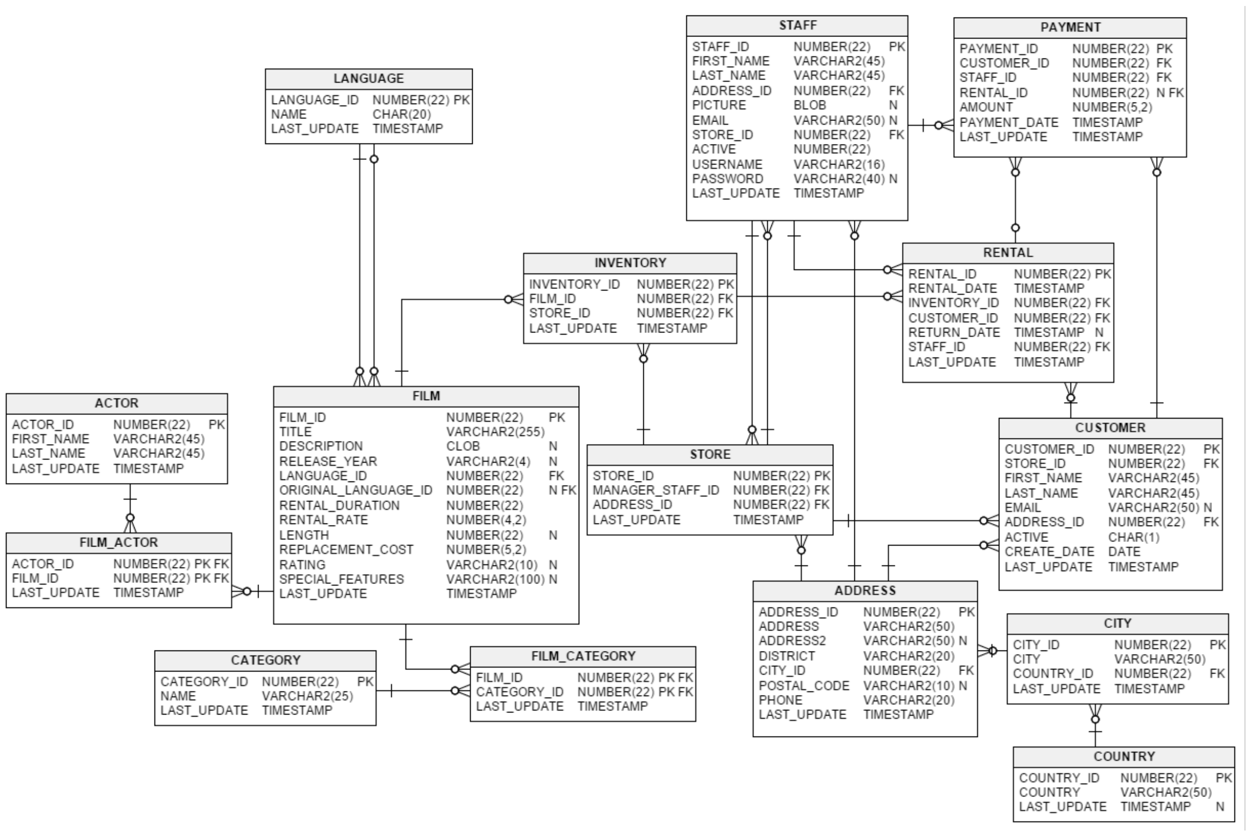
Introduction

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)
```

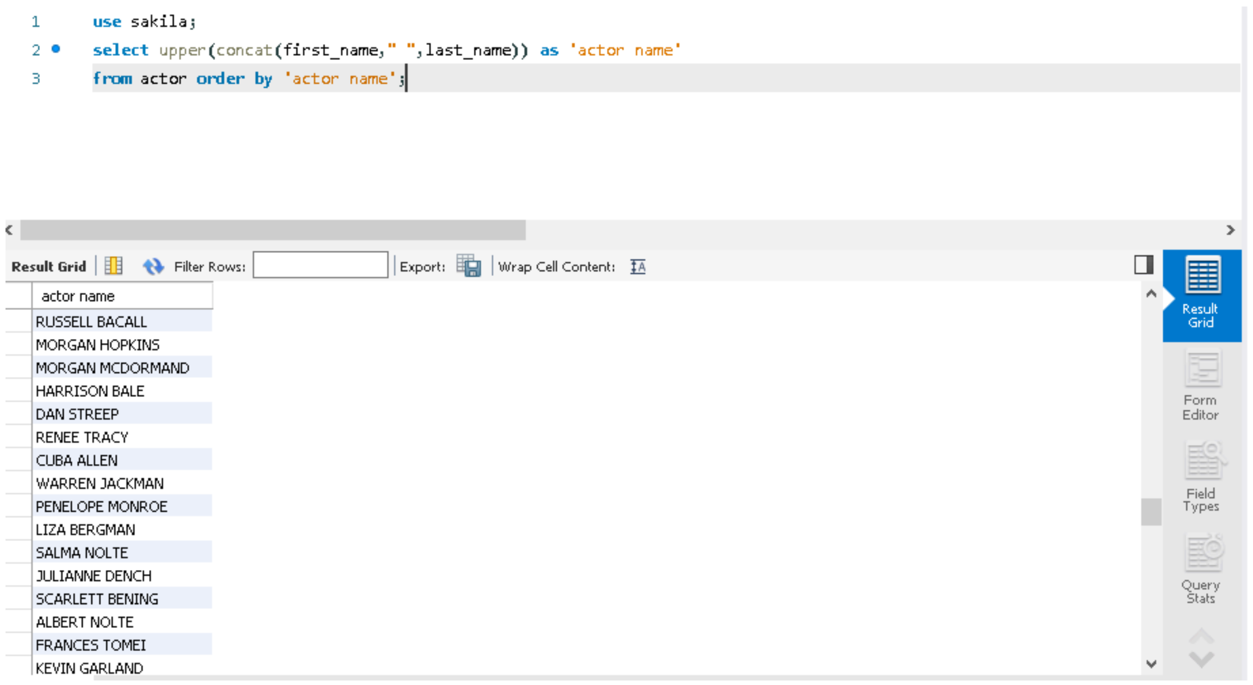
```
mysql> SELECT COUNT(*) FROM film;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

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

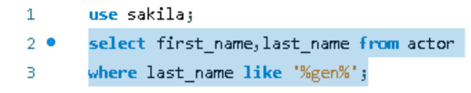
1. Display the first and last name of each actor in a single column in upper case letters in alphabetic order. Name the column Actor Name.



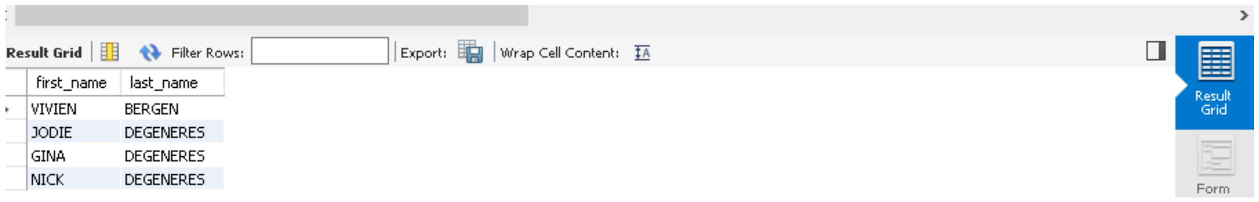
```
use sakila;
select upper(concat(first_name," ",last_name)) as 'actor name'
from actor order by 'actor name';
```

```
actor name
RUSSELL BACALL
MORGAN HOPKINS
MORGAN MCDORMAND
HARRISON BALE
DAN STREEP
RENEE TRACY
CUBA ALLEN
WARREN JACKMAN
PENELOPE MONROE
LIZA BERGMAN
SALMA NOLTE
JULIANNE DENCH
SCARLETT BENING
ALBERT NOLTE
FRANCES TOMEI
KEVIN GARLAND
```

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

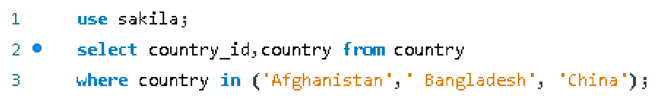


```
use sakila;
select first_name,last_name from actor
where last_name like '%gen%';
```

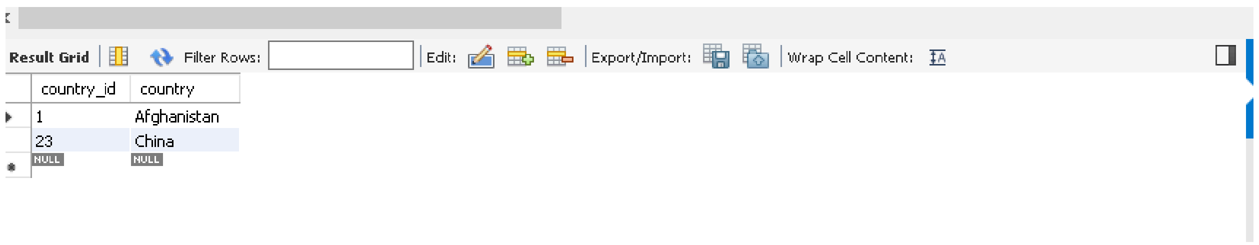


```
first_name  last_name
VIVIEN      BERGEN
JODIE       DEGENERES
GINA        DEGENERES
NICK        DEGENERES
```

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

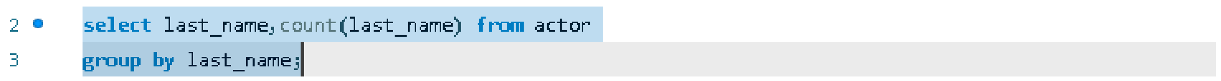


```
use sakila;
select country_id,country from country
where country in ('Afghanistan',' Bangladesh', 'China');
```

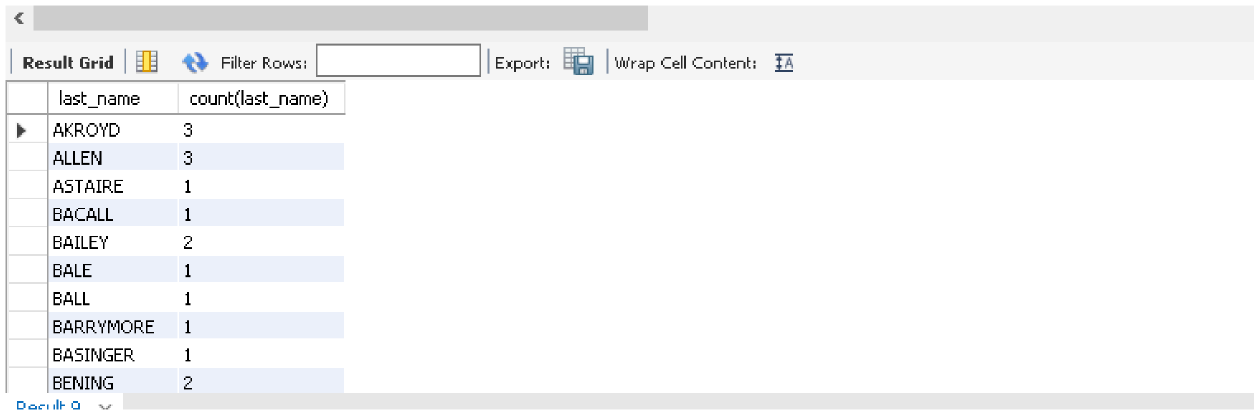


country_id	country
1	Afghanistan
23	China
NULL	NULL

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

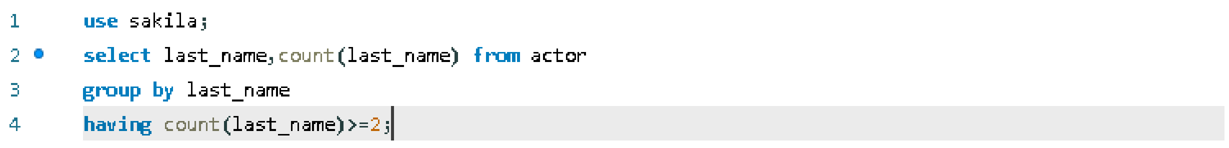


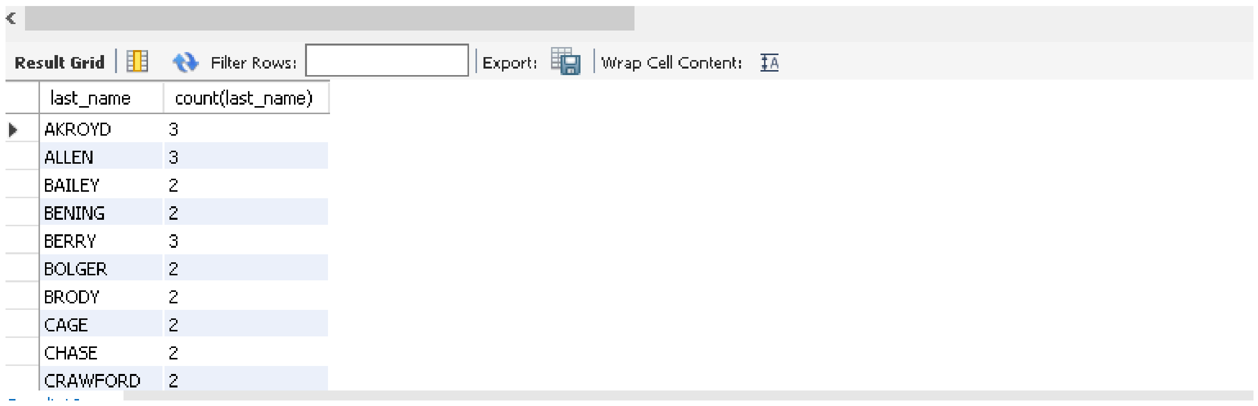
```
use sakila;
select last_name,count(last_name) from actor
group by last_name;
```



last_name	count(last_name)
AKROYD	3
ALLEN	3
ASTAIRE	1
BACALL	1
BAILEY	2
BALE	1
BALL	1
BARRYMORE	1
BASINGER	1
BENING	2

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

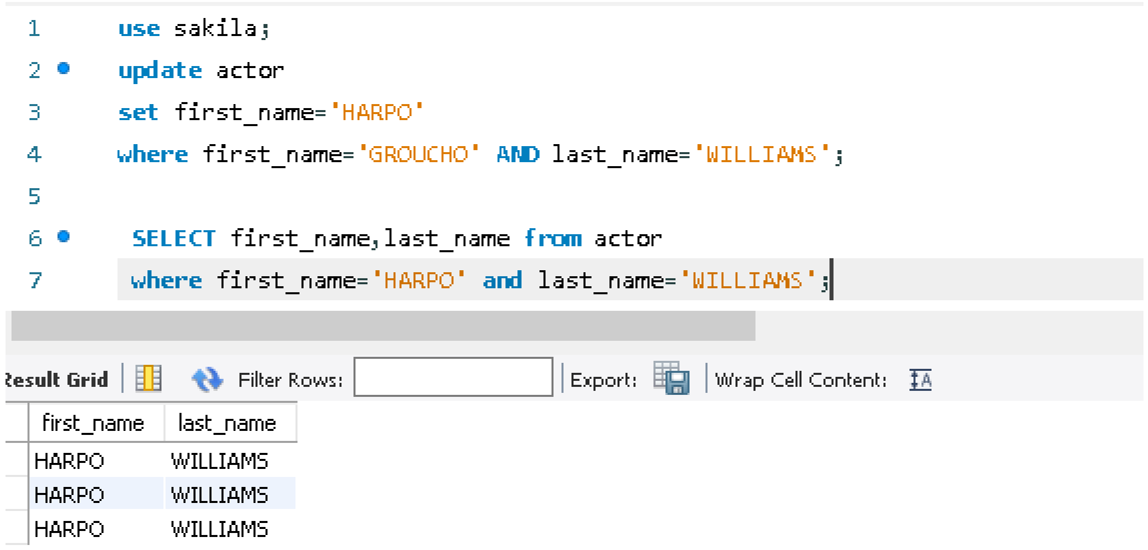

```
use sakila;
select last_name,count(last_name) from actor
group by last_name
having count(last_name)>=2;
```



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

last_name	count(last_name)
AKROYD	3
ALLEN	3
BAILEY	2
BENING	2
BERRY	3
BOLGER	2
BRODY	2
CAGE	2
CHASE	2
CRAWFORD	2

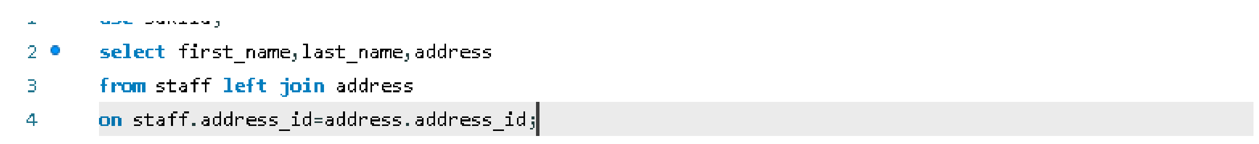
6. The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS. Write a query to fix the record.


```
use sakila;
update actor
set first_name='HARPO'
where first_name='GROUCHO' AND last_name='WILLIAMS';

 SELECT first_name,last_name from actor
 where first_name='HARPO' and last_name='WILLIAMS';
```

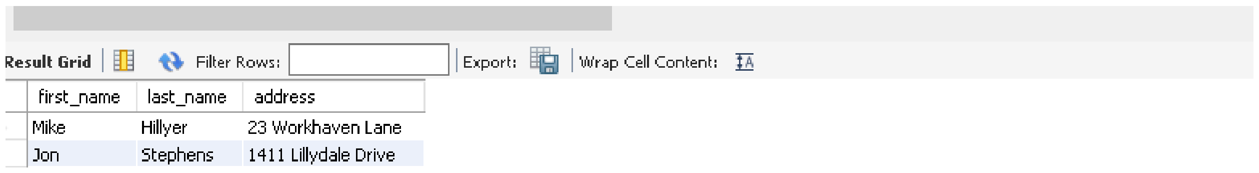
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name
HARPO	WILLIAMS
HARPO	WILLIAMS
HARPO	WILLIAMS

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:

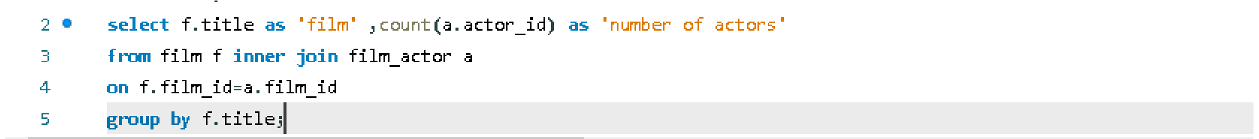


```
select first_name,last_name,address
from staff left join address
on staff.address_id=address.address_id;
```

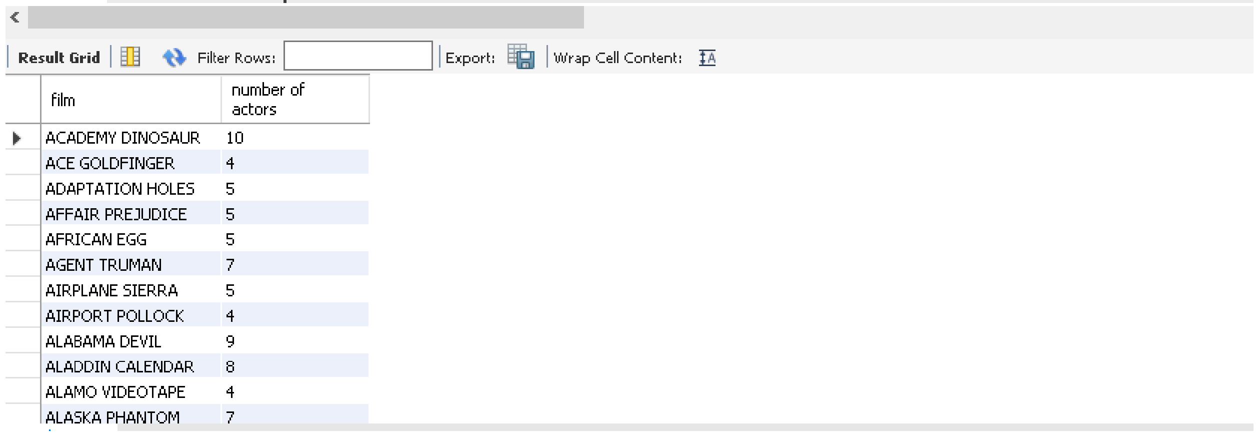


first_name	last_name	address
Mike	Hillyer	23 Workhaven Lane
Jon	Stephens	1411 Lillydale Drive

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

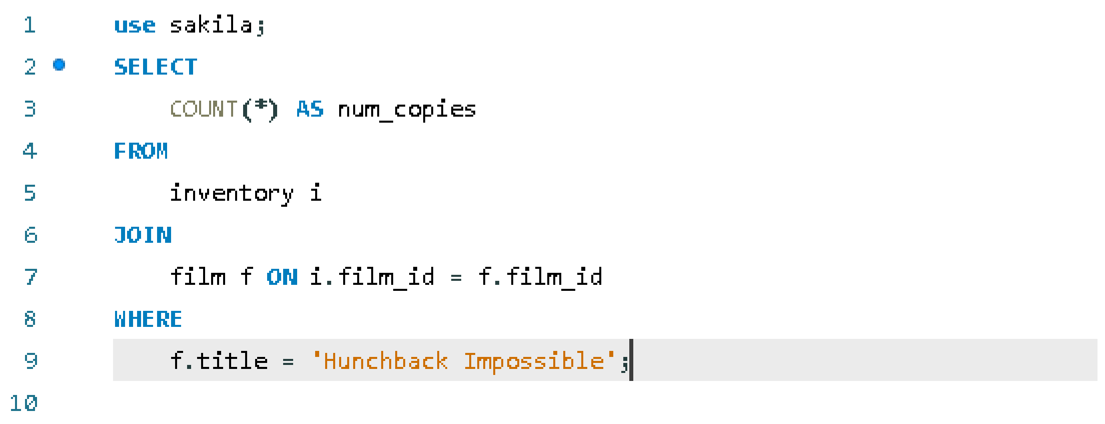


```
select f.title as 'film' ,count(a.actor_id) as 'number of actors'
from film f inner join film_actor a
on f.film_id=a.film_id
group by f.title;
```

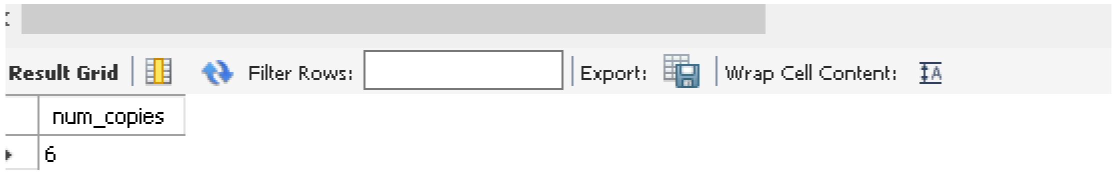


film	number of actors
ACADEMY DINOSAUR	10
ACE GOLDFINGER	4
ADAPTATION HOLES	5
AFFAIR PREJUDICE	5
AFRICAN EGG	5
AGENT TRUMAN	7
AIRPLANE SIERRA	5
AIRPORT POLLOCK	4
ALABAMA DEVIL	9
ALADDIN CALENDAR	8
ALAMO VIDEOTAPE	4
ALASKA PHANTOM	7

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

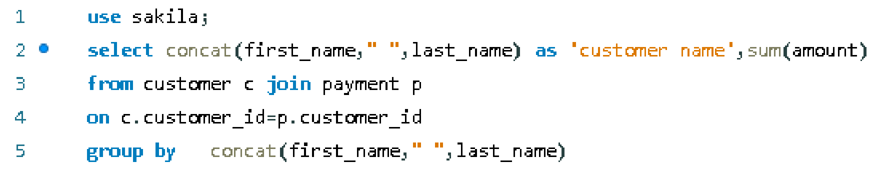


```
use sakila;
SELECT
    COUNT(*) AS num_copies
FROM
    inventory i
JOIN
    film f ON i.film_id = f.film_id
WHERE
    f.title = 'Hunchback Impossible';
```

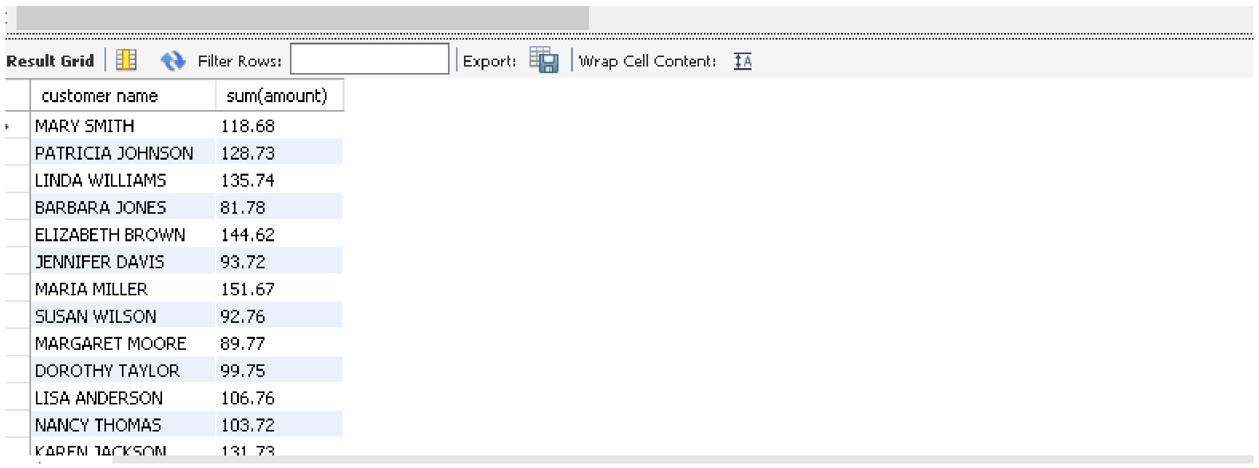


num_copies
6

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



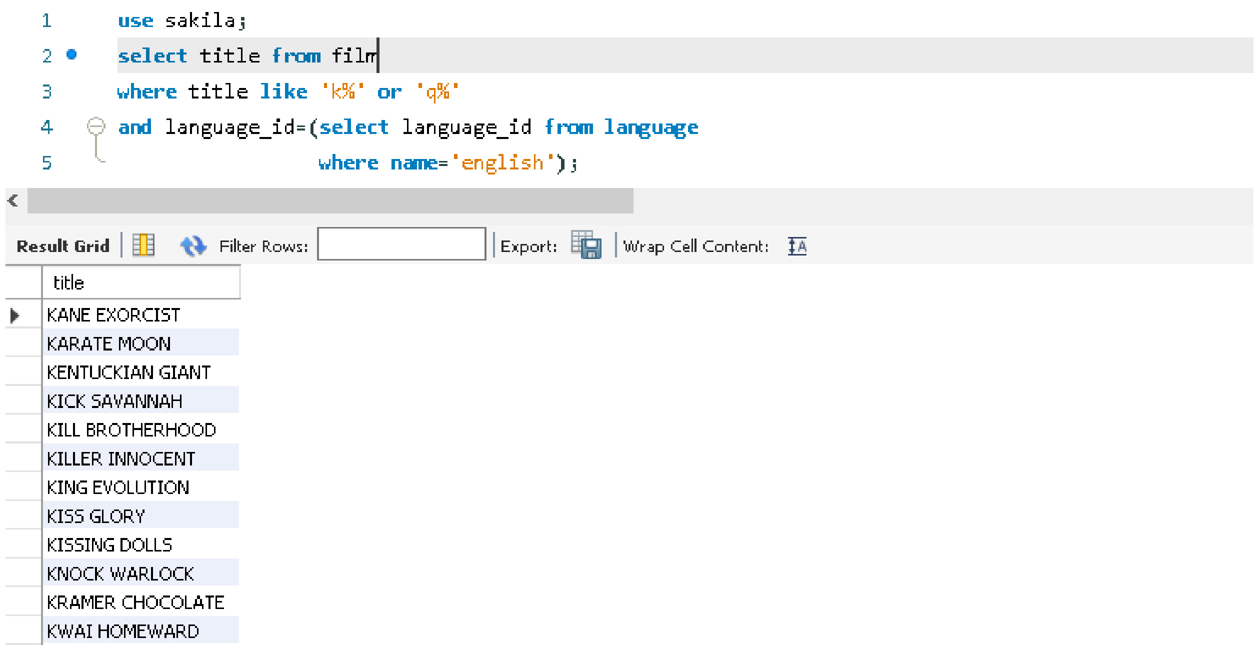
```
use sakila;
select concat(first_name," ",last_name) as 'customer name',sum(amount)
from customer c join payment p
on c.customer_id=p.customer_id
group by   concat(first_name," ",last_name)
```



customer name	sum(amount)
MARY SMITH	118.68
PATRICIA JOHNSON	128.73
LINDA WILLIAMS	135.74
BARBARA JONES	81.78
ELIZABETH BROWN	144.62
JENNIFER DAVIS	93.72
MARIA MILLER	151.67
SUSAN WILSON	92.76
MARGARET MOORE	89.77
DOROTHY TAYLOR	99.75
LISA ANDERSON	106.76
NANCY THOMAS	103.72
KAREN JACKSON	131.73

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.

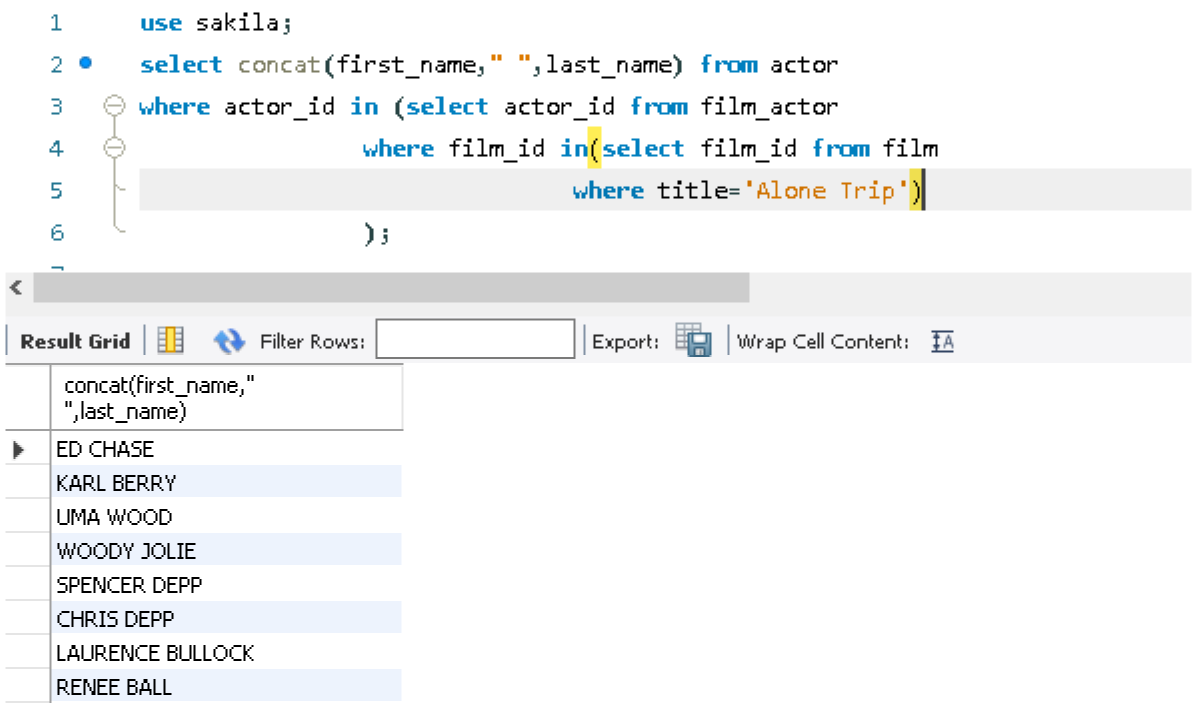


```
use sakila;
select title from film
where title like 'k%' or 'q%'
and language_id=(select language_id from language
                 where name='english');
```

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
```

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

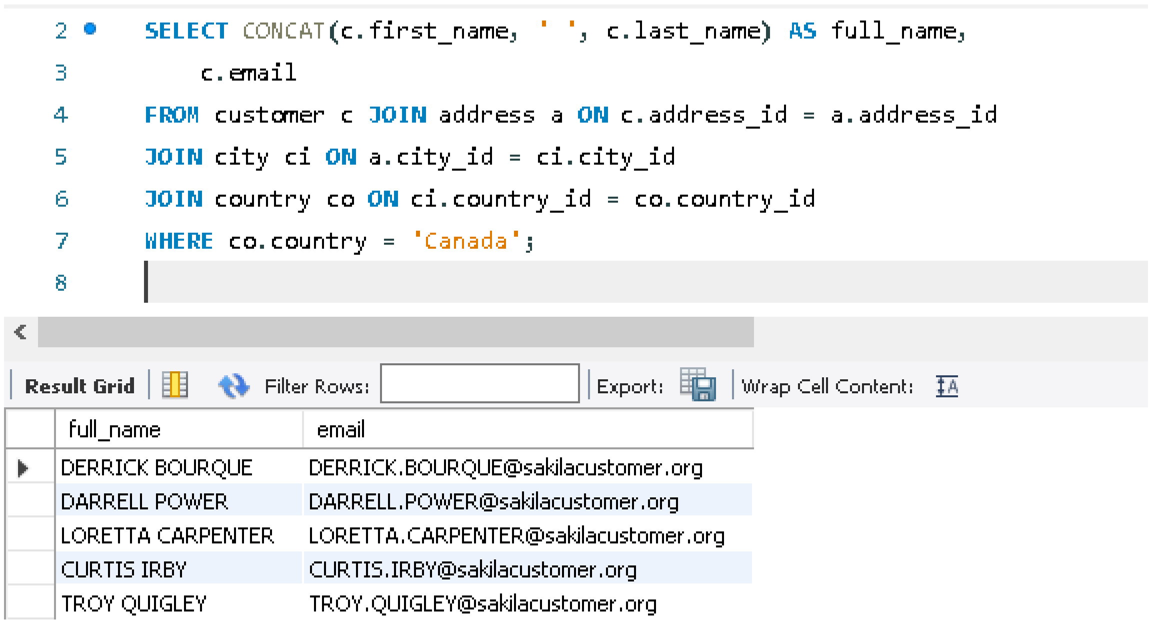


```
use sakila;
select concat(first_name," ",last_name) from actor
where actor_id in (select actor_id from film_actor
                   where film_id in(select film_id from film
                                    where title='Alone Trip')
                   );
```

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

```
concat(first_name,"
",last_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.

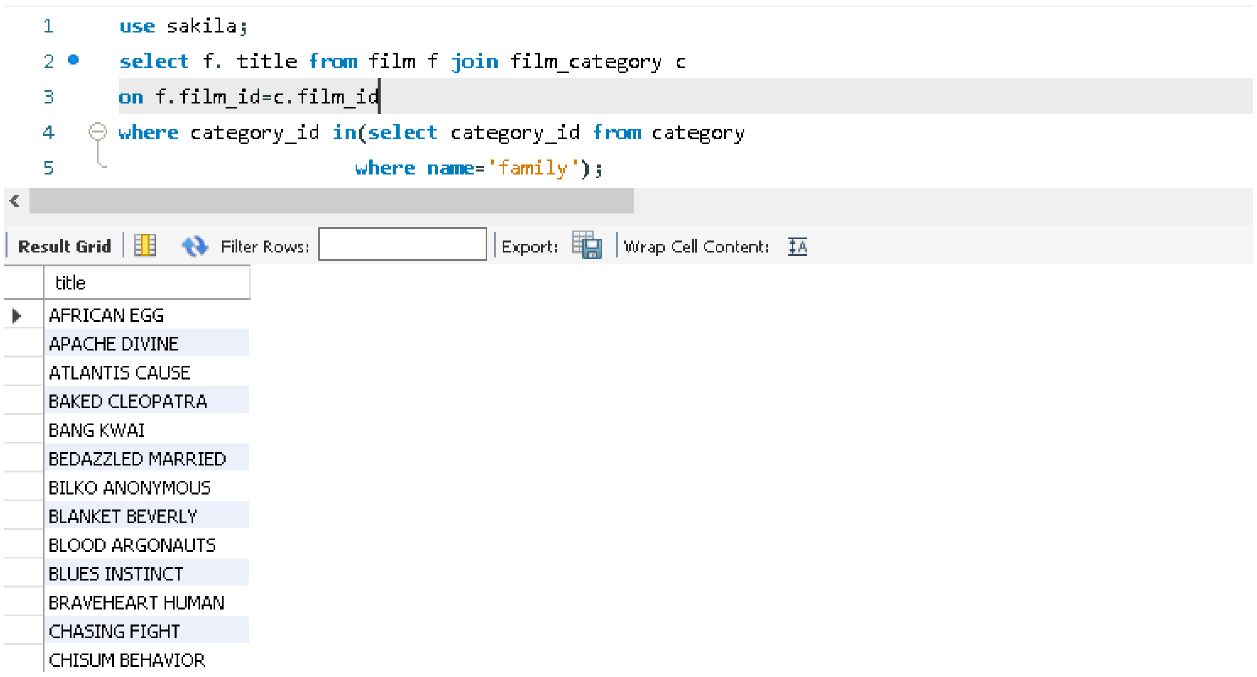


```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    c.email
FROM customer c JOIN address a ON c.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country co ON ci.country_id = co.country_id
WHERE co.country = 'Canada';

```

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

full_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.

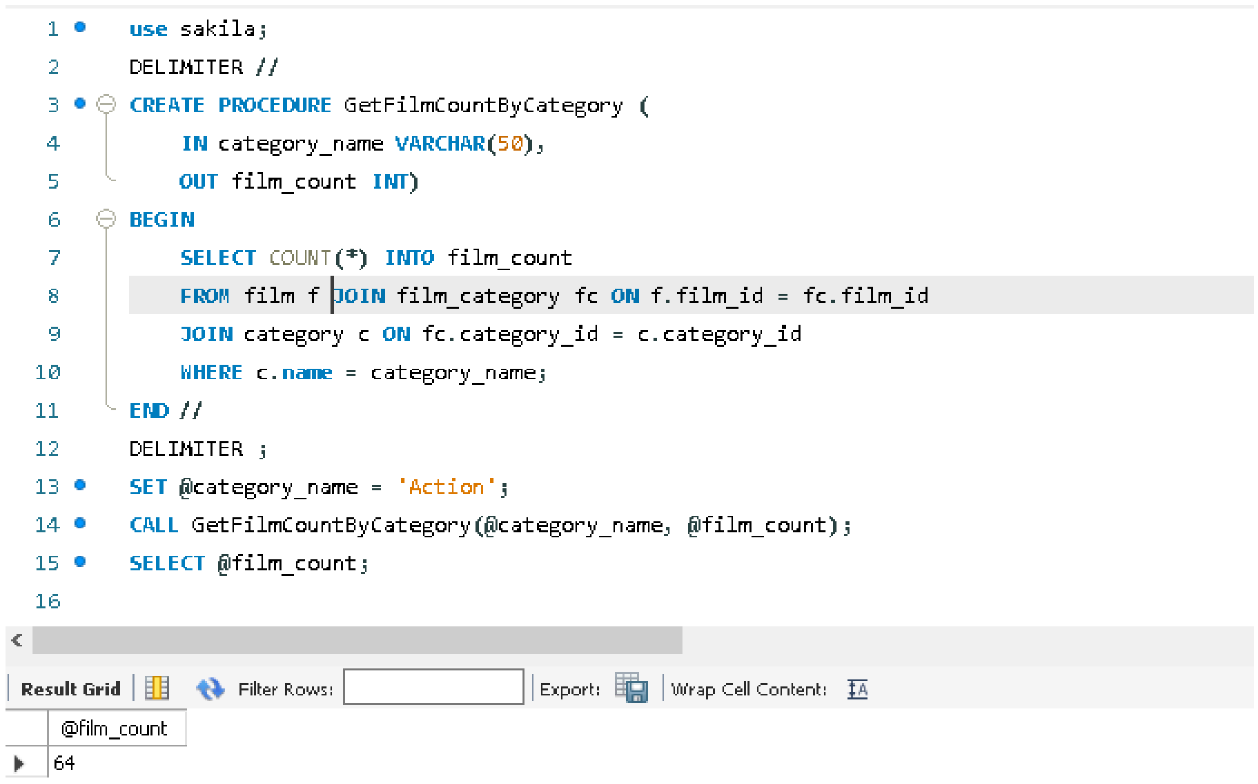


```
use sakila;
select f. title from film f join film_category c
on f.film_id=c.film_id
where category_id in(select category_id from category
                       where name='family');
```

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

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

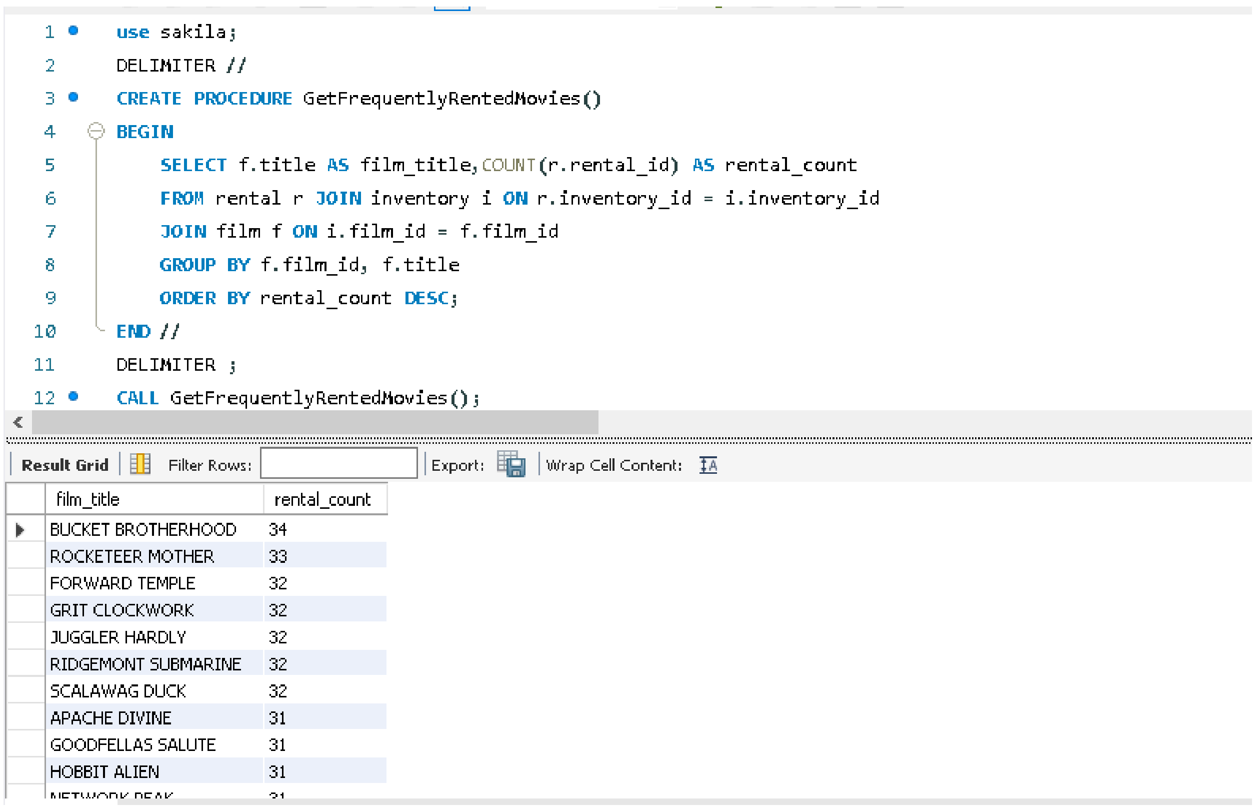
15. Create a Stored procedure to get the count of films in the input category (IN category_name, OUT count)



```
use sakila;
DELIMITER //
CREATE PROCEDURE GetFilmCountByCategory (
    IN category_name VARCHAR(50),
    OUT film_count INT)
BEGIN
    SELECT COUNT(*) 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 ;
SET @category_name = 'Action';
CALL GetFilmCountByCategory(@category_name, @film_count);
SELECT @film_count;
```

```
@film_count
64
```

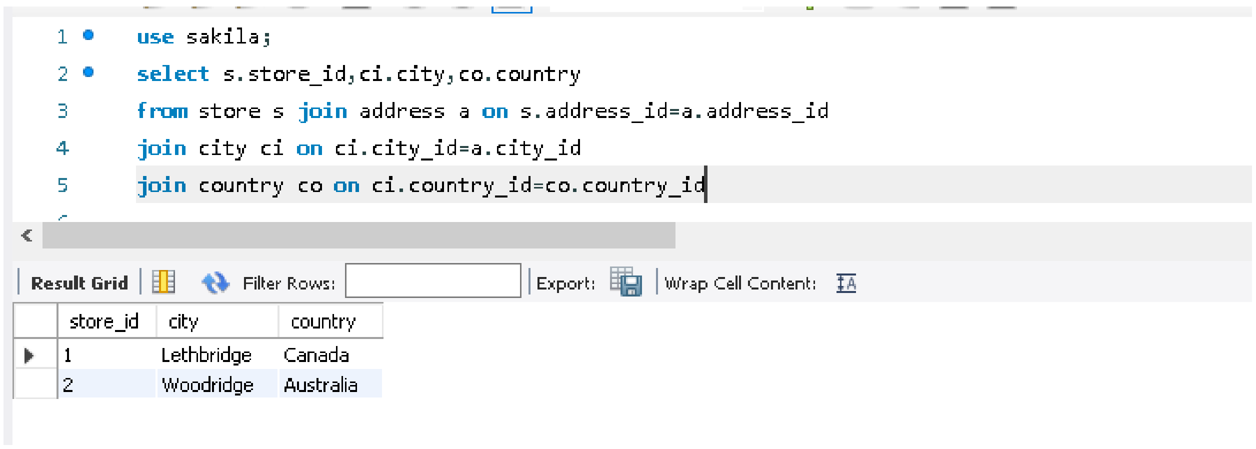
16. Display the most frequently rented movies in descending order.



```
use sakila;
DELIMITER //
CREATE PROCEDURE GetFrequentlyRentedMovies()
BEGIN
    SELECT f.title AS film_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.film_id, f.title
    ORDER BY rental_count DESC;
END //
DELIMITER ;
CALL GetFrequentlyRentedMovies();
```

film_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

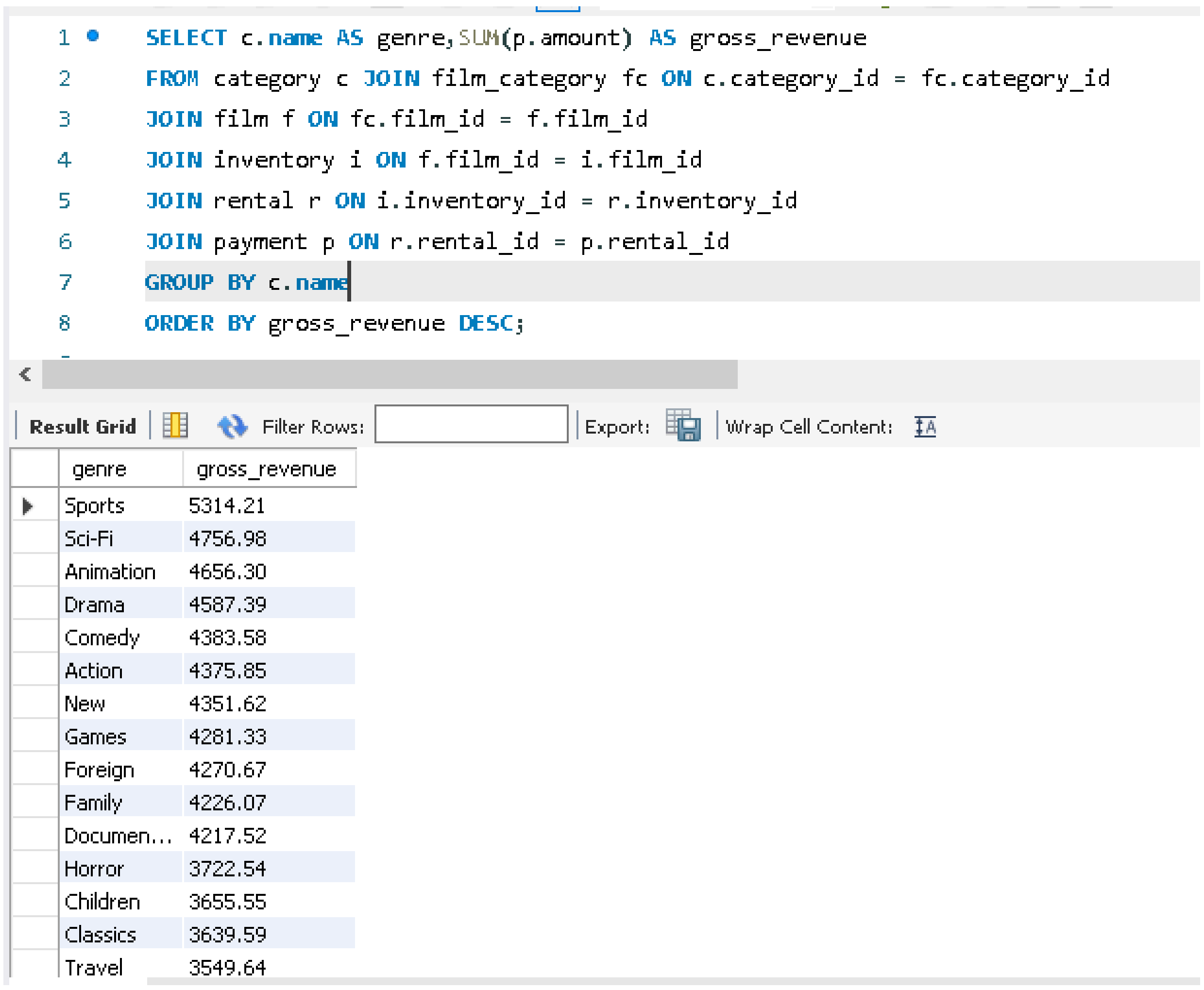
17. Write a query to display for each store its store ID, city, and country.



```
use sakila;
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 ci.city_id=a.city_id
join country co on ci.country_id=co.country_id
```

store_id	city	country
1	Lethbridge	Canada
2	Woodridge	Australia

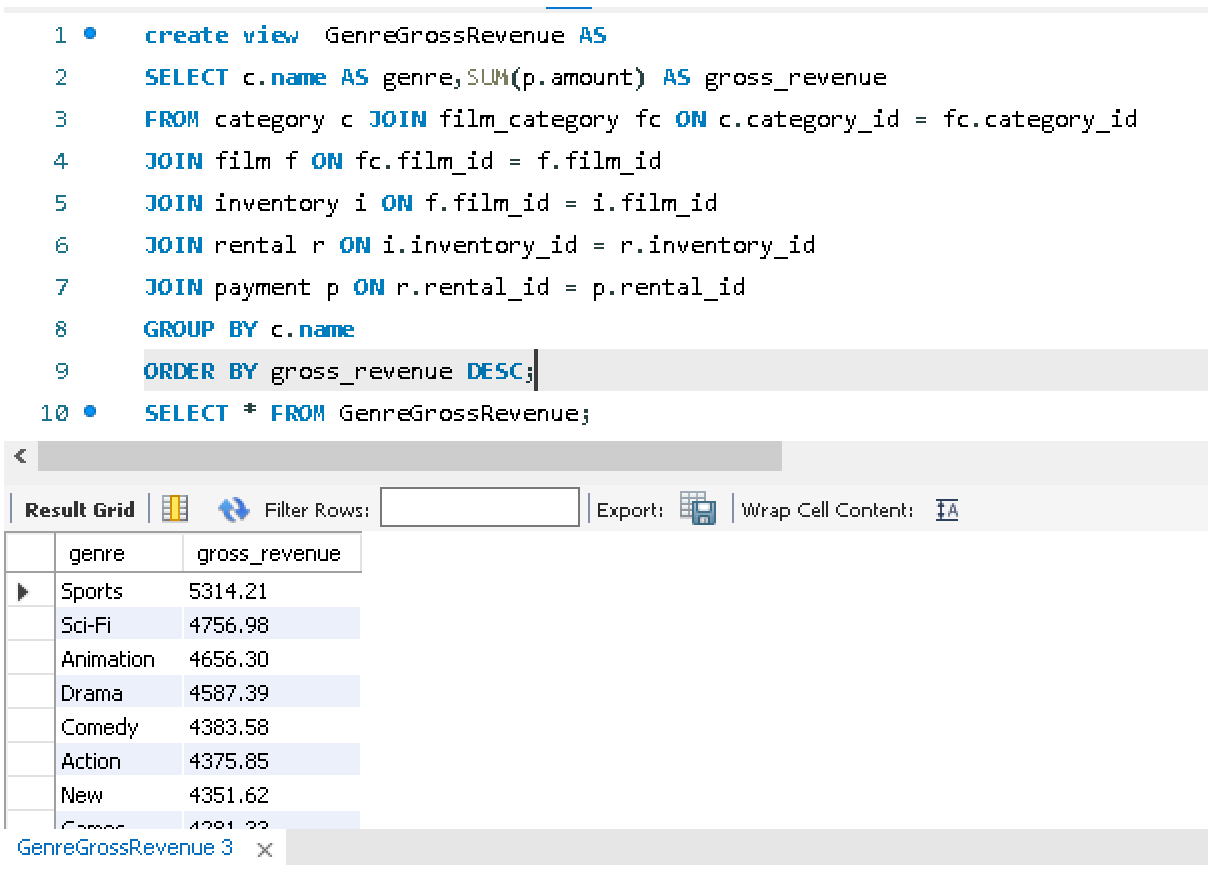
18. List the genres and its gross revenue.



```sql
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
Classics	3639.59
Travel	3549.64

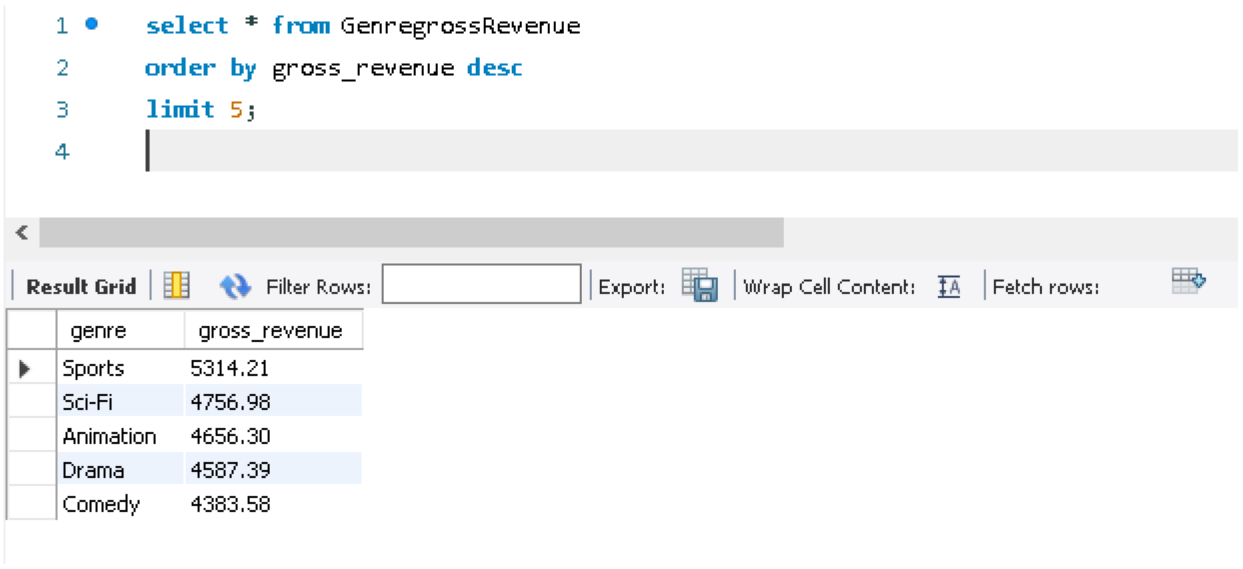
19. Create a View for the above query(18)



```
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;
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

20. Select top 5 genres in gross revenue view.



```
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