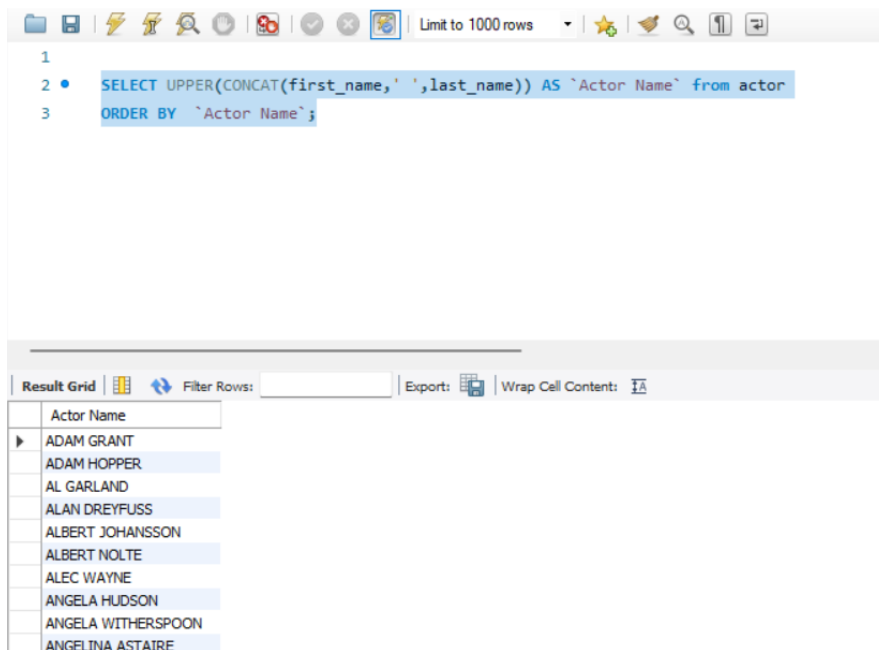


# Exercises

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`;
```



The screenshot shows a database client interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area:

```
1
2 • SELECT UPPER(CONCAT(first_name,' ',last_name)) AS `Actor Name` from actor
3 ORDER BY `Actor Name`;
```

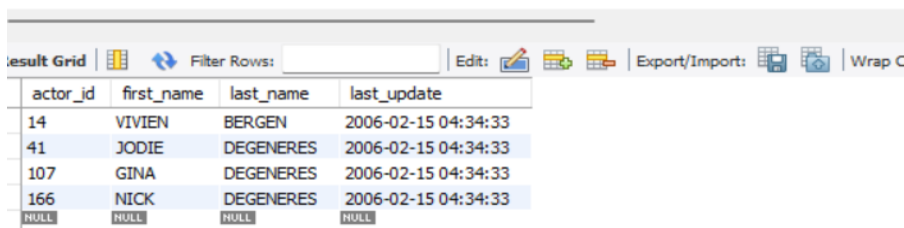
Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with one column, 'Actor Name', containing the following names in uppercase:

Actor Name
ADAM GRANT
ADAM HOPPER
AL GARLAND
ALAN DREYFUSS
ALBERT JOHANSSON
ALBERT NOLTE
ALEC WAYNE
ANGELA HUDSON
ANGELA WITHERSPOON
ANGELINA ASTAIRE

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

```
SELECT * from actor where last_name like '%GEN%';
```

```
4
5 • SELECT * from actor where last_name like '%GEN%';
```



The screenshot shows a database client interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area:

```
4
5 • SELECT * from actor where last_name like '%GEN%';
```

Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, and an 'Export/Import' button. The results are displayed in a table with four columns: 'actor\_id', 'first\_name', 'last\_name', and 'last\_update'.

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:

```
SELECT country_id,country from country where country IN('Afghanistan','Bangladesh','China');
```

```
2
3 • SELECT country_id,country from country where country IN('Afghanistan','Bangladesh','China');
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	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.

```
SELECT last_name,count(*) count from actor
```

```
GROUP BY last_name
```

```
ORDER BY count DESC;
```

```
2
3 • SELECT last_name,count(*) count from actor
4 GROUP BY last_name
5 ORDER BY count DESC;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
	last_name	count	
▶	KILMER	5	
	NOLTE	4	
	TEMPLE	4	
	AKROYD	3	
	ALLEN	3	
	BERRY	3	
	DAVIS	3	
	DEGENERES	3	
	GARLAND	3	
	GUINNESS	3	

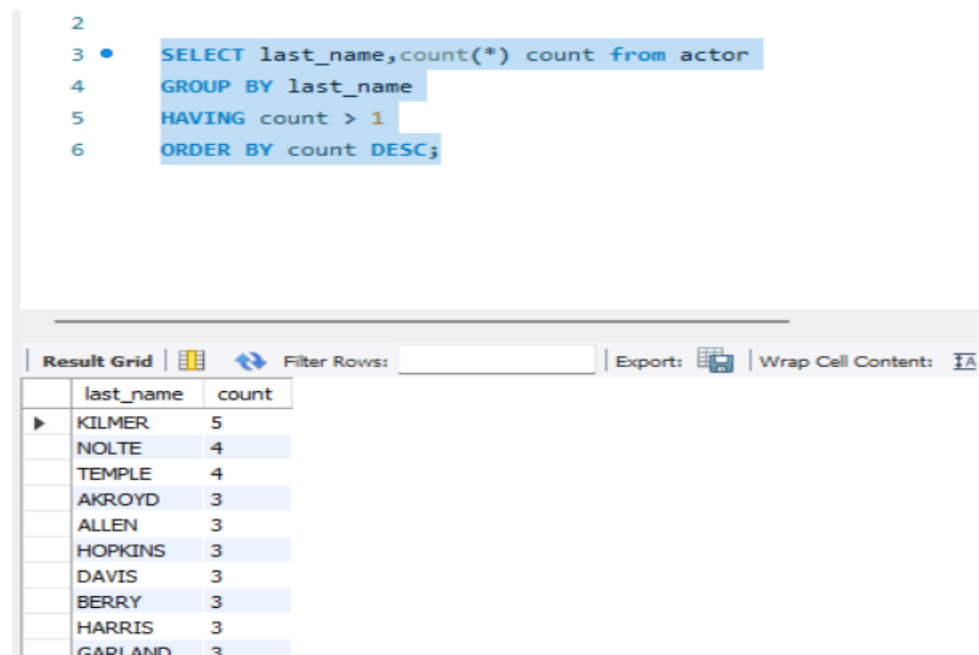
**5. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors**

```
SELECT last_name, count(*) count from actor
```

```
GROUP BY last_name
```

```
HAVING count > 1
```

```
ORDER BY count DESC;
```

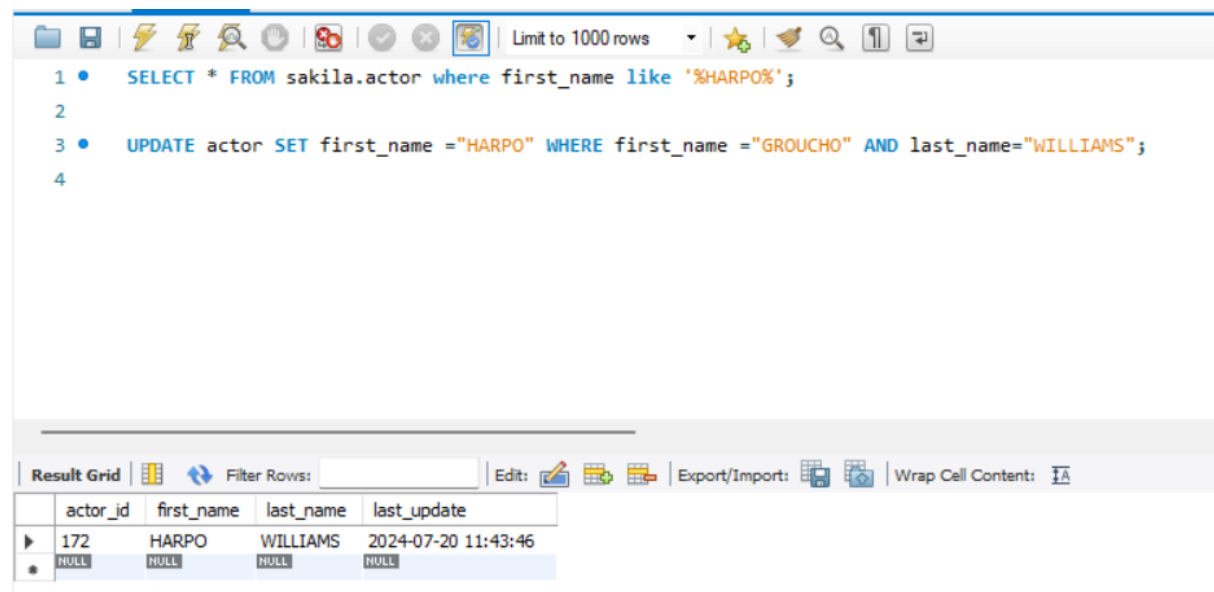


```
2
3 • SELECT last_name, count(*) count from actor
4   GROUP BY last_name
5   HAVING count > 1
6   ORDER BY count DESC;
```

	last_name	count
▶	KILMER	5
	NOLTE	4
	TEMPLE	4
	AKROYD	3
	ALLEN	3
	HOPKINS	3
	DAVIS	3
	BERRY	3
	HARRIS	3
	GARLAND	3

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

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



```
1 • SELECT * FROM sakila.actor where first_name like '%HARPO%';
2
3 • UPDATE actor SET first_name = "HARPO" WHERE first_name = "GROUCHO" AND last_name = "WILLIAMS";
4
```

	actor_id	first_name	last_name	last_update
▶	172	HARPO	WILLIAMS	2024-07-20 11:43:46
*	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:

```
SELECT first_name, last_name, address from staff JOIN address USING(address_id);
```

2

3 • `SELECT first_name, last_name, address from staff JOIN address USING(address_id);`

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
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.film_id, f.title,COUNT(fa.actor_id) actor_count FROM film f INNER JOIN film_actor  
fa USING(film_id)
```

```
GROUP BY f.film_id
```

```
ORDER BY actor_count DESC
```

actor

film\_actor

film

Limit to 1000 rows

1

•

SELECT \* FROM sakila.film;

2

•

SELECT f.film\_id, f.title,COUNT(fa.actor\_id) actor\_count FROM film f INNER JOIN film\_actor fa USING(film\_id)

3

GROUP BY f.film\_id

4

ORDER BY actor\_count DESC

Result Grid

Filter Rows:

Export:

Wrap Cell Content: W

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

Result Grid

Form Editor

Field Types

Query Stats

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

```
SELECT f.film_id,f.title,count(i.film_id) `number of copies` FROM film f JOIN inventory i  
USING(film_id) where f.title = 'HUNCHBACK IMPOSSIBLE'
```

```
GROUP BY i.film_id;
```

The screenshot shows a database query editor with the following SQL query:

```
1 • SELECT f.film_id,f.title,count(i.film_id) `number of copies` FROM film f JOIN inventory i USING(film_id)
2   where f.title = 'HUNCHBACK IMPOSSIBLE'
3   GROUP BY i.film_id
4
5   ;
```

The result grid shows the following data:

film_id	title	number of copies
439	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**

```
SELECT c.customer_id,CONCAT(c.first_name,' ',c.last_name) AS Name ,SUM(p.amount)  
AS `Total Paid` FROM customer c
```

```
JOIN payment p USING(customer_id)
```

```
GROUP BY p.customer_id
```

```
ORDER BY c.last_name
```

The screenshot shows a database query editor with the following SQL query:

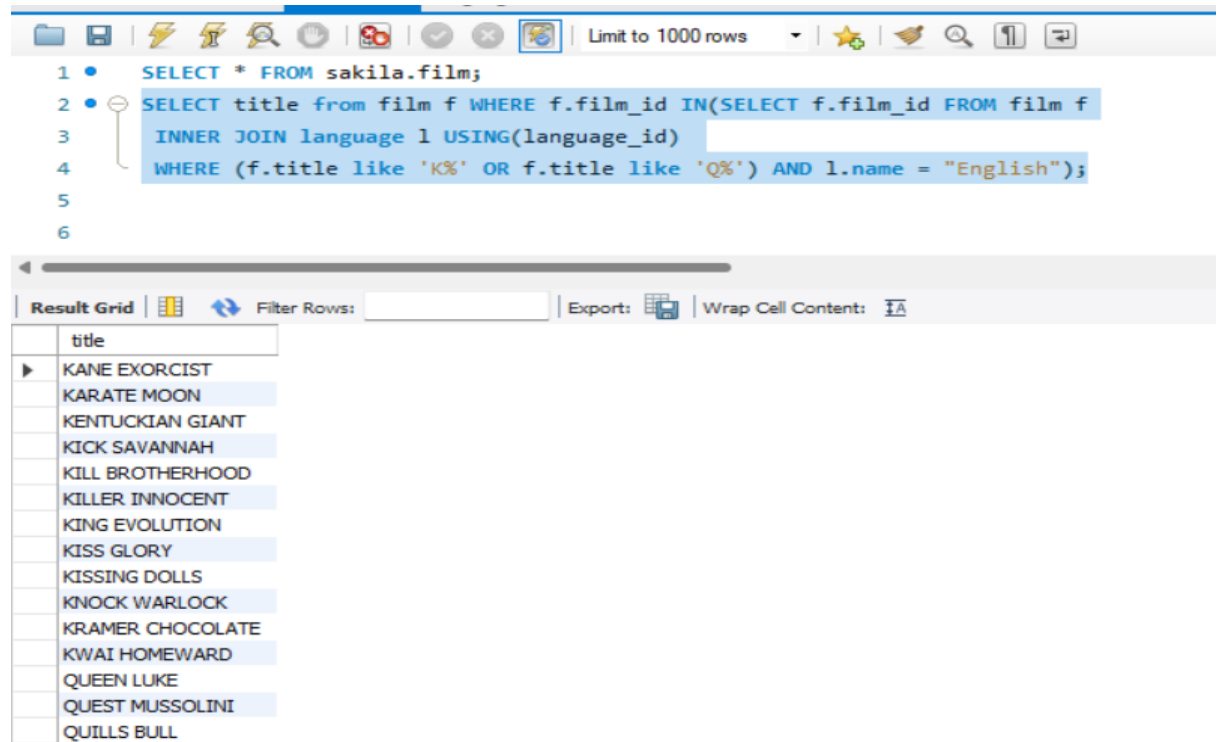
```
1 • SELECT * FROM sakila.customer;
2
3 • SELECT c.customer_id,CONCAT(c.first_name,' ',c.last_name) AS Name ,SUM(p.amount) AS `Total Paid` FROM customer c
4   JOIN payment p USING(customer_id)
5   GROUP BY p.customer_id
6   ORDER BY c.last_name
```

The result grid shows the following data:

customer_id	Name	Total Paid
505	RAFAEL ABNEY	97.79
504	NATHANIEL ADAM	133.72
36	KATHLEEN ADAMS	92.73
96	DIANA ALEXANDER	105.73
470	GORDON ALLARD	160.68
27	SHIRLEY ALLEN	126.69
220	CHARLENE ALVAREZ	114.73
11	LISA ANDERSON	106.76
326	JOSE ANDREW	96.75
183	IDA ANDREWS	76.77
449	OSCAR AQUINO	99.80
368	HARRY ARCE	157.65
560	JORDAN ARCHULETA	132.70
188	MELANIE ARMSTRO...	92.75
170	BEATRICE ARMSTRONG	110.74

- 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.

```
SELECT title from film f WHERE f.film_id IN(SELECT f.film_id FROM film f
INNER JOIN language l USING(language_id) WHERE (f.title like 'K%' OR
f.title like 'Q%') AND l.name = "English");
```



The screenshot shows a database query tool interface. The top toolbar includes icons for file operations, search, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

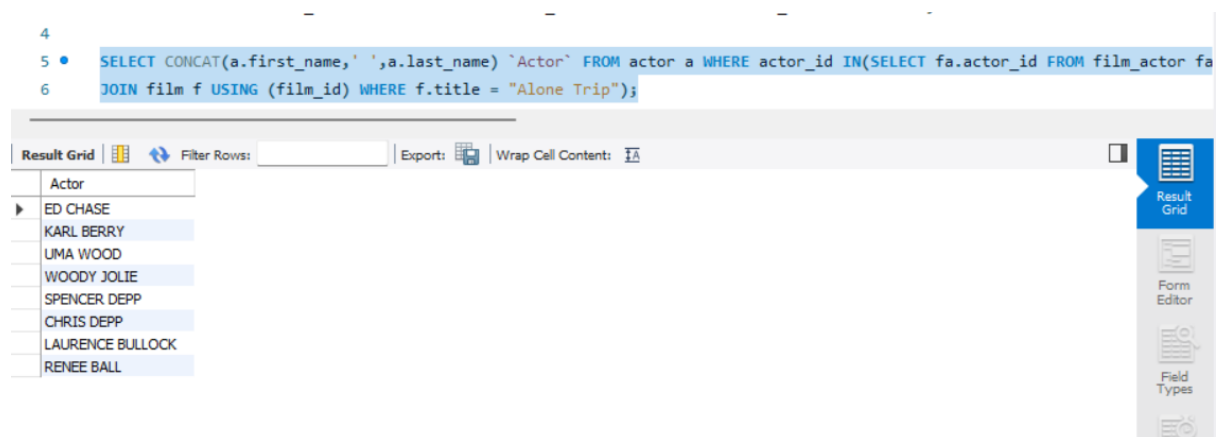
```
1 • SELECT * FROM sakila.film;
2 • SELECT title from film f WHERE f.film_id IN(SELECT f.film_id FROM film f
3   INNER JOIN language l USING(language_id)
4   WHERE (f.title like 'K%' OR f.title like 'Q%') AND l.name = "English");
5
6
```

Below the editor is the 'Result Grid' tab, which displays the results of the query. The results are as follows:

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

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

```
SELECT CONCAT(a.first_name,' ',a.last_name) `Actor` FROM actor a WHERE actor_id
IN(SELECT fa.actor_id FROM film_actor fa JOIN film f USING (film_id) WHERE f.title =
"Alone Trip");
```



The screenshot shows a database query tool interface. The SQL editor contains the following query:

```
4
5 • SELECT CONCAT(a.first_name,' ',a.last_name) `Actor` FROM actor a WHERE actor_id IN(SELECT fa.actor_id FROM film_actor fa
6   JOIN film f USING (film_id) WHERE f.title = "Alone Trip");
```

Below the editor is the 'Result Grid' tab, which displays the results of the query. The results are as follows:

Actor
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.

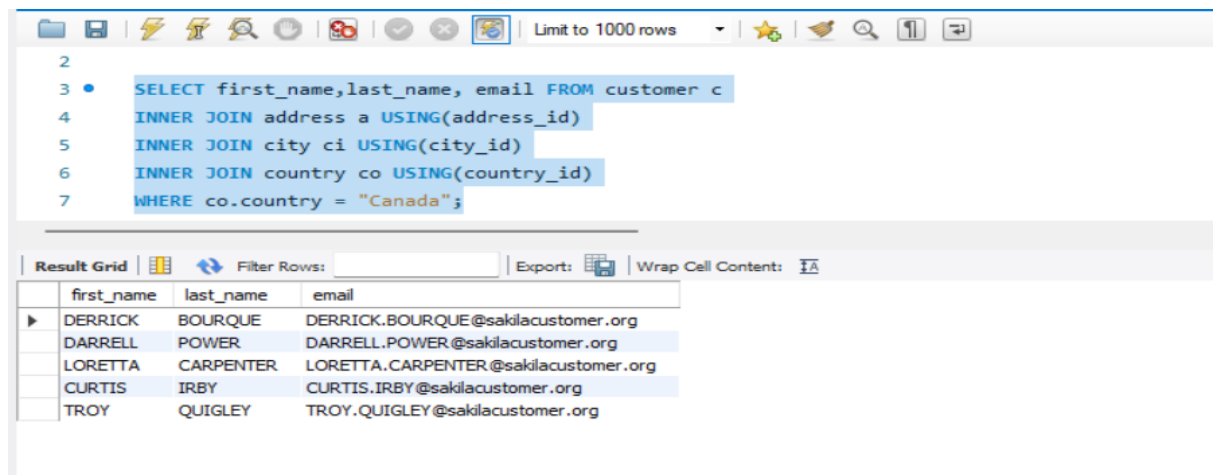
```
SELECT first_name,last_name, email FROM customer c
```

```
INNER JOIN address a USING(address_id)
```

```
INNER JOIN city ci USING(city_id)
```

```
INNER JOIN country co USING(country_id)
```

```
WHERE co.country = "Canada";
```



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
2
3 • SELECT first_name,last_name, email FROM customer c
4     INNER JOIN address a USING(address_id)
5     INNER JOIN city ci USING(city_id)
6     INNER JOIN country co USING(country_id)
7     WHERE co.country = "Canada";
```

Below the query, the 'Result Grid' is displayed with the following data:

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

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.

We can write query in two ways

```
SELECT title from film f
```

```
INNER JOIN film_category fc USING(film_id)
```

```
INNER JOIN category c USING(category_id)
```

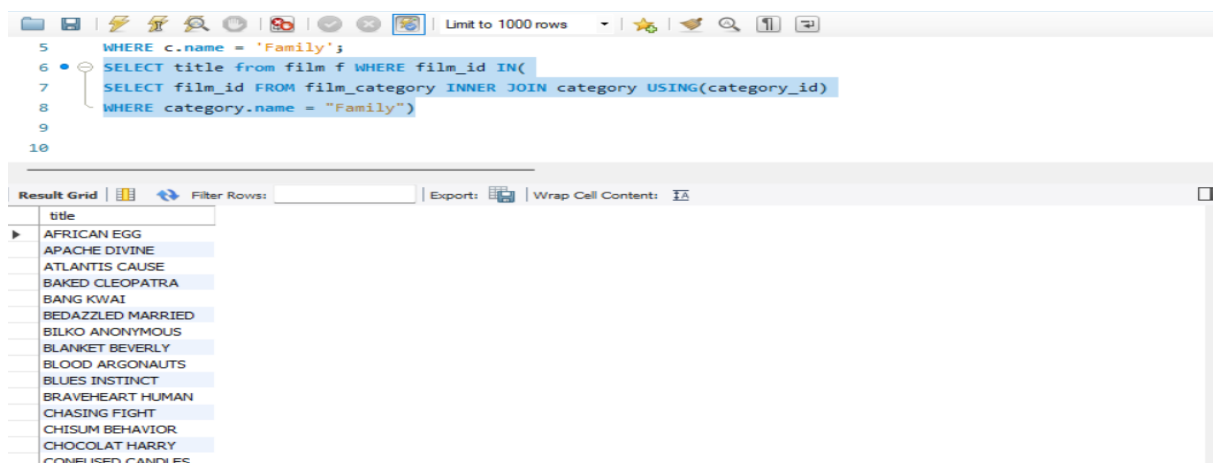
```
WHERE c.name = 'Family';
```

OR

```
SELECT title from film f WHERE film_id IN(
```

```
SELECT film_id FROM film_category INNER JOIN category USING(category_id)
```

```
WHERE category.name = "Family")
```



The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

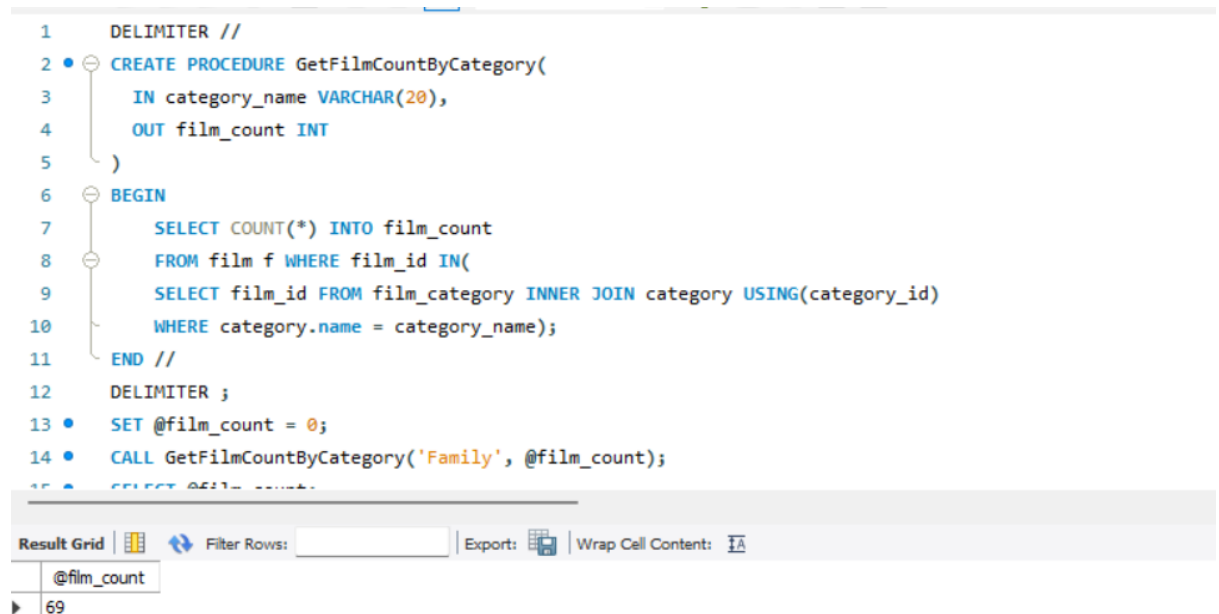
```
5 WHERE c.name = 'Family';
6 • SELECT title from film f WHERE film_id IN(
7     SELECT film_id FROM film_category INNER JOIN category USING(category_id)
8     WHERE category.name = "Family")
9
10
```

Below the query, the 'Result Grid' is displayed with the following data:

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. Create a Stored procedure to get the count of films in the input category (IN category\_name, OUT count)**

```
CREATE PROCEDURE GetFilmCountByCategory(  
    IN category_name VARCHAR(20),  
    OUT film_count INT  
)  
  
BEGIN  
    SELECT COUNT(*) INTO film_count  
    FROM film f WHERE film_id IN(  
        SELECT film_id FROM film_category INNER JOIN category USING(category_id)  
        WHERE category.name = category_name);  
  
END //  
  
DELIMITER ;  
  
SET @film_count = 0;  
  
CALL GetFilmCountByCategory('Family', @film_count);  
  
SELECT @film_count;
```



The screenshot shows a SQL IDE with a script editor and a result grid. The script editor contains the following SQL code:

```
1 DELIMITER //  
2 CREATE PROCEDURE GetFilmCountByCategory(  
3     IN category_name VARCHAR(20),  
4     OUT film_count INT  
5 )  
6 BEGIN  
7     SELECT COUNT(*) INTO film_count  
8     FROM film f WHERE film_id IN(  
9         SELECT film_id FROM film_category INNER JOIN category USING(category_id)  
10        WHERE category.name = category_name);  
11 END //  
12 DELIMITER ;  
13 SET @film_count = 0;  
14 CALL GetFilmCountByCategory('Family', @film_count);  
15 SELECT @film_count;
```

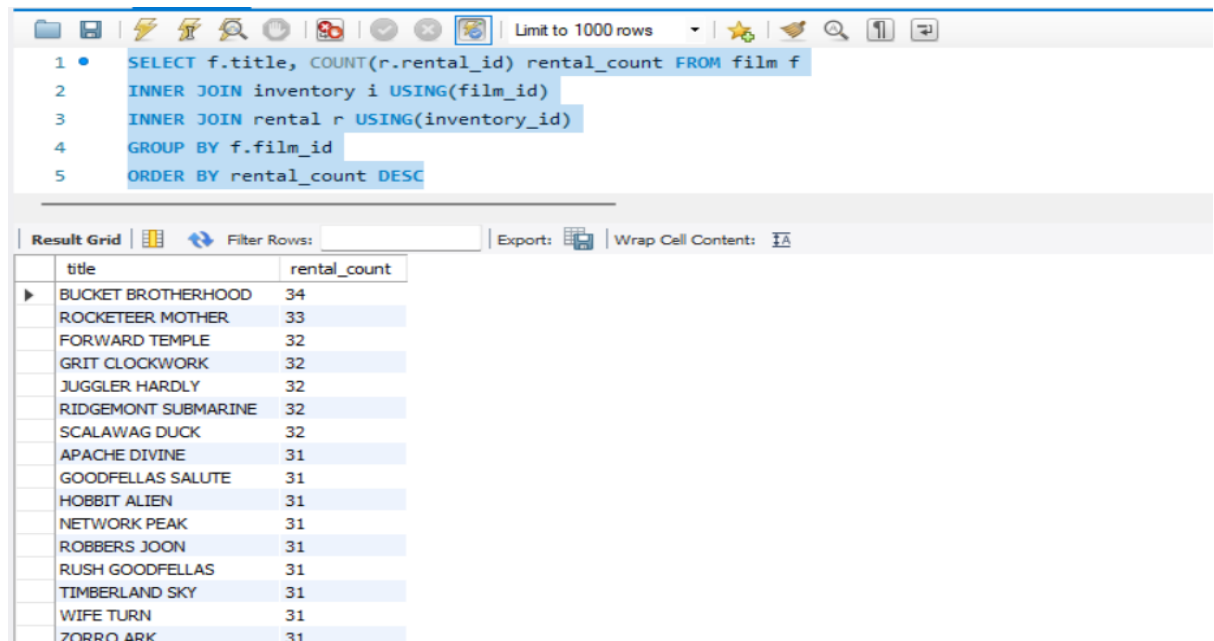
The result grid at the bottom shows the output of the final SELECT statement:

@film_count
69



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

```
SELECT f.title, COUNT(r.rental_id) rental_count FROM film f
INNER JOIN inventory i USING(film_id)
INNER JOIN rental r USING(inventory_id)
GROUP BY f.film_id
ORDER BY rental_count DESC
```



The screenshot shows a database query editor with the following SQL query:

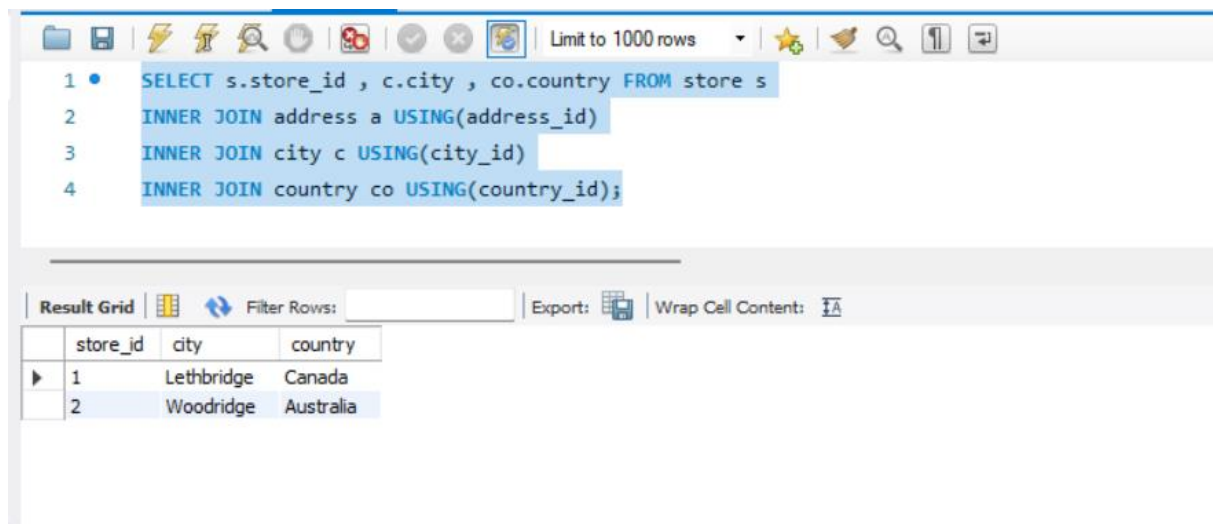
```
1 • SELECT f.title, COUNT(r.rental_id) rental_count FROM film f
2   INNER JOIN inventory i USING(film_id)
3   INNER JOIN rental r USING(inventory_id)
4   GROUP BY f.film_id
5   ORDER BY rental_count DESC
```

Below the query editor is the 'Result Grid' showing the results of the query. The grid has two columns: 'title' and 'rental\_count'.

	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

**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
INNER JOIN address a USING(address_id)
INNER JOIN city c USING(city_id)
INNER JOIN country co USING(country_id);
```



The screenshot shows a database query editor with the following SQL query:

```
1 • SELECT s.store_id , c.city , co.country FROM store s
2   INNER JOIN address a USING(address_id)
3   INNER JOIN city c USING(city_id)
4   INNER JOIN country co USING(country_id);
```

Below the query editor is the 'Result Grid' showing the results of the query. The grid has three columns: 'store\_id', 'city', and 'country'.

	store_id	city	country
▶	1	Lethbridge	Canada
	2	Woodridge	Australia

## 18. List the genres and its gross revenue.

```
SELECT c.name genres ,SUM(p.amount) gross_revenue FROM film f

INNER JOIN inventory i USING(film_id)

INNER JOIN rental r USING(inventory_id)

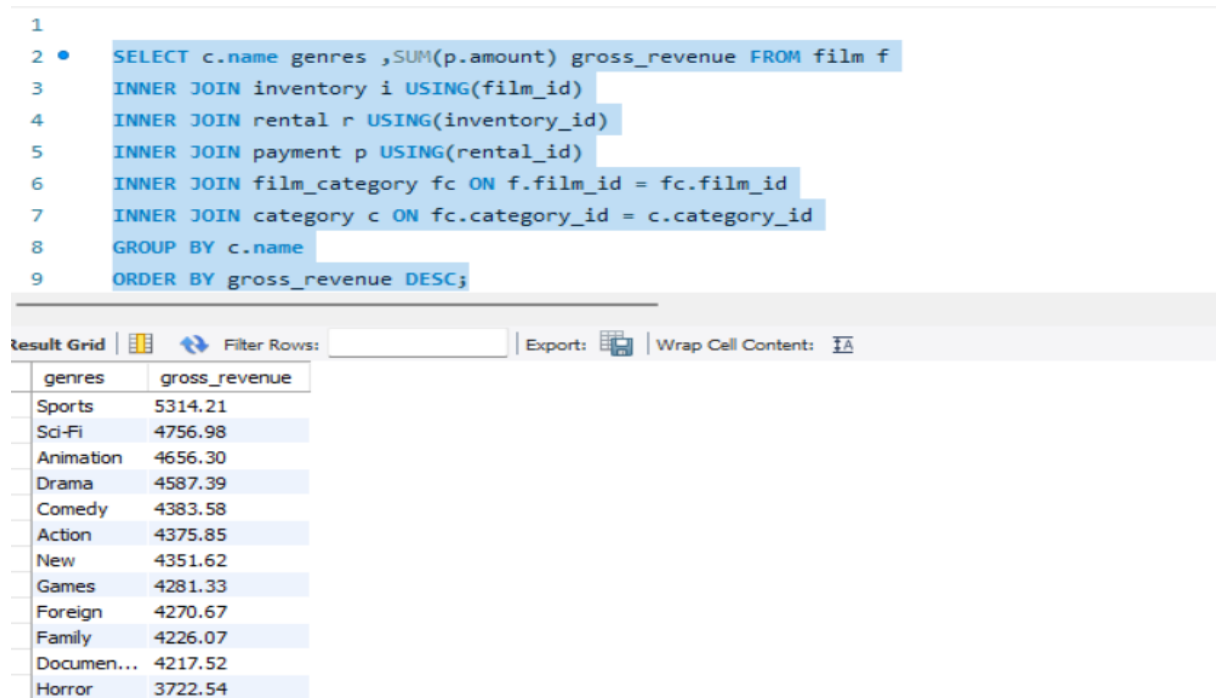
INNER JOIN payment p USING(rental_id)

INNER JOIN film_category fc ON f.film_id = fc.film_id

INNER JOIN category c ON fc.category_id = c.category_id

GROUP BY c.name

ORDER BY gross_revenue DESC;
```



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

```
1
2 • SELECT c.name genres ,SUM(p.amount) gross_revenue FROM film f
3     INNER JOIN inventory i USING(film_id)
4     INNER JOIN rental r USING(inventory_id)
5     INNER JOIN payment p USING(rental_id)
6     INNER JOIN film_category fc ON f.film_id = fc.film_id
7     INNER JOIN category c ON fc.category_id = c.category_id
8     GROUP BY c.name
9     ORDER BY gross_revenue DESC;
```

The results window displays the following table:

genres	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

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

```
CREATE VIEW genere_revenue_view AS

SELECT c.name genres ,SUM(p.amount) gross_revenue FROM film f

INNER JOIN inventory i USING(film_id)

INNER JOIN rental r USING(inventory_id)

INNER JOIN payment p USING(rental_id)

INNER JOIN film_category fc ON f.film_id = fc.film_id
```

INNER JOIN category c ON fc.category\_id = c.category\_id

GROUP BY c.name

ORDER BY gross\_revenue DESC;

SELECT \* FROM genre\_revenue\_view;

The screenshot shows a SQL IDE with a toolbar at the top. The SQL editor contains the following code:

```
1 • CREATE VIEW genre_revenue_view AS
2   SELECT c.name genres ,SUM(p.amount) gross_revenue FROM film f
3   INNER JOIN inventory i USING(film_id)
4   INNER JOIN rental r USING(inventory_id)
5   INNER JOIN payment p USING(rental_id)
6   INNER JOIN film_category fc ON f.film_id = fc.film_id
7   INNER JOIN category c ON fc.category_id = c.category_id
8   GROUP BY c.name
9   ORDER BY gross_revenue DESC;
10
11 • SELECT * FROM genre_revenue_view;
```

Below the editor, the 'Result Grid' tab is active, displaying the following data:

genres	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

## 20. Select top 5 genres in gross revenue view.

SELECT \* FROM genre\_revenue\_view LIMIT 5;

The screenshot shows the same SQL IDE with the following code in the editor:

```
1 • CREATE VIEW genre_revenue_view AS
2   SELECT c.name genres ,SUM(p.amount) gross_revenue FROM film f
3   INNER JOIN inventory i USING(film_id)
4   INNER JOIN rental r USING(inventory_id)
5   INNER JOIN payment p USING(rental_id)
6   INNER JOIN film_category fc ON f.film_id = fc.film_id
7   INNER JOIN category c ON fc.category_id = c.category_id
8   GROUP BY c.name
9   ORDER BY gross_revenue DESC;
10
11 • SELECT * FROM genre_revenue_view LIMIT 5;
```

The 'Result Grid' tab shows the top 5 results:

genres	gross_revenue
Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58

