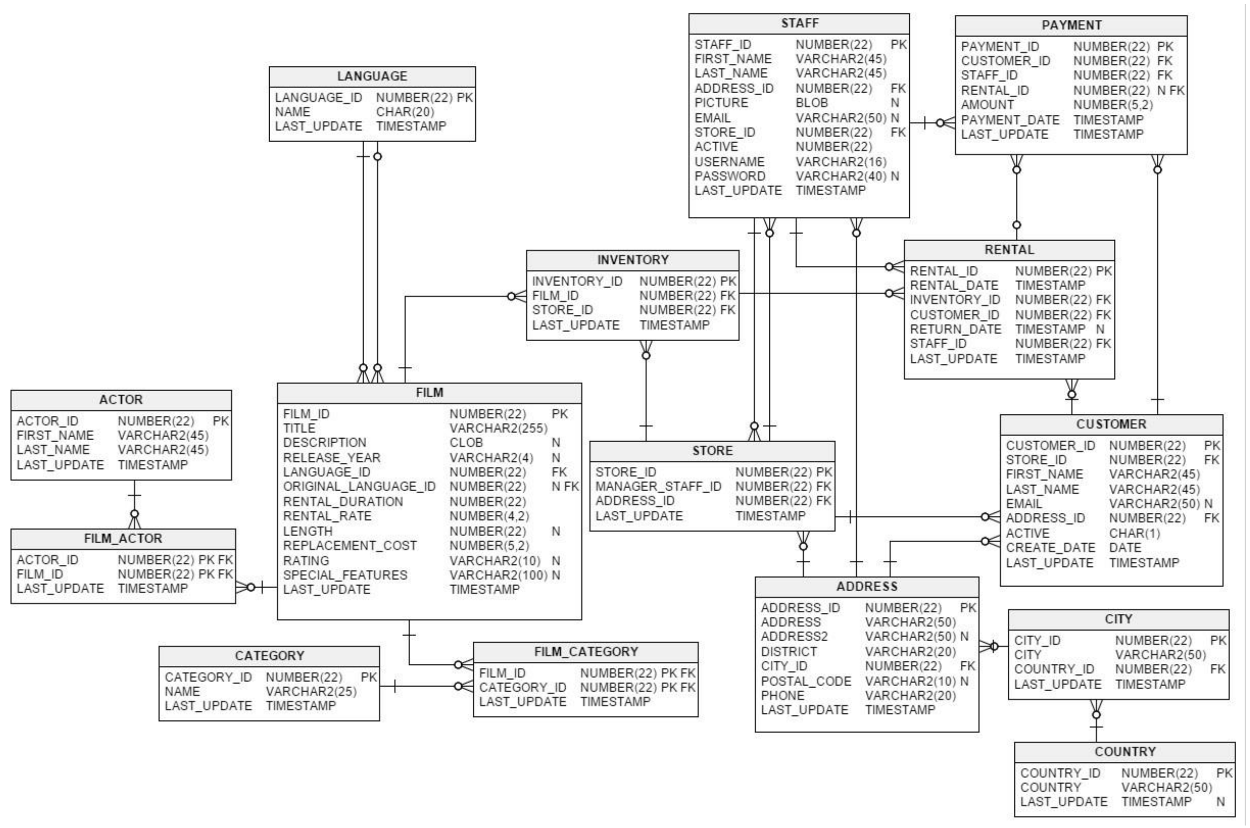
Introduction

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)
```

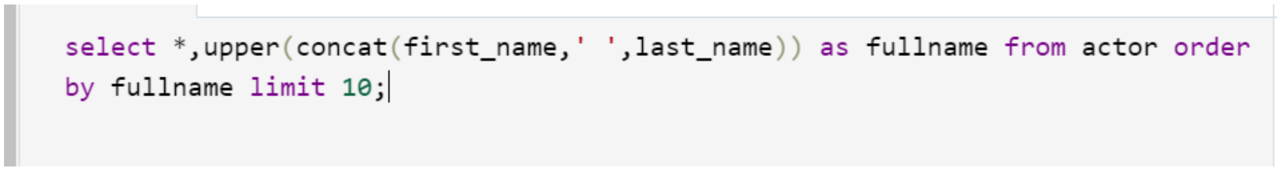
```
mysql> SELECT COUNT(*) FROM film;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

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`

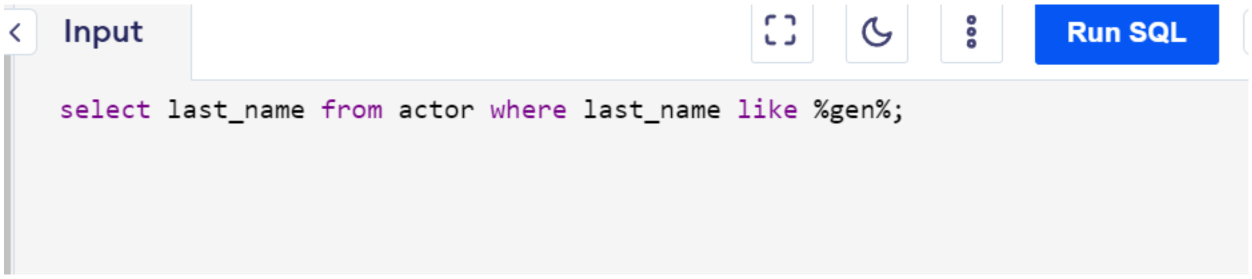
Exercises

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.



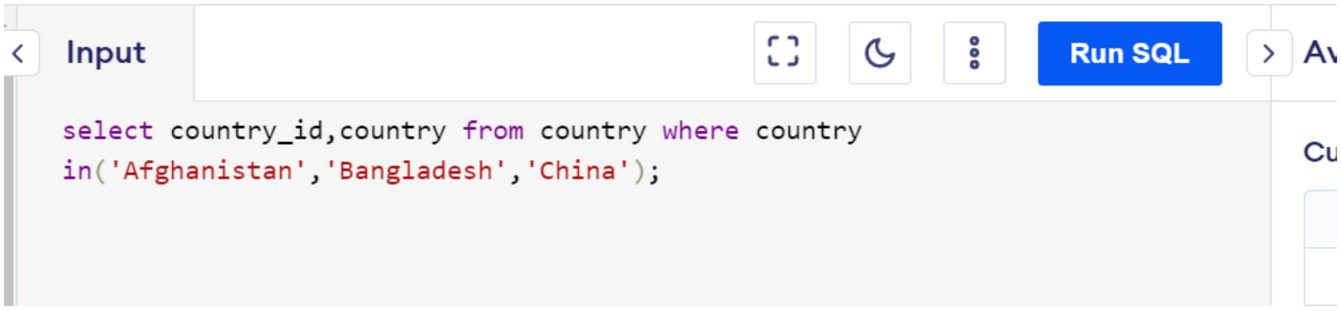
```sql
select *,upper(concat(first_name,' ',last_name)) as fullname from actor order
by fullname limit 10;
```

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



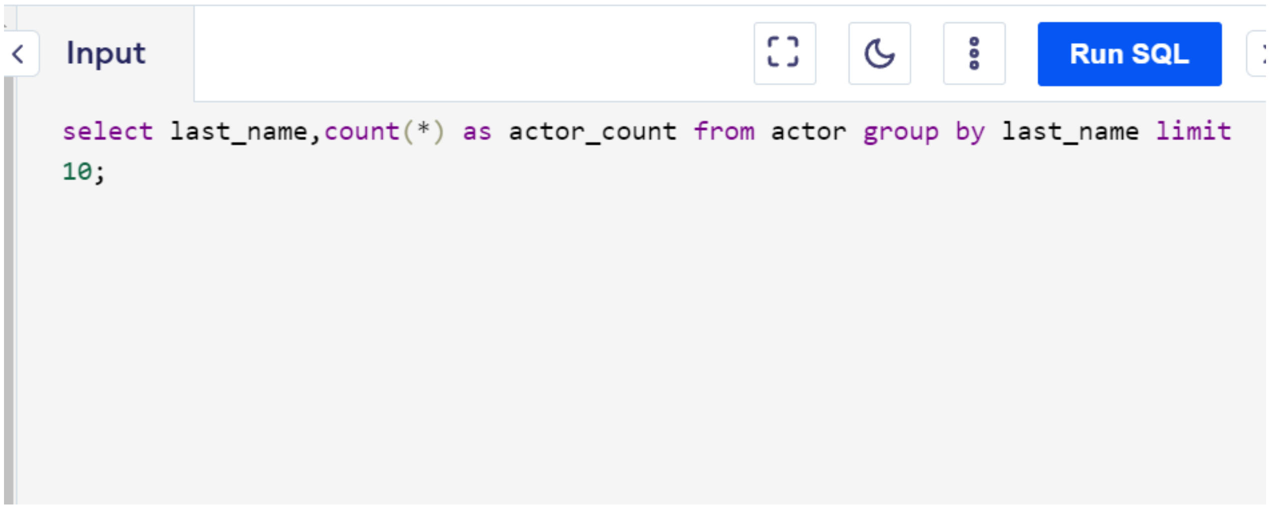
```sql
select last_name from actor where last_name like %gen%;
```

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



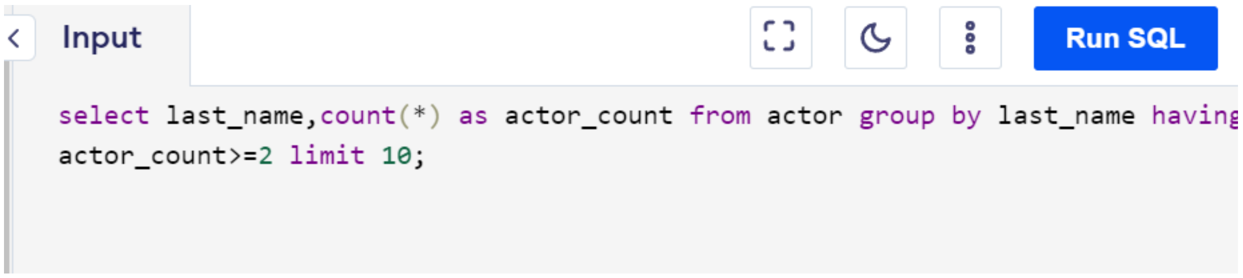
```sql
select country_id,country from country where country
in('Afghanistan','Bangladesh','China');
```

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



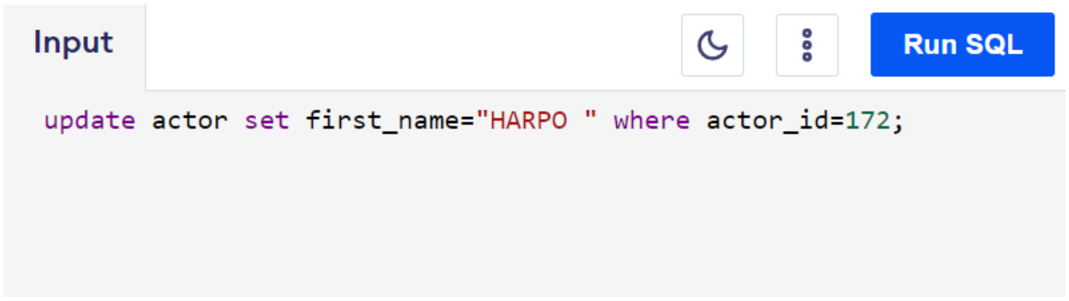
```
select last_name,count(*) as actor_count from actor group by last_name limit
10;
```

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



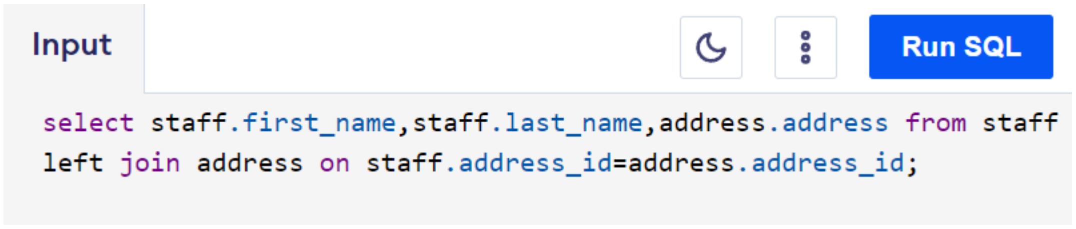
```
select last_name,count(*) as actor_count from actor group by last_name having
actor_count>=2 limit 10;
```

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



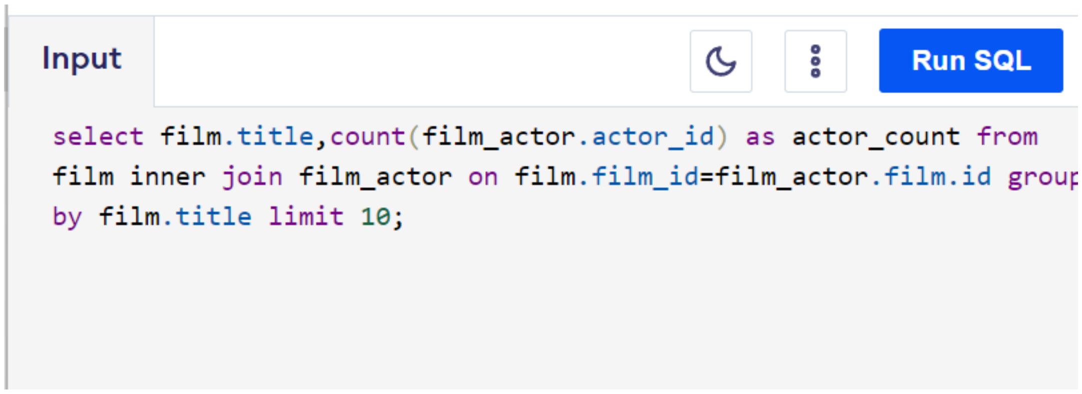
```
update actor set first_name="HARPO " where actor_id=172;
```

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:



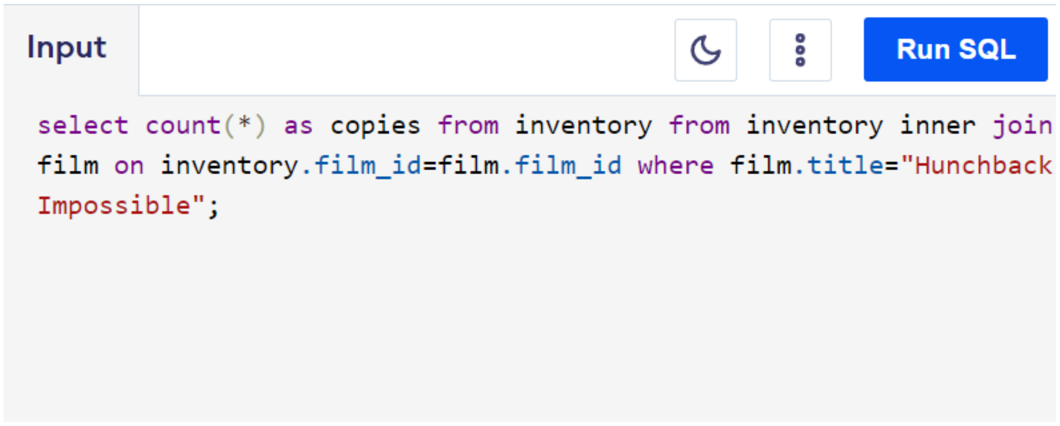
```
select staff.first_name,staff.last_name,address.address from staff
left join address on staff.address_id=address.address_id;
```

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 film.title,count(film_actor.actor_id) as actor_count from
film inner join film_actor on film.film_id=film_actor.film.id grouр
by film.title limit 10;
```

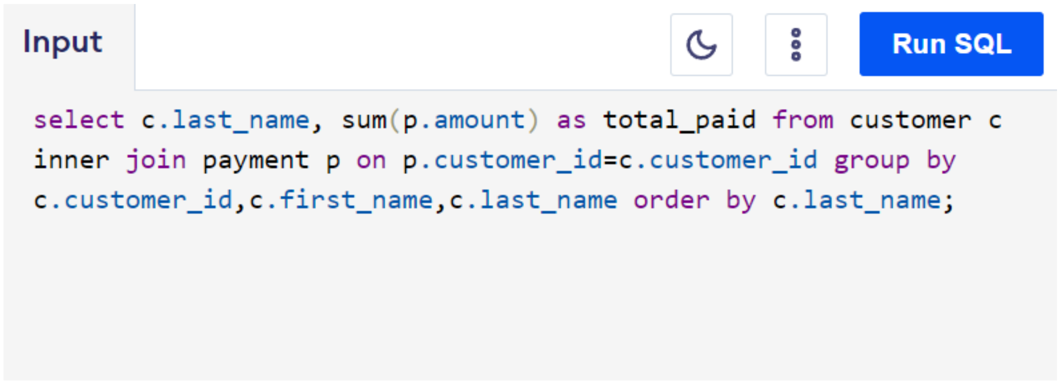
9. How many copies of the film Hunchback Impossible exist in the inventory system?



```
select count(*) as copies from inventory from inventory inner join
film on inventory.film_id=film.film_id where film.title="Hunchback
Impossible";
```

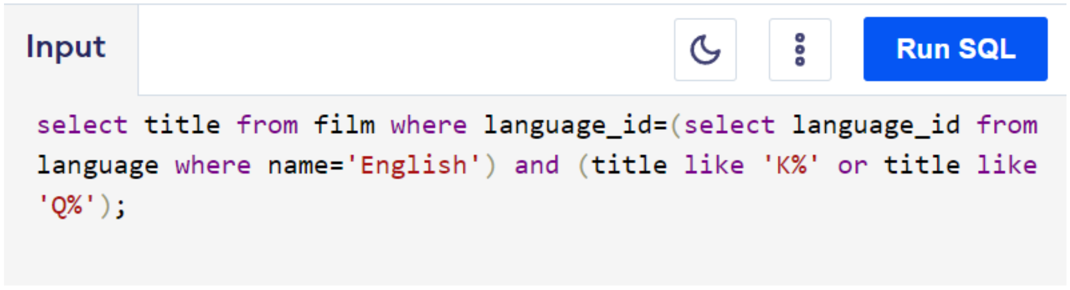
`

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



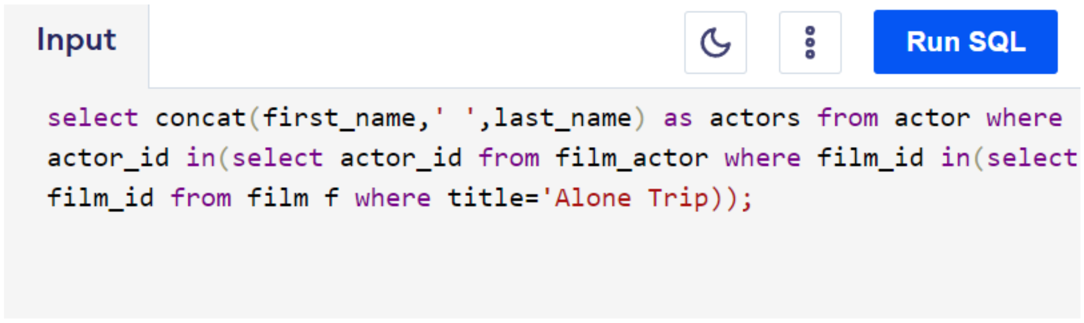
```
select c.last_name, sum(p.amount) as total_paid from customer c
inner join payment p on p.customer_id=c.customer_id group by
c.customer_id,c.first_name,c.last_name order by c.last_name;
```

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.



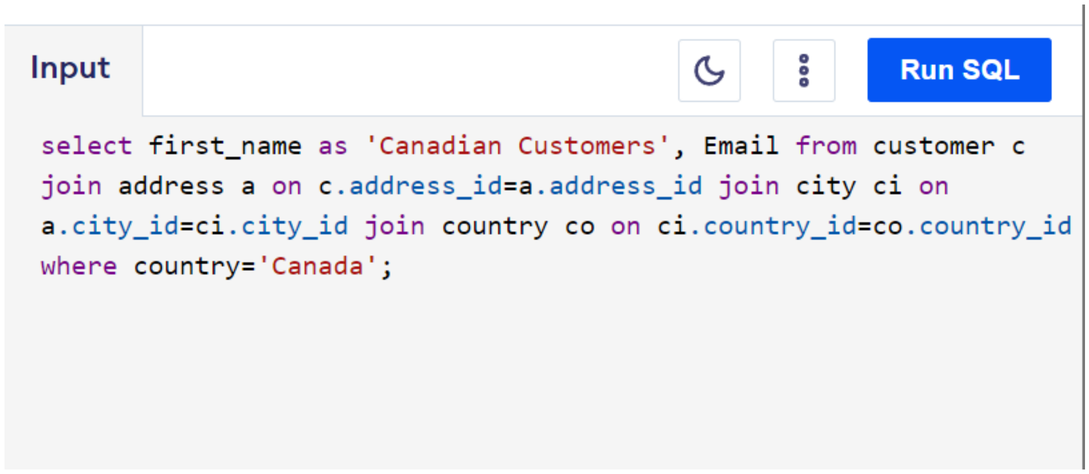
```
select title from film where language_id=(select language_id from
language where name='English') and (title like 'K%' or title like
'Q%');
```

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



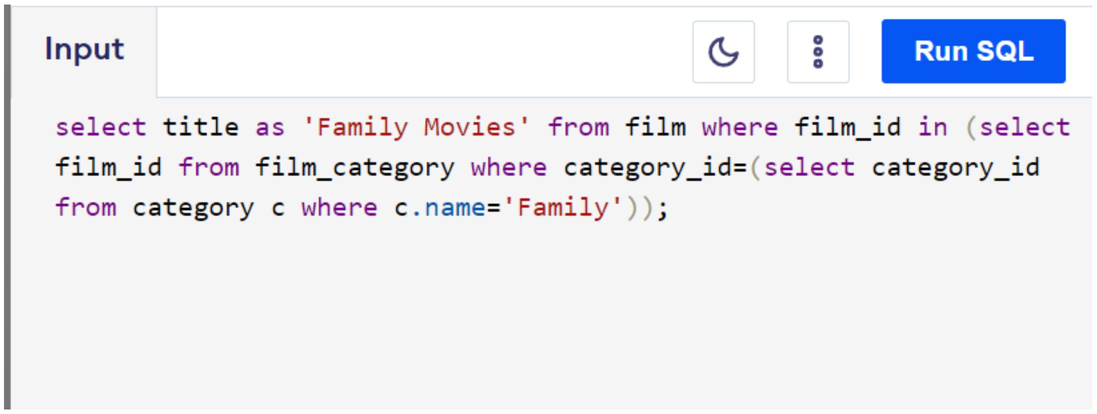
```
select concat(first_name,' ',last_name) as actors from actor where
actor_id in(select actor_id from film_actor where film_id in(select
film_id from film f where title='Alone Trip));
```

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.



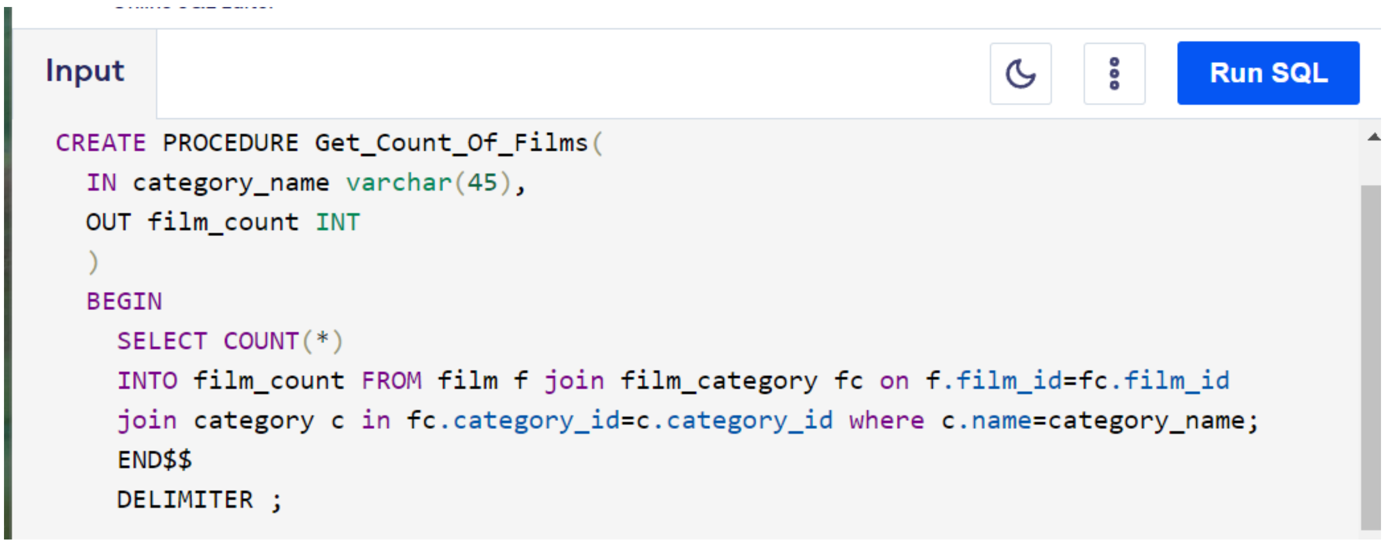
```
select first_name as 'Canadian Customers', Email from customer c
join address a on c.address_id=a.address_id join city ci on
a.city_id=ci.city_id join country co on ci.country_id=co.country_id
where country='Canada';
```

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.

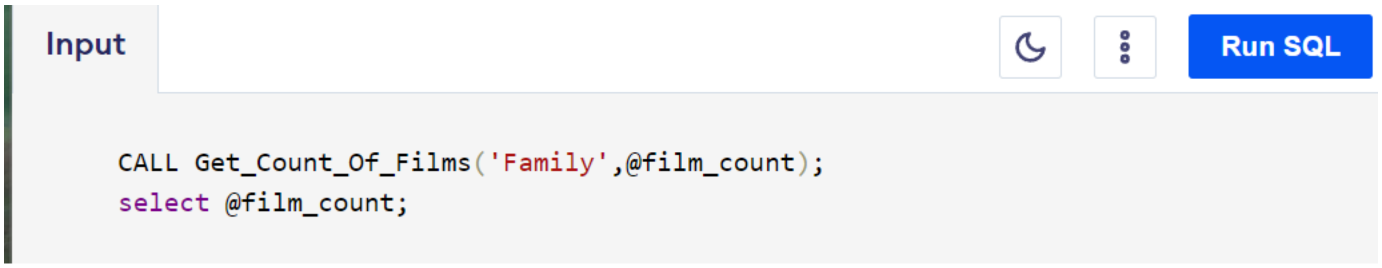


```
select title as 'Family Movies' from film where film_id in (select
film_id from film_category where category_id=(select category_id
from category c where c.name='Family'));
```

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

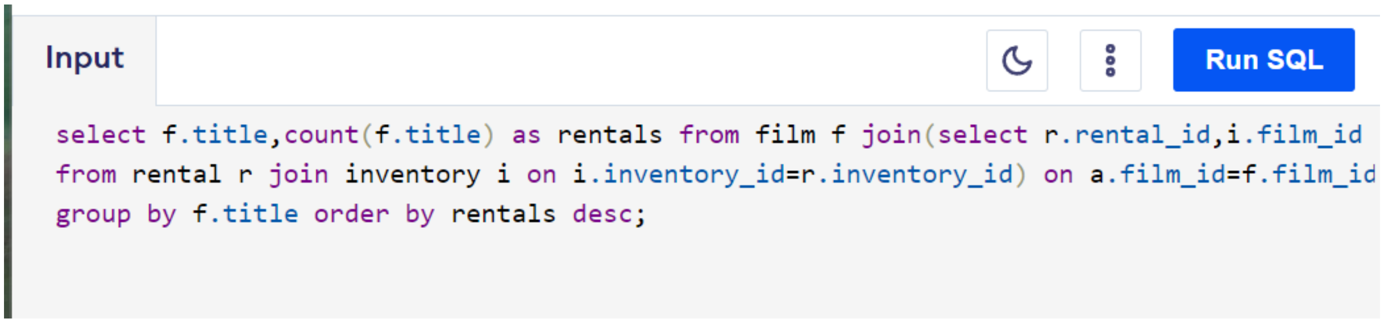


```
CREATE PROCEDURE Get_Count_Of_Films(
  IN category_name varchar(45),
  OUT film_count INT
  )
  BEGIN
    SELECT COUNT(*)
    INTO film_count FROM film f join film_category fc on f.film_id=fc.film_id
    join category c in fc.category_id=c.category_id where c.name=category_name;
    END$$
    DELIMITER ;
```



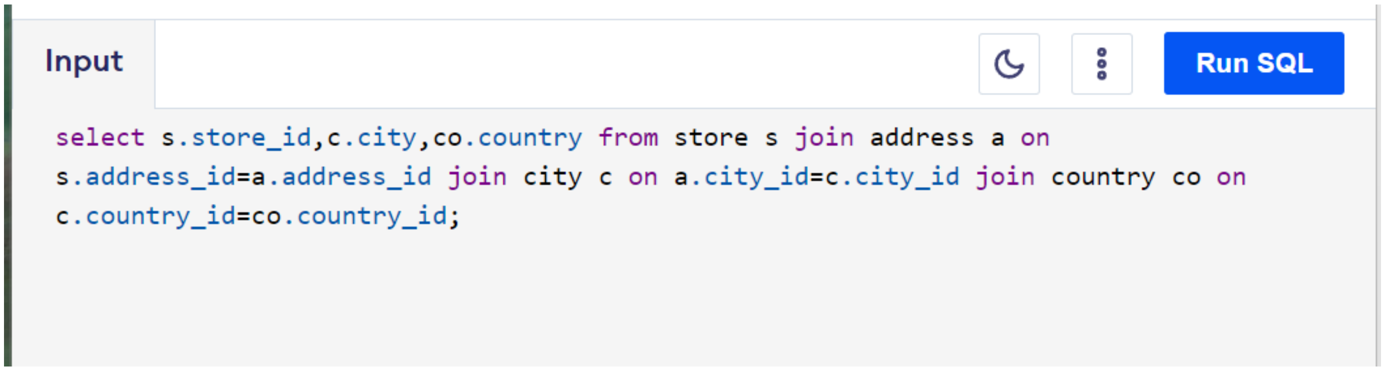
```
    CALL Get_Count_Of_Films('Family',@film_count);
    select @film_count;
```

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



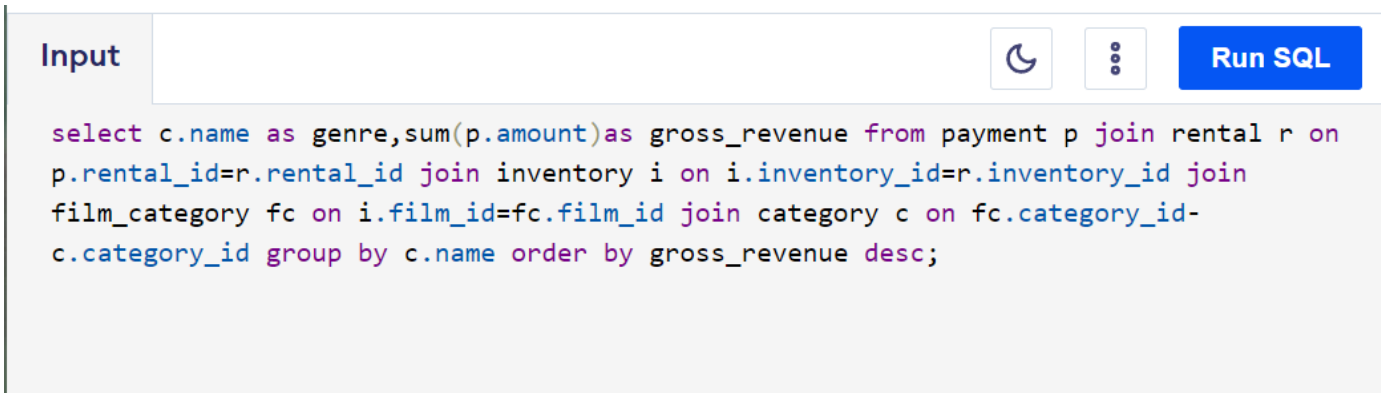
```sql
select f.title,count(f.title) as rentals from film f join(select r.rental_id,i.film_id
from rental r join inventory i on i.inventory_id=r.inventory_id) on a.film_id=f.film_id
group by f.title order by rentals desc;
```

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



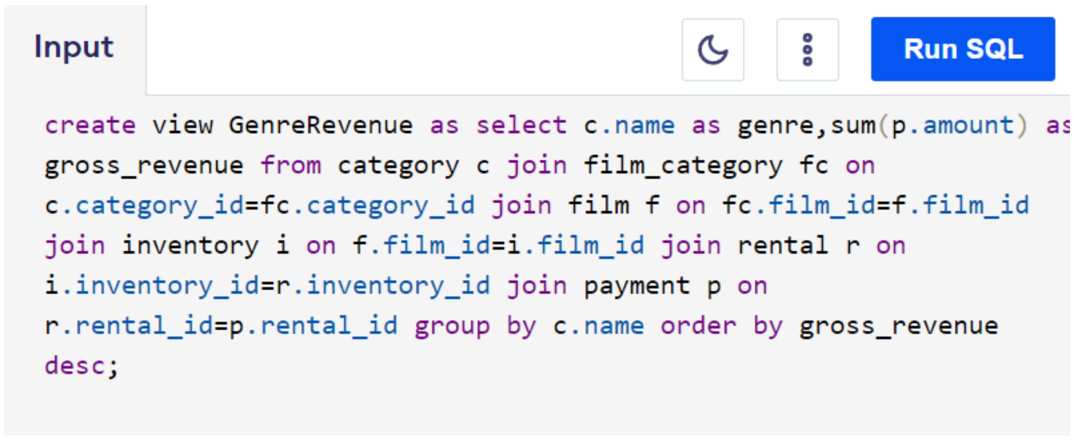
```sql
select s.store_id,c.city,co.country from store s join address a on
s.address_id=a.address_id join city c on a.city_id=c.city_id join country co on
c.country_id=co.country_id;
```

18. List the genres and its gross revenue.



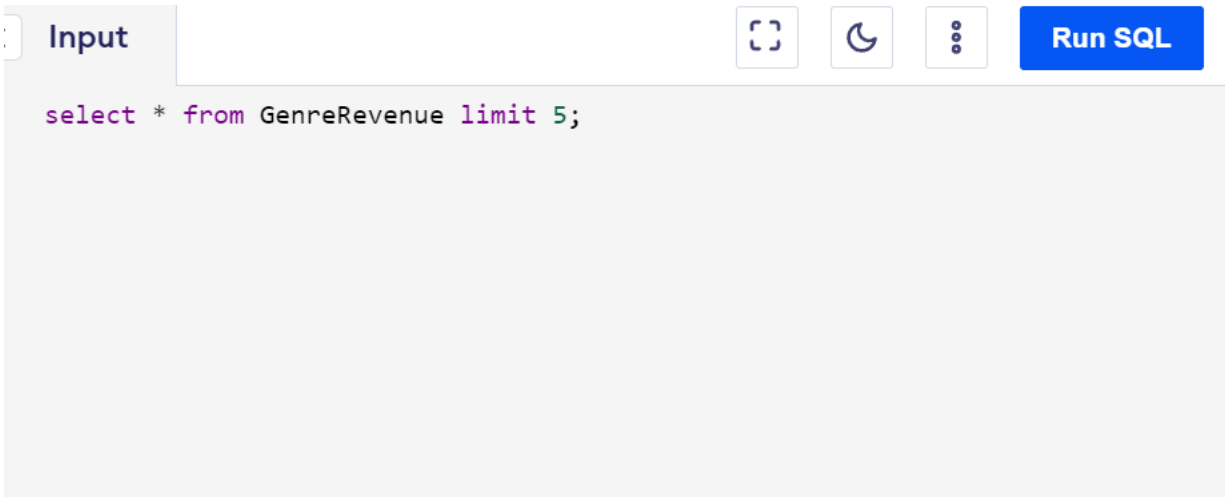
```sql
select c.name as genre,sum(p.amount)as gross_revenue from payment p join rental r on
p.rental_id=r.rental_id join inventory i on i.inventory_id=r.inventory_id join
film_category fc on i.film_id=fc.film_id join category c on fc.category_id-
c.category_id group by c.name order by gross_revenue desc;
```

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



```
create view GenreRevenue 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;
```

20. Select top 5 genres in gross revenue view.



```sql
select * from GenreRevenue limit 5;
```