Mysql Task Sakila

```
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 (0.02 sec)
```

```
mysql> select count(*) from film_text;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

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> use sakila;
Database changed
mysql> 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             |
| VIVIEN BERGEN        |
| CUBA OLIVIER         |
| FRED COSTNER         |
| HELEN VOIGHT         |
| DAN TORN             |
| BOB FAWCETT          |
| LUCILLE TRACY        |
| KIRSTEN PALTROW      |
| ELVIS MARX           |
| SANDRA KILMER        |
```

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 (0.00 sec)
```

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>
```

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 |
```


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 |
```

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 (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

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 (0.02 sec)
```

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 film.title;
```

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

```
mysql> SELECT COUNT(*)AS num_copies
    -> FROM inventory
    -> INNER JOIN film ON inventory.film_id = film.film_id
    -> WHERE film.title = 'Hunchback Impossible';
+------------+
| num_copies |
+------------+
|          6 |
+------------+
1 row in set (0.01 sec)
```

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
    ->     c.customer_id,
    ->     c.last_name,
    ->     c.first_name,
    ->     COALESCE(SUM(p.amount), 0) AS total_paid
    -> FROM
    ->     customer c
    -> LEFT JOIN
    ->     payment p ON c.customer_id = p.customer_id
    -> GROUP BY
    ->     c.customer_id, c.last_name, c.first_name
    -> ORDER BY
    ->     c.last_name, c.first_name;
+-------------+--------------+-------------+------------+
| customer_id | last_name    | first_name  | total_paid |
+-------------+--------------+-------------+------------+
|         505 | ABNEY        | RAFAEL      |      97.79 |
|         504 | ADAM         | NATHANIEL   |     133.72 |
|          36 | ADAMS        | KATHLEEN    |      92.73 |
|          96 | ALEXANDER    | DIANA       |     105.73 |
|         470 | ALLARD       | GORDON      |     160.68 |
|          27 | ALLEN        | SHIRLEY     |     126.69 |
|         220 | ALVAREZ      | CHARLENE    |     114.73 |
|          11 | ANDERSON     | LISA        |     106.76 |
|         326 | ANDREW       | JOSE        |      96.75 |
|         183 | ANDREWS      | IDA         |      76.77 |
|         449 | AQUINO       | OSCAR       |      99.80 |
|         368 | ARCE         | HARRY       |     157.65 |
```

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
    ->     (LEFT(title, 1) = 'K' OR LEFT(title, 1) = 'Q')
    ->     AND film_id IN (
    ->         SELECT film_id
    ->         FROM language
    ->         WHERE name = 'English'
    ->     );
+------------------+
| 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      |
+------------------+
```

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

```
mysql> SELECT actor_id, first_name, last_name
    -> FROM actor
    -> WHERE actor_id IN (
    ->     SELECT actor_id
    ->     FROM film_actor
    ->     WHERE film_id = (
    ->         SELECT film_id
    ->         FROM film
    ->         WHERE title = 'Alone Trip'
    ->     )
    -> );
+----------+------------+-----------+
| actor_id | first_name | last_name |
+----------+------------+-----------+
|        3 | ED         | CHASE     |
|       12 | KARL       | BERRY     |
|       13 | UMA        | WOOD      |
|       82 | WOODY      | JOLIE     |
|      100 | SPENCER    | DEPP      |
|      160 | CHRIS      | DEPP      |
|      167 | LAURENCE   | BULLOCK   |
|      187 | RENEE      | BALL      |
+----------+------------+-----------+
```

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
    ->     customer.first_name,
    ->     customer.last_name,
    ->     customer.email
    -> FROM
    ->     customer
    -> JOIN
    ->     address ON customer.address_id = address.address_id
    -> JOIN
    ->     city ON address.city_id = city.city_id
    -> JOIN
    ->     country ON city.country_id = country.country_id
    -> WHERE
    ->     country.country = 'Canada';
+------------+-----------+-------------------------------------+
| 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     |
+------------+-----------+-------------------------------------+
```

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.

```
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';
+--------------------------+
```

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

```
mysql> CREATE PROCEDURE GetFilmCountByCategory(
    ->   IN category_name VARCHAR(255),
    ->   OUT film_count INT
    -> )
    -> BEGIN
    ->   SELECT COUNT(*) INTO film_count
    ->   FROM films
    ->   WHERE category = category_name;
    -> END //
Query OK, 0 rows affected (2.00 sec)
```

Display the most frequently rented movies in descending order.

```
sql> 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;
```

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 |
+----------+------------+-----------+
```

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;
```

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;
Query OK, 0 rows affected (3.02 sec)

mysql>
mysql> SELECT * FROM GenreGrossRevenueView;
```

Select top 5 genres in gross revenue view

```
ysql> 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 |
```