

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.

```
5  /* Display the first and last name of each actor in a single column in upper case letters in alphabetic order.
6  Name the column Actor Name */
7  •  SELECT
8      CONCAT(UPPER(first_name), ' ', UPPER(last_name)) AS "Actor Name"
9  FROM
10     actor
11 ORDER BY
12     CONCAT(first_name, ' ', last_name);
13
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Actor Name			
▶	ADAM GRANT			
	ADAM HOPPER			
	AL GARLAND			
	ALAN DREYFUSS			
	ALBERT JOHANSSON			
	ALBERT NOLTE			
	ALEC WAYNE			
	ANGELA HUDSON			
	ANGELA WITHERSPOON			
	ANGELINA ASTAIRE			
	ANNE CRONYN			
	AUDREY BAILEY			
	AUDREY OLIVIER			
	BELA WALKEN			
	BEN HARRIS			
	BEN WILLIS			
	BETTE NICHOLSON			
	BOB FAWCETT			
	BURT REYNOLDS			

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

```
14 -- Find all actors whose last name contain the letters GEN:
15 • SELECT *
16 FROM actor
17 WHERE last_name LIKE "%GEN%";
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:				
Export/Import:				
Wrap Cell Content:				
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:


```
20 -- Using IN, display the country_id and country columns of the following countries: Afghanistan, Bangladesh, an
21 • SELECT country_id, country
22 FROM country
23 WHERE country IN ("Afghanistan", "Bangladesh", "China");
```


Result Grid		
Filter Rows: <input type="text"/>		
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

```
26 -- List the last names of actors, as well as how many actors have that last name
27 • SELECT last_name, COUNT(*) AS "Number of actors with the same last name"
28 FROM actor
29 GROUP BY last_name;
```

Result Grid

 Filter Rows:

Export: 

Wrap Cell Content: 

last_name	Number of actors with the same last name
AKROYD	3
ALLEN	3
ASTAIRE	1
BACALL	1
BAILEY	2
BALE	1
BALL	1
BARRYMORE	1
BASINGER	1
BENING	2
BERGEN	1
BERGMAN	1
BERRY	3
BIRCH	1
BLOOM	1
BOLGER	2
BRIDGES	1
BRODY	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

```
32 -- List last names of actors and the number of actors who have that last name, but only for names that are s
33 • SELECT last_name, COUNT(*) AS "Number of actors with the same last name"
34 FROM actor
35 GROUP BY last_name
36 HAVING COUNT(*) >= 2;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	last_name	Number of actors with the same last name
	AKROYD	3
	ALLEN	3
	BAILEY	2
	BENING	2
	BERRY	3
	BOLGER	2
	BRODY	2
	CAGE	2
	CHASE	2
	CRAWFORD	2
	CRONYN	2
	DAVIS	3
	DEAN	2
	DEE	2
	DEGENERES	3
	DENCH	2
	DEPP	2
	DIKAKIS	2

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

```

39  -- The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS. Write a query to fix
40  • UPDATE actor
41    SET first_name = "HARPO"
42    WHERE first_name = "GROUCHO" AND last_name = "WILLIAMS";
43
44  • SELECT * FROM actor WHERE first_name = 'HARPO';

```

actor_id	first_name	last_name	last_update
172	HARPO	WILLIAMS	2024-10-30 07:58:37
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

```

46  -- Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables
47  • SELECT s.first_name, s.last_name, a.address
48    FROM staff s
49    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.

```

51  -- List each film and the number of actors who are listed for that film. Use tables film_actor and film. Use in
52  • SELECT f.title AS "Film Title",
53         COUNT(*) AS "Number of actors listed"
54    FROM film f
55    INNER JOIN film_actor fa ON f.film_id = fa.film_id
56    GROUP BY f.title;

```

Film Title	Number of actors listed
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
ALICE FANTASIA	4

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

```
57
58 -- How many copies of the film Hunchback Impossible exist in the inventory system?
59
60 • SELECT COUNT(*) AS "Number of Hunchback Impossible copies in the inventory"
61 FROM inventory i
62 JOIN film f ON i.film_id = f.film_id
63 WHERE f.title="HUNCHBACK IMPOSSIBLE";
```

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

	Number of Hunchback Impossible copies in the inventory
▶	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

```
66 -- Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the cus
67
68 • SELECT
69     c.first_name,
70     c.last_name,
71     SUM(p.amount) AS "Total paid by each customer"
72 FROM customer c
73 JOIN payment p ON c.customer_id = p.customer_id
74 GROUP BY c.first_name, c.last_name ;
```

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

	first_name	last_name	Total paid by each customer
▶	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

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

```
77 -- The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence,
78
79 • SELECT
80     f.title
81 FROM film f
82 WHERE
83     (f.title LIKE "K%" OR f.title LIKE "Q%") AND f.language_id =
84     ( SELECT l.language_id FROM language l 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
QUEEN LUKE

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



```
89 -- Use subqueries to display all actors who appear in the film Alone Trip.
90 • SELECT
91     CONCAT(first_name, ' ', last_name) AS `Actors in the film Alone Trip`
92 FROM actor
93 WHERE actor_id IN (SELECT actor_id FROM film_actor WHERE film_id =
94     (SELECT film_id FROM film WHERE title = 'ALONE TRIP'));
95
```

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

Actors in the film Alone Trip
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

```
--
96  -- You want to run an email marketing campaign in Canada, for which you will need the names and email addresses
97
98  • SELECT c.first_name, c.last_name ,c.email
99  FROM customer c
100  JOIN address a ON c.address_id = a.address_id
101  JOIN city ci ON a.city_id = ci.city_id
102  JOIN country co ON ci.country_id = co.country_id
103  WHERE co.country = 'Canada';
104
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
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.

```
105  -- Sales have been lagging among young families, and you wish to target all family movies for a promotion. Id
106  • SELECT f.title AS "Family Movies" FROM film f
107  JOIN film_category fc ON f.film_id = fc.film_id
108  JOIN category c ON fc.category_id = c.category_id
109  WHERE c.name = "Family";
110
```

Result Grid	
Filter Rows: <input type="text"/>	
Export:  Wrap Cell Content: 	
Family Movies	
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	
CONVERSATION DO...	

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

```
111 -- Create a Stored procedure to get the count of films in the input category (IN category_name, OUT count)
112 DELIMITER //
113
114 • CREATE PROCEDURE FilmCountByCategory(IN category_name VARCHAR(255), OUT film_count INT)
115 BEGIN
116     SELECT COUNT(*) INTO film_count
117     FROM film f
118     JOIN film_category fc ON f.film_id = fc.film_id
119     JOIN category c ON fc.category_id = c.category_id
120     WHERE c.name = category_name;
121 END //
122 DELIMITER ;
123
124 • CALL FilmCountByCategory('Classics', @film_count);
125 • SELECT @film_count;
126
```

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

@film_count

57

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

```
130
131 • SELECT f.title, COUNT(r.rental_id) AS `Rental Count`
132 FROM film f
133 JOIN inventory i ON f.film_id = i.film_id
134 JOIN rental r ON i.inventory_id = r.inventory_id
135 GROUP BY f.title
136 ORDER BY `Rental Count` DESC;
```

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

	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

Result 27

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

```
138  -- Write a query to display for each store its store ID, city, and country.
139  •  SELECT s.store_id AS `Store ID`,
140         ci.city AS `City`,
141         co.country AS `Country`
142  FROM store s
143  JOIN address a ON s.address_id = a.address_id
144  JOIN city ci ON a.city_id = ci.city_id
145  JOIN country co ON ci.country_id = co.country_id;
146
```

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

	Store ID	City	Country
▶	1	Lethbridge	Canada
	2	Woodridge	Australia

18. List the genres and its gross revenue.

```
145
146  -- List the genres and its gross revenue.
147  •  SELECT c.name AS `Genre`, SUM(p.amount) AS `Gross Revenue`
148  FROM category c
149  JOIN film_category fc ON c.category_id = fc.category_id
150  JOIN film f ON fc.film_id = f.film_id
151  JOIN inventory i ON f.film_id = i.film_id
152  JOIN rental r ON i.inventory_id = r.inventory_id
153  JOIN payment p ON r.rental_id = p.rental_id
154  GROUP BY c.name;
```

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

	Genre	Gross Revenue
▶	Action	4375.85
	Animation	4656.30
	Children	3655.55
	Classics	3639.59
	Comedy	4383.58
	Documentary	4217.52
	Drama	4587.39
	Family	4226.07
	Foreign	4270.67
	Games	4281.33
	Horror	3722.54

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

```
160 • CREATE VIEW genre_revenue AS
161 SELECT c.name AS `Genre`, SUM(p.amount) AS `Gross Revenue`
162 FROM category c
163 JOIN film_category fc ON c.category_id = fc.category_id
164 JOIN film f ON fc.film_id = f.film_id
165 JOIN inventory i ON f.film_id = i.film_id
166 JOIN rental r ON i.inventory_id = r.inventory_id
167 JOIN payment p ON r.rental_id = p.rental_id
168 GROUP BY c.name;
169
170 • SELECT * FROM genre_revenue;
```

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

	Genre	Gross Revenue
▶	Action	4375.85
	Animation	4656.30
	Children	3655.55
	Classics	3639.59
	Comedy	4383.58
	Documentary	4217.52
	Drama	4587.39
	Family	4226.07
	Foreign	4270.67

20. Select top 5 genres in gross revenue view

```
172 -- Select top 5 genres in gross revenue view
173
174 • SELECT * FROM genre_revenue
175 ORDER BY `Gross Revenue` DESC
176 LIMIT 5;
177
```

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

	Genre	Gross Revenue
▶	Sports	5314.21
	Sci-Fi	4756.98
	Animation	4656.30
	Drama	4587.39
	Comedy	4383.58