**Use the Sakila Database in MySQL workbench and frame the SQL queries for below scenarios.**

1. Write a query to display the first name, last name, and rental date of all customers who rented a film. Use the customer and rental tables.

**Ans:**

SELECT

c.first\_name,

c.last\_name,

r.rental\_date

FROM

customer AS c

JOIN

rental AS r ON c.customer\_id = r.customer\_id;

1. Write a query to display the first name, last name, and address of all customers whose district is either Alberta or Texas. Use the customer and address tables.

**Ans:**

SELECT c.first\_name, c.last\_name, a.address FROM customer AS c JOIN address AS a ON c.address\_id = a.address\_id WHERE a.district IN ('Alberta', 'Texas');

1. List all films along with their category names, including films that may not belong to any category. Then sort it by category names. Use the film, film\_category and category tables.

**Ans:**

SELECT f.title, c.name FROM film AS f LEFT JOIN film\_category AS fc ON f.film\_id = fc.film\_id LEFT JOIN category AS c ON fc.category\_id = c.category\_id ORDER BY c.name;

1. Display the number of rentals per store. Use the rental, inventory, and store tables.

**Ans:**

SELECT

s.store\_id,

COUNT(r.rental\_id) AS number\_of\_rentals

FROM

rental AS r

JOIN

inventory AS i ON r.inventory\_id = i.inventory\_id

JOIN

store AS s ON i.store\_id = s.store\_id

GROUP BY

s.store\_id;

1. List the names of categories that have more than 50 films. Use table category and film\_category.

**Ans:**

SELECT

c.name

FROM

category AS c

JOIN

film\_category AS fc ON c.category\_id = fc.category\_id

GROUP BY

c.name

HAVING

COUNT(fc.film\_id) > 50;

1. Write a query to display the film titles that have been rented by the customer named 'Mary Smith'. Use a subquery to get the customer ID.

**Ans:**

SELECT DISTINCT

f.title

FROM

film AS f

JOIN

inventory AS i ON f.film\_id = i.film\_id

JOIN

rental AS r ON i.inventory\_id = r.inventory\_id

WHERE

r.customer\_id = (SELECT customer\_id FROM customer WHERE first\_name = 'Mary' AND last\_name = 'Smith');

1. Write a query to display the total rental income generated by each film. Use film, inventory, rental and payment tables.

**Ans:**

SELECT

f.title,

SUM(p.amount) AS total\_income

FROM

film AS f

JOIN

inventory AS i ON f.film\_id = i.film\_id

JOIN

rental AS r ON i.inventory\_id = r.inventory\_id

JOIN

payment AS p ON r.rental\_id = p.rental\_id

GROUP BY

f.title

ORDER BY

total\_income DESC;

1. Write a query to display the total number of films each actor has acted in. Use actor, film\_Actor tables.

**Ans:**

SELECT

a.first\_name,

a.last\_name,

COUNT(fa.film\_id) AS number\_of\_films

FROM

actor AS a

JOIN

film\_actor AS fa ON a.actor\_id = fa.actor\_id

GROUP BY

a.actor\_id, a.first\_name, a.last\_name

ORDER BY

number\_of\_films DESC;

1. Write a query to list all rentals with a case statement that displays 'New Customer' if the customer\_id is greater than 300, and 'Regular Customer' otherwise. Use table rental.

**Ans:**

SELECT

rental\_id,

rental\_date,

customer\_id,

CASE

WHEN customer\_id > 300 THEN 'New Customer'

ELSE 'Regular Customer'

END AS customer\_type

FROM

rental;

1. Write a query to find all films with titles that start with the letter 'A'. Display the film title and the release year. Use film table and inventory table.

**Ans:**

SELECT

title,

release\_year

FROM

film

WHERE

title LIKE 'A%';

Q11. Create a new DB and perform below operations using SQL commands.

1. Create a table with 2 columns – Student\_ID, Marks
2. Insert 2 records in this table.
3. Rename the Marks column with Score.
4. Then delete the table.
5. Then delete the Database.

**Ans:**

-- Create a new database named 'StudentDB'

CREATE DATABASE StudentDB;

-- Switch to the newly created database to perform operations within it

USE StudentDB;

-- (a) Create a table with 2 columns – Student\_ID, Marks

-- Student\_ID is set as the PRIMARY KEY and will auto-increment for each new record.

CREATE TABLE Students (

Student\_ID INT PRIMARY KEY AUTO\_INCREMENT,

Marks INT

);

-- (b) Insert 2 records in this table

INSERT INTO Students (Marks) VALUES (85);

INSERT INTO Students (Marks) VALUES (92);

-- Optional: You can run this command to verify the records were inserted

-- SELECT \* FROM Students;

-- (c) Rename the Marks column with Score

-- Note: The exact syntax can vary slightly between SQL dialects. This is standard for MySQL.

ALTER TABLE Students RENAME COLUMN Marks TO Score;

-- (d) Then delete the table

DROP TABLE Students;

-- (e) Then delete the Database

DROP DATABASE StudentDB;