1. Create a table called employees with the following structure emp_id (integer, should not be NULL and should be a primary key) emp_name (text, should not be NULL) age (integer, should have a check constraint to ensure the age is at least 18) email (text, should be unique for each employee) salary (decimal, with a default value of 30,000). Write the SQL query to create the above table with all constraints.

```
CREATE TABLE employees (
   emp_id INTEGER PRIMARY KEY NOT NULL,
   emp_name TEXT NOT NULL,
   age INTEGER CHECK (age >= 18),
   email TEXT UNIQUE NOT NULL,
   salary DECIMAL DEFAULT 30000
);
```

2. Explain the purpose of constraints and how they help maintain data integrity in a database. Provide examples of common types of constraints.

Constraints are rules that define and enforce the data integrity, consistency and validity of the data in a database.
It helps in preventing errors in a database.
Examples:-

Unique:- It ensures that no other row contains the same value in the specific column.
Not Null:- It ensures that there should not be a null value in the specific column.
Primary Key:- It is a combination of unique and not null constraints.
It ensures that a column uniquely identifies each row in a table.

3.Why would you apply the NOT NULL constraint to a column? Can a primary key contain NULL values? Justify your answer.

The not null constraint is applied to a column when we want that column to always have a value. This constraint ensures that no empty values are allowed in that column.

And primary key cant have null values because it is the combination of unique and not null constraints.
And it identifies all the rows uniquely so it cant be null.

4. Explain the steps and SQL commands used to add or remove constraints on an existing table. Provide an example for both adding and removing a constraint.

In SQL constraints can be added or removed from an existing table using the `ALTER TABLE` command.
Steps to add constraints :-

1. Select the column on which you want to apply a constraint and which constraint.
2. Define constraint after ALTER TABLE statement.
3. Execute command.
   Example :- ALTER TABLE employees
   ADD CONSTRAINT chk_age CHECK (age >= 18);

Steps to remove a constraint :-

1. Select the constraint you want to remove.
2. Use the ALTER TABLE command with DROP CONSTRAINT
3. Execute command.
   Example :- ALTER TABLE employees
   DROP CONSTRAINT chk_age;

5. Explain the consequences of attempting to insert, update, or delete data in a way that violates constraints. Provide an example of an error message that might occur when violating a constraint.

The consequences of attempting to insert, update, or delete data in a way that violates constraints is that the command will not execute and also raise an error.
When we violate a constraint we can get an error like :-

ERROR: null value in column "emp_name" violates not-null constraint

This error occurred when we tried to leave emp_name column (which is not null) empty while entering data.

6. You created a products table without constraints as follows:
CREATE TABLE products (product_id INT, product_name VARCHAR(50), price DECIMAL(10, 2));
Now, you realise that The product_id should be a primary key The price should have a default value of 50.00

To alter the table you need to use the ALTER TABLE command.

```
ALTER TABLE products
ADD CONSTRAINT pk_product_id PRIMARY KEY (product_id);

ALTER TABLE products
ALTER COLUMN price SET DEFAULT 50.00;
```

7. You have two tables:
Write a query to fetch the student_name and class_name for each student using an INNER JOIN.

```
SELECT s.student_name, c.class_name
FROM students s
INNER JOIN classes c ON s.class_id = c.class_id;
```

8. Consider the following three tables:
Write a query that shows all order_id, customer_name, and product_name, ensuring that all products are listed even if they are not associated with an order Hint: (use INNER JOIN and LEFT JOIN).

```
SELECT o.order_id, c.customer_name, p.product_name
FROM products p
LEFT JOIN orders o ON p.order_id = o.order_id
LEFT JOIN customers c ON o.customer_id = c.customer_id;
```

9. Given the following tables:
Write a query to find the total sales amount for each product using an INNER JOIN and the SUM() function.

```
SELECT p.product_name, SUM(s.amount) AS total_amount
FROM sale s
INNER JOIN product p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

10. You are given three tables
Write a query to display the order_id, customer_name, and the quantity of products ordered by each customer using an INNER JOIN between all three tables

```
SELECT o.order_id, c.customer_name, od.quantity
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id
INNER JOIN order_detail od ON o.order_id = od.order_id;
```

SQL Commands

1-Identify the primary keys and foreign keys in maven movies db. Discuss the differences

A primary key is a column in a table that uniquely identifies each row in that table.
A primary key in movie db can be movie_id which will be unique for every movie.
A foreign key is a column in one table that points to the primary key of another table.
A foreign key might be release_date which will point to movie_id.

2- List all details of actors

To see all detail from actors table we use:-
SELECT * FROM actor;
Output:-

1	PENELOPE	GUINESS	2006-02-15 04:34:33
2	NICK	WAHLBERG	2006-02-15 04:34:33
3	ED	CHASE	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33

3 -List all customer information from DB

To see all detail from customer table we use:-
SELECT * FROM customer;

Output:-

1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20

4 -List different countries.

To see list of countries from countries table we use:-
SELECT country_id, country FROM country;

Output:-

1 Afghanistan

2 Algeria

3 American Samoa

4 Angola

5 Anguilla

5 -Display all active customers.

To see active customer we use:-

SELECT customer_id, first_name FROM customer c
WHERE c.active=1;

Output:-

1 MARY

2 PATRICIA

3 LINDA

4 BARBARA

5 ELIZABETH

6 -List of all rental IDs for customer with ID 1.

To see all rental id of customer id 1 we can use:-

```
SELECT rental_id,customer_id
FROM rental
WHERE customer_id = 1;
```

Output:-

76	1
573	1
1185	1
1422	1
1476	1

7 - Display all the films whose rental duration is greater than 5 .

To see all the films whose rental duration is greater than 5 we can use:-

```
SELECT film_id, title, rental_duration
FROM film
WHERE rental_duration > 5;
```

Output:-

1	ACADEMY DINOSAUR	6
3	ADAPTATION HOLES	7
5	AFRICAN EGG	6
7	AIRPLANE SIERRA	6
8	AIRPORT POLLOCK	6

8 - List the total number of films whose replacement cost is greater than $15 and less than $20.

To get this result we use:-

```
SELECT COUNT(*) AS total_films
FROM film
WHERE replacement_cost > 15 AND replacement_cost < 20;
```

Output:-

```
total_films
214
```

9 - Display the count of unique first names of actors.

To get this result we use:-

```
SELECT first_name, COUNT(*) AS name_count
FROM actor
GROUP BY first_name;
```

Output:-

PENELOPE	4
NICK	3
ED	3
JENNIFER	1
JOHNNY	2

10- Display the first 10 records from the customer table .

To get this result we use:-

1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36

2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36

11 - Display the first 3 records from the customer table whose first name starts with ‘b’

To get this result we use:-

```
SELECT *
FROM customer
WHERE first_name LIKE 'B%'
LIMIT 3;
```

Output:-

4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:36
31	2	BRENDA	WRIGHT	BRENDA.WRIGHT@sakilacustomer.org	35	1	2006-02-14 22:04:36

12 -Display the names of the first 5 movies which are rated as ‘G’

To get this result we use:-

```
SELECT title,rating
FROM film
WHERE rating = 'G'
LIMIT 5;
```

Output:-

ACE GOLDFINGER	G
AFFAIR PREJUDICE	G
AFRICAN EGG	G
ALAMO VIDEOTAPE	G
AMISTAD MIDSUMMER	G

13-Find all customers whose first name starts with "a".

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE 'A%';
```

Output:-

29	ANGELA
32	AMY
33	ANNA
40	AMANDA
48	ANN

14- Find all customers whose first name ends with "a".

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE '%a';
```

Output:-

2	PATRICIA
3	LINDA
4	BARBARA
7	MARIA
11	LISA

15- Display the list of first 4 cities which start and end with ‘a’ .

To get this result we use:-

```
SELECT city_id,city
FROM city
WHERE city LIKE 'a%' AND city LIKE '%a'
LIMIT 4;
```

Output:-

2	Abha
4	Acuña
5	Adana
6	Addis Abeba

16- Find all customers whose first name have "NI" in any position.

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE '%NI%';
```

Output:-

6	JENNIFER
35	VIRGINIA
41	STEPHANIE
66	JANICE
68	NICOLE

17- Find all customers whose first name have "r" in the second position .

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE '_r%';
```

Output:-

```
 31    BRENDA
 47    FRANCES
 76    IRENE
102    CRYSTAL
108    TRACY
```

18 - Find all customers whose first name starts with "a" and are at least 5 characters in length.

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE 'a____%';
```

Output:-

```
 29    ANGELA
 40    AMANDA
 51    ALICE
 63    ASHLEY
 81    ANDREA
```

19- Find all customers whose first name starts with "a" and ends with "o"

To get this result we use:-

```
SELECT customer_id, first_name
FROM customer
WHERE first_name LIKE 'a%o';
```

Output:-

398	ANTONIO
556	ARMANDO
567	ALFREDO
568	ALBERTO

20 - Get the films with pg and pg-13 rating using IN operator

To get this result we use:-

```
SELECT title, rating
FROM film
WHERE rating IN ('PG', 'PG-13');
```

Output:-

ACADEMY DINOSAUR	PG
AGENT TRUMAN	PG
AIRPLANE SIERRA	PG-13
ALABAMA DEVIL	PG-13
ALASKA PHANTOM	PG

21 - Get the films with length between 50 to 100 using between operator

To get this result we use:-

```
SELECT title, length
FROM film
WHERE length BETWEEN 50 AND 100;
```

Output:-

ACADEMY DINOSAUR	86
ADAPTATION HOLES	50
AIRPLANE SIERRA	62
AIRPORT POLLOCK	54
ALADDIN CALENDAR	63

22 - Get the top 50 actors using limit operator.

To get this result we use:-

```
SELECT actor_id,first_name
FROM actor
LIMIT 50;
```

Output:-

1	PENELOPE
2	NICK
3	ED
4	JENNIFER
5	JOHNNY

23 - Get the distinct film ids from inventory table.

To get this result we use:-

```
SELECT DISTINCT inventory_id, film_id
FROM inventory;
```

Output:-

1	1
2	1
3	1
4	1
5	1

Functions

Question 1: Retrieve the total number of rentals made in the Sakila database. Hint: Use the COUNT() function.

To get this result we use:-

```
SELECT COUNT(rental_id) AS total_rentals
FROM rental;
```

Output:-

total_rentals
16044

Question 2: Find the average rental duration (in days) of movies rented from the Sakila database. Hint: Utilize the AVG() function

To get this result we use:-

```
SELECT AVG(DATEDIFF(return_date, rental_date)) AS avg_rental_duration
FROM rental
WHERE return_date IS NOT NULL;
```

Output:-

avg_rental _duration
5.0252

Question 3: Display the first name and last name of customers in uppercase. Hint: Use the UPPER () function

To get this result we use:-

```
SELECT UPPER(first_name) AS first_name_upper,
    UPPER(last_name) AS last_name_upper
FROM customer;
```

Output:-

MARY	SMITH
PATRICIA	JOHNSON
LINDA	WILLIAMS
BARBARA	JONES
ELIZABETH	BROWN

Question 4: Extract the month from the rental date and display it alongside the rental ID. Hint: Employ the MONTH() function.

To get this result we use:-

```
SELECT rental_id,
    MONTH(rental_date) AS rental_month
FROM rental;
```

Output:-

1	5
2	5
3	5
4	5
5	5

Question 5: Retrieve the count of rentals for each customer (display customer ID and the count of rentals). Hint: Use COUNT () in conjunction with GROUP BY.

To get this result we use:-

```
SELECT customer_id,
    COUNT(rental_id) AS rental_count
FROM rental
GROUP BY customer_id;
```

Output:-

1	32
2	27
3	26
4	22
5	38

Question 6: Find the total revenue generated by each store. Hint: Combine SUM() and GROUP BY.

To get this result we use:-

```
SELECT s.store_id,
    SUM(p.amount) AS total_revenue
FROM payment p
JOIN staff st ON p.staff_id = st.staff_id
JOIN store s ON st.store_id = s.store_id
GROUP BY s.store_id;
```

Output:-

1	33482.50
2	33924.06

Question 7: Determine the total number of rentals for each category of movies. Hint: JOIN film_category, film, and rental tables, then use cOUNT () and GROUP BY.

To get this result we use:-

```
SELECT fc.category_id,
      COUNT(r.rental_id) AS total_rentals
FROM rental r
JOIN film_category fc ON r.inventory_id = fc.film_id
JOIN film f ON fc.film_id = f.film_id
GROUP BY fc.category_id;
```

Output:-

1 228

2 226

3 212

4 204

5 192

Question 8: Find the average rental rate of movies in each language. Hint: JOIN film and language tables, then use AVG () and GROUP BY.

To get this result we use:-

```
SELECT l.name AS language,
      AVG(f.rental_rate) AS avg_rental_rate
FROM film f
JOIN language l ON f.language_id = l.language_id
GROUP BY l.name;
```

Output:-

English 2.980000

Questions 9 - Display the title of the movie, customer s first name, and last name who rented it. Hint: Use JOIN between the film, inventory, rental, and customer tables.

To get this result we use:-

```
SELECT f.title AS movie_title,
      c.first_name,
      c.last_name
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
JOIN customer c ON r.customer_id = c.customer_id;
```

Output:-

ACADEMY DINOSAUR	JOEL	FRANCISCO
ACADEMY DINOSAUR	GABRIEL	HARDER
ACADEMY DINOSAUR	DIANNE	SHELTON
ACADEMY DINOSAUR	NORMAN	CURRIER
ACADEMY DINOSAUR	BEATRICE	ARNOLD

Question 10: Retrieve the names of all actors who have appeared in the film "Gone with the Wind." Hint: Use JOIN between the film actor, film, and actor tables

To get this result we use:-

```
SELECT a.first_name,
      a.last_name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
WHERE f.title = 'Gone with the Wind';
```

Output:-
There is no movie named Gone with the Wind so no output is found.

Question 11: Retrieve the customer names along with the total amount they've spent on rentals. Hint: JOIN customer, payment, and rental tables, then use SUM() and GROUP BY

To get this result we use:-

```
SELECT c.first_name,
      c.last_name,
      SUM(p.amount) AS total_spent
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
JOIN rental r ON p.rental_id = r.rental_id
GROUP BY c.customer_id;
```

Output:-

MARY	SMITH	118.68
PATRICIA	JOHNSON	128.73
LINDA	WILLIAMS	135.74
BARBARA	JONES	81.78
ELIZABETH	BROWN	144.62

Question 12: List the titles of movies rented by each customer in a particular city (e.g., 'London'). Hint: JOIN customer, address, city, rental, inventory, and film tables, then use GROUP BY.

To get this result we use:-

```
SELECT c.first_name,
      c.last_name,
      f.title AS movie_title
FROM customer c
JOIN address a ON c.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
WHERE ci.city = 'London'  -- Correct column name
ORDER BY c.first_name, c.last_name, f.title;
```

Output:-

CECIL	VINES	AMADEUS HOLY
CECIL	VINES	ARABIA DOGMA
CECIL	VINES	BACKLASH UNDEFEATED
CECIL	VINES	BLOOD ARGONAUTS
CECIL	VINES	CAT CONEHEADS
CECIL	VINES	CAT CONEHEADS
CECIL	VINES	CHARIOTS CONSPIRACY

Question 13: Display the top 5 rented movies along with the number of times they've been rented. Hint: JOIN film, inventory, and rental tables, then use COUNT () and GROUP BY, and limit the results.

To get this result we use:-

```
SELECT f.title AS movie_title,
      COUNT(r.rental_id) AS rental_count
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY f.film_id
ORDER BY rental_count DESC
LIMIT 5;
```

Output:-

BUCKET BROTHERHOOD	34
ROCKETEER MOTHER	33
FORWARD TEMPLE	32
GRIT CLOCKWORK	32
JUGGLER HARDLY	32

Question 14: Determine the customers who have rented movies from both stores (store ID 1 and store ID 2). Hint: Use JOINS with rental, inventory, and customer tables and consider COUNT() and GROUP BY.

To get this result we use:-

```
SELECT c.customer_id,
      c.first_name,
      c.last_name
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
WHERE i.store_id IN (1, 2)
GROUP BY c.customer_id
HAVING COUNT(DISTINCT i.store_id) = 2;
```

Output:-

1	MARY	SMITH
2	PATRICIA	JOHNSON
3	LINDA	WILLIAMS
4	BARBARA	JONES
5	ELIZABETH	BROWN

Windows Function:

1. Rank the customers based on the total amount they've spent on rentals.

To get this result we use:-

```
SELECT customer_id,
      first_name,
      last_name,
      total_spent,
      @rank := @rank + 1 AS new_rank
FROM (
   SELECT c.customer_id,
         c.first_name,
         c.last_name,
         SUM(p.amount) AS total_spent
   FROM customer c
   JOIN payment p ON c.customer_id = p.customer_id
   GROUP BY c.customer_id
   ORDER BY total_spent DESC
) AS ranked_customers,
(SELECT @rank := 0) AS init_rank
LIMIT 0, 1000;
```

Output:-

526	KARL	SEAL	221.55	1
148	ELEANOR	HUNT	216.54	2
144	CLARA	SHAW	195.58	3
137	RHONDA	KENNEDY	194.61	4
178	MARION	SNYDER	194.61	5
459	TOMMY	COLLAZO	186.62	6

2. Calculate the cumulative revenue generated by each film over time

To get this result we use:-

```
SELECT f.title AS film_title,
       p.payment_date,
       SUM(p.amount) OVER (PARTITION BY f.film_id ORDER BY p.payment_date) AS
cumulative_revenue
FROM film f
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
ORDER BY f.film_id, p.payment_date;
```

Output:-

ACADEMY DINOSAUR	2005-05-27 07:03:28	0.99
ACADEMY DINOSAUR	2005-05-30 20:21:07	2.98
ACADEMY DINOSAUR	2005-06-15 02:57:51	3.97
ACADEMY DINOSAUR	2005-06-17 20:24:00	4.96
ACADEMY DINOSAUR	2005-06-21 00:30:26	6.95

3. Determine the average rental duration for each film, considering films with similar lengths

To get this result we use:-

```
SELECT
   CASE
      WHEN f.length BETWEEN 90 AND 120 THEN '90-120 minutes'
      WHEN f.length BETWEEN 120 AND 150 THEN '120-150 minutes'
      WHEN f.length BETWEEN 150 AND 180 THEN '150-180 minutes'
      ELSE 'Other lengths'
   END AS film_length_range,
   AVG(f.rental_duration) AS average_rental_duration
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY film_length_range
ORDER BY film_length_range
LIMIT 0, 1000;
```

Output:-

120-150 minutes	4.9080
150-180 minutes	5.0283
90-120 minutes	4.9806
Other lengths	4.8723

4. Identify the top 3 films in each category based on their rental counts

To get this result we use:-

```
SELECT fc.category_id,
      c.name AS category_name,
      f.title AS film_title,
      COUNT(r.rental_id) AS rental_count
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY fc.category_id, f.film_id
ORDER BY fc.category_id, rental_count DESC
LIMIT 3;
```

Output:-

1	Action	RUGRATS SHAKESPEARE	30
1	Action	SUSPECTS QUILLS	30
1	Action	STORY SIDE	28

5. Calculate the difference in rental counts between each customer's total rentals and the average rentals across all customers.

To get this result we use:-

```
SELECT c.customer_id,
       c.first_name,
       c.last_name,
       customer_rentals.total_rentals,
       avg_rentals.avg_rentals,
       (customer_rentals.total_rentals - avg_rentals.avg_rentals) AS rental_diff
FROM (
   SELECT r.customer_id, COUNT(r.rental_id) AS total_rentals
   FROM rental r
   GROUP BY r.customer_id
) AS customer_rentals
JOIN customer c ON customer_rentals.customer_id = c.customer_id
CROSS JOIN (
   SELECT AVG(total_rentals) AS avg_rentals
   FROM (
      SELECT COUNT(r.rental_id) AS total_rentals
      FROM rental r
      GROUP BY r.customer_id
   ) AS all_customer_rentals
) AS avg_rentals
ORDER BY rental_diff DESC;
```

Output:-

148	ELEANOR	HUNT	46	26.7846	19.2154
526	KARL	SEAL	45	26.7846	18.2154
144	CLARA	SHAW	42	26.7846	15.2154
236	MARCIA	DEAN	42	26.7846	15.2154
75	TAMMY	SANDERS	41	26.7846	14.2154

6. Find the monthly revenue trend for the entire rental store over time

To get this result we use:-

```
SELECT
   YEAR(p.payment_date) AS year,
   MONTH(p.payment_date) AS month,
   SUM(p.amount) AS total_revenue
FROM payment p
GROUP BY YEAR(p.payment_date), MONTH(p.payment_date)
ORDER BY year, month;
```

Output:-

2005	5	4823.44
2005	6	9629.89
2005	7	28368.91
2005	8	24070.14
2006	2	514.18

7. Identify the customers whose total spending on rentals falls within the top 20% of all customers.

To get this result we use:-

```
SELECT
   customer_id,
   first_name,
   last_name,
   total_spent
FROM (
   SELECT
      c.customer_id,
      c.first_name,
      c.last_name,
      SUM(p.amount) AS total_spent,
      @rank := @rank + 1 AS rank1
   FROM customer c
   JOIN payment p ON c.customer_id = p.customer_id
   GROUP BY c.customer_id
   ORDER BY total_spent DESC
```

```
) AS ranked_customers,
(SELECT @rank := 0) AS init_rank
WHERE rank1 <= (SELECT FLOOR(COUNT(*) * 0.2) FROM customer)
ORDER BY total_spent DESC;
```

Output:-

50	DIANE	COLLINS	169.65
21	MICHELLE	CLARK	155.65
75	TAMMY	SANDERS	155.59
119	SHERRY	MARSHALL	153.66
26	JESSICA	HALL	152.66

8. Calculate the running total of rentals per category, ordered by rental count.

To get this result we use:-

```
SELECT
    fc.category_id,
    c.name AS category_name,
    COUNT(r.rental_id) AS rental_count,
    SUM(COUNT(r.rental_id)) OVER (ORDER BY COUNT(r.rental_id) DESC) AS
running_total_rentals
FROM film_category fc
JOIN film f ON fc.film_id = f.film_id
JOIN rental r ON f.film_id = r.inventory_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY fc.category_id, c.name
ORDER BY rental_count DESC;
```

15	Sports	259	259
9	Foreign	252	511
6	Documentary	239	750
8	Family	235	985
7	Drama	229	1214

9. Find the films that have been rented less than the average rental count for their respective categories.

To get this result we use:-

```
SELECT
    f.title,
    c.name AS category_name,
    COUNT(r.rental_id) AS rental_count,
    avg_rental_count.avg_rentals_per_category
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN rental r ON f.film_id = r.inventory_id
JOIN category c ON fc.category_id = c.category_id
JOIN (
    SELECT
        category_id,
        AVG(rental_count) AS avg_rentals_per_category
    FROM (
        SELECT
            fc.category_id,
            COUNT(r.rental_id) AS rental_count
        FROM rental r
        JOIN inventory i ON r.inventory_id = i.inventory_id
        JOIN film_category fc ON i.film_id = fc.film_id
        GROUP BY fc.category_id, fc.film_id
    ) AS rental_counts_per_film
    GROUP BY category_id
) AS avg_rental_count ON fc.category_id = avg_rental_count.category_id
GROUP BY f.film_id, c.category_id
HAVING rental_count < avg_rental_count.avg_rentals_per_category
ORDER BY rental_count ASC;
```

Output:-

BERETS AGENT	Action	2	18.2295
CADDYSHACK JEDI	Action	2	18.2295
DRAGON SQUAD	Action	2	18.2295
EASY GLADIATOR	Action	2	18.2295
FORREST SONS	Action	2	18.2295

10. Identify the top 5 months with the highest revenue and display the revenue generated in each month.

To get this result we use:-

```
SELECT
   YEAR(p.payment_date) AS year,
   MONTH(p.payment_date) AS month,
   SUM(p.amount) AS total_revenue
FROM payment p
GROUP BY YEAR(p.payment_date), MONTH(p.payment_date)
ORDER BY total_revenue DESC
LIMIT 5;
```

Output:-

2005	7	28368.91
2005	8	24070.14
2005	6	9629.89
2005	5	4823.44
2006	2	514.18

Normalisation & CTE

1. First Normal Form (1NF): a. Identify a table in the Sakila database that violates 1NF. Explain how you would normalize it to achieve 1NF.

Address table in Sakila db violates 1NF rule.
Because it contain two column to store address which is not allowed in 1NF.
We can convert it into 1NF by removing one of the address column and add it in another row with same data.

2. Second Normal Form (2NF): a. Choose a table in Sakila and describe how you would determine whether it is in 2NF. If it violates 2NF, explain the steps to normalize it.

Lets choose actor table it contains actor_id, first_name, last_name, last_update.
And actor_id can uniquely define every column in the table so this table is in 2NF.
If it was not in 2NF we would have to break table into smaller table which contain a primary key which can define each column uniquely.

3. Third Normal Form (3NF): a. Identify a table in Sakila that violates 3NF. Describe the transitive dependencies present and outline the steps to normalize the table to 3NF.

In sakila db rental table do not follow 3NF rules because it have staff_name attribute which depends on staff_id attribute which is not allowed in 3NF.

A transitive dependency occurs when a non-key attribute depends on another non-key attribute which depends on the primary key.

To convert this table in 3NF we have to break the table into two parts.
New table will contain staff_id and staff_name where as other table will contain rest of the attributes with staff_id.

4. Normalization Process: a. Take a specific table in Sakila and guide through the process of normalizing it from the initial unnormalized form up to at least 2NF

Lets take rental table.
In this table first we have to find if it have any repeating column.
After analyzing we have found all the attributes are atomic.
Which means that table is in 1NF.
Now lets check if it is in 2NF.
If table have partial dependencies it will not be in 2NF.
As we can see it do not contain any partial dependencies so it is in 2NF also.

5. CTE Basics: a. Write a query using a CTE to retrieve the distinct list of actor names and the number of films they have acted in from the actor and film_actor tables

```
WITH ActorFilmCount AS (
    SELECT
        a.actor_id,
        CONCAT(a.first_name, ' ', a.last_name) AS actor_name,
        COUNT(fa.film_id) AS films_count
    FROM actor a
    JOIN film_actor fa ON a.actor_id = fa.actor_id
    GROUP BY a.actor_id
)
SELECT
    actor_name,
    films_count
FROM ActorFilmCount
ORDER BY films_count DESC;
```

Output:-

GINA DEGENERES	42
WALTER TORN	41
MARY KEITEL	40
MATTHEW CARREY	39
SANDRA KILMER	37

6. CTE with Joins: a. Create a CTE that combines information from the film and language tables to display the film title, language name, and rental rate.

```
WITH FilmLanguageDetails AS (
    SELECT
        f.title AS film_title,
        l.name AS language_name,
        f.rental_rate
    FROM film f
    JOIN language l ON f.language_id = l.language_id
)
SELECT
    film_title,
    language_name,
    rental_rate
```

```
FROM FilmLanguageDetails
ORDER BY rental_rate DESC;
```

Output:-

ACE GOLDFINGER	English	4.99
AIRPLANE SIERRA	English	4.99
AIRPORT POLLOCK	English	4.99
ALADDIN CALENDAR	English	4.99
ALI FOREVER	English	4.99

7.CTE for Aggregation: a. Write a query using a CTE to find the total revenue generated by each customer (sum of payments) from the customer and payment tables

```
WITH CustomerRevenue AS (
   SELECT
      c.customer_id,
      CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
      SUM(p.amount) AS total_revenue
   FROM customer c
   JOIN payment p ON c.customer_id = p.customer_id
   GROUP BY c.customer_id
)
SELECT
   customer_name,
   total_revenue
FROM CustomerRevenue
ORDER BY total_revenue DESC;
```

Output:-

KARL SEAL	221.55
ELEANOR HUNT	216.54
CLARA SHAW	195.58
RHONDA KENNEDY	194.61
MARION SNYDER	194.61

8. CTE with Window Functions: a. Utilize a CTE with a window function to rank films based on their rental duration from the film table

```
WITH RankedFilms AS (
    SELECT
        f.film_id,
        f.title,
        f.rental_duration,
        RANK() OVER (ORDER BY f.rental_duration DESC) AS rental_rank
    FROM film f
)
SELECT
    film_id,
    title,
    rental_duration,
    rental_rank
FROM RankedFilms
ORDER BY rental_rank;
```

Output:-

3	ADAPTATION HOLES	7	1
27	ANONYMOUS HUMAN	7	1
36	ARGONAUTS TOWN	7	1
70	BIKINI BORROWERS	7	1
78	BLACKOUT PRIVATE	7	1

9. CTE and Filtering: a. Create a CTE to list customers who have made more than two rentals, and then join this CTE with the customer table to retrieve additional customer details.

```
WITH CustomerRentals AS (
    SELECT
        c.customer_id,
        COUNT(r.rental_id) AS rental_count
    FROM customer c
    JOIN rental r ON c.customer_id = r.customer_id
    GROUP BY c.customer_id
    HAVING COUNT(r.rental_id) > 2
)
SELECT
```

```
    c.customer_id,
    c.first_name,
    c.last_name,
    c.email,
    cr.rental_count
FROM customer c
JOIN CustomerRentals cr ON c.customer_id = cr.customer_id
ORDER BY cr.rental_count DESC;
```

Output:-

148	ELEANOR	HUNT	ELEANOR.HUNT@sakilacustomer.org	46
526	KARL	SEAL	KARL.SEAL@sakilacustomer.org	45
144	CLARA	SHAW	CLARA.SHAW@sakilacustomer.org	42
236	MARCIA	DEAN	MARCIA.DEAN@sakilacustomer.org	42
75	TAMMY	SANDERS	TAMMY.SANDERS@sakilacustomer.org	41

10.CTE for Date Calculations: a. Write a query using a CTE to find the total number of rentals made each month, considering the rental_date from the rental table

```
WITH MonthlyRentals AS (
    SELECT
        YEAR(r.rental_date) AS rental_year,
        MONTH(r.rental_date) AS rental_month,
        COUNT(r.rental_id) AS rental_count
    FROM rental r
    GROUP BY YEAR(r.rental_date), MONTH(r.rental_date)
)
SELECT
    rental_year,
    rental_month,
    rental_count
FROM MonthlyRentals
ORDER BY rental_year, rental_month;
```

Output:-

2005	5	1156
2005	6	2311

2005 7 6709

2005 8 5686

2006 2 182

11. CTE and Self-Join: a. Create a CTE to generate a report showing pairs of actors who have appeared in the same film together, using the film_actor table

```
WITH ActorPairs AS (
    SELECT
        fa1.actor_id AS actor_id_1,
        fa2.actor_id AS actor_id_2,
        fa1.film_id
    FROM film_actor fa1
    JOIN film_actor fa2 ON fa1.film_id = fa2.film_id
    WHERE fa1.actor_id < fa2.actor_id  -- Ensure unique pairs (avoid reverse duplicates)
)
SELECT
    ap.actor_id_1 AS actor_1,
    ap.actor_id_2 AS actor_2,
    f.title AS film_title
FROM ActorPairs ap
JOIN film f ON ap.film_id = f.film_id
ORDER BY f.title, ap.actor_id_1, ap.actor_id_2;
```

Output:-

1 10 ACADEMY DINOSAUR

1 20 ACADEMY DINOSAUR

1 30 ACADEMY DINOSAUR

1 40 ACADEMY DINOSAUR

1 53 ACADEMY DINOSAUR

12. CTE for Recursive Search: a. Implement a recursive CTE to find all employees in the staff table who report to a specific manager, considering the reports_to column