6. Advanced SQL Commands.

6.1 Time stamps

→ Postgress SQL extract function extracts parts from a date.

- Extract (unit from date)

Extract function documents for reference

Postgres SQL Time date functions in documentations

```
SELECT * FROM payment;
SELECT payment_date FROM payment;
SELECT Extract(day from payment_date) FROM payment;
```

```
SELECT SUM(amount), Extract(month from payment_date) AS month
FROM payment
GROUP BY month
ORDER BY SUM(amount) DESC
LIMIT 1;
```

6.2 Mathematical Functions

Eg: SELECT * FROM payment;

SELECT cust_id + staff_id AS New_id
FROM payment;

SELECT AVG (amount)

SELECT ROUND (AVG (amount), 2)

6.3 String Function & operators

SELECT first_name || last_name FROM Customer;
(concatinate)

SELECT first_name || ' ' || last_name AS full_name
FROM Customer;

SELECT first name, char_length(first_name)
FROM Customer;

64 Sub-Query => very advanced & complicated

Regular way to solve the problem

→ Suppose we want to find the films whose rental rate is higher than the average rental rate.

↳ SELECT AVG(rental_rate) FROM film;

↳ SELECT title, rental_rate
FROM film
WHERE rental_rate > 2.98;

A Sub query is a query nested inside another query.
→ To construct a subquery, we put the second query in brackets and use it in the WHERE clause as an expression.

```
SELECT film_id, title, rental_rate FROM film
WHERE
rental_rate > (SELECT AVG(rental_rate) FROM film);
```

```
SELECT title, rental_rate
FROM film
WHERE rental_rate > (SELECT AVG(rental_rate)
                     FROM film)
```

```
(SELECT inventory.film_id
 FROM rental
 INNER JOIN inventory ON inventory.inventory_id = rental.inventory
                                                  _id.
 WHERE
 return_date BETWEEN '2005-05-29' And '2005-05-30')
```

Subquery above this

```
SELECT film_id, title
FROM film
WHERE film_id IN
```

6.5 # Self-JOIN

Eg:

Employee name	Employee location
Joe	New YORK
Sunil	India
Alex	Russia
Albert	Canada
Jack	New YORK.

We Can not Just say:

```
SELECT employee_name.
FROM employee.
WHERE Employee_location = "NEW YORK"
```

→ We Can use Sub-query like below

```
SELECT employee_name FROM employee.
WHERE employee_location IN
(SELECT employee_location FROM employee.
 WHERE Employee_name = "Joe")
```

→ while a subquery is a valid solution, it is actually more efficient to use a self join, where we join a table to itself.

→ As a general rule, we need to always use aliases (AS statement) when using self join

```
SELECT e1, employee_name.
FROM employee AS e1, employee AS e2.
WHERE
e1.employee_location = e2.employee_location
AND e2.employee_name = "Joe";
```

Now Back to original table Now

```
SELECT a.first_name, a.last_name, b.first_name,
                                  b.last_name
FROM Customer AS a, Customer AS b.
WHERE a.first_name = b.last_name;
```

Another way to do is below

```
SELECT a.Customer_id, a.first_name, a.last_name,
       b.Customer_id, b.first_name, b.last_name
FROM Customer AS a
JOIN Customer AS b
ON a.first_name = b.last_name
ORDER BY a.Customer_id;
```

Easier one ↑

Can also do LEFT JOIN / LEFT OUTER JOIN

Google search

Manager - Employee self join

Common interview question in SQL Join