→ Write Complexe Queries

↳ Subqueries

- Subqueries allow us to use the result of one query as the input to another query. In other words, we can use the output of one query as a filter condition or a join condition in another query.
- Subqueries can be used in various clauses such Where, From, Having and select.

Examples:

1)
```
SELECT *
FROM customers
WHERE customer_id IN (
        SELECT customer_id
        FROM orders
        WHERE order_date >= '2022-01-01';
```

In this example, the subquery return list of customers who have placed an order after Jan 1, 2022.

2)
```
SELECT product_name, unit_price,
    (SELECT AVG(unit_price) FROM products)
    AS avg_price
FROM product;
```

In this example, the subquery return the avg of 'unit_price', the result of this subquery is selected in the new column 'avg_price'

The IN Operator

- The IN operator is commonly used in subqueries to specify a list of values for comparison.
- The IN operator is used in the WHERE clause of a query to filter the result based on a list of values return by a subquery

Example:

```
SELECT *
FROM customers
WHERE customer_id IN (
        SELECT customer_id
        FROM orders
        WHERE order_total > 100);
```

In this example, the subquery return a list of customers who have placed an order with a total greater than 100.

- The IN operator can also be used to compare a single value against a list of values returned by a subquery

Example:

```
SELECT *
FROM products
WHERE category_id IN (
        SELECT category_id
        FROM categories
        WHERE category_name = 'Electronics');
```

In this example, the subquery return a list of category with a name 'Electronics'
so the outer query return all product that are Electronics

↳ The ALL Keyword

- The ALL Keyword in SQL is used to compare a single value to all the values returned by a subquery. The subquery should return a list of values that can be compared with the single value.
- The comparison operator used with ALL is usually '>', '<', '>=', '<=', '=' or '<>'

Example:

```
SELECT *
FROM product
WHERE unit_price >= ALL (
        SELECT unit price
        FROM products
        WHERE category_id = 1 );
```

In this example, the subquery return a list of unit price values for all products that belong to category one

- The ALL Keyword is often used with aggregate function such as 'SUM', 'AVG', 'MAX', and 'MIN' to compare a single value with the result of the aggregate function

↳ The ANY Keyword

- The any Key_word is another way to compair a single value with a list of values returned by a Sub.query

- ANY Keyword is equivalen to the IN operator

Example:

```
--using ANY
 Select *
 From products
 Where unit_price > ANY (
     Select unit_price
     From products
     Where category_id=1);

--using IN
 Select *
 From products
 Where unit_price IN(
      Select unit-price
      From products
      Where category_id=1);
```

Both queries turn the same result set, because they are filtering the rows where 'unit_price' is in the list of 'unit_price' values returned by the subquery.

Correlated Subqueries

- A correlated subquery is a type of subquery that uses a value from the outer query in the WHERE clause of the subquery.
- The subquery is executed for each row of the outer query, and the value from the outer query is used to filter the results of the subquery.

Example:

```
SELECT *
FROM orders o
WHERE order_date = (
        SELECT MAX (order_date)
        FROM orders
        WHERE customer_id = o.customer_id);
```

In this ex, the subquery returns the max order date for a particular customer. The subquery uses the 'customer_id' value from the outer query to filter results. The outer query select all rows from orders where the order_date matches the max order_date for each customer.