→ Sub-Query

a. A subquery is a query which is nested within another query
b. It is used to retrieve data from one or more tables and use a result of query as condition/criteria for another table
c. We can write almost 255 levels of subqueries
d. Larger the number of subqueries, slower the execution time
e. `JOIN` is faster than subqueries
f. In subqueries, we use operators like `IN`(logical OR), `ANY`(logical OR), `ALL`(logical AND)
g. Syntax for subquery in `FROM` clause:

```
SELECT <column_name> FROM (SELECT <column_name> FROM <table_name>)
WHERE <condition>;
```

h. Syntax for subquery in `WHERE` clause:

```
SELECT <colunm_name> FROM <table1_name> WHERE <column_name>
[operators =, NOT IN, IN, >, <, etc] (SELECT <column_name> FROM
<table2_name>);
```

i. Syntax for subquery in `HAVING` clause:

```
SELECT column_name FROM table1_name WHERE condition GROUP BY
column_name HAVING column_name [operator =, >, <, <>] (SELECT
coumn_name FROM table2_name);
```

→ Rules to write a subqueries

a. A subquery must be enclosed in parenthesis ( )
b. A subquery cannot return more than one column
c. A subquery can return only one row in most cases, however we can use some clauses like `IN`, `ANY`, which will allow subquery to return multiple rows
d. A subquery can be used in various clauses like `SELECT`, `FROM`, `WHERE`, `HAVING`, `DELETE`
e. Syntax of subquery will vary depending on the clause we have used
f. A subquery can be nested to multiple levels, but this can make the query more complex and harder to understand
g. A subquery slows down the query execution time. In order to have better performance, we need to optimize subqueries

→ Views

a. A `VIEW` is a virtual table, that contains specific results
b. Views are stored in the database like tables, and can be used to simplify queries, hide complex queries for end user and provide a layer of security or restriction to sensitive data

c. A `view` can be created on simple queries, bare statements, on joints, on subqueries

d. Key points for Views:

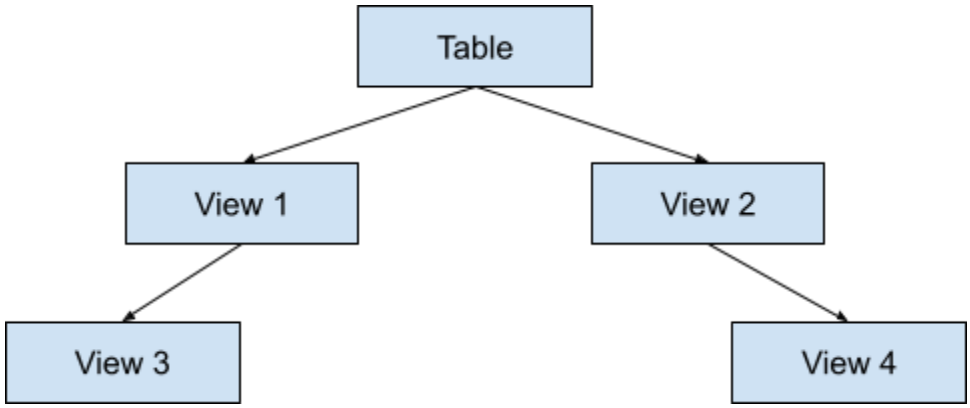
i. A view is a virtual table which displays selected columns from one or more tables

ii. It does not store actual data, but only `SELECT` query statement

iii. A view is a logical entity, while table is a physical entity

iv. You can create a view from another view, but if the table is dropped, view becomes inaccessible

v.



e. Advantages of Views:

i. It provides security and helps to hide columns having sensitive information underlying tables

ii. It reduces data redundancy

f. Creating View

i. Syntax to create a view:

```
CREATE VIEW <view_name> AS SELECT <column1>, <column2> FROM
<table_name> WHERE <condition>;
```

ii. Example of create view – Basic Example:

```
CREATE VIEW sal_emp AS SELECT FIRST_NAME, SALARY, DEPARTMENT_ID FROM
employees WHERE SALARY>10000;
```

iii. Example of create view – GROUP BY:

```
CREATE VIEW no_emp_dept AS SELECT DEPARTMENT_ID, count(*) FROM
employees GROUP BY DEPARTMENT_ID;
```

iv. Example of create view – JOIN:

```
CREATE VIEW join_view1 AS SELECT e.FIRST_NAME, d.DEPARTMENT_NAME,
e.SALARY FROM employees e JOIN departments d ON
e.DEPARTMENT_ID=d.DEPARTMENT_ID;
```

v. Example of create view – SubQuery:

```
CREATE VIEW sub_examp AS SELECT max(SALARY) FROM employees WHERE
SALARY<(SELECT max(SALARY) FROM employees);
```

g. Updating VIEW:
    i. This `UPDATE` statement will change data in both view as well as table
    ii. This query is considered as if we are updating the data on main table

```
SELECT * FROM employees;
UPDATE sal_emp SET FIRST_NAME="John"
WHERE SALARY=24000 AND DEPARTMENT_ID=90 AND FIRST_NAME="Steven";
SELECT * FROM employees;
```

    iii. **Note:** Updating the View may not always result in update to main table, so in order to update anything in view table, you need to update main table

h. Altering VIEW
    i. The schema / structure of VIEW can be altered using ALTER command
    ii. This means that columns can be added or removed on the applied condition in the view
    iii. This means that modification will be made on VIEW table
    iv. Example:

```
SELECT FIRST_NAME, LAST_NAME, SALARY, HIRE_DATE FROM employees
WHERE SALARY>5000;
```

i. Deleting record from VIEW:

```
SELECT * FROM employees; # show record of John in table employees
DELETE FROM sal_emp WHERE SALARY=24000 AND DEPARTMENT_ID=90 AND
FIRST_NAME='John'; # deletes record of John with salary 24000, dept
id of 90
SELECT * FROM employees; # does not show record of John
```

j. Dropping view:

```
SELECT * FROM sal_emp; #  shows view sal_emp
DROP VIEW sal_emp; # drops view sal_emp
SELECT * FROM sal_emp; # throws error as sal_emp does not exist
anymore
```

→ Random Tips:

a. The `SELECT` column is used to select required data from database
b. The `FROM` clause is used to specify the source from where the data has to be fetched
c. The `WHERE` condition is used to filter the records, the row will be filtered according to comparison between result of subquery
d. The `GROUP BY` clause is used to group rows having similar values
e. The `HAVING` clause is used to filter based on specified condition, filtering the group on basis of result of subquery

f. `IN` operator is faster than `ANY` operator, but `ANY` is is more powerful than `IN` operator

→