LET'S

UNDERSTAND THE DIFFERENCE BETWEEN VIEW VS CTE(COMMON TABLE EXPRESSION) VS TEMP TABLES VS SUB QUERY IN SQL

VIEW

- A VIEW IS A VIRTUAL TABLE THAT IS CREATED BASED ON THE DATA FROM ONE OR MORE TABLES
- VIEW STORES QUERY INSTEAD OF RESULT
- VIEW IS GOOD FOR SMALL DATASETS
- VIEWS CAN BE USED TO HIDE SENSITIVE DATA, AND TO PROVIDE SECURITY.
- YOU COULD CREATE A VIEW THAT JOINS SEVERAL TABLES TOGETHER TO CREATE A SINGLE TABLE THAT IS EASIER TO QUERY.

  (OR)

  YOU COULD CREATE A VIEW THAT ONLY SHOWS CERTAIN COLUMNS OF A TABLE.

EG. SUPPOSE I WANT ALL THE CUSTOMERS FROM “BHARAT” COUNTRY

CUSTOMERS TABLE

id	cus_name	cus_email	country
1	A	A@mail	BHARAT
2	B	B@mail	U.S.
3	C	C@mail	U.S.
4	D	D@mail	BHARAT

```
CREATE VIEW BHARAT_CUSTOMERS AS
SELECT ID,   CUS_NAME, CUS_EMAIL
FROM CUSTOMERS
WHERE COUNTRY = "BHARAT"
```

Here Instead Of Writing Query Again & Again For Particular Country I Will Create a VIEW

CTE (COMMON TABLE EXPRESSION)

- A CTE IS A NAMED TEMPORARY RESULT SET THAT IS CREATED WITHIN A SINGLE QUERY.
- IT IS SIMILAR TO A VIEW, BUT IT IS NOT STORED AS A PERMANENT OBJECT.
- CTES CAN BE RECURSIVE, WHICH MEANS THAT THEY CAN CALL THEMSELVES. THIS CAN BE USED TO PERFORM COMPLEX QUERIES THAT WOULD BE DIFFICULT OR IMPOSSIBLE TO DO WITH A SINGLE QUERY.
- CTES CAN BE SLOWER THAN TEMP TABLES FOR SOME QUERIES. THIS IS BECAUSE CTES ARE PROCESSED EACH TIME THEY ARE REFERENCED IN THE QUERY. CTES CAN CONSUME MEMORY.
- IF YOU HAVE MILLIONS OF RECORDS THEN DONT GO FOR CTE CHOOSE TEMP TABLES AS CTE USES RAM WHICH WILL AFFECT THE PERFORMANCE. (AGAIN IT DEPENDS UPON PROJECT REQUIREMENTS AND THE DATA)

EG. SUPPOSE YOU HAVE A DATABASE WITH A TABLE CALLED EMPLOYEES THAT CONTAINS INFORMATION ABOUT EMPLOYEES, INCLUDING THEIR NAMES, DEPARTMENTS, AND SALARIES. YOU WANT TO CREATE A CTE TO CALCULATE THE AVERAGE SALARY FOR EACH DEPARTMENT AND THEN USE THIS CTE TO RETRIEVE THE EMPLOYEES WHO HAVE SALARIES ABOVE THE DEPARTMENTAL AVERAGE.

EMPLOYEES TABLE

EmployeeID	FirstName	LastName	Department	Salary
1	John	Doe	HR	50000
2	Jane	Smith	HR	52000
3	Bob	Johnson	IT	60000
4	Alice	Brown	IT	62000
5	Eva	Lee	Finance	55000

```
WITH DEPARTMENTAVGSALARY AS (
    SELECT
        DEPARTMENT, AVG(SALARY) AS
AVGSALARY
    FROM
        EMPLOYEES
    GROUP BY
        DEPARTMENT
)
```

```
SELECT
E.EMPLOYEEID, E.FIRSTNAME,
E.LASTNAME, E.DEPARTMENT, E.SALARY
FROM
EMPLOYEES E
INNER JOIN
DEPARTMENTAVGSALARY D
ON
E.DEPARTMENT = D.DEPARTMENT
WHERE
   E.SALARY > D.AVGSALARY;
```

We first stored employees average salary by department in cte in yellow box. Then joined yellow box i.e cte with Employees table which is written in green box

TEMPORARY TABLES

- A TEMPORARY TABLE IS A PHYSICAL TABLE THAT IS CREATED IN THE TEMPDB DATABASE.
- TEMPORARY TABLES ARE GOOD FOR LARGE DATASETS.
- THERE ARE TWO TYPES OF TEMPORARY TABLES IN SQL SERVER: LOCAL TEMPORARY TABLES AND GLOBAL TEMPORARY TABLES.
- LOCAL TEMPORARY TABLES ARE ONLY VISIBLE TO THE CURRENT SESSION AND ARE DELETED WHEN THE SESSION IS CLOSED.
- GLOBAL TEMPORARY TABLES ARE VISIBLE TO ALL SESSIONS AND ARE DELETED WHEN THE LAST SESSION THAT REFERENCES THEM IS CLOSED.

LOCAL TEMPORARAY TABLE

```
CREATE TABLE #TEMP_TABLE (
  ID INT,
  NAME VARCHAR(10)
);
```

This creates a local temporary table called #temp_table with two columns: id and name.

The # symbol is used to indicate that the table is temporary.

GLOBAL TEMPORARY TABLE

```
CREATE TABLE ##TEMP_TABLE (
ID INT,

NAME VARCHAR(255)
);
```

This creates a global temporary table called ##temp_table with two columns: id and name.

The ## symbol is used to indicate that the table is global.

SUB QUERY

- A SUBQUERY IN SQL IS A QUERY NESTED WITHIN ANOTHER QUERY.
- IT IS ALSO KNOWN AS AN INNER QUERY OR NESTED QUERY.
- SUBQUERIES ARE USED TO RETRIEVE DATA THAT WILL BE USED BY THE MAIN (OR OUTER) QUERY FOR FILTERING, COMPARISON, OR OTHER OPERATIONS.

Types of Subqueries:
Single-Row Subquery: Returns only one row of results.
Multi-Row Subquery: Returns multiple rows.
Correlated Subquery: The inner query depends on the outer query for its values.
Uncorrelated Subquery: The inner query runs independently of the outer query.

SINGLE-ROW SUBQUERY

```
SELECT employee_id, name, salary
FROM employees
WHERE salary > (SELECT AVG(salary)
FROM employees);
```

$\rightarrow$

Inner Query: (SELECT AVG(salary) FROM employees) calculates the average salary for all employees (returns a single value).

Outer Query: Retrieves the employee(s) with that hire date.

MULTI-ROW SUBQUERY

```
SELECT column_name FROM table_name
WHERE column_name [operator] (SELECT
column_name FROM table_name WHERE
condition);
```

$\rightarrow$

Inner Query: (SELECT department_id FROM departments WHERE location IN ('New York', 'Los Angeles')) returns a list of department IDs for locations 'New York' and 'Los Angeles'.

Outer Query: Retrieves employees whose department_id is in the list returned by the inner query.

CORRELATED SUBQUERY

```
SELECT e.employee_id, e.name, e.salary
FROM employees e WHERE e.salary > (
   SELECT MIN(e2.salary) FROM employees e2
   WHERE e2.department_id = e.department_id );
```

Inner Query: For each row, it calculates the minimum salary for that employee's department, referencing e.department_id.

Outer Query: For each employee, it compares their salary against the lowest salary (MIN) in the same department.

SUBQUERY IN THE FROM CLAUSE

```
SELECT department_id, total_salary FROM ( SELECT
department_id, SUM(salary) AS total_salary FROM
employees GROUP BY department_id) AS
department_salaries WHERE total_salary > 100000;
```

Inner Query: (SELECT department_id, SUM(salary) AS total_salary FROM employees GROUP BY department_id) calculates the total salary per department.

Outer Query: Filters the departments where the total salary exceeds 100,000.