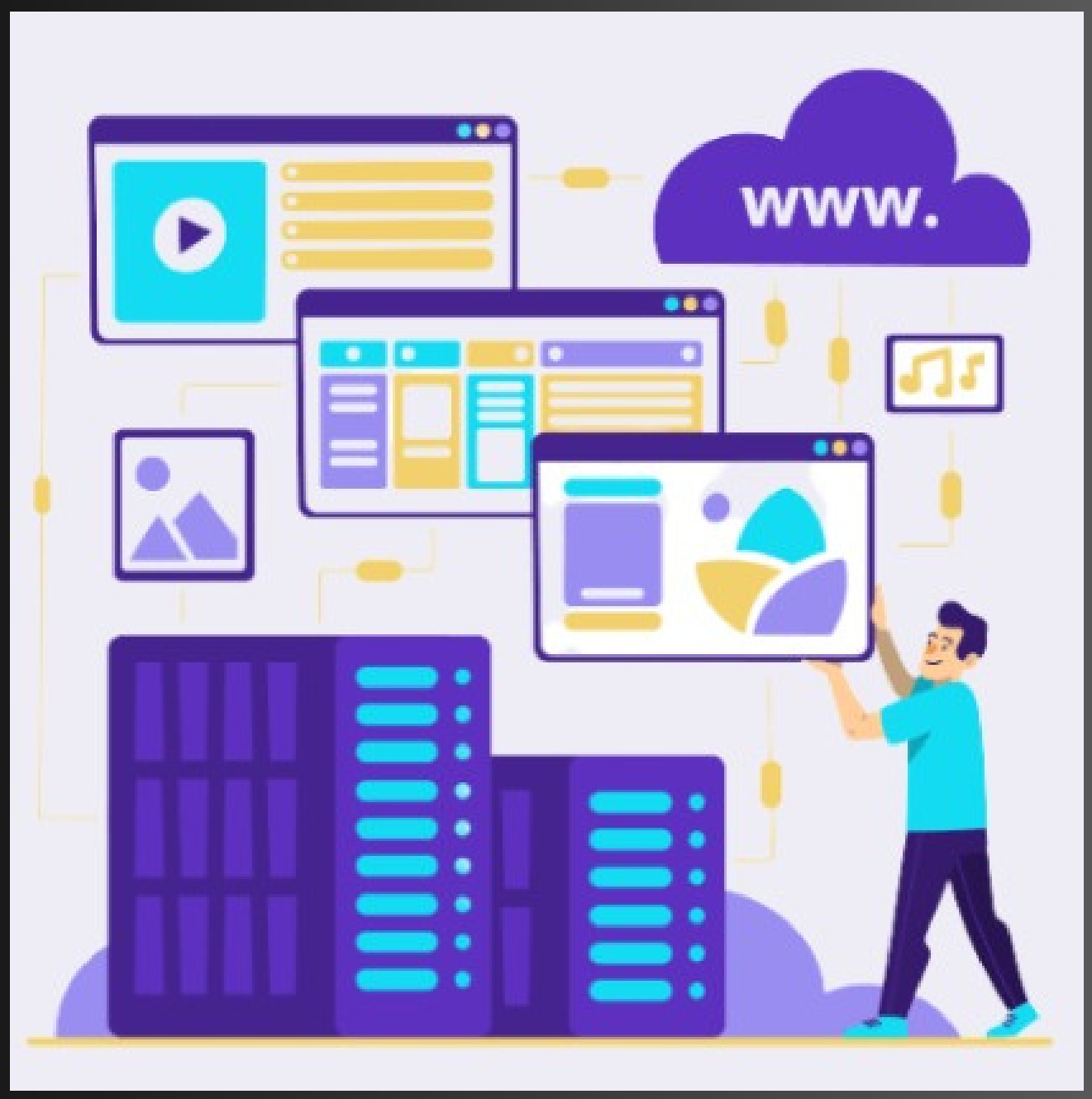
Subqueries and Derived Tables in SQL

Unlock the Power of Nested Queries for Advanced Data Analysis!



What is a Subquery?

A subquery (or nested query) is a query within another SQL query. It can return a single value, a list of values, or a complete table. Subqueries allow you to perform complex queries by leveraging the results of another query.

Types of Subqueries:

- Single-row subquery: Returns a single row.
- Multi-row subquery: Returns multiple rows.
- Scalar subquery: Returns a single value.

Example of a Simple Subquery

Employees Table

employee_id	first_name	last_name	department_id	salary
1	John	Doe	1	70000.00
2	Jane	Smith	2	80000.00
3	Alice	Johnson	1	90000.00
4	Bob	Brown	3	60000.00
5	Charlie	Davis	2	110000.00

departments table

department_id	department_name	location_id
1	Sales	1000
2	Marketing	1000
3	IT	2000

```
SELECT employee_id, first_name, last_name FROM
employees WHERE department_id = (SELECT
department_id FROM departments WHERE
department_name = 'Sales');
```

This query retrieves all employees who work in the Sales department by using a subquery to find the department_id.

Types of Subqueries - In Clauses

Subqueries can also be used with the IN clause to filter results based on a list of values returned by another query.

```
SELECT first_name, last_name FROM employees
WHERE department_id IN (SELECT department_id
FROM departments WHERE location_id = 1000);
```

This example retrieves employees from departments located in a specific location (identified by location_id).

first_name	last_name
John	Doe
Alice	Johnson
Jane	Smith
Charlie	Davis

Correlated Subqueries

A correlated subquery refers to a subquery that uses values from the outer query. It runs once for each row processed by the outer query.

```
SELECT e1.first_name, e1.last_name
FROM employees e1
WHERE e1.salary > (SELECT AVG(salary) FROM employees
e2 WHERE e1.department_id = e2.department_id);
```

This query returns employees whose salary is above the average salary of their respective departments. The subquery is correlated to the outer query.

first_name	last_name
Alice	Johnson
Charlie	Davis

Derived Tables

A derived table is a temporary table created in the FROM clause of a SQL statement. It is defined by a subquery and is used to simplify complex queries.

```
SELECT dept.department_name, avg_salaries.avg_salary
FROM (SELECT department_id, AVG(salary) AS avg_salary
FROM employees GROUP BY department_id) AS
avg_salaries JOIN departments dept ON
avg_salaries.department_id = dept.department_id;
```

In this query, we create a derived table called avg_salaries that computes the average salary for each department. We then join it with the departments table to get department names.

department_name	avg_salary
Sales	80000.000000
Marketing	95000.000000
IT	60000.000000

Why Use Subqueries and Derived Tables?

- Modularity: Break down complex queries into simpler parts.
- Clarity: Improve the readability of SQL statements.
- Encapsulation: Isolate logic that can be reused.

Practical Use Cases

- Sales Analysis: Finding products that sold above average prices.
- Employee Insights: Identifying employees with above-average salaries in their departments.
- Customer Segmentation: Analyzing customer behavior based on order patterns.

The best way to master subqueries and derived tables is through consistent practice! Experiment with your own queries and datasets to uncover deeper insights.