1. OOPS: Constructors (Initialization)

What is a Constructor?

A constructor is a special method that is called automatically when an object is created. Its primary purpose is to initialize the object's state.

- **Default Constructor:** If no constructor is defined, the compiler often provides a default one that initializes attributes to default values (0, null, etc.).
- **Parameterized Constructor:** Allows passing specific data at the time of object creation, making the object "ready for use" immediately.

Why use them?

- **Ensures Object Integrity:** An object shouldn't exist in an "invalid" state (e.g., a `BankAccount` without an `accountNumber`).
- **Concise Code:** Reduces the need for multiple "setter" calls after instantiation.

2. OOPS Scenario: Initializing User Profiles

Problem: In a social media app, when a user signs up, we need to ensure their `username` and `email` are set immediately.

Analysis:

1. **The Constructor Role:** By using a parameterized constructor `User(name, email)` , we force the creation of the object to include mandatory data.
2. **Real-world mapping:** Think of a constructor like a "Birth Certificate"—the moment the entity (Object) is born, its identifying details (State) are recorded.

3. Programming: Prefix Sum Technique

Problem: Given an array, answer multiple queries about the sum of elements between index i and j efficiently.

Approach 1: Brute Force

- For every query, loop from i to j and calculate the sum.
- **Time Complexity:** $O(Q \times N)$ where Q is the number of queries.

Approach 2: Prefix Sum (Optimized)

1. Precompute a `prefixSum` array where `prefixSum[i]` stores the sum of all elements from index 0 to i.
   - Formula: $prefixSum[i] = prefixSum[i-1] + arr[i]$
2. To find the sum of range $[L, R]$:

    - Formula: $Sum(L, R) = prefixSum[R] - prefixSum[L-1]$ (Handle $L = 0$ case separately).

- **Time Complexity:** - Precomputation: $O(N)$
    - Query: $O(1)$
- **Total:** $O(N + Q)$, which is significantly faster for many queries.

4. SQL: RIGHT JOIN (RIGHT OUTER JOIN)

The `RIGHT JOIN` returns **all** records from the right table and the **matched** records from the left table.

The Logic: "The Inclusive Right"

If there is no match, the result is `NULL` on the left side. It is the mirror image of a `LEFT JOIN`.

Syntax

```
SELECT employees.name, departments.dept_name
FROM employees
RIGHT JOIN departments ON employees.dept_id = departments.dept_id;
```

Practical Use Case:

- **Identifying Empty Entities:** In the query above, a `RIGHT JOIN` on `departments` would show all departments, including those that currently have **zero employees** assigned to them.

Summary Table

Topic	Focus	Key Takeaway
OOPS	Constructors	Automates initialization; ensures objects start in a valid state.
DSA	Prefix Sum	Converts $O(N)$ range queries into $O(1)$ after $O(N)$ preprocessing.
SQL	RIGHT JOIN	Useful for listing all categories/departments even if they are empty.

Every day you build a stronger foundation. See you for Day 4!