Problem Title:

Subarray Sum Equals K

You are working as a data analyst for a fitness app. Every day, users log the number of calories burned. Your task is to find how many continuous days (subarrays) exist where the total calories burned equals a certain target `k`. This helps the app identify specific goal-based streaks.

Problem Statement:

Given an integer array `nums` representing the calories burned each day, and an integer `k` representing a target calorie goal, return the total number of **continuous subarrays** whose sum is exactly equal to `k`.

Input Format:

- An integer `n` — the number of days.
- An array `nums` of `n` integers — calories burned each day.
- An integer `k` — the target calorie burn.

Output Format:

- A single integer — the total number of continuous subarrays whose sum equals `k`.

Constraints:

- `1 <= nums.length <= 2 * 10^4`
- `-1000 <= nums[i] <= 1000`
- `-10^7 <= k <= 10^7`

Example 1:

Input:

```
nums = [1, 2, 3]
k = 3
```

Output:

```
2
```

Explanation:
There are two subarrays that sum to 3:

- [1, 2]
- [3]

Example 2:

Input:

```
nums = [1, 1, 1]
k = 2
```

Output:

```
2
```

Explanation:
Subarrays [1,1] at indices (0,1) and (1,2) both sum to 2.

Approach:

✅ Efficient Approach — Prefix Sum + HashMap (Time: O(n), Space: O(n))

1. Use a variable `sum` to track the running sum.
2. Use a `HashMap` to store how many times each prefix sum has occurred.
3. For each element:
    - Add it to the running `sum`.
    - Check if `sum - k` is already in the map — if yes, add the count to the result.
    - Update the map with the new `sum`.

Key Insight:

If:

```
sum(i to j) = sum(j) - sum(i-1)
```

Then for subarrays ending at `j`, we check if `(sum - k)` was seen before.

🔗 Practice Links:

- Leetcode 560 – Subarray Sum Equals K
- GeeksforGeeks Variation (Count subarrays with given sum)

🎥 Video Explanation:

- Subarray Sum Equals K (Prefix Sum + HashMap) - Java | Youtube