Understanding Cosine Similarity and Its Use in Vector Databases

1. Idea
Each object (text, image, recipe, etc.) can be represented as a list of numbers called a **vector**. Cosine similarity measures how much two vectors point in the same direction in space. If they point the same way → similarity = 1. If they are perpendicular → 0. If they go opposite → -1.

2. The formula
Cosine Similarity(A, B) = $(A \cdot B) / (||A|| \times ||B||)$

Dot product ($A \cdot B$): multiply each component and add them up.
Norm ($||A||$): the length of the vector, calculated as the square root of the sum of squares.
Division: divides the dot product by the product of the lengths to normalize.

Example:
$A = [1, 2, 3]$, $B = [2, 4, 6]$
$A \cdot B = 1\times2 + 2\times4 + 3\times6 = 28$
$||A|| = \sqrt{(1^2 + 2^2 + 3^2)} = \sqrt{14} \approx 3.74$
$||B|| = \sqrt{(2^2 + 4^2 + 6^2)} = \sqrt{56} \approx 7.48$
Cosine similarity = $28 / (3.74 \times 7.48) = 1$ → perfectly aligned.

Intuitive meaning:
1 → Vectors go in the same direction (very similar)
0 → Unrelated vectors
-1 → Opposite directions (contradictory)

5. Use in Vector Databases
When data is stored as embeddings (numeric vectors), cosine similarity finds the most semantically similar items.

Steps:
1. Convert the query into a vector (embedding).
2. Compare it to all stored vectors using cosine similarity.
3. Return items with the highest similarity values.

SQL Example (Postgres + pgvector):
SELECT id, embedding <#> query_embedding AS distance FROM items ORDER BY embedding <#> query_embedding LIMIT 5;
Here, “<#>” is the cosine distance (1 - cosine similarity). Lower distance → higher similarity.

6. Advantages
- Ignores magnitude; focuses on direction.
- Works well for normalized embeddings.
- Ideal for semantic search in text, image, or recommendation systems.