**HASHING: DEEP DIVE**

**🔸 What is Hashing?**

Hashing is the process of converting an input (like a string, integer, or object) into a fixed-size numerical value called a **hash code** using a **hash function**.

**🔸 Example:**

print(hash("apple")) # Example: 5832674852391975120 (value varies each run)

print(hash(12345)) # Example: 12345

💡 In Python, hash() provides a unique ID (may vary between runs for security reasons).

**HASH TABLE: DEEP DIVE**

**🔸 What is a Hash Table?**

A data structure that uses a **hash function** to compute the **index** into an array of buckets or slots, from which the desired value can be found.

**🔸 Why Used in Data Engineering?**

* **ETL Deduplication**: Removing duplicate customer IDs or IPs
* **Streaming frequency count**: Counting how often a keyword appears in logs
* **Join optimization**: Hash joins in distributed processing like Spark, BigQuery

**🔸 Python Implementation:**

Python’s built-in dict is a hash table under the hood.

person = {'name': 'Alice', 'age': 30}

print(person['age']) # Fast O(1) lookup

**hashing ---**

* hash value calculation and putting at that index
* hash function i % length
* now problem arises same index ----collision
* to solve this problem 2 methods
* closed addressing
* rehashing (at the same position)
* creating like a linked list adding new element to node
* and other is by converting a ll to tree
* another is closed addressing
* linear probing – inserting element to next index if first not empty h(i) % length –
* quadratic probing

**MAIN FUNCTION OF HASHING TO FETCH AT TIME COMPLEXITY O(1)**

Why use enumerate()?

When you're looping over something and need both the index and the value, enumerate() is cleaner and more readable than using a manual index with range(len(...)).

What is a Bloom Filter?

A Bloom Filter is a space-efficient probabilistic data structure used to test whether an element is possibly in a set or definitely not.