📘 Day 8 Learning Log

🔸 Aptitude – Permutations

Focus: Arrangement logic, factorials, and position-oriented reasoning.

Key Formulas:

- **Factorial:** $n! = n \times (n-1) \times \cdots \times 1$
- **Permutation (Arrangement):** The number of ways to arrange r items from a set of n distinct items is:

$$P(n, r) = \frac{n!}{(n-r)!}$$

- **Distinguishable Permutations:** If there are n objects where p are alike, q are alike, etc.:

$$Ways = \frac{n!}{p! \cdot q! \dots}$$

🔸 Programming – Remove Duplicates from a List

Task: Implement logic to remove duplicate elements from a list.

Python Implementation

There are two common ways to handle this:

1. Using Sets (Fastest & Easiest) *Note: This does not preserve order.*

```
original_list = [1, 2, 2, 3, 4, 4, 5]
unique_list = list(set(original_list))
print(unique_list) # Output: [1, 2, 3, 4, 5]
```

2. Manual Iteration (Preserves Order)

```
def remove_duplicates(lst):
    unique_list = []
    seen = set()
    for item in lst:
        if item not in seen:
            unique_list.append(item)
            seen.add(item)
    return unique_list

print(remove_duplicates([1, 2, 2, 3, 4, 4, 5]))
```

🔸 Concept – Python Tuples vs. Sets

Focus: Immutability vs. Uniqueness.

Feature	Tuple `()`	Set `{}`
Syntax	`my_tuple = (1, 2, 3)`	`my_set = {1, 2, 3}`
Mutability	**Immutable** (Cannot change after creation)	**Mutable** (Can add/remove items)
Duplicates	Allowed	**Not Allowed** (Automatically removed)
Order	Ordered (Indexable)	Unordered (Cannot use index)
Use Case	Fixed collections (e.g., coordinates `(x, y)`)	Mathematical operations, checking existence

🔸 C/C++ Concept – Array Comparison

Focus: Comparing arrays element-by-element.

Unlike high-level languages where `arr1 == arr2` might work, C/C++ requires manual checking because arrays act as pointers to memory addresses.

C++ Example:

```
bool areArraysEqual(int arr1[], int arr2[], int size1, int size2) {
    // 1. Check if sizes are different
    if (size1 != size2) return false;

    // 2. Compare element by element
    for (int i = 0; i < size1; i++) {
        if (arr1[i] != arr2[i]) {
            return false;
        }
    }
    return true;
}
```

🔸 SQL – DISTINCT Keyword

Focus: Eliminating duplicate records in queries.

Syntax:

```
SELECT DISTINCT column_name1, column_name2
FROM table_name;
```

Example: If a table `Employees` has multiple entries for the same `Department`:

```
-- Gets a list of unique departments
SELECT DISTINCT Department
FROM Employees;
```