**Exercise-3 Sorting Customer Orders**

**Understanding Sorting Algorithms:**

**Bubble Sort:**

* Algorithm: Repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.
* It is simple to implement but inefficient for large datasets.

**Insertion Sort:**

* Algorithm: Builds the final sorted array one item at a time, with the assumption that the first element is already sorted. It picks the next element and inserts it into the correct position.
* It is efficient for small datasets or nearly sorted data.

**Quick Sort:**

* Algorithm: Uses a divide-and-conquer strategy. It selects a 'pivot' element and partitions the array into two sub-arrays: elements less than the pivot and elements greater than the
* It is generally faster in practice for large datasets and preferred over Bubble Sort.

**Merge Sort:**

* Algorithm: It is also a divide-and-conquer algorithm. It divides the array into two halves, recursively sorts them, and then merges the sorted halves.
* Stable sort with consistent O (n log n) performance, but requires additional space for the temporary arrays.

**Analysis:**

**Time Complexity:**

Bubble Sort:

* Best Case: O(n)
* Average Case: O(n²)
* Worst Case: O(n²)

Quick Sort:

* Best Case: O (n log n)
* Average Case: O (n log n)
* Worst Case: O(n²)
* Quick Sort is generally preferred Over Bubble sort due to its O (n log n) average-case complexity, making it suitable for large datasets. However, it requires careful pivot selection to avoid worst-case scenarios.