**Exercise 3: Sorting Customer Orders**

**1. Understand Sorting Algorithms:**

* **Bubble Sort:**
  + Repeatedly compares adjacent elements and swaps them if they are in the wrong order.
  + Simple to implement but inefficient for large datasets.
* **Insertion Sort:**
  + Builds the sorted array one item at a time by comparing each new element with the already sorted ones and inserting it at the right position.
  + Works efficiently on small or nearly sorted datasets.
* **Quick Sort:**
  + A divide-and-conquer algorithm that selects a pivot, partitions the array around the pivot, and recursively sorts the subarrays.
  + Performs well in practice and is widely used for sorting large datasets.
* **Merge Sort:**
  + Also a divide-and-conquer algorithm that splits the array into halves, recursively sorts them, and then merges them.
  + Guarantees O(n log n) performance but uses additional space for merging.

**4. Analysis:**

* **Time Complexity Comparison:**
  + **Bubble Sort:** Worst and average case is O(n²); Best case is O(n) if the array is already sorted.
  + **Quick Sort:** Average and best case is O(n log n); Worst case is O(n²), but this is rare with good pivot selection.
* **Why Quick Sort is Preferred:**
  + Quick Sort is significantly faster than Bubble Sort for large datasets.
  + It has better average-case performance due to efficient partitioning.
  + Bubble Sort is only suitable for educational purposes or very small datasets due to its poor performance.