**Exercise 3: Sorting Customer Orders**

**1. Understand Sorting Algorithms**

* **Bubble Sort** is a simple comparison-based algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. It's easy to understand but inefficient for large datasets.
* **Insertion Sort** builds the final sorted array one item at a time. It is efficient for small or nearly sorted datasets but not suitable for large data.
* **Quick Sort** is a divide-and-conquer algorithm. It picks a pivot and partitions the array around the pivot. It's very efficient for large datasets and is widely used in practice.
* **Merge Sort** also follows divide-and-conquer. It divides the list into halves, recursively sorts each half, and merges them. It guarantees O(n log n) time but uses more memory than Quick Sort.

**2. Analysis**

**Bubble Sort** has a time complexity of **O(n²)** in both worst and average cases. This makes it very inefficient when sorting a large number of orders. It performs repeated passes over the entire list, swapping adjacent elements.

**Quick Sort**, on the other hand, has an average time complexity of **O(n log n)** and performs much better on large datasets. Although its worst-case time complexity is O(n²), this rarely occurs in practice if a good pivot is chosen or randomized.

Quick Sort is generally preferred because it is significantly faster than Bubble Sort in most scenarios. It is more efficient in terms of the number of comparisons and swaps, making it ideal for performance-critical applications like sorting e-commerce orders.

**3. Output**

