**Exercise 3: Sorting Customer Orders**

**1. Understand Sorting Algorithms**

**1.1 Different Sorting Algorithms**

* **Bubble Sort**:
  + Repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.
  + Time Complexity: O(n^2) in the average and worst cases.
  + Space Complexity: O(1)
  + Best Case: O(n) when the array is already sorted.
* **Insertion Sort**:
  + Builds the sorted array one element at a time by repeatedly picking the next element and inserting it into its correct position.
  + Time Complexity: O(n^2) in the average and worst cases.
  + Space Complexity: O(1)
  + Best Case: O(n) when the array is already sorted.
* **Quick Sort**:
  + Divides the list into smaller sub-lists based on a pivot element and recursively sorts the sub-lists.
  + Time Complexity: O(n log n) on average, but O(n^2) in the worst case.
  + Space Complexity: O(log n) due to the recursive stack.
  + Best Case: O(n log n).
* **Merge Sort**:
  + Divides the list into smaller sub-lists, sorts them, and then merges them back together.
  + Time Complexity: O(n log n) in all cases.
  + Space Complexity: O(n) due to the temporary arrays used for merging.

**4. Analysis**

**Step 4.1: Compare the Performance (Time Complexity) of Bubble Sort and Quick Sort**

* **Bubble Sort**:
  + Best Case: O(n)
  + Average Case: O(n^2)
  + Worst Case: O(n^2)
  + Bubble Sort performs poorly on large datasets due to its quadratic time complexity.
* **Quick Sort**:
  + Best Case: O(n log n)
  + Average Case: O(n log n)
  + Worst Case: O(n^2) (rare, occurs when the smallest or largest element is always chosen as the pivot)
  + Quick Sort generally performs much better on large datasets due to its logarithmic average time complexity.

**Step 4.2: Discuss Why Quick Sort is Generally Preferred Over Bubble Sort**

* **Efficiency**: Quick Sort is significantly more efficient for large datasets due to its O(n log n) average time complexity compared to Bubble Sort's O(n^2).
* **Performance**: Quick Sort's divide-and-conquer approach makes it faster and more scalable for large datasets.
* **Flexibility**: Quick Sort can be optimized with different pivot selection strategies to avoid the worst-case scenario.