Based on the result, the conclusion is that the insertion sort performs well on small or partially ordered datasets. However, for large datasets or completely random datasets, the performance degrades significantly.

The following conclusion is group by these 4 different cases:

Random Arrays: The sorting time increases as the size of the array increases. This is expected because insertion sort has a time complexity of O(n2) in the average and worst-case scenarios, where n is the number of elements in the array. As the array size doubles, the sorting time increases significantly, indicating that insertion sort can become inefficient for large random arrays.

Ordered Arrays: The sorting time for completely ordered arrays does not always show a consistent pattern compared to random arrays. In some cases, the time slightly increases as the array size grows, but it remains relatively low. This is because when the array is already ordered, the inner loop of the insertion sort algorithm does minimal work, showcasing the best-case scenario time complexity of O(n).

Partially Ordered Arrays: The sorting time for partially ordered arrays tends to be lower than for completely random arrays but higher than for fully ordered arrays. This demonstrates that insertion sort can be more efficient if the array is partially sorted.

Reverse Ordered Arrays: The sorting time for reverse ordered arrays shows that insertion sort needs to do more work compared to ordered and partially ordered arrays. This is because each element needs to be moved all the way to the beginning of the array, making it a relatively heavy operation. So, it consumes the most time.