1) 3 way merge sort is $T(n) = 3T(n/3) + \theta(n)$. Each array is of length n/3 and there are 3 of them. $\theta(n)$ is cost of merging 3 sub collections into 1. From the master theorem where $a = 3, b = 3, k = 1, p = 0$ this algorithm is $\theta(nlogn)$.

2)
With alternating instances of worst case and evenly split, we can write the recurrence with these 2 equations. T_g is notation for the good case and T_w is notation for worst case.
$T_g(n) = 2T_w(n/2) + \theta(n)$
$T_w(n) = T_g(n-1) + \theta(n)$
We substitute T_w into the first equation and get:
$T_g(n) = 2(T_g(n/2-1) + \theta(n/2)) + \theta(n)$
$T_g(n) = 2T_g(n/2-1) + 2\theta(n/2) + \theta(n)$
$T_g(n) = 2T_g(n/2-1) + \theta(n)$
$= \theta(nlogn)$
Also substitute T_g we get:
$T_w(n) = 2T((n-1)/2) + \theta(n-1) + \theta(n)$
$T_w(n) = 2T(n/2 - 1/2) + \theta(n)$
$= \theta(nlogn)$

So under these circumstances the overall complexity of the quicksort is $\theta(nlogn)$.

3)
Since the transaction number is sorted and transaction number and transaction dates tend to correlate (but not foolproof), that implies transaction date is almost sorted. For a list where the elements are nearly sorted, quicksort is bad algorithm (assuming choosing first element as pivot) for this case as it tends to $\theta(n^2)$. Insertion sort is the best algorithm for when a list is nearly sorted as it tends towards $\theta(n)$ for nearly sorted. Therefore, we choose insertion sort for this problem.

4) We have n sets of size k arrays that are sorted:
A merge(k,k) is 2k
k k k k k k k k k – 2k
2k k k k k k k ——3k
3k k k k k k ———4k
(n-1)k——————nk
2k+3k+4k+...+nk = $(1/2)(n^2 + n - 2)k = \theta(n^2k)$
Therefore complexity is $\theta(n^2k)$.

5) We have n sets of size k arrays that are sorted:
A merge(k,k) is 2k
k k k k k k k k k – 2k
2k 2k 2k 2k 2k —-2k(n/2) = kn
4k 4k 4k 4k ——-4k(n/4) = kn
8k 8k 8k 8k ——-8k(n/8) = kn
(n/2)k, (n/2)k,..——(n/2)2*k = kn
So it's the summation of kn, $nlog_2(n)$ number of times.
Therefore complexity is $\theta(knlog_2(n))$.