7. Financial Forecasting

Understand Recursion

Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem. This approach can simplify the implementation of certain algorithms by breaking down complex problems into simpler subproblems.

Concept of Recursion

1. **Base Case**: Every recursive function must have a base case that defines when the recursion should stop. This prevents infinite recursion and eventually allows the function to return a value.
2. **Recursive Case**: This part of the function calls itself with modified arguments to progress towards the base case.
3. **Simplicity**: Recursion can often lead to cleaner and more readable code by eliminating the need for explicit looping constructs. Problems that involve hierarchical data structures (like trees) or repetitive calculations (like factorial, Fibonacci series, etc.) are often more naturally expressed with recursion.

Analysis

Analysis of Time Complexity

- The time complexity of a simple recursive function that calculates the future value based on the number of years n is O(n) because the function makes n
  n recursive calls until it reaches the base case.

Optimising the Recursive Solution

1. **Memoization**:
   - To avoid excessive computation in recursive algorithms, we can store the results of previous computations in a data structure (like an array or a hash map). This technique is known as memoization. By caching results, we can prevent redundant calculations for the same inputs.
   - For example, if the future value for a certain number of years has already been computed, the algorithm can simply retrieve it from the cache instead of recalculating it.
2. **Tail Recursion**:
   - If applicable, convert the recursive function into a tail-recursive function. Tail recursion allows the compiler to optimize the recursive calls, potentially reducing the overhead of maintaining multiple stack frames.

- A tail-recursive version of a function can reuse the current function's stack frame for the recursive call, which can help improve performance and avoid stack overflow errors for deep recursions.