Nama	: Angga Darmawan
Batch	: 25
Penempatan	: Jamkrindo, Jakarta
Role	: DBS

Buatlah DB Schema untuk basic chat messaging system dengan fitur:

1. 1 on 1 messaging
2. Group messaging, di dalam group ada role:
    1. Admin (yang bikin group, dan bisa menambah atau menghapus member)
    2. Member (yang di add admin ke dalam group, hanya bisa mengirim message)
3. Status (read/sent/pending)
4. Sharing images/files

1 nomor handphone digunakan untuk membuat 1 akun

Dibolehkan juga untuk menambahkan skenario/kondisi yang bisa ditambahkan ke dalam schema dari penjelasan di atas.

Anda bebas menentukan, boleh menggunakan SQL atau noSQL. Dan jelaskan kenapa anda memilih pilihan tersebut

Saya menggunakan SQL karena menyediakan cara yang terstruktur dan efisien untuk mengelola data dalam aplikasi kompleks. Dengan SQL, kita bisa memastikan data tersimpan secara terorganisir, menghindari duplikasi (misalnya, nomor telepon unik), menjaga integritas dan keamanan data, serta memudahkan dalam querying dan manajemen relasi antar data (seperti pengguna, pesan, dan grup). Selain itu, SQL mendukung skalabilitas dan kinerja tinggi, yang penting untuk aplikasi yang menangani banyak pengguna dan data.

Anda diberikan array angka integer positif yang mewakilkan harga stok dalam beragam hari (setiap index array mencerminkan hari yang berbeda)
Anda juga diberikan integer (i), yang mewakilkan jumlah transaksi yang boleh dilakukan
Satu transaksi terdiri dari pembelian stok pada suatu hari dan menjualnya di kemudian hari

Tulis sebuah fungsi yang menghasilkan keuntungan terbanyak yang dapat dilakukan melalui pembelian dan penjualan stok, bergantung dari (i) transaksi.

Note: Anda hanya dapat memegang satu share stok pada satu saat; jadi anda tidak dapat membeli lebih dari satu share stok pada hari manapun, dan anda tidak dapat membeli sebuah share dari stok manapun kalau anda masih memegang share lain. Anda juga tidak perlu menggunakan semua (i) transaksi yang diperbolehkan

```
function maxProfit(prices, k) {
  const n = prices.length;
  if (n === 0 || k === 0) return 0;
  const dp = Array.from({ length: k + 1 }, () => Array(n).fill(0));
  for (let i = 1; i <= k; i++) {
```

```
        let maxDiff = -prices[0];
        for (let j = 1; j < n; j++) {
            dp[i][j] = Math.max(dp[i][j - 1], prices[j] + maxDiff);
            maxDiff = Math.max(maxDiff, dp[i - 1][j] - prices[j]);
        }
    }
    return dp[k][n - 1];
}

const prices = [3, 2, 6, 5, 0, 3];
const k = 2;
const maxProfitValue = maxProfit(prices, k);
console.log("Keuntungan maksimum: " + maxProfitValue);
```