```sql
SELECT
    p.name,
    p.price,
    SUM(po.quantity) AS quantity,
    SUM(po.quantity * p.price) AS total_price
FROM
    products p
JOIN
    products_orders po ON p.id = po.product_id
JOIN
    orders o ON o.id = po.order_id AND o.status = 'Completed'
GROUP BY
    p.id, p.name, p.price
ORDER BY
    quantity DESC, total_price DESC;
```

```java
public class ChannelRating {

    private static final int MOD = 1000000007;

    public static int getChannelRating(int n, int[] views) {
        // Compute the prefix XORs
        int[] prefixXOR = new int[n + 1];
        for (int i = 1; i <= n; i++) {
            prefixXOR[i] = prefixXOR[i - 1] ^ views[i - 1];
        }

        int specialCount = 0;

        // Iterate through all subarrays of length >= 3
        for (int i = 0; i < n; i++) {
            for (int j = i + 2; j < n; j++) {
                int borderXOR = views[i] ^ views[j];
                int nonBorderXOR = prefixXOR[j] ^ prefixXOR[i + 1];

                if (borderXOR == nonBorderXOR) {
                    specialCount++;
                    if (specialCount >= MOD) {
                        specialCount -= MOD;
                    }
                }
            }
        }

        return specialCount;
    }

    public static void main(String[] args) {
```

```
        int n = 4;
        int[] views = {0, 3, 6, 5};
        System.out.println(getChannelRating(n, views)); // Expected
output: 2
    }
}
```

```
function getChannelRating(views) {

    console.log(views);

    const MOD = 1e9 + 7;
    const n = views.length;
    let rating = 0;

    if (views.every(view => view === 0)) {
        for (let i = 0; i < n; i++) {
            rating = (rating + Math.max(n - i - 2, 0)) % MOD;
        }
        return rating;
    }

    const prefixXOR = Array(n + 1).fill(0);

    for (let i = 0; i < n; i++) {
        prefixXOR[i + 1] = prefixXOR[i] ^ views[i];
    }

    const map = new Map();
    let i = 0;
    while (i <= n) {
        let j = i + 1;
        while (j < n && views[j] === views[i]) {
            j++;
        }
        if (j - i > 2) {
            const len = j - i;
            rating = (rating + ((len - 2) * (len - 1)) / 2) % MOD;
            i = j;
        } else {
            if (map.has(prefixXOR[i])) {
                const indices = map.get(prefixXOR[i]);
                for (const index of indices) {
                    if (i - index > 2) {
                        rating = (rating + 1) % MOD;
                    }
                }
                indices.push(i);
            } else {
                map.set(prefixXOR[i], [i]);
```

```
            }
            i++;
        }
    }

    return rating;
}
```

```
import java.util.*;

public class ByteDanceAlgorithm {

    public static int[] minimumCommonInteger(int n, int[] data) {
        int[] result = new int[n];
        Map<Integer, List<Integer>> indexMap = new HashMap<>();

        // Scan the array to record the indices at which each number
appears
        for (int i = 0; i < n; i++) {
            indexMap.computeIfAbsent(data[i], k -> new ArrayList<>
()).add(i);
        }

        for (int k = 1; k <= n; k++) {
            int commonMin = Integer.MAX_VALUE;

            for (Map.Entry<Integer, List<Integer>> entry :
indexMap.entrySet()) {
                int count = 0;
                List<Integer> indices = entry.getValue();

                for (int i = 0; i < indices.size() - 1; i++) {
                    if (indices.get(i + 1) - indices.get(i) < k) {
                        count++;
                    }
                }

                // Check if the number appears in every subarray of size k
                if (count == n - k + 1) {
                    commonMin = Math.min(commonMin, entry.getKey());
                }
            }

            result[k - 1] = commonMin == Integer.MAX_VALUE ? -1 :
commonMin;
        }

        return result;
    }

    public static void main(String[] args) {
```

```
        int n = 5;
        int[] data = {4, 3, 3, 4, 2};
        int[] result = minimumCommonInteger(n, data);

        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```