1) Given an array of objects representing employees, each object containing a `name` and `age` property, write a function that sorts the array by the `age` property in ascending order.

```
const employees = [
                    { name: 'John', age: 28 },
                    { name: 'Anna', age: 22 },
                    { name: 'Mike', age: 32 },
                ];
```

Output:

[{ name: 'Anna', age: 22 }, { name: 'John', age: 28 }, { name: 'Mike', age: 32 }]

2) You are given an array of numbers. Write a function that groups the numbers into two categories: odd and even, and returns an object where the keys are "odd" and "even", and the values are arrays containing the respective numbers

```
const nums = [1, 2, 3, 4, 5, 6];
```

Output: { odd: [1, 3, 5], even: [2, 4, 6] }

3) You have an array of objects, each containing an `id` and `name` property. Some objects may have the same `id` value but different `name` values. Write a function that removes the duplicate objects based on the `id` property and returns a new array with only unique `id` values.

```
const data = [ { id: 1, name: 'A' }, { id: 2, name: 'B' }, { id: 1, name: 'C' }, ];
```

Output: [{ id: 1, name: 'A' }, { id: 2, name: 'B' }];

4) Find the most frequent element in an array.

```
const arr = [1, 2, 2, 3, 3, 3];
```

Output: 3;

5) Find common elements in two arrays.

```
const arr1 = [1, 2, 3]; const arr2 = [2, 3, 4];
```

Output : [2, 3];

6) Transform array of objects into a key-value map.

const arr = [{ id: 1, name: 'A' }, { id: 2, name: 'B' },];

Output: { 1: 'A', 2: 'B' }

7) Get array elements that only appear once.

const arr = [1, 2, 2, 3, 4, 4, 5];

Output: [1, 3, 5]

8) Convert an object into an array of [key, value] pairs.

const obj = { a: 1, b: 2 };

Output: [['a', 1], ['b', 2]]

9) Filter keys in an object.

const obj = { a: 1, b: 2, c: 3 };

Output: { a: 1, b: 2 }

10) Given three values: an array `arr = [1, 2, 3, 5]`, an array `b = [4, 7]`, and a single number `c = 6`, merge them together into a single array `[1, 2, 3, 4, 5, 6, 7]`, without using the `sort()` or `concat()` methods, and ensuring no duplicates are present.

Output: [1, 2, 3, 5, 4, 7, 6]