# Sorted Orders

You would be creating a function that will sort the response from an endpoint in the hierarchy specified below in **descending order**: `$total > $metric_value_per_property`

## JSON Response Schema

```
[{
    "category": "string | $total",
    "itemName": "string | $total",
    "clicks": "number",
    "quantity": "number",
    "amount": "number"
}]
```

## Explanation

The orders data being returned from the Orders endpoint matches the response schema above. The goal of this exercise is to sort the JSON response in **descending order** such that the total value ($total) of each property (category, itemName) takes precedence over its individual metric values. As an example:

Given a sample field in the array with values as follows:

```
{
    "category": "$total",
    "itemName": "$total",
    "clicks": "160",
    "quantity": "18",
    "amount": "702"
}
```

This would take precendence over:

```
{
    "category": "Household",
    "itemName": "$total",
    "clicks": "105",
    "quantity": "50",
    "amount": "5000"
}
```

followed by individual items like:

```
{
    "category": "Household",
    "itemName": "Toothbrush",
    "clicks": "4",
    "quantity": "4",
    "amount": "12"
}
```

### Example Input

| Category | Item Name | Clicks | Quantity | Amount |
|---|---|---|---|---|
| Electronics | TV | 12 | 1 | 300 |
| Household | $total | 60 | 12 | 37 |
| Electronics | $total | 100 | 6 | 665 |
| Household | Fairy dish liquid | 44 | 5 | 10 |

| Category | Item Name | Clicks | Quantity | Amount |
|---|---|---|---|---|
| Electronics | Speakers | 43 | 3 | 240 |
| $total | $total | 160 | 18 | 702 |
| Electronics | Microwave | 4 | 1 | 85 |
| Household | Toothbrush | 4 | 4 | 12 |
| Household | Hand wash | 12 | 3 | 15 |
| Electronics | Toaster | 41 | 1 | 40 |

### Output Sorted by Amount metric

| Category | Item Name | Clicks | Quantity | Amount |
|---|---|---|---|---|
| $total | $total | 160 | 18 | 702 |
| Electronics | $total | 100 | 6 | 665 |
| Electronics | TV | 12 | 1 | 300 |
| Electronics | Speakers | 43 | 3 | 240 |
| Electronics | Microwave | 4 | 1 | 85 |
| Electronics | Toaster | 41 | 1 | 40 |
| Household | $total | 60 | 12 | 37 |
| Household | Hand wash | 12 | 3 | 15 |
| Household | Toothbrush | 4 | 4 | 12 |
| Household | Fairy dish liquid | 44 | 5 | 10 |

### Output Sorted by Quantity metric

| Category | Item Name | Clicks | Quantity | Amount |
|---|---|---|---|---|
| $total | $total | 160 | 18 | 702 |
| Household | $total | 60 | 12 | 37 |
| Household | Fairy dish liquid | 44 | 5 | 10 |
| Household | Toothbrush | 4 | 4 | 12 |
| Household | Hand wash | 12 | 3 | 15 |
| Electronics | $total | 100 | 6 | 665 |
| Electronics | Speakers | 43 | 3 | 240 |
| Electronics | TV | 12 | 1 | 300 |
| Electronics | Microwave | 4 | 1 | 85 |
| Electronics | Toaster | 41 | 1 | 40 |

### Output Sorted by Clicks metric

| Category | Item Name | Clicks | Quantity | Amount |
|---|---|---|---|---|
| $total | $total | 160 | 18 | 702 |
| Electronics | $total | 100 | 6 | 665 |
| Electronics | Speakers | 43 | 3 | 240 |
| Electronics | Toaster | 41 | 1 | 40 |
| Electronics | TV | 12 | 1 | 300 |
| Electronics | Microwave | 4 | 1 | 85 |
| Household | $total | 60 | 12 | 37 |
| Household | Fairy dish liquid | 44 | 5 | 10 |
| Household | Hand wash | 12 | 3 | 15 |
| Household | Toothbrush | 4 | 4 | 12 |

## Instructions

- Complete the OrderSummary.jsx file such that a request is made to this endpoint: https://mocki.io/v1/a7618665-b8e2-4304-91e5-e35b2ca5607d and the response is sorted and rendered in a HTML table for visualisation
  - Add logic to the `fetchData` function in `sort.js` file to load the data from the endpoint specified above
  - Complete the `sortData` function in `sort.js` to apply the sorting logic explained above
  - Import the methods above into the `OrderSummary.jsx` file and wire it up to render the sorted table
- Don't hardcode the sort metric. The `sortData` method hence, has the following schema:

```
const sortData = (data: Array<Object>, sortFn: Function) => {
  return sortedData as Array<Object>;
}
```

```
const sortFn = (row: Object) => row['metric']
```

## Available Scripts

In the project directory, you can run:

`npm start`

Runs the app in the development mode.\ Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.\ You may also see any lint errors in the console.

`npm test`

Launches the test runner in the interactive watch mode.\ See the section about running tests for more information.