# 6.3D: Conditions and Function

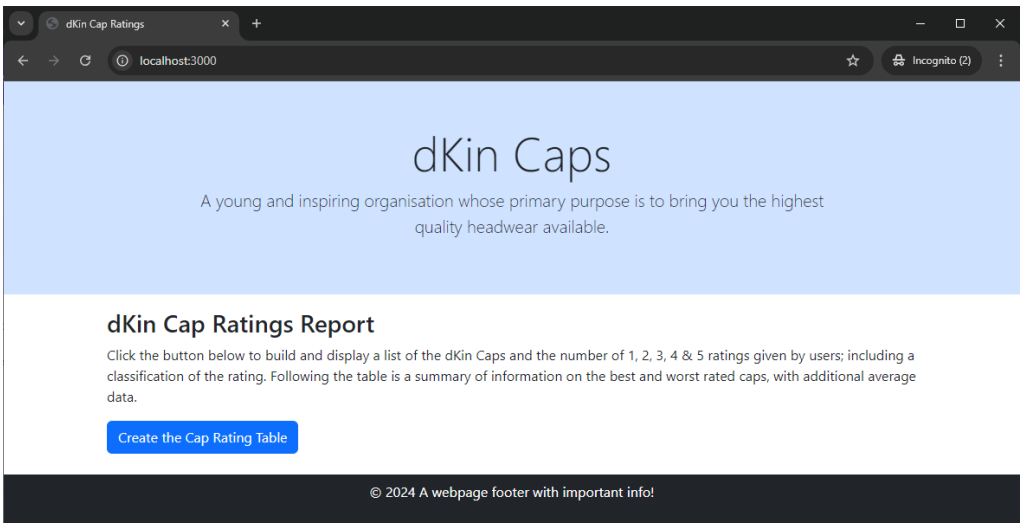
## Tasks

In this task you are asked to construct a table showing *Ratings Information* of customers of the ten **dKin Caps** types.

You will be provided with some raw data for the set cap ratings, which includes fields such as an abbreviation of the Cap type, the number of 1, 2, 3, 4, & 5 star ratings the cap has received.

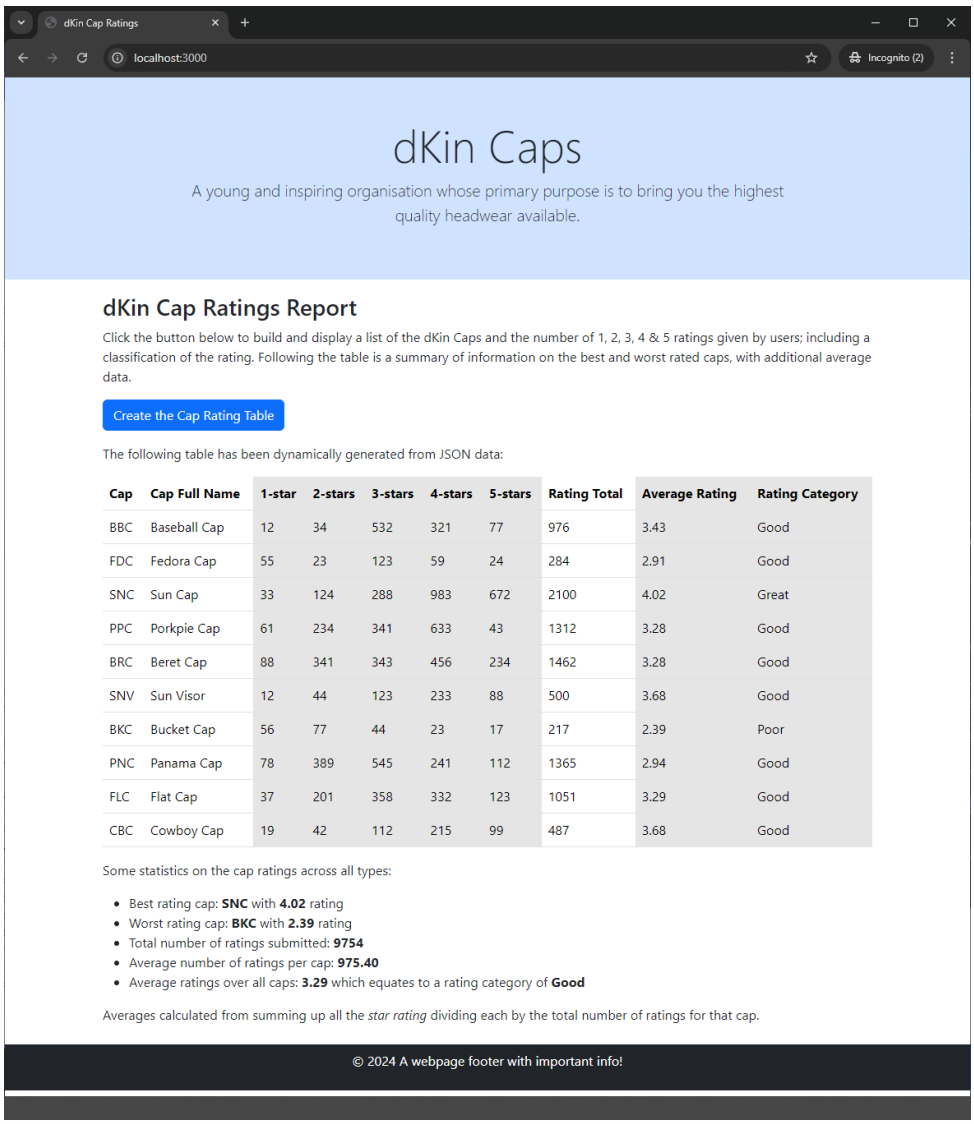
In completing this task, the code used to dynamically build the table should be allocated within a javascript function. This function is called when the user clicks a button.

The initial webpage should appear as shown below:




Task6.3.1: Initial page view

When the user clicks the button to generate the table, it should appear as shown in the following two images below:




Task6.3.2: List of Cap Ratings

You are provided with the raw data for the cap ratings in the form of a JSON object, that contains an array of `capratings`. This JSON object is shown below:

```
const capRatingListJSON = {
    capratings: [
        { cap:"BBC", stars: [12,34,532,321,77] },
        { cap:"FDC", stars: [55,23,123,59,24] },
        { cap:"SNC", stars: [33,124,288,983,672] },
        { cap:"PPC", stars: [61,234,341,633,43] },
        { cap:"BRC", stars: [88,341,343,456,234] },
        { cap:"SNV", stars: [12,44,123,233,88] },
        { cap:"BKC", stars: [56,77,44,23,17] },
        { cap:"PNC", stars: [78,389,545,241,112] },
        { cap:"FLC", stars: [37,201,358,332,123] },
        { cap:"CBC", stars: [19,42,112,215,99] }
    ]
};
```

> This cap rating data has been provided as a Javascript Array of objects which you may use, along with an spreadsheet example of the data. This is in a file named `capRatingData.zip` that is downloadable from the *Task6.3D* Ontrack page, in the **Task Details -> Resources** link at the bottom right

hand side of the web page.

**NOTE:** The JSON object provided has some fields that **are** shown in the table, while the last columns are **different** and not provided in the raw data. In completing this task, you will be required to write a support javascript function that takes an input (parameters) and return the processed result which is used to populate the table.

For example, the **Cap Name** is provided in the data in an abbreviated form, which needs to be presented in the second column in full-form, i.e., `BBC` should be presented as `Baseball Cap`. A javascript function that takes the *abbreviated capname* as input and returns the full name as a string could be used. An example of such a function outline is shown below:

```
function fullCapName( abbreviatedCapName ) {
    // Should use the input 'abbreviatedCapName' and return a string
    // with the full name (HINT: a switch() statement)
}
```

In addition the **Ratings Total**, **Average Rating** and **Rating Category** fields are not provided in the data so needs to be constructed and displayed in the final columns.

For example, a javascript function should be developed that that takes a respective cap's *stars* array as input and returns a *ratings total* could be used. An example of such a function outline is shown below:

```
function calcCapRatingTotal( capRatingsArray ) {
    // Should use the input array and return the sum of the number of ratings submitted for that
 cap.
}
```

Furthermore, a similar funciton could be developed that returns the *average cap rating*:

```
function calcCapAverageRating( capRatingsArray ) {
    // Should use the input array and return the average rating based on the caps' ratings.
    // Use the formula:
    //
    // avgrating = (ratings[0] * 1.0 + ratings[1] * 2.0
    //         + ratings[2] * 3.0 + ratings[3] * 4.0
    //         + ratings[4] * 5.0) / calcCapRatingTotal(ratings)
}
```

The final column in the table is **Rating Category** which is dependent on the *cap's rating* value. The following table details how the activity should be chosen:

| Cap Rating Range | Category Name |
|---|---|
| 0 <= capRating < 2.5 | 'Poor' |
| 2.5 <= capRating < 4.0 | 'Good' |
| 4.0 <= capRating | 'Great' |

In calculating the **Cap Category** field, a javascript function should be developed

that takes as input the *user rating* and using a cascading *if / then / else if* condition checks, returns the text (*string*) for the appropriate *category* from the table above, as shown in the code fragment below:

```
function capCategory( capRating ) {
    // Should use the input 'capRating' and return a string
    // with the full name (HINT: a cascading if/then/else if set of statements)
}
```

### Minimum / Maximum / Average Summary

The final section of the dynamically created table are the fields to show the various interesting features from the data. This includes:

- The cap type with the *best* rating and it's value
- The cap type with the *worst* rating and it's value
- The total (sum) of all ratings provided across all cap types
- The average number of ratings provided per cap
- The ice cream type with the least number sold and form which state
- The calculated *average cap rating* and the *average rating category* that this would equate to. This information should be displayed below the table.

## Hints:

- Use the JSON array of objects provided to store the Cap Rating details and information.
- Define Javascript functions that builds and populates the table (and summary infromation) when the button is clicked, i.e., implements the `buildCapRatingList()` funciton.
- Define a set support Javascript functions to return the *Full Cap Name*, *Total Number of Ratings*, *Average Cap Rating* and the *Average Cap rating category*.
- Use a table to organize the displayed data.
- Use additional variables to keep a total of the *total cap ratings submitted* and the *index* of the best & worst rated caps; these can be updated in the loop as you build the html to display the cap's row.

# What will you submit?

You should submit:

- Screen-shot(s) of the web page showing the final table.
- `HTML` base file
- The `JavaScript` code to construct the table.