1. Declaring an empty json data :

To do so we have to do

res.json({  data:[]   })

where data is the empty array , don’t just write res.json({}) , we have to give name of array too and now to access it in script.js file:  
const data = await response.json();

ie we are storing the json response from the server in variable ‘data’

hence to access its content we use: data.data.(…) as array name is data.

1. The flow of data between script.js , app.js and index.html file is:

First we send POST req by submitting the form (which will send the post request on the link /search) then that req is received by script.js first, now after that we use fetch() function which sends the data entered by user to server (app.js) and in it we search our data in our created database, then as per its result we send response to the script.js file which eventually shows the data on the HTML webpage using DOM concept

1. In script.js file we use fetch() function which returns promise => this actually sends request to our server (app.js file).

In case if we don’t use script.js in HTML then our browser automatically handles the POST req by directly sending data to server (app.js)

1. When we define our models then its not compulsory to declare all fields present in our database in Schema, instead we can define on our own (eg: There is no AdvRollNo field in Schema of ranks.js, similarly there is no SL. No. in branchcode.js)

Role of Schema is to connect our server via database so that we can easily do the query work and return required data.

1. In ImportAIO.js note that:
2. Array jsonFiles does not contain string data instead we have declared directly
3. MongoClient function is required so as to import the json data into MongoDB
4. In importData() function we first initialise the dbName and collectionName so as to connect to our required database. Then to read the contents of JSON file we use JSON.parse() function and hence jsonData contains whole of JSON data of that file. Then we use .map() which adds two other fields to whole data and then another .map() excludes the field named ‘AdvRollNo’. (These are just array methods as jsonData is an array). Now the collection variable actually contains function we need for CRUD. So we then just inserted whole pre-processed data in our Database.
5. Carefully note how the collection variable is made, we first used db = client.db(dbName) and then db.collection(coll..Name)
6. The code line:

new\_jsonData.map(obj => {

                const { "AdvRollNo": \_, ...rest } = obj;

                return rest;

            });

Here it means that we stored the value of key ‘AdvRollNo’ into \_ so that we could never use it and …rest takes all keys other than AdvRollNo , this is how we can destructure our ‘obj’ variable and then we are just returning the rest part.

1. The code line:

const filteredData = new\_jsonData.filter(obj => obj.category !== 'AdvRollNo');

is used when we have to exclude data on the basis of values instead of keys, so it’s searching all fields which have a value of AdvRollNo and excluding that document.