rocky
.monks

Explicação do código Javascript para correção dos arquivos broken_database:

1. **function executar()**: Define uma função chamada **executar()**. Essa função será chamada quando um botão for clicado na página.
2. **fetch('broken_database_1.json')**: Usa a função **fetch()** para buscar o arquivo JSON **broken_database_1.json** do servidor.
3. **.then(response => response.json())**: Chama o método **json()** da resposta do servidor (que é um objeto do tipo **Response**), para manipular os dados JSON contidos na resposta.
4. **.then(parsedData => {**: Usa o resultado da função anterior (ou seja, o objeto JSON analisado) e executa o código dentro dessa função de callback.
5. **parsedData.forEach(obj => {**: Chama o método **forEach()** do objeto JSON analisado, que executa uma função de callback para cada objeto no array JSON.
6. **obj.nome = obj.nome.replace(/æ/g, 'a');**: Modifica o valor do atributo **nome** de cada objeto, substituindo todos os caracteres **æ** por **a**.
7. **obj.nome = obj.nome.replace(/ø/g, 'o');**: Modifica novamente o valor do atributo **nome**, substituindo todos os caracteres **ø** por **o**.
8. **obj.vendas = parseInt(obj.vendas);**: Converte o valor do atributo **vendas** de cada objeto de string para inteiro, usando a função **parseInt()**.
9. **const newData = JSON.stringify(parsedData);**: Converte o objeto JSON modificado de volta para uma string JSON, usando o método **stringify()**.
10. **const blob = new Blob([newData], { type: 'application/json' });**: Cria um objeto Blob que contém os dados JSON modificados. Um objeto Blob é usado para representar um objeto de dados brutos, como um arquivo.
11. **const url = URL.createObjectURL(blob);**: Cria um URL temporário para o objeto Blob criado anteriormente.
12. **const a = document.createElement('a');**: Cria um elemento **<a>** no DOM (Documento Object Model).
13. **a.href = url;**: Define o atributo **href** do elemento **<a>** como o URL temporário criado anteriormente.
14. **a.download = 'new_data_1.json';**: Define o atributo **download** do elemento **<a>** como **new_data_1.json**. Isso indica que, quando o usuário clicar no link, o arquivo será baixado com esse nome.
15. **a.click();**: Clica programaticamente no elemento **<a>**, iniciando o download do arquivo.
16. **}).catch(error => console.error(error));**: Captura e trata quaisquer erros que ocorram durante o processo, escrevendo uma mensagem de erro no console do navegador.

A segunda parte do código é idêntica a primeira, mudando apenas o nome do arquivo a ser buscado e as correções que devem ser feitas.

Explicação do código SQL, utilizado para unificar os dois arquivos .JSON em um arquivo .CSV:

1. A primeira parte do código contém um comando **"CREATE TABLE"** que cria a tabela **"sales_data".** A tabela tem seis colunas, cada uma com um tipo de dados diferente.

2. A segunda parte do código contém um comando **"INSERT INTO"** que insere dados na tabela **"sales_data".** O comando usa o comando **"SELECT"** para buscar dados de duas fontes de dados diferentes: **"new_data_1"** e **"new_data_2".** O comando seleciona as colunas **"c1"** a **"c5"** de **"new_data_1"** e a coluna **"c2"** de **"new_data_2"**. As colunas selecionadas são então inseridas nas colunas correspondentes da tabela **"sales_data".**

3. O comando **"SELECT"** usa uma cláusula **"FROM"** com uma instrução **"INNER JOIN"** para relacionar os dados das duas fontes de dados. A junção é feita comparando o valor da coluna **"c2"** em **"new_data_1"** com o valor da coluna **"c1"** em **"new_data_2".**

4. O resultado é uma tabela com dados combinados das duas fontes de dados que são então inseridos na tabela **"sales_data".**

Observações sobre o código JS:

- Dado o uso da função fetch(), que permite realizar requisição HTTP, recomendo executar o código em servidor, em caso de uso do vscode recomendo o uso da extensão LiveServer, que executa o código em tempo real em servidor local.