Documentação do trabalho realizado

Passo a passo do que fiz para a entrega do projeto:

1. Escrever uma função para abrir e ler os arquivos broken_database_1 e broken_data_base_2, utilizando Javascript.

2. Escrever uma função para corrigir e transformar os “ø” em “o” e os “æ” em “a”, nos arquivos json.

3. Escrever uma função para transformar strings em number, no campo “vendas” dos arquivos json.

4. Escrever uma função para transformar o objeto js novamente em um arquivo json e exportar esses arquivos para uma pasta local no computador.

5. Importar os dois arquivos json corrigidos para SQL ONLINE .

6. Criar uma tabela única utilizando SQL.

7. Exportar a tabela SQL como um arquivo .CSV.

8. Utilizar o Excel para criar tabelas e gráficos com o arquivo .CSV.

9. Elaborar o relatório, utilizando texto, tabelas e gráficos.

10. Importar e organizar todos os arquivos em um repositório do meu Github.

Código JS (Optei por utilizar o VS Code para compilar, por já ter alguma experiência com o programa):

Função para abrir ler os arquivos json:

Const FileSystem = require('fs')

- O melhor modo que achei foi utilizar um módulo e um método do Node.JS para localizar e ler os aquivos .json no JS.
- Declarei uma variável e usei o módulo 'require', do node.js, para que o programa procurasse esse arquivo localmente, sempre que chamasse essa variável.

const dados = FileSystem.readFileSync('./broken_database_1.json', 'utf-8')

- Declarei uma variável dados e usei o método readFileSync para informar o arquivo que eu buscava. Esse método basicamente lê e retorna o conteúdo do arquivo passado como parâmetro.

let obj = JSON.parse(dados)

- Aqui declarei outra variável com let e utilizei o método .parse, que analisa o conteúdo do arquivo .json e o transforma em um objeto Javascript.

Corrigir letras e transformar strings em number:

```
for (let key in obj) {
obj[key].vendas = parseInt(obj[key].vendas)
obj[key].nome = obj[key].nome.replaceAll('ø', 'o').replaceAll('æ', 'a');
}
```

- Criei um loop utilizando for ..in para varrer toda lista (array) do objeto js. Key basicamente dita que todos os objetos no parâmetro vão ser afetados pelos comandos dentro das chaves.
  ".vendas" e ".nome" diz o campo onde o loop deve varrer.
  Usei .parseInt, que é um método utilizado para transformar uma string em um número inteiro.
  Utilizei o comando .replaceAll para transformar os símbolos em letras 'o' e 'a', em ".nome" e ".marca".

Transformar o objeto Javascript novamente em Json:

let fixed_database = JSON.stringify(obj)

- Aqui declarei ou variável fixed_data_base_1 e utilizei o método ".stringify" para transformar o objeto Javascript em arquivo .json, basicamente o inverso do que o método ".parse" faz.

Exportar o arquivo Json corrigido:

FileSystem.writeFile('fixed_database_1.json', fixed_database_1,
(Error) => {if (Error) throw Error;});

- Novamente utilizei um método do Node, o .writefile, para escrever os dados corrigidos no objeto jasvascript em um novo arquivo .json. Caso o arquivo não seja gerado corretamente o console apresenta uma mensagem de erro.

Código SQL:

SELECT fixed_dabase_1.DATA, fixed_database_2.MARCA, fixed_dabase_1.NOME,

fixed_dabase_1.VENDAS, fixed_dabase_1.VALOR_DO_VEICULO AS VALOR

FROM fixed_dabase_1

INNER JOIN fixed_dabase_2 ON fixed_dabase_1.ID_MARCA = fixed_dabase_2.ID.MARCA

- Utilizei SELECT para selecionar as colunas da tabela.
- Substitui a coluna fixed_dabase_1.ID_MARCA da primeira tabela pela coluna fixed_dabase_2.MARCA da segunda tabela
- Utilizei "AS" na última coluna, para substituir o nome para "VALOR"
- FROM indica de qual tabela estou selecionando as colunas.
- INNER JOIN indica onde estão os registros com valores semelhantes para mesclar em uma única coluna e ON indica em qual colunas esses valores devem ser buscados e mesclados.