1. Explicação do Código Javascript

```
1  // Função para substituir caracteres especiais
2  function alterarCaracteresEspeciais(texto) {
3    return texto
4      .replaceAll(/ø/g, "o")
5      .replaceAll(/JæC Motors/g, "JAC Motors")
6      .replaceAll(/æ/g, "a");
7  }
```

Figura 01: Função alterarCaracteresEspeciais

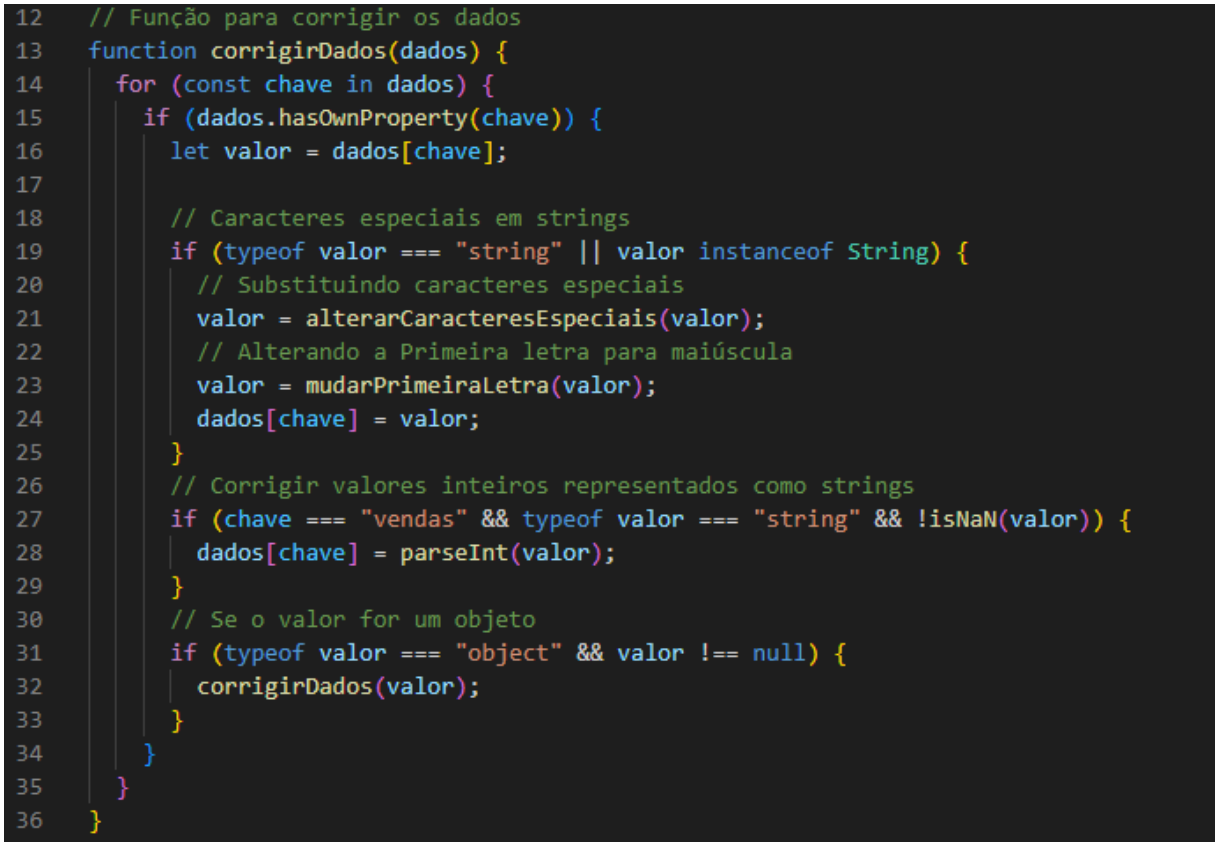
A função mostrada na figura 01 foi criada com o objetivo de substituir os caracteres especiais resultantes do banco de dados corrompido. Foi utilizado o método replaceAll, que substitui caracteres por outros de acordo com a necessidade. A escolha do replaceAll garante a substituição de todos os caracteres, ao contrário do método replace, que poderia não substituir alguns caracteres contidos dentro de uma mesma palavra ou frase nos dados.

```
8  // Função para corrigir a primeira letra para maiúscula
9  function mudarPrimeiraLetra(texto) {
10   return texto.charAt(0).toUpperCase() + texto.slice(1);
11 }
```

Figura 02: Função mudarPrimeiraLetra

A função mostrada na figura 02 tem como objetivo substituir a primeira letra dos dados. Devido à alteração de alguns nomes no banco de dados, onde a primeira letra precisa ser maiúscula, especialmente em nomes de veículos ou marcas.

O método charAt(0) é utilizado para obter a primeira letra do dado, enquanto o toUpperCase() converte essa letra para maiúscula. Em seguida, o '+ text.slice(1)' adiciona o restante do conteúdo do dado, começando a partir do segundo caractere, pois o primeiro caractere foi convertido para maiúscula com charAt(0).



```
// Função para corrigir os dados
function corrigirDados(dados) {
  for (const chave in dados) {
    if (dados.hasOwnProperty(chave)) {
      let valor = dados[chave];

      // Caracteres especiais em strings
      if (typeof valor === "string" || valor instanceof String) {
        // Substituindo caracteres especiais
        valor = alterarCaracteresEspeciais(valor);
        // Alterando a Primeira letra para maiúscula
        valor = mudarPrimeiraLetra(valor);
        dados[chave] = valor;
      }
      // Corrigir valores inteiros representados como strings
      if (chave === "vendas" && typeof valor === "string" && !isNaN(valor)) {
        dados[chave] = parseInt(valor);
      }
      // Se o valor for um objeto
      if (typeof valor === "object" && valor !== null) {
        corrigirDados(valor);
      }
    }
  }
}
```

Figura 03: Função corrigirDados

A função é criada para aplicar correções nos dados e recebe um objeto como parâmetro. Utilizando um loop 'for', itera em todas as chaves do objeto.

O uso do ‘hasOwnProperty(chave)’ é feito para verificar se a propriedade atual (chave) é uma propriedade específica e direta do objeto ‘dados’.

Há uma verificação para determinar se o valor de cada chave é uma string. Se for uma string, são aplicadas as funções 'alterarCaracteresEspeciais' e 'mudarPrimeiraLetra' para corrigir o texto.

Caso a chave observada seja 'vendas', seja uma string, e se a verificação '!isNaN' indicar que o valor dessa string pode ser transformado em um valor inteiro, é utilizado o comando 'parseInt' para essa conversão. Essa verificação foi necessária e aplicada exclusivamente para o campo de vendas, devido ao nome de alguns veículos serem compostos apenas por números, como é o caso de alguns carros da Peugeot, como 206, 2008, etc.

Também há uma verificação para determinar se o valor é um objeto e se não é nulo, para chamar recursivamente a função 'corrigirDados', corrigindo assim os dados aninhados.

```
38  // Leitura de arquivos utilizando require
39  const fs = require("fs");
40  // Ler o arquivo JSON 'broken_database_1.json'
41  const arquivo = "broken_database_1.json";
42  const dadosJSON = fs.readFileSync(arquivo, "utf8");
43  const dados = JSON.parse(dadosJSON);
44
45  // Chamando a função para corrigir os dados
46  corrigirDados(dados);
47  // Salvando os dados corrigidos no arquivo 'correct_database_1.json'
48  const dadosCorrigidosJSON = JSON.stringify(dados, null, 2);
49  fs.writeFileSync("correct_database_1.json", dadosCorrigidosJSON);
50
51  // Lendo o arquivo JSON 'broken_database_2.json'
52  const arquivo2 = "broken_database_2.json";
53  const dadosJSON2 = fs.readFileSync(arquivo2, "utf8");
54  const dados2 = JSON.parse(dadosJSON2);
55
56  // Chamando a função para corrigir os dados
57  corrigirDados(dados2);
58  // Salvando os dados corrigidos no arquivo 'correct_database_2.json'
59  const dadosCorrigidosJSON2 = JSON.stringify(dados2, null, 2);
60  fs.writeFileSync("correct_database_2.json", dadosCorrigidosJSON2);
61
```

Figura 04: Leitura e Gravação dos dados

O módulo 'fs' é utilizado para realizar a leitura e gravação dos dados.

Em seguida, são criadas constantes para receber os arquivos JSON, onde se identifica o arquivo, realiza-se a leitura e os converte através do ‘parse’ para que seja lido na linguagem JavaScript.

A função 'corrigirDados' é chamada para corrigir os dados, então é feita uma nova conversão do arquivo da linguagem JavaScript de volta para JSON. Através do método 'writeFileSync', os dados são escritos novamente, e o nome do arquivo é alterado para "correct_database_1.json".

Para otimização e melhor visualização do código, os dois arquivos do banco de dados JSON são importados dentro do mesmo código para correção. Para o segundo

banco de dados, são realizados os mesmos procedimentos descritos anteriormente, mas, é salvo com o nome "correct_database_2.json".

Após esse procedimento, os resultados salvos nos novos arquivos podem ser utilizados.

Para execução do código, foi utilizado o NodeJs, através do terminal. Lembrando de que os arquivos broken_database_1.json e broken_database_2.json devem estar no mesmo diretório do script de correção.

2. Explicação do Código SQL

O código SQL cria uma tabela chamada tabela_resultado através do CREATE TABLE e insere dados nela a que são consultados de outras tabelas chamadas correct_database_1 e correct_database_2 (Que foram os arquivos corrigidos pelo script em Javascript) através do comando INSERT INTO.

A criação dessa tabela foi o método escolhido para gerar um arquivo unificado no formato CSV.

SELECT: Utilizado para selecionar colunas específicas de tabelas.

INNER JOIN: Utilizado para combinar duas ou mais tabelas através de uma condição específica. No caso deste código é feita a conexão do arquivo correct_database_1 e correct_database_2 através do campo ID, obtendo assim o nome da marca que não estava disponível no primeiro arquivo (A.c2 = B.c1).

GROUP BY: Agrupa as linhas resultantes de alguma consulta com base em novos valores de alguma coluna.

ORDER BY: Utilizado para classificar o resultado de uma consulta em uma ordem determinada pelo usuário, facilitando a obtenção dos resultados que foram requeridos.

As variáveis c1, c2, c3, etc. são relacionadas as colunas do banco de dados.

As letras ‘A’ e ‘B’ inseridas ao se utilizar os arquivos correct_database_1 e correct_database_2 são para nomeá-las (referenciá-las) dentro do código SQL.

Pergunta 1:

```sql
CREATE TABLE tabela_resultado AS
-- Questão 1 Volume de Vendas
SELECT
B.c2 AS 'MARCA/MODELO',
SUM(A.c3) AS VALOR,
'-' AS COLUNA_3,
'Volume de Vendas por marca' AS LEGENDA
FROM correct_database_1 A
INNER JOIN correct_database_2 B ON A.c2 = B.c1
GROUP BY
B.c1,
B.c2
ORDER BY VALOR DESC;
```

Figura 05: Questão 01.

SUM(A.c3) AS VALOR: Utilizado para realizar a soma da coluna c3 da tabela correct_database_1 e é nomeada como 'VALOR', obtendo a informação de qual marca teve o maior volume de vendas.

'Volume de vendas por marca' AS LEGENDA: Foi atribuído uma coluna nova com o nome de ‘LEGENDA’ para ser utilizada como detalhe para as informações que serão consultadas e exibidas.

Nesse caso a tabela foi ordenada através do ORDER BY de forma decrescente, utilizando DESC.

Pergunta 2:

```sql
INSERT INTO tabela_resultado
-- Questão 2 Veiculo Maior Receita
SELECT
A.c5 AS MODELO,
SUM(A.c3*a.c4) AS TOTAL,
'-' AS COLUNA_3,
'Veiculos que geraram maior receita' AS LEGENDA
FROM correct_database_1 A
INNER JOIN correct_database_2 B ON A.c2 = B.c1
GROUP BY
B.c1,
A.c5
ORDER BY TOTAL DESC;

INSERT INTO tabela_resultado
-- Questão 2 Menor Receita
SELECT
A.c5 AS MODELO,
SUM(A.c3*a.c4) AS TOTAL,
'-' AS COLUNA_3,
'Veiculos que geraram menor Receita' AS LEGENDA
FROM correct_database_1 A
INNER JOIN correct_database_2 B ON A.c2 = B.c1
GROUP BY
B.c1,
A.c5
ORDER BY TOTAL ASC;
```

Figura 06: Questão 02.

SUM(A.c3*a.c4) AS TOTAL: É utilizado para realizar a soma da multiplicação da coluna c3 (valor da venda) e da coluna c4 (quantidade vendida) da tabela correct_database_1, obtendo-se o valor total das vendas dos veículos organizados por modelo.

Para exibir os que geraram maior receita foi utilizado o DESC (decrescente) e para exibir os que geraram menor receita foi utilizado o ASC (ascendente).

Pergunta 3:

```
43 INSERT INTO tabela_resultado
44 -- Questão 3 media de vendas do ano por marca
45 SELECT
46 B.c2 MARCA,
47 ROUND(AVG(A.c3),2) AS MEDIA,
48 CAST(A.c1 AS DATE) ANO,
49 'Media de Vendas no Ano' AS LEGENDA
50 FROM correct_database_1 A
51 INNER JOIN correct_database_2 B ON A.c2 = B.c1
52 GROUP BY
53 B.c1,
54 B.c2
55 ORDER BY MEDIA DESC;
```

Figura 07: Questão 03.

ROUND(AVG(A.c3),2) AS MEDIA: É utilizado para calcular a média das vendas (coluna c3 da tabela correct_database_1) para cada marca. A função AVG() calcula a média, e ROUND() corrige o número de casas decimais, realizando arredondamento, nesse caso para duas casas. Essa média é a resultante do número total de vendas dividido pela quantidade de vezes que foram registradas vendas no banco de dados.

A média foi exibida e organizada de forma decrescente para exibir as marcas com maior média de vendas primeiro.

Pergunta 4:

```
57 INSERT INTO tabela_resultado
58 -- Questão 4 Qual marca gera receita maior com menor numero de vendas
59 SELECT
60 B.c2 AS MARCA,
61 SUM(A.c4*A.c3)/ SUM(A.c3) AS RESULTADO_DIVISAO, --valor total dividido pela quantidade de vendas
62 SUM(A.c3) AS VENDAS,
63 'Maior Receita com menor numero de vendas' AS LEGENDA
64 FROM correct_database_1 A
65 INNER JOIN correct_database_2 B ON A.c2 = B.c1
66 GROUP BY
67 B.c1,
68 B.c2
69 ORDER BY RESULTADO_DIVISAO DESC;
```

Figura 08: Questão 04.

Para essa pergunta foi necessário realizar a soma do valor das vendas da marca e dividir pela soma de vendas, mostrando os valores mais altos que indicarão as marcas que geraram uma receita maior com um número menor de vendas.

SUM(A.c4*A.c3)/ SUM(A.c3) AS RESULTADO_DIVISAO: Utilizado para calcular a razão entre a receita total (produto da coluna c4(valor unitário), e da coluna c3, quantidade vendida) e o número total de vendas (soma da coluna c3). Esse resultado gera um valor de receita por unidade vendida.

Pergunta 5:

```
71 INSERT INTO tabela_resultado
72 -- Questão 5 Relação de veiculos mais vendidos
73 SELECT
74 A.c5 AS MODELO,
75 SUM(A.c3) AS QUANTIDADE_DE_VEICULOS_VENDIDOS,
76 B.c2 AS MARCA,
77 'Veiculos mais vendidos/Quantidade/Marca' AS LEGENDA
78 FROM correct_database_1 A
79 INNER JOIN correct_database_2 B ON A.c2 = B.c1
80 GROUP BY
81 B.c1, A.c5
82 ORDER BY
83 QUANTIDADE_DE_VEICULOS_VENDIDOS DESC;
84
85 SELECT * FROM tabela_resultado
```

Figura 09: Questão 05.

Para essa pergunta, foi necessário buscar a quantidade de vendas por veículo através de uma soma (SUM(A.c3)) separando pelo MODELO (A.c5). Em seguida, os resultados foram organizados em ordem decrescente usando o ORDER BY. Onde pôde-se observar os resultados e comparar a quantidade de vendas de acordo com o modelo dos veículos e assim verificar os primeiros veículos e buscar alguma característica comum entre eles.