Documentação dos códigos SQL utilizados

Após a correção/adequação das tabelas database_0 e database_1, foi realizado o upload das arquivos JSON para a plataforma SQL online. O primeiro passo foi visualizar ambas as tabelas, para verificação das colunas e linhas.

```
SELECT * FROM database_0;

SELECT * FROM database_1;
```

Identificou-se que as tabelas apresentavam um erro no nome das colunas, que estavam sendo erroneamente apresentadas como C1, C2, C3, C4 e C5. Para facilitar o entendimento e a junção das tabelas, todas as colunas foram renomeadas:

```
ALTER TABLE database_0 RENAME COLUMN c1 TO data;

ALTER TABLE database_0 RENAME COLUMN c2 TO id_marca;

ALTER TABLE database_0 RENAME COLUMN c3 TO vendas;

ALTER TABLE database_0 RENAME COLUMN c4 TO valor_do_veiculo;

ALTER TABLE database_0 RENAME COLUMN c5 TO nome;

ALTER TABLE database_1 RENAME COLUMN c1 TO id_marca;

ALTER TABLE database_1 RENAME COLUMN c2 TO marca;
```

Em seguida, foi feita a junção das tabelas, que possuíam em comum a coluna id_marca e criação da tabela database_2:

```
CREATE TABLE database_2 AS

SELECT

   db0.data AS data,

   db0.id_marca AS id_marca,

   db0.vendas AS vendas,

   db0.valor_do_veiculo AS valor_do_veiculo,

   db0.nome AS nome,

   db1.marca AS marca
```

```
FROM
   database_0 AS db0
INNER JOIN
   database_1 AS db1 ON db0.id_marca = db1.id_marca;
```

Após visualização da nova tabela, as tabelas utilizadas anteriormente foram descartadas.

```
DROP TABLE database_0;
 DROP TABLE database_1;
```

Para exportar o arquivo .CSV, foi feito um SELECT, e em seguida EXPORT > CSV

Com o objetivo de identificar qual marca teve o maior volume de vendas, utilizou-se o seguinte código:

```
SELECT
   marca,
   SUM(vendas) AS volume_de_vendas
FROM
   database_2
GROUP BY
   marca
ORDER BY
   volume_de_vendas DESC
LIMIT 1;
```

O código SQL apresentado acima realizou a somatória das vendas realizadas por marca, renomeando a coluna da consulta para volume_de_vendas, com os dados extraídos da database_2, e agrupou os resultados por marca. Os resultados foram apresentados em ordem decrescente, mas para facilitar a visualização, foi limitado a apresentar somente a marca com maior volume de vendas.

Para verificar qual veículo gerou a maior e menor receita, foi utilizado o código abaixo:

```
SELECT
        nome,
  SUM(valor_do_veiculo * vendas) AS receita_vendas
FROM
  database_2
GROUP BY
  marca
ORDER BY
  receita_vendas ASC
LIMIT 1;
```

O código apresentado realizou a soma do valor do veículo * vendas, por nome/modelo, para calcular a receita total de vendas que seria apresentada na consulta. Os dados foram extraídos da database_2, agrupados por marca, e, tendo em vista a visualização rápida da maior e menor receita, utilizou-se a consulta ASC e logo em seguida, DESC.

A média de vendas do ano por marca foi obtida da seguinte forma:

```
SELECT
  marca,
  AVG(vendas) AS media_de_vendas
FROM
  database_2
GROUP BY
  marca;
```

Os dados extraídos foram agrupados por marca e a média foi calculada utilizando AVG.

As marcas que geraram uma receita maior com número menor de vendas foram identificadas utilizando-se do trecho de código:

```
SELECT
    marca,
    SUM(vendas) AS total_de_vendas,
    SUM(vendas * valor_do_veiculo) AS receita_total
FROM
    database_2
GROUP BY
    marca
ORDER BY
    receita_total DESC;
```

Para essa consulta, um pouco mais complexa, foi selecionada a marca, calculada a soma de vendas por marca e a receita total (vendas * valor do veículo). Os resultados foram exibidos por ordem de receita decrescente.

Por último, foi realizada uma consulta com o objetivo de visualizar a quantidade total de vendas por nome/modelo e ainda, o valor de cada veículo. Nota-se, através da tabela abaixo, que não há uma relação direta de preço e quantidade de vendas. Os 03 veículos mais vendidos, se relacionam por serem veículos compactos.

```
SELECT
    nome,
    SUM(vendas) AS total_de_vendas, valor_do_veiculo
FROM
    database_2
GROUP BY
    nome
ORDER BY
    total_de_vendas DESC;
```

Documentação do código JavaScript utilizado

Foi utilizada a biblioteca fs para leitura e salvamento dos arquivos JSON.

Em seguida, foi criado um array com os caminhos dos arquivos broken_database_1 e broken_database_2. Foi criado também um array constante para poder guardar os arquivos separadamente. Depois, foi criada uma função para leitura dos arquivos e foi usada a função JSON.parse para a transformação dos dados em um objeto. Esse objeto foi passado como parâmetro da função posterior, corrigirDados.

Foi criada a função corrigirDados e em seguida, foi realizado o mapeamento dos dados usando a função .map. Cada item do objeto teve o conteúdo da chave "vendas" convertido em número. Logo em seguida, foi utilizada a função JSON.stringfy para cada item do .map, para que fosse possível utilizar a função replace e substituir os caracteres incorretos pelos corretos.

Logo após, foi retornado o mesmo item utilizando a função JSON.parse novamente. No final da função corrigir dados, foram passados os dados para o array de arquivos, utilizando a função JSON.stringfy para transformar os dados novamente em texto.

Depois desses passos, foi criada a função salvarArquivos, percorrendo o array de arquivos e criando um novo arquivo utilizando a função writeFile da biblioteca fs. Foi utilizado o index do array para nomear os arquivos separadamente. Também foi feita uma tratativa para informar se houve algum erro na exportação de alguns arquivos, evitando bugs no código.

Por fim, foi feita a função corrigir banco de forma assíncrona, para possibilitar que o sistema esperasse a leitura dos arquivos e o tratamento dos dados, evitando erros e não salvando um array vazio.