Códigos SQL

Criação da Tabela vendas_unificadas:

```
CREATE TABLE IF NOT EXISTS vendas_unificadas AS
SELECT
    d1.data,
    d1.id_marca_,
    d1.vendas,
    d1.valor_do_veiculo,
    d1.nome,
    d2.id_marca,
    d2.marca
FROM
    corrigido_database_1 d1
JOIN
    corrigido_database_2 d2 ON d1.id_marca_ = d2.id_marca;
```

Este bloco cria uma tabela chamada vendas_unificadas unindo os dados das duas tabelas corrigidas (corrigido_database_1 e corrigido_database_2). As colunas selecionadas são essenciais para as análises subsequentes.

Qual marca teve o maior volume de vendas?:

```
SELECT
    m.marca,
    COUNT(vu.id_marca) AS total_vendas
FROM
    vendas_unificadas vu
JOIN
    corrigido_database_2 m ON vu.id_marca = m.id_marca
GROUP BY
    m.marca
ORDER BY
    total_vendas DESC
LIMIT 10000;
```

Nesta consulta, você obtém o total de vendas por marca na tabela vendas_unificadas e, em seguida, ordena os resultados em ordem decrescente de vendas, limitando a 10.000 linhas.

Qual veículo gerou a maior e menor receita?:

```
SELECT
    vu.nome AS veiculo,
    SUM(vu.valor_do_veiculo * vu.vendas) AS receita_total
FROM
    vendas_unificadas vu
```

```
GROUP BY
    vu.nome
ORDER BY
    receita_total DESC
LIMIT 100000;
```

Esta consulta calcula a receita total para cada veículo multiplicando o valor do veículo pelas vendas. Os resultados são ordenados em ordem decrescente e limitados a 100.000 linhas.

Média de vendas do ano por marca:

```
SELECT
    m.marca,
    AVG(vu.vendas) AS media_vendas
FROM
    vendas_unificadas vu
JOIN
    corrigido_database_2 m ON vu.id_marca = m.id_marca
GROUP BY
    m.marca;
```

Nesta consulta, é calculada a média de vendas por ano para cada marca na tabela vendas_unificadas.

Marcas que geraram uma receita maior com número menor de vendas?:

```
SELECT
    m.marca,
    SUM(vu.valor_do_veiculo) AS receita_total,
    ROUND(AVG(vu.vendas), 2) AS media_vendas
FROM
    vendas_unificadas vu
JOIN
    corrigido_database_2 m ON vu.id_marca = m.id_marca
GROUP BY
    m.marca
ORDER BY
    receita_total DESC, media_vendas ASC;
```

Aqui, você obtém as marcas ordenadas pela receita total em ordem decrescente e, em caso de empate, pela média de vendas em ordem ascendente.

Relação entre os veículos mais vendidos:

```
WITH VeiculosMaisVendidos AS (
    SELECT
        vu.nome AS veiculo,
```

```
    SUM(vu.vendas) AS total_vendas
  FROM
    vendas_unificadas vu
  GROUP BY
    vu.nome
  ORDER BY
    total_vendas DESC
)
SELECT
  veiculo,
  total_vendas
FROM
  VeiculosMaisVendidos;
```

Nesta consulta, é utilizada uma Expressão Comum de Tabela (CTE) para calcular o total de vendas para cada veículo, e os resultados são retornados.

Códigos JavaScript

1. Leitura de Arquivos JSON:

```
function lerArquivosJson(arquivo1, arquivo2) {
  try {
    const data1 = JSON.parse(fs.readFileSync(arquivo1, 'utf-8'));
    const data2 = JSON.parse(fs.readFileSync(arquivo2, 'utf-8'));
    return [data1, data2];
  } catch (error) {
    console.error('Ocorreu um erro na leitura dos arquivos JSON:', error.message);
    return [];
  }
}
```

Esta função realiza a leitura de dois arquivos JSON, retornando um array contendo os dados desses arquivos. Em caso de erro, uma mensagem é exibida no console.

2. Reversão de Substituições de Caracteres:

```
function reverterSubstituicoes(data) {
  for (const registro of data) {
    for (const propriedade in registro) {
      if (registro.hasOwnProperty(propriedade) && typeof registro[propriedade] === 'string') {
        registro[propriedade] = registro[propriedade].replace(/æ/g, 'a').replace(/ø/g, 'o');
      }
    }
  }
}
```

A função reverterSubstituicoes percorre os registros e propriedades nos dados, revertendo substituições específicas em strings, como 'æ' por 'a' e 'ø' por 'o'.

3. Correção de Vendas:

```
function corrigirVendas(data) {
  for (const registro of data) {
    if (registro.vendas) {
      registro.vendas = Number(registro.vendas); // Transforma a string em número
    }
  }
}
```

Essa função assegura que a propriedade 'vendas' seja tratada como um número, não como uma string, percorrendo os registros de dados.

4. Exportação de Arquivo JSON:

```
function exportarArquivoJson(data, nomeArquivo) {
  fs.writeFileSync(nomeArquivo, JSON.stringify(data, null, 2));
}
```

A função exportarArquivoJson converte os dados em formato JSON e os escreve em um arquivo especificado.

5. Exemplo de Uso:

```
const arquivo1 = 'broken_database_1.json';
const arquivo2 = 'broken_database_2.json';

const [data1, data2] = lerArquivosJson(arquivo1, arquivo2);

reverterSubstituicoes(data1);
reverterSubstituicoes(data2);

corrigirVendas(data1);
corrigirVendas(data2);

exportarArquivoJson(data1, 'corrigido_database_1.json');
exportarArquivoJson(data2, 'corrigido_database_2.json');

console.log("Processo concluído com sucesso!");
```

Este exemplo ilustra o fluxo de processamento, desde a leitura dos arquivos até a exportação dos dados corrigidos. A mensagem "Processo concluído com sucesso!" é exibida no console ao finalizar.