Módulo 10 – Servidores – Atividade 05

Q1) Altere as queries de listagem para buscar registros unindo dados de mais de uma tabela. Por exemplo: para uma tabela de produtos que possui um relacionamento com uma tabela de categorias de produtos, a query de listagem de produtos deve retornar, além dos dados do produto, os dados da categoria a que o produto pertence.

Q2) Para implementar o carrinho de compras do sistema de e-commerce, crie as seguintes queries:

a. Adição de um produto ao carrinho do cliente;
b. Remoção de um produto do carrinho do cliente;
c. Listagem dos produtos do carrinho do cliente;
d. Cálculo da quantidade de produtos existente no carrinho do cliente;
e. Cálculo do valor total dos produtos do carrinho do cliente.

Q3) Para cada query criada na questão anterior, crie uma função na pasta “repositories” que receba como parâmetros os valores necessários e execute a referida query.

Respostas

Q1) A imagem a seguir mostra o código para a listagem de produtos, onde foi usado um JOIN para conectar com a tabela de categorias (categories) e outro join para conectar com a tabela de marcas (product_brands).

```
async function selectAllProducts(conn) {
    const sql = `
        SELECT products.id, products.name, products.price,
        categories.name AS category_name,product_brands.name AS brand_name,
        products.created_at
        FROM products
        JOIN categories ON products.category_id = categories.id
        JOIN product_brands ON products.brand_id = product_brands.id;
        `;
```

```
        //quando o sql tem parametro ($1,$2), qu
        const resp = await conn.query(sql);
        // console.log("veja a resposta do selec
```

```
    if (resp.rows.length === 0) {
        console.log(resp, resp.rows);
        throw "select product by brand and price";
    }

    return {
        productFiltered: resp.rows,
    };
```

No arquivo index.js:

```
145  //FUNCTION SELECT ALL PRODUCTS WITH JOIN CATEGORY AND BRAND
146  (async function query() {
147      const conn = await dbConnect();
148      const resp = await selectAllProducts(conn);
149
150      console.log("Resposta da query realizada - LISTA COMPLETA:", resp);
151
152      conn.release(); //libera essa conexão/client para que fique disponív
153      console.log("Conexão liberada para ser reaproveitada");
154  })();
155
```

Ao executar o comando “node index.js” na raiz do projeto o resultado é uma lista contendo todos os produtos:

```
letonio@letonio-Inspiron-15-3567:~/Documentos/alphaedtech/hardSkills/repositorio
Hard/Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node in
dex.js
Resposta da query realizada - LISTA COMPLETA: {
  productFiltered: [
    {
      id: 'e8ece67b-f036-4f38-8e02-fa34677b0333',
      name: 'Camisa do ben 10000',
      price: '250.00',
      category_name: 'Camisa',
      brand_name: 'Prada',
      created_at: 2023-02-11T15:53:19.321Z
    },
    {
      id: 'c14ea5c7-514c-4233-807a-a75d7bcc61ed',
      name: 'Camisa top top',
      price: '300.00',
      category_name: 'Camisa',
      brand_name: 'Prada',
      created_at: 2023-02-11T15:58:30.857Z
    },
```

Nota-se que, apesar de nome de categoria e nome de marca não fazerem parte da tabela “products”, essas informações também apareceram.

Q2) a)
Antes de implementar a query que permite adicionar produtos no carrinho do cliente, precisei inicializar a tabela de status do carrinho, pois será necessária quando iniciar o carrinho.

```
1  INSERT INTO cart_status (id,name)
2  VALUES (gen_random_uuid(),'Finished');
3  SELECT * FROM cart_status;
```

	id [PK] uuid	name text
1	7a41f2eb-b2d1-488b-81b2-5eea433863ae	Open
2	610a3991-4a69-4690-b002-476888f2ed42	Finished

O status Open implica que um carrinho foi inicializado e está aberto, onde poderá ser inseridos produtos. Caso o status de um carrinho mude para Finished a compra estará finalizada e o pedido será processado.

Antes de começar a inserir produtos, o usuário terá que inicializar um carrinho, de forma que um ID será gerado para o seu carrinho, será adicionada a informação de que o carrinho está aberto e a data de criação (created_at). A query para inicializar um carrinho precisa que o ID do usuário seja informado.
Para exemplo, considere um usuário com o ID = "aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa" (precisei adicionar esse usuário na tabela de users), e sabendo que o ID do status “Open ” é "7a41f2eb-b2d1-488b-81b2-5eea433863ae"
a query de criar um usuário foi:

```
INSERT INTO users
(id,name,email,password,created_at)
VALUES
(
'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa',
    'letonio',
    'lets@gmail.com',
    '1111',
    now()
);
SELECT * FROM users;
```

	id [PK] uuid	name text	email text	password text	created_at timestamp without time zone
1	aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa	letonio	lets@gmail.com	1111	2023-02-12 10:04:27.57953

Agora que o usuário foi adicionado, poderei inicializar um carrinho de compras:

```
INSERT INTO shopping_carts (id,user_id,status_id,created_at)
VALUES (
        gen_random_uuid(),
        'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa',
        '7a41f2eb-b2d1-488b-81b2-5eea433863ae',
        NOW()
);
SELECT * FROM shopping_carts;
```

	id [PK] uuid	user_id uuid	status_id uuid
1	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa	7a41f2eb-b2d1-488b-81b2-5eea433863ae

Gerei um carrinho de compras com o ID: "dd44fc81-3d31-4e32-ae8b-92c85e6705ff"

Agora que um carrinho está criado, é possível adicionar produtos, isso é feito numa tabela intermediária chamada: carts_products, que sempre que um produto é adicionado, cria-se uma

relação, id do carrinho com id do produto. Para exemplo de query envolvendo a adição de produtos, vamos considerar os seguintes produtos com seus respectivos ID:

ID DO PRODUTO	NOME DO PRODUTO
c14ea5c7-514c-4233-807a-a75d7bcc61ed	Camisa top top
e8ece67b-f036-4f38-8e02-fa34677b0333	Camisa do ben 10000

A próxima query permite adicionar um produto ao carrinho do cliente, sendo necessário passar como parâmetro o id do carrinho e o id do produto (serão usados numa tabela que combina ID do carrinho com ID do produto)

```
1  INSERT INTO carts_products
2  (cart_id,product_id, quantify,created_at)
3  VALUES
4  ('dd44fc81-3d31-4e32-ae8b-92c85e6705ff',
5   'c14ea5c7-514c-4233-807a-a75d7bcc61ed',
6   1,
7   now());
8   SELECT * FROM carts_products;
9
```

INSERT INTO carts_products
(cart_id,product_id, quantify,created_at)
VALUES
('dd44fc81-3d31-4e32-ae8b-92c85e6705ff',
 'c14ea5c7-514c-4233-807a-a75d7bcc61ed',
 1,
 now());
 SELECT * FROM carts_products;

Adicionarei mais um produto para que tenham dois produtos no carrinho:

INSERT INTO carts_products
(cart_id,product_id, quantify,created_at)
VALUES
('dd44fc81-3d31-4e32-ae8b-92c85e6705ff',
 'e8ece67b-f036-4f38-8e02-fa34677b0333',
 1,
 now());
 SELECT * FROM carts_products;

	cart_id uuid	product_id uuid	quantify integer	created_at timestamp without time zone
1	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	c14ea5c7-514c-4233-807a-a75d7bcc61...	1	2023-02-12 10:49:37.290119
2	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	e8ece67b-f036-4f38-8e02-fa34677b0333	1	2023-02-12 10:58:15.540125

Agora na tabela temos dois itens associados ao mesmo id do carro.

Q2)b) Para remover um produto do carrinho do cliente, basta usar o DELETE passando como condição o ID do carrinho e o ID do produto que precisa ser deletado.

DELETE FROM carts_products
WHERE
	cart_id ='dd44fc81-3d31-4e32-ae8b-92c85e6705ff'
	AND product_id = 'c14ea5c7-514c-4233-807a-a75d7bcc61ed';

SELECT * FROM carts_products;

```sql
DELETE FROM carts_products
WHERE
    cart_id ='dd44fc81-3d31-4e32-ae8b-92c85e6705ff'
    AND product_id = 'c14ea5c7-514c-4233-807a-a75d7bcc61ed';

SELECT * FROM carts_products;
```

Ao plotar a tabela “carts_products”, nota-se que está apenas um item nela:

	cart_id uuid	product_id uuid	quantify integer	created_at timestamp without time zone
1	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	e8ece67b-f036-4f38-8e02-fa34677b0333	1	2023-02-12 10:58:15.540125

é importante reforçar que aqui estamos aplicando um HARD DELETE, isto é, um item deletado realmente é excluído do banco de dados. Se fosse um SOFT DELETE, bastaria registrar no campo “delete_at”, e sempre que fosse filtrar produtos considerar apenas aqueles onde o delete_at continua NULL.

Q2)c) Listagem dos produtos no carrinho do cliente; Aqui faremos uso da função de JOIN para juntar o conteúdo de duas tabelas (a tabela de carrinhos) e a tabela de “carts_products” que associa um produto ao carrinho ao qual pertence.

```sql
SELECT carts_p.cart_id,carts_p.product_id,p.name,p.price
FROM carts_products AS carts_p
JOIN products AS p ON  p.id = carts_p.product_id
WHERE carts_p.cart_id='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';
```

SELECT carts_p.cart_id,carts_p.product_id,p.name,p.price
FROM carts_products AS carts_p
JOIN products AS p ON p.id = carts_p.product_id
WHERE carts_p.cart_id='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';

Comentando sobre a query apresentada acima, peguei só algumas informações relevantes, além disso, faz-se necessário adicionar o condicional de cart_id = valor específico porque numa situação prática não vai existir apenas um carrinho, terão vários carrinhos, um para cada cliente.
A próxima imagem mostra o resultado da query:

	cart_id uuid	product_id uuid	name text	price numeric
1	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	e8ece67b-f036-4f38-8e02-fa34677b0333	Camisa do ben 10000	250.00
2	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	e8ece67b-f036-4f38-8e02-fa34677b0333	Camisa do ben 10000	250.00
3	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	8d58ac7a-62b6-4b43-a4b1-89243d243d...	Camursa	220.00
4	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	71b0cd68-e41c-4b2f-b7e3-4b42f697e9ae	Ben10 novo	333

Q2)d)
Para calcular a quantidade de produtos existentes no carrinho, precisamos do id do carrinho do cliente e usar a função agregadora COUNT(*):

```
SELECT COUNT(*)
FROM carts_products
WHERE carts_products.cart_id='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';
```

SELECT COUNT(*)
FROM carts_products
WHERE carts_products.cart_id='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';

Como precisamos de todos os produtos que pertencem a um mesmo carrinho, precisamos do ID desse carrinho.

	count bigint
1	4

Q2) e)
A tabela “carts_products” contém uma coluna com o “product_id” e a tabela “products” contém um campo chamado “price” que aponta o preço do produto de acordo com o seu identificador. Para realizar a soma podemos utilizar a função SUM(), que soma os elementos de uma coluna que for especificada:

```
8  SELECT SUM(price)
9  FROM carts_products
10 JOIN products ON carts_products.product_id = products.id
11 WHERE carts_products.cart_id ='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';
```

SELECT SUM(price)
FROM carts_products
JOIN products ON carts_products.product_id = products.id
WHERE carts_products.cart_id ='dd44fc81-3d31-4e32-ae8b-92c85e6705ff';

	sum numeric
1	1053.00

Não faz parte da questão, mas enquanto estava resolvendo, encontrei uma forma que permite obter a soma total de cada carrinho, gerando uma tabela onde para cada id de carrinho tem o preço total contido naquele carrinho. Deixarei registrado ai caso precise implementar isso futuramente:
SELECT cart_id, SUM(price)
FROM carts_products
JOIN products ON carts_products.product_id = products.id
GROUP BY cart_id;

```sql
SELECT cart_id, SUM(price)
FROM carts_products
JOIN products ON carts_products.product_id = products.id
GROUP BY cart_id;
```

	cart_id uuid	sum numeric
1	dd44fc81-3d31-4e32-ae8b-92c85e6705ff	1053.00

Pronto, finalizamos as queries, a próxima etapa é implementar nas próximas questões.

Q3) a)
Abaixo está representado a função de adicionar produto no carrinho do cliente:

Dentro de repositories, criei um diretório chamado “shoppingCarts”. Esse diretório conterá todas as funções que estarão relacionadas ao carrinho de compra.
Para o item a), as funções desenvolvidas foram:

```sql
async function addProductToCart(conn, { cartId, productId }) {
    const sql = `
    INSERT INTO carts_products
    (cart_id,product_id, quantify,created_at)
    VALUES
    ($1,
     $2,
     1,
     now())
     RETURNING *;
    `;
```

```sql
    const resp = await conn.query(sql, [cartId, productId]);

    //  verdadeir implica que a query não tinha nenhum comando
    if (resp.rows.length === 0) {
        throw "Add product to cart";
    }

    return {
        productId: resp.rows,
    };
```

Uma observação, tem um campo de quantify que havia adicionado, decidi mantê-lo para não ter que excluir, porque a ideia inicial era que também fosse especificado quantas unidades daquele produto eram desejadas. No entanto, optei por considerar que toda compra é de um produto, por isso o “1”.

No arquivo index.js na raiz:

```
163   //FUNCTION ADD A PRODUCT TO SHOPPING CART
164   (async function query() {
165       const conn = await dbConnect();
166       const resp = await addProductToCart(conn, {
167           cartId: "773bfa86-42db-496c-878b-625a6af7ea04",
168           productId: "db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f",
169       });
170
171       console.log("Resposta da query realizada - add to CART:", resp);
172
173       conn.release(); //libera essa conexão/client para que fique dispon
174       console.log("Conexão liberada para ser reaproveitada");
175   })();
```

No terminal, apliquei o comando “node index.js”, então:

```
Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node index.js
Resposta da query realizada - add to CART: {
  productId: [
    {
      cart_id: '773bfa86-42db-496c-878b-625a6af7ea04',
      product_id: 'db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f',
      quantify: 1,
      created_at: 2023-02-12T15:41:13.516Z,
      update_at: null,
      delete_at: null
    }
  ]
}
Conexão liberada para ser reaproveitada
```

Q3)b)

Para a remoção do produto também se faz necessário saber duas informações: ID do carrinho, e ID do produto que desejamos remover.

Antes de remover o produto, usarei o pgAdmin para mostrar o produto que será removido

	cart_id uuid	product_id uuid	quantify integer	created_at timestamp without time zone
1	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243…	1	2023-02-12 12:20:19.182666
2	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243…	1	2023-02-12 12:20:39.661264
3	773bfa86-42db-496c-878b-625a6af7ea04	0382fef8-7b1e-44f8-94fb-f4b6af77bd3d	1	2023-02-12 12:21:03.267269
4	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243…	1	2023-02-12 12:21:19.451028
5	773bfa86-42db-496c-878b-625a6af7ea04	c4500132-e0a6-4a69-a3d4-db4c6d700…	1	2023-02-12 12:21:34.941435
6	773bfa86-42db-496c-878b-625a6af7ea04	db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f	1	2023-02-12 12:21:50.239085

Foi aplicado um filtro para mostrar apenas os produtos associado ao carrinho:
"773bfa86-42db-496c-878b-625a6af7ea04"
Será apagado para exemplo o últimoproduto adiconado,
cujo product_id = "db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f"

```
async function removeProductFromCart(conn, { cartId, productId }) {
    const sql = `
    DELETE FROM carts_products
    WHERE
    cart_id =$1
    AND product_id = $2
    RETURNING *;
    `;
    //quando o sql tem parâmetro ($1,$2), query() recebe um segundo
    const resp = await conn.query(sql, [cartId, productId]);

    //  verdadeir implica que a query não tinha nenhum comando de re
    if (resp.rows.length === 0) {
        throw "Remove product from cart";
    }

    return {
        productId: resp.rows,
    };
}

module.exports = {
    removeProductFromCart: removeProductFromCart,
};
```

```
//FUNCTION REMOVE PRODUCT FROM SHOPPING CART
(async function query() {
    const conn = await dbConnect();
    const resp = await removeProductFromCart(conn, {
        cartId: "773bfa86-42db-496c-878b-625a6af7ea04",
        productId: "db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f",
    });

    console.log("Resposta da query realizada - item removido:", resp);

    conn.release(); //libera essa conexão/client para que fique disponível
    console.log("Conexão liberada para ser reaproveitada");
})();
```

Ao aplicar o comando “node index.js”, ele dá como retorno da consulta o elemento que foi apagado

```
Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node index.js
Resposta da query realizada - item removido: {
  productId: [
    {
      cart_id: '773bfa86-42db-496c-878b-625a6af7ea04',
      product_id: 'db8fb2d1-6c0a-4ff0-ba75-db50e2b18b7f',
      quantify: 1,
      created_at: 2023-02-12T15:21:50.239Z,
      update_at: null,
      delete_at: null
```

Ao plotar a tabela no pgAdmin, nota-se que o elemento não existe mais

	cart_id uuid	product_id uuid	quantify integer	created_at timestamp without time zone
1	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	1	2023-02-12 12:20:19.182666
2	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	1	2023-02-12 12:20:39.661264
3	773bfa86-42db-496c-878b-625a6af7ea04	0382fef8-7b1e-44f8-94fb-f4b6af77bd3d	1	2023-02-12 12:21:03.267269
4	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	1	2023-02-12 12:21:19.451028
5	773bfa86-42db-496c-878b-625a6af7ea04	c4500132-e0a6-4a69-a3d4-db4c6d700777	1	2023-02-12 12:21:34.941435

Uma observação importante, depois tem que ser adicionado try catch, pois pode acontecer de uma solicitação tentar apagar um elemento que não tem id correspondente, sendo necessário fazer esse tratamento.

Q3)c)
Para a listagem de todos os produtos em um carro específico, faz-se necessário adicionar o id do carrinho. Além disso, a tabela “carts_products” deve ser consultada.
Para o presente teste, empregou-se
o ID do carrinho igual a "773bfa86-42db-496c-878b-625a6af7ea04"

```
1  async function selectAllProductsFromCart(conn, { cartId }) {
2      const sql = `
3      SELECT carts_p.cart_id,carts_p.product_id,p.name,p.price
4      FROM carts_products AS carts_p
5      JOIN products AS p ON  p.id = carts_p.product_id
6      WHERE carts_p.cart_id=$1;
7      `;
```

```
9      const resp = await conn.query(sql, [cartId]);
10
11     //  verdadeir implica que a query não tinha ner
12     if (resp.rows.length === 0) {
13         throw "List all products from cart";
14     }
```

No arquivo index.js que fica na raiz do projeto:

```
204  //FUNCTION GET/LIST ALL PRODUCTS FROM SHOPPING CART
205  (async function query() {
206      const conn = await dbConnect();
207      const resp = await selectAllProductsFromCart(conn, {
208          cartId: "773bfa86-42db-496c-878b-625a6af7ea04",
209      });
210
211      console.log("Resposta da query realizada - todos itens do carrinho:", resp);
212
213      conn.release(); //libera essa conexão/client para que fique disponível para n
214      console.log("Conexão liberada para ser reaproveitada");
215  })();
```

Ao executar o comando “node index.js”, o retorno da consulta é uma lista de todos os itens que estão associado ao id informado:

```
Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node index.js
Resposta da query realizada - todos itens do carrinho: {
  productId: [
    {
      cart_id: '773bfa86-42db-496c-878b-625a6af7ea04',
      product_id: '0382fef8-7b1e-44f8-94fb-f4b6af77bd3d',
      name: 'Curta verão',
      price: '450.00'
    },
    {
      cart_id: '773bfa86-42db-496c-878b-625a6af7ea04',
      product_id: '8d58ac7a-62b6-4b43-a4b1-89243d243deb',
      name: 'Camursa',
      price: '220.00'
    },
    {
      cart_id: '773bfa86-42db-496c-878b-625a6af7ea04',
```

Optou-se por retornar apenas algumas infomações (cart_id,product_id,name e price).

Q3)d) Para calcular a quantidade de produtos no carrinho, usa-se a função COUNT.
Mais uma vez será usado
o ID do carrinho igual a "773bfa86-42db-496c-878b-625a6af7ea04"

```
1  async function countProductsInCart(conn, { cartId }) {
2      const sql = `
3      SELECT COUNT(*)
4      FROM carts_products
5      WHERE carts_products.cart_id=$1;
6      `;
7      //quando o sql tem parâmetro ($1,$2), query() recebe
8      const resp = await conn.query(sql, [cartId]);
```

```
11      if (resp.rows.length === 0) {
12          throw "Count total of products in cart";
13      }
14
15      return {
16          productId: resp.rows,
17      };
```

Na raiz do projeto no index.js:

```
(async function query() {
    const conn = await dbConnect();
    const resp = await countProductsInCart(conn, {
        cartId: "773bfa86-42db-496c-878b-625a6af7ea04",
    });

    console.log(
        "Resposta da query realizada - quantidade de itens no carrinho:",
        resp
    );

    conn.release(); //libera essa conexão/client para que fique disponíve
    console.log("Conexão liberada para ser reaproveitada");
})();
```

Usando a linha de comando “node index.js”, o resultado da consulta é:

```
letonio@letonio-Inspiron-15-3567:~/Documentos/alphaedtech/hardSkills/repositorioHard/
Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node index.js
Resposta da query realizada - quantidade de itens no carrinho: { productId: [ { count
: '5' } ] }
Conexão liberada para ser reaproveitada
```

O que está consistente com o que foi abordado no item anterior que após a deleção havia restado apenas 5 produtos no carrinho.

Q3)e)

Novamente, para saber o preço total é necessário informar o id do carrinho.
Id: "773bfa86-42db-496c-878b-625a6af7ea04"

```
async function calculateTotalPriceInCart(conn, { cartId }) {
    const sql = `
    SELECT SUM(price)
    FROM carts_products
    JOIN products ON carts_products.product_id = products.id
    WHERE carts_products.cart_id =$1;
    `;
    //quando o sql tem parâmetro ($1,$2), query() recebe um se
    const resp = await conn.query(sql, [cartId]);
```

```
    //  verdadeir implica que a query não tinha nenhum
    if (resp.rows.length === 0) {
        throw "Sum total price of products in cart";
    }

    return {
        productId: resp.rows,
    };
```

O index.js:

```
245  //FUNCTION CALCULATE TOTAL PRICE PRODUCTS IN SHOPPING CART
246  (async function query() {
247      const conn = await dbConnect();
248      const resp = await calculateTotalPriceInCart(conn, {
249          cartId: "773bfa86-42db-496c-878b-625a6af7ea04",
250      });
251
252      console.log("Resposta da query realizada - preço total no carrinho:", resp
253
254      conn.release(); //libera essa conexão/client para que fique disponível par
255      console.log("Conexão liberada para ser reaproveitada");
256  })();
```

```
Alpha-edtech-cycle01/Module10-database/Aula05/app_js_connect_db_v2$ node index.js
Resposta da query realizada - preço total no carrinho: { productId: [ { sum: '1443.00
' } ] }
Conexão liberada para ser reaproveitada
```

O total está na chave sum: 1443.

A imagem abaixo confirma o valor, onde apresenta os 5 produtos:

	cart_id uuid	product_id uuid	name text	price numeric
1	773bfa86-42db-496c-878b-625a6af7ea04	0382fef8-7b1e-44f8-94fb-f4b6af77bd3d	Curta verão	450.00
2	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	Camursa	220.00
3	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	Camursa	220.00
4	773bfa86-42db-496c-878b-625a6af7ea04	8d58ac7a-62b6-4b43-a4b1-89243d243deb	Camursa	220.00
5	773bfa86-42db-496c-878b-625a6af7ea04	c4500132-e0a6-4a69-a3d4-db4c6d700777	Ben10 novo	333