Projeto - parte 2

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - UTFPR**

**SISTEMAS PARA INTERNET - SI**

MARIA EDUARDA GUEDES DE OLIVEIRA

MARIANA NICOLLE CORDEIRO

**SISTEMA DE VENDAS DE INGRESSO DE UM MUSEU**

**Desenvolvimento de uma Aplicação com a Manipulação em MySQL**

**GUARAPUAVA**

**2025**

**SUMÁRIO**

[**1. Utilize banco de dados relacional (Postgres ou MySQL). 3**](#_rh4r6ul93vek)

[**2. Defina a forma de conexão e acesso aos dados: 3**](#_gltxn8vuj7n1)

[a. No mínimo deve existir 3 tabelas relacionadas 3](#_3d3iiyqaziog)

[**3. Implemente os métodos para inserir, editar, buscar e excluir, sendo: 3**](#_h0wryakarnyz)

[a. Em cada tabela, insira 5 amostras de dados 3](#_aj1mxmulipc8)

[1. Tabela ingresso 3](#_o862wibmwqw9)

[3. Tabela sessao 4](#_t2iswe3annba)

[4. Tabela visitante 4](#_7r1iihdkwnea)

[b. Editar 3 opções diferentes de dados (independente da tabela) 4](#_c86d9qubpqsa)

[1. Editado o dado “cpf” do id “3” da tabela “visitante” 4](#_t8ddnki1bsqz)

[2. Editado o dado “valor” do id “3” da tabela “ingresso” 5](#_qu80gmk5ozgs)

[3. Editado o dado “numero\_carteirinha” do id “4” da tabela “pagamento” 6](#_u2jcxm3rx7sf)

[c. Em cada tabela, apresente busque todos os campos 7](#_ao3t8lzhum3z)

[1. Tabela ingresso 7](#_vf6nmeuvtc6t)

[2. Tabela pagamento 7](#_l2mpt644g9ph)

[3. Tabela sessao 8](#_lcrqjcsf5hv4)

[4. Tabela visitante 8](#_2flw7j6dh5p3)

[d. Exclua 2 amostras de dados em cada tabela 8](#_3vwwbo5pryz1)

[Tabela ingresso 9](#_l1ghlm6ysba2)

[Tabela pagamento 9](#_ek3478zg5nd1)

[Tabela sessao 10](#_749zcmk170ig)

[Tabela visitante 11](#_ceusa41eeu4h)

[**4. Permita o uso de pelo menos 2 buscas utilizando diferentes JOINS (INNER, LEFT, RIGHT ou FULL) 12**](#_d2zem6csnizj)

[a. Listar todos os visitantes que compraram ingressos e seus dados de pagamento (usando INNER JOIN). 12](#_gcmd3fryaixl)

[b. Listar informações sobre as sessões do museu e seus respectivos ingressos e pagamentos, mantendo todas as sessões mesmo que não tenham ingressos ou pagamentos associados (usando LEFT JOIN). 12](#_2frpmq44x1xw)

[**5. Faça pelo menos 2 buscas de dados em cada tabela utilizando diferentes funções de agregação (MAX, MIN, GROUP BY, HAVING, SUM, AVG) ou operadores de comparação (>, <, =, entre outros). 13**](#_sd1bd13cgncf)

[5.1 Tabela ingresso 13](#_ygo5yxaiy7ws)

[5.1.1 Média de valor dos ingressos (usando AVG) 13](#_vlaifvaxutty)

[5.1.2 Valor mínimo de ingresso (usando MIN). 13](#_2a26m7aqbkon)

[5.2 Tabela pagamento 14](#_2xkg62b8h6d6)

[5.2.1 Número de pagamentos e forma de pagamento (usando COUNT e GROUP BY) 14](#_pe333u7vjhoj)

[5.2.2 Encontrar o maior (ultimo) ID registrado na tabela pagamento (usando MAX) 14](#_2lazwsf8gz1f)

[5.3 Tabela sessao 14](#_s69tosevn9tm)

[5.3.1 Sessão mais recente (usando MAX) 14](#_cgis7relyxef)

[5.3.2 Sessões que ocorrerão após 03/06/2025 (usando operador de comparação >) 15](#_v0zqbu9wljlh)

[5.4 Tabela visitante 15](#_mdaxuln1icw8)

[5.4.1 Visitantes nascidos a partir dos anos 2000 (usando operadores de comparação >=) 15](#_c2sel3nuzh9w)

[5.4.2 Contagem total de visitantes (usando COUNT) 16](#_kau0y0no16yt)

[**6. Implemente 1 subquery. 16**](#_sp0w0hf9r4m6)

[**Implementação EXTRA (1.0 ponto): 17**](#_dh30tu9ccnjj)

[● Escolha uma linguagem Web de seu interesse e implemente a aplicação com interface para simular interação com usuário. Se possível, utilize o padrão MVC. 17](#_2won4x93lon2)

**2ª Parte do Projeto Final da Disciplina de Banco de Dados**

ITENS OBRIGATÓRIOS PARA A PARTE 2:

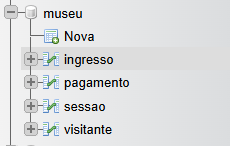
# 1. Utilize banco de dados relacional (Postgres ou **MySQL**).

# 

# 2. Defina a forma de conexão e acesso aos dados:

## a. No mínimo deve existir 3 tabelas relacionadas

Criado 4 tabelas chamadas: ***ingresso****,* ***pagamento****,* ***sessao****, e* ***visitante***



# 

# 3. Implemente os métodos para inserir, editar, buscar e excluir, sendo:

## a. Em cada tabela, insira 5 amostras de dados

### Tabela *ingresso*



1. Tabela *pagamento*



### Tabela *sessao*



### Tabela *visitante*

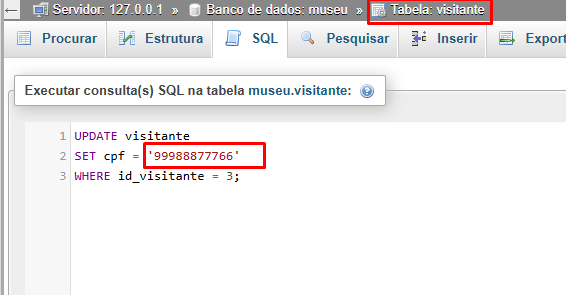


## b. Editar 3 opções diferentes de dados (independente da tabela)

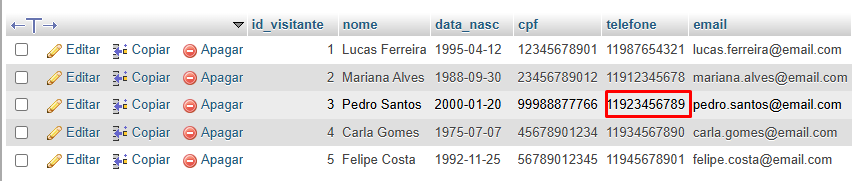
### 1. Editado o dado “cpf” do id “3” da tabela “*visitante*”



Segue SQL da edição.



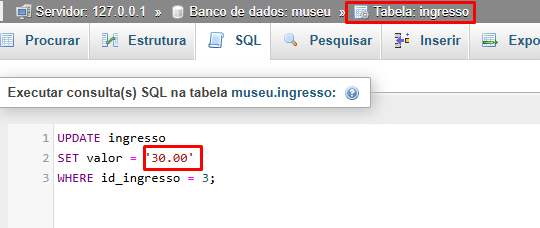
Dado alterado, segue:



### 2. Editado o dado “valor” do id “3” da tabela “*ingresso*”



Segue SQL da edição.



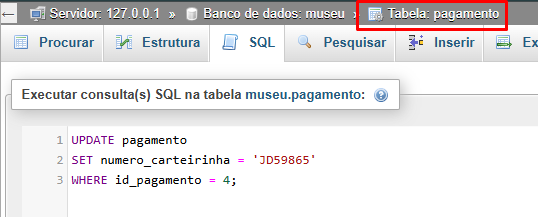
Dado alterado, segue:



### 3. Editado o dado “numero\_carteirinha” do id “4” da tabela “*pagamento*”



Segue SQL da edição

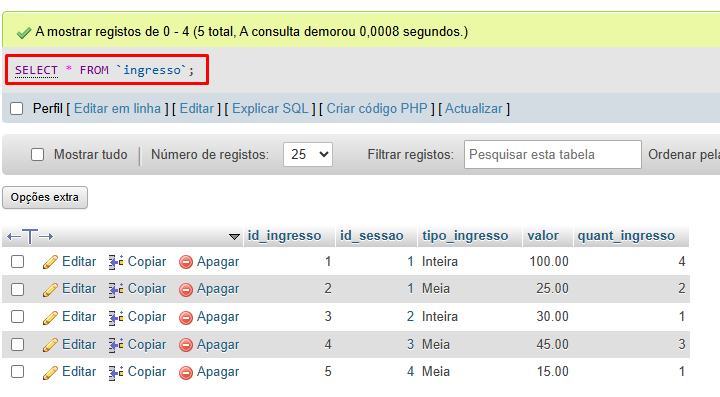


Dado alterado, segue:



## c. Em cada tabela, apresente busque todos os campos

### Tabela *ingresso*



### 

### Tabela *pagamento*



### Tabela *sessao*



### Tabela *visitante*



## d. Exclua 2 amostras de dados em cada tabela

Não é possível excluir as tabelas de *ingresso* porque ainda tem registros associados a tabela *sessao* que fazem referências por meio da chave estrangeira. Com isso, foi primeiramente excluído registros da tabela *ingresso* e posteriormente excluído da tabela *sessao*, segue cenário do localhost.

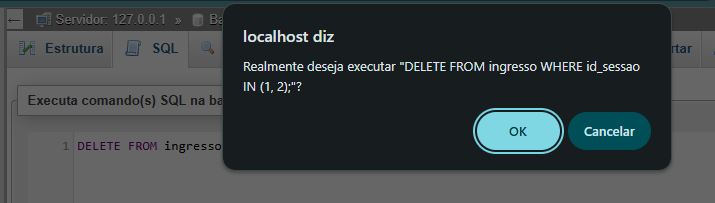
### **Tabela *ingresso***

-- Excluir os ingressos que estão associados às sessões 1 e 2

DELETE FROM ingresso WHERE id\_sessao IN (1, 2);

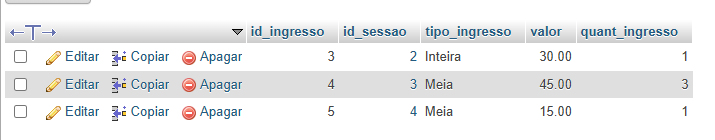


Confirmação de exclusão das colunas da tabela.



Certificando se os dados da tabela ingresso foram apagados:

* id\_ingresso 1, 2 foram apagados



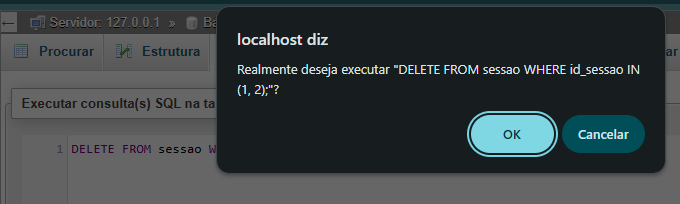
### **Tabela *pagamento***

-- Agora podemos excluir as sessões 1 e 2

DELETE FROM sessao WHERE id\_sessao IN (1, 2);



Confirmação de exclusão das colunas da tabela.



Certificando se os dados da tabela *pagamento* foram apagados:

* *id\_pagamento* 1, 2 foram apagados

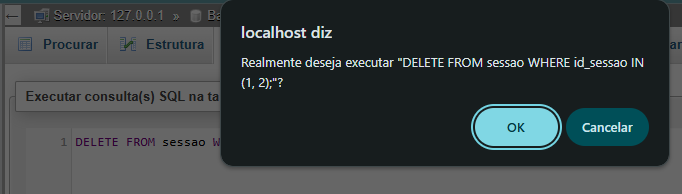


### **Tabela *sessao***

DELETE FROM sessao WHERE id\_sessao IN (1, 2);

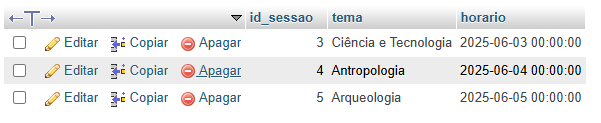


Confirmação de exclusão das colunas da tabela.



Certificando se os dados da tabela *sessao* foram apagados:

* *id\_sessao* 1, 2 foram apagados



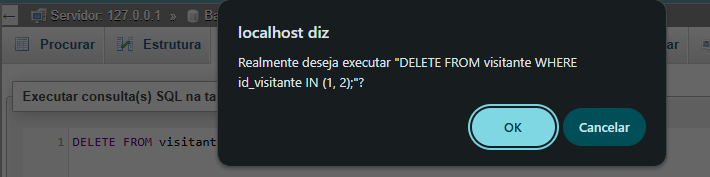
### 

### **Tabela *visitante***

DELETE FROM visitante WHERE id\_visitante IN (1, 2);



Confirmação de exclusão das colunas da tabela.



Certificando se os dados da tabela *visitante* foram apagados:

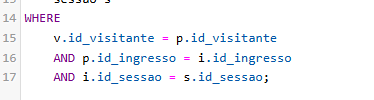
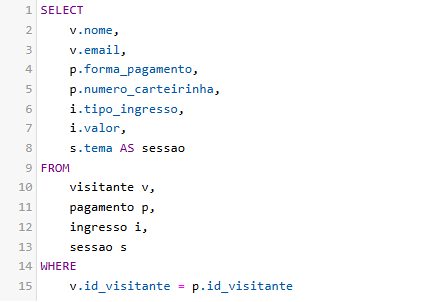
* *id\_visitante* 1, 2 foram apagados



# 4. Permita o uso de pelo menos 2 buscas utilizando diferentes JOINS (INNER, LEFT, RIGHT ou FULL)

### a. Listar todos os visitantes que compraram ingressos e seus dados de pagamento (usando **INNER JOIN**).

Segue SQL da busca.

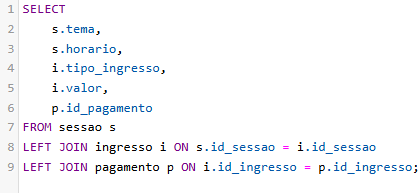


Resultado:



### b. Listar informações sobre as sessões do museu e seus respectivos ingressos e pagamentos, mantendo todas as sessões mesmo que não tenham ingressos ou pagamentos associados (usando **LEFT JOIN**).

Segue SQL da busca.



Resultado:



# 

# 5. Faça pelo menos 2 buscas de dados em cada tabela utilizando diferentes funções de agregação (MAX, MIN, GROUP BY, HAVING, SUM, AVG) ou operadores de comparação (>, <, =, entre outros).

## 

## **5.1 Tabela *ingresso***

### 5.1.1 Média de valor dos ingressos (usando **AVG**)

Segue SQL da busca.



Resultado:



### 5.1.2 Valor mínimo de ingresso (usando **MIN**).

Segue SQL da busca.



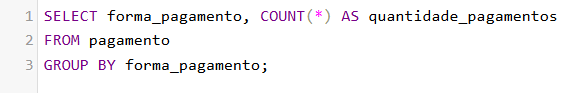
Resultado:



## **5.2 Tabela *pagamento***

### 5.2.1 Número de pagamentos e forma de pagamento (usando **COUNT** e **GROUP BY**)

Segue SQL da busca



Resultado:



### 5.2.2 Encontrar o maior (ultimo) ID registrado na tabela pagamento (usando **MAX**)

Segue SQL da busca



Resultado:



## 

## **5.3 Tabela *sessao***

### 5.3.1 Sessão mais recente (usando MAX)

Segue SQL da busca

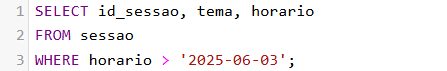


Resultado:



### 5.3.2 Sessões que ocorrerão após 03/06/2025 (usando operador de comparação **>**)

Segue SQL da busca



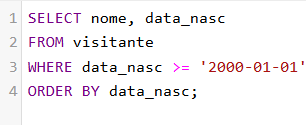
Resultado:



## **5.4 Tabela *visitante***

### 5.4.1 Visitantes nascidos a partir dos anos 2000 (usando operadores de comparação **>=**)

Segue SQL da busca

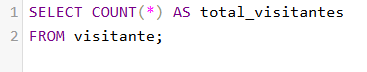


Resultado:



### 5.4.2 Contagem total de visitantes (usando COUNT)

Segue SQL da busca.



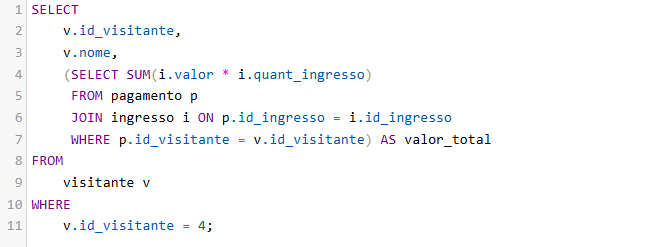
Resultado:



# 6. Implemente 1 subquery.

Obtendo o valor total de pedidos de um cliente específico.

Segue SQL da implementação.



Resultado:



Subquery Escalar

Encontrar funcionários com salário maior que a média:

SELECT \*

FROM funcionarios

WHERE salario > (SELECT AVG(salario) FROM funcionarios);

Obter o valor total de pedidos de um cliente específico

SELECT \*

FROM pedidos

WHERE valor\_total = (SELECT SUM(valor\_total) FROM pedidos

WHERE id\_cliente = 1);

# 

# 

# Implementação EXTRA (1.0 ponto):

## ● Escolha uma linguagem Web de seu interesse e implemente a aplicação com interface para simular interação com usuário. Se possível, utilize o padrão MVC.

## 