

BANCO DE DADOS

Trabalho – Relatório

Curso:	Tecnologia em Ciência de Dados
Aluno(a):	Bruna Gaino Lipovscek
RU:	5176496

1. 1ª Etapa – Modelagem

Pontuação: 30 pontos.

Dadas as regras de negócio abaixo listadas, referentes ao estudo de caso de uma Rede de Hotéis, elabore o Modelo Entidade-Relacionamento (MER), isto é, o modelo conceitual.

O Modelo Entidade-Relacionamento (MER) deve contemplar os seguintes itens:

- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

Uma Rede de Hotéis necessita controlar os dados dos funcionários, das unidades, dos quartos, dos hóspedes, das reservas e dos pagamentos. Para isso, contratou um profissional de Banco de Dados, a fim de modelar o Banco de Dados que armazenará todos os dados.

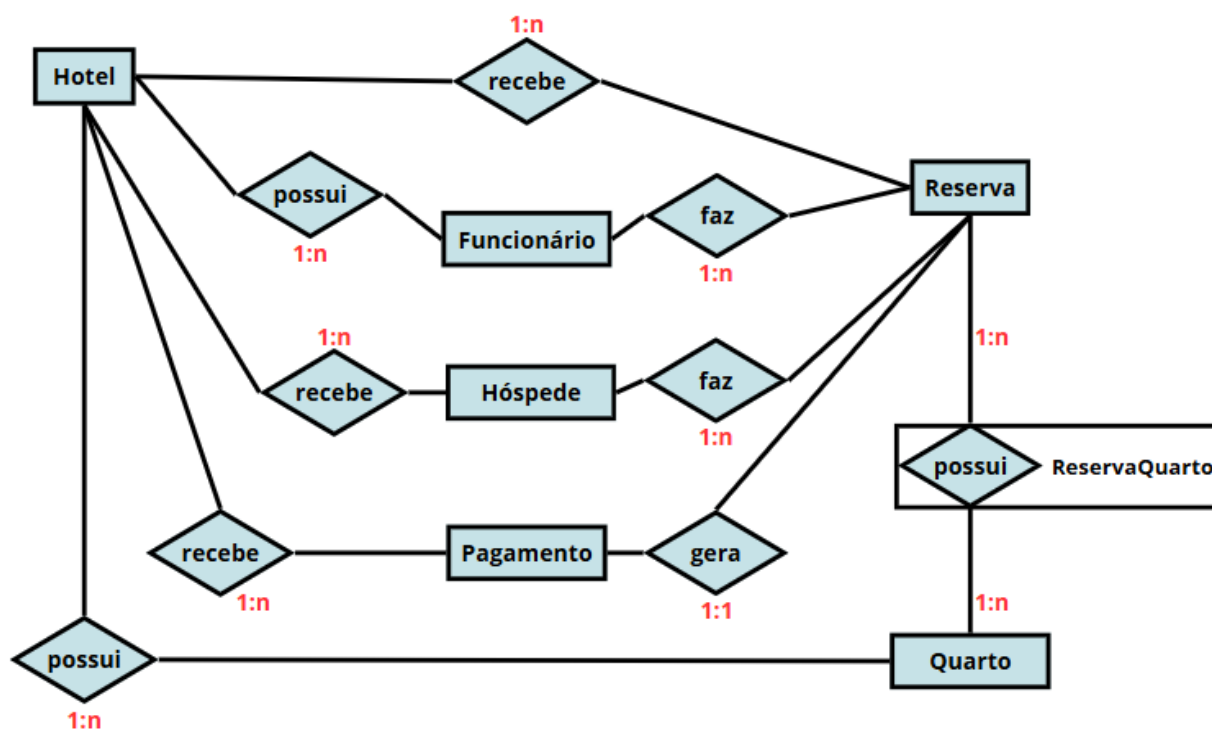
As regras de negócio são:

- Funcionário – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail, login e senha;

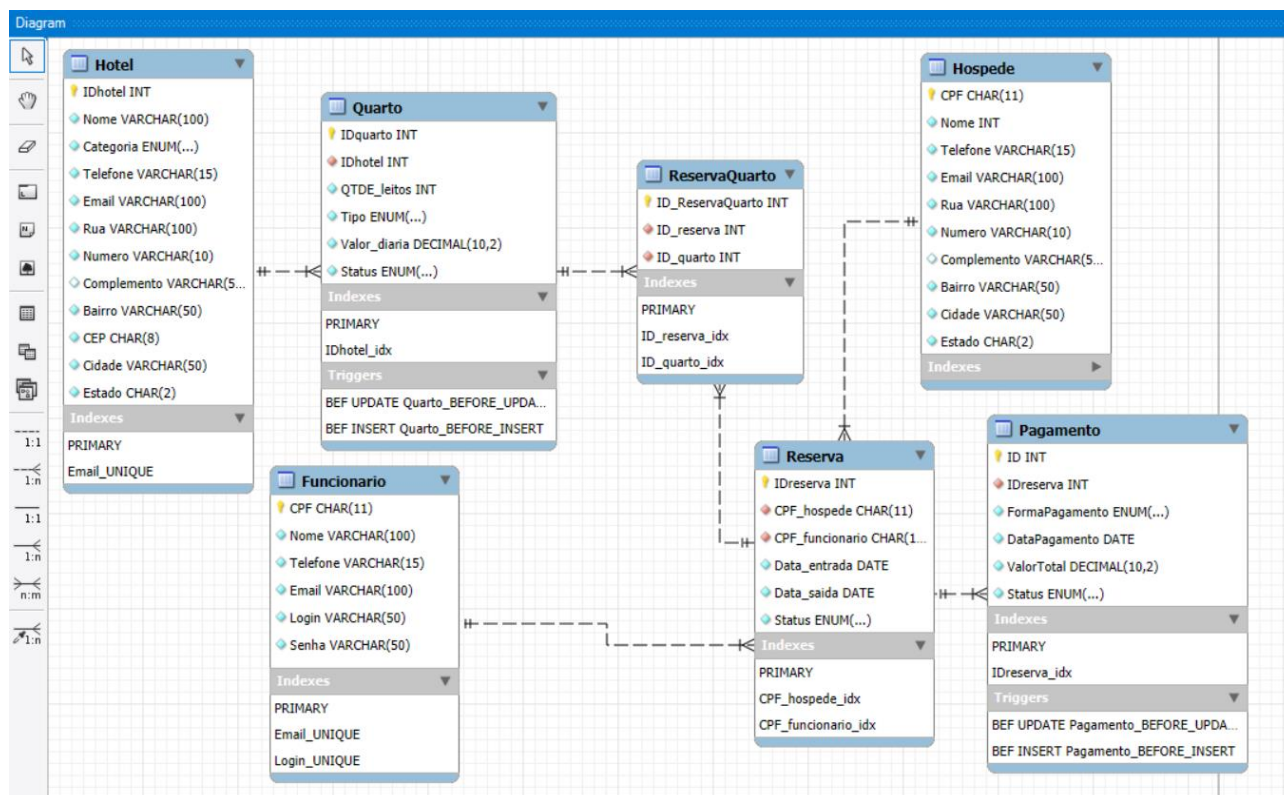
- Hotel – Deverão ser armazenados os seguintes dados: identificação do hotel, nome, categoria, telefone, e-mail e endereço, sendo o endereço composto por rua, número, complemento, bairro, CEP, cidade e estado;
- Quarto – Deverão ser armazenados os seguintes dados: identificação do quarto, número de leitos, tipo (*standard*, luxo ou suíte), preço da diária e *status* (disponível, ocupado ou manutenção);
- Hóspede – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail e endereço, sendo o endereço composto por rua, número, complemento, bairro, CEP, cidade e estado;
- Reserva – Deverão ser armazenados os seguintes dados: identificação da reserva, data de entrada, data de saída e *status* (ativa, cancelada ou concluída);
- Pagamento – Deverão ser armazenados os seguintes dados: identificação do pagamento, forma de pagamento (cartão, pix ou dinheiro), data do pagamento, valor total e *status* (pago ou pendente);
- Um hotel possui um ou vários quartos;
- Um ou vários funcionários trabalham em um hotel;
- Um funcionário realiza uma ou várias reservas;
- Um ou vários quartos fazem parte de uma ou várias reservas;
- Um hóspede pode fazer uma ou várias reservas;
- Uma reserva gera um pagamento.

Importante:

- O Modelo Entidade-Relacionamento (MER) deve considerar somente as regras de negócio dadas, não podendo ser criada nenhuma outra entidade ou atributo que não estejam nas regras de negócio;
- Em caso de haver entidade associativa, a mesma deve ser representada pela “Representação 1” (texto da Aula 1 – Fundamentos de Banco de Dados, Figura 25);
- Em caso de haver cardinalidade (1,1), a chave estrangeira deve fazer parte da entidade que possui o maior número de chaves estrangeiras.

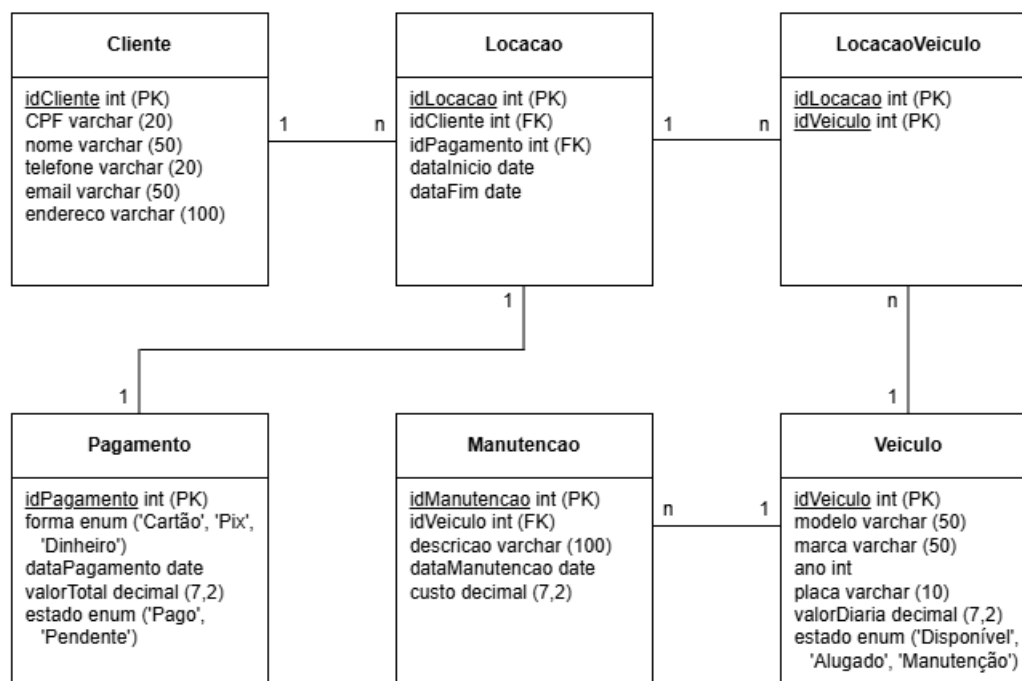


Bruna Lipovscek
Tec. em Ciência de Dados



2. 2ª Etapa – Implementação

Considere o seguinte Modelo Relacional (modelo lógico), referente ao estudo de caso de uma Locadora de Veículos:



Com base no Modelo Relacional dado e utilizando a *Structured Query Language* (SQL), no MySQL Workbench, implemente o que se pede.

Importante: Para testar o Banco de Dados após a implementação, utilize os comandos contidos no arquivo “Trabalho – Populando o Banco de Dados” para popular as tabelas. Tal arquivo contém todos os comandos de inserção dos dados (fictícios) necessários para a realização dos testes.

Pontuação: 30 pontos.

1. Implemente um Banco de Dados chamado “LocadoraVeiculos”. Após, implemente as tabelas, conforme o Modelo Relacional dado, observando as chaves primárias e as chaves estrangeiras. Todos os campos, de todas as tabelas, não podem ser nulos (*not null*).

```
CREATE DATABASE IF NOT EXISTS LocadoraVeiculos;  
USE LocadoraVeiculos;
```

```
CREATE TABLE Cliente (  
    idCliente INT NOT NULL AUTO_INCREMENT,  
    CPF VARCHAR(14) NOT NULL UNIQUE,  
    nome VARCHAR(50) NOT NULL,  
    telefone VARCHAR(20) NOT NULL,  
    email VARCHAR(50) NOT NULL UNIQUE,  
    endereco VARCHAR(100) NOT NULL,  
    PRIMARY KEY (idCliente)  
) ENGINE = InnoDB;
```

```
CREATE TABLE Veiculo (  
    idVeiculo INT NOT NULL AUTO_INCREMENT,  
    modelo VARCHAR(50) NOT NULL,  
    marca VARCHAR(50) NOT NULL,  
    ano INT NOT NULL,  
    placa VARCHAR(10) NOT NULL UNIQUE,  
    valorDiaria DECIMAL(7,2) NOT NULL,  
    estado ENUM('Disponível', 'Alugado', 'Manutenção') NOT NULL,  
    PRIMARY KEY (idVeiculo)  
) ENGINE = InnoDB;
```

```
CREATE TABLE Pagamento (  
    idPagamento INT NOT NULL AUTO_INCREMENT,  
    forma ENUM('Cartão', 'Pix', 'Dinheiro') NOT NULL,  
    dataPagamento DATE NOT NULL,  
    valorTotal DECIMAL(7,2) NOT NULL,  
    estado ENUM('Pago', 'Pendente') NOT NULL,  
    PRIMARY KEY (idPagamento)  
) ENGINE = InnoDB;
```

```
CREATE TABLE Locacao (  

```

```
idLocacao INT NOT NULL AUTO_INCREMENT,  
idCliente INT NOT NULL,  
idPagamento INT NOT NULL,  
dataInicio DATE NOT NULL,  
dataFim DATE NOT NULL,  
PRIMARY KEY (idLocacao),  
FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente) ON DELETE CASCADE  
ON UPDATE CASCADE,  
FOREIGN KEY (idPagamento) REFERENCES Pagamento(idPagamento) ON  
DELETE NO ACTION ON UPDATE CASCADE  
) ENGINE = InnoDB;
```

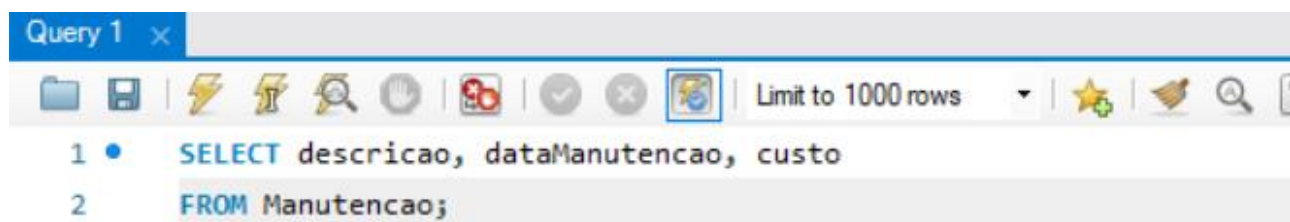
```
CREATE TABLE LocacaoVeiculo (  
idLocacao INT NOT NULL,  
idVeiculo INT NOT NULL,  
PRIMARY KEY (idLocacao, idVeiculo),  
FOREIGN KEY (idLocacao) REFERENCES Locacao(idLocacao) ON DELETE  
CASCADE ON UPDATE CASCADE,  
FOREIGN KEY (idVeiculo) REFERENCES Veiculo(idVeiculo) ON DELETE NO  
ACTION ON UPDATE CASCADE  
) ENGINE = InnoDB;
```

```
CREATE TABLE Manutencao (  
idManutencao INT NOT NULL AUTO_INCREMENT,  
idVeiculo INT NOT NULL,  
descricao VARCHAR(100) NOT NULL,  
dataManutencao DATE NOT NULL,  
custo DECIMAL(7,2) NOT NULL,  
PRIMARY KEY (idManutencao),  
FOREIGN KEY (idVeiculo) REFERENCES Veiculo(idVeiculo) ON DELETE  
CASCADE ON UPDATE CASCADE  
) ENGINE = InnoDB;
```

Pontuação: 10 pontos.

2. Implemente uma consulta para listar a descrição, a data e o custo de todas as manutenções realizadas nos veículos.

```
SELECT descricao, dataManutencao, custo  
FROM Manutencao;
```



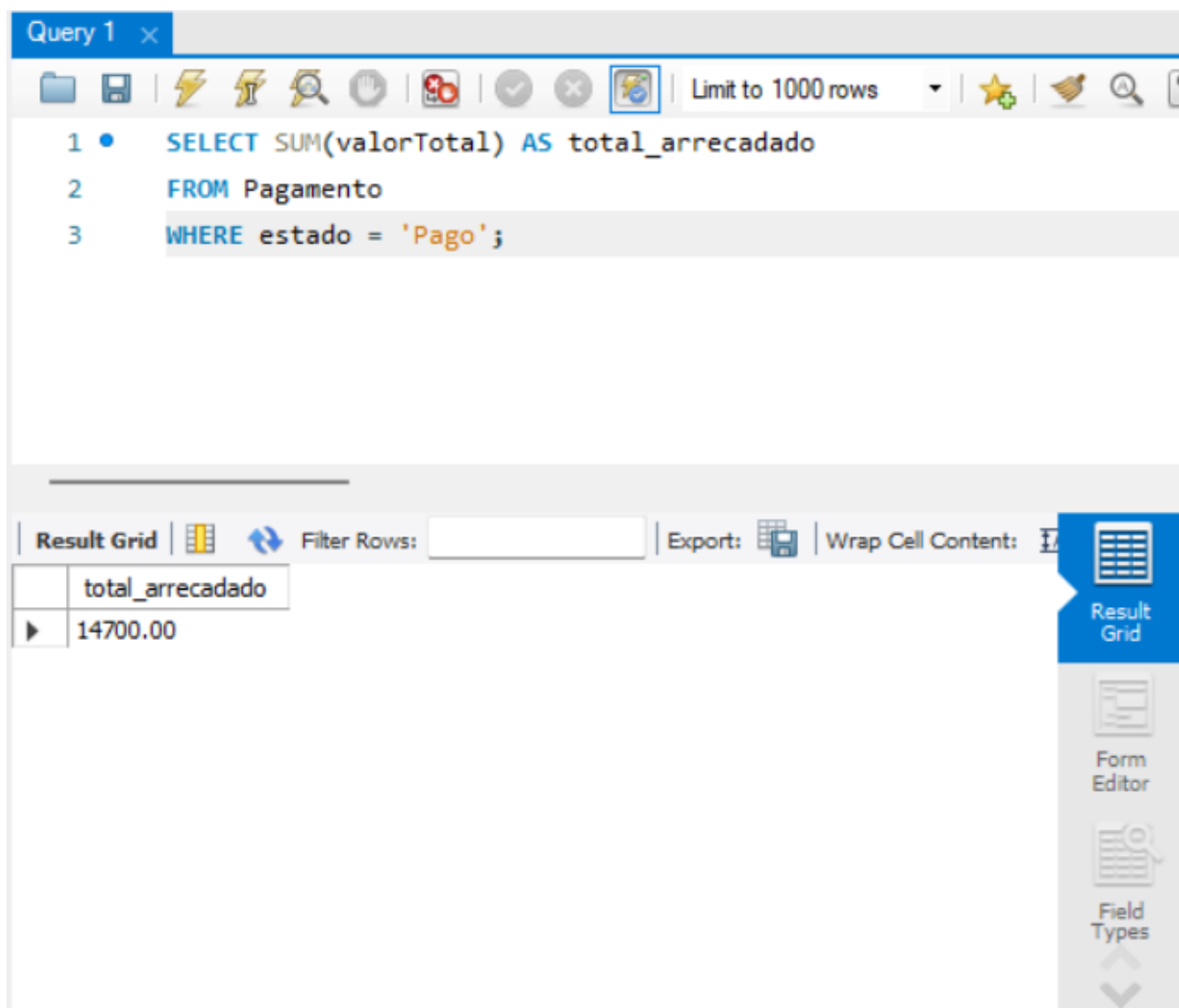
The screenshot shows the 'Result Grid' of the database application. The table has four columns: 'descricao', 'dataManutencao', and 'custo'. The data is as follows:

	descricao	dataManutencao	custo
▶	Troca de óleo e revisão geral	2024-12-09	200.00
	Substituição de pneu	2024-12-10	600.00
	Troca de pastilhas de freio	2024-12-14	450.00
	Alinhamento e balanceamento	2024-12-18	150.00
	Revisão elétrica completa	2024-12-28	500.00
	Reparo na suspensão	2025-01-05	700.00
	Troca do sistema de escapamento	2025-01-07	750.00
	Troca de bateria	2025-01-17	400.00
	Substituição do filtro de ar	2025-01-17	120.00
	Pintura e retoques na lataria	2025-01-28	900.00

Pontuação: 10 pontos.

3. Implemente uma consulta para listar o valor total arrecadado pela locadora. Lembre-se que pagamentos “pendentes” não fazem parte da soma.

```
SELECT SUM(valorTotal) AS total_arrecadado  
FROM Pagamento  
WHERE estado = 'Pago';
```

The screenshot shows a database query editor interface. At the top, there's a tab labeled "Query 1". Below the tab is a toolbar with various icons for file operations, execution, and settings. The main area contains a SQL query:

```
1 • SELECT SUM(valorTotal) AS total_arrecadado
2 FROM Pagamento
3 WHERE estado = 'Pago';
```

Below the query editor, there's a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The result grid itself shows a single row with the column "total_arrecadado" and the value "14700.00". On the right side of the interface, there's a vertical toolbar with buttons for "Result Grid", "Form Editor", and "Field Types".

Pontuação: 10 pontos.

4. Implemente uma consulta para listar o modelo e a marca dos veículos, bem como o número de vezes que cada um foi locado. A listagem deve ser mostrada em ordem decrescente pelo número de aluguéis.

Dica: Utilize a cláusula *group by*.

```
SELECT Veiculo.modelo, Veiculo.marca, COUNT(LocacaoVeiculo.idLocacao) AS
total_locacoes
FROM Veiculo
JOIN LocacaoVeiculo ON Veiculo.idVeiculo = LocacaoVeiculo.idVeiculo
GROUP BY Veiculo.idVeiculo
ORDER BY total_locacoes DESC;
```


Query 1

```
1 • SELECT Veiculo.modelo, Veiculo.marca, COUNT(LocacaoVeiculo.idLocac
2 FROM Veiculo
3 JOIN LocacaoVeiculo ON Veiculo.idVeiculo = LocacaoVeiculo.idVeicul
4 GROUP BY Veiculo.idVeiculo
5 ORDER BY total_locacoes DESC;
```

Result Grid

	modelo	marca	total_locacoes
▶	HB20	Hyundai	4
	Duster	Renault	3
	Gol	Volkswagen	2
	Corolla	Toyota	2
	Fiesta	Ford	2
	Toro	Fiat	2
	Compass	Jeep	2
	Onix	Chevrolet	1
	Civic	Honda	1
	Cruze	Chevrolet	1

Pontuação: 10 pontos.

5. Implemente uma consulta para listar o nome dos clientes que possuem pagamento “pendente”, bem como o valor devido por eles. A listagem deve ser mostrada em ordem alfabética crescente pelo nome dos clientes.

Dica: Utilize a cláusula *group by*.

```
SELECT Cliente.nome, SUM(Pagamento.valorTotal) AS valor_devido
FROM Cliente
JOIN Locacao ON Cliente.idCliente = Locacao.idCliente
JOIN Pagamento ON Locacao.idPagamento = Pagamento.idPagamento
WHERE Pagamento.estado = 'Pendente'
GROUP BY Cliente.idCliente
```

ORDER BY Cliente.nome ASC;

Query 1 x

Limit to 1000 rows

```
1 • SELECT Cliente.nome, SUM(Pagamento.valorTotal) AS valor_devido
2 FROM Cliente
3 JOIN Locacao ON Cliente.idCliente = Locacao.idCliente
4 JOIN Pagamento ON Locacao.idPagamento = Pagamento.idPagamento
5 WHERE Pagamento.estado = 'Pendente'
6 GROUP BY Cliente.idCliente
7 ORDER BY Cliente.nome ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	nome	valor_devido
▶	João da Silva	880.00
	Lucas Martins	2220.00
	Pedro dos Santos	280.00

Result Grid
Form Editor
Field Types