

Fundação Universidade Federal do Vale do São Francisco

Campus Juazeiro

Curso de Engenharia da Computação

Breno Gabriel de Souza Coelho  
Juan Henryco do Carmo Costa  
Maria Clara Mendes Da Silva

**Sistema de Banco de Dados para um Rede Hoteleira**

Juazeiro, BA  
2022

Breno Gabriel de Souza Coelho

Juan Henryco do Carmo Costa  
Maria Clara Mendes Da Silva

**Sistema de Banco de Dados para um Rede Hoteleira**

Projeto desenvolvido para obtenção de nota referente à disciplina de “Banco de Dados 2”, ministrada pelo prof. dr. Mário Godoy Neto, do curso de Engenharia de Computação, na Universidade Federal do Vale do São Francisco (UNIVASF)

Juazeiro, BA  
2022

1. **Concepção do Projeto e Regras do Negócio**

O objetivo do nosso trabalho é construir um banco de dados capaz de atender as necessidades gerais de uma rede hoteleira, permitindo o cadastro de clientes, análise da situação dos quartos, visão geral da ocupação dos diferentes hotéis presentes na rede, acesso às informações de interesse sobre funcionários e dados semelhantes.

Abaixo apresentamos uma definição das regras do negócio imaginado:

1. Ao fazer o cadastro, uma família, casal ou cliente sozinho registram sua estadia em nome de uma única pessoa (somente um cliente para todos os contratantes).
2. Todo quarto só pode ser alugado por um único cliente, para um mesmo período.
3. Todos os clientes alugam somente um quarto simultaneamente. Aluguel múltiplo exige o cadastro de um outro cliente.
4. Os quartos podem ser de casal, familiares ou para solteiros.
5. Todos os aluguéis tem uma data de início e fim, e o valor pago é calculado de acordo com a diária do quarto e a condição do cliente. Clientes VIP possuem um abatimento de 10% do valor a ser pago.
6. Cada quarto possui alguns produtos cujo uso é cobrado. É preciso poder registrar quais produtos foram consumidos sob nome de um determinado cliente, e qual o valor total que deve ser cobrado por isso.
7. É preciso atualizar a quantidade de produto presente em um determinado quarto toda vez que ele for consumido.
8. Cada quarto pertence a um dos hotéis da rede. Cada hotel possui uma série de funcionários. É preciso poder distinguir a qual hotel cada funcionário pertence.
9. Todos os funcionários trabalham em somente um hotel, e possuem um único cargo. Cada cargo possui um salário fixo.
10. Um hotel pode possuir vários quartos, porém cada quarto está associado a somente um hotel.

Essas foram as regras gerais acordadas num primeiro momento. Entende-se que em conversas posteriores alguns detalhes a mais foram determinados, como quais dados exatamente precisam ser armazenados dos clientes, funcionários, quartos, hóteis e afins. Com base nessas regras de negócio, e no acerto de detalhes posterior, construímos o banco de dados.

1. **Diagrama Entidade Relacionamento (DER)**
2. **Diagrama Relacional (DR)**
3. **Código de implementação do banco em PostgreSQL**

CREATE DATABASE rede\_hoteleira;

//criando a tabela dos clientes e colocando as restrições adequadas

CREATE TABLE clientes (

ID\_cliente SERIAL PRIMARY KEY,

CPF VARCHAR(11) UNIQUE,

nome VARCHAR(45) NOT NULL,

tipo boolean DEFAULT false

);

//inserindo alguns clientes

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('11122233344', 'João Coelho Silva Neto', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('99988877766', 'Jose da Silva Sampaio Junior', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('77766655544', 'Luiza Maria de Souza', true);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('22233344455', 'Breno Gabriel de Souza', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('12345678911', 'Maria Clara Mendes', true);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('33322244455', 'Rodrigo Rodrigues Pinto', true);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('85865625211', 'Junior Rios de Pires', true);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('41423256577', 'Juan Henryco de Carmo', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('68878897755', 'Felipe da Silva Junior', true);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('95945485877', 'Fernando Bruno Nascimento Filho', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('22115533664', 'Felipe Gabriel Campos Salles', true);

/\*Somente letras podem estar presentes no campo “nome”\*/

ALTER TABLE clientes

ADD CONSTRAINT nome\_only\_letters CHECK (nome ~ '^[A-Z, a-z]');

/\*só que essas condições possuem um erro. Eu preciso adicionar '+$' ao final do parágrafo para que a análise se aplique a todo o texto. Portanto as seguintes adições seriam válidas\*/

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('2a11233344', 'João Jorge Silva Neto', false);

INSERT INTO clientes(CPF, nome, tipo)

VALUES ('2811233344', 'João 14$%@o Silva Neto', false);

/\*vamos remover as constraints e corrigir isso\*/

ALTER TABLE clientes

DROP CONSTRAINT cpf\_only\_numbers;

ALTER TABLE clientes

DROP CONSTRAINT cpf\_only\_numbers;

/\*deletar os clientes incorretos\*/

DELETE FROM clientes

WHERE cpf = '2a11233344';

DELETE FROM clientes

WHERE cpf = '2811233344';

/\*adicionar as constraints certas\*/

ALTER TABLE clientes

ADD CONSTRAINT cpf\_only\_numbers CHECK (cpf ~ '^[0-9]+$');

ALTER TABLE clientes

ADD CONSTRAINT nome\_only\_letters CHECK (nome ~ '^[A-Z,a-z, ,À-û]+$');

/\*Essa parte dos nomes é meio complicada, porque os nomes podem ter acento. Eu adicionei um intervalo do UTF-8 que engloba praticamente todos os acentos latinos, mas ele também incorpora alguns poucos caracteres desagradáveis, que, no entanto, dificilmente serão usados por alguém, como Æ Ð Ø Þ e alguns outros poucos\*/

/\*Criando a tabela de endereços\*/

CREATE TABLE enderecos(

ID\_endereco SERIAL PRIMARY KEY,

cidade VARCHAR(20) NOT NULL,

estado VARCHAR(2) NOT NULL,

bairro VARCHAR(50) NOT NULL,

rua VARCHAR(50) NOT NULL,

numero int NOT NULL,

CHECK(cidade ~ '^[A-Z, a-z, , À-û]+$'),

CHECK(estado ~ '^[A-Z, a-z, , À-û]+$'),

CHECK(bairro ~ '^[A-Z, a-z, , À-û, 0-9]+$'),

CHECK(rua ~ '^[A-Z, a-z, , À-û, 0-9]+$')

);

/\*adicionando alguns endereços\*/

INSERT INTO enderecos(cidade, estado, bairro, rua, numero)

VALUES ('Petrolina', 'PE', 'Areia Branca', 'Rua Arco Verde', 902);

INSERT INTO enderecos(cidade, estado, bairro, rua, numero)

VALUES ('São Paulo', 'SP', 'Bairro X', 'Rua Felipe 2', 202);

INSERT INTO enderecos(cidade, estado, bairro, rua, numero)

VALUES ('Petrolina', 'PE', 'Orla', 'Rua Coelho', 86);

INSERT INTO enderecos(cidade, estado, bairro, rua, numero)

VALUES ('Recife', 'PE', 'Madalena', 'Rua José Sarney', 100);

/\*criando a tabela de hóteis\*/

CREATE TABLE hoteis(

ID\_hotel SERIAL PRIMARY KEY,

nome VARCHAR(45) UNIQUE NOT NULL,

tipo VARCHAR(12) NOT NULL DEFAULT 'comum',

lot\_max int NOT NULL DEFAULT 200,

endereco SERIAL,

CONSTRAINT fk\_hoteis\_endereco FOREIGN KEY(endereco) REFERENCES enderecos(ID\_endereco),

CHECK(nome ~ '^[A-Z, a-z, , À-û]+$'),

CHECK(tipo IN ('Comum', 'Luxo', 'Pousada', '5 estrelas')),

CHECK(lot\_max >= 200)

);

/\*adicionando alguns hoteis e colocando seus endereços\*/

INSERT INTO hoteis(nome, tipo, lot\_max, endereco)

VALUES ('Rio Grande', 'Comum', 300, 1);

INSERT INTO hoteis(nome, tipo, endereco)

VALUES ('Cidade Maravilhosa', 'Pousada', 2);

INSERT INTO hoteis(nome, tipo, lot\_max, endereco)

VALUES ('Águia dos Reis', 'Luxo', 450, 3);

INSERT INTO hoteis(nome, tipo, lot\_max, endereco)

VALUES ('Águas Claras', '5 estrelas', 500, 4);

/\*Criando a tabela de quartos\*/

CREATE TABLE quartos(

ID\_quarto SERIAL,

num\_quarto int NOT NULL,

tipo VARCHAR(10),

valor\_diaria REAL NOT NULL,

hotel SERIAL,

CONSTRAINT unique\_quarto\_hotel UNIQUE(num\_quarto, hotel),

CONSTRAINT fk\_hoteis\_quartos FOREIGN KEY(hotel) REFERENCES hoteis(ID\_hotel),

CHECK(LOWER(tipo) IN ('casal','familia','solteiro')),

CHECK(valor\_diaria > 0.0000)

);

ALTER TABLE quartos

ADD CONSTRAINT pk\_quartos PRIMARY KEY(id\_quarto);

/\*inserindo alguns quartos no primeiro hotel (que ficou com pk = 2 por alguma razão)\*/

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0001, 'solteiro', 100, 2);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0002, 'familia', 200, 2);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1001, 'familia', 250, 2);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1002, 'casal', 200, 2);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2001, 'solteiro', 50, 2);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2002, 'casal', 120, 2);

/\*inserindo quartos no segundo hotel\*/

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0001, 'solteiro', 20, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0002, 'solteiro', 20, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0003, 'solteiro', 20, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0004, 'solteiro', 20, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0005, 'casal', 50, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0006, 'casal', 50, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0007, 'casal', 50, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0008, 'casal', 50, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1001, 'familia', 100, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1002, 'familia', 120, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1003, 'familia', 80, 3);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1004, 'familia', 120, 3);

/\*inserindo quartos no terceiro hotel\*/

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0001, 'solteiro', 400, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0002, 'solteiro', 400, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1001, 'solteiro', 400, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1002, 'familia', 800, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2001, 'casal', 500, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2002, 'familia', 800, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (3001, 'casal', 600, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (3002, 'casal', 600, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (4001, 'familia', 900, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (4002, 'familia', 900, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (5001, 'casal', 700, 4);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (5002, 'casal', 650, 4);

/\*inserindo quartos no quarto hotel\*/

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0001, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (0002, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1001, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (1002, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2001, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (2002, 'solteiro', 2000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (3001, 'casal', 4000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (3002, 'casal', 4000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (4001, 'familia', 5000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (4002, 'familia', 5000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (5001, 'casal', 6000, 5);

INSERT INTO quartos(num\_quarto, tipo, valor\_diaria, hotel)

VALUES (5002, 'casal', 6000, 5);

ALTER TABLE hoteis

DROP CONSTRAINT hoteis\_lot\_max\_check;

ALTER TABLE hoteis

ADD CONSTRAINT hoteis\_lot\_max\_check CHECK(lot\_max > 0);

UPDATE hoteis

SET lot\_max = 30

WHERE id\_hotel = 2;

UPDATE hoteis

SET lot\_max = 20

WHERE id\_hotel = 3;

UPDATE hoteis

SET lot\_max = 50

WHERE id\_hotel = 2;

UPDATE hoteis

SET lot\_max = 60

WHERE id\_hotel = 2;

/\*criando a tabela de produtos\*/

CREATE TABLE produtos(

ID\_produto SERIAL PRIMARY KEY,

nome VARCHAR(50) UNIQUE NOT NULL,

marca VARCHAR(50),

preco REAL NOT NULL,

CHECK(nome ~ '^[A-Z, a-z, , À-û, 0-9]+$'),

CHECK(marca ~ '^[A-Z, a-z, , À-û, 0-9]+$'),

CHECK(preco > 0)

);

/\*inserindo alguns produtos\*/

INSERT INTO produtos (nome, marca, preco)

VALUES ('Barra de Chocolar Herkeys', 'Herkeys', 6);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Latinha Cosa Loca', 'Cosa Loca', 5);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Latinha Sprito', 'Sprito', 5);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Latinha Fantos', 'Fantos', 5);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Salgadinho Boritos Grande', 'Velma Chips', 8);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Salgadinho Boritos Pequeno', 'Velma Chips', 4);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Loção Especial Naturo', 'Naturo', 12);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Pacote de TV a cabo Padrão', 'TVS', 22);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Pacote de TV a cabo Padrão', 'TVS', 22);

INSERT INTO produtos (nome, marca, preco)

VALUES ('Pacote de TV a cabo Completo', 'TVS', 56);

SELECT \* FROM produtos;

/\*criando a relação entre quartos e produtos\*/

CREATE TABLE r\_quartos\_produtos(

fk\_quarto SERIAL,

fk\_produto SERIAL,

qtd\_produto int,

CONSTRAINT r\_fk\_quarto FOREIGN KEY(fk\_quarto) REFERENCES quartos(id\_quarto),

CONSTRAINT r\_fk\_produto FOREIGN KEY(fk\_produto) REFERENCES produtos(id\_produto),

CHECK (qtd\_produto >= 0)

);

/\*inserindo alguns produtos. Para evitar uma poluição visual, adicionei apenas em

alguns quartos\*/

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 2, 10);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 2, 10);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 3, 12);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 4, 5);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 5, 3);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 6, 3);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 7, 2);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 8, 2);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (1, 2, 10);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (2, 3, 12);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (2, 4, 5);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (2, 5, 3);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (3, 5, 3);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (3, 6, 3);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (4, 7, 2);

INSERT INTO r\_quartos\_produtos(fk\_quarto, fk\_produto, qtd\_produto)

VALUES (4, 2, 12);

/\*criando a tabela de cargos\*/

CREATE TABLE cargos(

ID\_cargo SERIAL PRIMARY KEY,

nome VARCHAR(45),

salario REAL,

area VARCHAR(45),

CHECK(salario > 0),

CHECK(nome ~ '^[A-Z, a-z, , À-û]+$'),

CHECK(area ~ '^[A-Z, a-z, , À-û]+$')

);

/\*adicionando alguns cargos\*/

INSERT INTO cargos(nome, salario, area)

VALUES ('Porteiro', 2100.00, 'Segurança');

INSERT INTO cargos(nome, salario, area)

VALUES ('Vigia Noturno', 4500.00, 'Segurança');

INSERT INTO cargos(nome, salario, area)

VALUES ('Atendente', 2200.00, 'Serviço');

INSERT INTO cargos(nome, salario, area)

VALUES ('Faxineira', 1700.00, 'Serviço');

INSERT INTO cargos(nome, salario, area)

VALUES ('Faxineira', 1700.00, 'Serviço');

INSERT INTO cargos(nome, salario, area)

VALUES ('Cozinheiro', 3500.00, 'Alimentação');

INSERT INTO cargos(nome, salario, area)

VALUES ('Mestre de Cozinha', 7800.00, 'Alimentação');

INSERT INTO cargos(nome, salario, area)

VALUES ('Supervisor', 3100.00, 'Admnistração');

INSERT INTO cargos(nome, salario, area)

VALUES ('Gerente', 6200.00, 'Admnistração');

INSERT INTO cargos(nome, salario, area)

VALUES ('Manobrista', 2500.00, 'Serviços');

INSERT INTO cargos(nome, salario, area)

VALUES ('Superintendente Regional', 12000.00, 'Administração');

INSERT INTO cargos(nome, salario, area)

VALUES ('Técnico Elétrica', 4200.00, 'Manutenção');

INSERT INTO cargos(nome, salario, area)

VALUES ('Técnico Hídrico e Construção', 2500.00, 'Manutenção');

INSERT INTO cargos(nome, salario, area)

VALUES ('Engenheiro de Computação', 5700.00, 'Manutenção');

/\*criando a tabela de funcionários\*/

CREATE TABLE funcionarios(

ID\_funcionario SERIAL PRIMARY KEY,

nome VARCHAR(45) NOT NULL,

CPF varchar(11) NOT NULL UNIQUE,

tempo\_trabalho int,

hotel\_contratante SERIAL,

cargo SERIAL,

CHECK(tempo\_trabalho > 0),

CHECK(nome ~ '^[A-Z, a-z, , À-û]+$'),

CONSTRAINT fk\_hotel FOREIGN KEY(hotel\_contratante) REFERENCES hoteis(id\_hotel),

CONSTRAINT fk\_cargo FOREIGN KEY(cargo) REFERENCES cargos(id\_cargo)

);

/\*adicionando alguns funcionários\*/

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Maria Suzana Nascimento', '88811245366', 40, 2, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Valéria Regina da Silva', '22211245366', 40, 2, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Jorge Prado do Vale', '33311245366', 40, 2, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Filomena do Nascimento Jesus', '44411245366', 40, 2, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Julia Maria dos Santos', '88815555366', 40, 3, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Pedro Cavalcanti Prado', '22216665366', 40, 3, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('João de Orvalho Peres', '33311232366', 40, 3, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Luiza Takahashi', '44411415366', 40, 3, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('José Pereira Ferreira Costa Rodrigues ', '88815511266', 40, 5, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Regina Pereira de Santos Pinto', '22236665366', 40, 5, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Breno da Silva Rodrigues', '33388932366', 40, 5, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Luiza Alves de Carvalho', '44557415366', 40, 5, 4);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Pedro de Pinto Souza', '88454511266', 30, 5, 1);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Coelho Salles Rodrigues', '22236667426', 30, 3, 5);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Luiza Oliveira Silva', '39998932366', 35, 2, 8);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Pedro Araújo Oliveira', '44557499966', 40, 2, 9);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Enzo Moraes Silveira', '88451231266', 35, 2, 13);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Carol Oliveira dos Santos', '25586665366', 40, 3, 10);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Breno Gabriel Silveira Salles', '33318932366', 40, 4, 7);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Erik Vanchestofen Coelho', '44557455266', 40, 4, 6);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Gildete Estrela Silveira', '86664511266', 20, 2, 7);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Bruno Campos dos Santos Lima', '22236623666', 35, 2, 7);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Mario Ramos de Souza', '33454932366', 40, 4, 7);

INSERT INTO funcionarios(nome, cpf, tempo\_trabalho, hotel\_contratante, cargo)

VALUES ('Rafael Cordeiro de Souza', '49985415366', 22, 2, 8);

UPDATE cargos

SET nome = 'Faxineiro'

WHERE nome = 'Faxineira';

/\*Criando a tabela de Alugueis\*/

CREATE TABLE alugueis(

fk\_quartos SERIAL,

fk\_clientes SERIAL,

data\_in DATE NOT NULL,

data\_fim DATE NOT NULL,

num\_pessoas INT NOT NULL,

valor\_aluguel REAL,

CHECK(valor\_aluguel >= 0),

CHECK(num\_pessoas > 0),

CONSTRAINT fk\_quartos FOREIGN KEY(fk\_quartos) REFERENCES quartos(id\_quarto),

CONSTRAINT fk\_clientes FOREIGN KEY(fk\_clientes) REFERENCES clientes(id\_cliente),

CHECK(data\_in < data\_fim)

);

/\*adicionando alguns alugueis\*/

INSERT INTO alugueis(fk\_quartos, fk\_clientes, data\_in, data\_fim, num\_pessoas)

VALUES (2, 5, '12/02/1990', '15/02/1990', 1);

INSERT INTO alugueis(fk\_quartos, fk\_clientes, data\_in, data\_fim, num\_pessoas)

VALUES (2, 3, '18/03/1990', '26/03/1990', 4);

INSERT INTO alugueis(fk\_quartos, fk\_clientes, data\_in, data\_fim, num\_pessoas)

VALUES (3, 4, '20/05/1990', '26/05/1990', 2);

INSERT INTO alugueis(fk\_quartos, fk\_clientes, data\_in, data\_fim, num\_pessoas)

VALUES (6, 6, '27/05/1990', '30/05/1990', 2);

INSERT INTO alugueis(fk\_quartos, fk\_clientes, data\_in, data\_fim, num\_pessoas)

VALUES (5, 3, '20/06/1990', '02/07/1990', 5);

CREATE TABLE consumiu(

fk\_cliente SERIAL,

fk\_produto SERIAL,

qtd\_consumida INT,

CHECK (qtd\_consumida > 0),

CONSTRAINT r\_fk\_cliente FOREIGN KEY(fk\_cliente) REFERENCES clientes(id\_cliente),

CONSTRAINT r\_fk\_produto FOREIGN KEY(fk\_produto) REFERENCES produtos(id\_produto)

);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (1, 1, 2);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (1, 2, 1);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (1, 5, 5);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (1, 4, 2);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (7, 7, 1);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (3, 8, 1);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (3, 6, 2);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (4, 9, 1);

INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (5, 3, 5);

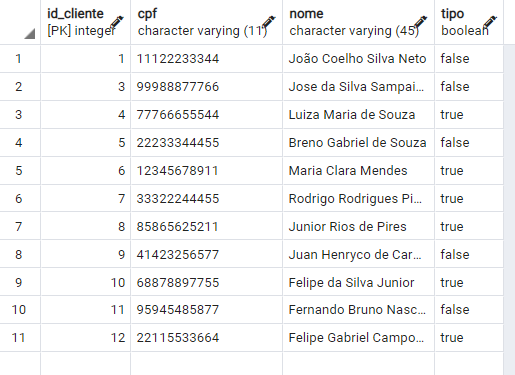
INSERT INTO consumiu (fk\_cliente, fk\_produto, qtd\_consumida)

VALUES (5, 1, 12);

1. **Mostrando os diferentes comandos da linguagem no banco criado**

/\*Mostrando os registros presentes na tabela de clientes\*/

SELECT \* FROM clientes;



//Fazendo algumas manipulações simples agora com a tabela de clientes

//Apresentando todos os nomes em ordem alfabética

SELECT \* FROM clientes ORDER BY nome;



//Vendo quantos clientes possuem conta VIP e quantos não

SELECT tipo as VIP, COUNT(tipo) FROM clientes GROUP BY tipo;

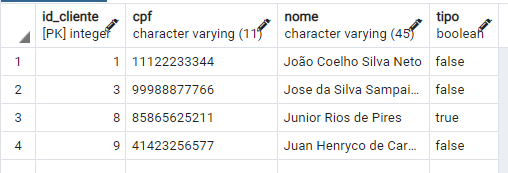
(true = é vip; false = não é vip)



//Algumas seleções

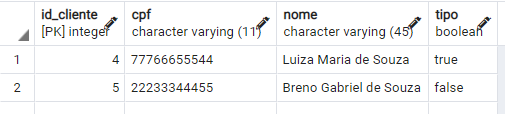
//Clientes com letra J inicial

SELECT \* FROM clientes WHERE nome LIKE 'J%';



//Clientes com 'Souza' no nome

SELECT \* FROM clientes WHERE nome LIKE '%Souza%';

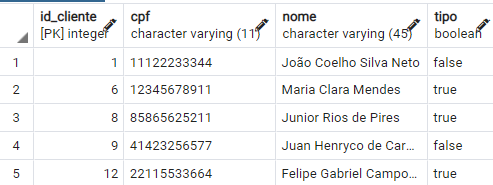


//Todos que não possuem 'Coelho' no nome

SELECT \* FROM clientes WHERE nome NOT LIKE '%Coelho%';

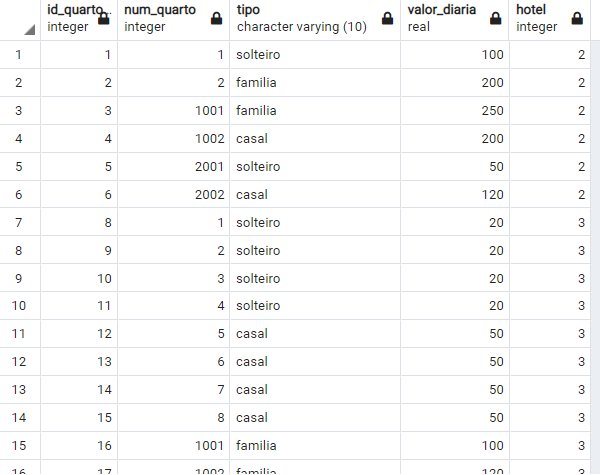
//Todos que possuem '1' no CPF que não é o primeiro número

SELECT \* FROM cliente WHERE cpf LIKE '\_%1%';



/\*Vendo alguns dos quartos\*/

SELECT \* FROM quartos;



/\* Vendo o total de quartos já adicionados em cada hotel, e apresentando o nome do hotel\*/

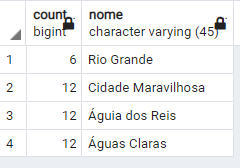
SELECT COUNT(tb1.num\_quarto), tb2.nome

FROM quartos tb1

INNER JOIN hoteis tb2

ON tb1.hotel = tb2.id\_hotel

GROUP BY tb2.nome;



/\*Ver o preço médio dos quartos em cada hotel\*/

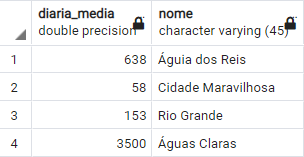
SELECT ROUND(AVG(tb1.valor\_diaria)) as diaria\_media, tb2.nome

FROM quartos tb1

INNER JOIN hoteis tb2

ON tb1.hotel = tb2.id\_hotel

GROUP BY tb2.nome;



/\*Ver o preço médio de cada tipo de quarto para o hotel 4\*/

SELECT tb1.tipo, AVG(tb1.valor\_diaria)

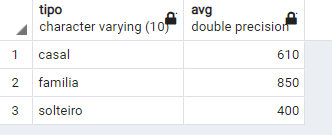
FROM quartos tb1

INNER JOIN hoteis tb2

ON tb1.hotel = tb2.id\_hotel

WHERE tb2.id\_hotel = 4

GROUP BY tb1.tipo;



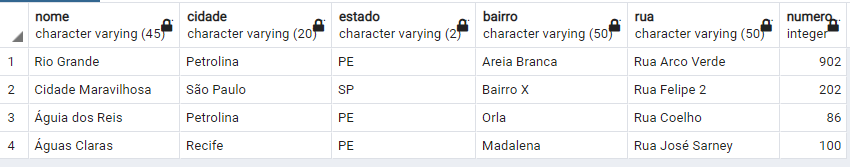
/\*Apresentando uma tabela com o nome do hotel e os dados de seu endereço\*/

SELECT tb1.nome, tb2.cidade, tb2.estado, tb2.bairro, tb2.rua, tb2.numero

FROM hoteis tb1

LEFT JOIN enderecos tb2

ON tb1.endereco = tb2.id\_endereco;



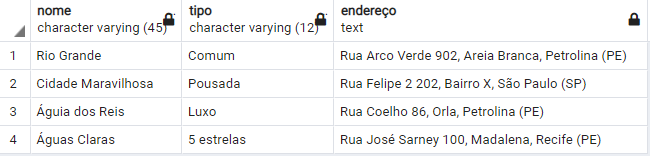
/\*Apresentando uma tabela com o nome do hotel e o seu endereço, porém com o endereço estando em uma coluna só e mais bem organizado\*/

SELECT h.nome, h.tipo, CONCAT(e.rua, ' ', e.numero::text, ', ', e.bairro, ', ', e.cidade, ' (', e.estado, ')') as Endereço

FROM enderecos e

RIGHT JOIN hoteis h

ON e.id\_endereco = h.endereco;



/\*Total de quartos vagos em cada hotel\*/

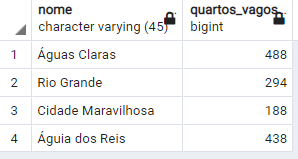
SELECT tb2.nome, tb2.lot\_max - COUNT(tb1.num\_quarto) as Quartos\_Vagos

FROM quartos tb1

INNER JOIN hoteis tb2

ON tb1.hotel = tb2.id\_hotel

GROUP BY tb2.nome, tb2.lot\_max;



/\*acabei colocando o total de quartos disponível em cada hotel como um valor grande demais. Vamos diminuir isso\*/

/\*Infelizmente eu coloquei uma constraint que restringe o tamanho mínimo a 200. Precisamos primeiro arrancar isso fora\*/

ALTER TABLE hoteis

DROP CONSTRAINT hoteis\_lot\_max\_check;

ALTER TABLE hoteis

ADD CONSTRAINT hoteis\_lot\_max\_check CHECK(lot\_max > 0);

/\*agora mudando os valores de lotação máxima…\*/

UPDATE hoteis

SET lot\_max = 30

WHERE id\_hotel = 2;

UPDATE hoteis

SET lot\_max = 20

WHERE id\_hotel = 3;

UPDATE hoteis

SET lot\_max = 50

WHERE id\_hotel = 4;

UPDATE hoteis

SET lot\_max = 60

WHERE id\_hotel = 5;

/\*Agora vamos ver o máximo de cada um, assim como o total de quartos vagos e ocupados, e colocar também o tipo de quarto\*/

SELECT tb2.nome, tb2.tipo, tb2.lot\_max as Total\_de\_Quartos,

tb2.lot\_max - COUNT(tb1.num\_quarto) as Quartos\_Vagos,

COUNT(tb1.num\_quarto) as Quartos\_Ocupados

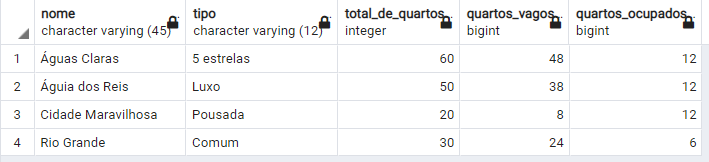
FROM quartos tb1

INNER JOIN hoteis tb2

ON tb1.hotel = tb2.id\_hotel

GROUP BY tb2.nome, tb2.tipo, tb2.lot\_max

ORDER BY nome ASC;

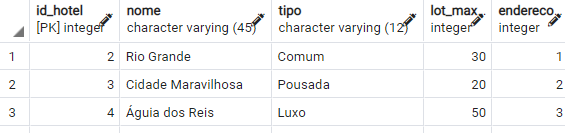


/\*Quais os hoteis com lotação máxima entre 20 e 50 pessoas?\*/

SELECT \*

FROM hoteis

WHERE lot\_max BETWEEN 20 and 50;

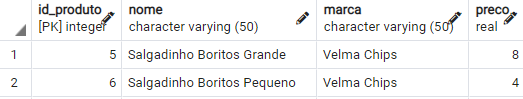


/\*Mostrar uma lista com todos os produtos da “Velma Chips” \*/

SELECT \*

FROM produtos

WHERE UPPER(marca) = 'VELMA CHIPS';

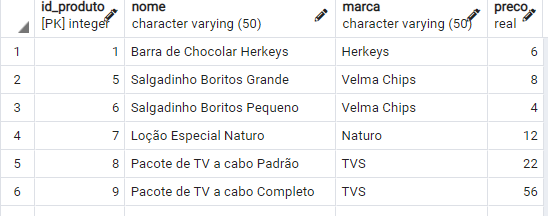


/\*Quais produtos não são refrigerantes (não têm “latinha” no nome?\*/

SELECT \*

FROM produtos

WHERE UPPER(nome) NOT LIKE '%LATINHA%';



/\*Apresentando os produtos com o nome seguido pela marca, e com o preço junto ao símbolo “R$”\*/

SELECT CONCAT(nome, ', ', marca) as produto, CONCAT('R$ ', preco) as preço

FROM produtos;



/\*Ver quantos produtos tem em cada quarto, com o número do quarto, hotel ao qual pertence e nome do produto\*/

SELECT LPAD(q.num\_quarto::text, 4, '0') as num\_quarto,

h.nome as hotel, prd.nome as produto, r.qtd\_produto

FROM (quartos q

INNER JOIN r\_quartos\_produtos r

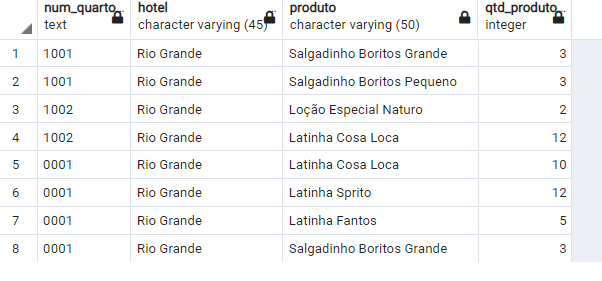
ON q.id\_quarto = r.fk\_quarto

INNER JOIN produtos prd

ON prd.id\_produto = r.fk\_produto)

INNER JOIN hoteis h

ON h.id\_hotel = q.hotel;



(todos os quartos em que coloquei produtos são do mesmo hotel, o Rio Grande)

/\*Quantos quartos ficaram sem produtos? Vamos verificar\*/

SELECT COUNT(\*) FROM

r\_quartos\_produtos r RIGHT JOIN quartos q

ON r.fk\_quarto = q.id\_quarto

WHERE fk\_quarto IS NULL ;



/\*Qual o produto mais barato? E qual o mais caro?\*/

SELECT MIN(prd.preco) as Preço\_Minimo\_e\_Maximo

FROM produtos prd

UNION

SELECT MAX(prd.preco)

FROM produtos prd;



/\*Quais os hotéis com quartos mais baratos de 500R$ e quantos quartos tem esse preço?\*/

SELECT h.nome, COUNT(q.id\_quarto) FROM

hoteis h RIGHT JOIN quartos q

ON h.id\_hotel = q.hotel

WHERE q.valor\_diaria <= 500

GROUP BY h.nome;



/\*Qual o gasto total que a empresa tem com seus funcionários?\*/

SELECT SUM(cr.salario) as Despesas\_Funcionarios

FROM funcionarios f

INNER JOIN cargos cr

ON f.cargo = cr.id\_cargo;

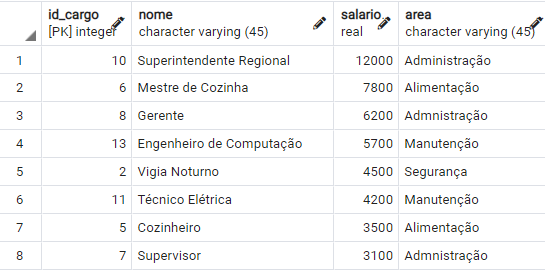


/\*Vamos organizar todos os cargos em ordem de descendente\*/

SELECT \*

FROM cargos

ORDER BY salario DESC;



/\*Calcular o custo total de cada area de contratação\*/

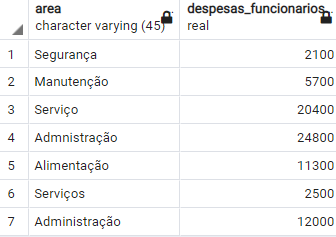
SELECT cr.area, SUM(cr.salario) as Despesas\_Funcionarios

FROM funcionarios f

INNER JOIN cargos cr

ON f.cargo = cr.id\_cargo

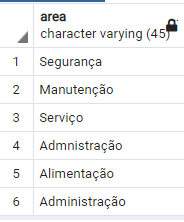
GROUP BY cr.area;



/\*Apresentando os cargos que possuem uma letra ‘a’ ou uma letra ‘e’\*/

SELECT DISTINCT area FROM cargos

WHERE LOWER(area) LIKE '%a%' OR LOWER(nome) LIKE '%e';



/\*Apresentando o nome de todas as pessoas presentes no banco de dados (entre clientes e funcionários) \*/

SELECT cl.nome

FROM clientes cl

UNION

SELECT f.nome

FROM funcionarios f;



/\*Existem funcionários que também são clientes? Quais são?\*/

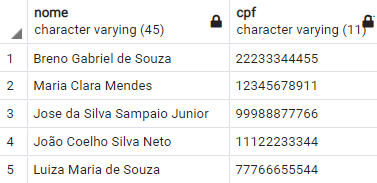
SELECT cl.nome, cl.cpf

FROM clientes cl

INTERSECT

SELECT f.nome, f.cpf

FROM funcionarios f;



/\*Qual o salário médio por área de atuação dos funcionários?\*/

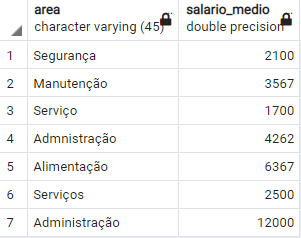
SELECT cr.area, ROUND(AVG(cr.salario)) as Salario\_Medio

FROM funcionarios f

INNER JOIN cargos cr

ON f.cargo = cr.id\_cargo

GROUP BY cr.area;



/\*Quais funcionários possuem um salário acima de 3000R$? Qual o valor desse salário e o nome do hotel no qual trabalham?\*/

SELECT f.nome, h.nome, cargos.salario

FROM funcionarios f

INNER JOIN hoteis h

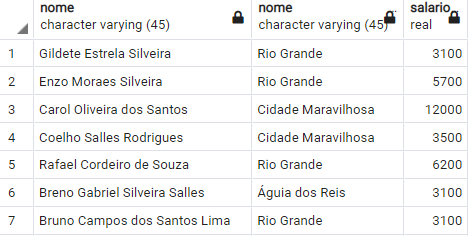
ON f.hotel\_contratante = h.id\_hotel

INNER JOIN cargos

ON f.cargo = cargos.id\_cargo

GROUP BY f.nome, h.nome, cargos.salario

HAVING cargos.salario > 3000;



/\*Qual a média do tamanho dos nomes dos funcionários?\*/

SELECT TRUNC(AVG(LENGTH(f.nome)))

FROM funcionarios f;



/\*Trocando o cargo ‘faxineiras’ para ‘faxineiros’ a fim de usar um termo neutro\*/

UPDATE cargos

SET nome = 'Faxineiro'

WHERE nome = 'Faxineira';

/\*Criando uma visão para poder visualizar todos os funcionários que são ‘faxineiros’\*/

CREATE VIEW geral\_funcionario(funcionario, cpf, hotel, cargo) AS

SELECT f.nome, f.cpf, h.nome, cr.nome

FROM funcionarios f

INNER JOIN hoteis h

ON f.hotel\_contratante = h.id\_hotel

INNER JOIN cargos cr

ON f.cargo = cr.id\_cargo;

SELECT \* FROM geral\_funcionario WHERE cargo = 'Faxineiro';



/\*Criando uma visão para visualizar quais produtos cada cliente comeu\*/

CREATE VIEW consumo\_cliente(cliente, produto, quantidade) AS

SELECT cli.nome, prd.nome, qtd\_consumida

FROM consumiu cns

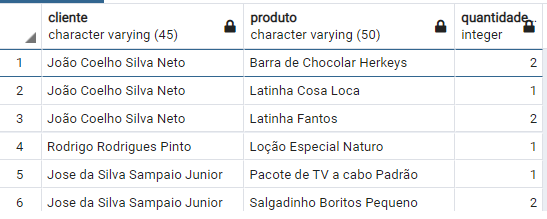
INNER JOIN clientes cli

ON cns.fk\_cliente = cli.id\_cliente

INNER JOIN produtos prd

ON cns.fk\_produto = prd.id\_produto;

SELECT \* FROM consumo\_cliente;



/\*Exemplo simples de UNION ALL com os nomes de produtos e de clientes\*/

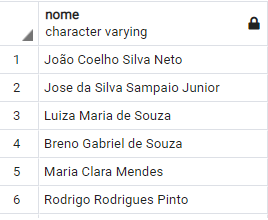
SELECT nome

FROM clientes

UNION ALL

SELECT nome

FROM produtos;



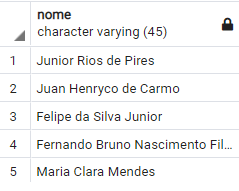
/\*Exemplo geral de INTERCEPT

/\*Quais clientes não comeram nada?\*/

SELECT nome FROM clientes

EXCEPT

SELECT cliente FROM consumo\_cliente



/\*Unindo isso com uma busca só pelos clientes distintos da visão, voltamos a tabela de clientes original\*/

SELECT nome FROM clientes

EXCEPT

SELECT cliente FROM consumo\_cliente

UNION ALL

SELECT DISTINCT cliente FROM consumo\_cliente;



/\*Apresentando os clientes e seus respectivos quartos\*/

SELECT cli.nome, LPAD(num\_quarto::text, 4, '0') as quarto

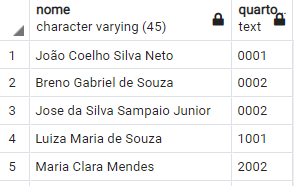
FROM clientes cli

INNER JOIN alugueis al

ON cli.id\_cliente = al.fk\_clientes

INNER JOIN quartos q

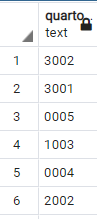
ON q.id\_quarto = al.fk\_quartos;



/\*Apresentando todos os quartos não repetidos\*/

SELECT DISTINCT LPAD(num\_quarto::text, 4, '0') as quarto

FROM quartos;



/\*Usando PL para mostrar a quantidade de registros de cargos e apresentar quais os que possuem menor e maior salários\*/

DO $$

DECLARE

varQtd INT;

varNome VARCHAR(50);

BEGIN

SELECT COUNT(\*)

INTO varQtd

FROM cargos;

RAISE NOTICE 'Quantidade de Cargos : %', varQtd;

SELECT nome, salario

INTO varNome, varQtd

FROM cargos ORDER BY salario ASC FETCH FIRST ROW ONLY;

RAISE NOTICE 'O cargo % recebe o menor salario, de %', varNome, varQtd;

SELECT nome, salario

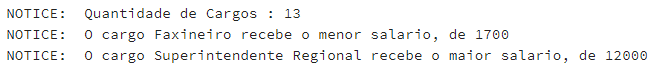
INTO varNome, varQtd

FROM cargos ORDER BY salario DESC FETCH FIRST ROW ONLY;

RAISE NOTICE 'O cargo % recebe o maior salario, de %', varNome, varQtd;

END;

$$;



/\*Usando PL para mostrar todos os produtos da lista de produtos com preço menor ou igual à 10 R$\*/

DO $$

DECLARE

varProduto produtos%ROWTYPE;

N INT;

i INT DEFAULT 0;

BEGIN

SELECT COUNT(\*)

INTO N

FROM produtos;

LOOP

SELECT \*

INTO varProduto

FROM produtos

LIMIT 1 OFFSET i;

IF (varProduto.preco <= 10) THEN

RAISE NOTICE '% : %', i, varProduto;

END IF;

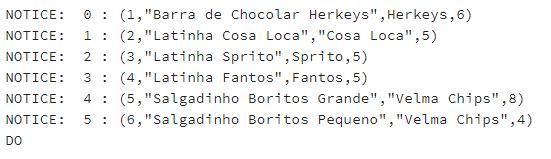
i = i + 1;

EXIT WHEN i = N;

END LOOP;

END;

$$;



/\*Criando procedimento que retorna todos os clientes que possuem o nome dado\*/

CREATE OR REPLACE PROCEDURE Buscar(

texto IN VARCHAR(50))

LANGUAGE plpgsql

AS $$

DECLARE

i INT := 0;

N INT;

bandeira BOOL := false;

varNome VARCHAR(50);

BEGIN

SELECT COUNT(\*)

INTO N

FROM clientes;

LOOP

--seleciona o nome

SELECT nome

INTO varNome

FROM clientes

LIMIT 1 OFFSET i;

texto := INITCAP(texto);

IF (INITCAP(varNome) LIKE CONCAT('%', texto, '%')) THEN

bandeira := true;

RAISE NOTICE '% : %', i::text, varNome;

END IF;

i := i + 1;

EXIT WHEN i = N;

END LOOP;

IF (bandeira = false) THEN

RAISE NOTICE 'Nenhum nome compatível';

END IF;

END;

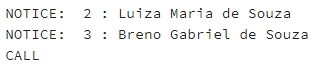
$$;

/\*Executando a procedure para buscar por alguns nomes\*/

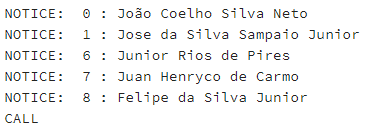
CALL buscar('BReNo');



CALL buscar('Souza');



CALL buscar('J');



/\*Criando um trigger para só permitir adicionar datas num intervalo não usado antes\*/

CREATE OR REPLACE FUNCTION verifica\_data()

RETURNS trigger

AS $$

DECLARE

i INT := 0;

N INT;

varData\_in DATE;

varData\_fim DATE;

BEGIN

SELECT COUNT(\*)

INTO n

FROM alugueis;

IF n = 0 THEN

RETURN NEW;

END IF;

WHILE (i < n) LOOP

SELECT data\_in, data\_fim

INTO varData\_in, varData\_fim

FROM alugueis

LIMIT 1 OFFSET i;

IF ((NEW.data\_fim BETWEEN varData\_in AND varData\_fim) OR

(NEW.data\_in BETWEEN varData\_in AND varData\_fim)) THEN

RAISE EXCEPTION 'Esse intervalo já foi alugado!';

RETURN OLD;

ELSE

RETURN NEW;

END IF;

i = i + 1;

END LOOP;

END;

$$ LANGUAGE plpgsql;

/\*Criando um procedimento para colocar o valor adequado dos alugueis de acordo com o tipo do cliente - comum ou vip - \*/

CREATE OR REPLACE FUNCTION determinar\_aluguel()

RETURNS TRIGGER

AS $$

DECLARE

varCliente BOOLEAN;

varValorDiaria REAL;

BEGIN

SELECT tipo

INTO varCliente

FROM clientes

WHERE id\_cliente = NEW.fk\_clientes;

SELECT valor\_diaria

INTO varValorDiaria

FROM quartos

WHERE id\_quarto = NEW.fk\_quartos;

IF (varCliente) THEN

NEW.valor\_aluguel = 0.9 \* varValorDiaria;

ELSE

NEW.valor\_aluguel = varValorDiaria;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER atualiza\_valor

BEFORE INSERT ON alugueis

FOR EACH ROW

EXECUTE PROCEDURE determinar\_aluguel();