



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, hotéis e afins. Com base nessas regras de negócio, e no acerto de detalhes posterior, construímos o banco de dados.

2. Diagrama Entidade Relacionamento (DER)

3. Diagrama Relacional (DR)

4. 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 hotéis*/
CREATE TABLE hotéis(
    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_hotéis_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 hotéis e colocando seus endereços*/
INSERT INTO hotéis(nome, tipo, lot_max, endereco)
VALUES ('Rio Grande', 'Comum', 300, 1);
INSERT INTO hotéis(nome, tipo, endereco)
VALUES ('Cidade Maravilhosa', 'Pousada', 2);
INSERT INTO hotéis(nome, tipo, lot_max, endereco)
VALUES ('Águia dos Reis', 'Luxo', 450, 3);
INSERT INTO hotéis(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, 'Administração');
INSERT INTO cargos(nome, salario, area)
VALUES ('Gerente', 6200.00, 'Administração');
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', '8881555366', 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);





```

5. Mostrando os diferentes comandos da linguagem no banco criado

```

/*Mostrando os registros presentes na tabela de clientes*/
SELECT * FROM clientes;

```

	 id_cliente [PK] integer	 cpf character varying (11)	 nome character varying (45)	 tipo boolean
1	1	11122233344	João Coelho Silva Neto	false
2	3	99988877766	Jose da Silva Sampai...	false
3	4	77766655544	Luiza Maria de Souza	true
4	5	22233344455	Breno Gabriel de Souza	false
5	6	12345678911	Maria Clara Mendes	true
6	7	33322244455	Rodrigo Rodrigues Pi...	true
7	8	85865625211	Junior Rios de Pires	true
8	9	41423256577	Juan Henryco de Car...	false
9	10	68878897755	Felipe da Silva Junior	true
10	11	95945485877	Fernando Bruno Nasc...	false
11	12	22115533664	Felipe Gabriel Campo...	true

//Fazendo algumas manipulações simples agora com a tabela de clientes
//Apresentando todos os nomes em ordem alfabética

SELECT * FROM clientes ORDER BY nome;

	 id_cliente [PK] integer	 cpf character varying (11)	 nome character varying (45)	 tipo boolean
1	5	22233344455	Breno Gabriel de Souza	false
2	10	68878897755	Felipe da Silva Junior	true
3	12	22115533664	Felipe Gabriel Campo...	true
4	11	95945485877	Fernando Bruno Nasc...	false
5	1	11122233344	João Coelho Silva Neto	false
6	3	99988877766	Jose da Silva Sampai...	false
7	9	41423256577	Juan Henryco de Car...	false
8	8	85865625211	Junior Rios de Pires	true
9	4	77766655544	Luiza Maria de Souza	true
10	6	12345678911	Maria Clara Mendes	true
11	7	33322244455	Rodrigo Rodrigues Pi...	true

//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)

	vip boolean	count bigint
1	false	5
2	true	6

//Algumas seleções

//Clientes com letra J inicial

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

	id_cliente [PK] integer	cpf character varying (11)	nome character varying (45)	tipo boolean
1	1	11122233344	João Coelho Silva Neto	false
2	3	99988877766	Jose da Silva Sampai...	false
3	8	85865625211	Junior Rios de Pires	true
4	9	41423256577	Juan Henryco de Car...	false

//Clientes com 'Souza' no nome

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

	id_cliente [PK] integer	cpf character varying (11)	nome character varying (45)	tipo boolean
1	4	77766655544	Luiza Maria de Souza	true
2	5	22233344455	Breno Gabriel de Souza	false

//Todos que não possuem 'Coelho' no nome

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

	id_cliente [PK] integer	cpf character varying (11)	nome character varying (45)	tipo boolean
1	3	99988877766	Jose da Silva Sampai...	false
2	4	77766655544	Luiza Maria de Souza	true
3	5	22233344455	Breno Gabriel de Souza	false
4	6	12345678911	Maria Clara Mendes	true
5	7	33322244455	Rodrigo Rodrigues Pi...	true
6	8	85865625211	Junior Rios de Pires	true
7	9	41423256577	Juan Henryco de Car...	false
8	10	68878897755	Felipe da Silva Junior	true
9	11	95945485877	Fernando Bruno Nasc...	false
10	12	22115533664	Felipe Gabriel Campo...	true

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

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

	id_cliente [PK] integer	cpf character varying (11)	nome character varying (45)	tipo boolean
1	1	11122233344	João Coelho Silva Neto	false
2	6	12345678911	Maria Clara Mendes	true
3	8	85865625211	Junior Rios de Pires	true
4	9	41423256577	Juan Henryco de Car...	false
5	12	22115533664	Felipe Gabriel Campo...	true

/*Vendo alguns dos quartos*/

SELECT * FROM quartos;

	id_quarto integer	num_quarto integer	tipo character varying (10)	valor_diaria real	hotel integer
1	1	1	solteiro	100	2
2	2	2	familia	200	2
3	3	1001	familia	250	2
4	4	1002	casal	200	2
5	5	2001	solteiro	50	2
6	6	2002	casal	120	2
7	8	1	solteiro	20	3
8	9	2	solteiro	20	3
9	10	3	solteiro	20	3
10	11	4	solteiro	20	3
11	12	5	casal	50	3
12	13	6	casal	50	3
13	14	7	casal	50	3
14	15	8	casal	50	3
15	16	1001	familia	100	3
16	17	1002	familia	120	3

/* 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;
```

	count bigint	nome character varying (45)
1	6	Rio Grande
2	12	Cidade Maravilhosa
3	12	Águia dos Reis
4	12	Águas Claras

/*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;

```

	diaria_media double precision	nome character varying (45)
1	638	Águia dos Reis
2	58	Cidade Maravilhosa
3	153	Rio Grande
4	3500	Águas Claras

```

/*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;

```

	tipo character varying (10)	avg double precision
1	casal	610
2	familia	850
3	solteiro	400

```

/*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;

```

	nome character varying (45)	cidade character varying (20)	estado character varying (2)	bairro character varying (50)	rua character varying (50)	numero integer
1	Rio Grande	Petrolina	PE	Areia Branca	Rua Arco Verde	902
2	Cidade Maravilhosa	São Paulo	SP	Bairro X	Rua Felipe 2	202
3	Águia dos Reis	Petrolina	PE	Orla	Rua Coelho	86
4	Águas Claras	Recife	PE	Madalena	Rua José Sarney	100

/*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 hotéis h  
ON e.id_endereco = h.endereco;
```

	nome character varying (45)	tipo character varying (12)	endereço text
1	Rio Grande	Comum	Rua Arco Verde 902, Areia Branca, Petrolina (PE)
2	Cidade Maravilhosa	Pousada	Rua Felipe 2 202, Bairro X, São Paulo (SP)
3	Águia dos Reis	Luxo	Rua Coelho 86, Orla, Petrolina (PE)
4	Águas Claras	5 estrelas	Rua José Sarney 100, Madalena, Recife (PE)

/*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 hotéis tb2
ON tb1.hotel = tb2.id_hotel
GROUP BY tb2.nome, tb2.lot_max;

	nome character varying (45)	quartos_vagos bigint
1	Águas Claras	488
2	Rio Grande	294
3	Cidade Maravilhosa	188
4	Águia dos Reis	438

/*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 hotéis
DROP CONSTRAINT hotéis_lot_max_check;
ALTER TABLE hotéis
ADD CONSTRAINT hotéis_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;

```

	nome character varying (45)	tipo character varying (12)	total_de_quartos integer	quartos_vagos bigint	quartos_ocupados bigint
1	Águas Claras	5 estrelas	60	48	12
2	Águia dos Reis	Luxo	50	38	12
3	Cidade Maravilhosa	Pousada	20	8	12
4	Rio Grande	Comum	30	24	6

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

```

SELECT *
FROM hoteis
WHERE lot_max BETWEEN 20 and 50;





```

	id_hotel [PK] integer	nome character varying (45)	tipo character varying (12)	lot_max integer	endereco integer
1	2	Rio Grande	Comum	30	1
2	3	Cidade Maravilhosa	Pousada	20	2
3	4	Águia dos Reis	Luxo	50	3

```

/*Mostrar uma lista com todos os produtos da “Velma Chips” */
SELECT *
FROM produtos
WHERE UPPER(marca) = 'VELMA CHIPS';

```

	 id_produto [PK] integer	 nome character varying (50)	 marca character varying (50)	 preco real
1	5	Salgadinho Boritos Grande	Velma Chips	8
2	6	Salgadinho Boritos Pequeno	Velma Chips	4

```





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

```

```

SELECT *
FROM produtos
WHERE UPPER(nome) NOT LIKE '%LATINHA%';

```

	 id_produto [PK] integer	 nome character varying (50)	 marca character varying (50)	 preco real
1	1	Barra de Chocolar Herkeys	Herkeys	6
2	5	Salgadinho Boritos Grande	Velma Chips	8
3	6	Salgadinho Boritos Pequeno	Velma Chips	4
4	7	Loção Especial Naturo	Naturo	12
5	8	Pacote de TV a cabo Padrão	TVS	22
6	9	Pacote de TV a cabo Completo	TVS	56

```

/*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;

```

	produto text	preço text
1	Barra de Chocolar Herkeys, Herkeys	R\$ 6
2	Latinha Cosa Loca, Cosa Loca	R\$ 5
3	Latinha Sprito, Sprito	R\$ 5
4	Latinha Fantos, Fantos	R\$ 5
5	Salgadinho Boritos Grande, Velma Chips	R\$ 8
6	Salgadinho Boritos Pequeno, Velma Chips	R\$ 4
7	Loção Especial Naturo, Naturo	R\$ 12
8	Pacote de TV a cabo Padrão, TVS	R\$ 22

/*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;
```

	num_quarto text	hotel character varying (45)	produto character varying (50)	qtd_produto integer
1	1001	Rio Grande	Salgadinho Boritos Grande	3
2	1001	Rio Grande	Salgadinho Boritos Pequeno	3
3	1002	Rio Grande	Loção Especial Naturo	2
4	1002	Rio Grande	Latinha Cosa Loca	12
5	0001	Rio Grande	Latinha Cosa Loca	10
6	0001	Rio Grande	Latinha Sprito	12
7	0001	Rio Grande	Latinha Fantos	5
8	0001	Rio Grande	Salgadinho Boritos Grande	3

(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 ;

```

	count bigint
1	38

```

/*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;

```

	preço_minimo_e_maximo real
1	4
2	56

```

/*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;

```

	nome character varying (45)	quartos_mais_baratos_que_500 bigint
1	Rio Grande	6
2	Águia dos Reis	4
3	Cidade Maravilhosa	12

```

/*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;

```

	despesas_funcionarios
	real
1	78800

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

```

SELECT *
FROM cargos
ORDER BY salario DESC;

```

	id_cargo	nome	salario	area
	[PK] integer	character varying (45)	real	character varying (45)
1	10	Superintendente Regional	12000	Administração
2	6	Mestre de Cozinha	7800	Alimentação
3	8	Gerente	6200	Admnistração
4	13	Engenheiro de Computação	5700	Manutenção
5	2	Vigia Noturno	4500	Segurança
6	11	Técnico Elétrica	4200	Manutenção
7	5	Cozinheiro	3500	Alimentação
8	7	Supervisor	3100	Admnistração

/*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;

```

	area character varying (45)	despesas_funcionarios. real
1	Segurança	2100
2	Manutenção	5700
3	Serviço	20400
4	Administração	24800
5	Alimentação	11300
6	Serviços	2500
7	Administração	12000

/*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';

	area character varying (45)
1	Segurança
2	Manutenção
3	Serviço
4	Administração
5	Alimentação
6	Administração

/*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;

	nome character varying (45)	
1	Pedro Cavalcanti Prado	
2	Luiza Takahashi	
3	Breno Gabriel de Souza	
4	Julia Maria dos Santos	
5	Pedro Araújo Oliveira	
6	Luiza Alves de Carvalho	
7	João de Orvalho Peres	
8	Gildete Estrela Silveira	

/*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;
```

	nome character varying (45)	cpf character varying (11)
1	Breno Gabriel de Souza	22233344455
2	Maria Clara Mendes	12345678911
3	Jose da Silva Sampaio Junior	99988877766
4	João Coelho Silva Neto	11122233344
5	Luiza Maria de Souza	77766655544

/*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;
```

	area character varying (45)	salario_medio double precision
1	Segurança	2100
2	Manutenção	3567
3	Serviço	1700
4	Administração	4262
5	Alimentação	6367
6	Serviços	2500
7	Administração	12000

/*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;
```

	nome character varying (45)	nome character varying (45)	salario real
1	Gildete Estrela Silveira	Rio Grande	3100
2	Enzo Moraes Silveira	Rio Grande	5700
3	Carol Oliveira dos Santos	Cidade Maravilhosa	12000
4	Coelho Salles Rodrigues	Cidade Maravilhosa	3500
5	Rafael Cordeiro de Souza	Rio Grande	6200
6	Breno Gabriel Silveira Salles	Águia dos Reis	3100
7	Bruno Campos dos Santos Lima	Rio Grande	3100

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

```
SELECT TRUNC(AVG(LENGTH(f.nome)))
FROM funcionarios f;
```

	trunc numeric
1	23

/*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 hotéis 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';
```

	funcionario character varying (45)	cpf character varying (11)	hotel character varying (45)	cargo character varying (45)
1	Maria Suzana Nascimento	88811245366	Rio Grande	Faxineiro
2	Valéria Regina da Silva	22211245366	Rio Grande	Faxineiro
3	Jorge Prado do Vale	33311245366	Rio Grande	Faxineiro
4	Filomena do Nascimento Jesus	44411245366	Rio Grande	Faxineiro
5	Julia Maria dos Santos	8881555366	Cidade Maravilhosa	Faxineiro
6	Pedro Cavalcanti Prado	22216665366	Cidade Maravilhosa	Faxineiro
7	João de Orvalho Peres	33311232366	Cidade Maravilhosa	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;
```

	cliente character varying (45)	produto character varying (50)	quantidade integer
1	João Coelho Silva Neto	Barra de Chocolar Herkeys	2
2	João Coelho Silva Neto	Latinha Cosa Loca	1
3	João Coelho Silva Neto	Latinha Fantos	2
4	Rodrigo Rodrigues Pinto	Loção Especial Naturo	1
5	Jose da Silva Sampaio Junior	Pacote de TV a cabo Padrão	1
6	Jose da Silva Sampaio Junior	Salgadinho Boritos Pequeno	2

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

```
SELECT nome
FROM clientes
UNION ALL
SELECT nome
FROM produtos;
```

	nome character varying
1	João Coelho Silva Neto
2	Jose da Silva Sampaio Junior
3	Luiza Maria de Souza
4	Breno Gabriel de Souza
5	Maria Clara Mendes
6	Rodrigo Rodrigues Pinto

/*Exemplo geral de INTERCEPT

/*Quais clientes não comeram nada?*/

```
SELECT nome FROM clientes
EXCEPT
SELECT cliente FROM consumo_cliente
```

	nome character varying (45)
1	Junior Rios de Pires
2	Juan Henryco de Carmo
3	Felipe da Silva Junior
4	Fernando Bruno Nascimento Fil...
5	Maria Clara Mendes

/*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;
```

	nome character varying (45)
1	Junior Rios de Pires
2	Juan Henryco de Carmo
3	Felipe da Silva Junior
4	Fernando Bruno Nascimento Fil...
5	Maria Clara Mendes
6	Luiza Maria de Souza
7	Jose da Silva Sampaio Junior
8	Breno Gabriel de Souza
9	Rodrigo Rodrigues Pinto
10	João Coelho Silva Neto

/*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;
```


	nome character varying (45)	quarto text
1	João Coelho Silva Neto	0001
2	Breno Gabriel de Souza	0002
3	Jose da Silva Sampaio Junior	0002
4	Luiza Maria de Souza	1001
5	Maria Clara Mendes	2002

/*Apresentando todos os quartos não repetidos*/
SELECT DISTINCT LPAD(num_quarto::text, 4, '0') as quarto
FROM quartos;

	quarto text
1	3002
2	3001
3	0005
4	1003
5	0004
6	2002

/*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;
$$;

```

```

NOTICE: Quantidade de Cargos : 13
NOTICE: O cargo Faxineiro recebe o menor salario, de 1700
NOTICE: O cargo Superintendente Regional recebe o maior salario, de 12000

```

/*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;
$$;

```

```

NOTICE: 0 : (1,"Barra de Chocolar Herkeys",Herkeys,6)
NOTICE: 1 : (2,"Latinha Cosa Loca","Cosa Loca",5)
NOTICE: 2 : (3,"Latinha Sprito",Sprito,5)
NOTICE: 3 : (4,"Latinha Fantos",Fantos,5)
NOTICE: 4 : (5,"Salgadinho Boritos Grande","Velma Chips",8)
NOTICE: 5 : (6,"Salgadinho Boritos Pequeno","Velma Chips",4)
DO

```

/*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;

```

```
        END IF;
    END;
$$;

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

```
CALL buscar('BReNo');

NOTICE:  3 : Breno Gabriel de Souza
```

```
CALL buscar('Souza');

NOTICE:  2 : Luiza Maria de Souza
NOTICE:  3 : Breno Gabriel de Souza
CALL
```

```
CALL buscar('J');

NOTICE:  0 : João Coelho Silva Neto
NOTICE:  1 : Jose da Silva Sampaio Junior
NOTICE:  6 : Junior Rios de Pires
NOTICE:  7 : Juan Henryco de Carmo
NOTICE:  8 : Felipe da Silva Junior
CALL
```

```
/*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;

```

```
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER atualiza_valor  
BEFORE INSERT ON alugueis  
FOR EACH ROW  
EXECUTE PROCEDURE determinar_aluguel();
```

