DCC060 – Trabalho Prático

Alunos: Gabriel Gomes Bahia, Gabriel Frasson Costa, João Pedro Ferreira Pedreira

1. INTRODUÇÃO

O projeto desenvolvido é um sistema de delivery de comida com uma plataforma online que permite aos usuários realizarem cadastro no sistema e fazer pedidos de comida de restaurantes parceiros e receberem entrega em suas residências. O sistema oferecerá uma ampla variedade de opções de restaurantes, incluindo fast food, comida italiana, comida chinesa e muito mais.

Para fazer um pedido, os usuários podem selecionar um restaurante e adicionar itens ao carrinho de compras, com possibilidade de edição do carrinho, inclusão e remoção de itens. Eles podem escolher entre diferentes formas de pagamento, como cartão de crédito ou débito, e rastrear o status de entrega em tempo real através do site do sistema. Os usuários também poderão incluir notas adicionais ou restrições no pedido e trocar mensagens com o restaurante. Existirá opção de cupom de desconto onde os usuários podem ganhar descontos em seus pedidos. Além disso, após finalizar o pedido, poderá dar a avaliação por meio de comentários e upvotes para ajudar a melhorar o serviço e a qualidade da comida.

O sistema contará com a parte administrativa que será acessada somente para os restaurantes parceiros, que poderão editar suas informações, realizar o gerenciamento de produtos, pedidos, frete, motoboys, entregas e fazer a comunicação com os clientes. Também existirá a área de compras para todos os clientes, onde todos os restaurantes e produtos serão exibidos. E por último, haverá uma outra área administrativa para o próprio sistema, onde será feito o gerenciamento de usuários e restaurantes.

O sistema também oferece integração com plataformas de pagamento online e aplicativos de localização para garantir a facilidade de uso e a eficiência do processo de entrega. Também existirá um suporte ao cliente que deve ajudar os usuários a resolver qualquer problema ou dúvida que possam ter. Isso pode incluir chat online, telefone ou e-mail.

2. REQUISITOS DO PROJETO

Requisitos do sistema
1. Gerenciamento de usuários: o sistema irá realizar o registro, edição e remoção de usuários. 2. Gerenciamento de restaurantes: o sistema irá realizar o registro, edição e remoção de restaurantes. 3. Gerenciamento de produtos: o sistema irá realizar o registro, edição e remoção de produtos. 4. Gerenciamento de pedidos: o sistema irá realizar o registro, edição e remoção de pedidos. 5. Pesquisa de restaurantes: os usuários devem ser capazes de pesquisar restaurantes por localização, tipo de comida ou outros critérios. 6. Catálogo de produtos: os usuários devem ser capazes de visualizar os produtos oferecidos pelos restaurantes, incluindo fotos, descrições e preços. 7. Carrinho de compras: os usuários devem ser capazes de adicionar itens ao carrinho de compras e ajustar a quantidade de cada item. 8. Escolha de forma de pagamento: os usuários devem ser capazes de escolher entre diferentes formas de pagamento, como cartão de crédito ou débito, e verificar o valor total do pedido. 9. Rastreamento de entrega: os usuários devem ser capazes de acompanhar o status

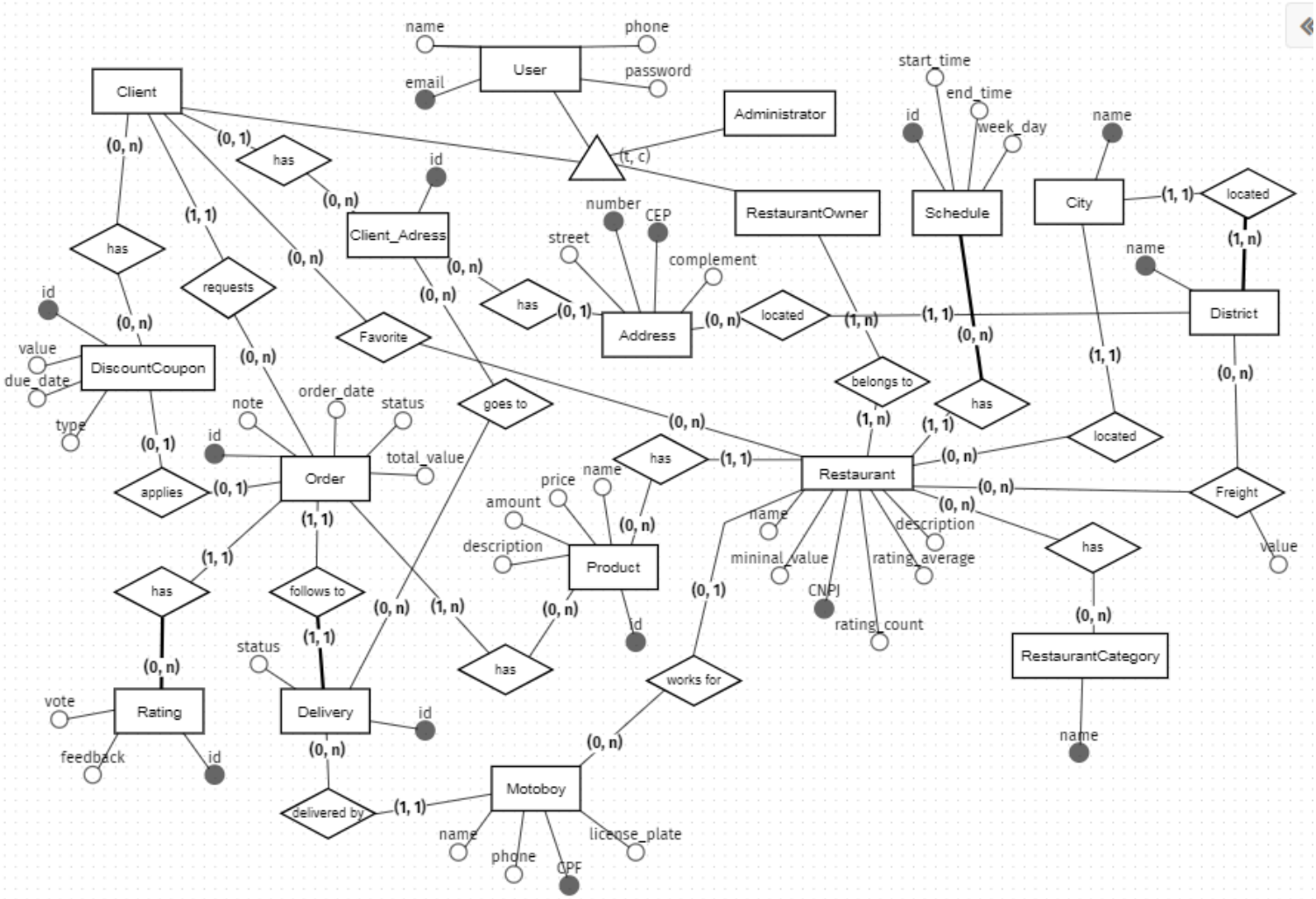
de entrega em tempo real através do aplicativo ou site do sistema.
10. Feedback e avaliação: os usuários devem ser capazes de deixar comentários e avaliar os restaurantes e os entregadores para ajudar a melhorar o serviço e a qualidade da comida.
11. Suporte ao cliente: o sistema deve oferecer suporte ao cliente para ajudar os usuários a resolver qualquer problema ou dúvida que possam ter. Isso pode incluir chat online, telefone ou e-mail.
12. Personalização de pedidos: os usuários devem ser capazes de adicionar notas especiais aos seus pedidos, como por exemplo "sem cebola" ou "extra quente", para garantir que os seus pedidos sejam preparados exatamente como eles desejam.
13. Descontos e promoções: o sistema deve oferecer descontos e promoções especiais para os usuários, por meio de cupons que são sorteados ou comprados.

Regras de Negócio

1. Disponibilidade de restaurantes: o sistema estará disponível apenas em área com restaurantes parceiros.
2. Restrições de horário: alguns restaurantes parceiros podem ter horários de funcionamento limitados, o que pode afetar a disponibilidade de pedidos.
3. Restrições de entrega: o sistema terá restrições em relação às áreas de entrega ou aos horários de entrega disponíveis, dependendo da localização e horário de funcionamento do restaurante.
4. Restrições de pagamento: o sistema terá restrições em relação às formas de pagamento aceitas, só serão aceitas as formas de pagamento: dinheiro, PIX, cartão de crédito e débito.
5. Restrições de personalização: alguns restaurantes parceiros podem ter restrições em relação às personalizações de pedidos, como por exemplo não permitir que os usuários adicionem ou retirem ingredientes de determinados pratos.
6. Restrições de usuários: no sistema existirão três tipos de usuários (Administrador, Dono do restaurante e cliente) e cada um terá suas permissões e restrições dentro do sistema. Como exemplo, o Administrador terá acesso a todas as permissões do sistema, o dono do restaurante poderá apenas realizar a gestão de seus produtos, pedidos e entregas, e o cliente poderá apenas realizar pedidos.
7. Restrições de pedidos mínimos: o sistema pode ter um valor mínimo para os pedidos, o que pode impedir que os usuários façam pedidos de valor muito baixo.
8. Restrições de cancelamento de pedidos: o sistema terá restrições em relação ao cancelamento de pedidos, não permitindo o cancelamento após o pedido ter sido confirmado pelo restaurante e ter saído para a entrega.
9. Restrições de alteração de pedidos: o sistema terá restrições em relação à alteração de pedidos já confirmados, não permitindo a adição ou remoção de itens.
10. Restrições de uso de descontos ou promoções: o sistema terá restrições em relação ao uso de descontos ou promoções, não permitindo o uso de mais de um desconto por pedido ou restringindo o uso de promoções em determinados dias ou horários.

3. MODELAGEM RELACIONAL

3.1 Diagrama Entidade Relacionamento



3.2 Modelagem lógica

Cities(id, name)

Districts(id, name, id_city)

id_city referencia Cities

Addresses(id, street, complement, number, CEP, id_district)

id_district referencia Districts

Restaurants(id, name, minimal_value, description, rating_average, rating_count, cnpj, id_address)

id_address referencia Addresses

Phones(id, country, ddd, phone_number)

Users(id, password, name, email, id_phone)

id_phone referencia Phones

Products(id, name, price, description, id_restaurant)

id_restaurant referencia Restaurants

Restaurant_owners(id_user)

id_user referencia Users

Administrators(id_user)

id_user referencia Users

Clients(id_user)

id_user referencia Users

Motoboys(id, name, license_plate, cpf, id_restaurant , id_phone)

id_restaurant referencia Restaurants

id_phone referencia Phones

Deliveries(id, status, id_motoboy)

id_motoboy referencia Motoboys

Orders(id, order_date, status, total_value, note, id_client, id_delivery)

id_client referencia Clients

id_delivery referencia Deliveries

Discount_coupons(id, value, due_date, type)

Restaurant_categories(id, name)

Ratings(id, vote, feedback, id_order)

id_order referencia Orders

Clients_addresses(id, id_client, id_address)

id_client referencia Clients

id_address referencia Addresses

Schedules(id, start_time, end_time, week_day, id_restaurant)

id_restaurant referencia Restaurants

Favorites(id, id_client, id_restaurant)

id_client referencia Clients

id_restaurant referencia Restaurants

Freights(id, value, id_restaurant, id_district)

id_restaurant referencia Restaurants

id_district referencia Districts

Clients_discount_coupons(id, id_client, id_discount_coupon, id_order)

id_client referencia Clients

id_discount_coupon referencia Discount_coupons
id_order referencia Orders

Clients_addresses_deliveries(id, id_clients_address, id_delivery)
id_clients_address referencia Clients_addresses
id_delivery referencia Deliveries

Orders_products(id, id_order, id_product, amount)
id_order referencia Orders
id_product referencia Products

Restaurant_owners_restaurants(id, id_restaurant_owner, id_restaurant)
id_restaurant_owner referencia Restaurant_owners
id_restaurant referencia Restaurants

Restaurants_restaurant_categories(id, id_restaurant, id_restaurant_category)
id_restaurant referencia Restaurants
id_restaurant_category referencia Restaurant_categories

4 CONSULTAS EM ÁLGEBRA RELACIONAL

Consulta 1: “Qual é o nome de todos os restaurantes com mais de 100 votos de avaliação na cidade de juiz de fora”
π{name} (σ vote > 100 E city = "Juiz de Fora" (((Restaurant ⋈ Order) ⋈ Rating) ⋈ Address))

Consulta 2: “Quais são os pedidos feitos por um determinado cliente:”

π{number, order_date, total_value} (σ name= "Fulano da silva" (Order ⋈ Client))

Consulta 3: “Encontre os clientes que fizeram pedidos de produtos com o nome hamburguer, mas que nunca fizeram pedidos de produtos com o nome pizza.:”

π(name_client)((Order ⋈ (σ(name= 'hamburguer')(Product))) ⋈ Client) -
π(name_client)((Order ⋈ (σ(name = 'pizza')(Product))) ⋈ Client)

Consulta 4: “Quais restaurantes possuem orders com rating menor ou igual X:”

id_delivery = id_delivery
A1 <= π{orders#id} (σ vote <= X (orders ⋈ ratings))

id = id_orders
A2 <= π{id_products} (A1 ⋈ orders_products)

id_products = id
A3 <= π{id_restaurants} (A2 ⋈ products)

id = CNPJ
A4 <= π{name} (A3 ⋈ restaurants)

Consulta 5: “Quais clientes possuem algum cupom de desconto:”

id_client = id_user
A1 <= π{id_user} (clients_discount_coupons ⋈ clients)

email = id_user
A2 <= π{name, email} (users ⋈ A1)

Consulta 6: “Quais clientes já utilizaram algum cupom de desconto:”

id_discount_coupons = id
A1 <= π{id_client} (σ id_order <> ‘ ’ (clients_discount_coupons ⋈ discount_coupons))

id_client = id_user
A2 <= π{id_user, id_discount_coupons} (A1 ⋈ clients)

email = id_user
A3 <= π{name, email} (users ⋈ A2)

Consulta 7: “Quais são os produtos que todos os clientes pediram:”

π{name}(Product) ÷ π{id_product}((Order ⋈ Product)⋈Client)

Consulta 8: “Quais são os pedidos que no dia de hoje que ainda não sairam para entrega:”

id_delivery = id
A1 <= π{order_id}(Order ⋈ σ(status != 'saiu para entrega')(Delivery))
A2 <= π{Order}((σ(order_date = '2022-03-01')(Order))) - A1

Consulta 9: “Quais as categorias de um restaurante X:”

A1 <= π{id_restaurant_category} (σ id_restaurant = X
(restaurants_restaurant_categories))

name = id_restaurant_category
A2 <= π{name} (restaurant_categories ⋈ A1)

Consulta 10: “Encontre os clientes que nunca fizeram pedidos:”

id_client = id
A1<= π{id_client}(Order ⋈Client)
A2<= π{Client}(Client) - A1

Consulta 11: “Obter a lista de todos os pedidos que foram entregues por um determinado entregador:”

A1<= σ(name = 'fulano')(Motoboy)
id_motoboy = id
A2<= π{id_delivery}(Delivery ⋈ A1)
id_delivery = id_delivery
A3<=π{Order}(Order ⋈ A2)

Consulta 12: “Obter a lista de todos os bairros que um restaurante atende:”

$$A1 \leftarrow (Restaurant \underset{id\_address = (cep,number)}{\bowtie} Address)$$
$$A2 \leftarrow \pi\{name\}(District \underset{id\_city = (cep,number)}{\bowtie} A1)$$

Consulta 13: “Obter a lista de todos os pedidos que foram feitos por clientes que moram no mesmo endereço que o restaurante que preparou o pedido:”

$$A1 \leftarrow (Clients\_adresses \underset{id = id\_clients\_addresses}{\bowtie} Clients\_addresses\_deliveries)$$
$$A2 \leftarrow (Restaurant \underset{id\_address = (number,cep)}{\bowtie} Address)$$
$$A3 \leftarrow (A1 \underset{id\_clients\_addresses = id\_address}{\bowtie} A2)$$
$$A4 \leftarrow (A3 \underset{id\_deliveries = id)}{\bowtie} Deliveries)$$
$$A5 \leftarrow \pi\{id\_order\} (Orders \underset{id\_delivery = id\_delivery}{\bowtie} A4)$$

Consulta 14: “Quais restaurantes pertencem a categoria X:”

$$A1 \leftarrow \pi\{id\_restaurant\} (\sigma\ id\_restaurant\_category = X\ (restaurants\_restaurant\_categories))$$
$$A2 \leftarrow \pi\{CNPJ\} (restaurants \underset{CNPJ = id\_restaurant}{\bowtie} A1)$$

Consulta 15: “Quais endereços nunca pediram em X restaurante:”

$$A1 \leftarrow \pi\{id\_client, id\_delivery, id\} (clients \underset{id\_user = id\_client}{\bowtie} orders)$$
$$A2 \leftarrow \pi\{id\_client, id\_products\} (A1 \underset{id = id\_orders}{\bowtie} orders\_products)$$
$$A3 \leftarrow \pi\{id\_client, id\_restaurants\} (products \underset{id\_products = id\_user}{\bowtie} A2)$$

CNPJ = id_restaurants

A4 <= π{id_client, id_restaurants} (σ name = X (restaurants ⋈ A2))

A5 <= π{id_client} (clients - A4)

A6 <= π{id_address} (clients_addresses ⋈ A5)

A7 <= π{*} (addresses ⋈ A6)

5 MODELO FÍSICO (Parte 1) E CARGA DE DADOS

Todos os itens abaixo devem ser descritos no relatório.

5.1 Tabelas

(DDL de criação de tabelas para o esquema relacional, consistente com a modelagem lógica com todas as restrições de integridade presentes)

Criação de tabelas

```
CREATE TABLE cities
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL
);

CREATE TABLE districts
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL,
	id_city INT NOT NULL,

	CONSTRAINT fk_districts_cities FOREIGN KEY (id_city) REFERENCES
cities(id)
);

CREATE TABLE addresses
(
	id SERIAL PRIMARY KEY,
	street VARCHAR(255) NOT NULL,
	complement VARCHAR(255),
	number INT NOT NULL,
	CEP INT NOT NULL,
	id_district INT NOT NULL,

	CONSTRAINT fk_addresses_districts FOREIGN KEY (id_district)
REFERENCES districts(id)
```

```
);

CREATE TABLE restaurants
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL,
	minimal_value NUMERIC,
	description VARCHAR(255),
	rating_average NUMERIC,
	rating_count INT DEFAULT 0 NOT NULL,
	cnpj VARCHAR(14) NOT NULL,
	id_address INT NOT NULL,

	CONSTRAINT fk_restaurants_addresses FOREIGN KEY (id_address)
REFERENCES addresses(id)
);

CREATE TABLE phones
(
	id SERIAL PRIMARY KEY,
	country VARCHAR(10) NOT NULL,
	ddd VARCHAR(10) NOT NULL,
	phone_number VARCHAR(255) NOT NULL
);

CREATE TABLE users
(
	id SERIAL PRIMARY KEY,
	password TEXT NOT NULL,
	name VARCHAR(255) NOT NULL,
	email VARCHAR(255) NOT NULL,
	id_phone INT NOT NULL,

	CONSTRAINT fk_users_phones FOREIGN KEY (id_phone) REFERENCES
phones(id)
);

CREATE TABLE products
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL,
	price NUMERIC NOT NULL,
	description VARCHAR(255),
	id_restaurant INT NOT NULL,

	CONSTRAINT fk_products_restaurants FOREIGN KEY (id_restaurant)
REFERENCES restaurants(id)
);


CREATE TABLE restaurant_owners
(
	id_user INT PRIMARY KEY,
```

```
	CONSTRAINT fk_restaurant_owners_users FOREIGN KEY (id_user)
REFERENCES users(id)
);

CREATE TABLE administrators
(
	id_user INT PRIMARY KEY,

	CONSTRAINT fk_administrators_users FOREIGN KEY (id_user) REFERENCES
users(id)
);

CREATE TABLE clients
(
	id_user INT PRIMARY KEY,

	CONSTRAINT fk_clients_users FOREIGN KEY (id_user) REFERENCES
users(id)
);

CREATE TABLE motoboys
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL,
	license_plate VARCHAR(255) NOT NULL,
	CPF VARCHAR(255) NOT NULL,
	id_restaurant INT NOT NULL,
	id_phone INT NOT NULL,

	CONSTRAINT fk_motoboys_restaurants FOREIGN KEY (id_restaurant)
REFERENCES restaurants(id),
	CONSTRAINT fk_motoboys_phones FOREIGN KEY (id_phone) REFERENCES
phones(id)
);

CREATE TYPE DeliveryStatus AS ENUM (
	'WAITING',
	'ON THE WAY',
	'DELIVERED'
);

CREATE TABLE deliveries
(
	id SERIAL PRIMARY KEY,
	status DeliveryStatus NOT NULL,
	id_motoboy INT NOT NULL,

	CONSTRAINT fk_deliveries_motoboys FOREIGN KEY (id_motoboy)
REFERENCES motoboys(id)
);

CREATE TYPE OrderStatus AS ENUM (
	'PENDANT',
	'APPROVED',
```

```
	'PREPARING',
	'READY',
	'FINISHED'
);

CREATE TABLE orders
(
	id SERIAL PRIMARY KEY,
	order_date TIMESTAMPTZ NOT NULL DEFAULT now(),
	status OrderStatus NOT NULL,
	total_value NUMERIC NOT NULL,
	note VARCHAR(255),
	id_client INT NOT NULL,
	id_delivery INT,

	CONSTRAINT fk_orders_clients FOREIGN KEY (id_client) REFERENCES
clients(id_user),
	CONSTRAINT fk_orders_deliveries FOREIGN KEY (id_delivery) REFERENCES
deliveries(id)
);

CREATE TYPE DiscountType AS ENUM (
	'PERCENT',
	'REAL'
);

CREATE TABLE discount_coupons
(
	id SERIAL PRIMARY KEY,
	value NUMERIC NOT NULL,
	due_date TIMESTAMPTZ,
	type DiscountType
);

CREATE TABLE restaurant_categories
(
	id SERIAL PRIMARY KEY,
	name VARCHAR(255) NOT NULL UNIQUE
);

CREATE TABLE ratings
(
	id SERIAL PRIMARY KEY,
	vote INT CHECK (vote BETWEEN 1 AND 5),
	feedback VARCHAR(255),
	id_order INT NOT NULL,

	CONSTRAINT fk_ratings_orders FOREIGN KEY (id_order) REFERENCES
orders(id)
);

CREATE TABLE clients_addresses
(
	id SERIAL PRIMARY KEY,
```

```
	id_client INT NOT NULL,
	id_address INT NOT NULL,

	CONSTRAINT fk_clients_addresses_clients FOREIGN KEY (id_client)
REFERENCES clients(id_user),
	CONSTRAINT fk_clients_addresses_addresses FOREIGN KEY (id_address)
REFERENCES addresses(id)
);

CREATE TYPE Weekday AS ENUM (
	'SUNDAY',
	'MONDAY',
	'TUESDAY',
	'WEDNESDAY',
	'THURSDAY',
	'FRIDAY',
	'SATURDAY'
);

CREATE TABLE schedules
(
	id SERIAL PRIMARY KEY,
	start_time TIME(0) NOT NULL,
	end_time TIME(0) NOT NULL,
	week_day Weekday NOT NULL,
	id_restaurant INT NOT NULL,

	CONSTRAINT fk_schedules_restaurants FOREIGN KEY (id_restaurant)
REFERENCES restaurants(id)
);

CREATE TABLE favorites
(
	id SERIAL PRIMARY KEY,
	id_client INT NOT NULL,
	id_restaurant INT NOT NULL,

	CONSTRAINT fk_favorites_clients FOREIGN KEY (id_client) REFERENCES
clients(id_user),
	CONSTRAINT fk_favorites_restaurants FOREIGN KEY (id_restaurant)
REFERENCES restaurants(id)
);

CREATE TABLE freights
(
	id SERIAL PRIMARY KEY,
	value NUMERIC NOT NULL,
	id_restaurant INT NOT NULL,
	id_district INT NOT NULL,

	CONSTRAINT fk_freights_restaurants FOREIGN KEY (id_restaurant)
REFERENCES restaurants(id),
	CONSTRAINT fk_freights_districts FOREIGN KEY (id_district) REFERENCES
districts(id)
```

```
);

CREATE TABLE clients_discount_coupons
(
	id SERIAL PRIMARY KEY,
	id_client INT NOT NULL,
	id_discount_coupon INT NOT NULL,
	id_order INT NOT NULL,

	CONSTRAINT fk_clients_discount_coupons_clients FOREIGN KEY (id_client)
REFERENCES clients(id_user),
	CONSTRAINT fk_clients_discount_coupons_discount_coupons FOREIGN KEY
(id_discount_coupon) REFERENCES discount_coupons(id),
	CONSTRAINT fk_clients_discount_coupons_orders FOREIGN KEY (id_order)
REFERENCES orders(id),

	UNIQUE (id_client, id_discount_coupon),
	UNIQUE (id_order)
);

CREATE TABLE clients_addresses_deliveries
(
	id SERIAL PRIMARY KEY,
	id_clients_address INT NOT NULL,
	id_delivery INT NOT NULL,

	CONSTRAINT fk_clients_addresses_deliveries_clients_addresses FOREIGN
KEY (id_clients_address) REFERENCES clients_addresses (id),
	CONSTRAINT fk_clients_addresses_deliveries_deliveries FOREIGN KEY
(id_delivery) REFERENCES deliveries (id)
);

CREATE TABLE orders_products
(
	id SERIAL PRIMARY KEY,
	id_order INT NOT NULL,
	id_product INT NOT NULL,
	amount INT NOT NULL,

	CONSTRAINT fk_orders_products_orders FOREIGN KEY (id_order)
REFERENCES orders(id),
	CONSTRAINT fk_orders_products_products FOREIGN KEY (id_product)
REFERENCES products(id)
);

CREATE TABLE restaurant_owners_restaurants
(
	id SERIAL PRIMARY KEY,
	id_restaurant_owner INT NOT NULL,
	id_restaurant INT NOT NULL,

	CONSTRAINT fk_restaurant_owners_restaurants_restaurant_owners FOREIGN
KEY (id_restaurant_owner) REFERENCES restaurant_owners(id_user),
	CONSTRAINT fk_restaurant_owners_restaurants_restaurants FOREIGN KEY
```

```
(id_restaurant) REFERENCES restaurants(id)
);

CREATE TABLE restaurants_restaurant_categories
(
	id SERIAL PRIMARY KEY,
	id_restaurant INT NOT NULL,
	id_restaurant_category INT NOT NULL,

	CONSTRAINT fk_restaurants_restaurant_categories_restaurant FOREIGN KEY
(id_restaurant) REFERENCES restaurants(id),
	CONSTRAINT fk_restaurants_restaurant_categories_restaurant_categories
FOREIGN KEY (id_restaurant_category) REFERENCES restaurant_categories(id)
);
```

5.2 Verificação

(INSERTs, UPDATEs e DELETEs que demonstram o correto comportamento dos controles de integridade criados no DDL)

```
UPDATE cascata.restaurants SET description='Somos um restaurant de comida japonesa' WHERE id=1
UPDATE cascata.orders SET status='READY' WHERE id=1
UPDATE cascata.products SET price='22.50' WHERE id=1
UPDATE cascata.freights SET value='4.00' WHERE id=1
UPDATE cascata.restaurant_categories SET name='Italiana' WHERE id=1
UPDATE cascata.motoboys SET license_plate='ABC1D23' WHERE id=1
UPDATE cascata.deliveries SET status='ON THE WAY' WHERE id=1
UPDATE cascata.districts SET name='Centro' WHERE id=1
UPDATE cascata.cities SET name='Guarani' WHERE id=1
UPDATE cascata.schedules SET start_time='18:00' WHERE id=1
UPDATE cascata.ratings SET vote=4 WHERE id=1

DELETE FROM cascata.ratings WHERE id=1
DELETE FROM  cascata.schedules WHERE id=1
DELETE FROM cascata.freights WHERE id=1
DELETE FROM cascata.favorites WHERE id=1
(Não pode apagar pois viola restrição de chave estrangeira)
DELETE FROM cascata.restaurants WHERE id=1
(Não pode apagar pois viola restrição de chave estrangeira)
DELETE FROM cascata.orders WHERE id=1
(Não pode apagar pois viola restrição de chave estrangeira)
DELETE FROM cascata.products WHERE id=1
```

5.3 Carga de Dados

(INSERTs para carga dos dados. Inserir uma quantidade razoável de dados para que façam sentido as consultas que serão feitas posteriormente em SQL)

Carga de Dados

```
-- Script Carga

-- Tabela: cities / Instâncias: 5
INSERT INTO
	cities (name)
VALUES
	('Juiz de Fora'),
	('Belo Horizonte'),
	('Vitória'),
	('São Paulo'),
	('Rio de Janeiro');

-- Tabela: districts / Instâncias: 17
INSERT INTO
	districts (name, id_city)
VALUES
	('São Pedro', 1),
	('Cascatinha', 1),
	('São Mateus', 1),
	('Centro', 1),
	('São Luís', 2),
	('Liberdade', 2),
	('Ouro Preto', 2),
	('Centro', 2),
	('Jardim da Penha', 3),
	('Goiabeiras', 3),
	('Jardim Camburi', 3),
	('Ibirapuera', 4),
	('Vila Formosa', 4),
	('Campo Limpo', 4),
	('Santa Teresa', 5),
	('Ipanema', 5),
	('Copacabana', 5);

-- Tabela: addresses / Instâncias: 20
INSERT INTO
	addresses (street, complement, number, cep,
id_district)
VALUES
	('José Silva', 'Apt 204', 190, '28706903', 1),
	('João', NULL, 192, '28706913', 1),
	('Presidente', NULL, 200, '28706904', 2),
```

```
        ('João Goulart', NULL, 260, '28566904', 2),
        ('Marechal Deodoro', NULL, 201, '28706904', 3),
        ('Getúlio Vargas', NULL, 205, '28706905', 4),
        ('Itamar Franco', NULL, 207, '28716904', 5),
        ('Juscelino', NULL, 15, '28712904', 6),
        ('Floriano Peixoto', NULL, 17, '28703404', 7),
        ('Prudente de Morais', NULL, 14, '28156904', 8),
        ('Campos Sales', NULL, 359, '38706904', 9),
        ('Rodrigues Alves', NULL, 412, '45706904', 10),
        ('Afonson Pena', NULL, 10, '23506904', 11),
        ('Nilo Peçanha', NULL, 27, '28676904', 12),
        ('Hermes da Fonseca', NULL, 32, '28236904', 13),
        ('Rodrigues Alves', NULL, 43, '28734504', 14),
        ('Epitácio Pessoa', NULL, 34, '28756704', 15),
        ('Artur Bernardes', NULL, 576, '28706701', 16),
        ('Júlio Prestes', NULL, 1001, '28706653', 17),
        ('Jânio Quadros', NULL, 1010, '28725704', 17);

-- Tabela: restaurants / Instâncias: 7
INSERT INTO
        restaurants (name, minimal_value, description,
cnpj, id_address)
VALUES
        ('McDonalds', 15.00, 'Aqui você encontra o melhor
Big Mac da cidade', '12345678912345', 1),
        ('Subway', 20.00, 'Baratíssimo!!',
'12345654612345', 7),
        ('La Dolina', 20.00, 'Casa de carnes Argentina',
'26755678512345', 11),
        ('Pizzaria Stroele', 35.00, 'A melhor pizza da
cidade', '56745678912345', 14),
        ('Bobs', 10.00, 'Milk Shakes em promoção!',
'50289679567345', 20),
        ('Forno a lenha', NULL, NULL, '50285479567335',
19),
        ('Oro Restaurante', NULL, 'O melhor do Brasil!',
'20285423567335', 18);


-- Tabela: phones / Instâncias: 19
INSERT INTO
        phones (country, ddd, phone_number)
VALUES
        ('55', '27', '999998888'),
        ('55', '32', '999998887'),
        ('55', '32', '999998886'),
        ('55', '32', '999998885'),
        ('55', '31', '999998884'),
        ('55', '31', '999998883'),
        ('55', '31', '999998882'),
        ('55', '31', '999998881'),
        ('55', '31', '999998880'),
        ('55', '32', '888889999'),
        ('55', '32', '888889998'),
```

```
        ('55', '32', '888889997'),
        ('55', '32', '888889996'),
        ('55', '32', '888889995'),
        ('55', '32', '888889994'),
        ('55', '32', '888889993'),
        ('55', '32', '888889992'),
        ('55', '32', '888889991'),
        ('55', '32', '888889990');

-- Tabela: users / Instâncias: 9
INSERT INTO
        users (password, name, email, id_phone)
VALUES
        (md5('123456'), 'Gabriel Frasson',
'frasson@email.com', 1),
        (md5('123456'), 'João', 'joao@email.com', 2),
        (md5('123456'), 'Gabriel Bahia',
'bahia@email.com', 3),
        (md5('123456'), 'José', 'jose@email.com', 4),
        (md5('123456'), 'Maria', 'maria@email.com', 5),
        (md5('123456'), 'Felipe', 'felipe@email.com', 6),
        (md5('123456'), 'Victor', 'victor@email.com', 7),
        (md5('123456'), 'Bárbara', 'barbara@email.com',
8),
        (md5('123456'), 'Pedro', 'pedro@email.com', 9);

-- Tabela: products / Instâncias: 24
INSERT INTO
        products (name, price, description, id_restaurant)
VALUES
        ('Big Mac', 25.00, 'Dois hambúrgueres (100% carne
bovina), alface americana, queijo cheddar, maionese Big
Mac, cebola, picles e pão com gergelim.', 1),
        ('Cheeseburger', 14.99, 'Um hamburguer (100%
carne bovina), queijo sabor cheddar, cebola, picles,
ketchup, mostarda e pão sem gergelim.', 1),
        ('McFritas', 8.99, 'A batata frita mais famosa do
mundo. Deliciosas batatas selecionadas, fritas, crocantes
por fora, macias por dentro, douradas, irresistíveis,
saborosas, famosas, e todos os outros adjetivos positivos
que você quiser dar.', 1),
        ('Frango Ranch', 17.99, 'Frango defumado em
cubos e molho ranch', 2),
        ('Frango defumado com Cream Cheese', 18.99,
'Frango defumado em cubos e Cream Cheese', 2),
        ('Baratíssimo', 14.50, 'Frango grelhado', 2),
        ('Polígrafo de Flores', 39.00, 'Saboroso
hambúrguer de fraldinha (200g) com tomates confitados,
bacon assado, rúcula e molho de champignons, alho poró
e vinho branco em pão de ervas.', 3),
        ('Queijo a La Parrilla', 38.00, NULL, 3),
        ('Moda do Chef', 54.00, NULL, 4),
        ('Marguerita', 60.00, NULL, 4),
        ('Portuguesa', 50.00, NULL, 4),
```

```
        ('Calabresa', 45.00, NULL, 4),
        ('Frango com catupiry', 45.00, NULL, 4),
        ('Milk shake 300ml', 11.00, NULL, 5),
        ('Milk shake 500ml', 13.00, NULL, 5),
        ('Milk shake 700ml', 15.00, NULL, 5),
        ('Casquinha', 3.50, NULL, 5),
        ('Cascão', 5.50, NULL, 5),
        ('Moda do Chef', 64.00, NULL, 6),
        ('Marguerita', 70.00, NULL, 6),
        ('Portuguesa', 60.00, NULL, 6),
        ('Picanha', 100.00, NULL, 7),
        ('Risoto', 150.00, NULL, 7),
        ('Escalope', 85.00, NULL, 7);

-- Tabela: restaurant_owners / Instâncias: 6
INSERT INTO
        restaurant_owners (id_user)
VALUES
        (1),
        (2),
        (3),
        (7),
        (8),
        (9);

-- Tabela: administrators / Instâncias: 3
INSERT INTO
        administrators (id_user)
VALUES
        (1),
        (2),
        (3);

-- Tabela: clients / Instâncias: 8
INSERT INTO
        clients (id_user)
VALUES
        (1),
        (2),
        (3),
        (4),
        (5),
        (6),
        (7),
        (8);

-- Tabela: motoboys / Instâncias: 10
INSERT INTO
        motoboys (name, license_plate, CPF,
id_restaurant, id_phone)
VALUES
        ('João', 'ABC1234', '12345678901', 1, 10),
        ('Maria', 'BCD1234', '22345678901', 1, 11),
        ('José', 'GCD1234', '22655678901', 1, 12),
```

```
        ('Pedro', 'GCJ2234', '22655678901', 2, 13),
        ('Paulo', 'GCA1224', '22698678901', 2, 14),
        ('Gustavo', 'GCB9224', '22698678031', 3, 15),
        ('Gisele', 'JCB9224', '22623878031', 4, 16),
        ('Marcos', 'GCB9202', '42598678031', 5, 17),
        ('Victor', 'GCB3214', '22698678281', 6, 18),
        ('Lucas', 'JLB9224', '89398678031', 7, 19);

-- Tabela: deliveries / Instâncias: 16
INSERT INTO
        deliveries (status, id_motoboy)
VALUES
        ('WAITING', 1),
        ('ON THE WAY', 2),
        ('ON THE WAY', 3),
        ('DELIVERED', 1),
        ('DELIVERED', 2),
        ('DELIVERED', 3),
        ('DELIVERED', 4),
        ('DELIVERED', 5),
        ('DELIVERED', 6),
        ('DELIVERED', 7),
        ('DELIVERED', 8),
        ('DELIVERED', 9),
        ('DELIVERED', 10),
        ('WAITING', 10),
        ('WAITING', 10),
        ('WAITING', 10);

-- Tabela: orders / Instâncias: 17
INSERT INTO
        orders (status, total_value, note, id_client,
id_delivery)
VALUES
        ('PENDANT', 25.00, NULL, 1, 1),          -- Big Mac
        ('READY', 14.99, NULL, 1, 2),            --
Cheeseburger
        ('READY', 33.99, NULL, 2, 3),            -- Big Mac +
McFritas
        ('READY', 25.00, 'Sem alface', 3, 4),    -- Big Mac
        ('READY', 14.99, NULL, 3, 5),            --
Cheeseburger
        ('READY', 39.99, NULL, 4, 6),            -- Big Mac +
Cheeseburger
        ('READY', 17.99, NULL, 5, 7),            -- Frango
Ranch
        ('READY', 18.99, NULL, 3, 8),            -- Frango
defumado com Cream Cheese
        ('READY', 39.00, NULL, 1, 9),            -- Polígrafo
de Flores
        ('READY', 60.00, NULL, 2, 10),           --
Marguerita
        ('READY', 11.00, NULL, 3, 11),           -- Milk
shake 300ml
```

```
        ('READY', 64.00, NULL, 7, 12),          -- Moda do
Chef
        ('READY', 85.00, NULL, 8, 13),          -- Escalope
        ('APPROVED', 185.00, NULL, 1, 14),      --
Escalope + Picanha
        ('PREPARING', 100.00, NULL, 2, 15),     --
Picanha
        ('READY', 150.00, NULL, 3, 16),         -- Risoto
        ('PENDANT', 150.00, NULL, 5, NULL);     --
Risoto

-- Tabela: discount_coupons / Instâncias: 5
INSERT INTO
        discount_coupons (value, due_date, type)
VALUES
        (15, timestamptz '2023-01-20 23:59:59
America/Sao_Paulo', 'REAL'),
        (10, timestamptz '2023-01-25 23:59:59
America/Sao_Paulo', 'PERCENT'),
        (10, timestamptz '2023-01-05 23:59:59
America/Sao_Paulo', 'REAL'),
        (20, timestamptz '2023-01-04 23:59:59
America/Sao_Paulo', 'REAL'),
        (15, timestamptz '2023-01-26 23:59:59
America/Sao_Paulo', 'PERCENT');

-- Tabela: restaurant_categories / Instâncias: 7
INSERT INTO
        restaurant_categories (name)
VALUES
        ('Fast Food'),
        ('Lanchonete'),
        ('Pizzaria'),
        ('Sorveteria'),
        ('Gourmet'),
        ('Vegano'),
        ('Churrascaria');

-- Tabela: ratings / Instâncias: 10
INSERT INTO
        ratings (vote, feedback, id_order)
VALUES
        (5, 'Muito bom', 1),
        (1, 'Não gostei', 2),
        (2, 'Ok', 3),
        (3, 'Mediano', 4),
        (4, 'Bom', 5),
        (5, 'Fantástico!', 6),
        (5, 'Muito bom mesmo', 7),
        (4, 'Bom demais', 8),
        (5, 'Gostei', 9),
        (3, 'Mediano', 10);

-- Tabela: clients_addresses / Instâncias: 13
```

```
INSERT INTO
	clients_addresses (id_client, id_address)
VALUES
	(1, 2),
	(2, 2),
	(2, 3),
	(3, 4),
	(4, 5),
	(5, 6),
	(6, 8),
	(7, 9),
	(8, 10),
	(8, 12),
	(8, 13),
	(8, 15),
	(8, 16);

-- Tabela: schedules / Instâncias: 36
INSERT INTO
	schedules (start_time, end_time, week_day,
id_restaurant)
VALUES
	('08:00', '22:00', 'MONDAY', 1),
	('08:00', '22:00', 'TUESDAY', 1),
	('08:00', '22:00', 'WEDNESDAY', 1),
	('08:00', '22:00', 'THURSDAY', 1),
	('08:00', '22:00', 'FRIDAY', 1),
	('08:00', '22:00', 'SATURDAY', 1),
	('10:00', '22:00', 'MONDAY', 2),
	('10:00', '22:00', 'TUESDAY', 2),
	('10:00', '22:00', 'WEDNESDAY', 2),
	('10:00', '22:00', 'THURSDAY', 2),
	('10:00', '22:00', 'FRIDAY', 2),
	('08:00', '20:00', 'MONDAY', 3),
	('08:00', '20:00', 'TUESDAY', 3),
	('08:00', '20:00', 'WEDNESDAY', 3),
	('08:00', '20:00', 'THURSDAY', 3),
	('10:00', '22:00', 'FRIDAY', 3),
	('08:00', '22:00', 'MONDAY', 4),
	('08:00', '22:00', 'TUESDAY', 4),
	('08:00', '22:00', 'WEDNESDAY', 4),
	('08:00', '22:00', 'THURSDAY', 4),
	('08:00', '22:00', 'FRIDAY', 4),
	('08:00', '22:00', 'MONDAY', 5),
	('08:00', '22:00', 'TUESDAY', 5),
	('08:00', '22:00', 'WEDNESDAY', 5),
	('08:00', '22:00', 'THURSDAY', 5),
	('08:00', '22:00', 'FRIDAY', 5),
	('08:00', '22:00', 'MONDAY', 6),
	('08:00', '22:00', 'TUESDAY', 6),
	('08:00', '22:00', 'WEDNESDAY', 6),
	('08:00', '22:00', 'THURSDAY', 6),
	('08:00', '22:00', 'FRIDAY', 6),
	('08:00', '22:00', 'MONDAY', 7),
```

```
        ('08:00', '22:00', 'TUESDAY', 7),
        ('08:00', '22:00', 'WEDNESDAY', 7),
        ('08:00', '22:00', 'THURSDAY', 7),
        ('08:00', '22:00', 'FRIDAY', 7);

-- Tabela: favorites / Instâncias: 7
INSERT INTO
        favorites (id_client, id_restaurant)
VALUES
        (1, 1),
        (1, 2),
        (1, 3),
        (1, 4),
        (2, 5),
        (2, 6),
        (3, 1);

-- Tabela: freights / Instâncias: 23
INSERT INTO
        freights (value, id_restaurant, id_district)
VALUES
        (3.50, 1, 1),
        (2.50, 1, 2),
        (4.50, 1, 3),
        (5.00, 1, 4),
        (5.00, 2, 5),
        (6.00, 2, 6),
        (5.50, 2, 7),
        (7.00, 2, 8),
        (5.00, 3, 9),
        (4.00, 3, 10),
        (3.00, 3, 11),
        (4.00, 4, 12),
        (4.00, 4, 13),
        (4.00, 4, 14),
        (5.00, 5, 15),
        (5.00, 5, 16),
        (5.00, 5, 17),
        (6.00, 6, 15),
        (6.00, 6, 16),
        (6.00, 6, 17),
        (3.00, 7, 15),
        (3.00, 7, 16),
        (3.00, 7, 17);

-- Tabela: clients_discount_coupons / Instâncias: 3
INSERT INTO
        clients_discount_coupons (id_client,
id_discount_coupon, id_order)
VALUES
        (1, 1, 1),
        (1, 2, 2),
        (2, 2, 3);
```

```
-- Tabela: clients_addresses_deliveries / Instâncias:
INSERT INTO
	clients_addresses_deliveries (id_clients_address,
id_delivery)
VALUES
	(1, 1),
	(1, 2),
	(2, 3),
	(4, 4),
	(4, 5),
	(4, 6),
	(6, 7),
	(4, 8),
	(1, 9),
	(3, 10),
	(4, 11),
	(8, 12),
	(9, 13),
	(1, 14),
	(2, 15),
	(4, 16);

-- Tabela: orders_products / Instâncias: 20
INSERT INTO
	orders_products (id_order, id_product, amount)
VALUES
	(1, 1, 1),
	(2, 2, 1),
	(3, 1, 1),
	(3, 3, 1),
	(4, 1, 1),
	(5, 2, 1),
	(6, 1, 1),
	(6, 2, 1),
	(7, 4, 1),
	(8, 5, 1),
	(9, 7, 1),
	(10, 10, 1),
	(11, 14, 1),
	(12, 19, 1),
	(13, 24, 1),
	(14, 22, 1),
	(14, 24, 1),
	(15, 22, 1),
	(16, 23, 1),
	(17, 23, 1);

-- Tabela: restaurant_owners_restaurants / Instâncias: 8
INSERT INTO
	restaurant_owners_restaurants
(id_restaurant_owner, id_restaurant)
VALUES
	(1, 1),
	(2, 1),
```

```
        (2, 2),
        (3, 3),
        (7, 4),
        (8, 5),
        (9, 6),
        (9, 7);

-- Tabela: restaurants_restaurant_categories / Instâncias:
14
INSERT INTO
        restaurants_restaurant_categories (id_restaurant,
id_restaurant_category)
VALUES
        (1, 1),
        (1, 2),
        (2, 1),
        (2, 2),
        (2, 6),
        (3, 2),
        (3, 5),
        (3, 7),
        (4, 3),
        (5, 1),
        (5, 2),
        (5, 4),
        (6, 3),
        (7, 5);
```

6. CONSULTAS EM SQL

(devem cobrir todas as sintaxes e casos-exemplo vistos em sala de aula; necessariamente incluir junção, de quatro formas distintas, junção externa, diferença e divisão; explicitar a pergunta que define a consulta e o comando em SQL necessário para conseguir a resposta à pergunta.)

Consulta 1: "Obter a lista de todos os pedidos feitos por um determinado cliente:"

```
SELECT * FROM orders WHERE id_client=1;
```

Consulta 2: "Obter a lista de todos os clientes e inclua os seus pedidos se houver algum:"

```
SELECT * FROM clients c
LEFT JOIN orders o ON c.id_user = o.id_client;
```

Consulta 3: “Quantas vezes cada cliente fez um pedido em seu restaurante favorito:”

```
SELECT u.id AS client_id, u.name, t2.id_restaurant,
r.name as restaurant_name, t2.total_pedido FROM
(
        SELECT COUNT(*) as total_pedido, t.id_client as
cliente, t.id_restaurant FROM
        (
          SELECT o.id, f.* FROM
          orders o,
          favorites f
          WHERE o.id_client = f.id_client
        ) t
        GROUP BY t.id_client, t.id_restaurant
) t2
JOIN restaurants r ON r.id = t2.id_restaurant
JOIN users u ON u.id = t2.cliente
ORDER BY t2.cliente
```

Consulta 4: “Obter a lista de todos os pedidos e inclua a sua avaliação se houver alguma:”

```
SELECT * FROM ratings
RIGHT JOIN orders ON ratings.id_order = orders.id;
```

Consulta 5: “Obter a lista de todos os produtos que nunca foram pedidos:”

```
select * from products AS p
FULL OUTER JOIN orders_products AS op ON p.id =
op.id_product
WHERE p.id is NULL OR op.id_product is NULL
```

Consulta 6: “Obter a lista de todos os produtos que já foram pedidos pelo menos uma vez:”

```
SELECT p.* from products p
WHERE exists
(
        SELECT op.id_product FROM orders_products AS
op
        WHERE op.id_product = p.id
)
```

```
ORDER BY p.id
```

Consulta 7: “Qual motoboy fez mais entregas no mês de janeiro:”

```
SELECT t.contagem, t.id_motoboy FROM
(
SELECT COUNT(d.id_motoboy) as contagem, d.id_motoboy FROM
orders AS o
INNER JOIN deliveries d ON d.id = o.id_delivery
WHERE extract(month FROM o.order_date) = 1
GROUP BY d.id_motoboy
) t
WHERE t.contagem =
(
        SELECT MAX(t2.contagem) FROM (
                SELECT COUNT(d.id_motoboy) as contagem, d.id_motoboy
FROM
                orders AS o
                INNER JOIN deliveries d ON d.id = o.id_delivery
                WHERE extract(month FROM o.order_date) = 1
                GROUP BY d.id_motoboy
        ) t2
)
```

Consulta 8: “Obter a lista de todos os produtos com preço entre R$ 10,00 e R$ 20,00:”

```
SELECT * FROM products WHERE price BETWEEN 10
AND 20;
```

Consulta 9: “ Qual é o pedido mais caro de um cliente:”

```
SELECT * FROM orders
WHERE id_client = 1
AND total_value= (
   SELECT MAX(total_value) FROM orders
   WHERE id_client = 1
);
```

Consulta 10: “Quanto cada restaurante já recebeu no total dos pedidos:”

```
SELECT SUM(o.total_value) AS valor_total, m.id_restaurant AS restaurant
FROM orders o, deliveries d, motoboys m
WHERE o.id_delivery = d.id
```

```
AND m.id = d.id_motoboy
GROUP BY m.id_restaurant
```

Consulta 11: Listar todos os produtos que já foram pedidos e a média de suas avaliações por ordem decrescente:”

```
SELECT p.id, p.name, AVG(r.vote) as media_avaliacoes
FROM products p
INNER JOIN orders_products op ON p.id = op.id_product
INNER JOIN orders o ON op.id_order = o.id
INNER JOIN ratings r ON o.id = r.id_order
GROUP BY p.id, p.name
ORDER BY media_avaliacoes DESC
```

Consulta 12: “Obter o frete de todos os bairros de um determinado restaurante e ordenar em ordem crescente de valor:”

```
SELECT d.* FROM restaurants AS r
INNER JOIN freights f ON f.id_restaurant = r.id
INNER JOIN districts d ON d.id = f.id_district
WHERE r.name = 'Pizzaria Stroele'
ORDER BY f.value
```

Consulta 13: “Listar todos os pedidos que foram avaliados com nota igual ou superior a 4:”

```
SELECT o.*
FROM orders o
WHERE EXISTS (
SELECT *
 FROM ratings r
WHERE r.id_order = o.id
AND r.vote >= 4
)
```

Consulta 14: “Listar todos os clientes que já fizeram pelo menos uma compra em um determinado período:”

```
SELECT c.id_user, u.name
FROM clients c
JOIN users u ON u.id = c.id_user
```

```
WHERE EXISTS (
SELECT *
FROM orders o
WHERE o.id_client = c.id_user
AND o.order_date BETWEEN '2022-05-01' AND
'2022-12-31'
)
```

Consulta 15: “Listar todos os restaurantes que pertencem a uma dada categoria:”

```
SELECT r.name
FROM restaurants r
JOIN restaurants_restaurant_categories rrc ON r.id =
rrc.id_restaurant
JOIN restaurant_categories rc ON
rrc.id_restaurant_category = rc.id
WHERE rc.name IN ('Pizzaria', 'Lanchonete')
```

7 MODELO FÍSICO (Parte 2)

7.1 Visões

(Determinar pelo menos 2 visões para usuários ou sistemas que vão acessar o teu banco de dados. Informar qual o ator que irá acessar a visão e o porque a visão ajudará esse sistema).

Visão 1: “Destinado ao dono do restaurante”

Objetivo: (A visão retorna os produtos mais vendidos de um mês de um restaurante, isso pode beneficiar o restaurante para melhorar a gestão de estoque e promoções dos produtos mais vendidos)

```
CREATE VIEW top_selling_products_of_month AS

SELECT p.name, SUM(op.amount) as quantidade,
EXTRACT(MONTH FROM o.order_date) as mes, r.name
as restaurant_name

FROM products p

JOIN orders_products op ON p.id = op.id_product

JOIN orders o ON op.id_order = o.id

JOIN restaurants r ON p.id_restaurant = r.id

GROUP BY p.id, EXTRACT(MONTH FROM
o.order_date), r.id

ORDER BY quantidade DESC, mes;
```

```
SELECT * FROM top_selling_products_of_month WHERE mes =
EXTRACT(MONTH FROM CURRENT_DATE)
AND restaurant_name = 'Pizzaria Stroele';
```

Visão 2: “Destinado aos clientes do sistema”

Objetivo: (A visão retorna todos os produtos de um restaurante, isso irá ajudar o cliente a consultar o cardápio de determinado restaurante)

```
CREATE VIEW restaurant_products AS

SELECT p.* FROM products p

JOIN restaurants r ON p.id_restaurant = r.id;
```

```
SELECT * FROM restaurant_products
WHERE id_restaurant = 1;
```

7.2 Triggers e Funções
(pelo menos duas funções acionadas por meio de triggers)

Trigger / Procedure 1: "Atualizar a média e contagem da avaliação de um restaurante, quando uma nova avaliação é criada"

```
CREATE TYPE rating_report AS (id_restaurant int,
rating_average numeric, rating_count int);

CREATE OR REPLACE FUNCTION
rating_report_by_restaurant()
	RETURNS SETOF rating_report AS
$$
	SELECT
		t.id_restaurant,
		AVG(r1.vote) AS rating_average,
		COUNT(*) AS rating_count
	FROM
		ratings r1
	INNER JOIN
		(
			SELECT
				p.id_restaurant,
				r2.id_order
			FROM
				ratings r2
				INNER JOIN
orders_products op
					ON r2.id_order =
op.id_order
				INNER JOIN products p
					ON op.id_product =
p.id
			WHERE
				r2.vote IS NOT NULL
```

```
                        GROUP BY
                                p.id_restaurant,
                                r2.id_order
                ) t
        ON r1.id_order = t.id_order
        GROUP BY
                t.id_restaurant
        ORDER BY
                rating_average DESC
$$
LANGUAGE SQL;


CREATE OR REPLACE FUNCTION
restaurant_rating_report(id_rest int)
        RETURNS rating_report AS
$$
        DECLARE
                r_report rating_report;
        BEGIN
                SELECT
                        *
                FROM
                        rating_report_by_restaurant() ra
                WHERE
                        ra.id_restaurant = id_rest
                INTO
                        r_report;


                RETURN r_report;
        END;
```

```
$$
LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION
update_restaurant_rating()
	RETURNS trigger AS
$$
	DECLARE
		id_rest int;
		rest_rating_report rating_report;
	BEGIN
		IF (TG_OP = 'DELETE') THEN
			SELECT
				p.id_restaurant
			FROM
				orders_products op
				INNER JOIN products p
					ON op.id_product =
p.id
			WHERE
				op.id_order = old.id_order
			LIMIT
				1
			INTO
				id_rest;
		ELSE
			SELECT
				p.id_restaurant
			FROM
				orders_products op
				INNER JOIN products p
```

```
                                        ON op.id_product =
p.id
                        WHERE
                                op.id_order = new.id_order
                        LIMIT
                                1
                        INTO
                                id_rest;
                END IF;


                SELECT * FROM
restaurant_rating_report(id_rest)
                INTO rest_rating_report;


                IF (rest_rating_report.rating_count IS NULL)
THEN
                        UPDATE
                                restaurants r
                        SET
                                rating_average =
rest_rating_report.rating_average,
                                rating_count = 0
                        WHERE
                                r.id = id_rest;
                ELSE
                        UPDATE
                                restaurants r
                        SET
                                rating_average =
rest_rating_report.rating_average,
                                rating_count =
rest_rating_report.rating_count
                        WHERE
```

```
                                r.id = id_rest;
        END IF;


        IF (TG_OP = 'DELETE') THEN
                RETURN old;
        ELSE
                RETURN new;
        END IF;
    END;
$$
LANGUAGE plpgsql;


CREATE TRIGGER restaurant_rating
    AFTER INSERT OR UPDATE ON ratings
    FOR EACH ROW
    WHEN (new.vote IS NOT NULL)
    EXECUTE PROCEDURE
update_restaurant_rating();


CREATE TRIGGER restaurant_rating_on_delete
    AFTER DELETE ON ratings
    FOR EACH ROW
    WHEN (old.vote IS NOT NULL)
    EXECUTE PROCEDURE
update_restaurant_rating();
```

Trigger / Procedure 2: Atualizar o status de um pedido para ‘FINISHED’, quando a entrega do pedido tiver seu status modificado para ‘DELIVERED’

```
CREATE OR REPLACE FUNCTION finish_order()
	RETURNS trigger AS
$$
	BEGIN
		UPDATE
			orders
		SET
			status = 'FINISHED'
		WHERE
			id_delivery = new.id;

		RETURN NEW;
	END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER order_status
	AFTER UPDATE ON deliveries
	FOR EACH ROW
	WHEN (new.status = 'DELIVERED')
	EXECUTE PROCEDURE finish_order();
```

7.3 Verificação

(INSERTs, UPDATEs ou DELETEs que demonstram o correto comportamento dos controles acionados através das triggers.)

Verificação Trigger / Procedure 1: “Atualizar a média e contagem da avaliação de um restaurante, quando uma nova avaliação é criada”

```
– Quando uma instância na tabela de ratings é atualizada,
ou quando uma nova instância é criada, ou quando uma
instância é deletada, os campos rating_average e
rating_count na tabela restaurants são recalculados.

UPDATE ratings SET vote = 3 WHERE id = 3;

INSERT INTO
	ratings (vote, feedback, id_order)
VALUES
	(5, NULL, 11);

DELETE FROM ratings where id = 10;
```

Verificação Trigger / Procedure 2: Atualizar o status de um pedido para ‘FINISHED’, quando a entrega do pedido tiver seu status modificado para ‘DELIVERED’
– Quando uma instância na tabela de deliveries tem o status modificado para ‘DELIVERED’, o status do pedido que gerou essa entrega é atualizado para ‘FINISHED’. UPDATE deliveries SET status = 'DELIVERED' WHERE id = 3;

8 OTIMIZAÇÃO DO BANCO

8.1 Índices

(definição e criação de índices secundários úteis, a serem utilizados nas consultas)

Índice 1: Índice para busca de restaurante por nome.
CREATE INDEX idx_restaurant_name ON restaurants (name);

Índice 2: Índice para facilitar a consulta à tabela que relaciona pedidos com seus produtos, pois é utilizada como tabela pivot em muitas consultas
CREATE INDEX idx_orders_products ON orders_products (id_order, id_product);

8.2 Verificação dos índices

(realizar testes com os índices criados para mostrar a diferença de desempenho de consultas com e sem o índice e porque o índice escolhido foi o melhor. Colocar uma tabela com os resultados de comparação. Realizar experimentos com pelo menos 2 consultas SQL).

Verificação Índice 1

Índice para busca de restaurante por nome. Em um cenário com muitos restaurantes, é bastante comum a realização de um filtro por nome de restaurante que precisa de um retorno rápido da consulta.

```
– Consulta 1
SELECT * FROM restaurants
WHERE name = 'Pizzaria Stroele';

– Consulta 2
SELECT d.* FROM restaurants AS r
INNER JOIN freights f ON f.id_restaurant = r.id
INNER JOIN districts d ON d.id = f.id_district
WHERE r.name = 'Pizzaria Stroele'
ORDER BY f.value
```

Verificação Índice 2

Índice para facilitar a consulta à tabela que relaciona pedidos com seus produtos, pois é utilizada como tabela pivot em muitas consultas.

```
-- Consulta 1
SELECT p.id, p.name, AVG(r.vote) as media_avaliacoes
FROM products p
INNER JOIN orders_products op ON p.id = op.id_product
INNER JOIN orders o ON op.id_order = o.id
INNER JOIN ratings r ON o.id = r.id_order
GROUP BY p.id, p.name
ORDER BY media_avaliacoes DESC

-- Consulta 2
SELECT * FROM restaurant_rating_report(1);
```

Tabelas com carga para teste:

- restaurants: 73728 registros
- orders_products: 245760 registros

Comparação de tempos (Consulta 1):

	Tempo sem índice	Tempo com índice
Índice 1	0.08s	0.05s
Índice 2	0.14s	0.06s

Comparação de tempos (Consulta 2):

	Tempo sem índice	Tempo com índice
Índice 1	0.08s	0.05s
Índice 2	0.14s	0.09s

9. TELAS DA APLICAÇÃO

(Criar o código, em qualquer linguagem de programação, de um protótipo de aplicação. Mostrar algumas telas da aplicação e relacioná-las com os requisitos de negócio da seção 2. O sistema não precisará ser completo. Basta implementar uma ou duas telas para inserção e resultados de consultas que sejam interessantes para a aplicação que está desenvolvendo, de forma a demonstrar que é capaz de criar uma aplicação que se comunique com um SGBD relacional. O código do sistema deve ser enviado em conjunto com o relatório.)