Apresentação Banco de Dados

Grupo: Luis Freitas, Raffael Paranhos e Raphael Leardini

1. INTRODRUÇÃO

TEMA

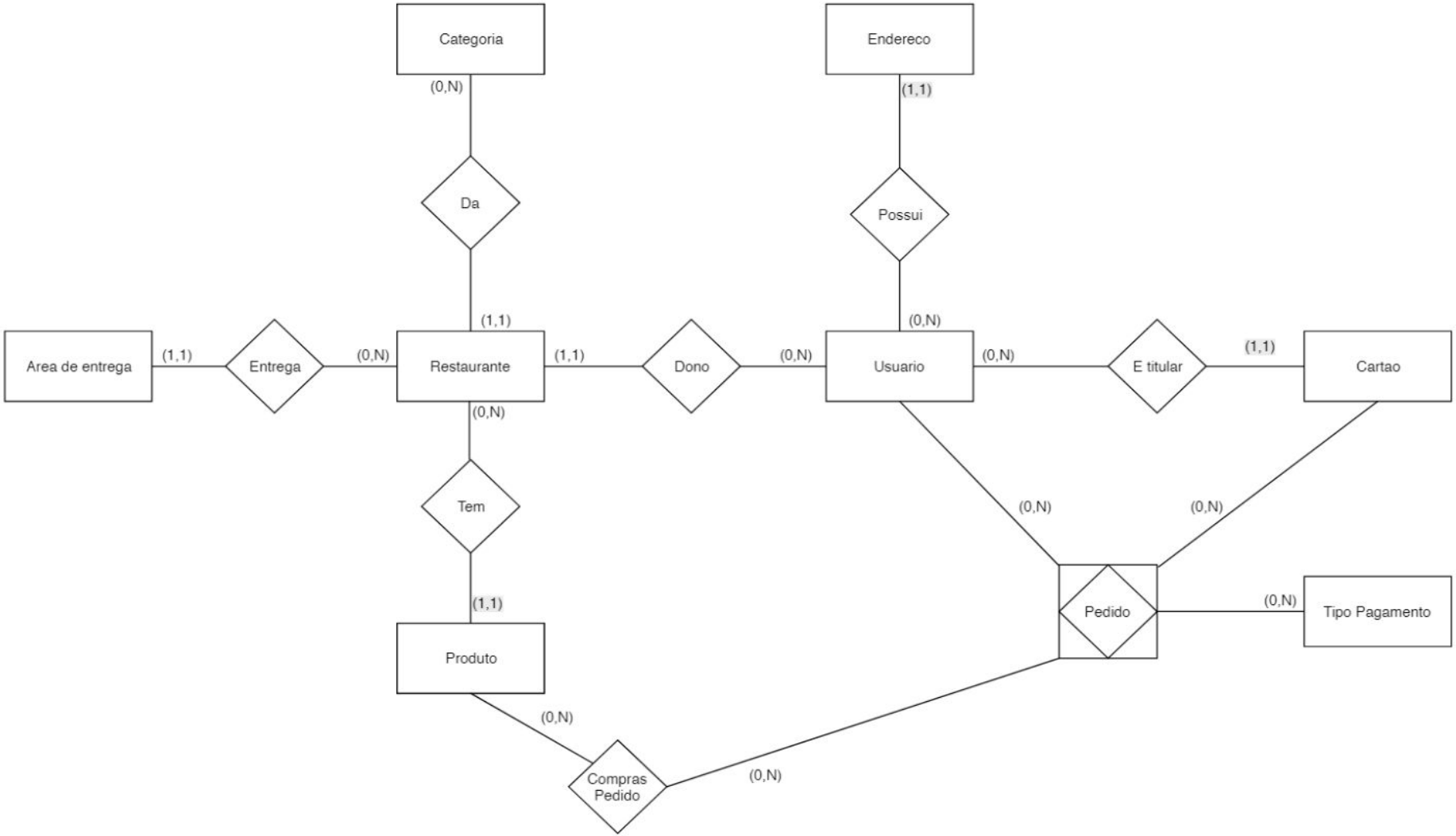
Escolhemos fazer uma interpretação do **iFood** pois:

- Nós usamos ele diariamente.
- Já possuíamos uma ideia de quais Regras de Negócio e Processos poderíamos criar.
- Era uma das opções.

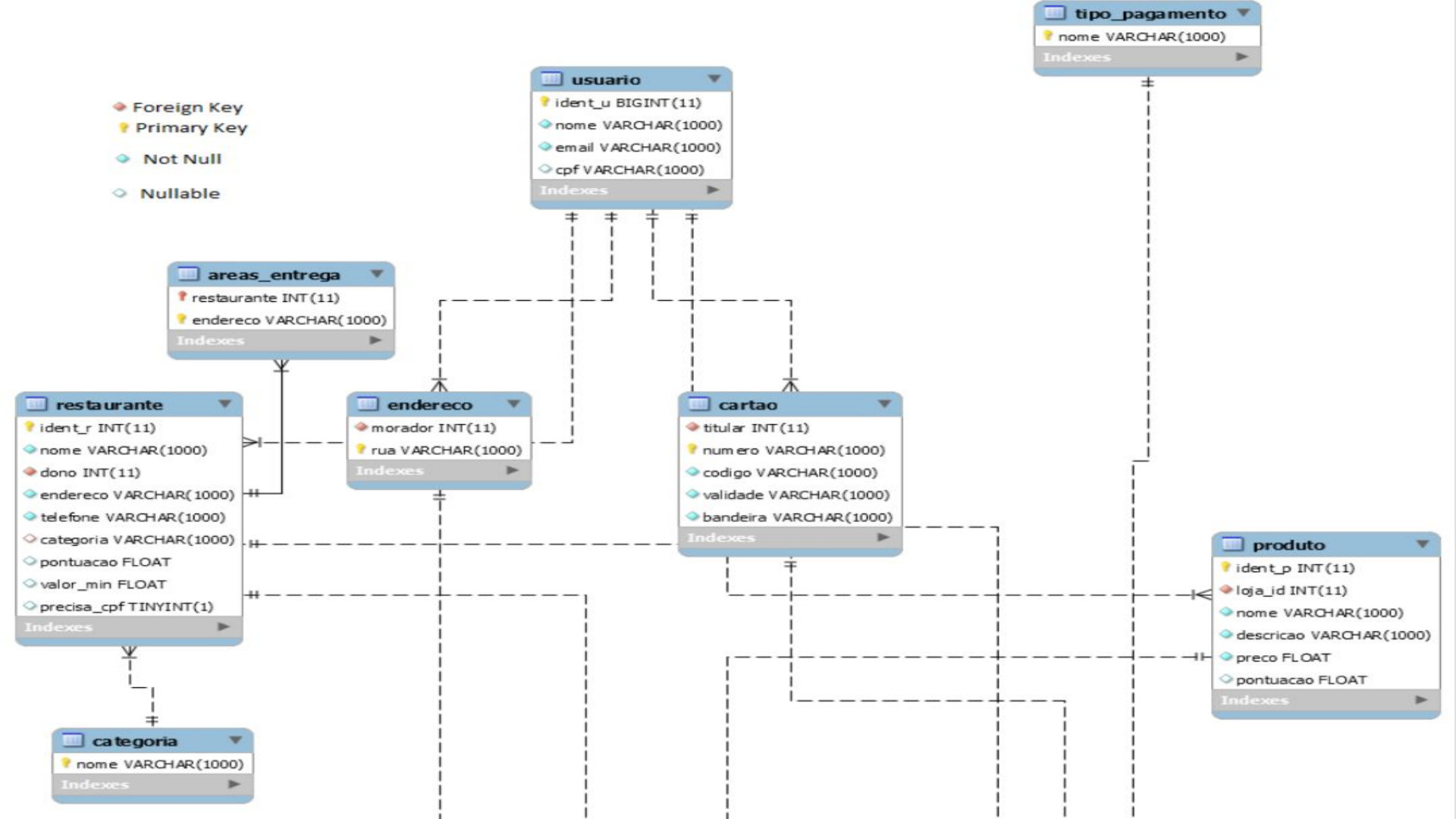
DETALHES

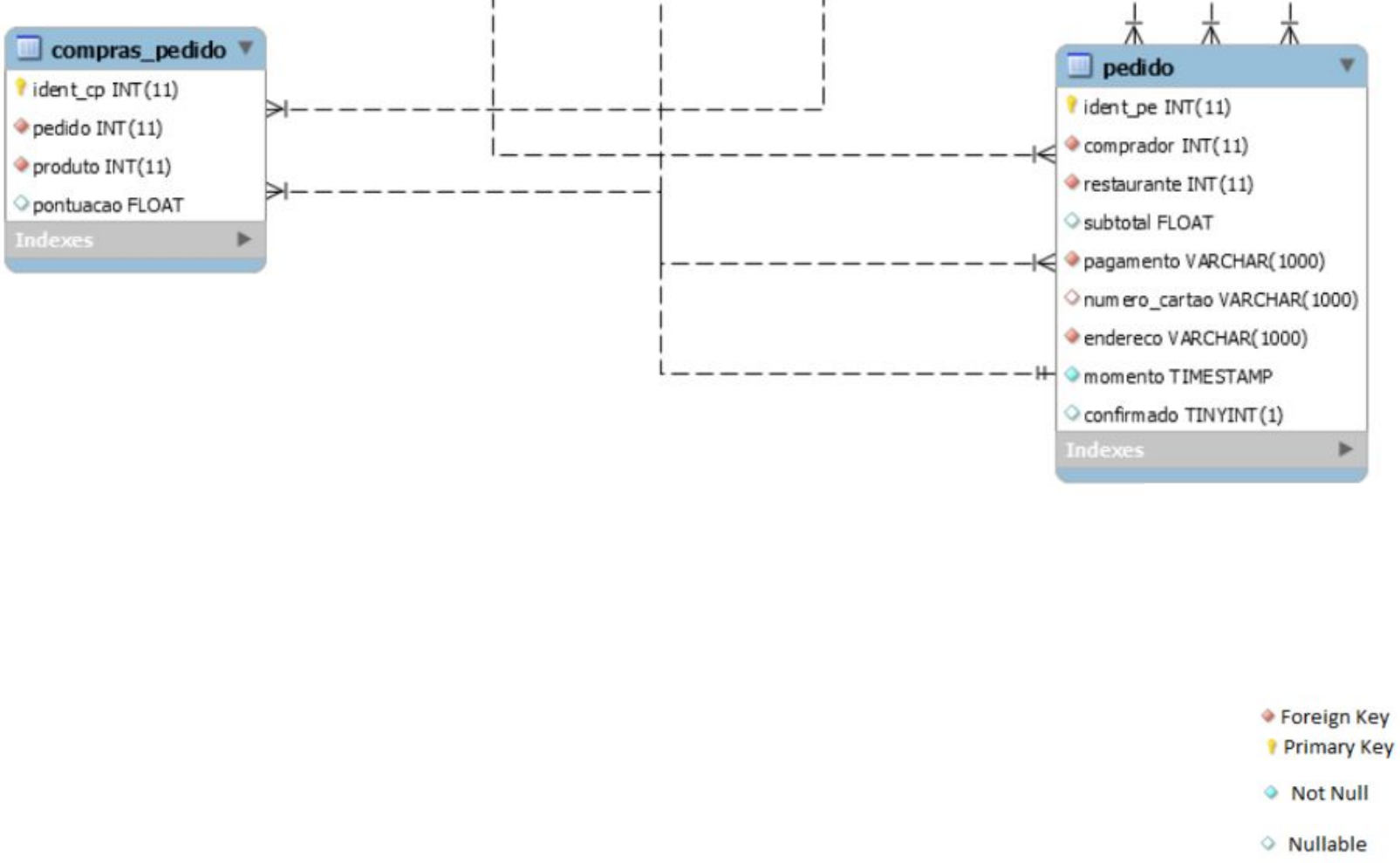
- A linguagem utilizada foi ***plpgsql.***
- Criamos no total:
  - 10 Tabelas, 10 Chaves Primárias e 13 Chaves Estrangeiras.
  - 10 ***Triggers***.
  - 2 ***Procedures***.

2. DIAGRAMA ENTIDADE-RELACIONAMENTO


Categoria
Endereco
(0,N)
(1,1)
Da
Possui
(1,1)
(0,N)
Area de entrega
(1,1)
Entrega
(0,N)
Restaurante
(1,1)
Dono
(0,N)
Usuario
(0,N)
E titular
(1,1)
Cartao
(0,N)
(0,N)
(0,N)
Tem
Pedido
(0,N)
Tipo Pagamento
(1,1)
Produto
(0,N)
Compras
Pedido
(0,N)

3.
DIAGRAMA MODELO LÓGICO RELACIONAL


Foreign Key
Primary Key
Not Null
Nullable
tipo_pagamento
nome VARCHAR(1000)
Indexes
usuario
ident_u BIGINT(11)
nome VARCHAR(1000)
email VARCHAR(1000)
cpf VARCHAR(1000)
Indexes
areas_entrega
restaurante INT(11)
endereco VARCHAR(1000)
Indexes
restaurante
ident_r INT(11)
nome VARCHAR(1000)
dono INT(11)
endereco VARCHAR(1000)
telefone VARCHAR(1000)
categoria VARCHAR(1000)
pontuacao FLOAT
valor_min FLOAT
precisa_cpf TINYINT(1)
Indexes
endereco
morador INT(11)
rua VARCHAR(1000)
Indexes
cartao
titular INT(11)
numero VARCHAR(1000)
codigo VARCHAR(1000)
validade VARCHAR(1000)
bandeira VARCHAR(1000)
Indexes
produto
ident_p INT(11)
loja_id INT(11)
nome VARCHAR(1000)
descricao VARCHAR(1000)
preco FLOAT
pontuacao FLOAT
Indexes
categoria
nome VARCHAR(1000)
Indexes


compras_pedido
ident_cp INT(11)
pedido INT(11)
produto INT(11)
pontuacao FLOAT
Indexes
pedido
ident_pe INT(11)
comprador INT(11)
restaurante INT(11)
subtotal FLOAT
pagamento VARCHAR(1000)
numero_cartao VARCHAR(1000)
endereco VARCHAR(1000)
momento TIMESTAMP
confirmado TINYINT(1)
Indexes
Foreign Key
Primary Key
Not Null
Nullable

4.
TRIGGERS

```
CREATE TRIGGER check_pontuacao
BEFORE INSERT ON compras_pedido
FOR EACH ROW
EXECUTE PROCEDURE check_pontuacao();

CREATE OR REPLACE FUNCTION check_pontuacao() RETURNS trigger AS $$
BEGIN
        if (new.pontuacao > 5 or new.pontuacao <0) then
                raise exception 'pontuacao fora do range 0 < x < 5';
        end if;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER calc_pontuacao_produto
AFTER INSERT ON compras_pedido
FOR EACH ROW
EXECUTE PROCEDURE calc_pontuacao_produto();

CREATE OR REPLACE FUNCTION calc_pontuacao_produto() RETURNS trigger AS $$
DECLARE
 curs cursor for select pontuacao from compras_pedido where produto = new.produto;
 soma float = 0;
 contador int = 0;
BEGIN
        for record in curs loop
                soma = soma + record.pontuacao;
                contador = contador + 1;
        end loop;
        update produto
        set pontuacao = soma/contador
        where ident_p = new.produto;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER calc_pontuacao_restaurante
AFTER UPDATE ON produto
FOR EACH ROW
EXECUTE PROCEDURE calc_pontuacao_restaurante();

CREATE OR REPLACE FUNCTION calc_pontuacao_restaurante() RETURNS trigger AS $$
DECLARE
 curs cursor for select pontuacao from produto where loja_id = new.loja_id;
 soma float = 0;
 contador int = 0;
BEGIN
        for record in curs loop
                soma = soma + record.pontuacao;
                contador = contador + 1;
        end loop;
        update restaurante
        set pontuacao = soma/contador
        where ident_r = new.loja_id;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER subtotal_pedido
AFTER INSERT ON compras_pedido
FOR EACH ROW
EXECUTE PROCEDURE subtotal_pedido();

CREATE OR REPLACE FUNCTION subtotal_pedido() RETURNS trigger AS $$
DECLARE
        soma float = 0;
        atual float = 0;
BEGIN
        select subtotal
        into soma
        from pedido
        where ident_pe = new.pedido;
        select preco
        into atual
        from produto
        where ident_p = new.produto;
        soma = soma + atual;
        update pedido
        set subtotal = soma
        where ident_pe = new.pedido;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER check_valor_min
BEFORE UPDATE of confirmado ON pedido
FOR EACH ROW
WHEN (new.confirmado = true)
EXECUTE PROCEDURE check_valor_min();

CREATE OR REPLACE FUNCTION check_valor_min() RETURNS trigger AS $$
DECLARE
        valor_min_f float = 0;
BEGIN
        select valor_min
        into valor_min_f
        from restaurante
        where ident_r = new.restaurante;
        if(valor_min_f = null) then

        else
                if(new.subtotal < valor_min_f) then
                        raise exception 'valor do pedido menor que o minimo da loja';
                end if;
        end if;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER check_cpf
BEFORE INSERT ON pedido
FOR EACH ROW
EXECUTE PROCEDURE check_cpf();

CREATE OR REPLACE FUNCTION check_cpf() RETURNS trigger AS $$
DECLARE
        precisa_cpf_bool boolean;
        cpf_usuario varchar;
BEGIN
        select precisa_cpf
        into precisa_cpf_bool
        from restaurante
        where ident_r = new.restaurante;
        if (precisa_cpf_bool = true) then
                select cpf
                into cpf_usuario
                from usuario
                where ident_u = new.comprador;
                raise notice '%f',cpf_usuario;
                if(cpf_usuario is null) then
                        raise exception 'usuario nao possui cpf cadastrado';
                end if;
        end if;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER check_pedido
BEFORE INSERT ON compras_pedido
FOR EACH ROW
EXECUTE PROCEDURE check_pedido();

CREATE OR REPLACE FUNCTION check_pedido() RETURNS trigger AS $$
DECLARE
        curs cursor for select produto from compras_pedido where pedido = new.ident_pe;
        soma float = 0;
        loja_id_f int;
        loja_produto int;
BEGIN
        select restaurante
        into loja_id_f
        from pedido
        where ident_pe = new.pedido;

        select loja_id
        into loja_produto
        from produto
        where ident_p = new.produto;

        if(loja_id_f <> loja_produto) then
                raise exception 'produto nao pertence a loja do pedido';
        end if;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER check_endereco
BEFORE INSERT ON pedido
FOR EACH ROW
EXECUTE PROCEDURE check_endereco()

CREATE OR REPLACE FUNCTION check_endereco() RETURNS trigger AS $$
DECLARE
        curs cursor for select endereco from areas_entrega where restaurante = new.restaurante;
        endereco_restaurante varchar;
BEGIN
        for record in curs loop
                if(new.endereco = record.endereco) then
                        return new;
                end if;
        end loop;
        select endereco
        into endereco_restaurante
        from restaurante
        where ident_r = new.restaurante;
        if(new.endereco = endereco_restaurante ) then
                return new;
        end if;
        raise exception 'endereco de entrega do pedido nao faz parte da area de entrega do restaurante';
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER impede_valormin_restaurante
BEFORE UPDATE OF valor_min ON restaurante
FOR EACH ROW
EXECUTE PROCEDURE impede_valormin_restaurante();

CREATE OR REPLACE FUNCTION impede_valormin_restaurante() RETURNS trigger AS $$
BEGIN
        raise exception 'valor minimo nao pode ser alterado, contate o suporte';
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER impede_preco_produto
BEFORE UPDATE OF preco OR DELETE ON produto
FOR EACH ROW
EXECUTE PROCEDURE impede_preco_produto();

CREATE OR REPLACE FUNCTION impede_preco_produto() RETURNS trigger AS $$
BEGIN
        raise exception 'preco nao pode ser alterado, contate o suporte';
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

5. PROCEDURES

```
create or replace function total_vendido_produto() returns
        table(ident_p int,
                qtd_vendas int,
                total float) as $$
declare
        c_cp cursor (idt int) for select * from compras_pedido where produto=idt for share of compras_pedido;
        r1 produto%rowtype;
        r2 compras_pedido%rowtype;
        qtd_vendas int;
        total float;
begin
        for r1 in select * from produto loop
                qtd_vendas = 0;
                total = 0;
                for r2 in c_cp(r1.ident_p) loop
                        qtd_vendas = qtd_vendas + 1;
                end loop;
                total = r1.preco * qtd_vendas;
                return query select r1.ident_p, qtd_vendas, total;
        end loop;
end; $$ language 'plpgsql';
```

```
create or replace function total_vendido_restaurante() returns
        table(restaurante varchar,
                qtd_vendas_f int,
                total_f float) as $$
declare
        curs cursor for select produto.nome,descricao,preco,produto.pontuacao,qtd_vendas,total,restaurante.nome
        as nome_res,restaurante.pontuacao
                                        from total_vendido_produto() as p
                                        inner join produto on p.ident_p = produto.ident_p
                                        inner join restaurante on produto.loja_id = restaurante.ident_r
                                        order by restaurante.nome;
        qtd_vendas_f int;
        total_f float;
        res_ant varchar;
begin
        for record in curs loop
                if(record.nome_res <> res_ant or res_ant is null) then
                        if(res_ant is not null) then
                                return query select res_ant, qtd_vendas_f, total_f;
                        end if;
                        qtd_vendas_f = 0;
                        total_f = 0;
                        res_ant = record.nome_res;
                end if;
                qtd_vendas_f = qtd_vendas_f + record.qtd_vendas;
                total_f = total_f + record.total;
        end loop;
        return query select res_ant, qtd_vendas_f, total_f;
end; $$ language 'plpgsql';
```