

Modelagem de Banco de Dados Relacional

Projeto: e-commerce Grupo HILACA



Objetivo:

Descrever a estrutura de dados, relacionamentos e regras de negócio para o projeto de *e-commerce* da loja HILACA.

Análise de Requisitos:

ENTIDADES PRINCIPAIS:

1. Cliente:

Atributo 1: id_cliente (chave primária)
Atributo 2: nome
Atributo 3: endereço
Atributo 4: e-mail (chave primária)
Atributo 5: pontuação
Atributo 6: RG
Atributo 7: CPF/CNPJ (chave primária)
Atributo 8: data de nascimento
Atributo 9: telefone (1, 2)

2. Pontuação cliente:

Atributo 1: id_pontos Acumulados
Atributo 2: pontos gerados
Atributo 3: data da pontuação

3. Pedido

Atributo 1: código do pedido (chave primária)
Atributo 2: quantidade
Atributo 3: valor total
Atributo 4: data do pedido
Atributo 5: data de entrega
Atributo 6: status do pedido
Atributo 7: forma de entrega

4. Produto:

Atributo 1: código produto (chave primária)
Atributo 2: nome do produto
Atributo 3: peso
Atributo 4: descrição do produto
Atributo 5: quantidade
Atributo 6: preço

5. Estoque:

Atributo 1: id_estoque

Atributo 2: quantidade disponível

6. Categoria produto:

Atributo 1: ID categoria (chave primária)

Atributo 2: nome da categoria

Atributo 3: descrição da categoria

RELACIONAMENTOS:

- Um Cliente pode fazer vários Pedidos.
- Um Pedido pode conter vários Itens do Pedido.
- Um Produto pode pertencer a uma ou várias Categorias de Produto.
- Um Produto pode estar presente em várias entradas de Estoque.
- Um Pedido é associado a um Cliente.
- Um Item do Pedido está relacionado a um Produto e a um Pedido.
- Um Cliente acumula Pontuação ao fazer Pedidos.

Modelagem Conceitual:

Foi definido a loja HILACA do Grupo HILACA como o e-commerce para fazer a modelagem.

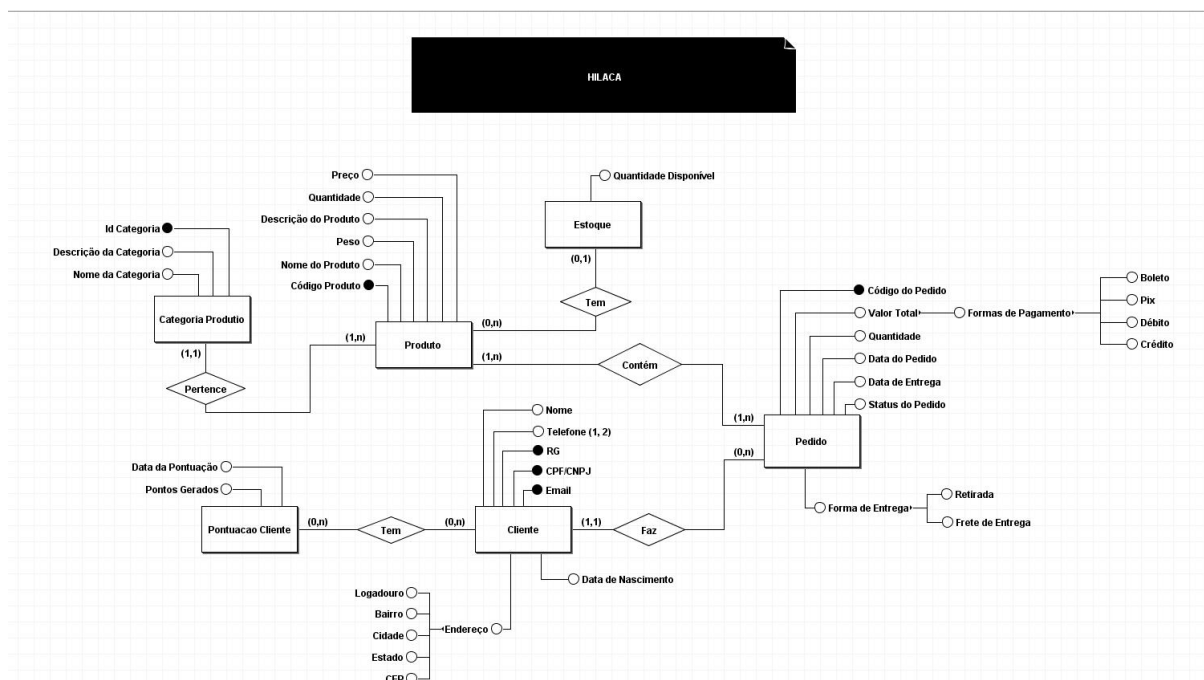


Imagem 1 - Modelo Conceitual do Projeto

Modelagem Física (PostgreSQL):

Na imagem abaixo, tem-se a representação do Modelo Físico, mostrando o relacionamento entre tabelas.

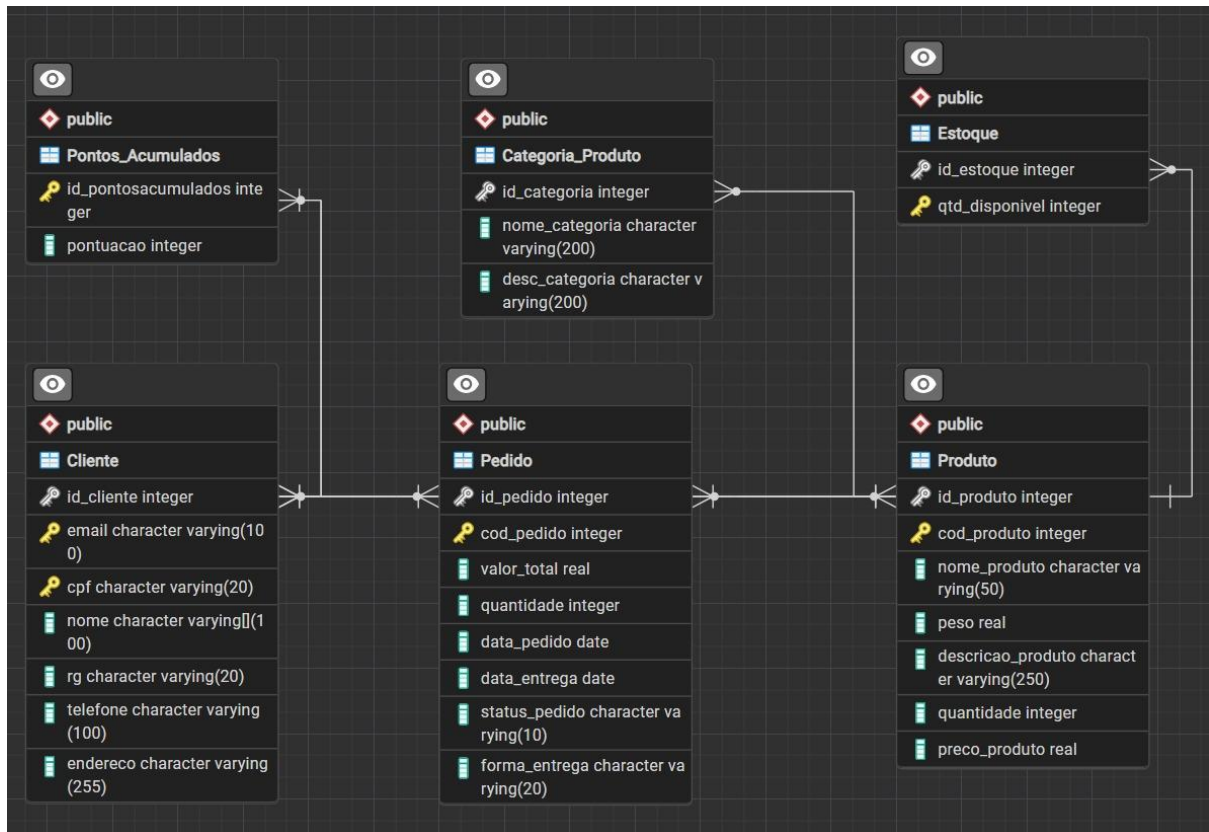


Imagem 2 - Modelo Físico do projeto

E em seguida, tem-se a imagem sendo executada e o script SQL para a criação das tabelas no PostgreSQL, com as chaves primárias, as chaves estrangeiras e os índices:

- Script SQL:

Na imagem abaixo tem-se a criação das tabelas:

```
Consulta  Histórico de consultas

1  -- This script was generated by the ERD tool in pgAdmin 4.
2  -- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.
3  BEGIN;
4
5
6  CREATE TABLE IF NOT EXISTS public."Cliente"
7  (
8      id_cliente integer NOT NULL DEFAULT nextval('"Cliente_id_cliente_seq"'::regclass),
9      email character varying(100) COLLATE pg_catalog."default" NOT NULL,
10     cpf character varying(20) COLLATE pg_catalog."default" NOT NULL,
11     nome character varying(100)[] COLLATE pg_catalog."default" NOT NULL,
12     rg character varying(20) COLLATE pg_catalog."default" NOT NULL,
13     telefone character varying(100) COLLATE pg_catalog."default" NOT NULL,
14     endereco character varying(255) COLLATE pg_catalog."default" NOT NULL,
15     CONSTRAINT "Cliente_pkey" PRIMARY KEY (cpf, email),
16     CONSTRAINT "Cliente_cpf_key" UNIQUE (cpf),
17     CONSTRAINT "Cliente_id_cliente_key" UNIQUE (id_cliente)
18 );
19
20
Data Output  Mensagens  Notificações

NOTA: relação "Cliente" já existe, ignorando
CREATE TABLE

Consulta retornada com sucesso em 81 ms.
```

Observação: a mensagem de erro é apresentada porque a tabela já havia sido criada.

Na imagem abaixo tem-se a criação das chaves estrangeiras:

```
Consulta  Histórico de consultas

70
71
72
73
74
75
76
77
78  ALTER TABLE IF EXISTS public."Cliente"
79      ADD FOREIGN KEY (id_cliente)
80      REFERENCES public."Pedido" (id_pedido) MATCH SIMPLE
81      ON UPDATE NO ACTION
82      ON DELETE NO ACTION
83      NOT VALID;
84
85
86  ALTER TABLE IF EXISTS public."Categoria_Produto"
87      ADD FOREIGN KEY (id_categoria)
88      REFERENCES public."Produto" (id_produto) MATCH SIMPLE
89      ON UPDATE NO ACTION
90      ON DELETE NO ACTION
91      NOT VALID;
92
93
94  ALTER TABLE IF EXISTS public."Estoque"
95      ADD FOREIGN KEY (id_estoque)
96
Data Output  Mensagens  Notificações

NOTA: relação "Cliente" já existe, ignorando
CREATE TABLE

Consulta retornada com sucesso em 81 ms.
```

Observação: a mensagem de erro é apresentada porque a tabela já havia sido criada.

Tabela Cliente

```
DROP TABLE public."Cliente"
CREATE TABLE IF NOT EXISTS public."Cliente"
(
    id_cliente serial NOT NULL UNIQUE,
    email character varying (100) NOT NULL UNIQUE,
    cpf character varying (20) NOT NULL UNIQUE,
    nome character varying (100) [] NOT NULL,
    rg character varying (20) NOT NULL UNIQUE,
    telefone character varying (100) NOT NULL UNIQUE,
    endereco character varying (255) NOT NULL,
    PRIMARY KEY (cpf, email)
);
```

Tabela Pontos Acumulados

```
DROP TABLE public."Pontos_Acumulados"
CREATE TABLE IF NOT EXISTS public."Pontos_Acumulados"
(
    id_pontosAcumulados SERIAL PRIMARY KEY,
    pontuacao integer NOT NULL,
    FOREIGN KEY (id_pontosAcumulados)
    REFERENCES public."Cliente" (id_cliente)
);
```

Tabela Pedidos

```
DROP TABLE public."Pedido"
CREATE TABLE IF NOT EXISTS public."Pedido"
(
    id_pedido SERIAL UNIQUE,
    cod_pedido integer NOT NULL PRIMARY KEY,
    valor_total real NOT NULL,
    quantidade integer NOT NULL,
    data_pedido date NOT NULL,
    data_entrega date NOT NULL,
    status_pedido character varying (10) NOT NULL,
    forma_entrega character varying (20) NOT NULL,
    FOREIGN KEY (id_pedido)
```

```
REFERENCES public."Cliente" (id_cliente)
);
```

Tabela Produtos

```
DROP TABLE public."Produto"
CREATE TABLE IF NOT EXISTS public."Produto"
(
    id_produto SERIAL UNIQUE,
    cod_produto integer NOT NULL PRIMARY KEY,
    nome_produto character varying (50) NOT NULL,
    peso real NOT NULL,
    descricao_produto character varying (250) NOT NULL,
    quantidade integer NOT NULL,
    preco_produto real NOT NULL,
    FOREIGN KEY (id_produto)
    REFERENCES public."Pedido" (id_pedido)
);
```

Tabela Estoque

```
DROP TABLE public."Estoque"
CREATE TABLE IF NOT EXISTS public."Estoque"
(
    id_estoque SERIAL UNIQUE,
    qtd_disponivel integer NOT NULL PRIMARY KEY,
    FOREIGN KEY (id_estoque)
    REFERENCES public."Produto" (id_produto)
);
```

Tabela Categoria Produto

```
DROP TABLE public."Categoria_Produto";
CREATE TABLE IF NOT EXISTS public."Categoria_Produto"
(
    id_categoria SERIAL NOT NULL UNIQUE,
    nome_categoria character varying (200) NOT NULL,
```

```
desc_categoria character varying (200) NOT NULL,  
FOREIGN KEY (id_categoria)  
REFERENCES public."Produto" (id_produto)  
);
```

Abaixo seguem os os inserts que criados para popular o banco de dados.

INSERT Cliente

```
INSERT INTO public."Cliente"(email,cpf,nome,rg,telefone,endereco)  
VALUES  
( 'a.felis@outlook.org','11.111.111-1','{Hikaru Yamanaka}','000.000.000-01','(614)  
776-1318','Ap #548-7547 Commodo Avenue'),  
( 'duis.risus.odio@aol.couk','11.111.111-2','{Layanne Mary}','000.000.000-02','(546)  
479-5384','Ap #673-6494 Ac Rd.'),  
( 'dados.carolyne@gmail.com','11.111.111-3','{Carolyne Oliveira}','000.000.000-  
03','1-473-553-0474','Ap #891-5968 Placerat Street'),  
( 'facilisi@protonmail.edu','11.111.111-4','{Deise Pestana}','000.000.000-04','1-153-  
313-8753','199-3157 Risus. St.'),  
( 'ipsum@outlook.edu','11.111.111-5','{Sacha Le}','000.000.000-05','(309) 850-  
6925','Ap #394-5985 Ut Street');
```

INSERT Pontos Acumulados

```
INSERT INTO "Pontos_Acumulados" (pontuacao)  
VALUES  
(1000),  
(2000),  
(3000),  
(4000),  
(5000);
```

INSERT Pedido

```
INSERT INTO "Pedido" (cod_pedido, valor_total, quantidade, data_pedido,  
data_entrega, status_pedido, forma_entrega)
```


VALUES

```
(1, 100, 2, '2024-01-01', '2024-01-02', 'Enviado', 'Entrega em domicílio'),  
(2, 200, 1, '2024-01-03', '2024-01-04', 'Pendente', 'Retirada na loja'),  
(3, 150, 3, '2024-01-05', '2024-01-06', 'Enviado', 'Entrega em domicílio'),  
(4, 300, 4, '2024-01-07', '2024-01-08', 'Pendente', 'Retirada na loja'),  
(5, 250, 5, '2024-01-09', '2024-01-10', 'Enviado', 'Entrega em domicílio');
```

INSERT Produto

```
INSERT INTO public."Produto"(cod_produto, nome_produto, peso,  
descricao_produto, quantidade, preco_produto)
```

VALUES

```
(1, 'Hidratante Corporal', 500.0, 'Creme Hidratante Corporal Sem Cheiro', 3, 100.0),  
(2, 'Shampoo', 150.0, 'Shampoo para Cabelos mistos e oleosos', 1, 200.0),  
(3, 'Lip Gloss', 13.0, 'Brilho Labial sabor cereja', 10, 300.0),  
(4, 'Condicionador', 150.0, 'Condicionador para todos os tipos de cabelo', 4, 400.0),  
(5, 'Hidratante Facial', 80.0, 'Hidratante facial com vitamina E', 2, 500.0);
```

INSERT Estoque

```
INSERT INTO "Estoque" (qtd_disponivel)
```

VALUES

```
(100),  
(2035),  
(300),  
(401),  
(50);
```

INSERT Categoria Produto

```
INSERT INTO "Categoria_Produto" (nome_categoria, desc_categoria)
```

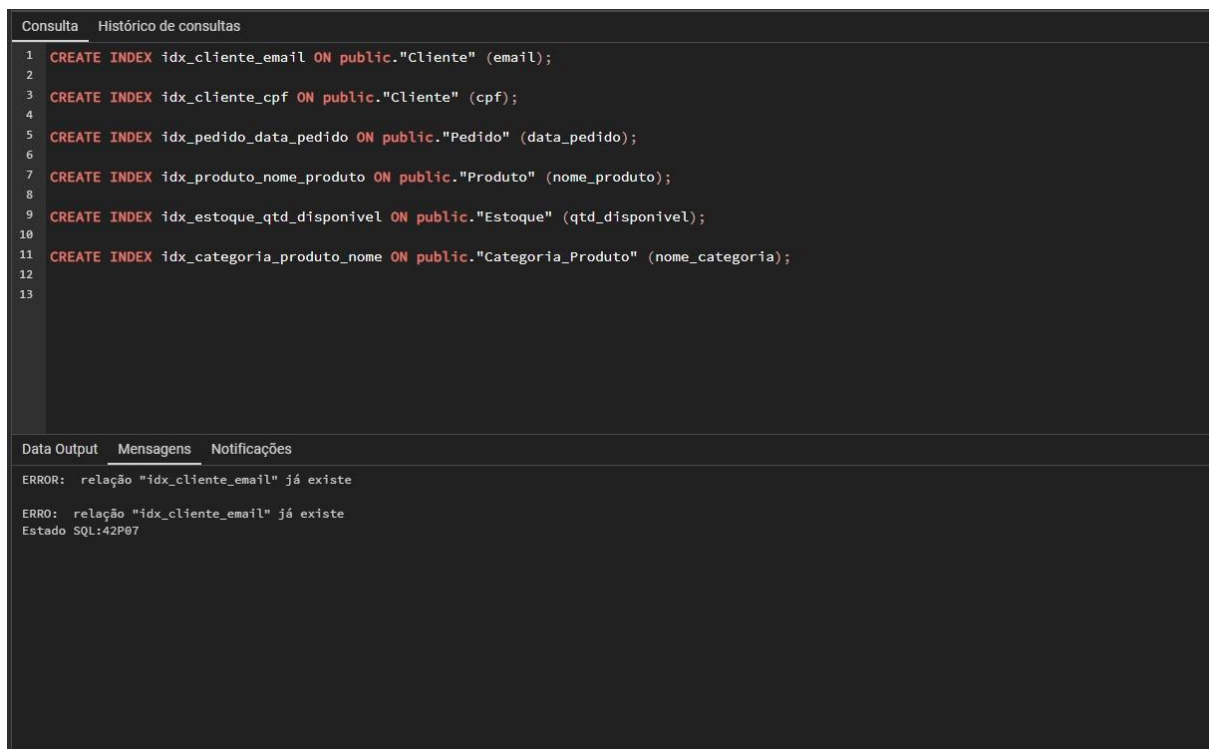
VALUES

```
('Cuidado Corporal', 'Hidratantes Corporal'),  
( 'Cabelo', 'Cuidados Cabelos Mistos e Oleosos'),  
( 'Maquiagem', 'Beleza'),  
( 'Cabelo', 'Cuidados Para todos os tipos de Cabelos'),  
( 'Cuidado Facial', 'Hidratante Facial');
```

INDEX

Os índices foram criados com o objetivo de ajudar a otimizar o tempo de consultas no banco de dados:

A imagem abaixo apresenta a criação dos Indexes:



The screenshot shows a SQL IDE interface with a dark theme. The top bar has tabs for 'Consulta' and 'Histórico de consultas'. The main editor area contains six SQL statements to create indexes, numbered 1 through 13. The bottom panel has tabs for 'Data Output', 'Mensagens', and 'Notificações'. The 'Mensagens' tab is active, displaying an error message: 'ERROR: relação "idx_cliente_email" já existe' and 'ERRO: relação "idx_cliente_email" já existe' with the state 'Estado SQL:42P07'.

```
1 CREATE INDEX idx_cliente_email ON public."Cliente" (email);
2
3 CREATE INDEX idx_cliente_cpf ON public."Cliente" (cpf);
4
5 CREATE INDEX idx_pedido_data_pedido ON public."Pedido" (data_pedido);
6
7 CREATE INDEX idx_produto_nome_produto ON public."Produto" (nome_produto);
8
9 CREATE INDEX idx_estoque_qtd_disponivel ON public."Estoque" (qtd_disponivel);
10
11 CREATE INDEX idx_categoria_produto_nome ON public."Categoria_Produto" (nome_categoria);
12
13
```

ERROR: relação "idx_cliente_email" já existe
ERRO: relação "idx_cliente_email" já existe
Estado SQL:42P07

```
CREATE INDEX idx_cliente_email ON public."Cliente" (email);
```

```
CREATE INDEX idx_cliente_cpf ON public."Cliente" (cpf);
```

```
CREATE INDEX idx_pedido_data_pedido ON public."Pedido" (data_pedido);
```

```
CREATE INDEX idx_produto_nome_produto ON public."Produto" (nome_produto);
```

```
CREATE INDEX idx_estoque_qtd_disponivel ON public."Estoque" (qtd_disponivel);
```

```
CREATE INDEX idx_categoria_produto_nome ON public."Categoria_Produto"  
(nome_categoria);
```

Consulta

Histórico de consultas

1

SELECT id_cliente, email, cpf, nome, rg, telefone, endereco

2

FROM public."Cliente";

Data Output

Mensagens

Notificações

	id_cliente integer	email [PK] character varying (100)	cpf [PK] character varying (20)	nome character varying (100)	rg character varying (20)	telefone character varying (100)	endereco character varying (255)
1	1	a.felis@outlook.org	11.111.111-1	("Hikaru Yamanaka")	000.000.000-01	(614) 776-1318	Ap #548-7547 Commodo Av...
2	2	duis.risus.odio@aol.couk	11.111.111-2	("Layanne Mary")	000.000.000-02	(546) 479-5384	Ap #673-6494 Ac Rd.
3	3	dados.carolyne@gmail.com	11.111.111-3	("Carolyne Oliveira")	000.000.000-03	1-473-553-0474	Ap #891-5968 Placerat Street
4	4	facilisi@protonmail.edu	11.111.111-4	("Deise Pestana")	000.000.000-04	1-153-313-8753	199-3157 Risus. St.
5	5	ipsum@outlook.edu	11.111.111-5	("Sacha Le")	000.000.000-05	(309) 850-6925	Ap #394-5985 Ut Street

JOINS

A imagem abaixo apresenta a junção das tabelas através de Joins:

ConsultaHistórico de consultas

1SELECT * FROM public."Pedido"

2INNER JOIN public."Cliente"

3ON public."Pedido".id_pedido = public."Cliente".id_cliente;

4

5SELECT * FROM public."Pedido"

6FULL JOIN public."Cliente"

7ON public."Pedido".id_pedido = public."Cliente".id_cliente;

8

9SELECT * FROM public."Pedido"

10RIGHT JOIN public."Cliente"

11ON public."Pedido".id_pedido = public."Cliente".id_cliente;

12

13SELECT * FROM public."Pedido"

14LEFT JOIN public."Cliente"

15ON public."Pedido".id_pedido = public."Cliente".id_cliente;

16

Data OutputMensagensNotificações

	id_pedido integer	cod_pedido integer	valor_total real	quantidade integer	data_pedido date	data_entrega date	status_pedido character varying (10)	forma_entrega character varying (20)	id_cliente integer	email character varying (100)	cpf cha
1	1	1	100	2	2024-01-01	2024-01-02	Enviado	Entrega em domicilio	1	a.felis@outlook.org	11.
2	2	2	200	1	2024-01-03	2024-01-04	Pendente	Retirada na loja	2	duis.risus.odio@aol.couk	11.
3	3	3	150	3	2024-01-05	2024-01-06	Enviado	Entrega em domicilio	3	dados.carolyne@gmail.com	11.
4	4	4	300	4	2024-01-07	2024-01-08	Pendente	Retirada na loja	4	facilisi@protonmail.edu	11.
5	5	5	250	5	2024-01-09	2024-01-10	Enviado	Entrega em domicilio	5	ipsum@outlook.edu	11.

```
SELECT * FROM public."Pedido"
INNER JOIN public."Cliente"
ON public."Pedido".id_pedido = public."Cliente".id_cliente;
```

```
SELECT * FROM public."Pedido"
FULL JOIN public."Cliente"
ON public."Pedido".id_pedido = public."Cliente".id_cliente;
```

```
SELECT * FROM public."Pedido"
RIGHT JOIN public."Cliente"
ON public."Pedido".id_pedido = public."Cliente".id_cliente;
```

```
SELECT * FROM public."Pedido"
LEFT JOIN public."Cliente"
ON public."Pedido".id_pedido = public."Cliente".id_cliente;
```

Normalização:

As tabelas foram normalizadas até a terceira forma normal (3NF), garantindo a eliminação de dependências transitivas e mantendo a integridade dos dados.

Documentação:

Esta documentação inclui a descrição detalhada de cada entidade, relacionamento, atributo, o diagrama ER, o script SQL para criação das tabelas, índices e chaves.

Regras de Negócio Relacionadas aos Dados:

- O endereço do cliente deve contemplar os seguintes dados: nome da rua, bairro, cidade, estado e CEP.
- O pedido deve conter as seguintes formas de entrega: retirada e frete.
- O pedido também deve contar as seguintes formas de pagamento: cartão de crédito, cartão de débito, pix e boleto.

Glossário de Termos:

Diagrama ER: Diagrama de Entidade-Relacionamento.

Responsabilidades:

Equipe de Desenvolvimento: Marcel Hikaru Yamanaka, Layanne Mary e Carlyne Oliveira.

Analistas de Negócios: Marcel Hikaru Yamanaka, Layanne Mary e Carlyne Oliveira.