

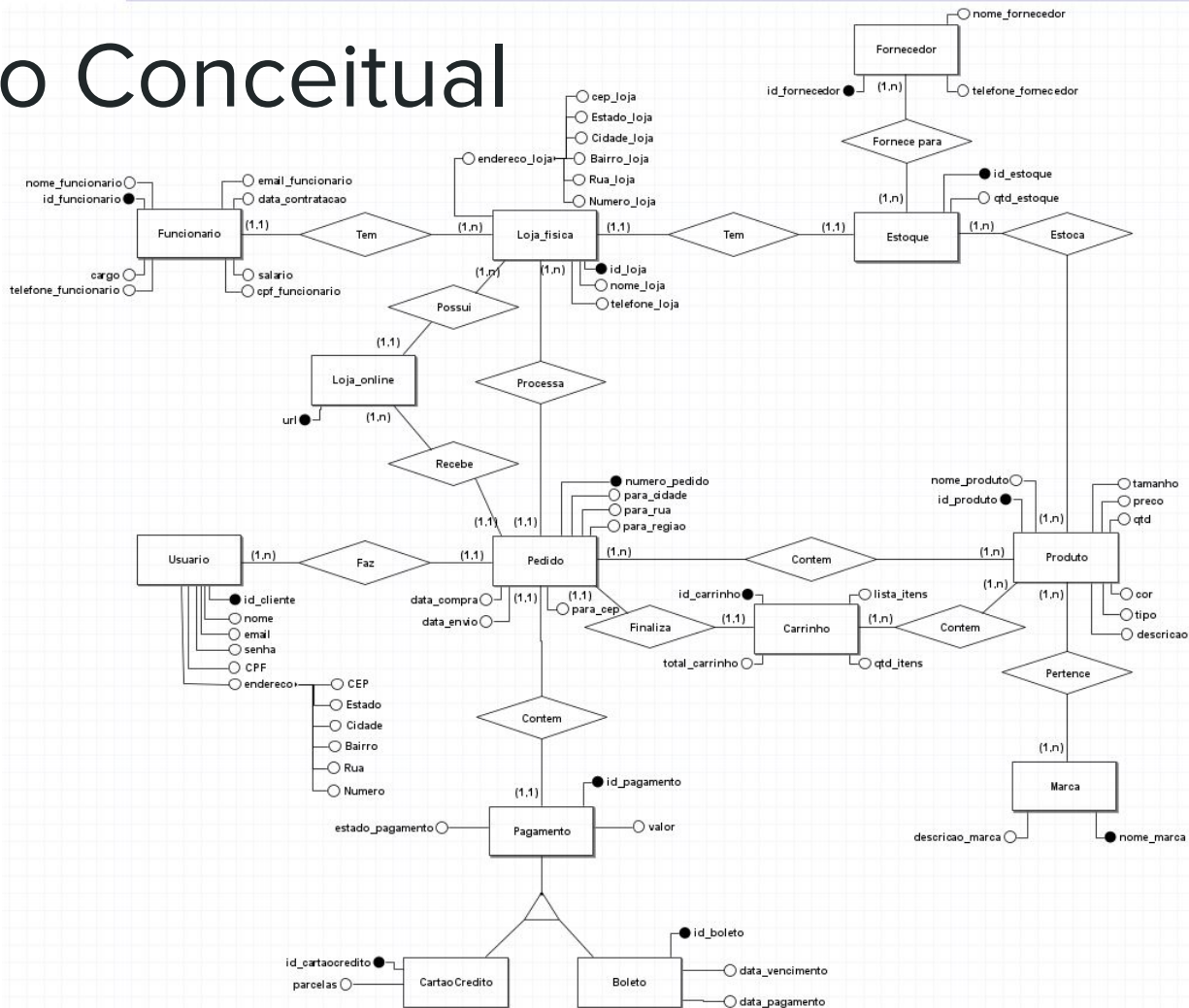
Projeto de Banco de dados

Por: Arthur Ramos, Henrique Martins, João Pedro, Vitor Oliveira, Victor Hugo

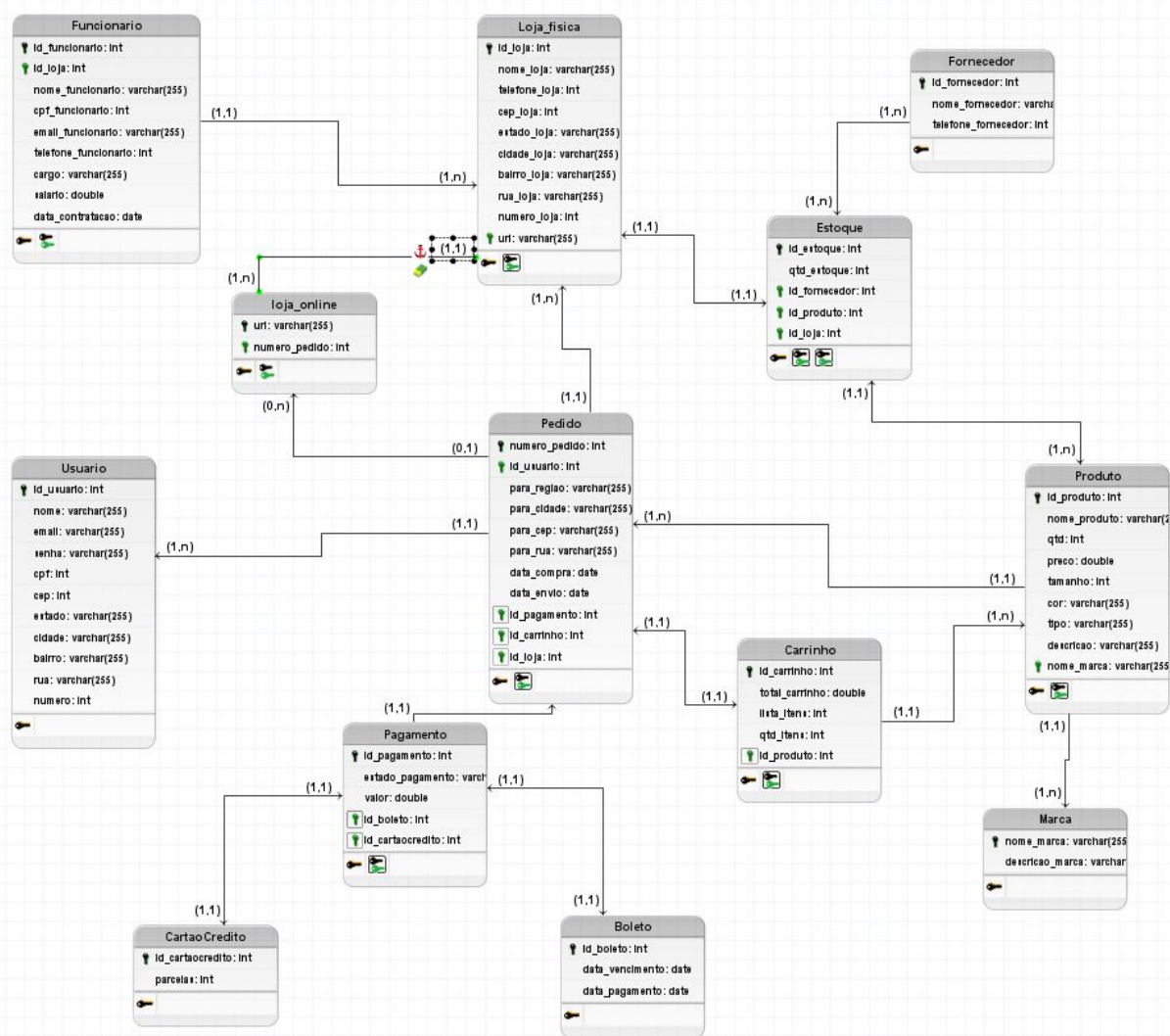
Disciplina: Banco de Dados II

Professor: Floriano Ferreira dos Reis Filho

Modelo Conceitual



Modelo Lógico



Modelo Físico

```
CREATE TABLE Usuario (  
    id_usuario INT PRIMARY KEY,  
    nome VARCHAR(255),  
    email VARCHAR(255),  
    senha VARCHAR(255),  
    cpf INT,  
    cep INT,  
    estado VARCHAR(255),  
    cidade VARCHAR(255),  
    bairro VARCHAR(255),  
    rua VARCHAR(255),  
    numero INT  
);  
  
CREATE TABLE Marca (  
    nome_marca VARCHAR(255) PRIMARY KEY,  
    descricao_marca VARCHAR(255)  
);  
  
CREATE TABLE Boleto (  
    id_boleto INT PRIMARY KEY,  
    data_vencimento DATE,  
    data_pagamento DATE  
);  
  
CREATE TABLE CartaoCredito (  
    id_cartaocredito INT PRIMARY KEY,  
    parcelas INT  
);
```

```
CREATE TABLE Pagamento (  
    id_pagamento INT PRIMARY KEY,  
    estado_pagamento VARCHAR(255),  
    valor DOUBLE,  
    id_boleto INT,  
    id_cartaocredito INT,  
    FOREIGN KEY (id_boleto) REFERENCES Boleto (id_boleto),  
    FOREIGN KEY (id_cartaocredito) REFERENCES CartaoCredito  
(id_cartaocredito)  
);  
  
CREATE TABLE Produto (  
    id_produto INT PRIMARY KEY,  
    nome_produto VARCHAR(255),  
    qtd INT,  
    preco DOUBLE,  
    tamanho INT,  
    cor VARCHAR(255),  
    tipo VARCHAR(255),  
    descricao VARCHAR(255),  
    nome_marca VARCHAR(255),  
    FOREIGN KEY (nome_marca) REFERENCES Marca (nome_marca)  
);  
  
CREATE TABLE Carrinho (  
    id_carrinho INT PRIMARY KEY,  
    total_carrinho DOUBLE,  
    lista_itens INT,  
    qtd_itens INT,  
    id_produto INT,  
    FOREIGN KEY (id_produto) REFERENCES Produto (id_produto)  
);
```

```

CREATE TABLE Pedido (
    numero_pedido INT PRIMARY KEY,
    id_usuario INT,
    para_regiao VARCHAR(255),
    para_cidade VARCHAR(255),
    para_cep VARCHAR(10),
    para_rua VARCHAR(255),
    data_compra DATE,
    data_envio DATE,
    id_pagamento INT,
    id_carrinho INT,
    id_loja INT,
    FOREIGN KEY (id_usuario) REFERENCES Usuario (id_usuario),
    FOREIGN KEY (id_pagamento) REFERENCES Pagamento
(id_pagamento),
    FOREIGN KEY (id_carrinho) REFERENCES Carrinho (id_carrinho),
    FOREIGN KEY (id_loja) REFERENCES Loja_fisica (id_loja)
);
CREATE TABLE loja_online (
    url VARCHAR(255) PRIMARY KEY,
    numero_pedido INT,
    FOREIGN KEY (numero_pedido) REFERENCES Pedido
(numero_pedido)
);
CREATE TABLE Loja_fisica (
    id_loja INT PRIMARY KEY,
    nome_loja VARCHAR(255),
    telefone_loja INT,
    cep_loja INT,
    estado_loja VARCHAR(255),
    cidade_loja VARCHAR(255),
    bairro_loja VARCHAR(255),
    rua_loja VARCHAR(255),
    numero_loja INT,
    url VARCHAR(255)
);

```

```

CREATE TABLE Funcionario (
    id_funcionario INT PRIMARY KEY,
    id_loja INT,
    nome_funcionario VARCHAR(255),
    cpf_funcionario INT,
    email_funcionario VARCHAR(255),
    telefone_funcionario INT,
    cargo VARCHAR(255),
    salario DOUBLE,
    data_contratacao DATE,
    FOREIGN KEY (id_loja) REFERENCES Loja_fisica (id_loja)
);
CREATE TABLE Fornecedor (
    id_fornecedor INT PRIMARY KEY,
    nome_fornecedor VARCHAR(255),
    telefone_fornecedor INT
);
CREATE TABLE Estoque (
    id_estoque INT PRIMARY KEY,
    qtd_estoque INT,
    id_fornecedor INT,
    id_produto INT,
    id_loja INT,
    FOREIGN KEY (id_fornecedor) REFERENCES Fornecedor (id_fornecedor),
    FOREIGN KEY (id_produto) REFERENCES Produto (id_produto),
    FOREIGN KEY (id_loja) REFERENCES Loja_fisica (id_loja)
);
CREATE TABLE Log (
    id_log INT PRIMARY KEY AUTO_INCREMENT,
    descricao VARCHAR(255),
    data_registro DATETIME
);

```

Triggers

- 1 - Após um novo usuário ser inserido na tabela Usuário
- 2 - Antes de um produto ser excluído da tabela Produto
- 3 - Após um novo pagamento ser atualizado na tabela Pagamento
- 4 - Após um novo pedido ser inserido na tabela Pedido
- 5 - Antes de uma linha ser excluída da tabela Carrinho

1 - Após um novo usuário ser inserido na tabela Usuário:

```
CREATE TRIGGER novo_usuario  
AFTER INSERT ON Usuario  
FOR EACH ROW  
BEGIN
```

```
INSERT INTO Log (descricao) VALUES (CONCAT('Novo usuário com ID ',  
NEW.id_usuario, ' foi criado.));  
END;
```

```
INSERT INTO Usuario (id_usuario, nome, email, senha, cpf, cep, estado, cidade, bairro, rua, numero)  
VALUES ('1', 'Henrique', 'henrique@gmail.com', '123', '1', 'SP', 'Santo Andre', 'Utinga', 'ruinha', '123');
```

	id_usuario	nome	email	senha	cpf	cep	estado	cidade	bairro	rua	numero
►	1	Henrique	henrique@gmail.com	123	1	NULL	SP	Santo Andre	Utinga	ruinha	123

	id_log	descricao	data_registro
►	1	Novo usuário com ID 1 foi criado.	NULL

2 - Antes de um produto ser excluído da tabela Produto:

```
CREATE TRIGGER deletar_produto
BEFORE DELETE ON Produto
FOR EACH ROW
BEGIN
INSERT INTO Log (descricao) VALUES (CONCAT('Produto com ID ', OLD.id_produto, ' foi
excluído.));
END;
```

	id_produto	nome_produto	qtd	preco	tamanho	cor	tipo	descricao	nome_marca
▶	1	tenis 1	5	100	40	Preto	tenis cano alto	tenis cano alto da marca nike	Nike

320 • `DELETE FROM Produto WHERE id_produto = 1;`

Result Grid									
Filter Rows: Edit: Export/Import:									
	id_produto	nome_produto	qtd	preco	tamanho	cor	tipo	descricao	nome_marca
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	id_log	descricao	data_registro
▶	1	Novo usuário com ID 1 foi criado.	NULL
	2	Produto com ID 1 foi excluído.	NULL

3 - Após um novo pagamento ser atualizado na tabela Pagamento:

```
CREATE TRIGGER atualizar_pagamento  
AFTER UPDATE ON Pagamento  
FOR EACH ROW  
BEGIN
```

```
INSERT INTO Log (descricao) VALUES (CONCAT('Pagamento com ID ',  
NEW.id_pagamento, ' foi atualizado.'));
```

```
END;
```

	id_pagamento	estado_pagamento	valor	id_boleto	id_cartaocredito
▶	1	pendente	100	NULL	1

```
331 • UPDATE Pagamento  
332 SET estado_pagamento = 'pago'  
333 WHERE id_pagamento = 1;
```

Result Grid					
Filter Rows:					
	id_pagamento	estado_pagamento	valor	id_boleto	id_cartaocredito
▶	1	pago	100	NULL	1
	id_log	descricao	data_registro		
▶	1	Novo usuário com ID 1 foi criado.	NULL		
	2	Produto com ID 1 foi excluído.	NULL		
	3	Pagamento com ID 1 foi atualizado.	NULL		

4 - Após um novo pedido ser inserido na tabela Pedido:

```
CREATE TRIGGER novo_pedido
AFTER INSERT ON Pedido
FOR EACH ROW
BEGIN
INSERT INTO Log (descricao) VALUES (CONCAT('Novo pedido com número ',
NEW.numero_pedido, ' foi criado.));
END;
```

```
336 • INSERT INTO Pedido (numero_pedido, id_usuario, para_regiao, para_cidade, para_cep, para_rua, data_compra, data_envio, id_pagamento, id_carrinho, id_loja
337 VALUES ('2', '1', 'SP', 'Santo Andre', '123', 'ruinha', '2023-11-10', '2023-11-10', '1', '1', '1');
```

Result Grid											
Filter Rows:											
Edit:											
Export/Import:											
Wrap Cell Content:											
	numero_pedido	id_usuario	para_regiao	para_cidade	para_cep	para_rua	data_compra	data_envio	id_pagamento	id_carrinho	id_loja
▶	2	1	SP	Santo Andre	123	ruinha	2023-11-10	2023-11-10	1	1	1
			id_log					descricao			
▶			1					Novo usuário com ID 1 foi criado.			
			2					Produto com ID 1 foi excluído.			
			3					Pagamento com ID 1 foi atualizado.			
			4					Novo pedido com número 2 foi criado.			

5 - Antes de uma linha ser excluída da tabela Carrinho:

```
CREATE TRIGGER deletar_carrinho
BEFORE DELETE ON Carrinho
FOR EACH ROW
BEGIN
INSERT INTO Log (descricao) VALUES (CONCAT('Carrinho com ID ', OLD.id_carrinho, ' foi
excluído.));
END;
```

	id_carrinho	total_carrinho	lista_itens	qtd_itens	id_produto
▶	1	100	Tenis nike	1	1

```
343 • DELETE FROM Carrinho WHERE id_carrinho = 1;
```

Result Grid   Filter Rows: Edit:  

	id_carrinho	total_carrinho	lista_itens	qtd_itens	id_produto
*	NULL	NULL	NULL	NULL	NULL

	id_log	descricao	data_registro
▶	1	Novo usuário com ID 1 foi criado.	NULL
	2	Produto com ID 1 foi excluído.	NULL
	3	Pagamento com ID 1 foi atualizado.	NULL
	4	Novo pedido com número 2 foi criado.	NULL
	5	Carrinho com ID 1 foi excluído.	NULL







Procedures

- 1 - Procedimento para inserir um novo usuário na tabela Usuário
- 2 - Procedimento para inserir um novo pedido na tabela Pedido
- 3 - Procedimento para deletar um produto da tabela Produto
- 4 - Procedimento para atualizar o estado de um pagamento na tabela Pagamento
- 5 - Procedimento para deletar um carrinho da tabela Carrinho

1 - Procedimento para inserir um novo usuário na tabela Usuário:

```
CREATE PROCEDURE InserirUsuario(IN id INT, IN nome VARCHAR(255), IN email VARCHAR(255),  
    IN senha VARCHAR(255), IN cpf INT, IN cep INT, IN estado VARCHAR(255), IN cidade  
    VARCHAR(255), IN bairro VARCHAR(255), IN rua VARCHAR(255), IN numero INT)  
    BEGIN  
    INSERT INTO Usuario (id_usuario, nome, email, senha, cpf, cep, estado, cidade, bairro, rua,  
        numero) VALUES (id, nome, email, senha, cpf, cep, estado, cidade, bairro, rua, numero);  
    END;
```

346 • `CALL InserirUsuario('2', 'Joao' , 'joao@gmail.com', '123456', '2' , '09110330', 'SP', 'Santo Andre', 'Camilopolis', 'rua', '10');`

Result Grid											
Filter Rows: <input type="text"/>											
Edit:   											
Export/Import:  											
Wrap Cell Content: 											
	id_usuario	nome	email	senha	cpf	cep	estado	cidade	bairro	rua	numero
▶	1	Henrique	henrique@gmail.com	123	1	NULL	SP	Santo Andre	Utinga	ruinha	123
	2	Joao	joao@gmail.com	123456	2	9110330	SP	Santo Andre	Camilopolis	rua	10

2 - Procedimento para inserir um novo pedido na tabela Pedido:

```
CREATE PROCEDURE InserirPedido(IN numero INT, IN id_usuario INT, IN regiao VARCHAR(255), IN cidade  
VARCHAR(255), IN cep VARCHAR(10), IN rua VARCHAR(255), IN data_compra DATE, IN data_envio DATE, IN  
id_pagamento INT, IN id_carrinho INT, IN id_loja INT)  
BEGIN  
    INSERT INTO Pedido (numero_pedido, id_usuario, para_regiao, para_cidade, para_cep, para_rua,  
data_compra, data_envio, id_pagamento, id_carrinho, id_loja) VALUES (numero, id_usuario, regiao, cidade,  
cep, rua, data_compra, data_envio, id_pagamento, id_carrinho, id_loja);  
END;
```

```
350 • INSERT INTO Pagamento (id_pagamento, estado_pagamento, valor, id_cartaocredito)  
351 VALUES ('2', 'pendente', '300', '2');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	id_pagamento	estado_pagamento	valor	id_boleto	id_cartaocredito
▶	1	pago	100	NULL	1
	2	pendente	300	NULL	2

```
353 • CALL AtualizarEstadoPagamento(2, 'pago');
```

Result Grid | Filter Rows: | Edit: | Exp

	id_pagamento	estado_pagamento	valor	id_boleto	id_cartaocredito
▶	1	pago	100	NULL	1
	2	pago	300	NULL	2
*	NULL	NULL	NULL	NULL	NULL

3 - Procedimento para deletar um produto da tabela Produto

```
CREATE PROCEDURE DeletarProduto(IN id INT)
BEGIN
DELETE FROM Produto WHERE id_produto = id;
END;
```

```
357 • INSERT INTO Produto (id_produto, nome_produto, qtd, preco, tamanho, cor, tipo, descricao, nome_marca)
358 VALUES ('2', 'chinelo 1', '6', '50', '41', 'Branco', 'chinelo de dedo', 'chinelo Havaianas', 'Havaianas');
```

[illegible]







```
360 • CALL DeletarProduto(2);
```

<div> <div>Result Grid</div> <div> <div>Filter Rows:</div> <div>Edit:</div> <div>Export/Import:</div> <div>Wrap Cell Content:</div> </div> </div>										
	id_produto	nome_produto	qtd	preco	tamanho	cor	tipo	descricao	nome	
▶	1	tenis 1	5	100	40	Preto	tenis cano alto	tenis cano alto da marca nike	Nike	

4 - Procedimento para atualizar o estado de um pagamento na tabela Pagamento:

```
CREATE PROCEDURE AtualizarEstadoPagamento(IN id INT, IN estado VARCHAR(255))
BEGIN
    UPDATE Pagamento SET estado_pagamento = estado WHERE id_pagamento = id;
END;
```







363 • `CALL InserirPedido('3', '1', 'SP', 'Santo Andre', '123', 'ruinha', '2023-11-10', '2023-11-10', '1', '1', '1');`

Result Grid  Filter Rows: <input type="text"/> Edit:    Export/Import:   Wrap Cell Content: 											
	numero_pedido	id_usuario	para_regiao	para_cidade	para_cep	para_rua	data_compra	data_envio	id_pagamento	id_carrinho	id_loja
▶	2	1	SP	Santo Andre	123	ruinha	2023-11-10	2023-11-10	1	1	1
	3	1	SP	Santo Andre	123	ruinha	2023-11-10	2023-11-10	1	1	1

5 - Procedimento para deletar um carrinho da tabela Carrinho:

```
CREATE PROCEDURE DeletarCarrinho(IN  
id INT)  
BEGIN  
    DELETE FROM Carrinho WHERE  
id_carrinho = id;  
END;
```

```
340 • INSERT INTO Carrinho (id_carrinho, total_carrinho, lista_itens, qtd_itens, id_produto)  
341 VALUES ('1', '100', 'Tenis nike', '1', '1');
```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
Export/Import:  					
Wrap Cell Content: 					
	id_carrinho	total_carrinho	lista_itens	qtd_itens	id_produto
▶	1	100	Tenis nike	1	1

```
366 • CALL DeletarCarrinho(1);
```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
	id_carrinho	total_carrinho	lista_itens	qtd_itens	id_produto
*	NULL	NULL	NULL	NULL	NULL

Cursors

- 1 - Cursor para percorrer todos os usuários e exibir seus nomes e e-mails:
- 2 - Cursor para calcular a média de salários de funcionários
- 3 - Cursor para exibir produtos com estoque baixo
- 4 - Cursor para exibir detalhes dos produtos no estoque
- 5 - Cursor para listar os produtos em promoção

1- Cursor para percorrer todos os usuários e exibir seus nomes e e-mails:

```
DECLARE usuario_cursor CURSOR FOR  
SELECT nome, email FROM Usuario;
```

```
OPEN usuario_cursor;  
FETCH usuario_cursor INTO @nome_usuario,  
@email_usuario;
```

```
WHILE @@FETCH_STATUS = 0  
BEGIN  
    -- Faça algo com os dados, como imprimir na  
    console  
    PRINT 'Nome: ' + @nome_usuario + ', Email: '  
    + @email_usuario;  
    FETCH usuario_cursor INTO  
    @nome_usuario, @email_usuario;  
END
```

```
CLOSE usuario_cursor;  
DEALLOCATE usuario_cursor;
```

2- Cursor para calcular a média de salários de funcionários:

```
DECLARE funcionario_cursor CURSOR FOR  
SELECT salario FROM Funcionario;
```

```
OPEN funcionario_cursor;  
FETCH funcionario_cursor INTO @salario_funcionario;
```

```
WHILE @@FETCH_STATUS = 0  
BEGIN  
    -- Adiciona o salário à soma  
    SET @soma_salarios = @soma_salarios +  
    @salario_funcionario;  
    SET @contador = @contador + 1;  
    FETCH funcionario_cursor INTO  
    @salario_funcionario;  
END
```

```
SET @media_salarios = @soma_salarios / @contador;
```

```
CLOSE funcionario_cursor;  
DEALLOCATE funcionario_cursor;
```

3 - Cursor para exibir produtos com estoque baixo:

```
DECLARE estoque_cursor CURSOR FOR  
SELECT p.nome_produto, e.qtd_estoque  
FROM Produto p  
JOIN Estoque e ON p.id_produto = e.id_produto  
WHERE e.qtd_estoque < 10;
```

```
OPEN estoque_cursor;  
FETCH estoque_cursor INTO @nome_produto,  
@qtd_estoque;
```

```
WHILE @@FETCH_STATUS = 0  
BEGIN
```

```
    PRINT 'Produto: ' + @nome_produto + ',  
    Quantidade em Estoque: ' + CAST(@qtd_estoque AS  
    VARCHAR(10));
```

```
    FETCH estoque_cursor INTO @nome_produto,  
    @qtd_estoque;  
END
```

```
CLOSE estoque_cursor;  
DEALLOCATE estoque_cursor;
```

4 - Cursor para exibir detalhes dos produtos no estoque:

```
DECLARE estoque_cursor CURSOR FOR  
SELECT p.nome_produto, e.qtd_estoque,  
f.nome_fornecedor  
FROM Produto p, Estoque e, Fornecedor f  
WHERE p.id_produto = e.id_produto AND  
e.id_fornecedor = f.id_fornecedor;
```

```
OPEN estoque_cursor;  
FETCH estoque_cursor INTO @nome_produto,  
@qtd_estoque, @nome_fornecedor;
```

```
WHILE @@FETCH_STATUS = 0  
BEGIN
```

```
    PRINT 'Produto: ' + @nome_produto + ', Estoque:  
' + CAST(@qtd_estoque AS VARCHAR(10)) +  
    ', Fornecedor: ' + @nome_fornecedor;
```

```
    FETCH estoque_cursor INTO @nome_produto,  
    @qtd_estoque, @nome_fornecedor;  
END
```

```
CLOSE estoque_cursor;  
DEALLOCATE estoque_cursor;
```

5 - Cursor para listar os produtos em promoção:

```
DECLARE promocao_cursor CURSOR FOR  
SELECT nome_produto, preco  
FROM Produto  
WHERE preco < 20.0;
```

```
OPEN promocao_cursor;  
FETCH promocao_cursor INTO  
@nome_produto_promo, @preco_promo;
```

```
WHILE @@FETCH_STATUS = 0  
BEGIN
```

```
    PRINT 'Produto em Promoção: ' +  
    @nome_produto_promo + ', Preço: ' +  
    CAST(@preco_promo AS VARCHAR(10));  
    FETCH promocao_cursor INTO  
    @nome_produto_promo, @preco_promo;  
END
```

```
CLOSE promocao_cursor;  
DEALLOCATE promocao_cursor;
```

Obrigado