Missão Prática | Nível 2 | Mundo 3

Lucas Figueirêdo Costa de Queiroz

Campus Via Corpos

Curso Desenvolvimento Full-Stack

Disciplina: Mundo 3 - Nível 2 Vamos Manter as Informações!

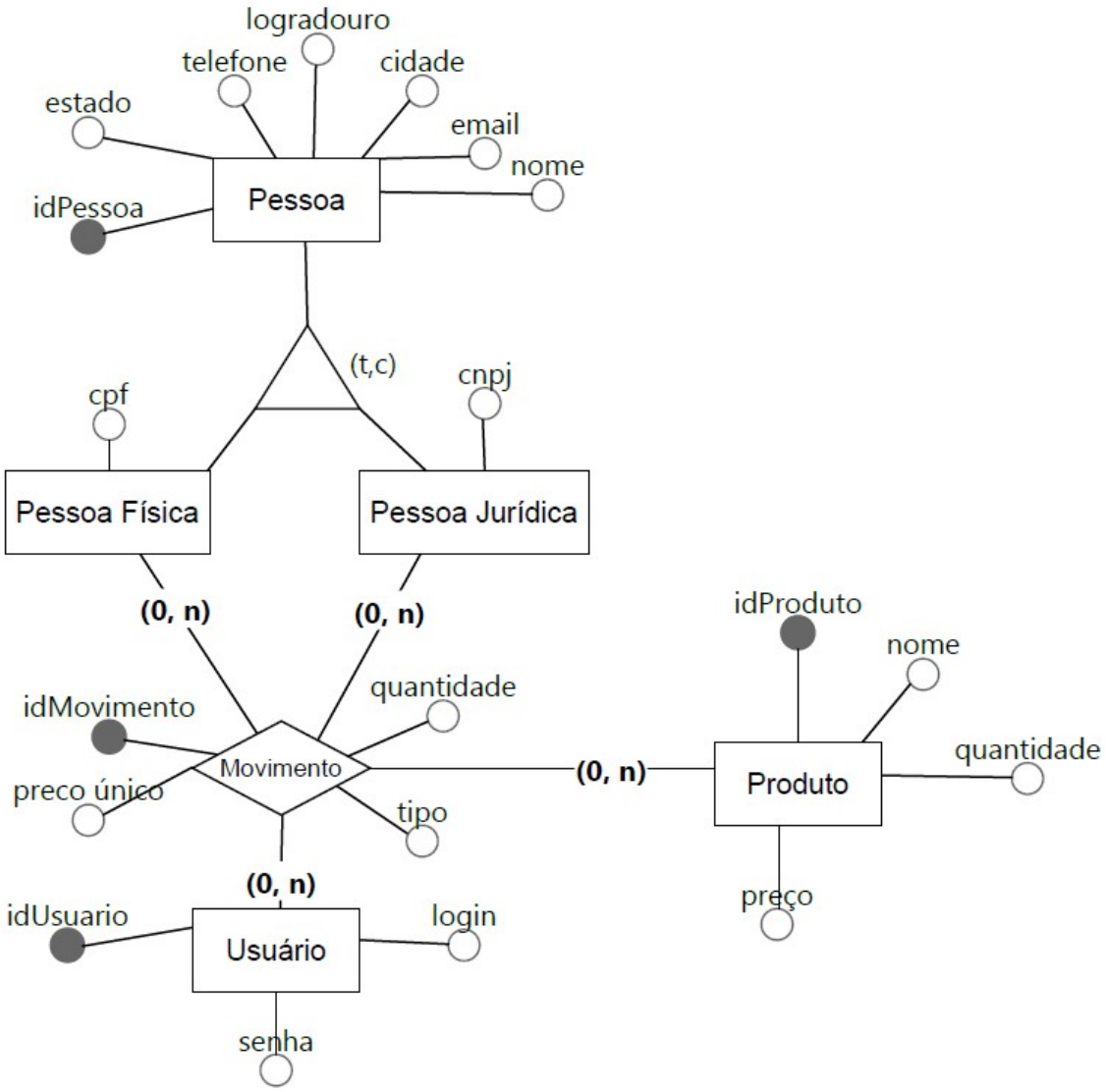
Repositório: https://github.com/Lucas8956/Mundo-3-MP2

Objetivo da Prática

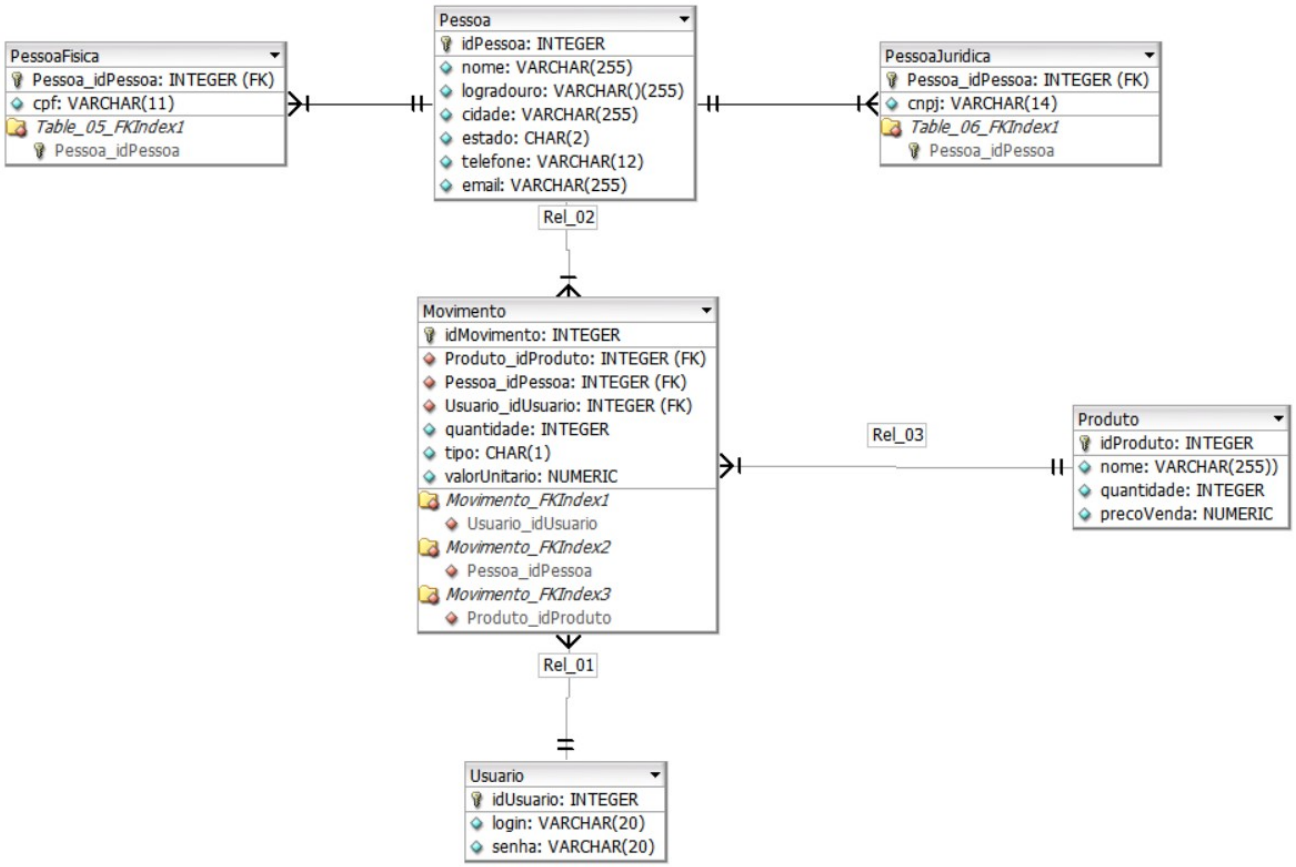
Modelar e implementar um banco de dados simples no SQL Server, inserir dados e fazer diversas tipos de consultas através da linguagem SQL seguindo os passos:

1) Identificar os requisites de um Sistema e gerar um modelo conceitual.
2) Utilizar DBDesigner para criar um modelo lógico.
3) Utilizar as ferramentas do SQL Server Management Studio para criar um novo banco e logon que irá criar as estruturas do banco.
4) Usar a linguagem SQL para criar as estruturas do banco.
5) Usar a linguagem SQL para inserir dados e fazer consultas no banco.

Modelo 1


logradouro
telefone
cidade
estado
email
nome
idPessoa
Pessoa
(t,c)
cnpj
cpf
Pessoa Física
Pessoa Jurídica
(0, n)
(0, n)
idProduto
nome
idMovimento
quantidade
Movimento
(0, n)
Produto
quantidade
preco único
tipo
(0, n)
preço
idUsuario
login
Usuário
senha

Modelo 2



1º Procedimento

Criando o Banco de Dados

```
CREATE DATABASE [Loja]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'Loja', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Loja.mdf' , SIZE = 8192KB , FILEGROWTH =
65536KB )
 LOG ON
( NAME = N'Loja_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\Loja_log.ldf' , SIZE = 8192KB , FILEGROWTH
= 65536KB )
 WITH LEDGER = OFF
GO
ALTER DATABASE [Loja] SET COMPATIBILITY_LEVEL = 160
GO
ALTER DATABASE [Loja] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Loja] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Loja] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Loja] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Loja] SET ARITHABORT OFF
GO
```

```
ALTER DATABASE [Loja] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Loja] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [Loja] SET AUTO_CREATE_STATISTICS ON(INCREMENTAL = OFF)
GO
ALTER DATABASE [Loja] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Loja] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Loja] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [Loja] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Loja] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Loja] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Loja] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Loja] SET  DISABLE_BROKER
GO
ALTER DATABASE [Loja] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [Loja] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [Loja] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [Loja] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [Loja] SET  READ_WRITE
GO
ALTER DATABASE [Loja] SET RECOVERY FULL
GO
ALTER DATABASE [Loja] SET  MULTI_USER
GO
ALTER DATABASE [Loja] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [Loja] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [Loja] SET DELAYED_DURABILITY = DISABLED
GO
USE [Loja]
GO
ALTER DATABASE SCOPED CONFIGURATION SET LEGACY_CARDINALITY_ESTIMATION = Off;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET
LEGACY_CARDINALITY_ESTIMATION = Primary;
GO
ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 0;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET MAXDOP = PRIMARY;
GO
ALTER DATABASE SCOPED CONFIGURATION SET PARAMETER_SNIFFING = On;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET PARAMETER_SNIFFING =
Primary;
GO
ALTER DATABASE SCOPED CONFIGURATION SET QUERY_OPTIMIZER_HOTFIXES = Off;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET QUERY_OPTIMIZER_HOTFIXES =
Primary;
GO
USE [Loja]
```

```
GO
IF NOT EXISTS (SELECT name FROM sys.filegroups WHERE is_default=1 AND name =
N'PRIMARY') ALTER DATABASE [Loja] MODIFY FILEGROUP [PRIMARY] DEFAULT
GO
```

Criando novo logone usuário

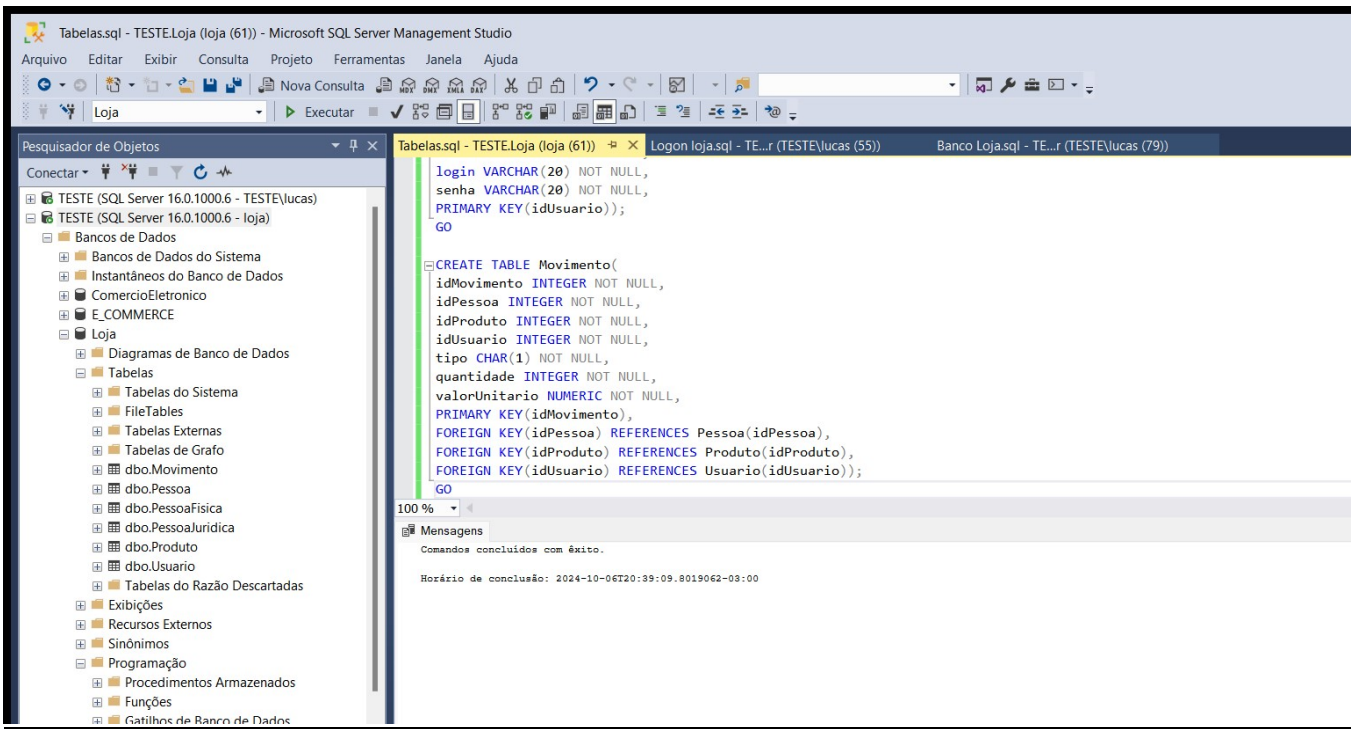
```
USE [master]
GO
CREATE LOGIN [loja] WITH PASSWORD=N'loja', DEFAULT_DATABASE=[Loja],
DEFAULT_LANGUAGE=[Português (Brasil)], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE [Loja]
GO
CREATE USER [loja] FOR LOGIN [loja]
GO
USE [Loja]
GO
ALTER ROLE [db_owner] ADD MEMBER [loja]
GO
```

Criando estruturas do banco

```
USE Loja;
GO

CREATE SEQUENCE sequenciaIdPessoa
START WITH 1
INCREMENT BY 1;
GO

CREATE TABLE Pessoa (
idPessoa INTEGER NOT NULL,
nome VARCHAR(255) NOT NULL,
logradouro VARCHAR(255) NOT NULL,
cidade VARCHAR(255) NOT NULL,
estado CHAR(2) NOT NULL,
telefone VARCHAR(12) NOT NULL,
email VARCHAR(255) NOT NULL,
PRIMARY KEY(idPessoa));
GO

CREATE TABLE PessoaJuridica (
idPessoa INTEGER NOT NULL,
cnpj VARCHAR(14) NOT NULL,
PRIMARY KEY(idPessoa),
FOREIGN KEY(idPessoa) REFERENCES Pessoa(idPessoa));
GO

CREATE TABLE PessoaFisica (
idPessoa INTEGER NOT NULL,
cpf VARCHAR(11) NOT NULL,
PRIMARY KEY(idPessoa),
FOREIGN KEY(idPessoa) REFERENCES Pessoa(idPessoa));
GO

CREATE TABLE Produto (
idProduto INTEGER NOT NULL,
nome VARCHAR(255) NOT NULL,
quantidade INTEGER NOT NULL,
precoVenda NUMERIC NOT NULL,
PRIMARY KEY(idProduto));
```

```
GO

CREATE TABLE Usuario(
idUsuario INTEGER NOT NULL,
login VARCHAR(20) NOT NULL,
senha VARCHAR(20) NOT NULL,
PRIMARY KEY(idUsuario));
GO

CREATE TABLE Movimento(
idMovimento INTEGER NOT NULL,
idPessoa INTEGER NOT NULL,
idProduto INTEGER NOT NULL,
idUsuario INTEGER NOT NULL,
tipo CHAR(1) NOT NULL CHECK (tipo IN ('E', 'S')),
quantidade INTEGER NOT NULL,
valorUnitario NUMERIC NOT NULL,
PRIMARY KEY(idMovimento),
FOREIGN KEY(idPessoa) REFERENCES Pessoa(idPessoa),
FOREIGN KEY(idProduto) REFERENCES Produto(idProduto),
FOREIGN KEY(idUsuario) REFERENCES Usuario(idUsuario));
GO
```

Resultados



Conclusão:

a) Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

R) 1x1: uma das tabelas terá uma chave estrangeira referente a outra tabela.

1xn: a tabela do lado n recebe chave estrangeira da outra.

NxN: é criada uma nova tabela que recebe chave estrangeira das outras duas tabelas.

b) Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

R) Cria uma tabela para a entidade genérica com seus atributos e outra para cada entidade especializada com os atributos exclusivos com chave estrangeira para a tabela genérica.

c) Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

R) A interface gráfica facilita e agiliza algumas tarefas em coporação com o script, pode-se usar o script gerado por ele para reduzir parte do Código que será escrito, identificação de erros,

2º Procedimento

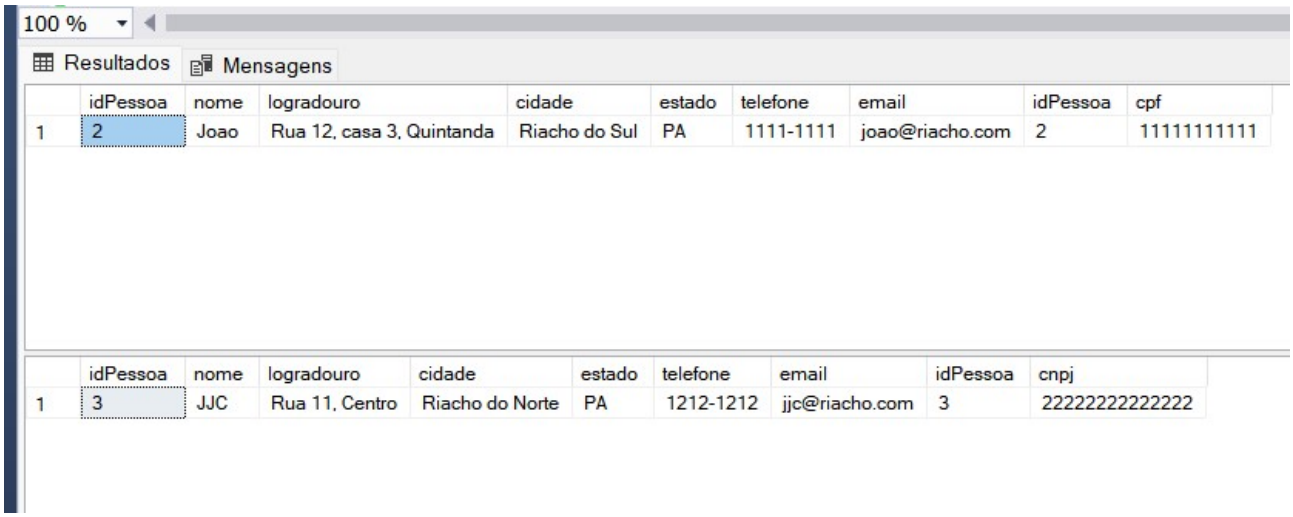
```
-- Conjunto Usuários
INSERT INTO Usuario (idUsuario, login, senha)
VALUES
(1, 'op1', 'op1'),
(2, 'op2', 'op2');
GO

SELECT * FROM Usuario;

GO
```

	idUsuario	login	senha
1	1	op1	op1
2	2	op2	op2

```
-- Conjunto Produtos
INSERT INTO Produto (idProduto, nome, quantidade, precoVenda)
VALUES
(1, 'Banana', 100, 5.00),
(3, 'Laranja', 500, 2.00),
(4, 'Manga', 800, 4.00);
GO

SELECT * FROM Produto;
GO
```

	idProduto	nome	quantidade	precoVenda
1	1	Banana	100	5
2	3	Laranja	500	2
3	4	Manga	800	4

```
-- Conjunto Pessoas
DECLARE @ID AS INTEGER = NEXT VALUE FOR sequenciaIdPessoa
INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)
VALUES (@ID, 'Joao', 'Rua 12, casa 3, Quintanda', 'Riacho do Sul', 'PA', '1111-
1111', 'joao@riacho.com');
INSERT INTO PessoaFisica (idPessoa, cpf)
VALUES (@ID, '11111111111');
GO

DECLARE @ID AS INTEGER = NEXT VALUE FOR sequenciaIdPessoa
INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email)
VALUES (@ID, 'JJC', 'Rua 11, Centro', 'Riacho do Norte', 'PA', '1212-1212',
'jjc@riacho.com');
INSERT INTO PessoaJuridica(idPessoa, cnpj)
VALUES (@ID, '22222222222222');
GO

SELECT * FROM Pessoa JOIN PessoaFisica ON (Pessoa.idPessoa =
PessoaFisica.idPessoa);
SELECT * FROM Pessoa JOIN PessoaJuridica ON (Pessoa.idPessoa =
PessoaJuridica.idPessoa);
GO
```



	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cpf
1	2	Joao	Rua 12, casa 3, Quintanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	2	11111111111

	idPessoa	nome	logradouro	cidade	estado	telefone	email	idPessoa	cnpj
1	3	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	3	22222222222222

```
-- Conjunto Movimentações
UPDATE Produto SET precoVenda = 4.00 WHERE idProduto = 1;
GO

INSERT INTO Movimento (idMovimento, idPessoa, idProduto, idUsuario, tipo,
quantidade, valorUnitario)
VALUES
(1, (SELECT idPessoa FROM Pessoa WHERE nome = 'Joao'), 1, 1, 'S', 20, (SELECT
precoVenda FROM Produto WHERE idProduto = 1)),
(4, (SELECT idPessoa FROM Pessoa WHERE nome = 'Joao'), 3, 1, 'S', 15, (SELECT
precoVenda FROM Produto WHERE idProduto = 3));
GO

UPDATE Produto SET precoVenda = 3.00 WHERE idProduto = 3;
GO

INSERT INTO Movimento (idMovimento, idPessoa, idProduto, idUsuario, tipo,
quantidade, valorUnitario)
VALUES
(5, (SELECT idPessoa FROM Pessoa WHERE nome = 'Joao'), 3, 2, 'S', 10, (SELECT
precoVenda FROM Produto WHERE idProduto = 3)),
(7, (SELECT idPessoa FROM Pessoa WHERE nome = 'JJC'), 3, 1, 'E', 15, 5.00),
(8, (SELECT idPessoa FROM Pessoa WHERE nome = 'JJC'), 4, 1, 'E', 20, 4.00);
GO

SELECT * FROM Movimento;
GO
```

	idMovimento	idPessoa	idProduto	idUsuario	tipo	quantidade	valorUnitario
1	1	2	1	1	S	20	4
2	4	2	3	1	S	15	2
3	5	2	3	2	S	10	3
4	7	3	3	1	E	15	5
5	8	3	4	1	E	20	4

```
-- Dados completos de pessoas físicas
SELECT p.idPessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email,
pf.cpf
FROM Pessoa p JOIN PessoaFisica pf ON (p.idPessoa = pf.idPessoa);
GO
```

	idPessoa	nome	logradouro	cidade	estado	telefone
1	2	Joao	Rua 12, casa 3, Quintanda	Riacho do Sul	PA	1111-1111

```
-- Dados completos de pessoas jurídicas
SELECT p.idPessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email,
pj.cnpj
FROM Pessoa p JOIN PessoaJuridica pj ON (p.idPessoa = pj.idPessoa);
GO
```

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	3	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com

```sql
-- Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário
e valor total
SELECT pr.nome, p.nome AS 'Fornecedor', m.quantidade, m.valorUnitario AS 'preço
unitário', (m.quantidade * m.valorUnitario) AS 'valor total'
FROM Movimento m JOIN Produto pr ON (m.idProduto = pr.idProduto) JOIN Pessoa p ON
(m.idPessoa = p.idPessoa)
WHERE m.tipo = 'E';
GO
```

	nome	Fornecedor	quantidade	preço unitário	valor total
1	Laranja	JJC	15	5	75
2	Manga	JJC	20	4	80

```sql
-- Movimentações de saída, com produto, comprador, quantidade, preço unitário e
valor total
SELECT pr.nome, p.nome AS 'Comprador', m.quantidade, m.valorUnitario AS 'preço
unitário', (m.quantidade * m.valorUnitario) AS 'valor total'
FROM Movimento m JOIN Produto pr ON (m.idProduto = pr.idProduto) JOIN Pessoa p ON
(m.idPessoa = p.idPessoa)
WHERE m.tipo = 'S';
GO
```

	nome	Comprador	quantidade	preço unitário	valor total
1	Banana	Joao	20	4	80
2	Laranja	Joao	15	2	30
3	Laranja	Joao	10	3	30

```sql
-- Valor total das entradas agrupadas por produto
SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) AS 'valor total'
FROM Movimento m JOIN Produto pr ON (m.idProduto = pr.idProduto) WHERE m.tipo =
'E' GROUP BY pr.nome;
GO
```

	nome	valor total
1	Laranja	75
2	Manga	80

```sql
-- Valor total das saídas agrupadas por produto
SELECT pr.nome, SUM(m.quantidade * m.valorUnitario) AS 'valor total'
FROM Movimento m JOIN Produto pr ON (m.idProduto = pr.idProduto) WHERE m.tipo =
'S' GROUP BY pr.nome;
GO
```

	nome	valor total
1	Banana	80
2	Laranja	60

```
-- Operadores que não efetuaram movimentações de entrada (compra)
SELECT idUsuario FROM Movimento WHERE idUsuario NOT IN (SELECT idUsuario FROM
Movimento WHERE tipo = 'E');
GO
```

	idUsuario
1	2

```
-- Valor total de entrada, agrupado por operador
SELECT idUsuario, SUM(quantidade * valorUnitario) AS 'valor total de entrada'
FROM Movimento WHERE tipo = 'E' GROUP BY idUsuario;
GO
```

	idUsuario	valor total de entrada
1	1	155

```
-- Valor total de saída, agrupado por operador
SELECT idUsuario, SUM(quantidade * valorUnitario) AS 'valor total de saída' FROM
Movimento WHERE tipo = 'S' GROUP BY idUsuario;
GO
```

	idUsuario	valor total de saída
1	1	110
2	2	30

```
-- Valor médio de venda por produto.
SELECT idProduto, (SUM(quantidade) / COUNT(idProduto)) AS 'valor médio de qt por
venda'
FROM Movimento WHERE tipo = 'S' GROUP BY idProduto;
GO
```

	idProduto	valor médio de qt por venda
1	1	20
2	3	12

Conclusão:

a.Quais as diferenças no uso de *sequence* e *identity*?

R) Sequence está vinculado ao esquema podendo ser usada por todo ele fazendo uso do commando NEXT VALUE FOR, enquanto identity está vinculado a coluna de uma tabela gerando automaticamente os valores para a coluna quando necessário.

b.Qual a importância das chaves estrangerias para a consistência do banco?

R) Garante o correto relacionamente esntre as diferentes tabelas.

c.Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

R) É usado para agrupar linhas que possuem o mesmo valor em determinadas colunas, sendo necessário utilizar no comando SELECT apenas colunas que estejam sendo agrupadas ou operadores de agregação.