Universidade Estácio de Sá

- Curso: Desenvolvimento Full stack - TURMA 9001
- Disciplina: RPG0015 - Vamos Manter as Informações!
- Semestre Letivo: 2024.4
- Aluno: Jonathan Sendi Inowe
- Matricula: 202311117502

Missão Prática | Nível 2 | Mundo 3

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

Procedimento 1: Criando o Banco de Dados

Procedimento 2: Alimentando a Base

Objetivos da Prática

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML).
- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server

---

Códigos

Procedimento 1: Criando o Banco de Dados

```
create database Loja;

use Loja;

create table pessoa(
  idpessoa int NOT NULL ,
```

```
  nome varchar(255) NOT NULL  ,
  logradouro varchar(255) NOT  NULL  ,
  cidade varchar(255)NOT  NULL  ,
  estado char(2)NOT  NULL  ,
  telefone varchar(11)NOT  NULL  ,
  email varchar(255)NOT  NULL    ,
primary key(idpessoa));

create table pessoa_fisica (
 idpessoa int NOT NULL,
  cpf varchar(255)  NOT NULL,
  primary key (idpessoa),
  foreign key (idpessoa) references pessoa(idpessoa));

create table pessoa_juridica (
  idpessoa int NOT NULL,
  cnpj varchar(255)  NOT NULL,
  primary key (idpessoa),
  foreign key (idpessoa) references pessoa(idpessoa));

create table produto (
  idproduto int NOT NULL  ,
  nome varchar(255)  NOT NULL  ,
  quantidade varchar(255)  NOT NULL  ,
  preco_venda numeric(5,2)  NOT NULL    ,
primary key(idproduto));

create table usuario (
  idusuario int NOT NULL ,
  login varchar(255)   NOT NULL  ,
  senha varchar(255)   NOT NULL    ,
primary key(idusuario));

create table movimento (
  idmovimento int NOT NULL,
  Usuario_idUsuario int  NOT NULL  ,
  pessoa_idpessoa int  NOT NULL  ,
  produto_idproduto int  NOT NULL  ,
  quantidade int  NOT NULL  ,
  tipo char  NOT NULL  ,
  valorUnitario numeric(5,2)  NOT NULL    ,
primary key(idmovimento),
foreign key (Usuario_idUsuario) references usuario(idusuario),
foreign key (produto_idproduto) references produto(idproduto),
foreign key (pessoa_idpessoa) references pessoa(idpessoa));

create sequence seq_Pessoa
      as numeric
      start with 1
      increment by 1
      no cycle;
```

Procedimento 2: Alimentando a Base

```
use Loja;

insert into usuario
values (1, 'op1', 'op1'),(2, 'op2', 'op2');
```

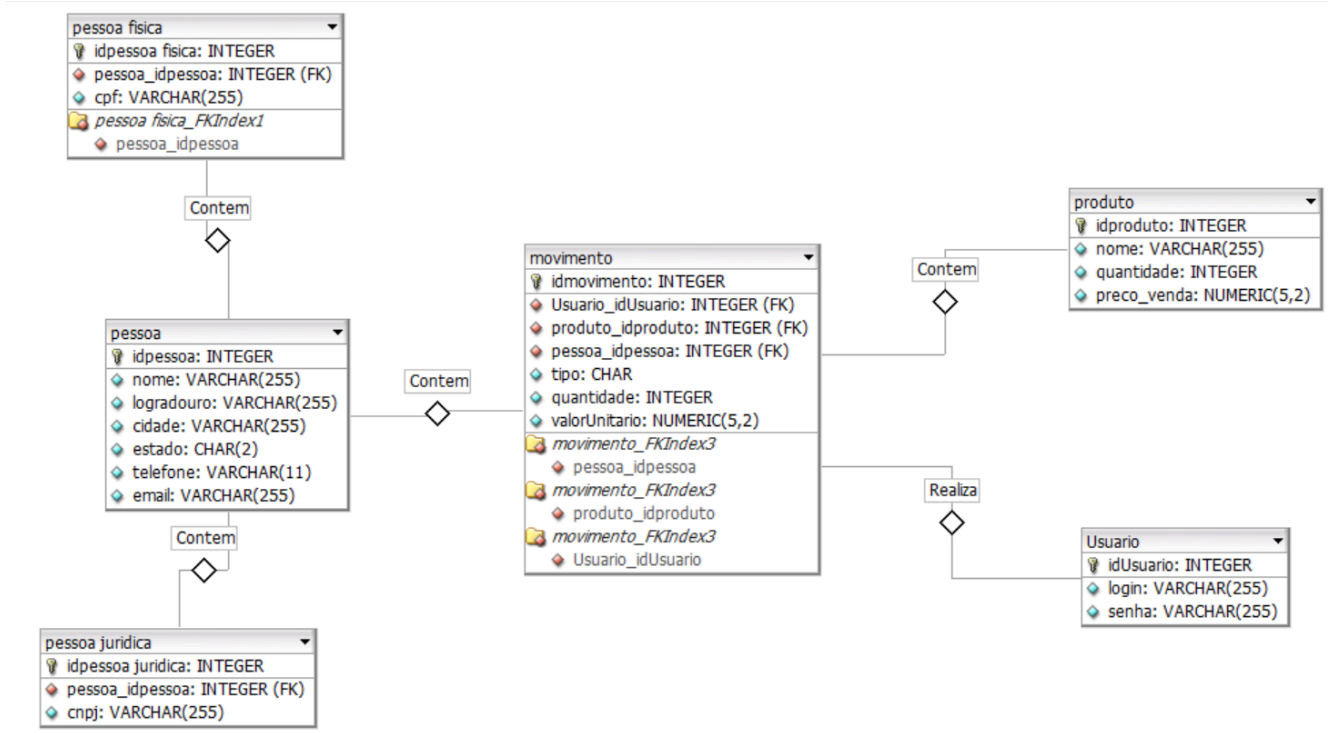
```
insert into produto
values (1, 'Banana', 100, 5.00),(3, 'Laranja', 500, 2.00),(4, 'Manga',
800, 4.00);

insert into pessoa
values (NEXT VALUE FOR seq_Pessoa, 'Joao', 'Rua 12, cas 3, Quitanda',
'Riacho do Sul', 'PA', '1111-1111','joao@riacho.com');

insert into pessoa
values (NEXT VALUE FOR seq_Pessoa, 'JJC', 'Rua 11, Centro',
'Riacho do Norte', 'PA', '1212-1212','jjc@riacho.com');

insert into pessoa_fisica
values (1,'11111111111');

insert into pessoa_juridica
values (2,'22222222222222');

insert into movimento
values (1,1,1,1,20,'S',4.00),
(4,1,1,3,15,'S',2.00),
(5,2,1,3,10,'S',3.00),
(7,1,2,3,15,'E',5),
(8,1,2,4,20,'E',4.00);

-- Dados completos de pessoas físicas.
select *
from pessoa, pessoa_fisica
where pessoa.idpessoa = pessoa_fisica.idpessoa;

--Dados completos de pessoas jurídicas.
select *
from pessoa, pessoa_juridica
where pessoa.idpessoa = pessoa_juridica.idpessoa;

--Movimentações de entrada, com produto, fornecedor, quantidade, preço
unitário e valor total.
select idmovimento, produto_idproduto, produto.nome as
'Produto',pessoa_idpessoa, pessoa.nome as 'Fornecedor',
movimento.quantidade, valorUnitario,
(movimento.quantidade *  valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'E';

--Movimentações de saída, com produto, comprador, quantidade, preço
unitário e valor total
select idmovimento, produto_idproduto, produto.nome as
'Produto',pessoa_idpessoa, pessoa.nome as 'Comprador',
movimento.quantidade, valorUnitario,
(movimento.quantidade * valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
```

```
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'S';

--Valor total das entradas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL ENTRADAS'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'E'
group by produto.nome;

--Valor total das saídas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL SAIDAS'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;

--Operadores que não efetuaram movimentações de entrada (compra).
select movimento.Usuario_idUsuario AS 'ID DO OPERADOR'
from movimento
except
select movimento.Usuario_idUsuario
from movimento
where movimento.tipo = 'E';

--Valor total de entrada, agrupado por operador.
select usuario.login AS OPERADOR, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL ENTRADAS'
from movimento
JOIN usuario
on usuario.idusuario = movimento.Usuario_idUsuario
where movimento.tipo = 'E'
group by usuario.login;

--Valor total de saída, agrupado por operador.
select usuario.login AS OPERADOR, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL SAIDAS'
from movimento
JOIN usuario
on usuario.idusuario = movimento.Usuario_idUsuario
where movimento.tipo = 'S'
group by usuario.login;

--Valor médio de venda por produto, utilizando média ponderada.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) / SUM(movimento.quantidade) as 'Valor médio de
venda'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;
```

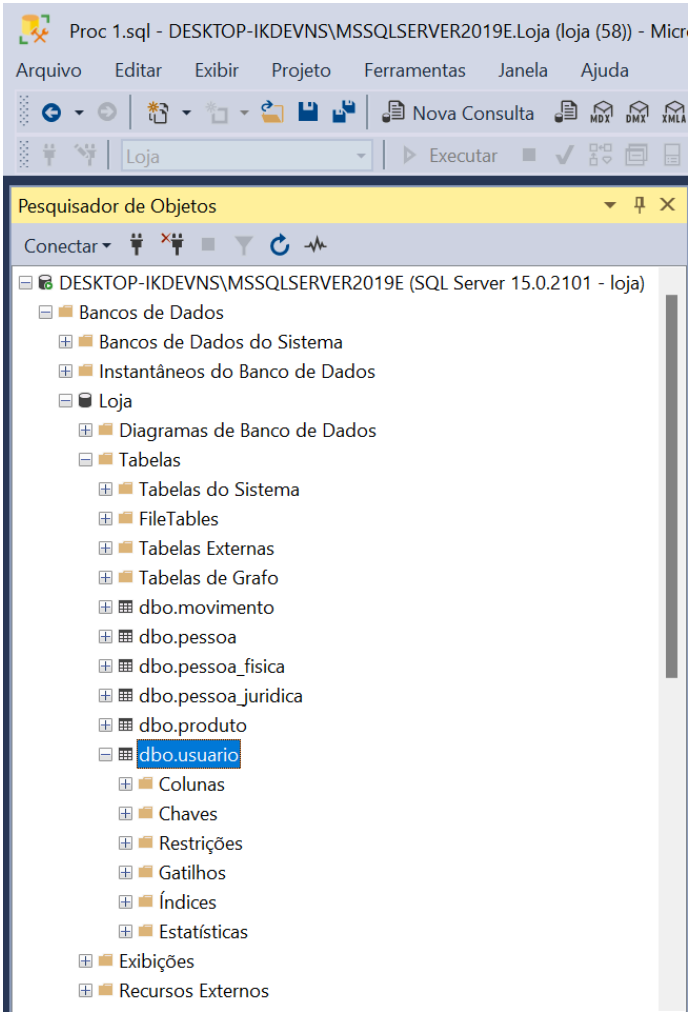
Resultados:

Procedimento 1:

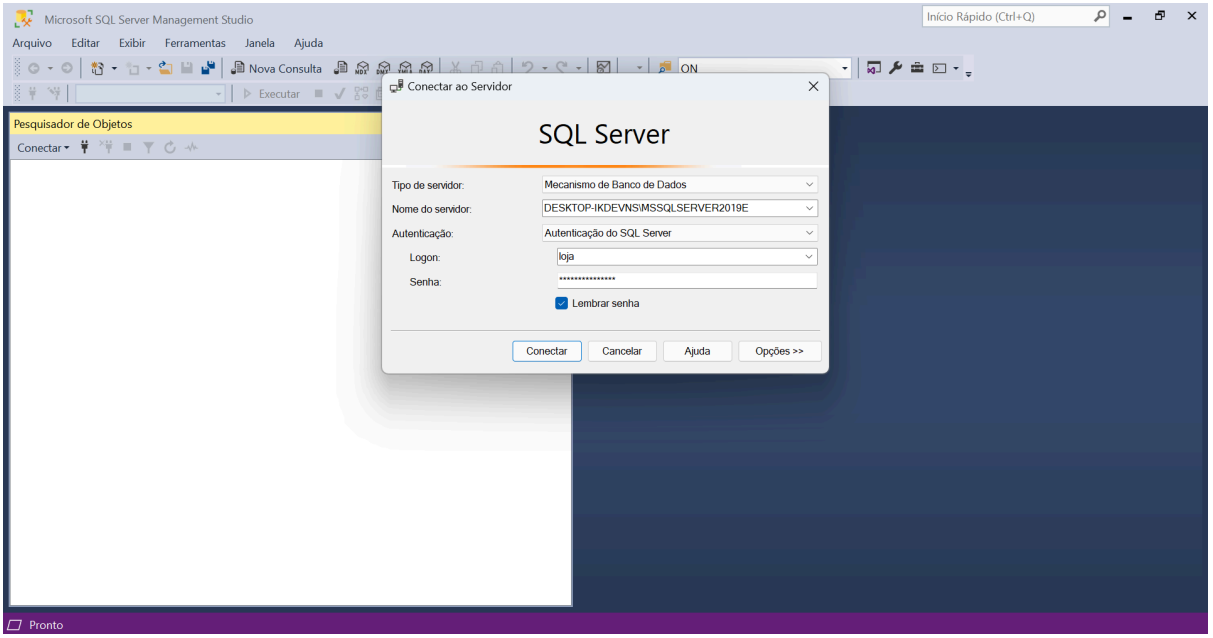
Modelo

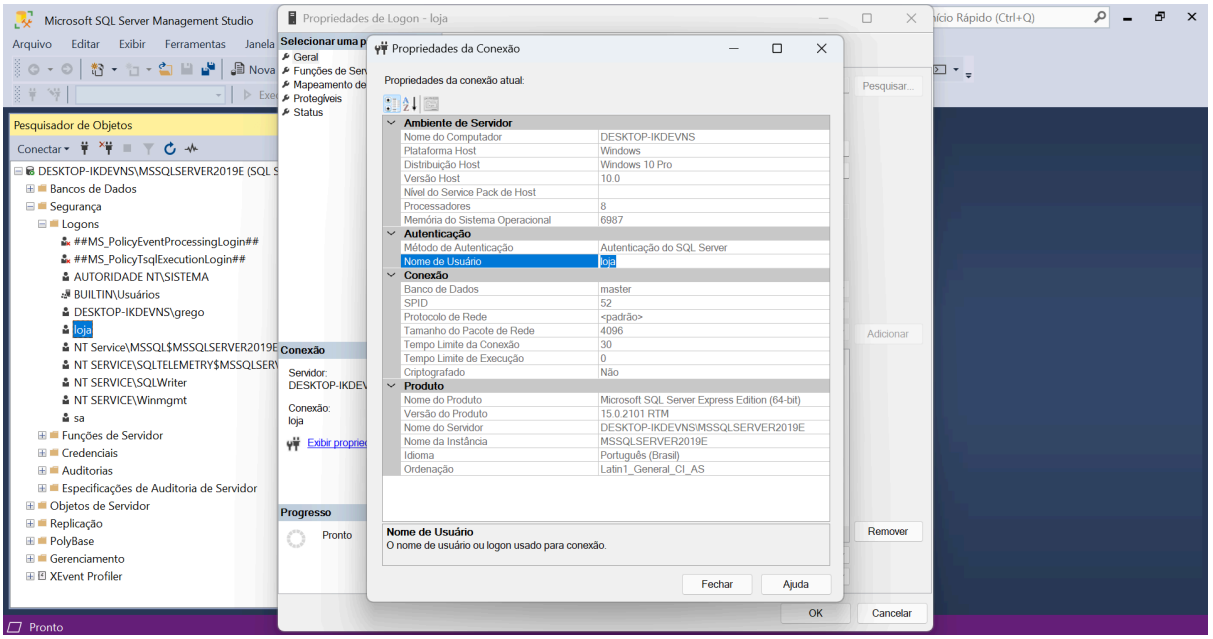


Banco e Tabelas



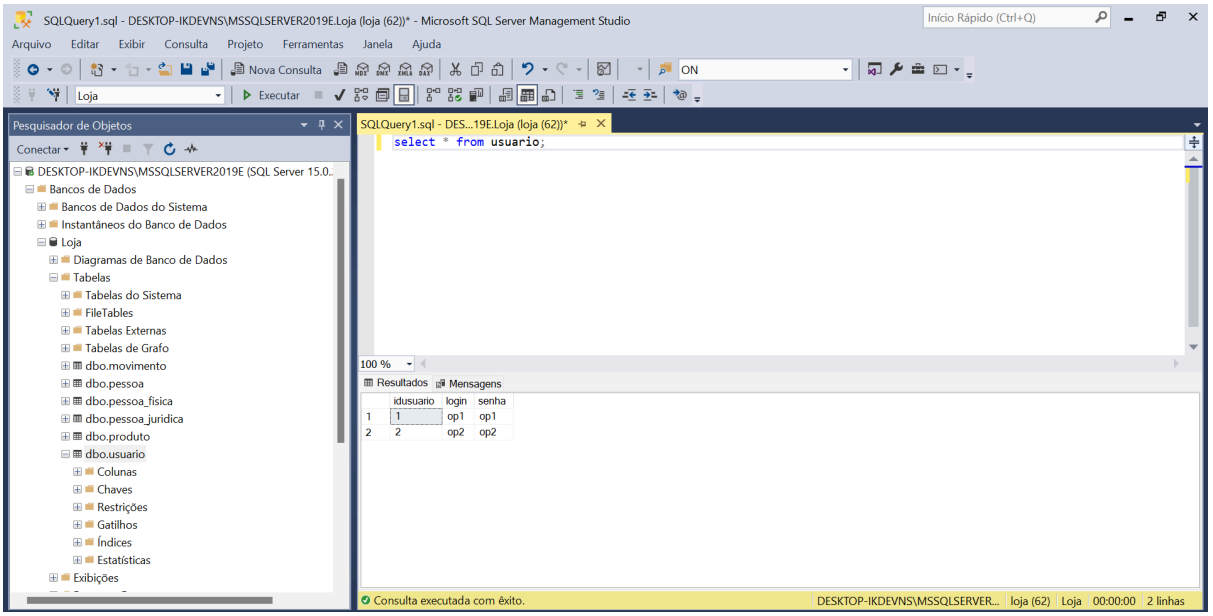
Usuário/Logon Loja



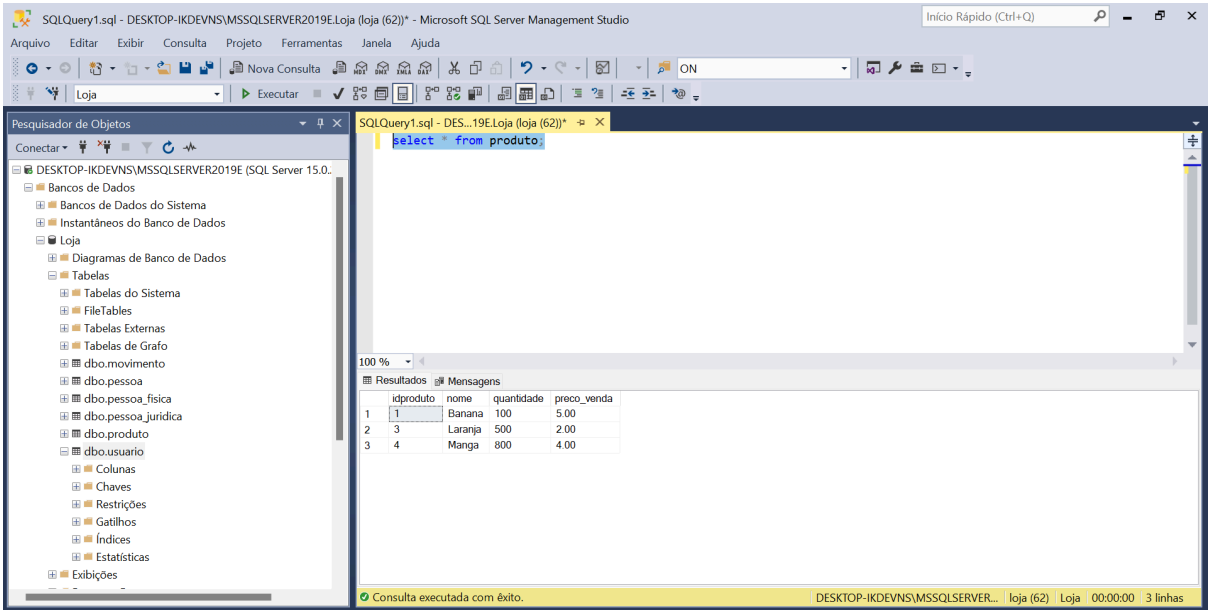


Procedimento 2:

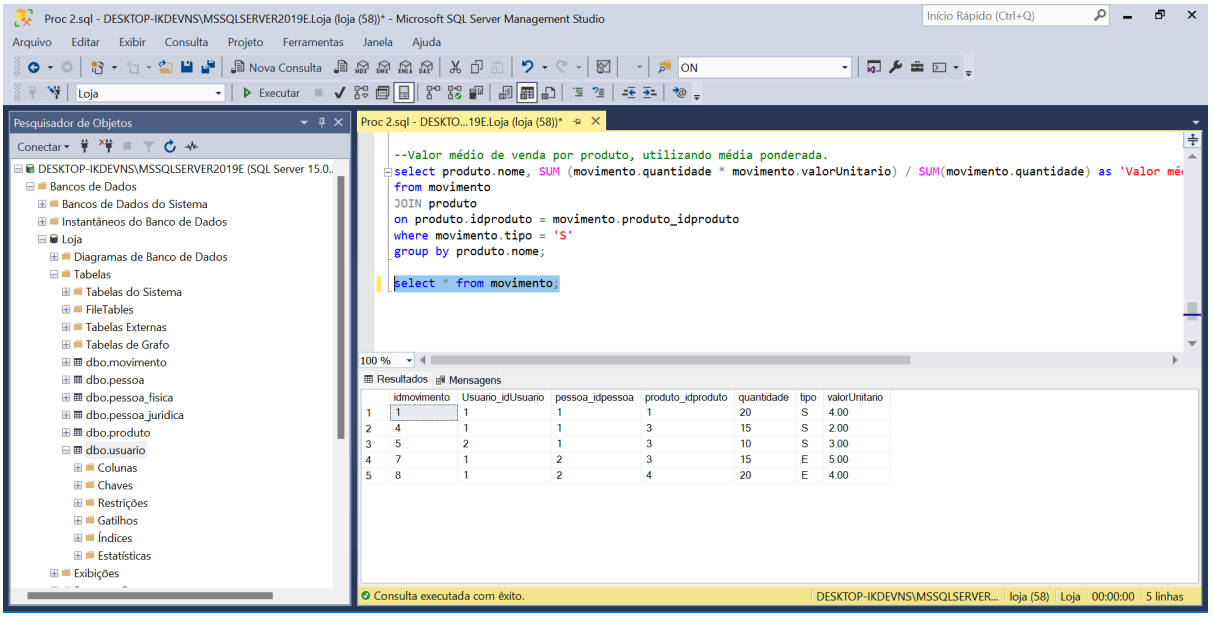
Usuários



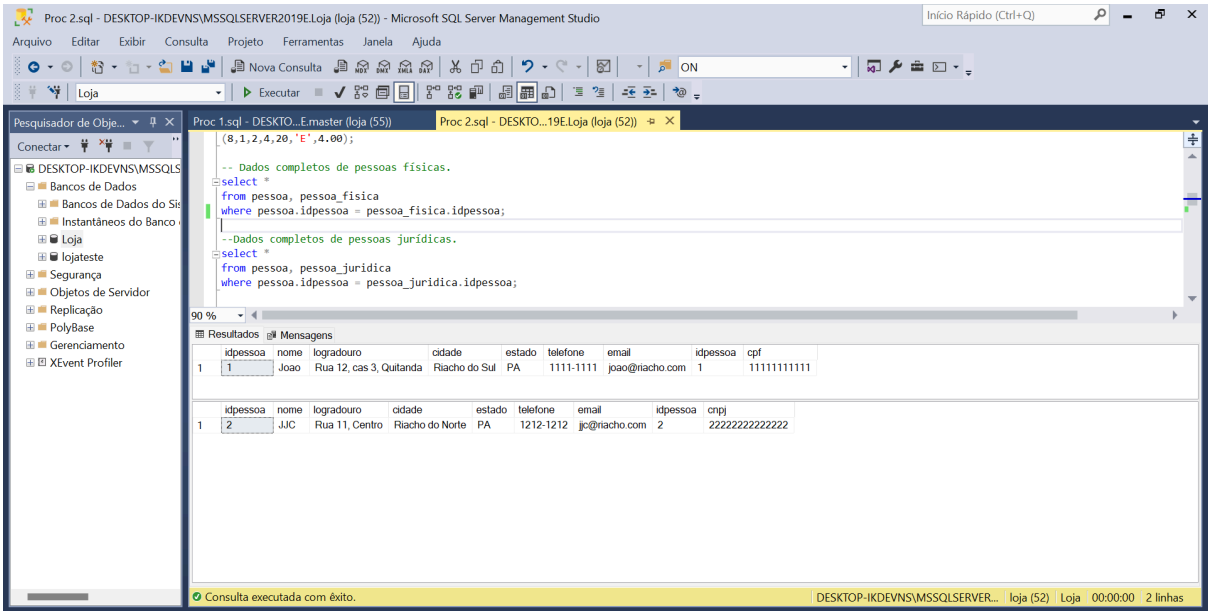
Produtos



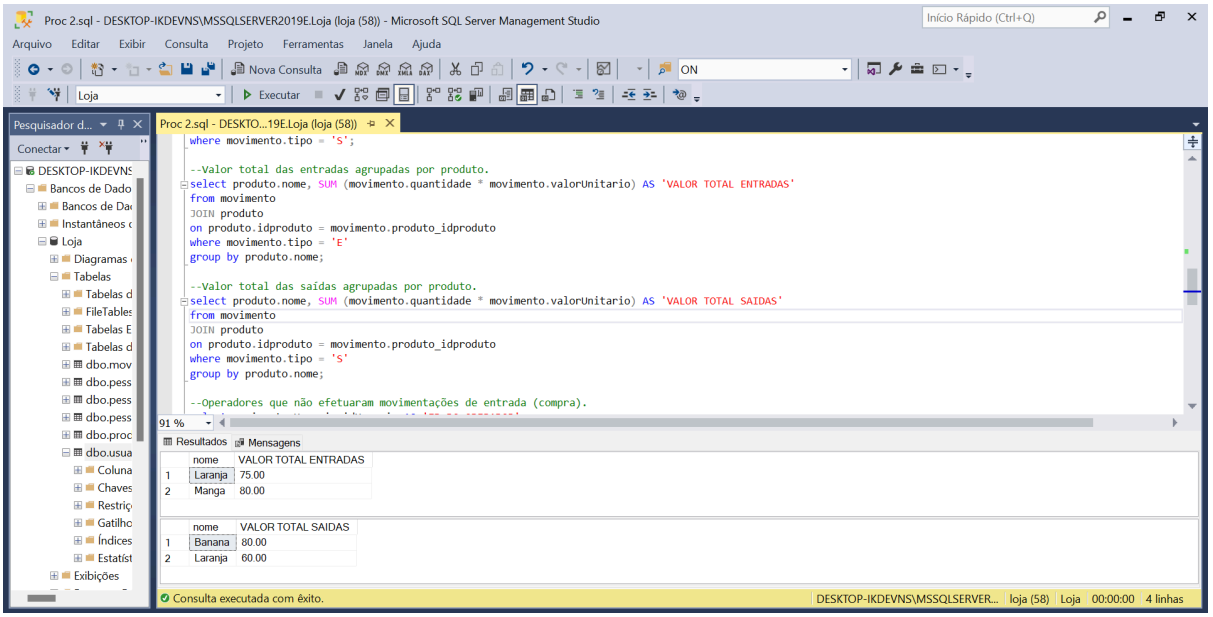
Movimentos



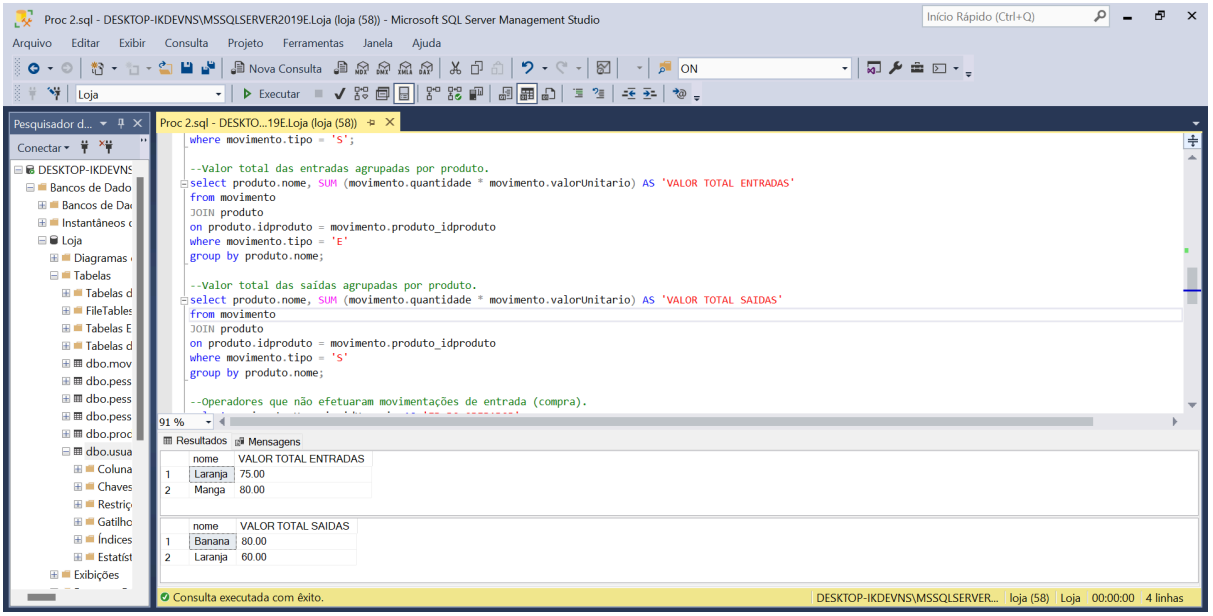
Dados completos de pessoas físicas e jurídicas



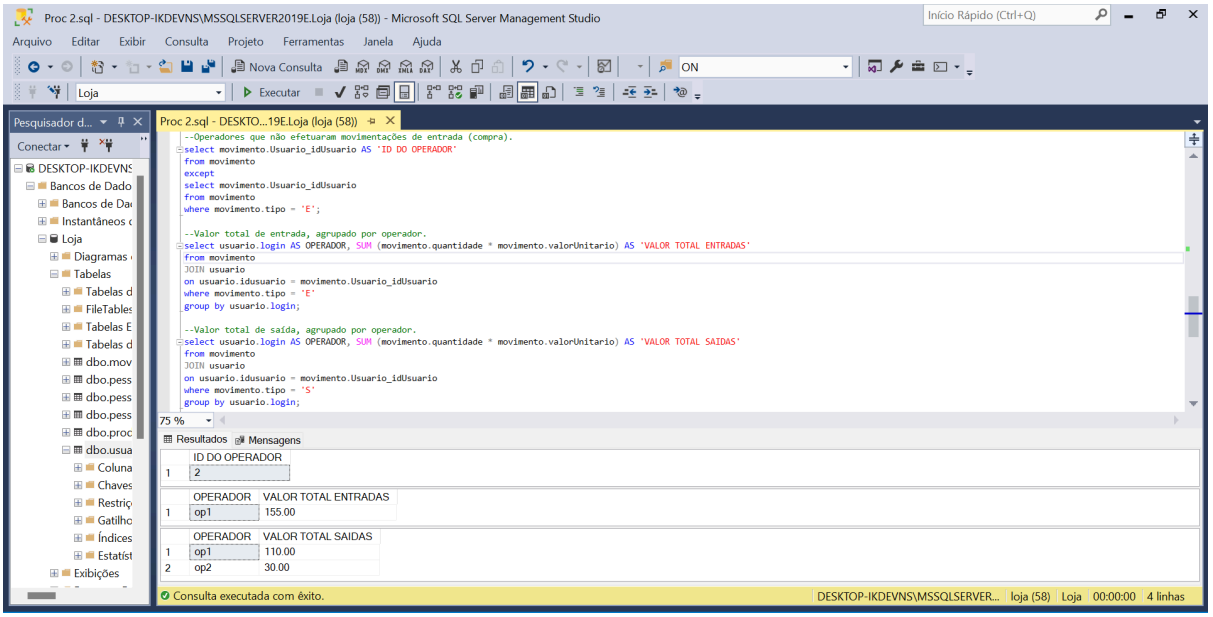
Movimentações de entrada e saída, com produto, comprador, fornecedor, quantidade, preço unitário e valor total



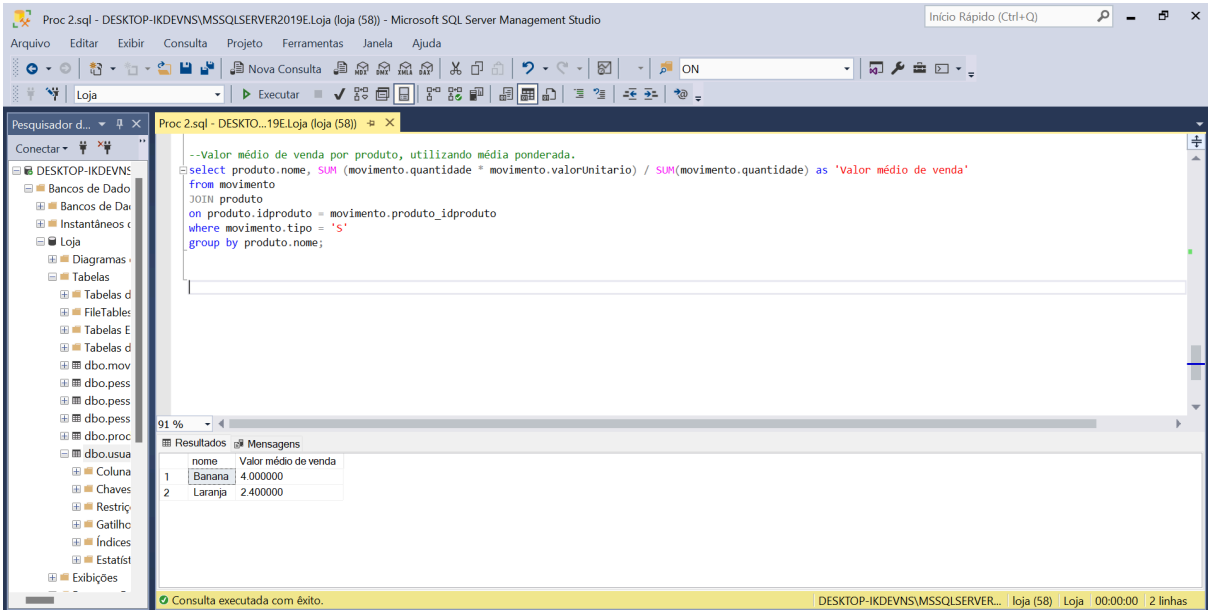
Valor total das entradas e saídas agrupadas por produto



Operadores que não efetuaram movimentações de entrada (compra).Valor total de entrada e saida agrupado por operador



Valor médio de venda por produto, utilizando média ponderada



Análise e Conclusão

- **Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?**
  Por meio dos níveis de relacionamento existentes entre entidades ou tabelas.

- **Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?**
  1x1

- **Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?**
  Utilizando um editor de consultas, ferramentas de monitoramento de desempenho, recursos de segurança e controle de permissões.

- **Quais as diferenças no uso de sequence e identity?**
  As SEQUENCES são acionadas sempre que necessário, de forma independente de tabelas e campos do banco de dados, podendo ser chamadas diretamente por aplicativos. Com as SEQUENCES, é possível obter o próximo valor antes de utilizá-lo em um comando, o que difere do comportamento do IDENTITY, que não permite essa funcionalidade. Além disso, o IDENTITY não suporta a geração de novos valores em uma instrução UPDATE, enquanto as SEQUENCES permitem essa operação. Outro benefício das SEQUENCES é a flexibilidade para definir valores máximos e mínimos, configurar seu funcionamento em modo cíclico, utilizar cache para otimização e gerar múltiplos valores sequenciais de uma só vez. Para isso, pode-se usar a procedure SP_SEQUENCE_GET_RANGE, que permite atribuir valores individuais, aumentando o desempenho. Por outro lado, o IDENTITY é útil em cenários que envolvem transações de INSERT, pois o próximo valor só é gerado quando o comando é efetivamente executado e a transação é confirmada. Já com as SEQUENCES, ao chamar o próximo valor, ele será alterado mesmo que ocorra um erro na transação.

- **Qual a importância das chaves estrangerias para a consistência do banco?**
  O uso de chaves estrangeiras permite implementar a integridade dos dados diretamente no banco, conhecida como integridade referencial. Uma chave estrangeira serve como a representação de um vínculo entre tabelas.

- **Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?**

  Operadores do SQL pertencem à álgebra relacional:
  SELEÇÃO, RESTRIÇÃO, PROJEÇÃO, UNIÃO, INTERSECÇÃO, DIFERENÇA DE CONJUNTOS, PRODUTO CARTESIANO, JUNÇÃO, DIVISÃO, RENOMEAÇÃO, ATRIBUIÇÃO;

  Operadores do SQL pertencem AO cálculo relacional:
  Igual, diferente, maior, menor, maior ou igual, menor ou igual.

- **Como é feito o agrupamento em consultas, e qual requisito é obrigatório?**
  Utilizando o "GRUPO BY".