



## Missão Prática - Nível 2 Mundo 3

**Campus : POLO COPACABANA**

**Curso : DESENVOLVIMENTO FULL STACK**

**Disciplina : RPG0015 - Vamos manter as informações**

**Turma : 9003**

**Semestre: 2023.3 FLEX**

**Nome : MARCO SERGIO ALBINO VITTORIO BAROZZI**









### **1º Procedimento | Criação das Entidades e Sistema de Persistência**

#### **Objetivos:**

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados Relacionais
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
5. 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 desenvolvidos :

### *Execução de scripts*

 01_Pratica.m3,n2.part1 banco user.sql	15/09/2023 20:13	Arquivo SQL	4 KB
 02_Pratica.m3,n2.part2 create tbs views.sql	03/10/2023 04:55	Arquivo SQL	9 KB
 03_Pratica.m3,n2.part3 insert tbs pessoas usuario.sql	02/10/2023 04:35	Arquivo SQL	3 KB
 04_Pratica.m3,n2.part4 prod.sql	14/09/2023 07:02	Arquivo SQL	1 KB
 05_Pratica.m3,n2.part5 mov E.sql	14/09/2023 07:52	Arquivo SQL	4 KB
 06_Pratica.m3,n2.part6 mov S.sql	14/09/2023 08:05	Arquivo SQL	4 KB
 07_Compras Extras part8.sql	14/09/2023 19:20	Arquivo SQL	2 KB
 08_Vendas Extrass part7.sql	14/09/2023 20:14	Arquivo SQL	2 KB

| Criação do banco loja e usuario loja

01\_Pratica.m3,n2.part1 banco user.sql

- Criação de todas as tabelas

02\_Pratica.m3,n2.part2 create tbs views.sql

*Obs: a execução dos scripts partem do pré-suposto que o banco "loja" não exista e usar inicialmente usuário sa*

### **01 - Criação do banco loja e usuario loja**

**---ATENÇÃO - NECESSÁRIO ESTAR LOGADO COM 'sa' e o management ter sido inicializado**

**-- Criação do banco e usuário**

```
USE master;  
GO
```

```
-- Mata todas as conexões para o banco de dados 'loja'  
DECLARE @Kill VARCHAR(8000) = '';
```

```
SELECT @Kill = @Kill + 'KILL ' + CONVERT(VARCHAR(5), SPID) + ';'   
FROM master..sysprocesses  
WHERE DBId = DB_ID('loja') AND SPID <> @@SPID;
```

```
EXEC (@Kill);
```

```
-- Agora, tente excluir o banco de dados  
DROP DATABASE loja;  
GO
```

```
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 , MAXSIZE = UNLIMITED, 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 , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )  
WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER =  
OFF  
GO
```

```
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))  
begin  
EXEC [loja].[dbo].[sp_fulltext_database] @action = 'enable'  
end  
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_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 TRUSTWORTHY OFF  
GO

ALTER DATABASE [loja] SET ALLOW\_SNAPSHOT\_ISOLATION OFF  
GO

ALTER DATABASE [loja] SET PARAMETERIZATION SIMPLE  
GO

ALTER DATABASE [loja] SET READ\_COMMITTED\_SNAPSHOT OFF  
GO

ALTER DATABASE [loja] SET HONOR\_BROKER\_PRIORITY OFF  
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 DB\_CHAINING OFF  
GO

ALTER DATABASE [loja] SET FILESTREAM(  
NON\_TRANSACTED\_ACCESS = OFF )  
GO

```
ALTER DATABASE [loja] SET TARGET_RECOVERY_TIME = 60
SECONDS
GO
```

```
ALTER DATABASE [loja] SET DELAYED_DURABILITY = DISABLED
GO
```

```
ALTER DATABASE [loja] SET
ACCELERATED_DATABASE_RECOVERY = OFF
GO
```

```
ALTER DATABASE [loja] SET QUERY_STORE = ON
GO
```

```
ALTER DATABASE [loja] SET QUERY_STORE (OPERATION_MODE
= READ_WRITE, CLEANUP_POLICY =
(STALE_QUERY_THRESHOLD_DAYS = 30),
DATA_FLUSH_INTERVAL_SECONDS = 900,
INTERVAL_LENGTH_MINUTES = 60, MAX_STORAGE_SIZE_MB =
1000, QUERY_CAPTURE_MODE = AUTO,
SIZE_BASED_CLEANUP_MODE = AUTO,
MAX_PLANS_PER_QUERY = 200, WAIT_STATS_CAPTURE_MODE =
ON)
GO
```

```
ALTER DATABASE [loja] SET READ_WRITE
GO
```

-- criação do usuario loja

```
USE [master]
```

```
GO
```

```
IF EXISTS (SELECT 1 FROM sys.server_principals WHERE name =
'loja')
BEGIN
    DROP LOGIN [loja] -- Substitua 'UserName' pelo nome do usuário
END
```

```

GO
CREATE LOGIN [loja] WITH PASSWORD=N'loja',
DEFAULT_DATABASE=[loja], CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF

-- Concede o papel de sistema sysadmin, que tem privilégios de nível
superior
-- verificar porque esta sendo necessário para conexão com app
EXEC sp_addsrvrolemember 'loja', 'sysadmin';
GO

GO
USE [loja]
GO
CREATE USER [loja] FOR LOGIN [LOJA]
GO
USE [loja]
GO
ALTER ROLE [db_datareader] ADD MEMBER [LOJA]
GO
USE [loja]
GO
ALTER ROLE [db_datawriter] ADD MEMBER [LOJA]
GO
USE [loja]
GO
ALTER ROLE [db_ddladmin] ADD MEMBER [LOJA]
GO
USE [loja]
GO
ALTER ROLE [db_owner] ADD MEMBER [LOJA]
GO

```

**02 - create tbs pessoa, pessoaFisica, pessoa juridica, usuário e produto.**

**---ATENÇÃO - NECESSÁRIO ESTAR LOGADO COM 'loja'**

```

USE loja;
GO
-----
--- DROP VIEWS
-----
---Movimento precisa ser removida antes devido a integridade com as
demais
-----
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME = 'movimento')
BEGIN
    DROP TABLE movimento;
END
-----
-- DROP Produto
-----
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME = 'Produto')
BEGIN
    DROP TABLE Produto;
END
---usuario
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_NAME = 'Usuario')
BEGIN
    DROP TABLE Usuario;
END

-----
--- DROP sequence pessoa
-----
IF EXISTS (SELECT * FROM sys.sequences WHERE name =
'PessoaldSeq')
BEGIN
    DROP SEQUENCE PessoaldSeq;
END
-----
--PessoaFisica e Juridica devem ser removidas antes
--PessoaFisica

```



```
-----  
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_NAME = 'PessoaFisica')  
BEGIN  
    DROP TABLE PessoaFisica;  
END  
-----
```

```
-- DROP PessoaJuridica
```

```
-----  
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_NAME = 'PessoaJuridica')  
BEGIN  
    DROP TABLE PessoaJuridica;  
END  
-----
```

```
--- DROP Pessoa
```

```
-----  
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_NAME = 'Pessoa')  
BEGIN  
    DROP TABLE Pessoa;  
END  
-----
```

```
---CREATE TABLES
```

```
--- Produto
```

```
-----  
CREATE TABLE Produto (  
    idProduto INT IDENTITY(1,1),  
    nome VARCHAR(45),  
    quantidade INT,  
    precoVenda DECIMAL(10, 2),  
    PRIMARY KEY (idProduto)  
);  
-----
```

```
-- Usuario
```

```
-----  
CREATE TABLE Usuario (  
    idUsuario INT IDENTITY(1,1),  
    -----
```

```
login CHAR(100) NOT NULL UNIQUE,  
senha CHAR(150) NOT NULL,  
PRIMARY KEY (idUserio)  
);
```

```
-----  
-- sequence PessoaFisica e Juridica  
-----
```

```
CREATE SEQUENCE PessoaIdSeq  
AS INT  
START WITH 1  
INCREMENT BY 1;  
-----
```

```
-- Pessoa  
-----
```

```
CREATE TABLE Pessoa (  
    id INT PRIMARY KEY NOT NULL,  
    nome NVARCHAR(60) NOT NULL,  
    logradouro VARCHAR(255),  
    cidade VARCHAR(100),  
    estado VARCHAR(2),  
    telefone VARCHAR(11),  
    email VARCHAR(255),  
    tipo CHAR(1) CHECK (tipo IN ('F', 'J')) NOT NULL -- F para física, J  
para jurídica  
);
```

```
-----  
-- Pessoafisica  
-----
```

```
CREATE TABLE PessoaFisica (  
    idpessoa INT PRIMARY KEY,  
    cpf CHAR(11),  
    FOREIGN KEY (idpessoa) REFERENCES Pessoa(id)  
    ON DELETE CASCADE  
);
```

```
-- PessoaJuridica
```

```
CREATE TABLE PessoaJuridica (  
    idpessoa INT PRIMARY KEY,  
    cnpj CHAR(14),  
    FOREIGN KEY (idpessoa) REFERENCES Pessoa(id)  
    ON DELETE CASCADE  
);
```

---

```
--      Movimento
```

---

```
CREATE TABLE Movimento (  
    idMovimento INT IDENTITY(1,1),  
    idUsuario INT,  
    idPessoa INT,  
    idProduto INT,  
    quantidade INT,  
    valorUnitario decimal(10,2),  
    tipo CHAR(1),  
    PRIMARY KEY (idMovimento),  
    FOREIGN KEY (idProduto)  
        REFERENCES Produto(idProduto)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE,  
    FOREIGN KEY (idUsuario)  
        REFERENCES Usuario(idUsuario)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE,  
    FOREIGN KEY (idPessoa)  
        REFERENCES Pessoa(id)  
        ON DELETE NO ACTION  
        ON UPDATE CASCADE  
);
```

---

```
-- CREATE VIEWS PESSOAS
```

---

```
-- fisica
```

---

```
GO
```

```
IF EXISTS (SELECT * FROM sys.views WHERE name =  
'vw_pessoa_fisica')  
DROP VIEW vw_pessoa_fisica;
```

```
-- vou usar top para manter order (regras da view)  
GO
```

```
CREATE VIEW vw_pessoa_fisica AS  
select  
p.id,pf.idPessoa,p.nome,p.logradouro,p.cidade,p.estado,p.telefone,p.em  
ail,pf.cpf  
from (pessoa p inner join pessoafisica pf on p.id=pf.idpessoa)
```

```
-----  
-- Juridica  
-----
```

```
GO
```

```
IF EXISTS (SELECT * FROM sys.views WHERE name =  
'vw_pessoa_juridica')  
DROP VIEW vw_pessoa_juridica;
```

```
-- vou usar top para manter order (regras da view)  
GO
```

```
CREATE VIEW vw_pessoa_juridica AS  
select  
p.id,pj.idPessoa,p.nome,p.logradouro,p.cidade,p.estado,p.telefone,p.em  
ail,pj.cnpj  
from (pessoa p inner join pessoaJuridica pj on p.id=pj.idpessoa)
```

```
-----  
-- Movimentações E e S, com produto, fornecedor  
-----
```

```
GO
```

```
IF EXISTS (SELECT * FROM sys.views WHERE name =
'vw_mov_compras')
DROP VIEW vw_mov_compras;
GO
```

```
-- vou usar top para manter order (regras da view)
GO
```

```
CREATE VIEW vw_mov_compras AS
select
u.login as operador,
mv.tipo as operacao,
p.nome as produto,
mv.quantidade,
mv.valorUnitario,
(mv.valorUnitario*mv.quantidade) as total,
pe.nome as comprador,
pj.cnpj
from (((movimento mv inner join produto p on
p.idProduto=mv.idProduto)
inner join usuario u on u.idUsuario=mv.idUsuario)
left join PessoaJuridica pj on mv.idPessoa=pj.idPessoa) inner join
Pessoa pe on pe.id=pj.idpessoa)
where mv.tipo='E'
GO
```

```
-----
--Movimentações de saída, com produto, comprador, quantidade, preço
unitário e valor total.
-----
```

```
IF EXISTS (SELECT * FROM sys.views WHERE name =
'vw_mov_vendas')
DROP VIEW vw_mov_vendas;
GO
```

```
-- vou usar top para manter order (regras da view)
GO
```

```
CREATE VIEW vw_mov_vendas AS
select
u.login as operador,
```

```

mv.tipo as operacao,
p.nome as produto,
mv.quantidade,
mv.valorUnitario,
(mv.valorUnitario*mv.quantidade) as total,
pe.nome as fornecedor,
pf.cpf
from (((movimento mv inner join produto p on
p.idProduto=mv.idProduto)
inner join usuario u on u.idUsuario=mv.idUsuario)
left join PessoaFisica pf on mv.idPessoa=pf.idPessoa) inner join Pessoa
pe on pe.id=pf.idpessoa)
where mv.tipo='S'

```

GO

```

-----
-- CREATE VIEWS OPERADORES
-----

```

```

--- Compras
-----

```

GO

```

IF EXISTS (SELECT * FROM sys.views WHERE name =
'vw_operador_compras')
DROP VIEW vw_operador_compras;
GO

```

```

-- vou usar top para manter order (regras da view)

```

```

GO
CREATE VIEW vw_operador_compras AS
select
u.login as operador,
mv.tipo as operacao,
p.nome as produto,
mv.quantidade,
mv.valorUnitario,
(mv.valorUnitario*mv.quantidade) as total,
pe.nome as comprador,
pj.cnpj

```

```
from (((movimento mv inner join produto p on
p.idProduto=mv.idProduto)
inner join usuario u on u.idUsuario=mv.idUsuario)
left join PessoaJuridica pj on mv.idPessoa=pj.idPessoa) inner join
Pessoa pe on pe.id=pj.idpessoa)
where mv.tipo='E' ;
GO
```

```
-----
-- vendas
-----
```

```
GO
```

```
IF EXISTS (SELECT * FROM sys.views WHERE name =
'vw_operador_vendas')
DROP VIEW vw_operador_vendas;
GO
```

```
-- vou usar top para manter order (regras da view)
```

```
CREATE VIEW vw_operador_vendas AS
select
u.login as operador,
mv.tipo as operacao,
p.nome as produto,
mv.quantidade,
mv.valorUnitario,
(mv.valorUnitario*mv.quantidade) as total,
pe.nome as fornecedor,
pf.cpf
from (((movimento mv inner join produto p on
p.idProduto=mv.idProduto)
inner join usuario u on u.idUsuario=mv.idUsuario)
left join PessoaFisica pf on mv.idPessoa=pf.idPessoa) inner join Pessoa
pe on pe.id=pf.idpessoa)
where mv.tipo='S'
GO
```

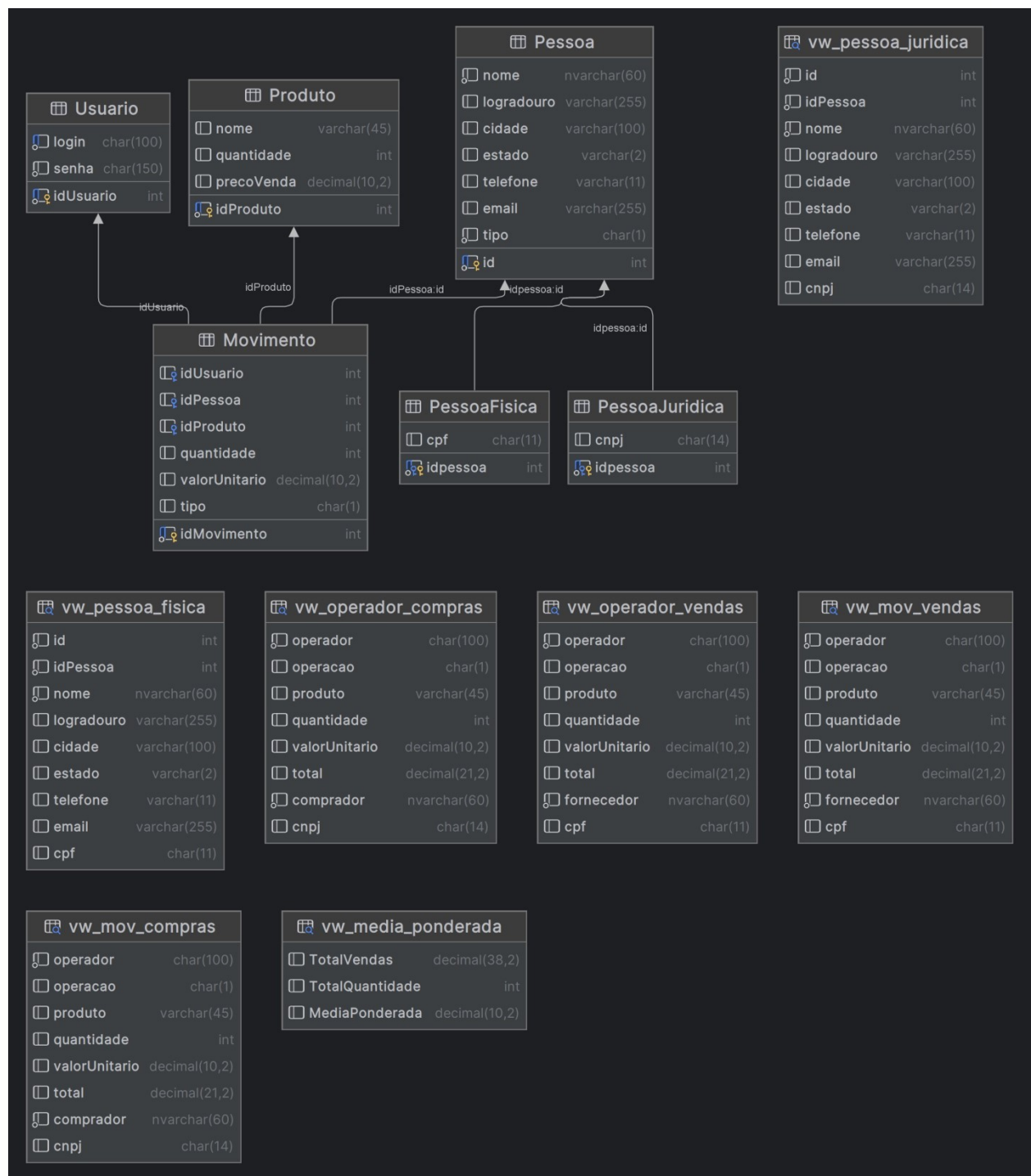
```
IF EXISTS (SELECT * FROM sys.views WHERE name =
'vw_media_ponderada')
```

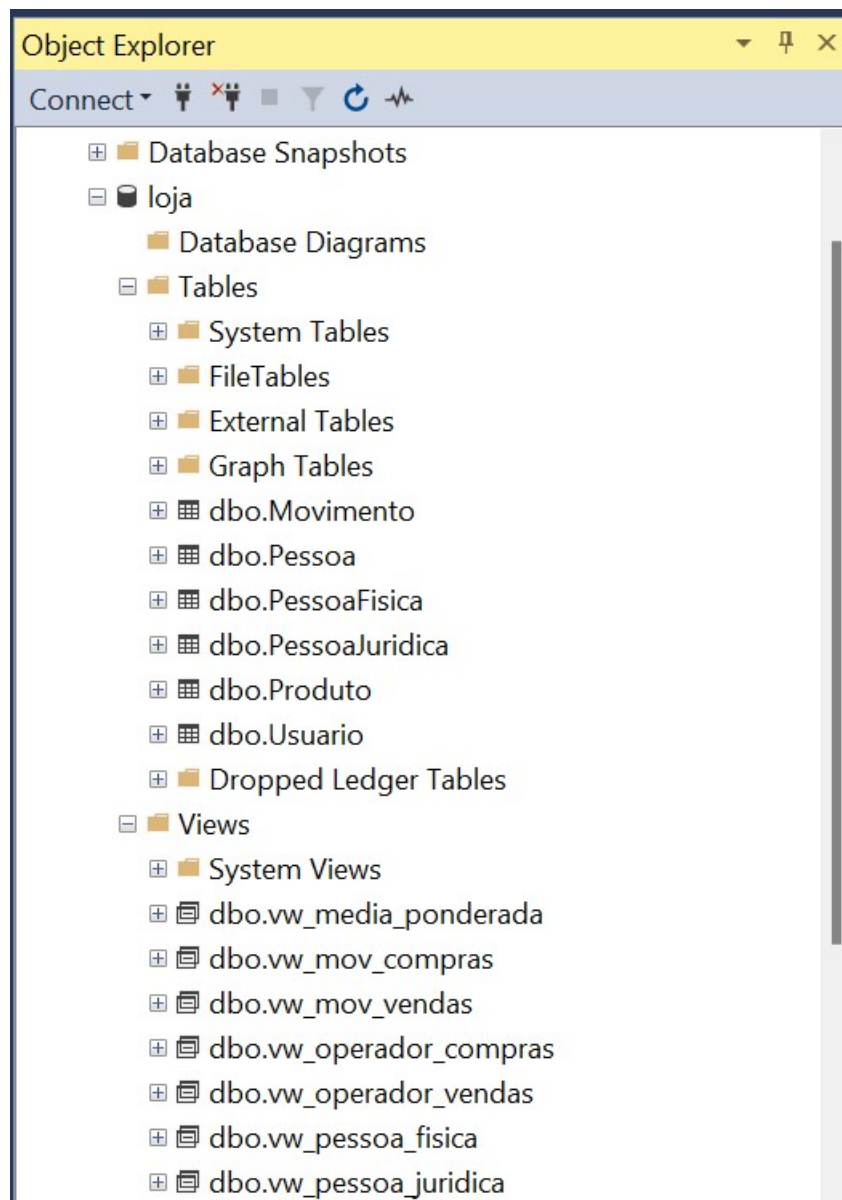
```
DROP VIEW vw_media_ponderada;  
GO
```

```
CREATE VIEW vw_media_ponderada AS  
SELECT  
    SUM(valorUnitario * quantidade) AS TotalVendas,  
    SUM(quantidade) AS TotalQuantidade,  
    CAST(FLOOR(SUM(valorUnitario * quantidade) * 100 /  
SUM(quantidade)) / 100 AS DECIMAL(10, 2)) AS MediaPonderada  
FROM  
    Movimento;
```

## **Resultados da execução dos códigos**







## **Análise e Conclusão:**

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

r)

**1X1** - Seria a associação de um registro de uma tabela no Maximo um de outra tabela pode se pensar em uma referencia entre chaves estrangeiras

ex: a relação de uma pessoa física a um cpf caso tivéssemos pessoa física em uma tabela e cpf em outra

**1XN** -Seria a associação de um registro de uma tabela a vários de outra tabela

ex: suponha um banco loja com uma tabela cliente relacionada compras onde teríamos a relação da chave primaria (PK) id cliente

relacionada a chave estrangeira (FK) id compras da tabela compras o que representaria que um cliente poderá realizar diversas compras.

**NxN** - Seria a associação de vários registro de uma tabela a vários de outra tabela e vice-versa.

ex: suponha um banco escola onde se tem uma tabela alunos relacionada a uma tabela cursos onde teremos aluno matriculado em vários cursos e cursos contendo vários alunos. Esse tipo de tabela é chamada de tabela de junção onde nelas se mantém como chaves estrangeiras a referencia á chaves primárias em outras tabelas.

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

r) três tipos

- Tabela única onde a chave seria primária.
- Tabela por classe concreta onde teríamos a representação da classe pai e filha(s) como sendo tabelas e nesse caso cada qual com sua chave primaria já que os campos herdados existiram nas tabelas filhas
- Tabela por hierarquia onde teríamos a tabela pai (base) como tendo a chave primaria que seria fornecida como chave estrangeira as tabela(s) filha(s) com relacionamento 1 X 1

obs: aqui no nosso trabalhos adotamos o tipo "tabela por hierarquia"

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

r) através de seus recursos como interface Gráfica, editor de consultas, ferramentas de análise de desempenho assim como boas praticas como a seleção de campos chave para criação de indexe(s)