A blue and black logo

Description automatically generated

|  |
| --- |
| Complementos de Bases de Dados 2023/2024  Licenciatura em Engª. Informática |
| 1ª Fase Relatório Técnico |

Turma: 2

Horário de Laboratório: 10:30 ao 12:30

Docente: João Portelinha

Grupo

Nº202100728, André Rolo  
Nº202100744, Tomás Gonçalves

# Introdução

*No presente relatório tem como objetivo explicar o que o grupo com dois elementos (André Rolo e Tomás Gonçalves) fizeram no projeto de fase 1 da Unidade Curricular de Complementos de Bases de Dados. Neste Projeto elaborámos com lógica a criação de novas tabelas a uma base de dados (usámos a migração de dados para facilitar a criação das novas tabelas e as suas relações), onde foi fornecida tabelas já criadas em Excel. Tivemos de aplicar os conhecimentos em aula. Uma das matérias que aplicámos foi os metadados e outros conhecimentos adquiridos em outras disciplinas relacionadas a criação de base de dados (Sistemas de Gestão de Bases de Dados).*

*Nós esperamos que este projeto ajude-nos a melhorar os nossos conhecimentos de criação de tabelas eficientes e com um sistema de segurança, para isso declaramos funções e procedimentos, também usamos as declaramos de views, with e trigger, onde iremos explicar com mais detalhes no decorrer do relatório. Para além disso nosso objetivo é melhorar a capacidade de trabalhar em equipe e aprendermos a resolver problemas que possam aparecer no mundo do trabalho.*

# Especificação de Requisitos

*Nós ao vermos o enunciado dado pelo professor, foi nos propostos fazer uma base de dados em que os produtos podiam ter categorias e subcategorias, onde fomos ter um discussão em grupo de decido-mos declarar uma tabela para categorias e outra para subcategorias. Como pode ver no modelo relacional e no modelo Entidade Relação*

|  |  |  |
| --- | --- | --- |
| ID | Descrição | Implementado  (S/N) |
| *R01* | *Este R01 é o mesmo que o MER mas com a diferença de ter mais especificado as chaves primarias e estrangeiras mostrando as suas colunas.* | *S* |
| *MER01* | *Criamos um sistema que irá permitir adicionar clientes , produtos dar categorias aos produtos e nas categorias estão a usar subcategorias, assim sendo mais fácil procurar por um produto existente no banco de dados. Também foi adicionado o sendEmail para enviar um email ao cliente e assim mostrando possíveis alterações na sua conta (Ex.: Password alterada)* | *S* |

# Modelo Relacional (*Modelo de dados*)

## Diagrama do Modelo Entidade Relação

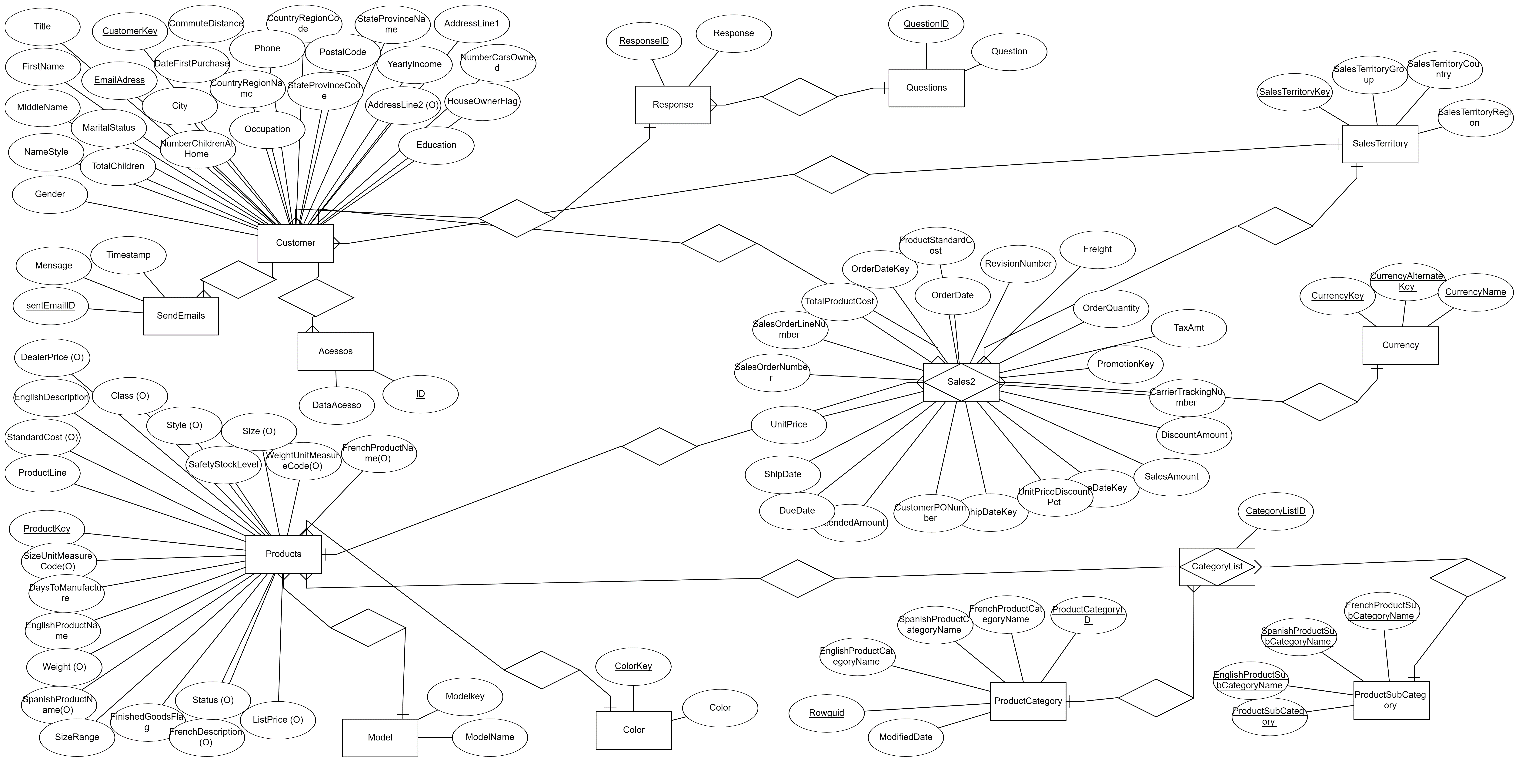
**

Figura - Diagrama de Modelo Entidade Relação

## Diagrama do Modelo Relacional

Uma imagem com texto, captura de ecrã, diagrama, Paralelo

Descrição gerada automaticamente

Figura - Modelo Relacional

# Definição do Layout

*No nosso Layout começamos por criar a possibilidade de ver os espaços de cada tabela dando a informação do número de linhas e espaço que foi reservado para a tabela e seu espaço usado, para além de mostrar os espaços que esta a ser ocupado pelos índices da tabela (são partes para melhorar o desempenho), e mostro ainda o espaço que ainda pode ser usado. Ainda fazemos um select em que mostra os nomes de todas as tabelas e exibe o tamanho máximo, número de registos e espaço total. Com base no select de cima geramos uma tabela temporária onde ira mostrar a taxa de crescimento da tabela e ainda colocamos um gatilho em cada tabela para não crascar para quando se faz insert, update ou delete numa tabela.*

## Identificação do espaço ocupado por tabela

|  |  |  |
| --- | --- | --- |
| Nome Tabela | Dimensão do Registo | Nº de Registos (inicial/final) |
| *Acessos* |  | *0* |
| *CategoryList* |  | *37* |
| *Currency* |  | 105 |
| *Customer* |  | 18484 |
| *ErrorLog* |  | 9 |
| *EstatisticasTabelas* |  | 18 |
| *MetadadosTabelas* |  | 139 |
| *Model* |  | 119 |
| *ProductCategory* |  | 4 |
| *Products* |  | 397 |
| *ProductSubCategory* |  | 37 |
| *Questions* |  | 2 |
| *Response* |  | 1 |
| *Sales2* |  | 6072 |
| *SalesTerritory* |  | 11 |
| *SentEmails* |  | 65 |
| *sysdiagrams* |  | 1 |

## Especificação dos FIlegroups

|  |  |  |
| --- | --- | --- |
| Nome Filegroup | Tabelas associadas | Parâmetros |
| *PRIMARY* | *Sales2, Customer, ErrorLog, EstatisticasTabelas, MetadadosTabelas, Model, ProductCategory, Products, ProductSubCategory, Questions, Response, SalesTerritory, SentEmails* | *Nome, Localização de Ficheiros* |

## Schemas

|  |  |
| --- | --- |
| Nome | Descrição |
| *sp\_spaceused* | *Usado para obter rapidamente informações resumidas sobre o espaço ocupado por uma tabela. Ele fornece dados como o número de linhas, espaço total alocado, espaço usado para dados e índices, e a diferença entre o espaço reservado e o espaço realmente utilizado* |
| *sys.tables* | *Fornece uma lista de todas as tabelas no banco de dados, incluindo detalhes como o nome da tabela, a data de criação, a última data de modificação, e outras propriedades relacionadas às tabelas* |
| *sys.indexes* | *oferece uma lista de todos os índices no banco de dados, incluindo detalhes como o nome do índice, o tipo de índice, a tabela à qual o índice pertence, e informações sobre a fragmentação do índice* |
| *sys.partitions* | *fornece informações sobre as partições de tabelas e índices, incluindo detalhes como o nome do objeto, a função da partição, o tipo de índice, o número de linhas na partição e informações sobre o armazenamento de dados* |
| *sys.allocation\_units* | *oferece informações resumidas sobre como o espaço físico é alocado para objetos, como tabelas e índices, no banco de dados. Ela inclui detalhes sobre o tipo de alocação, objeto associado, quantidade de espaço alocado e localização física* |
| *sys.data\_spaces* | *oferece informações resumidas sobre a organização de espaços de dados no banco. Fornece dados como o nome do espaço de dados, tipo (filegroup ou partition scheme), e local de armazenamento* |
| *sys.filegroups* | *fornece informações resumidas sobre os filegroups no banco de dados, incluindo detalhes como nome, tipo e localização física* |
| *sys.database\_files* | *fornece informações resumidas sobre os arquivos físicos de um banco de dados, incluindo detalhes como nome, tipo, tamanho e localização.* |
| *sys.columns* | *fornece informações resumidas sobre as colunas em tabelas e exibições, incluindo nome, tipo de dados e outras propriedades.* |
| *sys.default\_constraints* | *fornece informações sobre restrições padrão em colunas de tabelas, incluindo nome da restrição, tabela, coluna e o valor padrão.* |
| *sys.types* | *fornece informações resumidas sobre os tipos de dados disponíveis no banco de dados, incluindo tipos padrão e definidos pelo utilizador.* |

# Verificação da migração de dados

## Consultas sobre a base de dados original

## Consultas sobre a nova base de dados

# Programação

## Views

|  |  |
| --- | --- |
| Nome | Descrição |
| *CustomerDetailsView* | *Esta view permite obter a lista de clientes* |
| *CustomerPurchasesView* | *Esta View permite obter a lista de os produtos que os clientes compraram* |
| *HistoricoComprasCliente* | *Esta View permite obter a lista de todos os clientes que fizeram compras* |
| *VendasPorTerritorioView* | *Esta Views permite obter a lista de locais onde foram mais compradas* |

## Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Atributos | Requisito | Descrição |
| *dbo.udf\_getUtilizador* | *@id\_user INT* | *R0#* | *Permite obter informação detalhada sobre um utilizador* |
|  |  |  |  |

## Stored procedures

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Atributos | Requisito | Descrição |
| *dbo.MigrateCategoryListData* | *@nome VARCHAR(50)* | *R0#* | *Este procedimento dbo.MigrateCategoryListData move dados da tabela CategoryList de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateColorData* |  |  | *Este procedimento dbo.MigrateColorData move dados da tabela Color de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateCurrencyData* |  |  | *Este procedimento dbo.MigrateCurrencyData move dados da tabela Currency de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateCustomerData* |  |  | *Este procedimento dbo.MigrateCustomerData move dados da tabela Customer de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateModelData* |  |  | *Este procedimento dbo.MigrateModelData move dados da tabela Model de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateProductCategoryData* |  |  | *Este procedimento dbo.MigrateProductCategoryData move dados da tabela ProductCategory de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateProductsData* |  |  | *Este procedimento dbo.MigrateProductsData move dados da tabela Products de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateProductSubCategoryData* |  |  | *Este procedimento dbo.MigrateProductsSubCategoryData move dados da tabela ProductSubCategory de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.MigrateSalesData* |  |  | *Este procedimento dbo.MigrateSalesData move dados da tabela Sales2 de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.InsertSalesTerritoryData* |  |  | *Este procedimento dbo.InsertSalesTerritory move dados da tabela SalesTerritory de uma base de dados chamado AdventureOldData para a base de dados AdventureWorks.* |
| *dbo.CreateErrorTrigger* | *@TableName NVARCHAR(255)* |  | *O procedimento dbo.CreateErrorTrigger cria gatilhos de erro dinâmicos para tabelas específicas. Recebe o nome da tabela como parâmetro e gera um gatilho que é acionado após operações de inserção, atualização ou exclusão nessa tabela. Esse gatilho simula um erro (que pode ser substituído por lógica real de verificação de erro) e, se condições específicas forem atendidas, regista o erro em um log por meio de um procedimento chamado dbo.ErrorLog. Essa abordagem oferece flexibilidade na criação de gatilhos personalizados para diferentes tabelas da base de dados.* |
| *GetSalesByYear* |  |  | *Este procedimento armazenado chamado GetSalesByYear busca obter informações de vendas por ano. Ele seleciona o ano da data do pedido e a soma total dos valores de vendas da tabela Sales2, agrupando os resultados por ano. Se ocorrer um erro durante a execução da consulta, o bloco CATCH captura o erro e utiliza RAISERROR para sinalizar a ocorrência do erro, exibindo a mensagem de erro* |
| *GetSalesByProductCategory* |  |  | *Este procedimento GetSalesByProductCategory busca obter informações de vendas agregadas por categoria de produto. Ele realiza uma consulta que junta tabelas como Sales2, Products, CategoryList e ProductCategory para calcular a soma total dos valores de vendas (SalesAmount) para cada categoria de produto (EnglishProductCategoryName). Se ocorrer algum erro durante a execução da consulta, o bloco CATCH captura e sinaliza o erro, exibindo a mensagem de erro, sua severidade e estado.* |
| *GetSalesByProductSubCategory* |  |  | *Este procedimento GetSalesByProductSubCategory busca informações de vendas agregadas por subcategoria de produto. Ele realiza uma consulta que junta tabelas como Sales2, Products, CategoryList e ProductSubCategory para calcular a soma total dos valores de vendas (SalesAmount) para cada subcategoria de produto (EnglishProductSubCategoryName). Se algo der errado durante a execução da consulta, o bloco CATCH captura o erro e sinaliza, exibindo detalhes como mensagem de erro, severidade e estado.* |
| *GetProductsSoldByCategoryAndYear* |  |  | *Esse procedimento GetProductsSoldByCategoryAndYear busca informações sobre os produtos vendidos, agregando por categoria de produto e ano de venda. Realiza uma consulta complexa unindo tabelas como Sales2, Products, CategoryList e ProductCategory para contar o número de produtos vendidos por categoria e ano. Em caso de erro durante a execução da consulta, o bloco CATCH captura e sinaliza o erro, exibindo detalhes como a mensagem de erro, a severidade e o estado.* |

## Triggers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Tabela | Requisito | Descrição |
| *dbo.tr\_utlizador\_historico* | *AFTER UPDATE* | *dbo.utilizador* | *R0#* | *Guarda o histórico de alterações sobre o utilizador* |
|  |  |  |  |  |

# Catálogo/Metadados

## Monitorização

|  |  |  |
| --- | --- | --- |
| Nome | Atributos | Descrição |
| *dbo.MetadadosTabelas* | *TabelaID INT IDENTITY(1,1) PRIMARY KEY,*  *NomeTabela NVARCHAR(128),*  *NomeColuna NVARCHAR(128),*  *TipoDados NVARCHAR(128),*  *Tamanho INT,*  *Restricoes NVARCHAR(512),*  *DataAlteracao DATETIME* | *Cria uma tabela chamada MetadadosTabelas para armazenar informações sobre tabelas e colunas numa base de dados. Ele extrai dados das tabelas do sistema para preencher essa tabela com detalhes como nome da tabela, nome da coluna, tipo de dados, tamanho, restrições e data de alteração. É uma forma de registar informações sobre a estrutura das tabelas e colunas presentes no banco de dados.* |
| *dbo.MetadadosTabelas\_View* | *NomeTabela,*  *NomeColuna,*  *TipoDados,*  *Tamanho,*  *Restricoes,*  *DataAlteracao* | *Cria uma view chamada MetadadosTabelas\_View que seleciona informações específicas, como nome da tabela, nome da coluna, tipo de dados, tamanho, restrições e data da última alteração, da tabela MetadadosTabelas. A view retorna apenas os metadados mais recentes das tabelas, usando a informação da data de alteração mais atual disponível na tabela.* |
| *dbo.EstatisticasTabelas* | *TabelaID INT IDENTITY(1,1) PRIMARY KEY,*  *NomeTabela NVARCHAR(128),*  *NomeColuna NVARCHAR(128),*  NumRegistros INT,  EspacoOcupadoKB DECIMAL(10, 2),  DataRegistro DATETIME | *Cria uma tabela chamada EstatisticasTabelas para armazenar estatísticas sobre outras tabelas da base de dados. Em seguida, ele coleta informações sobre o número de registos e o espaço ocupado por cada tabela existente no banco de dados e insere esses dados na tabela EstatisticasTabelas* |

# Descrição da Demonstração

*Nesta trabalho nos tivemos de aplicar a migração de dados onde tínhamos uma base de dados delegada “AdventureOldData” onde tinha varias tabelas críticas (Currency, Customer, ProductSubCategory, Products, SalesTerritory e sales2), mas o objetivo principal fornecido pelo enunciado do projeto, que nos pedia para nós criarmos as melhorias na estrutura da base de dados “AdventureWorks”, Esta seria a base de dados que a empresa ia começar a usar.*

*Para alcançar este Objetivo tivemos de implementar procedimentos (PROCEDURE) para cada tabela. Cada procedimento tem como objetivo transferir os dados de forma segura e eficiente, assim fazendo com que os dados não sejam corrompidos ou perdidos, mas caso desse erro, esse erro é registado no “LogError” e assim mantendo os dados seguros. Esta mudança permitiu não apenas a movimentação dos dados, mas também fazer melhorias significativas na estrutura da base de dados.*

*Essa abordagem não apenas cumpriu os requisitos do projeto, mas também possibilitou a realização de melhorias substanciais na qualidade e integridade dos dados na nova base de dados. O facto de usar os procedimentos não só facilitou, mas também fez com que fosse mais fácil fazer manutenção e reaproveitamento do código.*

## Script de demonstração

*Com o decorrer do projeto, tivemos de escrever vários códigos de SQL, mas para isso tivemos de pensar em que tipo de estrutura utilizar, ou seja, foi a partir dai que começamos a criar um modelo de entidade relação e um modelo relacional para podermos basear em uma estrutura já pensada.*

*Começamos por criar os requisitos adicionais, que pedia no enunciado, mas para isso tínhamos de ter um campo de password ligado os clientes (Customer). Com este passo feito, focamo-nos em fazer o que enunciado nos pedia que era caso o user fizesse login ou pedisse uma nova password, nós tínhamos de criar um corpo de email e enviar-lhe o email com a nova password.*

*Para elaborar esta tarefa declaramos um novo procedimento onde tem o nome de ‘sp\_SendPasswordResetEmail’ este vai gerar uma nova password no e com essa fazer um corpo de email para enviar ao email do cliente, mas para isso ele vai colocar o email no “SendEmail”.*

[Imagem do code]

USE [AdventureWorks]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[MetadadosTabelas]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[MetadadosTabelas](

    [TabelaID] [int] IDENTITY(1,1) NOT NULL,

    [NomeTabela] [nvarchar](128) NULL,

    [NomeColuna] [nvarchar](128) NULL,

    [TipoDados] [nvarchar](128) NULL,

    [Tamanho] [int] NULL,

    [Restricoes] [nvarchar](512) NULL,

    [DataAlteracao] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

    [TabelaID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  View [dbo].[MetadadosTabelas\_View]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[MetadadosTabelas\_View]

AS

SELECT

    NomeTabela,

    NomeColuna,

    TipoDados,

    Tamanho,

    Restricoes,

    DataAlteracao

FROM dbo.MetadadosTabelas

WHERE DataAlteracao = (SELECT MAX(DataAlteracao) FROM dbo.MetadadosTabelas);

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Sales2]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Sales2](

    [ProductKey] [int] NOT NULL,

    [OrderDateKey] [int] NOT NULL,

    [DueDateKey] [int] NOT NULL,

    [ShipDateKey] [int] NOT NULL,

    [CustomerKey] [int] NOT NULL,

    [PromotionKey] [int] NOT NULL,

    [CurrencyKey] [int] NOT NULL,

    [SalesTerritoryKey] [int] NOT NULL,

    [SalesOrderNumber] [varchar](255) NOT NULL,

    [SalesOrderLineNumber] [int] NOT NULL,

    [RevisionNumber] [int] NOT NULL,

    [OrderQuantity] [int] NOT NULL,

    [UnitPrice] [int] NOT NULL,

    [ExtendedAmount] [int] NOT NULL,

    [UnitPriceDiscountPct] [int] NOT NULL,

    [DiscountAmount] [int] NOT NULL,

    [ProductStandardCost] [int] NOT NULL,

    [TotalProductCost] [int] NOT NULL,

    [SalesAmount] [int] NOT NULL,

    [TaxAmt] [int] NOT NULL,

    [Freight] [int] NOT NULL,

    [CarrierTrackingNumber] [nvarchar](255) NULL,

    [CustomerPONumber] [nvarchar](255) NULL,

    [OrderDate] [date] NOT NULL,

    [DueDate] [date] NOT NULL,

    [ShipDate] [date] NULL,

PRIMARY KEY CLUSTERED

(

    [ProductKey] ASC,

    [CustomerKey] ASC,

    [SalesTerritoryKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Products]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Products](

    [ProductKey] [int] IDENTITY(1,1) NOT NULL,

    [CategoryListID] [int] NOT NULL,

    [ColorKey] [int] NULL,

    [ModelKey] [int] NULL,

    [WeightUnitMeasureCode] [varchar](5) NULL,

    [SizeUnitMeasureCode] [varchar](5) NULL,

    [EnglishProductName] [varchar](255) NOT NULL,

    [SpanishProductName] [varchar](255) NULL,

    [FrenchProductName] [varchar](255) NULL,

    [StandardCost] [float] NULL,

    [FinishedGoodsFlag] [varchar](30) NOT NULL,

    [SafetyStockLevel] [int] NOT NULL,

    [ListPrice] [float] NULL,

    [Size] [varchar](55) NULL,

    [SizeRange] [varchar](55) NULL,

    [Weight] [float] NULL,

    [DaysToManufacture] [bit] NOT NULL,

    [ProductLine] [char](1) NULL,

    [DealerPrice] [float] NULL,

    [Class] [char](1) NULL,

    [Style] [char](1) NULL,

    [EnglishDescription] [varchar](255) NULL,

    [FrenchDescription] [varchar](max) NULL,

    [Status] [varchar](55) NULL,

PRIMARY KEY CLUSTERED

(

    [ProductKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [ProductKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Customer]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Customer](

    [CustomerKey] [int] IDENTITY(1,1) NOT NULL,

    [Title] [varchar](55) NULL,

    [FirstName] [varchar](40) NOT NULL,

    [MiddleName] [varchar](20) NULL,

    [LastName] [varchar](20) NULL,

    [EmailAddress] [varchar](80) NOT NULL,

    [Password] [varchar](60) NULL,

    [NameStyle] [varchar](20) NOT NULL,

    [BirthDate] [date] NOT NULL,

    [MaritalStatus] [char](1) NOT NULL,

    [Gender] [char](1) NOT NULL,

    [YearlyIncome] [int] NOT NULL,

    [TotalChildren] [int] NOT NULL,

    [NumberChildrenAtHome] [int] NOT NULL,

    [Education] [varchar](255) NOT NULL,

    [Occupation] [varchar](40) NOT NULL,

    [HouseOwnerFlag] [bit] NOT NULL,

    [NumberCarsOwned] [int] NOT NULL,

    [AddressLine1] [varchar](60) NOT NULL,

    [AddressLine2] [varchar](60) NULL,

    [City] [varchar](60) NOT NULL,

    [StateProvinceCode] [varchar](60) NULL,

    [StateProvinceName] [varchar](60) NULL,

    [CountryRegionCode] [varchar](6) NULL,

    [CountryRegionName] [varchar](20) NULL,

    [PostalCode] [nvarchar](55) NULL,

    [Phone] [varchar](55) NOT NULL,

    [DateFirstPurchase] [date] NOT NULL,

    [CommuteDistance] [varchar](30) NOT NULL,

    [SalesTerritoryKey] [int] NOT NULL,

    [ResponseID] [int] NULL,

PRIMARY KEY CLUSTERED

(

    [CustomerKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [CustomerKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  View [dbo].[HistoricoComprasCliente]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[HistoricoComprasCliente] AS

    SELECT

        S2.SalesOrderNumber,

        S2.OrderDate,

        P.EnglishProductName AS NomeDoProduto,

        P.ListPrice AS PrecoDoProduto,

        C.CustomerKey

    FROM Sales2 AS S2

    INNER JOIN Customer AS C ON S2.CustomerKey = C.CustomerKey

    INNER JOIN Products AS P ON S2.ProductKey = P.ProductKey;

GO

/\*\*\*\*\*\* Object:  View [dbo].[CustomerPurchasesView]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[CustomerPurchasesView] AS

    SELECT

        S2.SalesOrderNumber,

        S2.SalesOrderLineNumber,

        S2.OrderDate,

        S2.DueDate,

        S2.ShipDate,

        P.EnglishProductName AS ProductName,

        P.ListPrice AS ProductPrice,

        C.CustomerKey

    FROM Sales2 AS S2

    INNER JOIN Customer AS C ON S2.CustomerKey = C.CustomerKey

    INNER JOIN Products AS P ON S2.ProductKey = P.ProductKey;

GO

/\*\*\*\*\*\* Object:  View [dbo].[CustomerDetailsView]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[CustomerDetailsView] AS

    SELECT \*

    FROM Customer;

GO

/\*\*\*\*\*\* Object:  Table [dbo].[SalesTerritory]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SalesTerritory](

    [SalesTerritoryKey] [int] IDENTITY(1,1) NOT NULL,

    [SalesTerritoryRegion] [nvarchar](30) NOT NULL,

    [SalesTerritoryGroup] [nvarchar](30) NOT NULL,

    [SalesTerritoryCountry] [nvarchar](30) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [SalesTerritoryKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [SalesTerritoryKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  View [dbo].[VendasPorTerritorioView]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[VendasPorTerritorioView] AS

    SELECT

        ST.SalesTerritoryRegion AS RegiaoDeVendas,

        COUNT(S2.SalesOrderNumber) AS TotalDeVendas

    FROM Sales2 AS S2

    INNER JOIN SalesTerritory AS ST ON S2.SalesTerritoryKey = ST.SalesTerritoryKey

    GROUP BY ST.SalesTerritoryRegion;

GO

/\*\*\*\*\*\* Object:  View [dbo].[ProdutosEmEstoqueView]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[ProdutosEmEstoqueView] AS

    SELECT

        P.EnglishProductName AS NomeDoProduto,

        P.ListPrice AS PrecoDoProduto,

        P.StandardCost AS CustoDoProduto

    FROM Products AS P

    WHERE P.FinishedGoodsFlag = 'TRUE';

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Acessos]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Acessos](

    [ID] [int] IDENTITY(1,1) NOT NULL,

    [CustomerKey] [int] NULL,

    [DataAcesso] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

    [ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[CategoryList]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CategoryList](

    [CategoryListID] [int] IDENTITY(1,1) NOT NULL,

    [ProductCategoryKey] [int] NOT NULL,

    [ProductSubCategoryKey] [int] NULL,

PRIMARY KEY CLUSTERED

(

    [CategoryListID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Color]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Color](

    [ColorKey] [int] IDENTITY(1,1) NOT NULL,

    [Color] [nvarchar](55) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [ColorKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Currency]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Currency](

    [CurrencyKey] [int] IDENTITY(1,1) NOT NULL,

    [CurrencyAlternateKey] [nvarchar](5) NOT NULL,

    [CurrencyName] [nvarchar](55) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [CurrencyKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [CurrencyName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [CurrencyAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [CurrencyKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[ErrorLog]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ErrorLog](

    [ErrorLogID] [int] IDENTITY(1,1) NOT NULL,

    [ErrorMessage] [nvarchar](4000) NULL,

    [ErrorNumber] [int] NULL,

    [ErrorSeverity] [int] NULL,

    [ErrorState] [int] NULL,

    [Timestamp] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

    [ErrorLogID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[EstatisticasTabelas]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[EstatisticasTabelas](

    [TabelaID] [int] IDENTITY(1,1) NOT NULL,

    [NomeTabela] [nvarchar](128) NULL,

    [NumRegistros] [int] NULL,

    [EspacoOcupadoKB] [decimal](10, 2) NULL,

    [DataRegistro] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

    [TabelaID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Model]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Model](

    [ModelKey] [int] IDENTITY(1,1) NOT NULL,

    [ModelName] [varchar](255) NULL,

PRIMARY KEY CLUSTERED

(

    [ModelKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[ProductCategory]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ProductCategory](

    [ProductCategoryKey] [int] IDENTITY(1,1) NOT NULL,

    [EnglishProductCategoryName] [nvarchar](55) NULL,

    [FrenchProductCategoryName] [nvarchar](55) NOT NULL,

    [SpanishProductCategoryName] [nvarchar](55) NULL,

    [ModifiedDate] [date] NULL,

PRIMARY KEY CLUSTERED

(

    [ProductCategoryKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[ProductSubCategory]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ProductSubCategory](

    [ProductSubCategoryKey] [int] IDENTITY(1,1) NOT NULL,

    [EnglishProductSubCategoryName] [varchar](55) NOT NULL,

    [SpanishProductSubCategoryName] [varchar](55) NOT NULL,

    [FrenchProductSubCategoryName] [varchar](55) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [ProductSubCategoryKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [FrenchProductSubCategoryName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [SpanishProductSubCategoryName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [EnglishProductSubCategoryName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Questions]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Questions](

    [QuestionID] [int] IDENTITY(1,1) NOT NULL,

    [Question] [varchar](255) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [QuestionID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Response]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Response](

    [ResponseID] [int] IDENTITY(1,1) NOT NULL,

    [Response] [varchar](255) NOT NULL,

    [QuestionID] [int] NULL,

PRIMARY KEY CLUSTERED

(

    [ResponseID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[SentEmails]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SentEmails](

    [SentEmailID] [int] IDENTITY(1,1) NOT NULL,

    [Mensage] [nvarchar](max) NULL,

    [Timestamp] [datetime] NULL,

    [CustomerKey] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

    [SentEmailID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Acessos]  WITH CHECK ADD FOREIGN KEY([CustomerKey])

REFERENCES [dbo].[Customer] ([CustomerKey])

GO

ALTER TABLE [dbo].[CategoryList]  WITH CHECK ADD FOREIGN KEY([ProductCategoryKey])

REFERENCES [dbo].[ProductCategory] ([ProductCategoryKey])

GO

ALTER TABLE [dbo].[CategoryList]  WITH CHECK ADD FOREIGN KEY([ProductSubCategoryKey])

REFERENCES [dbo].[ProductSubCategory] ([ProductSubCategoryKey])

GO

ALTER TABLE [dbo].[Customer]  WITH CHECK ADD FOREIGN KEY([SalesTerritoryKey])

REFERENCES [dbo].[SalesTerritory] ([SalesTerritoryKey])

GO

ALTER TABLE [dbo].[Customer]  WITH CHECK ADD  CONSTRAINT [FK\_Customer\_Response] FOREIGN KEY([ResponseID])

REFERENCES [dbo].[Response] ([ResponseID])

GO

ALTER TABLE [dbo].[Customer] CHECK CONSTRAINT [FK\_Customer\_Response]

GO

ALTER TABLE [dbo].[Products]  WITH CHECK ADD FOREIGN KEY([CategoryListID])

REFERENCES [dbo].[CategoryList] ([CategoryListID])

GO

ALTER TABLE [dbo].[Products]  WITH CHECK ADD FOREIGN KEY([ColorKey])

REFERENCES [dbo].[Color] ([ColorKey])

GO

ALTER TABLE [dbo].[Products]  WITH CHECK ADD FOREIGN KEY([ModelKey])

REFERENCES [dbo].[Model] ([ModelKey])

GO

ALTER TABLE [dbo].[Response]  WITH CHECK ADD FOREIGN KEY([QuestionID])

REFERENCES [dbo].[Questions] ([QuestionID])

GO

ALTER TABLE [dbo].[Sales2]  WITH CHECK ADD FOREIGN KEY([CurrencyKey])

REFERENCES [dbo].[Currency] ([CurrencyKey])

GO

ALTER TABLE [dbo].[Sales2]  WITH CHECK ADD FOREIGN KEY([CustomerKey])

REFERENCES [dbo].[Customer] ([CustomerKey])

GO

ALTER TABLE [dbo].[Sales2]  WITH CHECK ADD FOREIGN KEY([ProductKey])

REFERENCES [dbo].[Products] ([ProductKey])

GO

ALTER TABLE [dbo].[Sales2]  WITH CHECK ADD FOREIGN KEY([SalesTerritoryKey])

REFERENCES [dbo].[SalesTerritory] ([SalesTerritoryKey])

GO

ALTER TABLE [dbo].[SentEmails]  WITH CHECK ADD FOREIGN KEY([CustomerKey])

REFERENCES [dbo].[Customer] ([CustomerKey])

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[AdicionarAcesso]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[AdicionarAcesso]

    @UserID INT

AS

BEGIN

    INSERT INTO Acessos (CustomerKey, DataAcesso)

    VALUES (@UserID, CURRENT\_TIMESTAMP);

    -- Verificar se a inserção foi bem-sucedida

    IF @@ROWCOUNT > 0

    BEGIN

        PRINT 'Acesso adicionado com sucesso.';

    END

    ELSE

    BEGIN

        PRINT 'Erro ao adicionar o acesso.';

    END

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[CreateErrorTrigger]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[CreateErrorTrigger]

    @TableName NVARCHAR(255)

AS

BEGIN

    DECLARE @TriggerScript NVARCHAR(MAX);

    -- Construir o script do gatilho dinâmico

   SET @TriggerScript = '

    CREATE TRIGGER Trg\_' + @TableName + '\_Error

    ON ' + @TableName + '

    AFTER INSERT, UPDATE, DELETE

    AS

    BEGIN

        -- Simular um erro (substitua por sua lógica de verificação de erro)

        IF (EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted))

        BEGIN

            DECLARE @ErrorMessage NVARCHAR(4000) = ''Erro simulado ocorreu na tabela ' + @TableName + ' durante uma operação de UPDATE.'';

            DECLARE @ErrorNumber INT = 50000;  -- Número de erro personalizado

            DECLARE @ErrorSeverity INT = 16;   -- Severidade do erro

            DECLARE @ErrorState INT = 1;       -- Estado do erro

            -- Chamar o procedimento armazenado para registrar o erro

            EXEC dbo.ErrorLog @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        END;

    END;

';

    -- Executar o script do gatilho dinâmico

    EXEC sp\_executesql @TriggerScript;

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[EditarAcessos]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[EditarAcessos]

    @UserID INT

AS

BEGIN

    UPDATE Acessos

    SET DataAcesso = CURRENT\_TIMESTAMP

    WHERE CustomerKey = @UserID;

    -- Verificar se a atualização foi bem-sucedida

    IF @@ROWCOUNT > 0

    BEGIN

        PRINT 'Acesso atualizado com sucesso.';

    END

    ELSE

    BEGIN

        PRINT 'Utilizador não encontrado ou acesso não atualizado.';

    END

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[GerarEntradasTabelaMetadados]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- Metadaos -> Stored Procedure para gerar entradas na tabela de metadados:

CREATE PROCEDURE [dbo].[GerarEntradasTabelaMetadados]

AS

BEGIN

    -- Crie uma tabela de metadados (se ainda n�o existir)

    IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'MetadadosTabelas')

    BEGIN

        CREATE TABLE dbo.MetadadosTabelas (

            TabelaID INT IDENTITY(1,1) PRIMARY KEY,

            NomeTabela NVARCHAR(128),

            NomeColuna NVARCHAR(128),

            TipoDados NVARCHAR(128),

            Tamanho INT,

            Restricoes NVARCHAR(512),

            DataAlteracao DATETIME

        );

    END

    -- Insira as informa��es das tabelas e colunas no cat�logo

    INSERT INTO dbo.MetadadosTabelas (NomeTabela, NomeColuna, TipoDados, Tamanho, Restricoes, DataAlteracao)

    SELECT

        t.name AS NomeTabela,

        c.name AS NomeColuna,

        ty.name AS TipoDados,

        c.max\_length AS Tamanho,

        dc.definition AS Restricoes,

        GETDATE() AS DataAlteracao

    FROM sys.tables t

    INNER JOIN sys.columns c ON t.object\_id = c.object\_id

    LEFT JOIN sys.default\_constraints dc ON c.default\_object\_id = dc.object\_id

    INNER JOIN sys.types ty ON c.user\_type\_id = ty.user\_type\_id;

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[GetProductsSoldByCategoryAndYear]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetProductsSoldByCategoryAndYear]

AS

BEGIN

    BEGIN TRY

        SELECT PC.EnglishProductCategoryName, YEAR(S2.OrderDate) AS Ano, COUNT(S2.ProductKey) AS NumeroDeProdutosVendidos

        FROM dbo.Sales2 S2

        INNER JOIN dbo.Products P ON S2.ProductKey = P.ProductKey

        INNER JOIN dbo.CategoryList CL ON P.CategoryListID = CL.CategoryListID

        INNER JOIN dbo.ProductCategory PC ON CL.ProductCategoryKey = PC.ProductCategoryKey

        GROUP BY PC.EnglishProductCategoryName, YEAR(S2.OrderDate)

        ORDER BY Ano, PC.EnglishProductCategoryName;

    END TRY

    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000);

        DECLARE @ErrorSeverity INT;

        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR\_MESSAGE(),

               @ErrorSeverity = ERROR\_SEVERITY(),

               @ErrorState = ERROR\_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

    END CATCH

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[GetSalesByProductCategory]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetSalesByProductCategory]

AS

BEGIN

    BEGIN TRY

        SELECT PC.EnglishProductCategoryName, SUM(S2.SalesAmount) AS TotalMonetarioDeVendas

        FROM dbo.Sales2 S2

        INNER JOIN dbo.Products P ON S2.ProductKey = P.ProductKey

        INNER JOIN dbo.CategoryList CL ON P.CategoryListID = CL.CategoryListID

        INNER JOIN dbo.ProductCategory PC ON CL.ProductCategoryKey = PC.ProductCategoryKey

        GROUP BY PC.EnglishProductCategoryName

        ORDER BY TotalMonetarioDeVendas DESC;

    END TRY

    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000);

        DECLARE @ErrorSeverity INT;

        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR\_MESSAGE(),

               @ErrorSeverity = ERROR\_SEVERITY(),

               @ErrorState = ERROR\_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

    END CATCH

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[GetSalesByProductSubCategory]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetSalesByProductSubCategory]

AS

BEGIN

    BEGIN TRY

        SELECT PSC.EnglishProductSubCategoryName, SUM(S2.SalesAmount) AS TotalMonetarioDeVendas

        FROM dbo.Sales2 S2

        INNER JOIN dbo.Products P ON S2.ProductKey = P.ProductKey

        INNER JOIN dbo.CategoryList CL ON P.CategoryListID = CL.CategoryListID

        INNER JOIN dbo.ProductSubCategory PSC ON CL.ProductSubCategoryKey = PSC.ProductSubCategoryKey

        GROUP BY PSC.EnglishProductSubCategoryName

        ORDER BY TotalMonetarioDeVendas DESC;

    END TRY

    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000);

        DECLARE @ErrorSeverity INT;

        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR\_MESSAGE(),

               @ErrorSeverity = ERROR\_SEVERITY(),

               @ErrorState = ERROR\_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

    END CATCH

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[GetSalesByYear]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[GetSalesByYear]

AS

BEGIN

    BEGIN TRY

        SELECT YEAR(S2.OrderDate) AS Ano, SUM(S2.SalesAmount) AS TotalMonetarioDeVendas

        FROM dbo.Sales2 S2

        GROUP BY YEAR(S2.OrderDate)

        ORDER BY Ano;

    END TRY

    BEGIN CATCH

        -- Captura de erro e tratamento

        DECLARE @ErrorMessage NVARCHAR(4000);

        DECLARE @ErrorSeverity INT;

        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR\_MESSAGE(),

               @ErrorSeverity = ERROR\_SEVERITY(),

               @ErrorState = ERROR\_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

    END CATCH

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertSalesTerritoryData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[InsertSalesTerritoryData]

AS

BEGIN

    BEGIN TRY

        INSERT INTO SalesTerritory (SalesTerritoryRegion, SalesTerritoryGroup, SalesTerritoryCountry)

        SELECT SalesTerritoryRegion, SalesTerritoryGroup, SalesTerritoryCountry

        FROM AdventureOldData.dbo.SalesTerritory;

    END TRY

    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se j� existir a stored procedure de log de erros)

        -- EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela SalesTerritory. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[LogError]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[LogError]

    @ErrorMessage NVARCHAR(4000),

    @ErrorNumber INT,

    @ErrorSeverity INT,

    @ErrorState INT

AS

BEGIN

    INSERT INTO ErrorLog (ErrorMessage, ErrorNumber, ErrorSeverity, ErrorState, Timestamp)

    VALUES (@ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState, GETDATE());

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateCategoryListData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateCategoryListData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transação para garantir consistência

        BEGIN TRANSACTION;

        -- Inserção de dados na tabela CategoryList

        INSERT INTO CategoryList (ProductCategoryKey, ProductSubCategoryKey)

        SELECT DISTINCT c.ProductCategoryKey, b.ProductSubCategoryKey

        FROM AdventureOldData.dbo.Products a

        INNER JOIN AdventureOldData.dbo.ProductSubCategory b ON a.ProductSubcategoryKey = b.ProductSubcategoryKey

        INNER JOIN AdventureWorks.dbo.ProductCategory c ON a.EnglishProductCategoryName COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS = c.EnglishProductCategoryName COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS;

        -- Commit da transação se a inserção for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transação em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amigável para o usuário

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela CategoryList. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateColorData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateColorData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transação para garantir consistência

        BEGIN TRANSACTION;

        -- Inserção de dados na tabela Color

        INSERT INTO Color (Color)

        SELECT DISTINCT Color

        FROM AdventureOldData.dbo.Products;

        -- Commit da transação se a inserção for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transação em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amigável para o usuário

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela Color. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateCurrencyData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateCurrencyData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transa��o para garantir consist�ncia

        BEGIN TRANSACTION;

        -- Inser��o de dados na tabela Currency

        INSERT INTO Currency (CurrencyAlternateKey, CurrencyName)

        SELECT CurrencyAlternateKey, CurrencyName

        FROM AdventureOldData.dbo.Currency;

        -- Commit da transa��o se a inser��o for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transa��o em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se tiver um)

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amig�vel para o usu�rio

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela Color. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateCustomerData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateCustomerData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transa��o para garantir consist�ncia

        BEGIN TRANSACTION;

        SET IDENTITY\_INSERT Customer ON;

        -- Inser��o de dados na tabela Customer

        INSERT INTO Customer (CustomerKey, Title, FirstName, MiddleName, LastName, EmailAddress, NameStyle, BirthDate, MaritalStatus, Gender, YearlyIncome, TotalChildren, NumberChildrenAtHome, Education, Occupation, HouseOwnerFlag, NumberCarsOwned, AddressLine1, AddressLine2, City, StateProvinceCode, StateProvinceName, CountryRegionCode, CountryRegionName, PostalCode, Phone, DateFirstPurchase, CommuteDistance, SalesTerritoryKey)

        SELECT CustomerKey, Title, FirstName, MiddleName, LastName, EmailAddress, NameStyle, BirthDate, MaritalStatus, Gender, YearlyIncome, TotalChildren, NumberChildrenAtHome, Education, Occupation, HouseOwnerFlag, NumberCarsOwned, AddressLine1, AddressLine2, City, StateProvinceCode, StateProvinceName, CountryRegionCode, CountryRegionName, PostalCode, Phone, DateFirstPurchase, CommuteDistance, SalesTerritoryKey

        FROM AdventureOldData.dbo.Customer;

        SET IDENTITY\_INSERT Customer OFF;

        -- Commit da transa��o se a inser��o for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transa��o em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se tiver um)

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amig�vel para o usu�rio

        THROW 50000, 'Ocorreu um erro durante a migra��o de dados da tabela Customer. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateModelData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateModelData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transação para garantir a consistência dos dados

        BEGIN TRANSACTION;

        -- Inserção de dados na tabela Model

        INSERT INTO Model (ModelName)

        SELECT DISTINCT ModelName

        FROM AdventureOldData.dbo.Products;

        -- Commit da transação se a inserção for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transação em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se disponível)

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amigável para o usuário

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela Model. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateProductCategoryData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateProductCategoryData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transação para garantir a consistência dos dados

        BEGIN TRANSACTION;

        -- Inserção de dados na tabela ProductCategory

        INSERT INTO ProductCategory (EnglishProductCategoryName, FrenchProductCategoryName, SpanishProductCategoryName)

        SELECT DISTINCT EnglishProductCategoryName, FrenchProductCategoryName, SpanishProductCategoryName

        FROM AdventureOldData.dbo.Products;

        -- Commit da transação se a inserção for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transação em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se disponível)

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amigável para o usuário

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela ProductCategory. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateProductsData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateProductsData]

AS

BEGIN

    BEGIN TRY

        SET IDENTITY\_INSERT Products ON;

        INSERT INTO Products (ProductKey, WeightUnitMeasureCode, SizeUnitMeasureCode, EnglishProductName, SpanishProductName, FrenchProductName, StandardCost, FinishedGoodsFlag,

            SafetyStockLevel, ListPrice, Size, SizeRange, Weight, DaysToManufacture, ProductLine, DealerPrice, Class, Style, EnglishDescription, FrenchDescription, Status, CategoryListID, ModelKey, ColorKey)

        SELECT DISTINCT p.ProductKey, p.WeightUnitMeasureCode, p.SizeUnitMeasureCode, p.EnglishProductName, p.SpanishProductName, p.FrenchProductName,

            p.StandardCost, p.FinishedGoodsFlag, p.SafetyStockLevel, p.ListPrice, p.Size, p.SizeRange, p.Weight, p.DaysToManufacture, p.ProductLine, p.DealerPrice,

            p.Class, p.Style, p.EnglishDescription, p.FrenchDescription, p.Status, d.CategoryListID, b.ModelKey, c.ColorKey

        FROM AdventureOldData.dbo.Products p

        INNER JOIN AdventureWorks.dbo.Color c ON p.Color = c.Color

        INNER JOIN AdventureWorks.dbo.Model b ON p.ModelName = b.ModelName

        INNER JOIN AdventureWorks.dbo.CategoryList d ON d.ProductSubCategoryKey = p.ProductSubcategoryKey;

        SET IDENTITY\_INSERT Products OFF;

    END TRY

    BEGIN CATCH

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amigável para o usuário

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela Products. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateProductSubCategoryData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateProductSubCategoryData]

AS

BEGIN

    BEGIN TRY

        -- Inicie a transa��o para garantir a consist�ncia dos dados

        BEGIN TRANSACTION;

        -- Inser��o de dados na tabela ProductSubCategory

        INSERT INTO ProductSubCategory (EnglishProductSubCategoryName, SpanishProductSubCategoryName, FrenchProductSubCategoryName)

        SELECT EnglishProductSubCategoryName, SpanishProductSubCategoryName, FrenchProductSubCategoryName

        FROM AdventureOldData.dbo.ProductSubCategory;

        -- Commit da transa��o se a inser��o for bem-sucedida

        COMMIT;

    END TRY

    BEGIN CATCH

        -- Rollback da transa��o em caso de erro

        ROLLBACK;

        -- Tratamento de erros

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        -- Registre o erro em um log de erros (se dispon�vel)

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        -- Exiba uma mensagem amig�vel para o usu�rio

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela ProductSubCategory. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[MigrateSalesData]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[MigrateSalesData]

AS

BEGIN

    BEGIN TRY

        INSERT INTO Sales2

        (

            ProductKey, OrderDateKey, DueDateKey, ShipDateKey, CustomerKey, PromotionKey, CurrencyKey, SalesTerritoryKey,

            SalesOrderNumber, SalesOrderLineNumber, RevisionNumber, OrderQuantity, UnitPrice, ExtendedAmount, UnitPriceDiscountPct,

            DiscountAmount, ProductStandardCost, TotalProductCost, SalesAmount, TaxAmt, Freight, CarrierTrackingNumber,

            CustomerPONumber, OrderDate, DueDate, ShipDate

        )

        SELECT

            ProductKey, OrderDateKey, DueDateKey, ShipDateKey, CustomerKey, PromotionKey, CurrencyKey, SalesTerritoryKey,

            SalesOrderNumber, SalesOrderLineNumber, RevisionNumber, OrderQuantity, UnitPrice, ExtendedAmount, UnitPriceDiscountPct,

            DiscountAmount, ProductStandardCost, TotalProductCost, SalesAmount, TaxAmt, Freight, CarrierTrackingNumber,

            CustomerPONumber, OrderDate, DueDate, ShipDate

        FROM AdventureOldData.dbo.sales2;

    END TRY

    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

        DECLARE @ErrorNumber INT = ERROR\_NUMBER();

        DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

        DECLARE @ErrorState INT = ERROR\_STATE();

        EXEC dbo.LogError @ErrorMessage, @ErrorNumber, @ErrorSeverity, @ErrorState;

        THROW 50000, 'Ocorreu um erro durante a migração de dados da tabela Sales2. Entre em contato com o suporte.', 1;

    END CATCH

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[RegistrarEstatisticasTabelas]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[RegistrarEstatisticasTabelas]

AS

BEGIN

    -- Crie uma tabela de estatísticas (se ainda não existir)

    IF NOT EXISTS (SELECT 1 FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'EstatisticasTabelas')

    BEGIN

        CREATE TABLE dbo.EstatisticasTabelas (

            TabelaID INT IDENTITY(1,1) PRIMARY KEY,

            NomeTabela NVARCHAR(128),

            NumRegistros INT,

            EspacoOcupadoKB DECIMAL(10, 2),

            DataRegistro DATETIME

        );

    END

    -- Insira as estatísticas para cada tabela

    INSERT INTO dbo.EstatisticasTabelas (NomeTabela, NumRegistros, EspacoOcupadoKB, DataRegistro)

    SELECT

        t.name AS NomeTabela,

        SUM(p.rows) AS NumRegistros,

        SUM(a.total\_pages) \* 8 / 1024 AS EspacoOcupadoKB,

        GETDATE() AS DataRegistro

    FROM sys.tables t

    INNER JOIN sys.indexes i ON t.object\_id = i.object\_id

    INNER JOIN sys.partitions p ON i.object\_id = p.object\_id AND i.index\_id = p.index\_id

    INNER JOIN sys.allocation\_units a ON p.partition\_id = a.container\_id

    GROUP BY t.name;

END;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[RemoverAcesso]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[RemoverAcesso]

    @UserID INT

AS

BEGIN

    DELETE FROM Acessos

    WHERE CustomerKey = @UserID;

    -- Verificar se a exclusão foi bem-sucedida

    IF @@ROWCOUNT > 0

    BEGIN

        PRINT 'Acesso removido com sucesso.';

    END

    ELSE

    BEGIN

        PRINT 'Erro ao remover o acesso. Utilizador não encontrado.';

    END

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[sp\_SendPasswordResetEmail]    Script Date: 15/11/2023 23:59:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- Criar uma stored procedure fictícia para enviar e-mails

CREATE PROCEDURE [dbo].[sp\_SendPasswordResetEmail]

  @RecipientEmail NVARCHAR(80),

  @EmailSubject NVARCHAR(255),

  @EmailBody NVARCHAR(MAX),

  @CustomerKey INT,

  @NewPassword NVARCHAR(60),

  @SecurityQuestion NVARCHAR(255),

  @SecurityAnswer NVARCHAR(255)

AS

BEGIN

  -- Simular o envio do e-mail (este é um exemplo fictício)

  PRINT 'Simulating email sending to ' + @RecipientEmail;

  PRINT 'Subject: ' + @EmailSubject;

  PRINT 'Body: ' + @EmailBody;

  -- Insira os dados na tabela SentEmails

  INSERT INTO SentEmails (Mensage, Timestamp, CustomerKey)

  VALUES (@EmailBody, GETDATE(), @CustomerKey);

  -- Verifica se a simulação foi bem-sucedida antes de atualizar a senha

  IF @@ROWCOUNT > 0

  BEGIN

    -- Verifica a resposta à pergunta de segurança usando a tabela Response

    IF EXISTS (

      SELECT 1

      FROM Response AS r

      JOIN Questions AS q ON r.QuestionID = q.QuestionID

      WHERE r.Response = @SecurityAnswer

        AND q.Question = @SecurityQuestion

    )

    BEGIN

      -- Simular uma atualização de senha na tabela Customer

      UPDATE Customer

      SET Password = @NewPassword

      WHERE CustomerKey = @CustomerKey;

      PRINT 'Senha atualizada com sucesso!';

    END

    ELSE

    BEGIN

      PRINT 'Resposta à pergunta de segurança incorreta. A senha não foi alterada.';

    END

  END

  ELSE

  BEGIN

    PRINT 'Falha ao enviar o e-mail. A senha não foi alterada.';

  END

END

-- Executar o procedimento com um @RecipientEmail, @EmailSubject, @EmailBody, @CustomerKey, @NewPassword, @SecurityQuestion e @SecurityAnswer

EXEC sp\_SendPasswordResetEmail 'jon24@adventure-works.com', 'Recuperação de senha', 'Nova Senha: 1234567890', 11000, '1234567890', 'Qual é o nome do teu cão?', 'Billy';

GO

# Conclusões