Scripts

Script for setting up our mssql database:

```
USE [master]
GO
/****** Object:  Database [Exam]    Script Date: 26-05-2023 11:43:07 ******/
CREATE DATABASE [Exam]
use Exam
GO
CREATE TABLE [dbo].[OrderLines](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [orderId] [int] NULL,
    [productName] [nchar](25) NULL,
    [productPrice] [float] NULL,
    [quantity] [int] NULL,
 CONSTRAINT [PK_OrderLines] 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].[Orders]    Script Date: 26-05-2023 11:43:07 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Orders](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [userId] [int] NULL,
    [created] [date] NULL,
    [status] [bit] NULL,
    [total] [float] NULL,
    [billingAddress] [nchar](30) NULL,
    [billingCity] [nchar](30) NULL,
    [postalCode] [int] NULL,
 CONSTRAINT [PK_Orders] 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].[Users]    Script Date: 26-05-2023 11:43:07 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [username] [nchar](20) NOT NULL,
    [password] [nchar](30) NOT NULL,
    [firstName] [nchar](20) NOT NULL,
    [lastName] [nchar](20) NOT NULL,
    [age] [int] NOT NULL,
    [email] [nchar](50) NOT NULL,
    [phone] [int] NULL,
    [admin] [bit] NOT NULL,
    [orderId] [int] NOT NULL,
 CONSTRAINT [PK_Users] 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
ALTER TABLE [dbo].[OrderLines]  WITH CHECK ADD  CONSTRAINT [FK_OrderLines_Orders] FOREIGN
KEY([orderId])
REFERENCES [dbo].[Orders] ([id])
GO
ALTER TABLE [dbo].[OrderLines] CHECK CONSTRAINT [FK_OrderLines_Orders]
GO
ALTER TABLE [dbo].[Orders]  WITH CHECK ADD  CONSTRAINT [FK_Orders_Users] FOREIGN KEY([userId])
REFERENCES [dbo].[Users] ([id])
GO
ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK_Orders_Users]
GO
```

```
USE [master]
GO
ALTER DATABASE [Exam] SET  READ_WRITE
GO
```

User defined table type:

```
USE [DBExam]
CREATE TYPE typeProduct AS TABLE
(
    productId INT,
    productName VARCHAR(35),
    productPrice FLOAT,
    quantity INT
)
```

Stored procedure

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[create_order]
@tblProducts typeProduct READONLY,
@userId INT,
@total FLOAT,
@billingAddress VARCHAR(35),
@billingCity VARCHAR(35),
@postalCode INT

AS
DECLARE @numberOfTypeProduct int, @orderId INT, @rowCount int;
DECLARE @typeProductWithRowId TABLE (rowId INT, productId INT, productName VARCHAR(35),
productPrice FLOAT, quantity INT)

DECLARE @rowId INT, @productId INT, @productName VARCHAR(35), @productPrice FLOAT, @quantity
INT;
BEGIN TRY
   BEGIN TRAN
      SET NOCOUNT ON;
```

```
    INSERT INTO [dbo].Orders([userId], [created], [status], [total], [billingAddress], [billingCity], [postalCode])
    VALUES (@userId, GETDATE(), 1, @total, @billingAddress, @billingCity, @postalCode)
    SET @orderId = SCOPE_IDENTITY();

    BEGIN
      INSERT INTO @typeProductWithRowId (rowId,productId, productName, productPrice, quantity)
      SELECT ROW_NUMBER() OVER(ORDER BY productId) rowId, * FROM @tblProducts;
    END

    SET @numberOfTypeProduct = (SELECT count(*) FROM @tblProducts);
    SET @rowCount = 1;
    WHILE @rowCount <= @numberOfTypeProduct
      BEGIN
        (SELECT @productId = productId, @productName = productName, @productPrice = productPrice,
@quantity = quantity FROM @typeProductWithRowId WHERE rowId = @rowCount)
        INSERT INTO dbo.OrderLines([orderId], [productId], [productName], [productPrice], [quantity])
        VALUES (@orderId, @productId,@productName, @productPrice, @quantity)
        SET @rowCount = @rowCount + 1;
      END
  COMMIT TRAN
END TRY

BEGIN CATCH
  ROLLBACK TRAN
END CATCH
GO
```

1. What the stored prodcedure does:
   The stored procedure is named [dbo].[create_order].
2. It has parameters @tblProducts of type typeProduct (assumed to be a user-defined table type), @userId of type INT, @total of type FLOAT, @billingAddress of type VARCHAR(35), @billingCity of type VARCHAR(35), and @postalCode of type INT.
3. The stored procedure begins with setting the necessary options (SET ANSI_NULLS ON and SET QUOTED_IDENTIFIER ON).

4. It declares variables @numberOfTypeProduct, @orderId, @rowCount, and @typeProductWithRowId.
5. The BEGIN TRY block begins a transaction (BEGIN TRAN) and sets NOCOUNT ON.
6. It inserts a new record into the [dbo].Orders table with the provided @userId, current timestamp (GETDATE()), status 1, and other details, and assigns the SCOPE_IDENTITY() to @orderId.
7. It populates the @typeProductWithRowId table variable by assigning a row number to each row of @tblProducts.
8. It retrieves the count of rows in @tblProducts and assigns it to @numberOfTypeProduct.
9. It initiates a loop that iterates over each row in @typeProductWithRowId.
10. Within the loop, it selects the corresponding values from @typeProductWithRowId into variables @productId, @productName, @productPrice, and @quantity.
11. It inserts a new record into the dbo.OrderLines table for each iteration, using the @orderId, @productId, @productName, @productPrice, and @quantity.
12. The transaction is committed (COMMIT TRAN) if there were no errors in the try block.
13. If an error occurs, the BEGIN CATCH block rolls back the transaction (ROLLBACK TRAN).
14. The stored procedure ends with the END CATCH and GO statements.

Test data to test our stored procedure instead of our application:

```
DECLARE @tblProducts dbo.typeProduct;
DECLARE @userId INT, @total FLOAT, @billingAddress VARCHAR(35), @billingCity VARCHAR(35),
@postalCode INT;

SET @userId = 1;
SET @total = 250;
SET @billingAddress = Lyngbyhovedgade;
SET @billingCity = 'Lyngby';
SET @postalCode = 2800;

INSERT INTO @tblProducts ([productId], [productName], [productPrice], [quantity])
VALUES (1, 'test', 200, 1);

EXEC [dbo].[create_order]
   @tblProducts = @tblProducts,
   @userId = @userId,
   @total = @total,
   @billingAddress = @billingAddress,
   @billingCity = @billingCity,
```

```
@postalCode = @postalCode;
```