Systemy Baz Danych

Projekt Semestralny

Autor: Damian Wyrębiak

Spis treści

[1. Wybrane Tematy 3](#_Toc200353936)

[1.1. System zarządzania zamówieniami klientów (Temat nr 1) 3](#_Toc200353937)

[1.2. Analiza sprzedaży w wybranym okresie (Temat nr 3) 3](#_Toc200353938)

[1.3. System rejestracji nowych produktów (Temat nr 5) 3](#_Toc200353939)

[1.4. Zarządzanie kategoriami produktów (Temat dodatkowy) 3](#_Toc200353940)

[2. Kody Procedur i ich Wykorzystanie w C# 4](#_Toc200353941)

[2.1. System Zarządzania Zamówieniami Klientów 4](#_Toc200353942)

[2.2. Analiza Sprzedaży w Wybranym Okresie 5](#_Toc200353943)

[2.2.1. Znajdowanie Nazwisk 5](#_Toc200353944)

[2.2.2. Znajdowanie Imion 7](#_Toc200353945)

[2.2.3. Znajdowanie Zamówień po Filtrach 9](#_Toc200353946)

[2.2.4. Znajdowanie konkretnego zamówienia 13](#_Toc200353947)

[2.2.5. Lista Produktów Zamówienia 15](#_Toc200353948)

[2.3. System Rejestracji Nowych Produktów 17](#_Toc200353949)

[2.3.1. Jednostki Miar 17](#_Toc200353950)

[2.3.2. Produkty Pasujące do Podkategorii lub Kategorii 19](#_Toc200353951)

[2.3.3. Wyszukiwanie Produktu po Nazwie 22](#_Toc200353952)

[2.3.4. Dodawanie Nowego Produktu 24](#_Toc200353953)

[2.4. Zarządzanie Kategoriami Produktów 27](#_Toc200353954)

[2.4.1. Wyszukanie Wszystkich Kategorii 27](#_Toc200353955)

[2.4.2. Wyszukanie Kategorii po Nazwie 28](#_Toc200353956)

[2.4.3. Dodanie Kategorii 29](#_Toc200353957)

[2.4.4. Aktualizowanie Kategorii 31](#_Toc200353958)

[2.4.5. Usuwanie Kategorii 33](#_Toc200353959)

[2.4.6. Wyszukiwanie Wszystkich Podkategorii Danej Kategorii 35](#_Toc200353960)

[2.4.7. Wyszukanie Podkategorii po Nazwie 37](#_Toc200353961)

[2.4.8. Dodawanie Podkategorii 39](#_Toc200353962)

[2.4.9. Aktualizowanie Podkategorii 41](#_Toc200353963)

[2.4.10. Usuwanie Podkategorii 43](#_Toc200353964)

# Wybrane Tematy

## System zarządzania zamówieniami klientów (Temat nr 1)

Opis:  
Aplikacja umożliwiająca tworzenie, modyfikowanie i anulowanie

Zakres:  
Transakcje, procedury składowane, INSERT/UPDATE/DELETE, obsługa błędów, JOIN-y, funkcje daty i czasu.

## Analiza sprzedaży w wybranym okresie (Temat nr 3)

Opis:  
Aplikacja analizująca dane sprzedaży według różnych kryteriów.

Zakres:  
Funkcje agregujące, podzapytania, JOIN-y, zmienne, procedury z parametrami.

## System rejestracji nowych produktów (Temat nr 5)

Opis:  
Aplikacja umożliwiająca dodawanie nowych produktów z walidacją danych.

Zakres:  
INSERT, wyzwalacze INSTEAD OF, procedury składowane, typy danych.

## Zarządzanie kategoriami produktów (Temat dodatkowy)

Opis:  
Moduł umożliwiający tworzenie, edycję i usuwanie kategorii produktów, z kontrolą powiązań z produktami.

Zakres:  
Procedury składowane, relacje z kluczem obcym, obsługa błędów (np. przy próbie usunięcia kategorii przypisanej do produktu), podstawowe transakcje, walidacja danych wejściowych.

# Kody Procedur i ich Wykorzystanie w C#

## System Zarządzania Zamówieniami Klientów

Brak

## Analiza Sprzedaży w Wybranym Okresie

### Znajdowanie Nazwisk

CREATE OR ALTER PROCEDURE dbo.uspGetPeopleLastNames

AS

BEGIN

SELECT LastName AS PersonLastName

FROM Person.Person

GROUP BY LastName

ORDER BY LastName;

END

GO

Procedura nie przyjmuje żadnych parametrów, a zwraca wszystkie nazwiska osób. Powinno się tu użyć SELECT DISCTINCT zamiast GROUP BY.

Wykorzystanie:

ComboBoxLastName.Items.Clear();

ComboBoxLastName.Items.Add("");

{

using var cmd = new SqlCommand("dbo.uspGetPeopleLastNames", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ComboBoxLastName.Items.Add($"{reader["PersonLastName"]}");

}

}

ComboBoxLastName.SelectedIndex = 0;

### Znajdowanie Imion

CREATE OR ALTER PROCEDURE dbo.uspGetPeopleFirstNamesByLastName

(

@LastName NVARCHAR(50) = NULL

)

AS

BEGIN

IF @LastName IS NULL

BEGIN

SELECT FirstName AS PersonFirstName

FROM Person.Person

GROUP BY FirstName;

END

ELSE

BEGIN

SELECT FirstName AS PersonFirstName

FROM Person.Person

WHERE LastName = @LastName

GROUP BY FirstName;

END

END

GO

Procedura przyjmuje nazwisko i znajduje wszystkie imiona osób o podanym nazwisku. Jeśli nazwisko nie zostało podane, zwraca listę wszystkich imion. Ponownie, powinno tu być użyte SELECT DISCTINCT zamiast GROUP BY.

Wykorzystanie:

string lastName = ComboBoxLastName.Text;

ComboBoxFirstName.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetPeopleFirstNamesByLastName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@LastName", lastName);

using var reader = cmd.ExecuteReader();

if (!reader.HasRows)

{

ComboBoxFirstName.Items.Add("");

ComboBoxFirstName.Enabled = false;

}

else while (reader.Read())

{

ComboBoxFirstName.Items.Add($"{reader["PersonFirstName"]}");

ComboBoxFirstName.Enabled = true;

}

}

ComboBoxFirstName.SelectedIndex = 0;

### Znajdowanie Zamówień po Filtrach

CREATE OR ALTER PROCEDURE dbo.uspGetOrders

(

@OrderDateFrom DATETIME = NULL,

@OrderDateTo DATETIME = NULL,

@ProductCategoryName NVARCHAR(50) = 'All',

@ProductSubcategoryName NVARCHAR(50) = 'All',

@ProductName NVARCHAR(50) = 'All',

@PersonLastName NVARCHAR(50) = NULL,

@PersonFirstName NVARCHAR(50) = NULL

)

AS

BEGIN

IF @OrderDateFrom IS NULL

SET @OrderDateFrom = SYSDATETIME()

IF @OrderDateTo IS NULL

SET @OrderDateTo = SYSDATETIME()

IF @PersonLastName IS NULL

SET @PersonLastName = '%'

IF @PersonFirstName IS NULL

SET @PersonFirstName = '%'

IF @ProductName = 'All'

BEGIN

IF @ProductSubcategoryName = 'All'

BEGIN

IF @ProductCategoryName = 'All'

BEGIN

SELECT DISTINCT

head.SalesOrderNumber AS OrderNumber,

head.PurchaseOrderNumber AS PurchaseOrderNumber,

head.OrderDate AS OrderDate

FROM

Sales.SalesOrderDetail det

INNER JOIN Sales.SalesOrderHeader head

ON det.SalesOrderID = head.SalesOrderID

INNER JOIN Sales.Customer cust

ON head.CustomerID = cust.CustomerID

INNER JOIN Person.Person per

ON cust.PersonID = per.BusinessEntityID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

INNER JOIN Production.ProductSubcategory subcat

ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

INNER JOIN Production.ProductCategory cat

ON subcat.ProductCategoryID = cat.ProductCategoryID

WHERE

head.OrderDate BETWEEN @OrderDateFrom AND @OrderDateTo

AND per.LastName LIKE @PersonLastName

AND per.FirstName LIKE @PersonFirstName

ORDER BY

head.OrderDate;

END

ELSE

BEGIN

SELECT DISTINCT

head.SalesOrderNumber AS OrderNumber,

head.PurchaseOrderNumber AS PurchaseOrderNumber,

head.OrderDate AS OrderDate

FROM

Sales.SalesOrderDetail det

INNER JOIN Sales.SalesOrderHeader head

ON det.SalesOrderID = head.SalesOrderID

INNER JOIN Sales.Customer cust

ON head.CustomerID = cust.CustomerID

INNER JOIN Person.Person per

ON cust.PersonID = per.BusinessEntityID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

INNER JOIN Production.ProductSubcategory subcat

ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

INNER JOIN Production.ProductCategory cat

ON subcat.ProductCategoryID = cat.ProductCategoryID

WHERE

head.OrderDate BETWEEN @OrderDateFrom AND @OrderDateTo

AND per.LastName LIKE @PersonLastName

AND per.FirstName LIKE @PersonFirstName

AND cat.Name = @ProductCategoryName

ORDER BY

head.OrderDate;

END

END

ELSE

BEGIN

SELECT DISTINCT

head.SalesOrderNumber AS OrderNumber,

head.PurchaseOrderNumber AS PurchaseOrderNumber,

head.OrderDate AS OrderDate

FROM

Sales.SalesOrderDetail det

INNER JOIN Sales.SalesOrderHeader head

ON det.SalesOrderID = head.SalesOrderID

INNER JOIN Sales.Customer cust

ON head.CustomerID = cust.CustomerID

INNER JOIN Person.Person per

ON cust.PersonID = per.BusinessEntityID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

INNER JOIN Production.ProductSubcategory subcat

ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

INNER JOIN Production.ProductCategory cat

ON subcat.ProductCategoryID = cat.ProductCategoryID

WHERE

head.OrderDate BETWEEN @OrderDateFrom AND @OrderDateTo

AND per.LastName LIKE @PersonLastName

AND per.FirstName LIKE @PersonFirstName

AND subcat.Name = @ProductSubcategoryName

ORDER BY

head.OrderDate;

END

END

ELSE

BEGIN

SELECT DISTINCT

head.SalesOrderNumber AS OrderNumber,

head.PurchaseOrderNumber AS PurchaseOrderNumber,

head.OrderDate AS OrderDate

FROM

Sales.SalesOrderDetail det

INNER JOIN Sales.SalesOrderHeader head

ON det.SalesOrderID = head.SalesOrderID

INNER JOIN Sales.Customer cust

ON head.CustomerID = cust.CustomerID

INNER JOIN Person.Person per

ON cust.PersonID = per.BusinessEntityID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

INNER JOIN Production.ProductSubcategory subcat

ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

INNER JOIN Production.ProductCategory cat

ON subcat.ProductCategoryID = cat.ProductCategoryID

WHERE

head.OrderDate BETWEEN @OrderDateFrom AND @OrderDateTo

AND per.LastName LIKE @PersonLastName

AND per.FirstName LIKE @PersonFirstName

AND prod.Name = @ProductName

ORDER BY

head.OrderDate;

END

END

GO

Procedura przyjmuje przedział dat złożenia zamówienia, nazwy kategorii, podkategorii, oraz nazwiska i imiona osób, które złożyły dane zamówienie.

Posiada cztery zapytania, które różnią się wyłącznie mankamentami w klauzuli WHERE, w zależności od tego, które parametry zostały podane, a które są domyślne.

Wykorzystanie:

DateTime orderDateFrom = DateTimePickerFrom.Value;

DateTime orderDateTo = DateTimePickerTo.Value;

string categoryName = ComboBoxCategories.Text;

string subcategoryName = ComboBoxSubcategories.Text;

string productName = ComboBoxProducts.Text;

string? personLastName = ComboBoxLastName.Text.IsNullOrEmpty() ? null : ComboBoxLastName.Text;

string? personFirstName = ComboBoxFirstName.Text.IsNullOrEmpty() ? null : ComboBoxFirstName.Text;

ListBoxOrders.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetOrders", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@OrderDateFrom", orderDateFrom);

cmd.Parameters.AddWithValue("@OrderDateTo", orderDateTo);

cmd.Parameters.AddWithValue("@ProductCategoryName", categoryName);

cmd.Parameters.AddWithValue("@ProductSubcategoryName", subcategoryName);

cmd.Parameters.AddWithValue("@ProductName", productName);

cmd.Parameters.AddWithValue("@PersonLastName", personLastName);

cmd.Parameters.AddWithValue("@PersonFirstName", personFirstName);

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ListBoxOrders.Items.Add($"{reader["OrderNumber"]} --- {reader["PurchaseOrderNumber"]} --- {reader["OrderDate"]}");

}

}

### Znajdowanie konkretnego zamówienia

CREATE OR ALTER PROCEDURE dbo.uspGetOrderDetails

(

@OrderNumber NVARCHAR(25)

)

AS

BEGIN

SELECT

head.SalesOrderNumber AS OrderNumber,

head.OrderDate AS OrderDate,

head.DueDate AS DueDate,

head.OnlineOrderFlag AS OnlineOrderFlag,

head.PurchaseOrderNumber AS PurchaseNumber,

CONCAT(per.FirstName, ' ', per.LastName) AS CustomerName,

billaddr.AddressLine1 AS BillingAddress,

billaddr.AddressLine2 AS BillingAddressL2,

billaddr.City AS BillingCityName,

billstat.Name AS BillingStateName,

billcountry.Name AS BillingCountryName,

shipaddr.AddressLine1 AS ShippingAddress,

shipaddr.AddressLine2 AS ShippingAddressL2,

shipaddr.City AS ShippingCityName,

shipstat.Name AS ShippingStateName,

shipcountry.Name AS ShippingCountryName,

head.TotalDue AS TotalDue

FROM

Sales.SalesOrderDetail det

INNER JOIN Sales.SalesOrderHeader head

ON det.SalesOrderID = head.SalesOrderID

INNER JOIN Sales.Customer cust

ON head.CustomerID = cust.CustomerID

INNER JOIN Person.Person per

ON cust.PersonID = per.BusinessEntityID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

INNER JOIN Production.ProductSubcategory subcat

ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

INNER JOIN Production.ProductCategory cat

ON subcat.ProductCategoryID = cat.ProductCategoryID

INNER JOIN Person.Address billaddr

ON head.BillToAddressID = billaddr.AddressID

INNER JOIN Person.StateProvince billstat

ON billaddr.StateProvinceID = billstat.StateProvinceID

INNER JOIN Person.CountryRegion billcountry

ON billstat.CountryRegionCode = billcountry.CountryRegionCode

INNER JOIN Person.Address shipaddr

ON head.ShipToAddressID = shipaddr.AddressID

INNER JOIN Person.StateProvince shipstat

ON shipaddr.StateProvinceID = shipstat.StateProvinceID

INNER JOIN Person.CountryRegion shipcountry

ON shipstat.CountryRegionCode = shipcountry.CountryRegionCode

WHERE

head.SalesOrderNumber = @OrderNumber;

END

GO

Procedura przyjmuje numer zamówienia i zwraca parametry opisujące zamówienie (poza listą produktów, która jest zwracana przez inną procedurę).

Wywołanie:

using var cmd = new SqlCommand("dbo.uspGetOrderDetails", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@OrderNumber", \_orderNumber);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

TextBoxOrderNumber.Text = $"{reader["OrderNumber"]}";

DateTimeOderDate.Value = (DateTime)reader["OrderDate"];

DateTimeDueDate.Value = (DateTime)reader["DueDate"];

CheckBoxOnlineOrder.Checked = (bool)reader["OnlineOrderFlag"];

TextBoxPurchaseNumber.Text = $"{reader["PurchaseNumber"]}";

TextBoxCustomerName.Text = $"{reader["CustomerName"]}";

TextBoxBillingAddress.Text = $"{reader["BillingAddress"]}";

TextBoxBillingAddressL2.Text = $"{reader["BillingAddressL2"]}";

TextBoxBillingCity.Text = $"{reader["BillingCityName"]}";

TextBoxBillingState.Text = $"{reader["BillingStateName"]}";

TextBoxBillingCountry.Text = $"{reader["BillingCountryName"]}";

TextBoxShippingAddress.Text = $"{reader["ShippingAddress"]}";

TextBoxShippingAddressL2.Text = $"{reader["ShippingAddressL2"]}";

TextBoxShippingCity.Text = $"{reader["ShippingCityName"]}";

TextBoxShippingState.Text = $"{reader["ShippingStateName"]}";

TextBoxShippingCountry.Text = $"{reader["ShippingCountryName"]}";

TextBoxTotal.Text = $"{reader["TotalDue"]}";

}

### Lista Produktów Zamówienia

CREATE OR ALTER PROCEDURE dbo.uspGetProductsInOrder

(

@OrderNumber NVARCHAR(25)

)

AS

BEGIN

SELECT DISTINCT

head.SalesOrderID AS OrderID,

prod.Name AS ProductName,

det.OrderQty AS Quantity,

det.LineTotal AS LineTotal

FROM

Sales.SalesOrderHeader head

INNER JOIN Sales.SalesOrderDetail det

ON head.SalesOrderID = det.SalesOrderID

INNER JOIN Production.Product prod

ON det.ProductID = prod.ProductID

WHERE

head.SalesOrderNumber = @OrderNumber

ORDER BY

prod.Name,

det.OrderQty;

END

GO

Procedura przyjmuje numer zamówienia i zwraca listę produktów, które były w tym zamówieniu, tj. listę produktów, ich ilości oraz cen cząstkowych.

Wywołanie:

ListBoxOrderedProducts.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetProductsInOrder", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@OrderNumber", \_orderNumber);

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ListBoxOrderedProducts.Items.Add($"{reader["Quantity"]}x {reader["ProductName"]} (total: {reader["LineTotal"]})");

}

}

## System Rejestracji Nowych Produktów

### Jednostki Miar

CREATE OR ALTER PROCEDURE dbo.uspGetSizeUnitMeasureCodes

AS

BEGIN

SELECT SizeUnitMeasureCode

FROM Production.Product

GROUP BY SizeUnitMeasureCode

END

GO

CREATE OR ALTER PROCEDURE dbo.uspGetWeightUnitMeasureCodes

AS

BEGIN

SELECT WeightUnitMeasureCode

FROM Production.Product

GROUP BY WeightUnitMeasureCode

END

GO

Obie procedury wyszukują wszystkie jednostki miar dostępne w bazie danych. Ponownie, powinno tu być wykorzystane SELECT DISCTINCT, a nie GROUP BY.

Wykorzystanie:

ComboBoxSizeUnit.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetSizeUnitMeasureCodes", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ComboBoxSizeUnit.Items.Add($"{reader["SizeUnitMeasureCode"]}");

}

}

ComboBoxSizeUnit.SelectedItem = ComboBoxSizeUnit.Items[0];

ComboBoxWeightUnit.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetWeightUnitMeasureCodes", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ComboBoxWeightUnit.Items.Add($"{reader["WeightUnitMeasureCode"]}");

}

}

ComboBoxWeightUnit.SelectedItem = ComboBoxWeightUnit.Items[0];

### Produkty Pasujące do Podkategorii lub Kategorii

CREATE OR ALTER PROCEDURE dbo.uspGetProductsBySubcategoryName

(

@SubcategoryName NVARCHAR(50),

@CategoryName NVARCHAR(50) = 'All'

)

AS

BEGIN

IF @SubcategoryName = 'All'

BEGIN

IF @CategoryName = 'All'

BEGIN

SELECT

pp.ProductID AS ProductID,

pp.Name AS ProductName,

pp.ProductNumber AS ProductNumber,

ppc.ProductCategoryID AS CategoryID,

ppc.Name AS CategoryName,

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

pp.Color AS ProductColor,

pp.SafetyStockLevel AS ProductSafetyStockLevel,

pp.ReorderPoint AS ProductReorderPoint,

pp.StandardCost AS ProductStandardCost,

pp.ListPrice AS ProductListPrice,

pp.Weight AS ProductWeight,

pp.WeightUnitMeasureCode AS ProductWeightUnit,

pp.Size AS ProductSize,

pp.SizeUnitMeasureCode AS ProductSizeUnit,

pp.SellStartDate AS ProductSellStartDate

FROM

Production.Product pp

INNER JOIN Production.ProductSubcategory pps

ON pp.ProductSubcategoryID = pps.ProductSubcategoryID

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

ORDER BY

ProductNumber,

ProductName;

END

ELSE

BEGIN

SELECT

pp.ProductID AS ProductID,

pp.Name AS ProductName,

pp.ProductNumber AS ProductNumber,

ppc.ProductCategoryID AS CategoryID,

ppc.Name AS CategoryName,

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

pp.Color AS ProductColor,

pp.SafetyStockLevel AS ProductSafetyStockLevel,

pp.ReorderPoint AS ProductReorderPoint,

pp.StandardCost AS ProductStandardCost,

pp.ListPrice AS ProductListPrice,

pp.Weight AS ProductWeight,

pp.WeightUnitMeasureCode AS ProductWeightUnit,

pp.Size AS ProductSize,

pp.SizeUnitMeasureCode AS ProductSizeUnit,

pp.SellStartDate AS ProductSellStartDate

FROM

Production.Product pp

INNER JOIN Production.ProductSubcategory pps

ON pp.ProductSubcategoryID = pps.ProductSubcategoryID

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

WHERE

ppc.Name = @CategoryName

ORDER BY

ProductNumber,

ProductName;

END

END

ELSE

BEGIN

SELECT

pp.ProductID AS ProductID,

pp.Name AS ProductName,

pp.ProductNumber AS ProductNumber,

ppc.ProductCategoryID AS CategoryID,

ppc.Name AS CategoryName,

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

pp.Color AS ProductColor,

pp.SafetyStockLevel AS ProductSafetyStockLevel,

pp.ReorderPoint AS ProductReorderPoint,

pp.StandardCost AS ProductStandardCost,

pp.ListPrice AS ProductListPrice,

pp.Weight AS ProductWeight,

pp.WeightUnitMeasureCode AS ProductWeightUnit,

pp.Size AS ProductSize,

pp.SizeUnitMeasureCode AS ProductSizeUnit,

pp.SellStartDate AS ProductSellStartDate

FROM

Production.Product pp

INNER JOIN Production.ProductSubcategory pps

ON pp.ProductSubcategoryID = pps.ProductSubcategoryID

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

WHERE

pps.Name = @SubcategoryName

ORDER BY

ProductNumber,

ProductName;

END

END

GO

Procedura przyjmuje nazwę podkategorii i/lub kategorii, i wyszukuje wszystkie produkty należące do danej podkategorii. Jeśli podkategorii nie określono, wyszukuje wszystkie produkty w kategorii. Jeśli jej też nie określono, wyszukuje wszystkie produkty w bazie danych.

Wywołanie:

string subcategoryName = ComboBoxSubcategories.Text;

string categoryName = ComboBoxCategories.Text;

ListBoxProducts.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetProductsBySubcategoryName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@SubcategoryName", subcategoryName);

cmd.Parameters.AddWithValue("@CategoryName", categoryName);

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ListBoxProducts.Items.Add($"{reader["ProductNumber"]} --- {reader["ProductName"]}");

}

}

### Wyszukiwanie Produktu po Nazwie

CREATE OR ALTER PROCEDURE dbo.uspGetProductByName

(

@ProductName NVARCHAR(50)

)

AS

BEGIN

SELECT

pp.ProductID AS ProductID,

pp.Name AS ProductName,

pp.ProductNumber AS ProductNumber,

ppc.ProductCategoryID AS CategoryID,

ppc.Name AS CategoryName,

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

pp.Color AS ProductColor,

pp.SafetyStockLevel AS ProductSafetyStockLevel,

pp.ReorderPoint AS ProductReorderPoint,

pp.StandardCost AS ProductStandardCost,

pp.ListPrice AS ProductListPrice,

pp.Weight AS ProductWeight,

pp.WeightUnitMeasureCode AS ProductWeightUnit,

pp.Size AS ProductSize,

pp.SizeUnitMeasureCode AS ProductSizeUnit,

pp.SellStartDate AS ProductSellStartDate

FROM

Production.Product pp

INNER JOIN Production.ProductSubcategory pps

ON pp.ProductSubcategoryID = pps.ProductSubcategoryID

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

WHERE

pp.Name = @ProductName;

END

GO

Procedura przyjmuje nazwę produktu, zakładając unikalność, i zwraca listę parametrów opisujących ten produkt.

Wywołanie:

using var cmd = new SqlCommand("dbo.uspGetProductByName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@ProductName", \_name);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

TextBoxName.Text = $"{reader["ProductName"]}";

TextBoxProductNumber.Text = $"{reader["ProductNumber"]}";

TextBoxColor.Text = $"{reader["ProductColor"]}";

TextBoxCategory.Text = $"{reader["CategoryName"]}";

TextBoxSubcategory.Text = $"{reader["SubcategoryName"]}";

TextBoxSafetyStockLevel.Text = $"{reader["ProductSafetyStockLevel"]}";

TextBoxReorderPoint.Text = $"{reader["ProductReorderPoint"]}";

TextBoxStandardCost.Text = $"{reader["ProductStandardCost"]}";

TextBoxListPrice.Text = $"{reader["ProductListPrice"]}";

TextBoxWeight.Text = $"{reader["ProductWeight"]}";

TextBoxWeightUnit.Text = $"{reader["ProductWeightUnit"]}";

TextBoxSize.Text = $"{reader["ProductSize"]}";

TextBoxSizeUnit.Text = $"{reader["ProductSizeUnit"]}";

DateTimePickerSellStartDate.Text = $"{reader["ProductSellStartDate"]}";

}

### Dodawanie Nowego Produktu

CREATE OR ALTER PROCEDURE dbo.uspCreateProduct

(

@Name NVARCHAR(50) = NULL,

@ProductNumber NVARCHAR(25) = NULL,

@CategoryID INT = NULL,

@SubcategoryID INT = NULL,

@Color NVARCHAR(15) = NULL,

@SafetyStockLevel SMALLINT,

@ReorderPoint SMALLINT,

@StandardCost MONEY,

@ListPrice MONEY,

@Weight DECIMAL(8,2) = NULL,

@WeightUnit NCHAR(3) = NULL,

@Size NVARCHAR(5) = NULL,

@SizeUnit NCHAR(3) = NULL,

@SellStartDate DATETIME

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF EXISTS (SELECT 1 FROM Production.Product WHERE ProductNumber = @ProductNumber)

THROW 50101, 'Product with that number already exists!', 1;

IF EXISTS (SELECT 1 FROM Production.Product WHERE Name = @Name)

THROW 50102, 'Product with that name already exists!', 1;

IF @ProductNumber IS NULL

THROW 50103, 'Product number is required!', 1;

IF @Name IS NULL

THROW 50104, 'Name is required!', 1;

IF LEN(@ProductNumber) < 2

THROW 50105, 'Product number must be at least 2 characters long!', 1;

IF LEN(@Name) < 6

THROW 50103, 'Name must be at least 6 characters long!', 1;

INSERT INTO Production.Product

(

Name,

ProductNumber,

ProductSubcategoryID,

Color,

SafetyStockLevel,

ReorderPoint,

StandardCost,

ListPrice,

Weight,

WeightUnitMeasureCode,

Size,

SizeUnitMeasureCode,

SellStartDate,

DaysToManufacture

)

VALUES

(

@Name,

@ProductNumber,

@SubcategoryID,

@Color,

@SafetyStockLevel,

@ReorderPoint,

@StandardCost,

@ListPrice,

@Weight,

@WeightUnit,

@Size,

@SizeUnit,

@SellStartDate,

1

);

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura przyjmuje listę parametrów i próbuje stworzyć nowy produkt w bazie danych. Niektóre pola są wymagane i te są następne walidowane przez procedurę, tj. numer produktu musi mieć co najmniej dwa znaki długości, a nazwa – co najmniej 6.

Wykorzystanie:

string? name = TextBoxName.Text.IsNullOrEmpty() ? null : TextBoxName.Text;

string? productNumber = TextBoxProductNumber.Text.IsNullOrEmpty() ? null : TextBoxProductNumber.Text;

string? color = TextBoxColor.Text.IsNullOrEmpty() ? null : TextBoxColor.Text;

short safetyStockLevel = (short)NumericSafetyStockLevel.Value;

short reorderPoint = (short)NumericReorderPoint.Value;

decimal standardCost = NumericStandardCost.Value;

decimal listPrice = NumericListPrice.Value;

decimal? weight = NumericWeight.Value == 0 ? null : NumericWeight.Value;

string? weightUnit = ComboBoxWeightUnit.Text.IsNullOrEmpty() ? null : ComboBoxWeightUnit.Text;

string? size = TextBoxSize.Text.IsNullOrEmpty() ? null : TextBoxSize.Text;

string? sizeUnit = ComboBoxSizeUnit.Text.IsNullOrEmpty() ? null : ComboBoxSizeUnit.Text;

DateTime sellStartDate = DateTimePickerSellStartDate.Value;

{

using var cmd = new SqlCommand("dbo.uspCreateProduct", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Name", name);

cmd.Parameters.AddWithValue("@ProductNumber", productNumber);

cmd.Parameters.AddWithValue("@CategoryID", categoryId);

cmd.Parameters.AddWithValue("@SubcategoryID", subcategoryId);

cmd.Parameters.AddWithValue("@Color", color);

cmd.Parameters.AddWithValue("@SafetyStockLevel", safetyStockLevel);

cmd.Parameters.AddWithValue("@ReorderPoint", reorderPoint);

cmd.Parameters.AddWithValue("@StandardCost", standardCost);

cmd.Parameters.AddWithValue("@ListPrice", listPrice);

cmd.Parameters.AddWithValue("@Weight", weight);

cmd.Parameters.AddWithValue("@WeightUnit", weightUnit);

cmd.Parameters.AddWithValue("@Size", size);

cmd.Parameters.AddWithValue("@SizeUnit", sizeUnit);

cmd.Parameters.AddWithValue("@SellStartDate", sellStartDate);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

## Zarządzanie Kategoriami Produktów

### Wyszukanie Wszystkich Kategorii

CREATE OR ALTER PROCEDURE dbo.uspGetCategories

AS

BEGIN

SELECT

ProductCategoryID as CategoryID,

Name AS CategoryName

FROM

Production.ProductCategory

ORDER BY

CategoryName;

END

GO

Procedura wyszukuje wszystkie kategorie w systemie.

Wykorzystanie:

ListBoxCategories.Items.Clear();

ListBoxCategories.Items.Add("All");

ListBoxCategories.SelectedItem = ListBoxCategories.Items[0];

using var cmd = new SqlCommand("dbo.uspGetCategories", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ListBoxCategories.Items.Add($"{reader["CategoryName"]}");

}

### Wyszukanie Kategorii po Nazwie

CREATE OR ALTER PROCEDURE dbo.uspGetCategoryByName

(

@CategoryName NVARCHAR(50)

)

AS

BEGIN

SELECT

ProductCategoryID AS CategoryID,

Name AS CategoryName

FROM

Production.ProductCategory

WHERE

Name = @CategoryName;

END

GO

Procedura przyjmuje nazwę kategorii i zwraca jej parametry – tutaj tylko nazwę oraz ID.

Wykorzystanie:

using var cmd = new SqlCommand("dbo.uspGetCategoryByName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryName", \_categoryName);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

TextBoxID.Text = reader["CategoryID"].ToString();

TextBoxName.Text = reader["CategoryName"].ToString();

}

### Dodanie Kategorii

CREATE OR ALTER PROCEDURE dbo.uspCreateCategory

(

@Name NVARCHAR(50)

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF LEN(@Name) < 3

BEGIN

;THROW 50013, 'Category name must be at least 3 characters long!', 1;

END

IF EXISTS (

SELECT 1

FROM Production.ProductCategory

WHERE Name = @Name

)

BEGIN

;THROW 50014, 'Another Category already has that name!', 1;

END

INSERT INTO Production.ProductCategory (Name)

VALUES (@Name);

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura przyjmuje pole na nazwę, a następnie je waliduje i próbuje utworzyć nową kategorię z podaną nazwą.

Wykorzystanie:

string name = TextBoxName.Text;

using var cmd = new SqlCommand("dbo.uspCreateCategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Name", name);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

### Aktualizowanie Kategorii

CREATE OR ALTER PROCEDURE dbo.uspUpdateCategory

(

@CategoryID INT,

@NewName NVARCHAR(50)

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF EXISTS (

SELECT 1

FROM Production.ProductCategory

WHERE Name = @NewName AND ProductCategoryID != @CategoryID

)

BEGIN

;THROW 50012, 'Another Category already has that name!', 1;

END

UPDATE Production.ProductCategory

SET

Name = @NewName

WHERE

ProductCategoryID = @CategoryID;

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura przyjmuje ID oraz nową nazwę kategorii, a następnie waliduje nazwę i próbuje zmienić nazwę kategorii.

Wykorzystanie:

int id = int.Parse(TextBoxID.Text);

string newName = TextBoxName.Text;

using var cmd = new SqlCommand("dbo.uspUpdateCategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryID", id);

cmd.Parameters.AddWithValue("@NewName", newName);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

### Usuwanie Kategorii

CREATE OR ALTER PROCEDURE dbo.uspDeleteCategory

(

@CategoryID INT

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF EXISTS (

SELECT 1

FROM Production.ProductSubcategory

WHERE ProductCategoryID = @CategoryID

)

BEGIN

;THROW 50011, 'Cannot delete a Category which has associated Subcategories!', 1;

END

DELETE FROM Production.ProductCategory

WHERE ProductCategoryID = @CategoryID;

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura usuwa kategorię o podanym ID. Jeśli kategoria posiada jakieś skojarzone podkategorie, usunięcie jest niemożliwe.

Wykorzystanie:

int id = int.Parse(TextBoxID.Text);

using var cmd = new SqlCommand("dbo.uspDeleteCategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryID", id);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

### Wyszukiwanie Wszystkich Podkategorii Danej Kategorii

CREATE OR ALTER PROCEDURE dbo.uspGetSubcategoriesByCategoryName

(

@CategoryName NVARCHAR(50)

)

AS

BEGIN

IF @CategoryName = 'All'

BEGIN

SELECT

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

ppc.Name AS CategoryName

FROM

Production.ProductSubcategory pps

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

ORDER BY

CategoryName,

SubcategoryName;

END

ELSE

BEGIN

SELECT

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

ppc.Name AS CategoryName

FROM

Production.ProductSubcategory pps

INNER JOIN Production.ProductCategory ppc

ON pps.ProductCategoryID = ppc.ProductCategoryID

WHERE

ppc.Name = @CategoryName

ORDER BY

CategoryName,

SubcategoryName;

END

END

GO

Procedura przyjmuje nazwę kategorii, a następnie wyszukuje wszystkie podkategorie skojarzone z podaną kategorią. Jeśli ta nie będzie określona, procedura zwróci wszystkie podkategorie.

Wykorzystanie:

var categoryName = ListBoxCategories.SelectedItem!.ToString();

ListBoxSubcategories.Items.Clear();

using var cmd = new SqlCommand("dbo.uspGetSubcategoriesByCategoryName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryName", categoryName);

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ListBoxSubcategories.Items.Add($"{reader["SubcategoryName"]}");

}

### Wyszukanie Podkategorii po Nazwie

CREATE OR ALTER PROCEDURE dbo.uspGetSubcategoryByName

(

@SubcategoryName NVARCHAR(50)

)

AS

BEGIN

SELECT

pps.ProductSubcategoryID AS SubcategoryID,

pps.Name AS SubcategoryName,

ppc.ProductCategoryID AS CategoryID,

ppc.Name AS CategoryName

FROM

Production.ProductCategory ppc

INNER JOIN Production.ProductSubcategory pps

ON ppc.ProductCategoryID = pps.ProductCategoryID

WHERE

pps.Name = @SubcategoryName

ORDER BY

CategoryName,

SubcategoryName;

END

GO

Procedura wyszukuje podkategorię o podanej nazwie i zwraca parametry, które ją opisują.

Wykorzystanie:

ComboBoxCategories.Items.Clear();

{

using var cmd = new SqlCommand("dbo.uspGetCategories", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

using var reader = cmd.ExecuteReader();

while (reader.Read())

{

ComboBoxCategories.Items.Add($"{reader["CategoryName"]}");

}

}

{

using var cmd = new SqlCommand("dbo.uspGetSubcategoryByName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@SubcategoryName", \_subcategoryName);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

TextBoxID.Text = reader["SubcategoryID"].ToString();

ComboBoxCategories.SelectedItem = reader["CategoryName"];

}

}

TextBoxName.Text = \_subcategoryName;

### Dodawanie Podkategorii

CREATE OR ALTER PROCEDURE dbo.uspCreateSubcategory

(

@CategoryID INT,

@Name NVARCHAR(50)

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF NOT EXISTS (

SELECT 1

FROM Production.ProductCategory

WHERE ProductCategoryID = @CategoryID

)

BEGIN

;THROW 50025, 'Given Category does not exist!', 1;

END

IF LEN(@Name) < 3

BEGIN

;THROW 50023, 'Subcategory name must be at least 3 characters long!', 1;

END

IF EXISTS (

SELECT 1

FROM Production.ProductSubcategory

WHERE Name = @Name

)

BEGIN

;THROW 50024, 'Another Subcategory already has that name!', 1;

END

INSERT INTO Production.ProductSubcategory (Name, ProductCategoryID)

VALUES (@Name, @CategoryID);

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura przyjmuje nazwę podkategorii oraz ID skojarzonej kategorii. Następnie waliduje ID oraz nazwę i próbuje stworzyć podkategorię o podanej nazwie i skojarzoną z kategorią o podanym ID.

Wykorzystanie:

string newName = TextBoxName.Text.Trim();

string selectedCategoryName = ComboBoxCategories.SelectedItem!.ToString()!;

int selectedCategoryId = -1;

{

using var cmd = new SqlCommand("dbo.uspGetCategoryByName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryName", selectedCategoryName);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

selectedCategoryId = Convert.ToInt32(reader["CategoryID"]);

}

}

if (selectedCategoryId != -1)

{

using var cmd = new SqlCommand("dbo.uspCreateSubcategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryID", selectedCategoryId);

cmd.Parameters.AddWithValue("@Name", newName);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

### Aktualizowanie Podkategorii

CREATE OR ALTER PROCEDURE dbo.uspUpdateSubcategory

(

@SubcategoryID INT,

@NewCategoryID INT,

@NewName NVARCHAR(50)

)

AS

BEGIN

BEGIN TRANSACTION

BEGIN TRY

IF EXISTS (

SELECT 1

FROM Production.ProductSubcategory

WHERE Name = @NewName AND ProductSubcategoryID != @SubcategoryID

)

BEGIN

;THROW 50022, 'Another Subcategory already has that name!', 1;

END

UPDATE Production.ProductSubcategory

SET

Name = @NewName,

ProductCategoryID = @NewCategoryID

WHERE

ProductSubcategoryID = @SubcategoryID;

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura przyjmuje ID podkategorii, nowe ID kategorii oraz nową nazwę, następne waliduje dane i próbuje zaktualizować podkategorię.

Wykorzystanie:

int id = int.Parse(TextBoxID.Text);

string newName = TextBoxName.Text.Trim();

string selectedCategoryName = ComboBoxCategories.SelectedItem!.ToString()!;

int selectedCategoryId = -1;

{

using var cmd = new SqlCommand("dbo.uspGetCategoryByName", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@CategoryName", selectedCategoryName);

using var reader = cmd.ExecuteReader();

if (reader.Read())

{

selectedCategoryId = Convert.ToInt32(reader["CategoryID"]);

}

}

if (selectedCategoryId != -1)

{

using var cmd = new SqlCommand("dbo.uspUpdateSubcategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@SubcategoryID", id);

cmd.Parameters.AddWithValue("@NewCategoryID", selectedCategoryId);

cmd.Parameters.AddWithValue("@NewName", newName);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

### Usuwanie Podkategorii

CREATE OR ALTER PROCEDURE dbo.uspDeleteSubcategory

(

@SubcategoryID INT

)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF EXISTS (

SELECT 1

FROM Production.Product

WHERE ProductSubcategoryID = @SubcategoryID

)

BEGIN

;THROW 50021, 'Cannot delete a Subcategory which has associated Products!', 1;

END

DELETE FROM Production.ProductSubcategory

WHERE ProductSubcategoryID = @SubcategoryID;

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

;THROW;

END CATCH

END

GO

Procedura próbuje usunąć podkategorię o podanym ID. Jeśli podkategoria posiada skojarzone produkty, usunięcie jest niemożliwe.

Wykorzystanie:

int id = int.Parse(TextBoxID.Text);

using var cmd = new SqlCommand("dbo.uspDeleteSubcategory", \_connection);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@SubcategoryID", id);

try

{

cmd.ExecuteNonQuery();

Close();

}

catch (SqlException ex)

{

MessageBox.Show($"Database error: {ex.Message}", "SQL Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Unexpected error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}