

Bazy Danych - projekt
Sklep Internetowy

Karolina Prygiel

Mateusz Gwóźdź

Rozdział 1

Podstawowe założenia projektu

1.1 Cel i główne założenia projektu

Celem projektu jest implementacja bazy danych do obsługi sklepu internetowego z kosmetykami.

Baza danych przechowuje informacje o **klientach**, aktualnej zawartości ich **koszyków**, a także o dokonanych przez nich **zamówieniach**. Każdy klient po dokonaniu zamówienia może wystawić recenzję zakupionym produktom lub ocenić istniejący już komentarz.

Przechowywane są również informacje o **produktach** dostępnych w sklepie. Ich **kategoriach**, **podkategoriach**, **markach** oraz aktualnym stanie w magazynie. Ponadto baza zawiera informacje o **pracownikach** sklepu, którzy zatrudnieni są w pewnych **działach** a także o **dostawach** produktów i ich **dostawcach**.

1.2 Klienci

Klienci są dodawani do bazy za pomocą procedury ADD CUSTOMER, która jednocześnie przypisuje im pusty koszyk na zakupy oraz dodaje ich do listy mailingowej, co umożliwia późniejsze wysyłanie newslettera. Po pomyślnym dodaniu do bazy, klient otrzymuje maila powitalnego - trigger CONFIRMATION EMAIL, przez który musi potwierdzić swoje konto.

1.3 Dodawanie produktów do koszyka oraz składanie zamówień

Produkty można zakupić wyłącznie poprzez uprzednie dodanie ich do koszyka. Służy do tego procedura ADD TO CART. Można dodawać wiele sztuk tego samego produktu oraz usuwać wybrane artykuły z koszyka - procedura DELETE FROM CART. Zostały zaimplementowane zabezpieczenia chroniące przed dodaniem/usunięciem z koszyka nieprawidłowej ilości produktów.

Złożenia zamówienia realizuje procedura BUY, która przepisuje zawartość koszyka do tabeli ORDER DETAILS, następnie usuwa go i przydziela klientowi nowy (pusty) koszyk. Losowo przydzielany jest również pracownik, który spakuje zamówienie.

Możliwe jest również wygenerowanie faktury - funkcja GENERATE NOTE, zawierającej informacje o zamawiającym, dacie i całkowitej wartości zamówienia oraz o ilości kupionych produktów.

1.4 Opinie o produktach

Klienci mogą wystawiać opinie oraz oceniać produkty w skali 1-5. Ponadto można oceniać recenzje innych użytkowników pozytywnie /negatywnie odpowiednio poprzez '+' oraz '-'. Za realizację czego odpowiadają procedury ADD REVIEW oraz ADD REVIEW RATING.

Zostały zaimplementowane funkcje służące do manipulowania opiniami. Można wyświetlić wszystkie recenzje na temat danego produktu - funkcja GET REVIEWS, wyświetlić jego średnią ocenę (w skali 1-5) - funkcja AVG. PRODUCT RATE, a także wyświetlić ilość '+' i '-' poszczególnych recenzji - GET REVIEW RATINGS.

1.5 Produkty

Baza danych przechowuje informacje na temat produktów dostępnych w sklepie.

Każdy produkt posiada przypisaną kategorię, podkategorię oraz jest produkowany przez określoną markę. Ze względu na to, że jest to sklep z kosmetykami, pojęcie marki ma tu szczególnie duże znaczenie. Tabela **brands**

przechowuje informacje o dostępnych w sklepie markach. Wyzwalacz ASSIGN BRAND ID dla każdego nowego produktu, sprawdza czy jego marka istnieje już w bazie i jeśli tak nie jest - automatycznie dodaje ją do tabeli Brands. Również gdy w ofercie sklepu pojawi się zupełnie nowy produkt, do klientów sklepu zostaje wysłany mail informujący o dostępności nowych produktów - wyzwalacz NEW PRODUCT NOTIFICATION.

Przechowujemy również informacje na temat **ilości** dostępnych produktów w magazynie. Widok STORAGE STATUS pozwala w przejrzysty sposób uzyskać informacje na temat stanu wszystkich produktów w bazie, a gdy któregoś z produktów zaczyna brakować, trigger STOCK ALERT zawiadamia personel o wyczerpaniu jego zapasów.

Produkty są dostarczane przez **dostawców**. W bazie przechowujemy informacje o poszczególnych **dostawach** i ich **szczegółach**.

1.6 Wyszukiwanie produktów

Zaprejektowano 2 sposoby wyszukiwania produktów w bazie:

1. Funkcja SEARCH pozwalająca określić:

- Markę;
- kategorię;
- podkategorię;
- przedział cenowy.

Wszystkie parametry są opcjonalne, a niesprecyzowanie któregokolwiek z nich powoduje wyświetlenie produktów ze wszystkich dostępnych marek, kategorii, przedziałów cenowych itd. Dzięki temu możemy wyszukiwać produkty tylko z określonej markii, kategorii lub w pewnym przedziale cenowym.

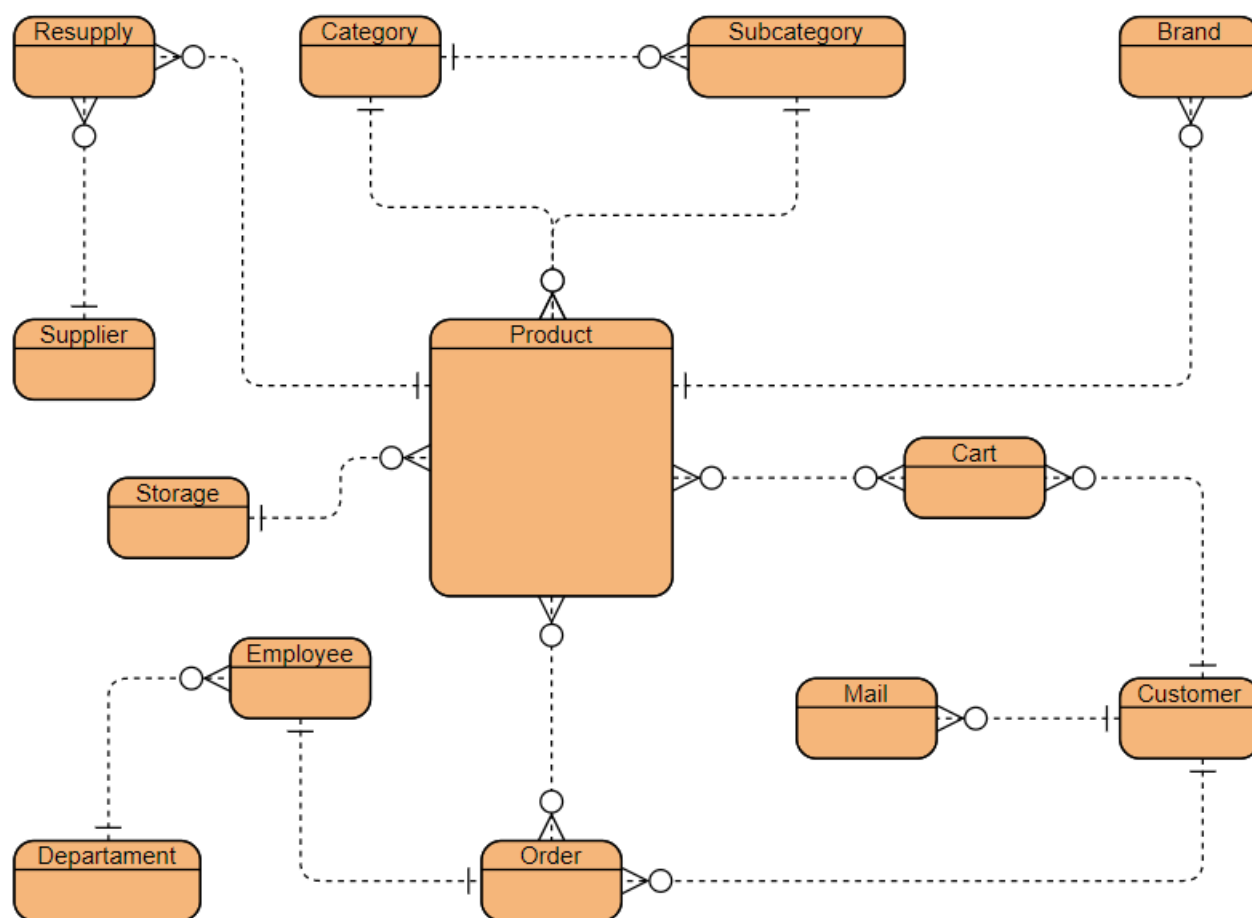
2. Funkcja PRODUCT LOOKUP pozwalająca wyszukiwać produkty po fragmencie jego nazwy lub nazwy jego marki.

1.7 Przyjęte ograniczenia

Baza nie obsługuje zwrotów produktów, reklamacji a także informacji o sposobie realizacji dostaw i zamówień przez klientów. Ponadto nie utworzono kont użytkowników, przechowujących hasła i umożliwiających m.in logowanie się do sklepu.

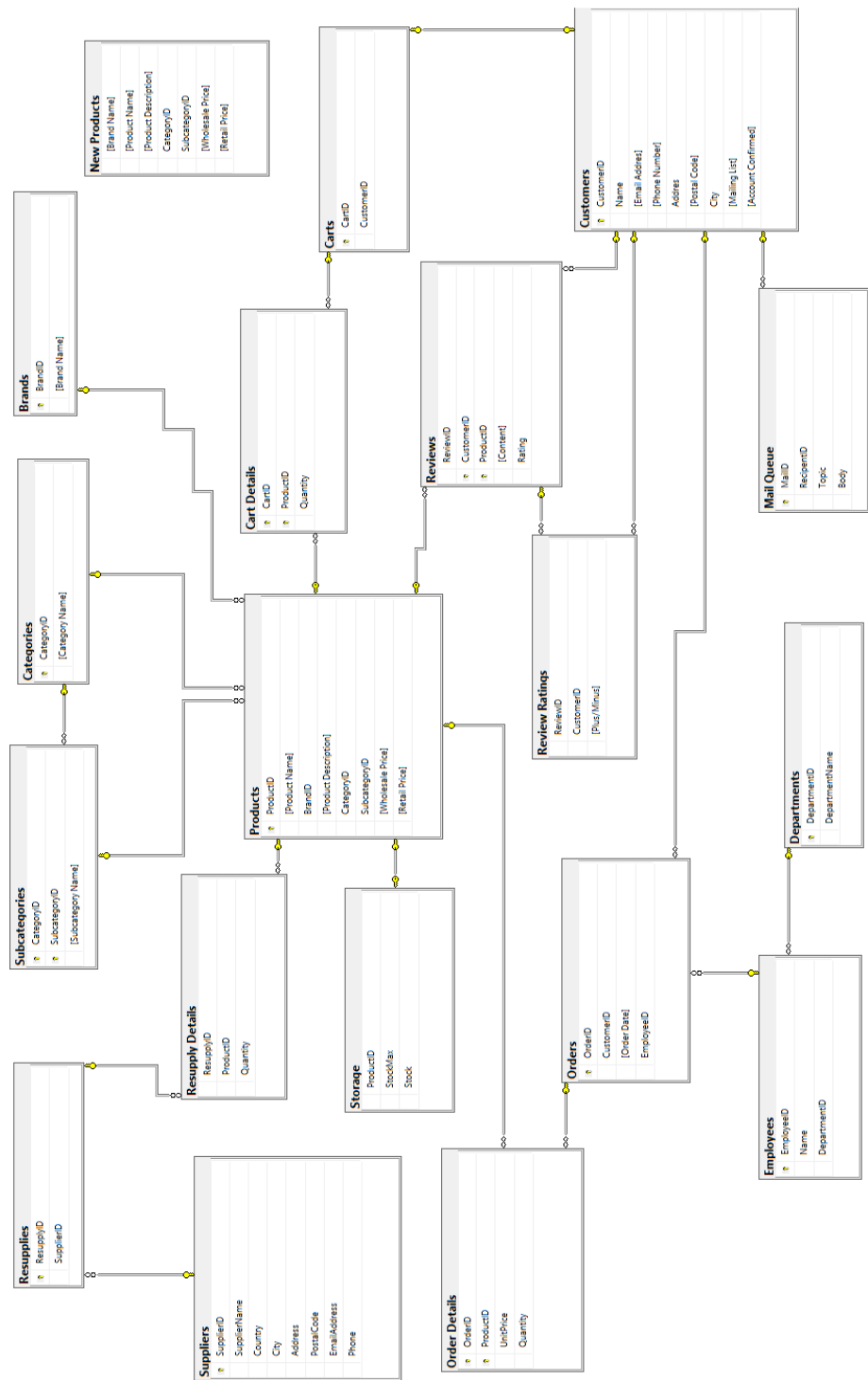
Rozdział 2

Diagram ER



Rozdział 3

Schemat bazy danych



Rozdział 4

Widoki

4.1 Storage Status

Widok przedstawiający informacje na temat aktualnego stanu magazynu.

	ProductID	Product Name	Stock
1	1	Colorstay	994 of 1000
2	2	X-CEPTIONAL WEAR	991 of 1000
3	3	GOLDEN FAIREST	893 of 1000

Widok *Storage Status* dla przykładowych danych

```
1 USE TestDB;
2 GO
3
4 DROP VIEW IF EXISTS [dbo].[Storage Status];
5 GO
6
7 CREATE VIEW [Storage Status]
8 AS
9 SELECT P.ProductID, P.[Product Name],
10        CONVERT(VARCHAR(10), S.Stock) + ' of ' + CONVERT(VARCHAR(10),
11        S.StockMax) AS [Stock]
12 FROM [dbo].[Storage] AS S
13 RIGHT JOIN [dbo].[Products] AS P ON S.ProductID = P.ProductID
14
15 GO
```

Kod generujący widok *Storage Status*

4.2 Prodcfs Bought

Widok przedstawiający informacje na temat aktualnej ilości sumy zamówień wszystkich produktów.

	Brand	Product Name	Quantity
1	Anabelle Minerals	GOLDEN FAIREST	7
2	Revlon	Colorstay	5
3	Gosh	X-CEPTIONAL WEAR	4

Widok *Products Bought* dla przykładowych danych

```

1 USE TestDB;
2 GO
3
4 DROP VIEW IF EXISTS [dbo].[Products Bought];
5 GO
6
7 CREATE VIEW [Products Bought]
8 AS
9 SELECT B.[Brand Name] AS [Brand], P.[Product Name], Pr.[Quantity]
10 FROM Products AS P
11 JOIN (SELECT P.ProductID, SUM(OD.Quantity) AS [Quantity]
12       FROM [dbo].[Products] AS P
13       JOIN [dbo].[Order Details] AS OD ON P.ProductID = OD.ProductID
14       GROUP BY P.ProductID) AS Pr ON P.ProductID = Pr.ProductID
15 JOIN [dbo].[Brands] AS B ON P.BrandID = B.BrandID
16
17 GO

```

Kod generujący widok *Prodcts Bought*

4.3 Hot Brands

Widok przedstawiający informacje na temat najchętniej kupowanych marek.

	Brand Name	Total Quantity
1	Anabelle Minerals	7
2	Revlon	5
3	Gosh	4

Widok *Hot Brands* dla przykładowych danych

```

1 USE TestDB
2 ;
3 GO
4
5 DROP VIEW IF EXISTS [dbo].[Hot Brands]
6 ;
7 GO
8
9 CREATE VIEW [Hot Brands]
10 AS
11 SELECT TOP 3 Brands.[Brand Name], SUM(Quantity) AS [Total Quantity] FROM [Order Details] OD
12     JOIN Products ON OD.ProductID = Products.ProductID
13     JOIN Brands ON Products.BrandId = Brands.BrandId
14 GROUP BY ([Brand Name])
15 Order by [Total Quantity] DESC
16 ;
17 GO

```

Kod generujący widok *Hot Brands*

4.4 Category Organisation

Widok przedstawiający strukturę kategorii i podkategorii będących podziałami produktów.

CategoryID	Category Name	SubcategoryID	Subcategory Name
3	Oczy	21	Pedzle do oczu
3	Oczy	22	Akcesoria
4	Brwi	23	Kredki do brwi
4	Brwi	24	Cienie
4	Brwi	25	Korektory do brwi
4	Brwi	26	Maskary, zele, woski, po...
4	Brwi	27	Pedzle do brwi
4	Brwi	28	Akcesoria
5	Paznokcie	29	Lakier do paznokci
5	Paznokcie	30	Lakier hybrydowy
5	Paznokcie	31	Pielęgnacja
5	Paznokcie	32	Zmywacze

Widok *Category Organisation* dla przykładowych danych

```
1 USE TestDB;
2 GO
3
4 DROP VIEW IF EXISTS [dbo].[Category Organisation];
5 GO
6
7 CREATE VIEW [Category Organisation]
8 AS
9 SELECT C.CategoryID, C.[Category Name], S.SubcategoryID, S.[Subcategory Name]
10 FROM [dbo].[Categories] AS C
11 LEFT JOIN [dbo].[Subcategories] AS S ON C.CategoryID = S.CategoryID
12
13 GO
```

Kod generujący widok *Category Organisation*

Rozdział 5

Procedury składowane

5.1 Add Customer

Procedura pozwalająca dodać nowego klienta do tabeli *Customers*. Automatycznie zostaje mu przypisany nowy, pusty *koszyk*.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS dbo.AddCustomer;
5 GO
6
7 CREATE PROC dbo.AddCustomer
8
9 @Name NVARCHAR(40) = NULL,
10 @Email NVARCHAR(40) = NULL,
11 @PhoneNumber NVARCHAR(5) = NULL,
12 @Address NVARCHAR(40) = NULL,
13 @PostalCode NVARCHAR(15) = NULL,
14 @City NVARCHAR(25) = NULL
15
16 AS
17
18 DECLARE @error AS NVARCHAR(400);
19
20 IF @Name IS NULL OR @Email IS NULL
21 BEGIN
22     SET @error = 'Error!';
23     RAISERROR(@error, 16,1);
24     RETURN;
25 END
26
27 INSERT INTO Customers([Name], [Email Address], [Phone Number], [Address], [Postal Code], City)
28 VALUES (@Name, @Email, @PhoneNumber, @Address, @PostalCode, @City);
29
30
31 INSERT INTO Carts(CustomerID)
32 VALUES ((SELECT CustomerID FROM Customers WHERE [Email Address] = @Email))
33
34 GO
```

Kod generujący procedurę *Add Customer*

5.2 Add Review

Procedura służąca do dodawania recenzji na temat produktu.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS dbo.AddReview;
5 GO
6
7 CREATE PROC dbo.AddReview
8 @CustomerID INT = NULL,
9 @ProductID INT = NULL,
10 @Content NTEXT = NULL,
11 @Rating TINYINT = NULL
12
13 AS
14 DECLARE @error AS NVARCHAR(50);
15
16 IF @CustomerID IS NULL OR @ProductID IS NULL OR @Rating IS NULL
17 BEGIN
18     SET @error = 'Error! Rating lacking information.';
19     RAISERROR(@error, 16,1);
20     RETURN;
21 END
22
23 INSERT INTO dbo.[Reviews] VALUES
24 (@CustomerID, @ProductID, @Content, @Rating)
25
26 GO
```

Kod generujący procedurę *Add Review*

5.3 Add Review Rating

Procedura służąca do ocenienia recenzji.

```
1 IF OBJECT_ID('dbo.AddReviewRating', 'P') IS NOT NULL
2 DROP PROC dbo.AddReviewRating
3 GO
4
5 CREATE PROC dbo.AddReviewRating
6
7 @ReviewID INT = NULL,
8 @CustomerID INT = NULL,
9 @PlusMinus CHAR(1) = NULL
10
11 AS
12
13 DECLARE @error AS NVARCHAR(100);
14
15 IF @ReviewID IS NULL OR @CustomerID IS NULL OR @PlusMinus IS NULL
16 BEGIN
17     SET @error = 'Error! Rating lacking information.';
18     RAISERROR(@error, 16,1);
19     RETURN;
20 END
21
22 INSERT INTO dbo.[Review Ratings] VALUES
23 (@ReviewID, @CustomerID, @PlusMinus)
24
25
26 GO
```

Kod generujący procedurę *Add Review Rating*

5.4 New Mail

Procedura służąca do dodawania nowego maila do tabeli *Mail Queue*, będącej skrzynką nadawczą.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS [dbo].[New Mail];
5 GO
6
7 CREATE PROC [dbo].[New Mail]
8 @CustomerID INT,
9 @Topic VARCHAR(50),
10 @Body VARCHAR(MAX)
11 AS
12 BEGIN
13
14     IF @CustomerID IS NULL OR @Topic IS NULL OR @Body IS NULL
15     BEGIN
16         DECLARE @error NVARCHAR(50) = 'Error! Mail elements missing!';
17         RAISERROR(@error, 16,1);
18         RETURN;
19     END
20
21     INSERT INTO [dbo].[Mail Queue]
22     ([RecipientID], [Topic], [Body])
23     VALUES
24     (@CustomerID, @Topic, @Body)
25
26 END
27
28 GO
```

Kod generujący procedurę *New Mail*

5.5 Confirm Account

Procedura służąca do potwierdzania konta użytkownika w systemie.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS [dbo].[Confirm Account];
5 GO
6
7 CREATE PROC [dbo].[Confirm Account]
8 @CustomerID INT
9 AS
10 IF @CustomerID IS NULL
11 BEGIN
12     DECLARE @error NVARCHAR(50) = 'Error! Account Information missing!';
13     RAISERROR(@error, 16,1);
14     RETURN;
15 END
16
17 UPDATE [Customers]
18 SET [Account Confirmed] = 1
19 WHERE [CustomerID] = @CustomerID
20
21 GO
```

Kod generujący procedurę *Confirm Account*

5.6 Add To Cart

Procedura służąca do dodawania przez użytkownika produktu do koszyka.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS dbo.AddToCart;
5 GO
6
7
8 CREATE PROC dbo.AddToCart
9 @CustomerID INT = NULL,
10 @ProductID INT = NULL,
11 @Quantity INT = NULL
12
13
14
15 AS
16
17 DECLARE @error AS NVARCHAR(400);
18
19 IF @CustomerID IS NULL OR @ProductID IS NULL OR @Quantity IS NULL
20 BEGIN
21     SET @error = 'Error!';
22     RAISERROR(@error, 16,1);
23     RETURN;
24 END
25
26 DECLARE @CartID AS INT;
27
28 SELECT @CartID = CartID
29 FROM Carts
30 WHERE CustomerID = @CustomerID
31
32
33 DECLARE @RealQuantity AS INT
34
35 Select @RealQuantity = Stock FROM Storage WHERE ProductID = @ProductID;
36
37
38 IF @RealQuantity >= @Quantity
39 BEGIN
40     IF NOT EXISTS(Select * from [Cart Details] where CartID = @CartID and ProductID = @ProductID
41     )
42     BEGIN
43         INSERT INTO [Cart details]
44         VALUES (@CartID, @ProductId, @Quantity)
45     END
46     ELSE
47     BEGIN
48         UPDATE [Cart Details]
49         SET Quantity = Quantity + @Quantity
50         where CartID = @CartID and ProductID = @ProductID
51     end
52
53
54     UPDATE Storage
55     SET Stock = @RealQuantity - @Quantity
56     WHERE ProductID = @ProductID
57
58     END
59
60 ELSE
61 BEGIN
62     SET @error = 'Brak wystarczającej ilości produktów w magazynie';
63     RAISERROR(@error, 16,1);
64     RETURN;
65 END
66
67 GO
```

Kod generujący procedurę *Add To Cart*

5.7 Delete From Cart

Procedura służąca do usuwania produktów z koszyka.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS dbo.DeleteFromCart;
5 GO
6
7 CREATE PROC dbo.DeleteFromCart
8 @CustomerID INT = NULL,
9 @ProductID INT = NULL,
10 @Quantity INT = NULL
11
12 AS
13
14 DECLARE @error AS NVARCHAR(400);
15
16 IF @CustomerID IS NULL OR @ProductID IS NULL OR @Quantity IS NULL
17 BEGIN
18     SET @error = 'Error!';
19     RAISERROR(@error, 16,1);
20     RETURN;
21 END
22
23 DECLARE @CartID AS INT;
24
25 SELECT @CartID = CartID
26 FROM Carts
27 WHERE CustomerID = @CustomerID
28
29 IF NOT EXISTS (Select * From [cart details] where CartID = @CartID and ProductID = @ProductID)
30 BEGIN
31     SET @error = 'Nie ma takiego produktu w koszyku!';
32     RAISERROR(@error, 16,1);
33     RETURN;
34 END
35
36 DECLARE @QuantityInCart AS INT
37
38
39 Select @QuantityInCart = Quantity FROM [cart details]
40 WHERE ProductID = @ProductID and CartID = @CartID;
41
42
43 IF @QuantityInCart >= @Quantity
44 BEGIN
45     IF @QuantityInCart > @Quantity
46     BEGIN
47         UPDATE [Cart Details]
48         SET Quantity = Quantity - @Quantity
49         WHERE ProductID = @ProductID and CartID = @CartID;
50     end
51 ELSE
52 BEGIN
53     DELETE FROM [Cart Details]
54     WHERE ProductID = @ProductID and CartID = @CartID;
55     END
56
57 UPDATE Storage
58 SET Stock = Stock + @Quantity
59 WHERE ProductID = @ProductID
60 END
61
62 ELSE
63 BEGIN
64     SET @error = 'Nie masz aż tylu produktów w koszyku!';
65     RAISERROR(@error, 16,1);
66     RETURN;
67 END
68
69 GO
```

Kod generujący procedurę *Delete From Cart*

5.8 Buy

Procedura pozwalająca zamówić produkty będące w koszyku.

```
1 USE TestDB;
2 GO
3
4 DROP PROC IF EXISTS dbo.Buy;
5 GO
6
7 CREATE PROC dbo.Buy
8 @CustomerID INT = NULL
9
10 AS
11 DECLARE @EmployeeID INT
12 SELECT TOP 1 @EmployeeID = EmployeeID FROM Employees
13 WHERE DepartmentID = 4
14 ORDER BY NEWID()
15
16
17 DECLARE @CartID AS INT;
18
19 SELECT @CartID = CartID
20 FROM Carts
21 WHERE CustomerID = @CustomerID
22
23
24 DECLARE @OrderID INT
25
26
27 INSERT INTO Orders
28 Values(@CustomerID, GETDATE(), @EmployeeID)
29 SET @OrderID = @@IDENTITY
30
31
32 DECLARE @ProductID INT
33 DECLARE @Quantity INT
34
35 DECLARE Cart CURSOR
36 FOR SELECT ProductID, Quantity FROM [Cart Details]
37 WHERE CartID = @CartID
38
39 DECLARE @UnitPrice MONEY
40
41
42 OPEN Cart
43 Fetch Cart INTO @ProductID, @Quantity
44
45 WHILE @@FETCH_STATUS = 0
46 BEGIN
47
48     SELECT @UnitPrice = [Retail Price] FROM Products WHERE ProductID = @ProductID
49
50     INSERT INTO [Order Details]
51     VALUES(@OrderID, @ProductID, @UnitPrice, @Quantity)
52     Fetch Cart INTO @ProductID, @Quantity
53
54 END
55 CLOSE Cart
56 DEALLOCATE Cart
57
58
59 DELETE FROM [Cart Details]
60 WHERE CartID = @CartID
61
62 DELETE FROM Carts
63 WHERE CartID = @CartID
64
65 INSERT INTO Carts(CustomerID)
66 VALUES (@CustomerID)
67
68 GO
```

Kod generujący procedurę *Buy*

Rozdział 6

Funkcje

6.1 Generate Note

Funkcja przyjmująca jako argument *@OrderID* zamówienia, tworząca fakturę na podstawie tego zamówienia.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[GenerateNote];
5 GO
6
7 CREATE FUNCTION dbo.GenerateNote (@OrderID INT)
8 RETURNS @Note TABLE(
9     OrderID INT,
10    ClientName NVARCHAR(40),
11    Amout MONEY,
12    OrderDate Date,
13    [Number of Products] INT
14 )
15 AS
16 BEGIN
17
18 DECLARE @CustomerName NVARCHAR(40)
19
20 SELECT @CustomerName = [Name] from Customers JOIN Orders
21 ON Orders.CustomerID = Customers.CustomerID
22 WHERE Orders.OrderID = @OrderID
23
24 DECLARE @Amount MONEY
25 SELECT @Amount = SUM(UnitPrice * Quantity) FROM [Order Details]
26 WHERE OrderID = @OrderID
27
28 DECLARE @OrderDate Date
29 SELECT @OrderDate = [Order Date] from Orders
30 WHERE OrderID = @OrderID
31
32 DECLARE @Quantity INT
33 SELECT @Quantity = SUM(Quantity) FROM [Order Details]
34 WHERE OrderID = @OrderID
35
36
37 INSERT INTO @Note
38 VALUES(@OrderID, @CustomerName, @Amount, @OrderDate, @Quantity)
39
40 RETURN
41 END
42
43 GO
```

Kod generujący funkcję *Generate Note*

6.2 Get Reviews

Funkcja przyjmująca jako argument *@ProductID* produktu, zwracająca recenzje na jego temat.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[Get Reviews];
5 GO
6
7 CREATE FUNCTION [dbo].[Get Reviews] (@ProductID INT)
8 RETURNS TABLE
9 AS
10 RETURN
11 (
12     SELECT [Content] AS [Review], [Rating]
13     FROM [dbo].[Reviews]
14     WHERE ProductID = @ProductID
15 )
16
17 GO
```

Kod generujący funkcję *Get Reviews*

6.3 Get Review Ratings

Funkcja przyjmująca jako argument *@ReviewID* recenzji, zwracająca jej oceny.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[Get Review Ratings];
5 GO
6
7 CREATE FUNCTION [dbo].[Get Review Ratings] (@ReviewID INT)
8 RETURNS @Ratings TABLE(
9     ReviewID INT,
10    Pluses INT,
11    Minuses INT
12 )
13 AS
14 BEGIN
15     DECLARE @Pluses INT
16     SELECT @Pluses = COUNT(*) FROM [Review Ratings]
17     WHERE ReviewID = @ReviewID AND [Plus/Minus] = '+'
18
19     DECLARE @Minuses INT
20     SELECT @Minuses = COUNT(*) FROM [Review Ratings]
21     WHERE ReviewID = @ReviewID AND [Plus/Minus] = '-'
22
23     INSERT INTO @Ratings VALUES(@ReviewID, @Pluses, @Minuses)
24
25     RETURN
26 END
27
28 GO
```

Kod generujący funkcję *Get Review Ratings*

6.4 Avg. Product Rate

Funkcja przyjmująca jako argument *@ProductID* produktu, zwracająca jego średnią ocenę.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[AvgProductRate];
5 GO
6
7 CREATE FUNCTION dbo.AvgProductRate (@ProductID INT)
8 RETURNS FLOAT
9 AS
10 BEGIN
11
12 DECLARE @AvgRate FLOAT
13
14
15 SELECT @AvgRate = CAST(AVG(cast(rating as float)) AS DECIMAL(3,2)) from Reviews
16 WHERE ProductID = @ProductID
17
18
19
20 RETURN @AvgRate
21 END
22
23 GO
```

Kod generujący funkcję *Avg. Product Rate*

6.5 Product Bought

Funkcja przyjmująca jako argument *@ProductName* nazwę produktu, zwracająca informację o ilości sprzedanych sztuk produktu.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[Product Bought];
5 GO
6
7 CREATE FUNCTION dbo.[Product Bought] (@ProductName NVARCHAR(40) = '')
8 RETURNS TABLE
9 AS
10 RETURN
11 (
12     SELECT *
13     FROM dbo.[Products Bought] AS P
14     WHERE (P.[Product Name] LIKE @ProductName)
15 )
16
17 GO
```

Kod generujący funkcję *Product Bought*

6.6 Search

Funkcja służąca do precyzyjnego wyszukiwania produktu przyjmująca jako argumenty:

1. Nazwę Marki Produktu *@brandName*
2. Nazwę Kategorii Produktu *@CategoryName*
3. Nazwę Podkategorii Produktu *@Subcategory*
4. Dolny Zakres Cenowy *@MinPrice*
5. Górny Zakres Cenowy *@MaxPrice*

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[Search];
5 GO
6
7 CREATE FUNCTION dbo.Search (@BrandName NVARCHAR(20) = '%', @CategoryName NVARCHAR(20) = '%',
8     @Subcategory NVARCHAR(40) = '%',
9     @MinPrice INT = 0, @MaxPrice INT = 10000)
10 RETURNS TABLE
11 AS
12 RETURN
13 (
14     SELECT P.[Product Name], P.[Product Description], B.[Brand Name], C.[Category Name], S.[
15         Subcategory Name], P.[Retail Price] AS Price
16 FROM Products AS P
17 JOIN Brands AS B
18 ON P.BrandID = B.BrandID
19 JOIN Categories AS C
20 ON P.CategoryID = C.CategoryID
21 JOIN Subcategories AS S
22 ON P.SubcategoryID = S.SubcategoryID
23 WHERE ( [Brand Name] LIKE @BrandName AND
24     [Category Name] LIKE @CategoryName AND
25     [Subcategory Name] LIKE @Subcategory AND
26     [Retail Price] >= @MinPrice AND
27     [Retail Price] <= @MaxPrice
28 )
29 )
30 GO
31
32 -- SELECT * FROM Search(DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT )
```

Kod generujący funkcję *Search*

6.7 Product Lookup

Funkcja służąca do wyszukiwania produktu na podstawie małej ilości informacji.

Przyjmuje argument *@ProductName* który może być prefiksem Nazwy Produktu, lub Marki.

```
1 USE TestDB;
2 GO
3
4 DROP FUNCTION IF EXISTS [dbo].[Product Lookup];
5 GO
6
7 CREATE FUNCTION dbo.[Product Lookup] (@ProductName NVARCHAR(40) = '')
8 RETURNS TABLE
9 AS
10 RETURN
11 (
12     SELECT Pr.[Product Name], Br.[Brand Name] AS [Brand], Ss.[Stock] AS [Availability]
13     FROM dbo.[Storage Status] AS Ss
14     JOIN dbo.[Products] AS Pr
15         ON SS.[ProductID] = Pr.[ProductID]
16     JOIN dbo.[Brands] AS Br
17         ON Pr.[BrandID] = Br.[BrandID]
18     WHERE Ss.[Product Name] LIKE @ProductName + '%'
19         OR Br.[Brand Name] LIKE @ProductName + '%'
20 )
21
22 GO
```

Kod generujący funkcję *Product Lookup*

Rozdział 7

Wyzwalacze

7.1 Assign Brand ID

Wyzwalacz automatycznie dodający nowe Marki do Bazy Danych, oraz przypisujący je dodawanym produktom.

```
1 USE TestDB;
2 GO
3
4 DROP TRIGGER IF EXISTS [dbo].[TR_AssignBrandID];
5 GO
6
7 CREATE TRIGGER [dbo].[TR_AssignBrandID]
8     ON [dbo].[New Products]
9     INSTEAD OF INSERT
10 AS
11 BEGIN
12
13     -- add new Brands to Database
14     INSERT INTO [dbo].[Brands] ([Brand Name])
15     SELECT DISTINCT I.[Brand Name]
16     FROM INSERTED AS I
17     LEFT JOIN [dbo].[Brands] AS B ON I.[Brand Name] = B.[Brand Name]
18     WHERE B.[Brand Name] IS NULL
19
20     -- add new Products with according linking to Brands
21     INSERT INTO [dbo].[Products] ([Product Name], [BrandID]
22     , [Product Description], [CategoryID], [SubcategoryID]
23     , [Wholesale Price], [Retail Price])
24     SELECT I.[Product Name], B.[BrandID], I.[Product Description], I.[CategoryID], I.[
25     SubcategoryID], I.[Wholesale Price], I.[Retail Price]
26     FROM INSERTED AS I
27     JOIN [dbo].[Brands] AS B ON I.[Brand Name] = B.[Brand Name]
28 END
29
30 GO
```

Kod generujący wyzwalacz *Assign Brand ID*

7.2 Confirmation Email

Wyzwalacz automatycznie wysyłający nowym użytkownikom Maila z prośbą o potwierdzenia aktywacji konta.

```
1 USE TestDB;
2 GO
3
4 DROP TRIGGER IF EXISTS [dbo].[Confirmation Email];
5 GO
6
7 CREATE TRIGGER [dbo].[Confirmation Email]
8 ON [dbo].[Customers]
9 AFTER INSERT
10 AS
11 IF (@@ROWCOUNT = 0)
12 RETURN;
13
14 DECLARE @TopicContent VARCHAR(50) = 'Witamy w Sklepie!';
15 DECLARE @BodyContent VARCHAR(MAX) =
16 'Jeszcze tylko jeden krok i możesz cieszyć się zakupami!' + CHAR(13)
17 + 'Wystarczy że potwierdzisz swój adres email.' + CHAR(13)
18 + 'Pozdrawiamy,' + CHAR(13)
19 + 'Ekipa Twojego Sklepu';
20
21 DECLARE c CURSOR
22 FOR SELECT [CustomerID] FROM INSERTED
23 WHERE [Mailing List] = 1 AND [Account Confirmed] = 0
24
25 DECLARE @cID INT
26
27 OPEN c
28 FETCH c INTO @cID
29
30 WHILE @@FETCH_STATUS = 0
31 BEGIN
32 EXEC [dbo].[New Mail] @cID, @TopicContent, @BodyContent
33
34 FETCH c INTO @cID
35 END
36
37 CLOSE c
38 DEALLOCATE c
39
40
41 GO
```

Kod generujący wyzwalacz *Confirmation Email*

7.3 Guard Categories

Wyzwalacz powstrzymujący modyfikacje lub usunięcie kluczowych elementów Bazy Danych.

```
1 USE TestDB;
2 GO
3
4 DROP TRIGGER IF EXISTS [dbo].[Guard Categories];
5 GO
6
7 CREATE TRIGGER [dbo].[Guard Categories] ON [dbo].[Categories]
8 INSTEAD OF DELETE, UPDATE
9 AS
10 IF (@@ROWCOUNT = 0)
11 RETURN;
12 RAISERROR ('Action prohibited. This incident will be reported.', 16, 0)
13 ROLLBACK
14 GO
```

Kod generujący wyzwalacz *Guard Categories*

7.4 Stock Alert

Wyzwalacz zawiadamiający personel o wyczerpaniu zapasów w magazynie.

```
1 USE TestDB;
2 GO
3
4 DROP TRIGGER IF EXISTS [dbo].[Storage Alert];
5 GO
6
7 CREATE TRIGGER [dbo].[Storage Alert]
8 ON [dbo].[Storage]
9 AFTER UPDATE
10 AS
11
12 IF NOT EXISTS (
13 SELECT *
14 FROM [dbo].[Storage]
15 WHERE [Stock] = 0)
16 RETURN;
17
18 SELECT FORMATMESSAGE('Alert! Some products are out of stock! Notify Logistics Dep.')
19 AS [System Notification];
20
21 GO
```

Kod generujący wyzwalacz *Stock Alert*

7.5 New Product Notification

Wyzwalacz zawiadamiający subskrybujących użytkowników o nowych produktach dostępnych w sklepie.

```
1 USE TestDB;
2 GO
3
4 DROP TRIGGER IF EXISTS [dbo].[New Product Notification];
5 GO
6
7 CREATE TRIGGER [dbo].[New Product Notification]
8 ON [dbo].[Products]
9 AFTER INSERT
10 AS
11 IF (@@ROWCOUNT = 0)
12 RETURN;
13
14 DECLARE @TopicContent VARCHAR(50) = 'Newsletter sklepu'
15 DECLARE @BodyContent VARCHAR(MAX) = 'Nowe produkty ędostpne już ędzi! Sprawdź!'
16
17 DECLARE c CURSOR FOR
18 SELECT [CustomerID] FROM [dbo].[Customers]
19 WHERE [Mailing List] = 1 AND [Account Confirmed] = 1
20
21 DECLARE @cID INT
22
23 OPEN c
24 FETCH c INTO @cID
25
26 WHILE @@FETCH_STATUS = 0
27 BEGIN
28     EXEC [dbo].[New Mail] @cID, @TopicContent, @BodyContent
29
30     FETCH c INTO @cID
31 END
32
33 CLOSE c
34 DEALLOCATE c
35
36 GO
```

Kod generujący wyzwalacz *New Product Notification*

Rozdział 8

Skrypt tworzący bazę danych

Kod generujący bazę danych z oprogramowaniem:

```
1  -- create DB
2  PRINT 'Creating Database...'
3  :r <path>\DB-Project\CreateDB.sql
4  GO
5  PRINT 'Database created.'
6  ,
7
8
9  PRINT 'Deploying Views...'
10 :r <path>\DB-Project\VIEW_Products_Bought.sql
11 :r <path>\DB-Project\VIEW_Storage_Status.sql
12 :r <path>\DB-Project\VIEW_Hot_Brands.sql
13 :r <path>\DB-Project\VIEW_Category_Organisation.sql
14 GO
15 PRINT 'Views deployed.'
16 ,
17
18
19 PRINT 'Deploying Procedures...'
20 :r <path>\DB-Project\PROC_Add_Customer.sql
21 :r <path>\DB-Project\PROC_Add_Review.sql
22 :r <path>\DB-Project\PROC_Add_Review_Rating.sql
23 :r <path>\DB-Project\PROC_Add_To_Cart.sql
24 :r <path>\DB-Project\PROC_Buy.sql
25 :r <path>\DB-Project\PROC_Delete_From_Cart.sql
26 :r <path>\DB-Project\PROC_New_Mail.sql
27 :r <path>\DB-Project\PROC_Confirm_Account.sql
28 GO
29 PRINT 'Procedures deployed.'
30 ,
31
32
33 PRINT 'Deploying Functions...'
34 :r <path>\DB-Project\FUNC_Generate_Note.sql
35 :r <path>\DB-Project\FUNC_Product_Bought.sql
36 :r <path>\DB-Project\FUNC_Product_Lookup.sql
37 :r <path>\DB-Project\FUNC_Get_Review_Ratings.sql
38 :r <path>\DB-Project\FUNC_Get_Reviews.sql
39 GO
40 PRINT 'Functions deployed.'
41 ,
42
43
44 PRINT 'Deploying Triggers...'
45 :r <path>\DB-Project\TRIG_AssignBrandID.sql
46 :r <path>\DB-Project\TRIG_Guard_Categories.sql
47 :r <path>\DB-Project\TRIG_New_Product_Notification.sql
48 :r <path>\DB-Project\TRIG_Confirmation_Email.sql
49 :r <path>\DB-Project\TRIG_Stock_Alert.sql
50 GO
51 PRINT 'Triggers deployed.'
52 ,
53
54
55 PRINT 'Filling Tables...'
```



```

56 :r <path>\DB-Project\FillDB.sql
57 GO
58 PRINT '
59 Tables Filled.
60 '
61 GO

```

Kod generujący bazę danych z oprogramowaniem

Kod generujący tabele:

```

1  /* CREATE DATABASE */
2  USE master;
3  GO
4
5  IF DB_ID('TestDB') IS NOT NULL
6      DROP DATABASE TestDB;
7  GO
8
9  CREATE DATABASE TestDB;
10 GO
11
12 USE TestDB;
13 GO
14
15 /* /CREATE DATABASE */
16 -----
17 /* CREATE TABLES */
18
19 -- -- /* FIRST DROP TABLES*/
20 PRINT 'Performing cleanup...'
21 GO
22
23 DROP TABLE IF EXISTS [dbo].[Brands] ;
24 GO
25
26 DROP TABLE IF EXISTS [dbo].[Cart Details] ;
27 GO
28
29 DROP TABLE IF EXISTS [dbo].[Carts] ;
30 GO
31
32 DROP TABLE IF EXISTS [dbo].[Customers] ;
33 GO
34
35 DROP TABLE IF EXISTS [dbo].[Categories] ;
36 GO
37
38 DROP TABLE IF EXISTS [dbo].[Subcategories] ;
39 GO
40
41 DROP TABLE IF EXISTS [dbo].[Products] ;
42 GO
43
44 DROP TABLE IF EXISTS [dbo].[Suppliers] ;
45 GO
46
47 DROP TABLE IF EXISTS [dbo].[Resupplies] ;
48 GO
49
50 DROP TABLE IF EXISTS [dbo].[Resupply Details] ;
51 GO
52
53 DROP TABLE IF EXISTS [dbo].[Orders] ;
54 GO
55
56 DROP TABLE IF EXISTS [dbo].[Order Details] ;
57 GO
58
59 DROP TABLE IF EXISTS [dbo].[Reviews] ;
60 GO
61
62 DROP TABLE IF EXISTS [dbo].[Review Ratings] ;
63 GO

```

```

64
65 DROP TABLE IF EXISTS [dbo].[Departments] ;
66 GO
67
68 DROP TABLE IF EXISTS [dbo].[Employees] ;
69 GO
70
71 DROP TABLE IF EXISTS [dbo].[Storage] ;
72 GO
73
74 DROP TABLE IF EXISTS [dbo].[New Products] ;
75 GO
76
77 DROP TABLE IF EXISTS [dbo].[Mail Queue] ;
78 GO
79
80
81
82 PRINT 'Done.'
83 ,
84 GO
85 -- -- /* /FIRST DROP TABLES*/
86
87 PRINT 'Creating tables...'
88 GO
89
90 -- Customers
91 CREATE TABLE [dbo].[Customers]
92 (
93     [CustomerID] INT IDENTITY(1,1) NOT NULL
94     , CONSTRAINT [PK_Customers] PRIMARY KEY (CustomerID)
95     , [Name] NVARCHAR(40) NOT NULL
96     , [Email Address] NVARCHAR(40) NOT NULL UNIQUE
97     , [Phone Number] NVARCHAR(15)
98     , [Address] NVARCHAR(40)
99     , [Postal Code] NVARCHAR(8)
100    , [City] NVARCHAR(25)
101    , [Mailing List] BIT NOT NULL DEFAULT 1
102    , [Account Confirmed] BIT NOT NULL DEFAULT 0
103 )
104 ;
105 GO
106
107 -- Mail Queue
108 CREATE TABLE [dbo].[Mail Queue]
109 (
110     [MailID] INT IDENTITY(1,1) NOT NULL
111     , CONSTRAINT [PK_Mail_Queue] PRIMARY KEY (MailID)
112     , [RecipientID] INT NOT NULL
113     , CONSTRAINT [FK_Mail_Customer] FOREIGN KEY (RecipientID)
114     REFERENCES [dbo].[Customers] (CustomerID)
115     ON DELETE CASCADE
116     ON UPDATE CASCADE
117     , [Topic] VARCHAR(50) NOT NULL
118     , [Body] VARCHAR(MAX) NOT NULL
119 )
120 ;
121 GO
122
123 -- Categories
124 CREATE TABLE [dbo].[Categories]
125 (
126     [CategoryID] TINYINT NOT NULL
127     , CONSTRAINT [PK_Categories] PRIMARY KEY (CategoryID)
128     , [Category Name] NVARCHAR(20) NOT NULL
129 )
130 ;
131 GO
132
133 -- Subcategories
134 CREATE TABLE [dbo].[Subcategories]
135 (
136     [CategoryID] TINYINT NOT NULL
137     , CONSTRAINT [FK_Subcategories_Categories] FOREIGN KEY (CategoryID)
138     REFERENCES [dbo].[Categories] (CategoryID)
139     ON DELETE CASCADE

```

```

140         ON UPDATE CASCADE
141     , [SubcategoryID] TINYINT IDENTITY(1,1) NOT NULL UNIQUE
142     , [Subcategory Name] NVARCHAR(40) NOT NULL
143     , CONSTRAINT [PK_Subcategories] PRIMARY KEY (CategoryID, SubcategoryID)
144 )
145 ;
146 GO
147
148 -- Brands
149 CREATE TABLE [dbo].[Brands]
150 (
151     [BrandID] INT IDENTITY(1,1)
152     , CONSTRAINT [PK_Brands] PRIMARY KEY (BrandID)
153     , [Brand Name] NVARCHAR(20) NOT NULL
154 )
155 ;
156 GO
157
158
159
160 -- Products
161 CREATE TABLE [dbo].[Products]
162 (
163     [ProductID] INT IDENTITY(1,1) NOT NULL
164     , CONSTRAINT [PK_Products] PRIMARY KEY (ProductID)
165     , [Product Name] NVARCHAR(40) NOT NULL
166     , [BrandID] INT NOT NULL
167     , [Product Description] NVARCHAR(100)
168     , [CategoryID] TINYINT
169     , CONSTRAINT [FK_Products_Categories] FOREIGN KEY (CategoryID)
170     REFERENCES [dbo].[Categories] (CategoryID)
171     ON DELETE SET NULL
172     ON UPDATE CASCADE
173     , [SubcategoryID] TINYINT
174     , CONSTRAINT [FK_Products_Subcategories] FOREIGN KEY (SubcategoryID)
175     REFERENCES [dbo].[Subcategories] (SubcategoryID)
176     , [Wholesale Price] MONEY NOT NULL -- TODO trigger: price >=0
177     , [Retail Price] MONEY NOT NULL -- TODO trigger
178 )
179 ;
180 GO
181
182 -- temporary table for adding new products
183 CREATE TABLE [dbo].[New Products]
184 (
185     [Brand Name] NVARCHAR(20)
186     , [Product Name] NVARCHAR(40) NOT NULL
187     , [Product Description] NVARCHAR(100)
188     , [CategoryID] TINYINT
189     , [SubcategoryID] TINYINT
190     , [Wholesale Price] MONEY NOT NULL
191     , [Retail Price] MONEY NOT NULL
192 )
193 ;
194 GO
195
196
197 -- Suppliers
198 CREATE TABLE [dbo].[Suppliers]
199 (
200     [SupplierID] INT IDENTITY(1,1) NOT NULL
201     , CONSTRAINT [PK_Suppliers] PRIMARY KEY (SupplierID)
202     , [SupplierName] NVARCHAR(40) NOT NULL
203     , [Country] NVARCHAR(20)
204     , [City] NVARCHAR(20)
205     , [Address] NVARCHAR(40)
206     , [PostalCode] NVARCHAR(6)
207     , [EmailAddress] NVARCHAR(30)
208     , [Phone] NVARCHAR(20)
209 )
210 ;
211 GO
212
213 -- Resupplies
214 CREATE TABLE [dbo].[Resupplies]
215 (

```

```

216         [ResupplyID] INT IDENTITY(1,1) NOT NULL
217     , CONSTRAINT [PK_Resupplies] PRIMARY KEY (ResupplyID)
218     , [SupplierID] INT NOT NULL
219     , CONSTRAINT [FK_Resupplies_Suppliers] FOREIGN KEY (SupplierID)
220     REFERENCES [dbo].[Suppliers] (SupplierID)
221     ON DELETE CASCADE
222     ON UPDATE CASCADE
223 )
224 ;
225 GO
226
227 -- ResupplyDetails
228 CREATE TABLE [dbo].[Resupply Details]
229 (
230     [ResupplyID] INT NOT NULL
231     , CONSTRAINT [FK_ResupplyDetails_Resupplies] FOREIGN KEY (ResupplyID)
232     REFERENCES [dbo].[Resupplies] (ResupplyID)
233     ON DELETE CASCADE
234     ON UPDATE CASCADE
235     , [ProductID] INT NOT NULL
236     , CONSTRAINT [FK_ResupplyDetails_Products] FOREIGN KEY (ProductID)
237     REFERENCES [dbo].[Products] (ProductID)
238     ON DELETE CASCADE
239     ON UPDATE CASCADE
240     , [Quantity] SMALLINT NOT NULL
241 )
242 ;
243 GO
244
245 -- Departments
246 -- Employee can be assigned to a department
247 CREATE TABLE [dbo].[Departments]
248 (
249     [DepartmentID] TINYINT NOT NULL
250     , CONSTRAINT [PK_Departments] PRIMARY KEY (DepartmentID)
251     , [DepartmentName] VARCHAR(20) NOT NULL
252 )
253 ;
254 GO
255
256 -- Employees
257 CREATE TABLE [dbo].[Employees]
258 (
259     [EmployeeID] INT IDENTITY(1,1) NOT NULL
260     , CONSTRAINT [PK_Employees] PRIMARY KEY (EmployeeID)
261     , [Name] NVARCHAR(40) NOT NULL UNIQUE
262     , [DepartmentID] TINYINT DEFAULT NULL
263     , CONSTRAINT [FK_Employees_Departments] FOREIGN KEY (DepartmentID)
264     REFERENCES [dbo].[Departments] (DepartmentID)
265     ON DELETE SET NULL
266     ON UPDATE CASCADE
267 )
268 ;
269 GO
270
271 -- Orders
272 CREATE TABLE [dbo].[Orders]
273 (
274     [OrderID] INT IDENTITY(1,1) NOT NULL
275     , CONSTRAINT [PK_Orders] PRIMARY KEY (OrderID)
276     , [CustomerID] INT
277     , CONSTRAINT [FK_Orders_Customers] FOREIGN KEY (CustomerID)
278     REFERENCES [dbo].[Customers] (CustomerID)
279     ON DELETE CASCADE
280     ON UPDATE CASCADE
281     , [Order Date] DATETIME
282     , [EmployeeID] INT
283     , CONSTRAINT [FK_Orders_Employees] FOREIGN KEY (EmployeeID)
284     REFERENCES [dbo].[Employees] (EmployeeID)
285     ON DELETE CASCADE
286     ON UPDATE CASCADE
287 )
288 ;
289 GO
290
291

```

```

292 -- Order Details
293 CREATE TABLE [dbo].[Order Details]
294 (
295     [OrderID] INT NOT NULL
296     , CONSTRAINT [FK_OrderDetails_Orders] FOREIGN KEY (OrderID)
297     REFERENCES [dbo].[Orders] (OrderID)
298     ON DELETE CASCADE
299     ON UPDATE CASCADE
300     , [ProductID] INT NOT NULL
301     , CONSTRAINT [FK_OrderDetails_Products] FOREIGN KEY (ProductID)
302     REFERENCES [dbo].[Products] (ProductID)
303     ON DELETE CASCADE
304     ON UPDATE CASCADE
305     , [UnitPrice] MONEY
306     , [Quantity] SMALLINT NOT NULL
307     , CONSTRAINT [PK_OrderDetails] PRIMARY KEY (OrderID, ProductID)
308 )
309 ;
310 GO
311
312 -- Reviews
313 CREATE TABLE [dbo].[Reviews]
314 (
315     [ReviewID] INT IDENTITY(1,1) NOT NULL UNIQUE
316     , [CustomerID] INT NOT NULL
317     , CONSTRAINT [FK_Reviews_Customers] FOREIGN KEY (CustomerID)
318     REFERENCES [dbo].[Customers] (CustomerID)
319     ON DELETE CASCADE
320     ON UPDATE CASCADE
321     , [ProductID] INT NOT NULL
322     , CONSTRAINT [FK_Reviews_Products] FOREIGN KEY (ProductID)
323     REFERENCES [dbo].[Products] (ProductID)
324     ON DELETE CASCADE
325     ON UPDATE CASCADE
326     , [Content] NTEXT
327     , [Rating] TINYINT NOT NULL
328     , CONSTRAINT [CK_Reviews_ValidRating] CHECK
329     (Rating >= 1 AND Rating <= 5)
330     , CONSTRAINT [PK_Reviews] PRIMARY KEY (CustomerID, ProductID)
331 )
332 ;
333 GO
334
335 -- Review Ratings
336 -- customer can 'thumbUp' a review to mark it as helpful
337 CREATE TABLE [dbo].[Review Ratings]
338 (
339     [ReviewID] INT NOT NULL
340     , CONSTRAINT [FK_ReviewRatings_Reviews] FOREIGN KEY (ReviewID)
341     REFERENCES [dbo].[Reviews] (ReviewID)
342     ON DELETE CASCADE
343     ON UPDATE CASCADE
344     , [CustomerID] INT NOT NULL
345     , CONSTRAINT [FK_ReviewRatings_Customers] FOREIGN KEY (CustomerID)
346     REFERENCES [dbo].[Customers] (CustomerID),
347     [Plus/Minus] CHAR(1)
348 )
349 ;
350 GO
351
352 -- Storage
353 -- every item has to be stored in Storage
354 CREATE TABLE [dbo].[Storage]
355 (
356     -- every product has only one spot in storage
357     [ProductID] INT NOT NULL UNIQUE
358     , CONSTRAINT [FK_Storage_Products] FOREIGN KEY (ProductID)
359     REFERENCES [dbo].[Products] (ProductID)
360     ON DELETE CASCADE
361     ON UPDATE CASCADE
362     , [StockMax] INT NOT NULL
363     , CONSTRAINT [CK_Storage_StockMaxValid] CHECK ( StockMax > 0 ) -- TODO trigger
364     , [Stock] INT NOT NULL
365     , CONSTRAINT [CK_Storage_StockValid] CHECK ( Stock BETWEEN 0 AND StockMax ) -- TODO
366     trigger

```

```

367 )
368
369 ;
370 GO
371
372 -- Carts
373 -- every user, while shopping, adds items to Cart.
374 -- Orders are made by ordering what is currently in a Cart
375 CREATE TABLE [dbo].[Carts]
376 (
377     [CartID] INT IDENTITY(1,1)
378     , CONSTRAINT [PK_Carts] PRIMARY KEY (CartID)
379     -- every customer can have only one cart at the time
380     , [CustomerID] INT NOT NULL UNIQUE
381     , CONSTRAINT [FK_Carts_Customers] FOREIGN KEY (CustomerID)
382     REFERENCES [dbo].[Customers] (CustomerID)
383     ON DELETE CASCADE
384     ON UPDATE CASCADE
385 )
386 ;
387 GO
388
389 -- Cart Details
390 CREATE TABLE [dbo].[Cart Details]
391 (
392     [CartID] INT NOT NULL
393     , CONSTRAINT [FK_CartDetails_Carts] FOREIGN KEY (CartID)
394     REFERENCES [dbo].[Carts] (CartID)
395     ON DELETE CASCADE
396     ON UPDATE CASCADE
397     , [ProductID] INT NOT NULL
398     , CONSTRAINT [FK_CartDetails_Products] FOREIGN KEY (ProductID)
399     REFERENCES [dbo].[Products] (ProductID)
400     ON DELETE CASCADE
401     ON UPDATE CASCADE
402     , CONSTRAINT [PK_CartDetails] PRIMARY KEY (CartID, ProductID)
403     , [Quantity] INT NOT NULL
404 )
405 ;
406 GO
407
408 PRINT 'Done.'
409
410 GO
411 /* /CREATE TABLES */
412 -----

```

Kod generujący tabele

Kod wypełniający tabele:

```

1  /* FILL TABLES */
2
3  SET NOCOUNT ON
4
5  PRINT 'Adding Customers...'
6      EXEC dbo.AddCustomer @Name = 'Celina Johansen', @Email = 'celinajohansen@gmail.com',
7      @PhoneNumber = '123456789',
8      @Address = 'Skarbowa 16', @PostalCode = '30-056', @City = 'Krakow'
9  GO
10
11 EXEC dbo.AddCustomer @Name = 'Cyryl Jensen', @Email = 'cJensen@gmail.com', @PhoneNumber = '
12 222333444',
13 @Address = 'Lipowa 18/2', @PostalCode = '12-819', @City = 'Warszawa'
14 GO
15
16 EXEC dbo.AddCustomer @Name = 'Cameron Moller', @Email = 'mollercameron@wp.pl', @PhoneNumber = '
17 367235980',
18 @Address = 'Lea 13', @PostalCode = '26-600', @City = 'Radom'
19 GO
20
21 EXEC dbo.AddCustomer @Name = 'Cyryl Andersen', @Email = 'candersen@wp.pl', @PhoneNumber = '
22 456789333',
23 @Address = 'Wesola 13', @PostalCode = '33-222', @City = 'Katowice'
24 GO

```

```

20
21 PRINT 'Done.'
22 GO
23
24
25 PRINT 'Confirming some Customers...'
26 EXEC dbo.[Confirm Account] 1
27 PRINT 'Done.'
28 GO
29
30
31 PRINT 'Adding Categories...'
32 INSERT INTO [dbo].[Categories] VALUES
33     (1, 'Twarz')
34     , (2, 'Usta')
35     , (3, 'Oczy')
36     , (4, 'Brwi')
37     , (5, 'Paznokcie')
38     , (6, 'Akcesoria')
39     , (7, 'Dermokosmetyki')
40     , (8, 'Zestawy')
41 ;
42 PRINT 'Done.'
43 '
44 GO
45
46 PRINT 'Adding Subcategories...'
47 INSERT INTO [dbo].[Subcategories] VALUES
48 -- Twarz
49     (1, 'ŁPodkady')
50     , (1, 'Pudry')
51     , (1, 'Korektory')
52     , (1, 'ŚRozwielacze i bronzery')
53     , (1, 'Róże')
54     , (1, 'Kremy BB i CC')
55     , (1, 'Bazy pod makijaż')
56     , (1, 'Spraye utrwalające makijaż')
57     , (1, 'ęPdźle do makijażu')
58     , (1, 'Akcesoria')
59 -- Usta
60     , (2, 'ŁByszczki')
61     , (2, 'Pomadki do ust')
62     , (2, 'Konturówki do ust')
63     , (2, 'ęPdźle do ust')
64 -- Oczy
65     , (3, 'Tusze do ęrsz')
66     , (3, 'Bazy i cienie do powiek')
67     , (3, 'Paletki cieni')
68     , (3, 'Kredki do oczu i eyelinery')
69     , (3, 'Sztuczne ęrszy')
70     , (3, 'Odżywki do ęrsz')
71     , (3, 'ęPdźle do oczu')
72     , (3, 'Akcesoria')
73 -- Brwi
74     , (4, 'Kredki do brwi')
75     , (4, 'Cienie')
76     , (4, 'Korektory do brwi')
77     , (4, 'Maskary, żełe, woski, pomady, henna')
78     , (4, 'ęPdźle do brwi')
79     , (4, 'Akcesoria')
80 -- Paznokcie
81     , (5, 'Lakiery do paznokci')
82     , (5, 'Lakiery hybrydowe')
83     , (5, 'ęPielgnacja')
84     , (5, 'Zmywacze')
85     , (5, 'Zdobienie paznokci')
86     , (5, 'Bazy i utwardzanie, top coat')
87     , (5, 'Sztuczne paznokcie')
88     , (5, 'Akcesoria')
89 -- Akcesoria
90     , (6, 'Do twarzy')
91     , (6, 'Do ust')
92     , (6, 'Do oczu')
93     , (6, 'Do paznokci')
94     , (6, 'ęPdźle do makijażu')
95     , (6, 'Kosmetyczki')

```

```

96 -- Dermokosmetyki nie mają podkategorii
97 -- Zestawy też nie mają podkategorii
98 ;
99 PRINT 'Done.
100 '
101 GO
102
103 PRINT 'Adding Brands...'
104 INSERT INTO [dbo].[Brands] VALUES
105 ('Hello Kitty') -- test
106 PRINT 'Done.
107 '
108 GO
109
110 PRINT 'Adding Products...'
111 INSERT INTO [dbo].[New Products] VALUES
112 ('Revlon', 'Colorstay', 'Łpodkad z pompką do cery łtustej i mieszanej, 30 ml', 1, 1, 24,
113 48.99)
114 , ('Gosh', 'X-CEPTIONAL WEAR', 'kryjący łpodkad do twarzy w kremie do twarzy, 35 ml', 1, 1, 21,
115 41.99)
116 , ('Anabelle Minerals', 'GOLDEN FAIREST', 'Łpodkad matujący do twarzy, 4 g', 1, 1, 22, 44.99)
117 , ('Catrice', 'HD LIQUID COVERAGE', 'Łpodkad do twarzy, 30 ml', 1, 1, 10, 23.19)
118 , ('Eveline', 'Matt Magic Lip Cream', ' pomadka w łpynie do ust', 2, 2, 13, 19.99)
119 , ('Revolution Makeup', 'Iconic Division', 'Paleta cieni do powiek', 3,3, 20, 29.99)
120 , ('Eveline', 'Eyebrow Pencil', 'kredka do brwi', 4,1,8, 11.50)
121 , ('Glov', 'On the go', 'Łerkawica do makijażu', 6,1,30, 39.99)
122 ;
123 PRINT 'Done.
124 '
125 GO
126
127 PRINT 'Adding Departments...'
128 INSERT INTO [dbo].[Departments] VALUES
129 (1, 'Management')
130 , (2, 'IT')
131 , (3, 'Sales')
132 , (4, 'Packaging')
133 ;
134 PRINT 'Done.
135 '
136 GO
137
138 PRINT 'Adding Employees...'
139 INSERT INTO [dbo].[Employees] VALUES
140 ('Eryk Sorensen', 4)
141 , ('Elif Jensen', 3)
142 , ('Eliza Moller', 2)
143 , ('Elisabeth Christiansen', 1)
144 , ('Edward Larsen', 1)
145 , ('Emanuel Poulsen', 3)
146 , ('Eugeniusz Johansen', 3)
147 , ('Eryk Pedersen', 4)
148 , ('Elif Olsen', 4)
149 , ('Elzna Jorgensen', 4)
150 ;
151 PRINT 'Done.
152 '
153 GO
154
155 PRINT 'Adding Storage...'
156 INSERT INTO [dbo].[Storage] VALUES
157 (1, 1000, 100)
158 , (2, 1000, 500)
159 , (3, 1000, 200)
160 , (4, 1000, 300)
161 , (5, 1000, 500)
162 , (6, 500, 100)
163 , (7, 1000, 600)
164 , (8, 1000, 200)
165 ;
166 PRINT 'Done.
167 '
168 GO
169 PRINT 'Adding some products to carts'

```



```

170
171 EXEC dbo.AddToCart @CustomerID = 1, @ProductID = 1, @Quantity = 2
172 GO
173
174 EXEC dbo.AddToCart @CustomerID = 1, @ProductID = 2, @Quantity = 3
175 GO
176
177 EXEC dbo.AddToCart @CustomerID = 3, @ProductID = 6, @Quantity = 5
178 GO
179
180 EXEC dbo.AddToCart @CustomerID = 2, @ProductID = 2, @Quantity = 1
181 GO
182
183 EXEC dbo.AddToCart @CustomerID = 1, @ProductID = 3, @Quantity = 2
184 GO
185
186 EXEC dbo.AddToCart @CustomerID = 2, @ProductID = 5, @Quantity = 3
187 GO
188
189 EXEC dbo.AddToCart @CustomerID = 4, @ProductID = 6, @Quantity = 4
190 GO
191
192 EXEC dbo.AddToCart @CustomerID = 3, @ProductID = 1, @Quantity = 1
193 GO
194
195 EXEC dbo.AddToCart @CustomerID = 3, @ProductID = 3, @Quantity = 3
196 GO
197
198 EXEC dbo.AddToCart @CustomerID = 4, @ProductID = 5, @Quantity = 1
199 GO
200
201 EXEC dbo.AddToCart @CustomerID = 4, @ProductID = 2, @Quantity = 3
202 GO
203
204
205 PRINT 'Done.'
206
207
208 PRINT 'Buying products...'
209 EXEC dbo.Buy @CustomerID = 1
210 GO
211 EXEC dbo.Buy @CustomerID = 2
212 GO
213 /*EXEC dbo.Buy @CustomerID = 3
214 GO*/
215
216 PRINT 'Done.'
217
218 EXEC dbo.AddToCart @CustomerID = 1, @ProductID = 5, @Quantity = 2
219 GO
220
221 PRINT 'Adding Reviews ...'
222 EXEC dbo.AddReview @CustomerID = 2, @ProductID = 1, @Content = 'Super, Polecam!', @Rating = 5
223 GO
224 EXEC dbo.AddReview @CustomerID = 3, @ProductID = 2, @Content = 'Może być.', @Rating = 3
225 GO
226 EXEC dbo.AddReview @CustomerID = 3, @ProductID = 1, @Content = 'Nawet, nawet', @Rating = 3
227 GO
228
229 PRINT 'Done.'
230
231 PRINT 'Adding Reviews Ratings ...'
232 EXEC dbo.AddReviewRating @CustomerID = 1, @ReviewID = 1, @PlusMinus = '+'
233 GO
234 EXEC dbo.AddReviewRating @CustomerID = 3, @ReviewID = 1, @PlusMinus = '-'
235
236 PRINT 'Done.'
237
238 PRINT 'Adding some Suppliers ...'
239 INSERT INTO dbo.Suppliers VALUES
240 ('BestCosmetics', 'Polska', 'Warszawa', 'ul. Łwesoła 25', '05-077', 'bestcosmetics@gmail.com', '
561321673' )
241 ,('MakeUp for you', 'USA', 'Nowy York', 'Kings Avenue 33', '11-222', 'makeupforyou@mufy.com', '
222333444' )
242 ,('Look good', 'Wielka Brytania', 'Londyn', 'Baker Street 29', '33-123', 'LookingGood@gmail.com
', '321765089' )

```

```

243 ;
244 PRINT 'Done.'
245
246 PRINT 'Adding some Resupplies ...'
247 INSERT INTO dbo.Resupplies VALUES
248 (3)
249 ,(2)
250 ;
251 PRINT 'Done.'
252
253 PRINT 'Adding some Resupplies Details ...'
254 INSERT INTO dbo.[Resupply Details] VALUES
255 (1, 5, 100)
256 ,(1, 2, 150)
257 ,(1, 3, 50)
258 ,(1, 8, 100)
259 ,(2, 4, 50)
260 ,(2, 1, 100)
261 ,(2, 5, 120)
262 ,(2, 6, 70)
263 ;
264
265 PRINT 'Done.'
266
267 SET NOCOUNT OFF
268
269
270 GO
271 /* /FILL TABLES */
272 -----

```

Kod wypełniający tabele

Rozdział 9

Typowe zapytania

```
1  -- proste wyszukiwanie produktu przez użytkownika na podstawie części nazwy
2  EXEC [dbo].[Product Lookup] 'Oriflame'
3  EXEC [dbo].[Product Lookup] 'Orif'
4  EXEC [dbo].[Product Lookup] 'Av'
5  EXEC [dbo].[Product Lookup] 'Revlon'
6
7
8  -- dokładniejsze wyszukiwanie na podstawie ścisłych parametrów
9  EXEC [dbo].[Search](DEFAULT, 'Paznokcie', DEFAULT, DEFAULT, 20.00)
10 EXEC [dbo].[Search](DEFAULT, 'Maseczki', 'Peelingi', 20.00, 40.00)
11
12
13 -- sprawdzenie popularnych w sklepie
14 SELECT * FROM [dbo].[Hot Brands]
15
16
17 -- dodawanie do koszyka
18 EXEC [dbo].[Add To Cart] 1 1 1
19 EXEC [dbo].[Add To Cart] 1 2 2
20 EXEC [dbo].[Add To Cart] 1 3 4
21
22
23 -- zakup
24 EXEC [dbo].[Buy] 1
25 EXEC [dbo].[Buy] 14
26
27
28 -- generowanie faktury dla danego zamówienia
29 EXEC [dbo].[Generate Note] 1
30 EXEC [dbo].[Generate Note] 3
31 EXEC [dbo].[Generate Note] 4
32
33
34 -- pobranie przez aplikację webową recenzji na temat produktu oraz gwiazdek dla recenzji do
   zrenderowania strony
35 SELECT * FROM [dbo].[Get Reviews] 1
36 SELECT * FROM [dbo].[Get Review Ratings] 1
37
38 SELECT * FROM [dbo].[Get Reviews] 2
39 SELECT * FROM [dbo].[Get Review Ratings] 4
40
41 SELECT * FROM [dbo].[Get Reviews] 3
42 SELECT * FROM [dbo].[Get Review Ratings] 2
43
44 SELECT * FROM [dbo].[Get Reviews] 5
45 SELECT * FROM [dbo].[Get Review Ratings] 3
46
47
48 -- sprawdzanie przez pracowników stanu magazynu
49 SELECT * FROM [dbo].[Storage Status]
50 SELECT * FROM [dbo].[Storage Status] WHERE STOCK LIKE '0%'
```

Przykładowe zapytania