Podstawy Baz Danych Projekt: Restauracja

Sebastian Kozak Marcin Mikuła Dominik Grzesznik

2021/2022

Użytkownicy

Manager restauracji

- zarządzanie pozycjami w menu(zmienianie menu)
- generowanie raportów

Pracownicy restauracji

- przyjmowanie zamówień
- przyjmowanie rezerwacji
- udzielanie zniżek przysługujących odpowiednim klientom

Klienci indywidualni

- składanie zamówień:
    - -na miejscu
    - -na wynos
- dokonywanie rezerwacji stolików
- dostęp do informacji o przysługujących im rabatach
- dostęp do menu

Klient - firma

- składanie zamówień:
    - -na wynos
    - -na miejscu
- dokonywanie rezerwacji stolików na firmę (pracownik firmy również może złożyć zamówienie bądź rezerwację na siebie, podając imię i nazwisko)
- dostęp do menu
- dostęp do informacji na temat dostępnych dla firmy rabatów

Funkcje systemu

Menu i jego aktualizacja

- połowa menu zmienia się co 2 tygodnie
- produkty są wymieniane w obrębie kategorii
- produkty wybierane są losowo z puli tych, których nie było w ostatniej rotacji

Zamówienia

- klient może dokonać zamówienia spośród produktów aktualnej rotacji menu
- produkty z sekcji “Owoce Morza” mogą być zamówione na dany tydzień do poniedziałku, który go poprzedza, w przeciwnym razie(od wtorku), zamówienie jest złożone na tydzień następny
- klient ma dostęp do informacji o możliwości złożenia danego zamówienia

Rezerwacje

- klienci indywidualni mogą dokonywać rezerwacji stolików, tylko jeżeli dokonali wcześniej minimalnie 5-ciu zamówień, jednocześnie składają zamówienie o minimalnej wartości 50 zł
- rezerwację stolików dla firm, w dwóch opcjach: rezerwacji stolików na firmę lub rezerwację stolików dla konkretnych pracowników firmy (imiennie)
- rezerwacja stolika może być dokonana maksymalnie na dwa tygodnie do przodu, z powodu zmian w menu i warunku z punktu wyżej

Stoliki

- każdy stolik ma własną pojemność (od 2 do 6)
- wybór rozmiaru stolika jest zależny od ilości klientów, na których dokonana jest rezerwacja(jeśli 3 osoby, nie ma potrzeby na stolik 6 - miejscowy)

Rabaty

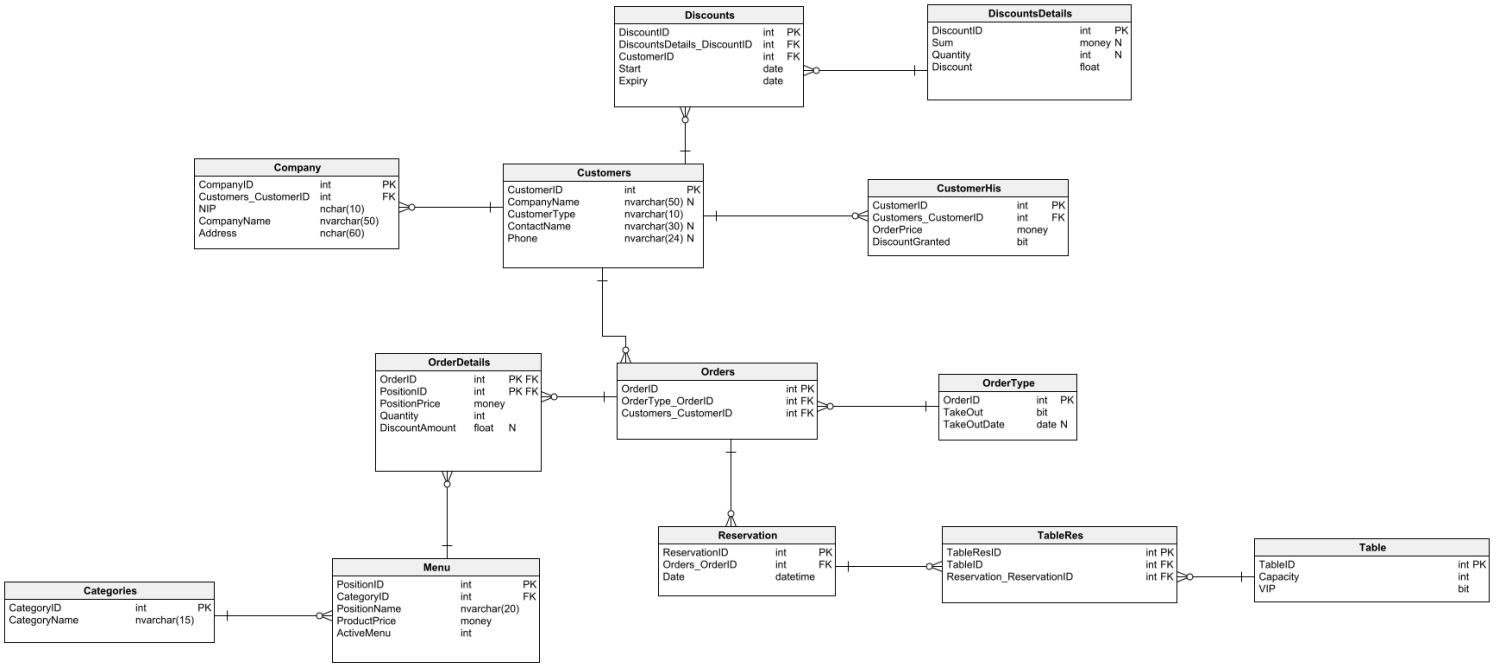
- **Jednorazowy** - przyznawany po realizacji zamówień za łączną kwotę 1000 zł zniżka 10% na zamówienia złożone przez 7 dni , począwszy od dnia przyznania zniżki
- **Stały -** przyznawany po realizacji 30 zamówień za co najmniej 60 zł, obejmuje 4% zniżki na wszystkie zamówienia od momentu przyznania

Jeśli klientowi będą przysługiwać obie zniżki, dostaje on korzystniejszą.

Raporty

System umożliwia generowanie raportów miesięcznych i tygodniowych, dotyczących rezerwacji stolików, rabatów, menu, a także statystyk zamówienia – dla klientów indywidualnych oraz firm – dotyczących kwot oraz czasu składania zamówień.

DIAGRAM BAZY DANYCH RESTAURACJI



Vertabelo

TABELE

Tabela Categories

tabela w której jeden rekord odpowiada jednej kategorii. Każda kategoria ma swoją nazwę, oraz unikalny klucz.

```
CREATE TABLE Categories (
    CategoryID nchar(5)  NOT NULL,
    CategoryName nvarchar(15)  NOT NULL,
    CONSTRAINT Categories_pk PRIMARY KEY  (CategoryID)
);
```

Tabela Company

tabela zawiera dane firmy składającej zamówienie. Pole **CompanyID** to klucz łączący z tabelą Customers.

```
CREATE TABLE Company (
    CompanyID nchar(5)  NOT NULL,
    NIP nchar(10)  NOT NULL,
    CompanyName nvarchar(50)  NOT NULL,
    Address nchar(60)  NOT NULL,
    Customers_CustomerID nchar(5)  NOT NULL,
    CONSTRAINT Company_pk PRIMARY KEY  (CompanyID)
);
```

Tabela CustomerHis

tabela zawiera informację o wszystkich zamówieniach dokonanych przez danego klienta. Pole **DiscountGranted** zawiera informacje
o tym jaka zniżka została przyznana.

```
CREATE TABLE CustomerHis (
    CustomerID nchar(5)  NOT NULL,
    OrderPrice money  NOT NULL,
    DiscountGranted bit  NOT NULL,
    Customers_CustomerID nchar(5)  NOT NULL,
    CONSTRAINT CustomerHis_pk PRIMARY KEY  (CustomerID)
);
```

Tabela Customers

tabela przechowuje informację, kto złożył zamówienie. Pole **CustomerType** określa czy jest to firma, pracownik firmy, osoba z bonem od firmy czy też klient indywidualny.

```
CREATE TABLE Customers (
    CustomerID nchar(5)  NOT NULL,
    CompanyName nvarchar(40)  NOT NULL,
    CustomerType nvarchar(10)  NOT NULL,
    ContactName nvarchar(30)  NOT NULL,
    Phone nvarchar(24)  NULL,
    Discounts_DiscountID nchar(5)  NOT NULL,
    CONSTRAINT Customers_pk PRIMARY KEY  (CustomerID)
);
```

Tabela Discounts

tabela zawiera informacje o dostępnych w restauracji zniżkach. Pola **DiscountActive**, **Start** i **Expiry** wyznaczają czy jakaś zniżka jest aktywna oraz, jeżeli tak, to również okres czasu, który obejmuje.

```
CREATE TABLE Discounts (
    CustomerID nchar(5)  NOT NULL,
    DiscountID nchar(5)  NOT NULL,
    DiscountActive bit  NOT NULL,
    Start date  NOT NULL,
    Expiry date  NOT NULL,
    CONSTRAINT Discounts_pk PRIMARY KEY  (DiscountID)
);
```

Tabela DiscountDetails

tabela zawiera warunki, które muszą zostać spełnione, aby klient mógł otrzymać daną zniżkę.

```
CREATE TABLE DiscountsDetails (
    DiscountID char(5)  NOT NULL,
    Sum money  NULL,
    Quantity int  NULL check(Quantity >= 0),
    Discount float NOT NULL check(Discount between 0 and 1),
    Discounts_DiscountID nchar(5)  NOT NULL,
    CONSTRAINT DiscountsDetails_pk PRIMARY KEY  (DiscountID)
);
```

Tabela Menu

tabela zawiera wszystkie pozycje i dostępne aktualnie w rotacji oraz te z poza rotacji, zapewnia to kolumna **ActiveMenu**.

```
CREATE TABLE Menu (
    PositionID nchar(5)  NOT NULL,
    CategoryID nchar(5)  NOT NULL,
    PositionName nvarchar(20)  NOT NULL,
    ProductPrice int NOT NULL check(ProductPrice > 0),
    ActiveMenu int NOT NULL,
    Categories_CategoryID nchar(5)  NOT NULL,
    CONSTRAINT Menu_pk PRIMARY KEY  (PositionID)
);
```

Tabela OrderType

tabela zawiera informację czy zamówienie jest realizowane na miejscu czy na wynos, jeżeli na wynos pole **TakeOutDate** określa datę, oraz godzinę odbioru.

```
CREATE TABLE OrderType (
    OrderID nchar(5)  NOT NULL,
    TakeOut bit  NOT NULL,
    TakeOutDate date  NULL,
    CONSTRAINT OrderType_pk PRIMARY KEY  (OrderID)
);
```

Tabela Orders

tabela trzyma informacje o typie zamówienia dla danego klienta, w przypadku zamówień zawierających owoce morza OrderDate ma ograniczenie na kiedy najwcześniej może być zrealizowane

```
CREATE TABLE Orders (
    OrderID nchar(5)  NOT NULL,
    OrderType_OrderID nchar(5)  NOT NULL,
    Customers_CustomerID nchar(5)  NOT NULL,
    OrderDate date  NOT NULL,
    CONSTRAINT Orders_pk PRIMARY KEY  (OrderID)
);
```

Tabela Reservation

tabela zawiera informacje na temat danej rezerwacji dokonanej w restauracji.

```
CREATE TABLE Reservation (
    ReservationID nchar(5)  NOT NULL,
    CustomerID nchar(5)  NOT NULL,
    Date date  NOT NULL,
    Hours datetime  NOT NULL,
    Customers_CustomerID nchar(5)  NOT NULL,
    Orders_OrderID nchar(5)  NOT NULL,
    Table_TableID nchar(5)  NOT NULL,
    CONSTRAINT Reservation_pk PRIMARY KEY  (ReservationID)
);
```

Tabela OrderDetails

tabela zawiera szczegółowe informacje o dokonanym zamówieniu, indeks pozycji z menu, cenę, oraz ilość.

```
CREATE TABLE OrderDetails (
    OrderID nchar(5)  NOT NULL,
    PositionID nchar(5)  NOT NULL,
    PositionPrice money  NOT NULL,
    Quantity int  NOT NULL check(Quantity >= 0),
    Discount int  NOT NULL check(Discount between 0 and 1),
    Menu_PositionID nchar(5)  NOT NULL,
    Orders_OrderID nchar(5)  NOT NULL,
    CONSTRAINT OrderDetails_pk PRIMARY KEY  (OrderID,PositionID)
);
```

Tabela Table

tabela zawiera pola charakteryzujące każdy stolik, jego unikalny numer, ilość miejsc, oraz numer rezerwacji, do której jest przypisany.

```
CREATE TABLE "Table" (
    TableID nchar(5)  NOT NULL,
    Capacity int  NOT NULL check(Capacity between 2 and 6),
    Reservation_ReservationID nchar(5)  NOT NULL,
    CONSTRAINT Table_pk PRIMARY KEY  (TableID)
);
```

Tabela TableRes

tabela zawiera informacje na temat zarezerwowanego stołu. Pole **ReservationDate** zawiera datę rezerwacji, a pole **ReservationTime** przetrzymuje informację o czasie, na który dany stolik jest zarezerwowany.

```
CREATE TABLE TableRes (
    TableID nchar(5)  NOT NULL,
    ReservationDate date  NOT NULL,
    ReservationTime datetime  NOT NULL,
    Reservation_ReservationID nchar(5)  NOT NULL,
    CONSTRAINT TableRes_pk PRIMARY KEY  (TableID)
);
```

WIDOKI

Widoki wyświetlające wszystkie firmy, wszystkich klientów i klientów z voucherem od firmy.

VIEW COMPANIES

```
CREATE VIEW COMPANIES
as
select CompanyName, CustomerID, CustomerType
from Customers
where CustomerType = 'Company'
```

VIEW PRIVATE_CUSTOMERS

```
CREATE VIEW PRIVATE_CUSTOMERS
as
select CompanyName, CustomerID, CustomerType
from Customers
where CustomerType = 'Private'
```

VIEW COMPANY_EMPLOYE

VIEW COMPANY_EMPLOYEE

```
CREATE VIEW COMPANY_EMPLOYEE
as
select CompanyName, CustomerID, CustomerType
from Customers
where CustomerType = 'Voucher'
```

Widoki wyświetlający aktualne menu.

VIEW CURRENT_MENU

```
CREATE VIEW CURRENT_MENU
as
select PositionID, PositionName, CategoryName,
ProductPrice, ActiveMenu
from Menu
        INNER JOIN Categories C on C.CategoryID =
Menu.CategoryID
where ActiveMenu = 1
```

Widoki wyświetlający produkty z kategorii owoce morza.

VIEW SEA_FOOD

```
CREATE VIEW SEA_FOOD
as
select PositionID, PositionName, CategoryName,
ProductPrice, ActiveMenu
from Menu
         INNER JOIN Categories C on C.CategoryID =
Menu.CategoryID
where CategoryName = 'Seafood'
```

Widoki wyświetlający stoliki oznaczone jako Vip.

VIEW VIP_TABLES

```
CREATE VIEW VIP_TABLES
as
select TableID, Capacity, Vip
from [Table]
where Vip = 1
```

Widoki wyświetlający klientów z aktywnymi zniżkami.

VIEW DISCOUNT

```
CREATE VIEW DISCOUNT
as
select CustomerID, DiscountActive,
DiscountsDetails.Discount
from Discounts
        inner join DiscountsDetails on
Discounts.DiscountID = DiscountsDetails.DiscountID
where DiscountActive = 1
```

Widoki wyświetlający wszystkie zamówienia wszystkich klientów.

VIEW CUSTOMER_ORDERS

```
CREATE VIEW CUSTOMER_ORDERS
as
select Customers.ContactName,
       Customers.CompanyName,
       Customers_CustomerID,
       Orders.OrderID,
       OrderType.TakeOut,
       OrderType.TakeOutDate
from Orders
        inner join Customers on Customers.CustomerID =
Orders.Customers_CustomerID
        inner join OrderType on OrderType.OrderID =
Orders.OrderType_OrderID
```

Widoki wyświetlający wszystkie produkty z zamówień zrealizowanych za pomocą vouchera.

VIEW FOOD_FROM_VOUCHERS

```
CREATE VIEW FOOD_FROM_VOUCHERS
as
select C.ContactName, C.CompanyName, M.PositionID,
M.PositionName
from Orders
        INNER JOIN Customers C on C.CustomerID =
Orders.Customers_CustomerID
        INNER JOIN OrderDetails OD on Orders.OrderID =
OD.OrderID
        INNER JOIN Menu M on OD.PositionID =
M.PositionID
where CustomerType = 'Voucher'
```

VIEW PRODUCTS_SOLD_PER_YEAR

```
CREATE VIEW PRODUCTS_SOLD_PER_YEAR
as
SELECT YEAR(O.OrderDate) as
Year,M.PositionID,PositionName, Sum(Quantity) as Sold
from OrderDetails
inner join Orders O on OrderDetails.OrderID = O.OrderID
inner join Menu M on M.PositionID =
OrderDetails.PositionID
group by YEAR(O.OrderDate),M.PositionID,PositionName
```

PROCEDURY

Procedura AddCategory

dodaję nową kategorię do tablicy Categories o podanej nazwie

```
CREATE PROCEDURE [dbo].[AddCategory] @CategoryName nvarchar(15)
AS
BEGIN
    DECLARE @CategoryID int;
    SELECT @CategoryID = ISNULL(MAX(CategoryID), 0) + 1
    FROM Categories
    INSERT INTO Categories(CategoryID, CategoryName) VALUES (@CategoryID, @CategoryName)
END
go
```

Procedura uspRemoveCategory

usuwa kategorię z tablicy Categories o podanej nazwie

```
CREATE PROCEDURE [dbo].[uspRemoveCategory] @categoryName nvarchar(15)
AS
    BEGIN
        DELETE FROM Categories
        WHERE CategoryName = @categoryName
    end
go
```

Procedura uspAddNewCustomer

dodaję nowego klienta do tablicy Customers,

jeżeli klient jest powiązany z firmą, do tabeli Company dodawany jest nowy rekord z odpowiednimi danymi

```
CREATE PROCEDURE [dbo].[uspAddNewCustomer] @CompanyName nvarchar(40),
@CustomerType nvarchar(10), @ContactName nvarchar(30), @Phone nvarchar(24),
@NIP nchar(10), @Address nchar(60)

AS
    BEGIN
        DECLARE @CustomerID INT
        IF EXISTS(
            SELECT * FROM Company
            WHERE CompanyName = @CompanyName
            ) BEGIN
            SELECT @CustomerID = ISNULL(MAX(CustomerID), 0) + 1
            FROM Customers
            INSERT INTO Customers(CustomerID, CompanyName, CustomerType, ContactName, Phone)
            VALUES (@CustomerID, @CompanyName, @CustomerType, @ContactName, @Phone)
            end

        ELSE
            BEGIN
            SELECT @CustomerID = ISNULL(MAX(CustomerID), 0) + 1
            FROM Customers
            INSERT INTO Customers(CustomerID, CompanyName, CustomerType, ContactName, Phone)
            VALUES (@CustomerID, @CompanyName, @CustomerType, @ContactName, @Phone)

            DECLARE @CompanyID INT
            SELECT @CompanyID = ISNULL(MAX(CompanyID), 0) + 1
            FROM Company
            INSERT INTO Company(CompanyID, NIP, CompanyName, Address, Customers_CustomerID)
            VALUES (@CompanyID, @NIP, @CompanyName, @Address, @CustomerID)
        end

    end
go
```

Procedura uspAddPosition

dodaję nową pozycję do menu o podanej nazwie, kategorii i cenie

```
CREATE PROCEDURE [dbo].[uspAddPosition] @PositionName nvarchar(20),
@CategoryName nvarchar(15), @ProductPrice money
AS
    BEGIN
        DECLARE @CategoryID INT
        SELECT @CategoryID = CategoryID
        FROM Categories
        WHERE CategoryName = @CategoryName
        DECLARE @PositionID INT
        SELECT @PositionID = ISNULL(MAX(PositionID), 0) + 1
        FROM Menu
        INSERT INTO Menu(PositionID, CategoryID, PositionName, ProductPrice, ActiveMenu)
        VALUES (@PositionID, @CategoryID, @PositionName, @ProductPrice, 0)
end
go
```

Procedura uspRemovePosition

usuwa pozycję z menu o podanej nazwie

```
CREATE PROCEDURE [dbo].[uspRemovePosition] @PositionName nvarchar(20)
AS
    BEGIN
        DELETE FROM Menu
        WHERE PositionName = @PositionName
    end
go
```

Procedura uspAddTable

dodaję nowy stolik do tablicy Table o podanej pojemności

```
CREATE PROCEDURE [dbo].[uspAddTable] @Capacity int,
@Vip bit
AS
    BEGIN
        DECLARE @TableID INT
        SELECT @TableID = ISNULL(MAX(TableID), 0) + 1
        FROM [Table]
        INSERT INTO [Table](TableID, Capacity, Vip)
        VALUES (@TableID, @Capacity, @Vip)
end
go
```

Procedura uspRemoveTable

usuwa stolik z tablicy Table o podanym indeksie

```
CREATE PROCEDURE [dbo].[uspRemoveTable] @TableID int
AS
    BEGIN
        DELETE FROM [Table]
        WHERE TableID = @TableID
    end
go
```

Procedura AddOrder

```
CREATE Procedure [dbo].[AddOrder]
(
@CustomerID INT,
@OrderType varchar,
@Date datetime
)
AS
BEGIN
IF (@OrderType IS NULL)
BEGIN
SET @OrderType = 1
END
ELSE
BEGIN
SET @OrderType = 2
END
DECLARE @OrderId INT
SELECT @OrderId = ISNULL(MAX(OrderId), 0)+1 FROM Orders
INSERT INTO dbo."Orders"
(
OrderID,
OrderType_OrderID,
Customers_CustomerID,
OrderDate
)
VALUES
(
@OrderId,
@OrderType,
@CustomerID,
@Date
)
END
go
```

Procedura AddPositionToOrder

```
CREATE Procedure [dbo].[AddPositionToOrder] (
@PositionID SMALLINT,
@Quantity TINYINT,
@OrderID SMALLINT
)
AS
BEGIN
DECLARE @PositionPrice SMALLINT
SET @PositionPrice = (SELECT ProductPrice FROM Menu WHERE PositionID=@PositionID)
INSERT INTO dbo.OrderDetails
(
OrderID,
PositionID,
PositionPrice,
Quantity
)
VALUES
(
@OrderID,
@PositionID,
@PositionPrice,
@Quantity
)
END
go
```

Procedura AddTableToRes

```
CREATE Procedure [dbo].[AddTableToRes](
@ReservationID INT,
@TableID SMALLINT
)
AS
BEGIN
DECLARE @TableResId SMALLINT
SELECT @TableResId = ISNULL(MAX(TableResID), 0)+1 FROM TableRes

BEGIN
INSERT INTO [dbo].[TableRes]
(
TableResID,
TableID,
Reservation_ReservationID
)
VALUES
(
@TableResId,
@TableID,
@ReservationID
)
END
END
go
```

Procedura AddReservation

```
CREATE Procedure [dbo].[AddReservation]
(
@CustomerID INT,
@Table INT,
@DateReservation DATETIME,
@OrderId INT,
@OrderType varchar
)
AS
BEGIN
DECLARE @ReservationID INT
SELECT @ReservationID = ISNULL(MAX(ReservationID), 0)+1 FROM Reservation
DECLARE @orderhis INT
SELECT @orderhis=count(OrderID) FROM Orders WHERE Customers_CustomerID=@CustomerID GROUP BY Customers_CustomerID
DECLARE @occupied INT
SELECT @occupied=count(TableResID) from TableRes INNER JOIN dbo.[Reservation] ON TableRes.Reservation_ReservationID = Reservation.ReservationID
WHERE TableID=@Table and DAY(@DateReservation)=DAY(Reservation.Date)
IF(@occupied=0 and @orderhis>5)
BEGIN
INSERT INTO dbo."Reservation"
(
ReservationID,
Orders_OrderID,
Date
)
VALUES
(
@ReservationID,
@OrderId,
@DateReservation
)
EXECUTE AddTableToRes @ReservationID: @ReservationID, @TableID: @Table
EXECUTE AddOrder @CustomerID: @CustomerID, @OrderType: @OrderType, @Date: @DateReservation
END
END
go
```

FUNKCJE

Funkcja CountConstDis

wylicza jaka zniżka przysługuje klientowi o podanym indeksie

```
CREATE FUNCTION [dbo].[CountConstDis](@CustomerId int)
    RETURNS table as return
        SELECT Customers_CustomerID, (SELECT Discount FROM DiscountsDetails WHERE Sum is NULL) as dis
        FROM (
                 SELECT Customers_CustomerID, count(Orders.OrderID) AS OrdersOverMin
                 FROM (
                          SELECT Orders.OrderID, SUM(Quantity * PositionPrice) as sum
                          FROM Orders
                                   INNER JOIN OrderDetails OD on Orders.OrderID = OD.OrderID
                          GROUP BY Orders.OrderID) as OrdersSum
                          INNER JOIN Orders on Orders.OrderID = OrdersSum.OrderID
                 where sum > (SELECT MinOrderVal FROM DiscountsDetails WHERE Sum is NULL)
                 group by Customers_CustomerID) as tab
        WHERE OrdersOverMin >= (SELECT Quantity FROM DiscountsDetails WHERE Sum is NULL)
          and Customers_CustomerID = @CustomerId
go
```

Funkcja CountOne_TimeDis

wylicza czy klientowi o podanym indeksie przysługuje zniżka jednorazowa

```
CREATE FUNCTION [dbo].[CountOne_TimeDis](@CustomerId int)
    RETURNS table as return
        SELECT Customers_CustomerID, (SELECT Discount FROM DiscountsDetails WHERE Sum is NULL) as dis
        FROM (
                 SELECT Customers_CustomerID, SUM(sums) AS OrdersSum
                 FROM (
                          SELECT Orders.OrderID, SUM(Quantity * PositionPrice) as sums
                          FROM Orders
                          INNER JOIN OrderDetails OD on Orders.OrderID = OD.OrderID
                          GROUP BY Orders.OrderID) as OrdersSum
                 INNER JOIN Orders on Orders.OrderID = OrdersSum.OrderID
                 group by Customers_CustomerID) as tab
        WHERE OrdersSum >= (SELECT Sum FROM DiscountsDetails WHERE Sum is NOT NULL) and Customers_CustomerID = @CustomerId
go
```

Funkcja GetAllActivePositions

zwraca dania które obecnie znajdują się w menu

```
CREATE FUNCTION  [dbo].[GetAllActivePositions]()
    RETURNS table AS
        RETURN
        select PositionID, CategoryID, PositionName,ProductPrice,ActiveMenu from Menu
        where Menu.ActiveMenu=1
go
```

Funkcja GetAllCustomersOfCompany

zwraca wszystkich pracowników firmy o podanej nazwie

```
CREATE FUNCTION  [dbo].[GetAllCustomersOfCompany] (@companyname nvarchar(50))
    RETURNS table AS
        RETURN
        select CustomerID, CompanyName, CustomerType, ContactName, Phone from Customers
        where Customers.CompanyName=@companyname
go
```

Funkcja GetIndividualCustomerStatistics

zwraca tablicę średnich wartości wszystkich zamówień każdego klienta

```
CREATE FUNCTION [dbo].[GetIndividualCustomerStatistics] ()
    RETURNS table as return
                SELECT Customers_CustomerID, AVG(sum) AS AvgOrders
                FROM (

                        SELECT Orders.OrderID, Quantity * PositionPrice as sum
                        FROM Orders
                                INNER JOIN OrderDetails OD on Orders.OrderID = OD.OrderID
                        ) as OrdersSum

                        INNER JOIN Orders on Orders.OrderID = OrdersSum.OrderID
                group by Customers_CustomerID
go
```

Funkcja GetPositionsWithPriceHigherThan

zwraca tablicę pozycji których cena jest wyższa od podanej

```
CREATE FUNCTION  [dbo].[GetPositionsWithPriceHigherThan] (@price money)
    RETURNS table AS
        RETURN
        select PositionID, CategoryID, PositionName,ProductPrice,ActiveMenu from Menu
        where Menu.ProductPrice>@price
go
```

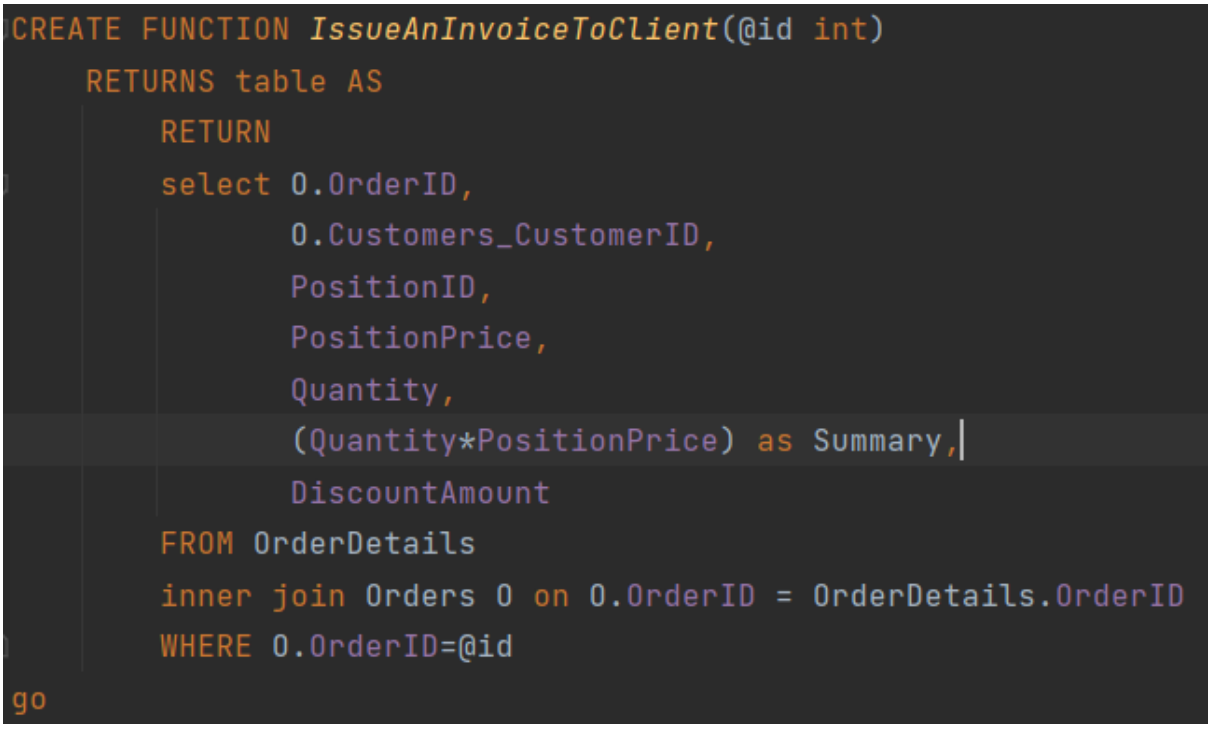
Funkcja uspGetPositionByCategory

zwraca tablicę wszystkich produktów z podanej kategorii

```
CREATE FUNCTION [dbo].[uspGetPositionByCategory](@CategoryName nvarchar(15))
    RETURNS TABLE AS
        RETURN
        SELECT PositionName, ProductPrice
        FROM Menu
                 INNER JOIN dbo.Categories C on C.CategoryID = Menu.CategoryID
        WHERE C.CategoryName = @CategoryName
go
```

Funkcja IssueAnInvoiceToClient

zwraca dane do faktury do danego zamówienia



```
CREATE FUNCTION IssueAnInvoiceToClient(@id int)
    RETURNS table AS
        RETURN
        select O.OrderID,
               O.Customers_CustomerID,
               PositionID,
               PositionPrice,
               Quantity,
               (Quantity*PositionPrice) as Summary,
               DiscountAmount
        FROM OrderDetails
        inner join Orders O on O.OrderID = OrderDetails.OrderID
        WHERE O.OrderID=@id
go
```