**Podstawy baz danych**

**Projekt Restauracja**

**Daniel Krzykawski, Przemysław Węglik, Szymon Bielówka**

1. **Rodzaje użytkowników dostępne w systemie:**

Administrator

Szef restauracji

Szef zmiany

Pracownik

Klient: indywidualny lub firmowy

1. **Funkcjonalności:**

**Menu:  
  
Klient -** Dostęp i wyświetlanie listy menu zarówno obecne jak i przyszłe

**Szef restauracji -** Edycja obecnej listy i ustalanie przyszłego menu. System trzyma w pamięci poprzednie menu i sprawdza czy nowe spełnia warunki.

**Zamówienia:  
  
Klient -** Złożenie zamówienia i dodawanie do niego produktów.

**Klient firmowy -** prośba wystawienia faktury do zamówienia.

**Pracownik, Szef zmiany, Szef restauracji** - wystawienie faktury dla danego zamówienia lub zbiorczej (raz na miesiąc).  
  
**Klient** - złożenie zamówienia na danie z owocami morza, tylko na czwartek, piątek lub sobotę; najpóźniej do poniedziałku poprzedzającego dany weekend.

**Rezerwacja miejsca:**  
**Klient indywidualny -** rezerwacja miejsca (dla co najmniej dwóch osób) wraz ze złożeniem zamówienia (nie musi być od razu, ale jest konieczne by zatwierdzić rezerwację).Klient musi mieć wcześniej co najmniej WK zamówień i zamówienie do rezerwacji musi mieć co najmniej wartość WZ. Możliwa jest płatność przy zamówieniu lub po pojawieniu się w restauracji.

**Klient firmowy -** rezerwacja stolików poprzez formularz internetowy. Można zarezerwować stolik dla całej firmy lub dla pracowników imiennie. Nie jest wymagane WZ i WK.

**Pracownik, Szef zmiany, Szef restauracji -** potwierdzenie rezerwacji stolika i wskazanie go (np. poprzez nr stolika) lub odmówienie i wysłanie zapytania czy klient jest zainteresowany jedzeniem na wynos, innym terminem.

**Rabaty:**

**Szef restauracji -** ustalanie parametrów od których zależy przyznanie rabatów (które są przydzielane automatycznie) - Z1, K1, R1, K2, R2, D1.

**Raporty:**

**Szef restauracji, Szef zmiany, Pracownik** - generowanie raportów miesięcznych i tygodniowych dotyczących rezerwacji stolików, rabatów, menu .

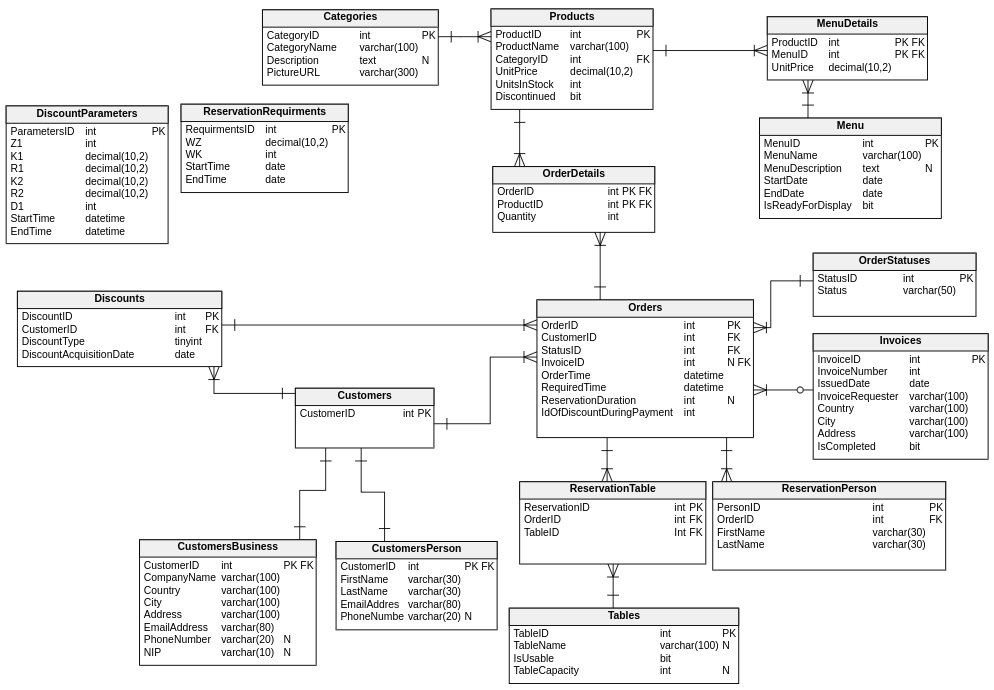
**Szef restauracji, Szef zmiany, Pracownik** - generowanie statystyk zamówienia - dla klientów indywidualnych, oraz firm - dotyczących kwoty zamówienia.

**Klient** - dostęp do swoich statystyk - zamówienia - kwoty, postęp w uzyskiwaniu rabatów. Rabat tygodniowy nalicza się od momentu spełnienia warunków.

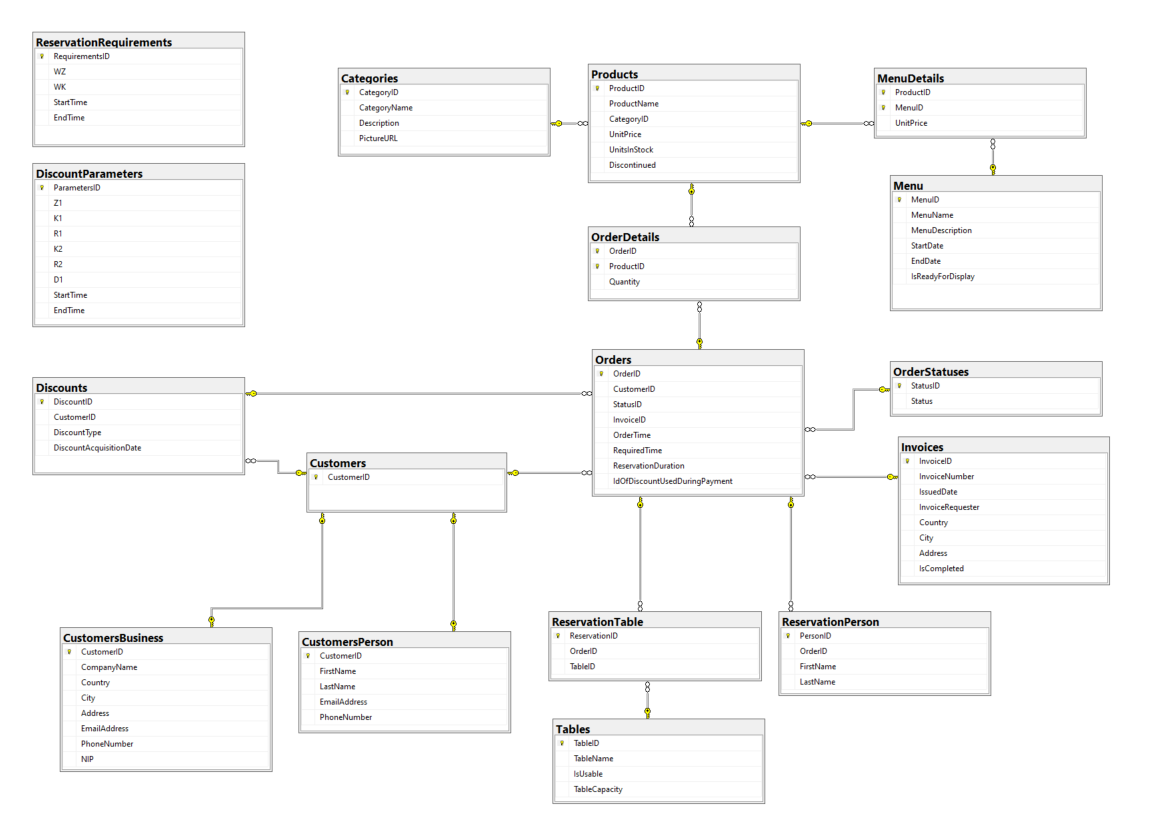
**Zarządzanie systemem:**

**Szef restauracji -** dodanie nowego produktu

1. **Schemat bazy danych:**



**Rysunek 1.** Schemat bazy danych wygenerowany przy pomocy programu Vertabelo



**Rysunek 2.** Schemat bazy danych wygenerowany przy pomocy Database Diagram w Microsoft SQL Server Management Studio

**Komentarze do schematu:**

**OrderStatuses** - taki enum (placed, accepted, being prepared, delivered)

**Orders** - jeżeli zamówienia na miejscu to jako adres podajemy adres restauracji

**Reservations** - nie ma daty na kiedy rezerwacja, bo jest ona tożsama z RequiredDate w Orders

**ReservationDetails** - 1. wiersz odpowiada jednej osobie (jeżeli zamówienie nie jest imienne - w FirstName i LastName nulle)

**Products** - jeżeli produkt zmienił cenę, to tworzony jest nowy produkt

**DiscountParameters** - tabela nie połączona relacja z innymi tabelami, służy tylko do przetrzymywania parametrów potrzebnych przy niektórych zapytaniach

**Kod generujący bazę danych o schemacie przedstawionym powyżej:**

| **-- tables -- Table: Categories CREATE TABLE Categories (  CategoryID INT NOT NULL IDENTITY (1,1),  CategoryName VARCHAR(100) NOT NULL DEFAULT 'Category',  Description TEXT NULL,  PictureURL VARCHAR(300) NOT NULL,  CONSTRAINT Categories\_pk PRIMARY KEY (CategoryID) );  -- Table: Customers CREATE TABLE Customers (  CustomerID INT NOT NULL IDENTITY (1,1),  CONSTRAINT Customers\_pk PRIMARY KEY (CustomerID) );  -- Table: CustomersBusiness CREATE TABLE CustomersBusiness (  CustomerID INT NOT NULL,  CompanyName VARCHAR(100) NOT NULL,  Country VARCHAR(100) NOT NULL,  City VARCHAR(100) NOT NULL,  Address VARCHAR(100) NOT NULL,  EmailAddress VARCHAR(80) NOT NULL,  PhoneNumber VARCHAR(20) NULL,  NIP VARCHAR(10) NULL,  CONSTRAINT CustomersBusiness\_ak\_1 UNIQUE (EmailAddress),  CONSTRAINT CustomersBusiness\_ak\_2 UNIQUE (PhoneNumber),  CONSTRAINT CustomersBusiness\_pk PRIMARY KEY (CustomerID) );  -- Table: CustomersPerson CREATE TABLE CustomersPerson (  CustomerID INT NOT NULL,  FirstName VARCHAR(30) NOT NULL,  LastName VARCHAR(30) NOT NULL,  EmailAddress VARCHAR(80) NOT NULL,  PhoneNumber VARCHAR(20) NULL,  CONSTRAINT CustomersPerson\_ak\_1 UNIQUE (EmailAddress),  CONSTRAINT CustomersPerson\_ak\_2 UNIQUE (PhoneNumber),  CONSTRAINT CustomersPerson\_pk PRIMARY KEY (CustomerID) );  -- Table: DiscountParameters CREATE TABLE DiscountParameters (  ParametersID INT NOT NULL IDENTITY (1,1),  Z1 INT NOT NULL,  K1 DECIMAL(10, 2) NOT NULL,  R1 DECIMAL(10, 2) NOT NULL,  K2 DECIMAL(10, 2) NOT NULL,  R2 DECIMAL(10, 2) NOT NULL,  D1 INT NOT NULL,  StartTime DATETIME NOT NULL,  EndTime DATETIME NOT NULL,  CONSTRAINT ParametersCheck CHECK (Z1 > 0 AND K1 > 0 AND R1 > 0 AND  K2 > 0 AND R2 > 0 AND D1 > 0),  CONSTRAINT TimeCheck CHECK (EndTime > StartTime AND StartTime > '2010-01-01'),  CONSTRAINT DiscountParameters\_pk PRIMARY KEY (ParametersID) );  -- Table: Discounts CREATE TABLE Discounts (  DiscountID INT NOT NULL IDENTITY (1,1),  CustomerID INT NOT NULL,  DiscountType TINYINT NOT NULL,  DiscountAcquisitionDate DATE NOT NULL,  CONSTRAINT DiscountTypeCheck CHECK (DiscountType IN (1, 2)),  CONSTRAINT CheckDate CHECK (DiscountAcquisitionDate > '2010-01-01'),  CONSTRAINT Discounts\_pk PRIMARY KEY (DiscountID) );  -- Table: Invoices CREATE TABLE Invoices (  InvoiceID INT NOT NULL IDENTITY (1,1),  InvoiceNumber INT NOT NULL,  IssuedDate DATE NOT NULL,  InvoiceRequester VARCHAR(100) NOT NULL,  Country VARCHAR(100) NOT NULL,  City VARCHAR(100) NOT NULL,  Address VARCHAR(100) NOT NULL,  IsCompleted BIT NOT NULL DEFAULT 0,  CONSTRAINT Invoices\_ak\_1 UNIQUE (InvoiceNumber),  CONSTRAINT Invoices\_pk PRIMARY KEY (InvoiceID) );  -- Table: Menu CREATE TABLE Menu (  MenuID INT NOT NULL IDENTITY (1,1),  MenuName VARCHAR(100) NOT NULL DEFAULT 'Menu',  MenuDescription TEXT NULL,  StartDate DATE NOT NULL,  EndDate DATE NOT NULL,  IsReadyForDisplay BIT NOT NULL,  CONSTRAINT DatesCheck CHECK (EndDate > StartDate AND StartDate > '2010-01-01' ),  CONSTRAINT Menu\_pk PRIMARY KEY (MenuID) );  -- Table: MenuDetails CREATE TABLE MenuDetails (  ProductID INT NOT NULL,  MenuID INT NOT NULL,  UnitPrice DECIMAL(10, 2) NOT NULL,  CONSTRAINT UnitPricePositive CHECK (UnitPrice > 0),  CONSTRAINT MenuDetails\_pk PRIMARY KEY (ProductID, MenuID) );  -- Table: OrderDetails CREATE TABLE OrderDetails (  OrderID INT NOT NULL,  ProductID INT NOT NULL,  Quantity INT NOT NULL,  CONSTRAINT QuantityPositive CHECK (Quantity > 0),  CONSTRAINT OrderDetails\_pk PRIMARY KEY (OrderID, ProductID) );  -- Table: OrderStatuses CREATE TABLE OrderStatuses (  StatusID INT NOT NULL IDENTITY (1,1),  Status VARCHAR(50) NOT NULL DEFAULT 'Status',  CONSTRAINT OrderStatuses\_pk PRIMARY KEY (StatusID) );  -- Table: Orders CREATE TABLE Orders (  OrderID INT NOT NULL IDENTITY (1,1),  CustomerID INT NOT NULL,  StatusID INT NOT NULL,  InvoiceID INT NULL,  OrderTime DATETIME NOT NULL,  RequiredTime DATETIME NOT NULL,  ReservationDuration INT NOT NULL DEFAULT 2,  CONSTRAINT TimesCheck CHECK (RequiredTime >= OrderTime AND  OrderTime > '2010-01-01'),  CONSTRAINT Orders\_pk PRIMARY KEY (OrderID) );  -- Table: Products CREATE TABLE Products (  ProductID INT NOT NULL IDENTITY (1,1),  ProductName VARCHAR(100) NOT NULL DEFAULT 'Product',  CategoryID INT NOT NULL,  UnitPrice DECIMAL(10, 2) NOT NULL,  UnitsInStock INT NOT NULL DEFAULT 0,  Discontinued BIT NOT NULL DEFAULT 0,  CONSTRAINT UnitPricePositiveProducts CHECK (UnitPrice > 0),  CONSTRAINT UnitsInStockNotNegative CHECK (UnitsInStock >= 0),  CONSTRAINT Products\_pk PRIMARY KEY (ProductID) );  -- Table: ReservationPerson CREATE TABLE ReservationPerson (  PersonID INT NOT NULL IDENTITY (1,1),  OrderID INT NOT NULL,  FirstName VARCHAR(30) NOT NULL,  LastName VARCHAR(30) NOT NULL,  CONSTRAINT ReservationPerson\_pk PRIMARY KEY (PersonID) );  -- Table: ReservationTable CREATE TABLE ReservationTable (  ReservationID INT NOT NULL IDENTITY (1,1),  OrderID INT NOT NULL,  TableID INT NOT NULL,  CONSTRAINT ReservationTable\_pk PRIMARY KEY (ReservationID) );  -- Table: Tables CREATE TABLE Tables (  TableID INT NOT NULL IDENTITY (1,1),  TableName VARCHAR(100) NULL,  IsUsable BIT NOT NULL DEFAULT 1,  CONSTRAINT Tables\_pk PRIMARY KEY (TableID) );  -- foreign keys -- Reference: Categories\_Products (table: Products) ALTER TABLE Products  ADD CONSTRAINT Categories\_Products  FOREIGN KEY (CategoryID)  REFERENCES Categories (CategoryID);  -- Reference: CustomersPerson\_Customers (table: CustomersPerson) ALTER TABLE CustomersPerson  ADD CONSTRAINT CustomersPerson\_Customers  FOREIGN KEY (CustomerID)  REFERENCES Customers (CustomerID);  -- Reference: CustomerBusiness\_Customers (table: CustomersBusiness) ALTER TABLE CustomersBusiness  ADD CONSTRAINT CustomerBusiness\_Customers  FOREIGN KEY (CustomerID)  REFERENCES Customers (CustomerID);   -- Reference: Customers\_Discounts (table: Discounts) ALTER TABLE Discounts  ADD CONSTRAINT Customers\_Discounts  FOREIGN KEY (CustomerID)  REFERENCES Customers (CustomerID);  -- Reference: Invoices\_Orders (table: Orders) ALTER TABLE Orders  ADD CONSTRAINT Invoices\_Orders  FOREIGN KEY (InvoiceID)  REFERENCES Invoices (InvoiceID);  -- Reference: MenuDetails\_Products (table: MenuDetails) ALTER TABLE MenuDetails  ADD CONSTRAINT MenuDetails\_Products  FOREIGN KEY (ProductID)  REFERENCES Products (ProductID);  -- Reference: Menu\_MenuDetails (table: MenuDetails) ALTER TABLE MenuDetails  ADD CONSTRAINT Menu\_MenuDetails  FOREIGN KEY (MenuID)  REFERENCES Menu (MenuID);  -- Reference: OrderDetails\_Order (table: OrderDetails) ALTER TABLE OrderDetails  ADD CONSTRAINT OrderDetails\_Order  FOREIGN KEY (OrderID)  REFERENCES Orders (OrderID);  -- Reference: OrderDetails\_Products (table: OrderDetails) ALTER TABLE OrderDetails  ADD CONSTRAINT OrderDetails\_Products  FOREIGN KEY (ProductID)  REFERENCES Products (ProductID);  -- Reference: OrderStatuses\_Orders (table: Orders) ALTER TABLE Orders  ADD CONSTRAINT OrderStatuses\_Orders  FOREIGN KEY (StatusID)  REFERENCES OrderStatuses (StatusID);  -- Reference: Order\_Customers (table: Orders) ALTER TABLE Orders  ADD CONSTRAINT Order\_Customers  FOREIGN KEY (CustomerID)  REFERENCES Customers (CustomerID);  -- Reference: PersonOnReservation\_Orders (table: ReservationPerson) ALTER TABLE ReservationPerson  ADD CONSTRAINT PersonOnReservation\_Orders  FOREIGN KEY (OrderID)  REFERENCES Orders (OrderID);  -- Reference: ReservationDetails\_Orders (table: ReservationTable) ALTER TABLE ReservationTable  ADD CONSTRAINT ReservationDetails\_Orders  FOREIGN KEY (OrderID)  REFERENCES Orders (OrderID);  -- Reference: Tables\_ReservationDetails (table: ReservationTable) ALTER TABLE ReservationTable  ADD CONSTRAINT Tables\_ReservationDetails  FOREIGN KEY (TableID)  REFERENCES Tables (TableID);** |
| --- |

| -- Table: ReservationRequirements CREATE TABLE ReservationRequirements (  RequirementsID INT NOT NULL IDENTITY (1,1),  WZ INT NOT NULL,  WK INT NOT NULL,  StartTime DATETIME NOT NULL,  EndTime DATETIME NOT NULL,  CONSTRAINT ReservationRequirementsParametersCheck  CHECK (WZ > 0 AND WK > 0),  CONSTRAINT ReservationRequirementsTimeCheck  CHECK (EndTime > StartTime AND StartTime > '2010-01-01'),  CONSTRAINT ReservationRequirements\_pk PRIMARY KEY (RequirementsID) ); |
| --- |

| - Adding TableCapacity column to table Tables  ALTER TABLE Tables ADD TableCapacity INT NOT NULL  ALTER TABLE Tables ADD CONSTRAINT TableCapacityCheck CHECK (TableCapacity > 0) |
| --- |

| - Adding column IdOfDiscountUsedDuringPayment to Orders table  ALTER TABLE Orders ADD IdOfDiscountUsedDuringPayment INT NULL  ALTER TABLE Orders  ADD CONSTRAINT Discounts\_Orders  FOREIGN KEY (IdOfDiscountUsedDuringPayment)  REFERENCES Discounts (DiscountID); |
| --- |

Baza została wypełniona mała ilością danych testowych potrzebnych do przetestowania funkcjonalności.

**Warunki integralnościowe:**Warunki integralnościowe zostały zawarte w kodzie SQL w punkcie 3. Zawarto tam między innymi wartości domyślne, przedziały do których powinny należeć ceny, oczekiwane typy danych, sprawdzanie czy wartości pól określających daty mają sens.

1. **Procedury i widoki**

**4.1 Widoki**

1. Widok łączący zamówienie i status zamówienia

| **CREATE VIEW OrdersWithStatus AS SELECT OrderID,  CustomerID,  OS.Status,  InvoiceID,  OrderTime,  RequiredTime,  ReservationDuration FROM Orders O  INNER JOIN OrderStatuses OS ON O.StatusID = OS.StatusID** |
| --- |

2. Widok łączący zamówienie z nazwą klienta (niezależnie czy jest to klient indywidualny czy przedsiębiorca)

| **CREATE VIEW OrderWithClient AS SELECT OrderID,  ISNULL(CB.CompanyName, CP.FirstName + ' ' + CP.LastName) CustomerName,  O.CustomerID,  StatusID,  InvoiceID,  OrderTime,  RequiredTime,  ReservationDuration,  IdOfDiscountUsedDuringPayment FROM Orders O  INNER JOIN Customers C ON O.CustomerID = C.CustomerID  LEFT JOIN CustomersBusiness CB ON C.CustomerID = CB.CustomerID  LEFT JOIN CustomersPerson CP ON C.CustomerID = CP.CustomerID** |
| --- |

3. Widok łączący łączący produkty z nazwami kategorii

| **CREATE VIEW ProductsWithCategory AS SELECT ProductID,  ProductName,  C.CategoryName,  UnitPrice,  UnitsInStock,  Discontinued FROM Products  JOIN Categories C ON Products.CategoryID = C.CategoryID** |
| --- |

4. Widok z obliczoną wartością dla każdego szczegółu zamówienia’

| CREATE VIEW OrderDetailsWithValue AS SELECT OD.OrderID,  OD.ProductID,  (OD.Quantity \* MD.UnitPrice) AS OrderDetailValue,  OD.Quantity,  MD.UnitPrice FROM Orders O  JOIN OrderDetails OD ON O.OrderID = OD.OrderID  JOIN Products P ON OD.ProductID = P.ProductID  JOIN MenuDetails MD ON P.ProductID = MD.ProductID  JOIN Customers C ON C.CustomerID = O.CustomerID |
| --- |

5. Widok z menu które nie są jeszcze zatwierdzone i trwa nad nimi praca

| **CREATE VIEW MenusInProgressWithProducts AS SELECT M.MenuID,  M.MenuName,  M.MenuDescription,  StartDate,  EndDate,  P.ProductID,  ProductName,  MD.UnitPrice FROM Menu M  INNER JOIN MenuDetails MD ON M.MenuID = MD.MenuID  INNER JOIN Products P ON P.ProductID = MD.ProductID WHERE IsReadyForDisplay = 0** |
| --- |

6. Widok wszystkich menu z ich produktami

| **CREATE VIEW MenusWithProducts AS SELECT M.MenuID,  M.MenuName,  M.MenuDescription,  StartDate,  EndDate,  P.ProductID,  ProductName,  MD.UnitPrice FROM Menu M  INNER JOIN MenuDetails MD ON M.MenuID = MD.MenuID  INNER JOIN Products P ON P.ProductID = MD.ProductID** |
| --- |

7. Widok wszystkich zamówień które mają rezerwacje na stoliki z informacjami o tychże rezerwacjach

| CREATE VIEW OrdersWithTables AS SELECT O.OrderID,  CustomerID,  OS.Status,  OrderTime,  RequiredTime,  ReservationDuration,  ReservationID,  RT.TableID,  T.TableName,  T.IsUsable FROM Orders O  INNER JOIN OrderStatuses OS ON O.StatusID = OS.StatusID  INNER JOIN ReservationTable RT ON O.OrderID = RT.OrderID  INNER JOIN Tables T ON RT.TableID = T.TableID |
| --- |

**4.2 Procedury**

1. Wartość danego zamówienia

| CREATE PROCEDURE usp\_valueOfOrder @VarOrderID INT AS BEGIN  IF EXISTS(SELECT \* FROM Orders O WHERE O.OrderID = @VarOrderID)  BEGIN  SELECT SUM(ODValue.OrderDetailValue)  FROM Customers C  JOIN Orders O ON C.CustomerID = O.CustomerID  JOIN OrderDetailsWithValue ODValue  ON O.OrderID = ODValue.OrderID  WHERE O.OrderID = @VarOrderID  END  ELSE  BEGIN  ;THROW 60000, 'Specified @order\_id does not exist!!!', 1;  END END |
| --- |

2. Menu ważne danego dnia

| CREATE PROCEDURE usp\_menuValidInDate @VarDate DATE AS BEGIN  SELECT ProductName, C2.CategoryName, M.UnitPrice  FROM Products  JOIN MenuDetails M ON Products.ProductID = M.ProductID  JOIN Menu M2 ON M2.MenuID = M.MenuID  JOIN Categories C2 ON C2.CategoryID = Products.CategoryID  WHERE @VarDate >= M2.StartDate  AND @VarDate <= M2.EndDate  AND M2.IsReadyForDisplay = 1 END |
| --- |

3. Rezerwacje w danym przedziale czasowym

| **CREATE PROCEDURE usp\_reservationsInTimePeriod @VarStartDate DATE, @VarEndDate DATE AS BEGIN  SELECT ReservationID,  O.OrderID,  ISNULL(CB.CompanyName, CP.FirstName + ' ' +  CP.LastName) CustomerName,  T.TableID,  T.TableName,  OrderTime  FROM ReservationTable RT  INNER JOIN Orders O ON RT.OrderID = O.OrderID  INNER JOIN Tables T ON RT.TableID = T.TableID  INNER JOIN Customers C ON O.CustomerID = C.CustomerID  LEFT JOIN CustomersBusiness CB  ON C.CustomerID = CB.CustomerID  LEFT JOIN CustomersPerson CP  ON C.CustomerID = CP.CustomerID   WHERE  OrderTime >= @VarStartDate and OrderTime <= @VarEndDate  ORDER BY ReservationID; END  -- przykładowe wywołanie EXEC usp\_reservationsInTimePeriod "20220101", "20220104"** |
| --- |

4. Faktury danego klienta

| CREATE PROCEDURE usp\_invoicesOfRequester @VarCustomerID INT AS BEGIN  IF ((@VarCustomerID) NOT IN (SELECT CustomerID  FROM Customers))  BEGIN  ;THROW 60000, 'There is no such order!!!', 1;  END   SELECT \*  FROM Invoices   WHERE InvoiceID IN  (SELECT Orders.InvoiceID FROM Orders WHERE CustomerID = @VarCustomerID) END |
| --- |

5. Informacje na temat zniżek typu 1. danego klienta

| CREATE PROCEDURE usp\_discount1Progress @CustomerID INT AS BEGIN  IF NOT EXISTS(SELECT \* FROM Customers WHERE CustomerID = @CustomerID)  BEGIN  ;THROW 60000, 'Specified Customer does not exist!!!', 1;  END  ELSE  IF IIF(@CustomerID IN (SELECT CustomerID FROM Discounts WHERE DiscountType = 1),  1, NULL) IS NULL  BEGIN  SELECT COUNT(\*) AS HowManyOrders, 0 HasDiscountR1  FROM (SELECT SUM(MD.UnitPrice) price  FROM OrderDetails OD  INNER JOIN Orders O ON O.OrderID = OD.OrderID  INNER JOIN Products P ON P.ProductID = OD.ProductID  INNER JOIN MenuDetails MD ON P.ProductID = MD.ProductID  WHERE CustomerID = @CustomerID  GROUP BY O.OrderID  HAVING SUM(MD.UnitPrice) > (SELECT K1  FROM DiscountParameters  WHERE StartTime =  (SELECT MAX(StartTime)  FROM DiscountParameters))) AS orders  END  ELSE  BEGIN  SELECT COUNT(\*) AS HowManyOrders,1 HasDiscountR1  FROM (SELECT SUM(MD.UnitPrice) price, O.OrderTime  FROM OrderDetails OD  INNER JOIN Orders O ON O.OrderID = OD.OrderID  INNER JOIN Products P ON P.ProductID = OD.ProductID  INNER JOIN MenuDetails MD ON P.ProductID = MD.ProductID  WHERE CustomerID = @CustomerID  GROUP BY O.OrderID, O.OrderTime  HAVING SUM(MD.UnitPrice) \*  ISNULL(1 - (SELECT R1  FROM DiscountParameters  WHERE StartTime < O.OrderTime  AND EndTime > O.OrderTime),1)  > (SELECT K1 FROM DiscountParameters  WHERE StartTime =  (SELECT MAX(StartTime)  FROM DiscountParameters))) AS orders  END  END |
| --- |

6. Informacje na temat zniżek typu 2. danego klienta

| CREATE PROCEDURE usp\_discount2Progress @CustomerID INT AS BEGIN  IF NOT EXISTS(SELECT \*  FROM Customers  WHERE CustomerID = @CustomerID)  BEGIN  ;THROW 60000, 'Specified Customer does not exist!!!', 1;  END  ELSE  BEGIN  IF IIF(@CustomerID IN (SELECT CustomerID FROM Discounts WHERE DiscountType = 2),  1, NULL) IS NULL  BEGIN  SELECT SUM(MD.UnitPrice) MoneySpent, 0 AS HasDiscountR2  FROM OrderDetails OD  INNER JOIN Orders O ON O.OrderID = OD.OrderID  INNER JOIN Products P ON P.ProductID = OD.ProductID  INNER JOIN MenuDetails MD ON P.ProductID = MD.ProductID  WHERE CustomerID = @CustomerID  GROUP BY CustomerID  END  ELSE  BEGIN  SELECT SUM(MD.UnitPrice) MoneySpent, 1 AS HasDiscountR2  FROM OrderDetails OD  INNER JOIN Orders O ON O.OrderID = OD.OrderID  INNER JOIN Products P ON P.ProductID = OD.ProductID  INNER JOIN MenuDetails MD ON P.ProductID = MD.ProductID  WHERE CustomerID = @CustomerID  AND O.OrderTime >  (SELECT MAX(DiscountAcquisitionDate)  FROM Discounts WHERE CustomerID = @CustomerID AND DiscountType = 2)  GROUP BY CustomerID  END  END END |
| --- |

7. Informacje na temat zniżek obu typów dla danego klienta

| CREATE PROCEDURE usp\_discountProgress @CustomerID INT AS BEGIN  IF NOT EXISTS(SELECT \* FROM Customers WHERE CustomerID = @CustomerID)  BEGIN  ;THROW 60000, 'Specified Customer does not exist!!!', 1;  END  ELSE  BEGIN  DECLARE @t TABLE  (  [OrdersDone/MoneySpent] INT,  HasDiscount INT  )  INSERT INTO @t EXEC usp\_discount1Progress @CustomerID  INSERT INTO @t EXEC usp\_discount2Progress @CustomerID  SELECT \* FROM @t  END END |
| --- |

8. Stoliki danej rezerwacji

| CREATE PROCEDURE usp\_tablesOfOrderReservation @VarOrderID INT AS BEGIN  IF NOT EXISTS(SELECT \* FROM Orders WHERE OrderID = @VarOrderID)  BEGIN  ;THROW 60000, 'Specified OrderID does not exist!!!', 1;  END  ELSE  BEGIN  SELECT O.OrderID, T.TableID, T.TableName  FROM Orders O  INNER JOIN ReservationTable RT ON O.OrderID = RT.OrderID  INNER JOIN Tables T ON RT.TableID = T.TableID  WHERE O.OrderID = @VarOrderID  ORDER BY TableID;  END END |
| --- |

9. Ustawienie stanu gotowości do wystawienia danego menu

| CREATE PROCEDURE usp\_setMenuReadyForDisplayState @VarMenuID INT, @VarMenuState BIT AS BEGIN  IF NOT EXISTS(SELECT \* FROM Menu WHERE MenuID = @VarMenuID)  BEGIN  ;THROW 60000, 'Specified MenuID does not exist!!!', 1;  END   ELSE  BEGIN  UPDATE Menu  SET IsReadyForDisplay = @VarMenuState  WHERE MenuID = @VarMenuID;  END END  -- przykładowe wywołanie EXEC usp\_setMenuReadyForDisplayState 1, 0 -- sprawdzenie stanu menu SELECT \* FROM Menu WHERE MenuID = 1 |
| --- |

10. Wystawianie faktury (zmiana wartości isCompleted na 1 w tabeli Invoices)

| CREATE PROCEDURE usp\_completeInvoice @VarInvoiceID INT AS BEGIN  IF IIF(@VarInvoiceID IN (SELECT InvoiceID  FROM Invoices),  1,  NULL) IS NULL  BEGIN  ;THROW 60000, 'Specified InvoiceID does not exist!!!', 1;  END   ELSE IF IIF(@VarInvoiceID IN (SELECT InvoiceID  FROM Invoices  WHERE IsCompleted = 0),  1,  NULL) IS NULL  BEGIN  ;THROW 60000, 'Specified invoice was already completed!!!', 1;  END  ELSE  BEGIN  UPDATE Invoices  SET IsCompleted = 1  IssuedDate = GETDATE()  WHERE InvoiceID = @VarInvoiceID;  END END  -- przykładowe wywołanie EXEC usp\_completeInvoice 1 |
| --- |

11. Stworzenie faktury z jednym zamówieniem

| CREATE PROCEDURE usp\_createInvoiceAndAddFirstOrder @VarOrderID INT,  @VarInvoiceNumber INT,  @VarIssuedDate DATE AS BEGIN  BEGIN TRANSACTION  IF ((@VarOrderID) NOT IN (SELECT OrderID  FROM Orders))  BEGIN  ;THROW 60000, 'Specified OrderID does not exist!!!', 1;  END   ELSE  BEGIN  IF (1 IN (SELECT IsCompleted  FROM Invoices  WHERE InvoiceID = (SELECT InvoiceID  FROM Orders  WHERE OrderID = @VarOrderID)))  BEGIN  ;THROW 60000, 'That order is already included in completed invoice', 1;  END   INSERT INTO Invoices (InvoiceNumber, IssuedDate,  InvoiceRequester,  Country,  City, Address, IsCompleted)  SELECT @VarInvoiceNumber,  @VarIssuedDate,  CompanyName,  Country,  City,  Address,  0  FROM CustomersBusiness  WHERE CustomerID =  (SELECT CustomerID  FROM Orders  WHERE OrderID = @VarOrderID)   DECLARE @LastID INT;  SET @LastID = @@IDENTITY   UPDATE Orders  SET InvoiceID = @LastID  WHERE OrderID = @VarOrderID  END  COMMIT END |
| --- |

12. Dodanie pojedynczego zamówienia do faktury

| CREATE PROCEDURE usp\_addSingleOrderToInvoice @VarInvoiceID INT, @VarOrderID INT AS BEGIN  BEGIN TRANSACTION  IF (1 IN (SELECT IsCompleted  FROM Invoices  WHERE InvoiceID = @VarInvoiceID))  BEGIN  ;THROW 60000, 'That invoice is already completed!!!', 1;  END   IF ((@VarInvoiceID) NOT IN (SELECT InvoiceID  FROM Invoices))  BEGIN  ;THROW 60000, 'Specified InvoiceID does not exist!!!', 1;  END   IF ((@VarOrderID) NOT IN (SELECT OrderID  FROM Orders))  BEGIN  ;THROW 60000, 'Specified OrderID does not exist!!!', 1;  END   IF (1 IN (SELECT IsCompleted  FROM Invoices  WHERE InvoiceID = (SELECT InvoiceID  FROM Orders  WHERE OrderID = @VarOrderID)))  BEGIN  ;THROW 60000, 'That order is already included in completed invoice', 1;  END   DECLARE @InvoiceOwnerID INT;  SET @InvoiceOwnerID =  (SELECT CustomerID FROM Orders WHERE OrderID = @VarOrderID)   IF ((SELECT COUNT(DISTINCT CustomerID)  FROM Orders  WHERE InvoiceID = @VarInvoiceID  AND CustomerID != @InvoiceOwnerID) >= 1)  BEGIN  ;THROW 60000, 'That invoice already contains orders from other customer!!!', 1;  END   UPDATE Orders  SET InvoiceID = @VarInvoiceID  WHERE OrderID = @VarOrderID;  COMMIT END |
| --- |

13. Utworzenie faktury miesięcznej dla danego klienta

| CREATE PROCEDURE usp\_createMonthlyInvoice @VarCustomerID INT,  @VarYear INT,  @VarMonth INT,  @VarInvoiceNumber INT,  @VarIssuedDate DATE AS BEGIN   IF ((@VarCustomerID) NOT IN (SELECT CustomerID  FROM CustomersBusiness))  BEGIN  ;THROW 60000, 'There is no such Business Client!!!', 1;  END    INSERT INTO Invoices (InvoiceNumber, IssuedDate,  InvoiceRequester,  Country,  City, Address, IsCompleted)   SELECT @VarInvoiceNumber,  @VarIssuedDate,  CompanyName,  Country,  City,  Address,  0  FROM CustomersBusiness  WHERE CustomerID = @VarCustomerID   DECLARE @LastID INT;  SET @LastID = @@IDENTITY;   UPDATE Orders  SET InvoiceID = @LastID  WHERE CustomerID = @VarCustomerID  AND YEAR(OrderTime) = @VarYear  AND MONTH(OrderTime) = @VarMonth  AND InvoiceID IS NULL  END |
| --- |

14. Uzyskanie danych na temat danej faktury

| CREATE PROCEDURE usp\_getInvoiceInfo @VarInvoiceID INT AS BEGIN  IF IIF(@VarInvoiceID IN (SELECT InvoiceID  FROM Invoices),  1,  NULL) IS NULL  BEGIN  ;THROW 60000, 'Specified InvoiceID does not exist!!!', 1;  END   ELSE  BEGIN  SELECT I.InvoiceID, I.InvoiceNumber,  I.IsCompleted, I.IssuedDate, I.InvoiceRequester,  I.Country, I.City, I.Address,  O.OrderID, P.ProductName, OD.Quantity,  P.UnitPrice, P.UnitPrice \* OD.Quantity as 'Sum'  FROM Invoices I   INNER JOIN Orders O ON O.InvoiceID = I.InvoiceID  INNER JOIN OrderDetails OD ON O.OrderID = OD.OrderID  INNER JOIN Products P ON OD.ProductID = P.ProductID  END END |
| --- |

15. Stworzenie nowego zamówienia



| CREATE PROCEDURE usp\_createOrder @VarCustomerID INT, @VarRequiredTime DATETIME, @VarReservatonDuration INT = NULL AS BEGIN  IF IIF(@VarCustomerID IN (SELECT CustomerID  FROM Customers),  1,  NULL) IS NULL  BEGIN  ;THROW 60000, 'Specified CustomerID does not exist!!!', 1;  END  ELSE IF @VarRequiredTime < GETDATE()  THROW 60000, 'Required time cannot be set before current time!!!', 1;  ELSE IF @VarReservatonDuration IS NOT NULL AND @VarReservatonDuration <= 0   THROW 60000, 'Reservation duration cannot be negative!!!', 1;  ELSE  BEGIN  INSERT INTO Orders(CustomerID, StatusID, InvoiceID, OrderTime, RequiredTime, ReservationDuration)  VALUES(@VarCustomerID, 6, NULL, GETDATE(), @VarRequiredTime, @VarReservatonDuration)  END END   - przykładowe wywołanie: EXEC usp\_createOrder 1, '2022-01-18', '01:02:03' |
| --- |

16 Anulowanie zamówienia

| CREATE PROCEDURE usp\_orderCancelling @OrderID INT AS BEGIN  UPDATE Orders  SET StatusID = 9  WHERE OrderID = @OrderID END |
| --- |

17. Ustawienie statusu zamówienia na zrealizowane

| CREATE PROCEDURE usp\_orderCompleted @OrderID INT AS BEGIN  IF ((SELECT StatusID FROM Orders WHERE OrderID = @OrderID) != 2)  BEGIN  ;THROW 60000, 'Order have to be payed first!!!', 1;  END  ELSE  BEGIN  UPDATE Orders  SET StatusID = 4  WHERE OrderID = @OrderID  END END |
| --- |

18. Dodawanie produktów do zamówienia

| CREATE PROCEDURE usp\_addProductToOrder @VarProductID INT, @VarOrderID INT,  @VarQuantity INT AS BEGIN  IF @VarOrderID NOT IN (SELECT OrderID FROM Orders)  THROW 60000, 'Specified order does not exist!!!', 1;  ELSE  IF (SELECT Status  FROM OrdersWithStatus  WHERE OrderID = @VarOrderID) != 'W trakcie składania'  THROW 60000, 'Specified order has been closed!!!', 1;  ELSE  BEGIN  DECLARE @VarRequiredTime AS DATETIME,  @VarValidMenuID AS INT;  SET @VarRequiredTime = (SELECT RequiredTime  FROM Orders  WHERE OrderID = @VarOrderID);  SET @VarValidMenuID = (SELECT MenuID  FROM Menu  WHERE @VarRequiredTime >= StartDate AND @VarRequiredTime <= EndDate  AND IsReadyForDisplay = 1);   IF @VarValidMenuID IS NULL  THROW 60000, 'There is no valid menu in order required time!!!', 1;  IF @VarProductID NOT IN (SELECT ProductID  FROM MenusWithProducts  WHERE MenuID = @VarValidMenuID)  THROW 60000, 'Specified product does not appear in valid menu!!!', 1;  IF (SELECT CategoryName  FROM ProductsWithCategory  WHERE ProductID = @VarProductID) = 'Owoce morza'  BEGIN  IF (DATENAME(WEEKDAY, @VarRequiredTime) !=  'Thursday' AND  DATENAME(WEEKDAY, @VarRequiredTime) !=  'Friday' AND  DATENAME(WEEKDAY, @VarRequiredTime) !=  'Saturday')  THROW 60000, 'RequiredTime for Seafood can be only Thursday/Friday/Saturday!!!', 1;  ELSE  DECLARE @VarTuesdayBeforeRequiredTime AS DATETIME;  SET @VarTuesdayBeforeRequiredTime =  (SELECT DATEADD(DD, -1 \*  (DATEPART(DW, @VarRequiredTime) - 2),  @VarRequiredTime));  IF GETDATE() >= @VarTuesdayBeforeRequiredTime  THROW 60000, 'Seafood cannot be order later than monday before RequiredTime!!!', 1;  END  INSERT INTO OrderDetails  VALUES (@VarOrderID, @VarProductID, @VarQuantity)  UPDATE Orders  SET OrderTime = GETDATE()  WHERE OrderID = @VarOrderID  END END |
| --- |

**Proces płacenia**

19. Sprawdzenie czy klient ma aktualnie obowiązujące zniżki

| CREATE PROCEDURE usp\_discountToUse @Date DATE, @CustomerID INT AS BEGIN  SELECT TOP 1 DiscountID  FROM Discounts  WHERE ((DiscountType = 1 AND DiscountAcquisitionDate <= @Date)  OR  ((DiscountType = 2 AND DiscountAcquisitionDate <= @Date AND  @Date <= DATEADD(DAY, (SELECT TOP 1 D1  FROM DiscountParameters  WHERE StartTime <= @Date  AND @Date <= EndTime  ORDER BY StartTime DESC), @Date)))  )  AND CustomerID = @CustomerID  ORDER BY IIF(DiscountType = 1, (SELECT TOP 1 R1  FROM DiscountParameters  WHERE StartTime <= @Date  AND @Date <= EndTime  ORDER BY StartTime DESC),  (SELECT TOP 1 R2  FROM DiscountParameters  WHERE StartTime <= @Date  AND @Date <= EndTime  ORDER BY StartTime DESC)) DESC END |
| --- |

20. Ile klient ma zapłacić, uwzględniając ew. Zniżkę

| CREATE PROCEDURE usp\_showPrice @OrderID INT AS BEGIN  DECLARE @CustomerID INT = (SELECT CustomerID  FROM Orders  WHERE OrderID = @OrderID)  DECLARE @Date DATE = GETDATE()  DECLARE @d INT EXEC usp\_discountToUse @Date, @CustomerID   SELECT SUM(OrderDetailValue) \* ISNULL(1 - @d, 1)  FROM OrderDetailsWithValue  WHERE OrderID = @OrderID END |
| --- |

21. Płacenie - zmiana Order Status i DiscountUsedDuringPayment

| CREATE PROCEDURE usp\_payment @OrderID INT, @CustomerID INT AS BEGIN  DECLARE @Date DATE = GETDATE()  DECLARE @p INT EXEC usp\_showPrice @OrderID, @CustomerID  DECLARE @d INT EXEC usp\_discountToUse @Date, @CustomerID   UPDATE Orders  SET StatusID = 2  WHERE OrderID = @OrderID   IF (@d IS NOT NULL)  BEGIN  UPDATE Orders  SET IdOfDiscountUsedDuringPayment = @d  WHERE OrderID = @OrderID  END END |
| --- |

22. Sprawdzenie czy po transakcji klientowi będzie przysługiwać jakaś nowa zniżka

| CREATE PROCEDURE usp\_addNewDiscountIfGranted @CustomerID INT, @Date DATE AS BEGIN  DECLARE @t1 TABLE  (  [OrdersDone/MoneySpent] INT,  HasDiscount INT,  type INT  )  DECLARE @t2 TABLE  (  [OrdersDone/MoneySpent] INT,  HasDiscount INT,  type INT  )  INSERT INTO @t1 EXEC usp\_discount1Progress @CustomerID  INSERT INTO @t2 EXEC usp\_discount2Progress @CustomerID   IF ((SELECT HasDiscount FROM @t1) = 0 AND  (SELECT [OrdersDone/MoneySpent] FROM @t1) > (SELECT TOP 1 Z1  FROM DiscountParameters  WHERE StartTime <= @Date  AND @Date <= EndTime  ORDER BY StartTime DESC))  BEGIN  INSERT INTO Discounts(CustomerID, DiscountType,  DiscountAcquisitionDate)  VALUES (@CustomerID, 1, @Date)  END   IF ((SELECT HasDiscount FROM @t2) = 0 AND  (SELECT [OrdersDone/MoneySpent] FROM @t2) > (SELECT TOP 1 K1  FROM DiscountParameters  WHERE StartTime <= @Date  AND @Date <= EndTime  ORDER BY StartTime DESC))  BEGIN  INSERT INTO Discounts(CustomerID, DiscountType,  DiscountAcquisitionDate)  VALUES (@CustomerID, 2, @Date)  END  END |
| --- |

**Proces tworzenia menu**

23. Dodawanie nowego produktu

| CREATE PROCEDURE usp\_addProduct @ProductName VARCHAR(100), @CategoryID INT,  @UnitPrice DECIMAL(10, 2), @UnitsInStock INT AS BEGIN  IF (@CategoryID > (SELECT MAX(CategoryID) FROM Categories))  BEGIN  ;THROW 60000, 'No such category! Consider adding one first', 1;  END  ELSE  BEGIN  INSERT INTO Products(ProductName, CategoryID, UnitPrice,  UnitsInStock, Discontinued)  VALUES (@ProductName, @CategoryID, @UnitPrice, @UnitsInStock, 0)  END  END |
| --- |

24. Dodawanie stworzonego produktu do menu

| CREATE PROCEDURE usp\_addProductToMenu @ProductID INT, @MenuID INT,  @UnitPrice DECIMAL(10, 2) AS BEGIN  IF ((SELECT ProductID  FROM MenuDetails  WHERE ProductID = @ProductID AND MenuID = @MenuID) IS NOT NULL)  BEGIN  ;THROW 60000, 'Product has already been added ', 1;  END  ELSE  BEGIN  INSERT INTO MenuDetails(ProductID, MenuID, UnitPrice)  VALUES (@ProductID, @MenuID, @UnitPrice)  END END |
| --- |

25. Zatwierdzenie menu gdy jest gotowe

| CREATE PROCEDURE usp\_confirmMenu @StartDate DATE, @EndDate DATE AS BEGIN  DECLARE @NumOfNewProducts INT = (  SELECT COUNT(CurrMenu.ProductID)  FROM (SELECT ProductID  FROM MenusInProgressWithProducts  WHERE StartDate <= GETDATE()  AND GETDATE() <= EndDate) CurrMenu  LEFT JOIN  (SELECT ProductID  FROM MenuDetails MD  INNER JOIN Menu M ON M.MenuID = MD.MenuID  WHERE StartDate <= DATEADD(DAY, -14, GETDATE())  AND DATEADD(DAY, -14, GETDATE()) <= EndDate  AND IsReadyForDisplay = 'true') PrevMenu  ON PrevMenu.ProductID = CurrMenu.ProductID  WHERE PrevMenu.ProductID IS NULL  )   IF ((SELECT TOP 1 EndDate  FROM Menu  WHERE IsReadyForDisplay = 'true'  ORDER BY EndDate DESC) > @StartDate)  BEGIN  ;THROW 60000, 'Menus are overlapping!!!', 1;  END  ELSE  IF (@NumOfNewProducts < (SELECT COUNT(ProductID)  FROM MenuDetails MD  INNER JOIN Menu M ON M.MenuID = MD.MenuID  WHERE StartDate <= DATEADD(DAY, -14, GETDATE())  AND DATEADD(DAY, -14, GETDATE()) <= EndDate  AND IsReadyForDisplay = 'true') / 2)  BEGIN  ;  THROW 60000, 'Not enough products products are different that two weeks ago!!!', 1;  END  ELSE  BEGIN  UPDATE Menu  SET IsReadyForDisplay = 'true'  WHERE Menu.StartDate = @StartDate  AND Menu.EndDate = @EndDate  END  END 2 osoby myślące wsp |
| --- |

26. Dodawanie klienta indywidualnego

| CREATE PROCEDURE usp\_addCustomerPerson @VarFirstName VARCHAR(50),  @VarLastName VARCHAR(50),  @VarEmailAddress VARCHAR(50),  @VarPhoneNumber VARCHAR(50) AS BEGIN   IF ((@VarEmailAddress) IN (SELECT EmailAddress  FROM CustomersPerson  UNION  SELECT EmailAddress  FROM CustomersBusiness))  BEGIN  ;THROW 60000, 'That email is already used!!!', 1;  END   INSERT INTO Customers DEFAULT VALUES;   DECLARE @LastID INT;  SET @LastID = @@IDENTITY;   INSERT INTO CustomersPerson (CustomerID, FirstName, LastName,  EmailAddress, PhoneNumber)   VALUES (@LastID,  @VarFirstName,  @VarLastName,  @VarEmailAddress,  @VarPhoneNumber)  END |
| --- |

27. Dodawanie klienta biznesowego

| CREATE PROCEDURE usp\_addCustomerBusiness @VarCompanyName VARCHAR(50),  @VarCountry VARCHAR(50),  @VarCity VARCHAR(50),  @VarAddress VARCHAR(50),  @VarEmailAddress VARCHAR(50),  @VarPhoneNumber VARCHAR(50),  @VarNIP VARCHAR(50) AS BEGIN   IF ((@VarEmailAddress) NOT IN (SELECT EmailAddress  FROM CustomersPerson  UNION  SELECT EmailAddress  FROM CustomersBusiness))  BEGIN  ;THROW 60000, 'That email is already used!!!', 1;  END   INSERT INTO Customers DEFAULT VALUES;   DECLARE @LastID INT;  SET @LastID = @@IDENTITY;   INSERT INTO CustomersBusiness (CustomerID, CompanyName, Country, City,  Address, EmailAddress, PhoneNumber, NIP)   VALUES (@LastID,  @VarCompanyName,  @VarCountry,  @VarCity,  @VarAddress,  @VarEmailAddress,  @VarPhoneNumber,  @VarNIP)  END |
| --- |

28. Stoliki danego zamówienia

| CREATE PROCEDURE usp\_TablesOfOrder @VarOrderID INT AS BEGIN  IF ((@VarOrderID) NOT IN (SELECT OrderID  FROM Orders))  BEGIN  ;THROW 60000, 'There is no such order!!!', 1;  END    SELECT TableName, IsUsable, TableCapacity FROM Tables  JOIN ReservationTable RT ON Tables.TableID = RT.TableID  JOIN Orders O ON O.OrderID = RT.OrderID  WHERE O.OrderID = @VarOrderID;  END |
| --- |

29. Raport zniżek

| CREATE PROCEDURE usp\_ReportOfDiscounts @VarStartDate DATE, @VarEndDate DATE AS BEGIN  IF (@VarStartDate > @VarEndDate)  BEGIN  ;THROW 60000, 'Starting Date has to be less than EndDate', 1;  END    SELECT \* FROM Discounts  WHERE DiscountAcquisitionDate BETWEEN @VarStartDate AND @VarEndDate END |
| --- |

30. Raport menu

| CREATE PROCEDURE usp\_ReportOfMenu @VarStartDate DATE, @VarEndDate DATE AS BEGIN  IF (@VarStartDate > @VarEndDate)  BEGIN  ;THROW 60000, 'Starting Date has to be less than EndDate', 1;  END    SELECT \*  FROM Menu  WHERE ((StartDate BETWEEN @VarStartDate AND @VarEndDate) OR  (EndDate BETWEEN @VarStartDate AND @VarEndDate))  AND IsReadyForDisplay = 1 END |
| --- |

31. Zamówienia danego klienta

| CREATE PROCEDURE usp\_ordersOfCustomer @VarCustomerID INT AS BEGIN  IF ((@VarCustomerID) NOT IN (SELECT CustomerID  FROM Customers))  BEGIN  ;THROW 60000, 'There is no such customer!!!', 1;  END   SELECT \* FROM Orders  WHERE CustomerID = @VarCustomerID END |
| --- |

32. Zniżki danego klienta

| CREATE PROCEDURE usp\_discountsOfCustomer @VarCustomerID INT AS BEGIN  IF ((@VarCustomerID) NOT IN (SELECT CustomerID  FROM Customers))  BEGIN  ;THROW 60000, 'There is no such customer!!!', 1;  END   SELECT \* FROM Discounts  WHERE CustomerID = @VarCustomerID END |
| --- |

33. Średnia wartość zamówienia danego klienta

| **CREATE PROCEDURE usp\_meanOrderValueOfCustomer @VarCustomerID INT AS BEGIN  IF ((@VarCustomerID) NOT IN (SELECT CustomerID  FROM Customers))  BEGIN  ;THROW 60000, 'There is no such customer!!!', 1;  END   SELECT AVG(sum)  FROM (  SELECT O.OrderID, SUM(ODValue.OrderDetailValue) AS sum  FROM Customers C  JOIN Orders O ON C.CustomerID = O.CustomerID  JOIN OrderDetailsWithValue ODValue  ON O.OrderID = ODValue.OrderID  WHERE C.CustomerID = @VarCustomerID  GROUP BY O.OrderID  ) OrdersValue END** |
| --- |

34. Dodanie stolika do zamówienia

| CREATE PROCEDURE usp\_addTableToReservation @OrderID INT, @NumOfPeople INT,  @DateTime DATETIME,  @ReservationDuration INT AS BEGIN  DECLARE @CustomerID INT = (SELECT CustomerID  FROM Orders  WHERE OrderID = @OrderID)  IF ((SELECT OrderID FROM Orders WHERE OrderID = @OrderID) IS NULL)  BEGIN  ;THROW 60000, 'There no such order!!!', 1;  END  ELSE  BEGIN  DECLARE @val DECIMAL(10, 2) EXEC usp\_showPrice @OrderID  IF (@val < (SELECT TOP 1 WZ  FROM ReservationRequirements  ORDER BY StartTime DESC))  BEGIN  ;THROW 60000, 'Value of order is too low!!!', 1;  END  ELSE  IF ((SELECT COUNT(OrderID)  FROM Orders  WHERE CustomerID = @CustomerID) <= (SELECT TOP 1 WK  FROM ReservationRequirements  ORDER BY StartTime DESC))  BEGIN  ;THROW 60000, 'Not enough orders done!!!', 1;  END  ELSE  BEGIN  DECLARE @TableProposition INT  SELECT TOP 1 TA.TableID  FROM Tables TA  WHERE TableCapacity >= @NumOfPeople  AND (  SELECT O.OrderID  FROM Orders O  INNER JOIN ReservationTable RT ON O.OrderID = RT.OrderID  INNER JOIN Tables T ON RT.TableID = T.TableID  WHERE ((RequiredTime > @DateTime AND  Requiredtime <  DATEADD(HOUR, @ReservationDuration, @DateTime))  OR (DATEADD(HOUR, ReservationDuration,  RequiredTime) > @DateTime AND  DATEADD(HOUR, ReservationDuration,  Requiredtime) <  DATEADD(HOUR, @ReservationDuration, @DateTime))  OR (RequiredTime < @DateTime AND  DATEADD(HOUR, ReservationDuration,  RequiredTime) >  DATEADD(HOUR, @ReservationDuration, @DateTime)))  AND T.TableID = TA.TableID  ) IS NULL  IF (@TableProposition IS NULL)  BEGIN  ;THROW 60000, 'No free tables left', 1;  END  ELSE  BEGIN  INSERT INTO ReservationTable(OrderID, TableID)  VALUES (@OrderID, @TableProposition)  END  END  END END |
| --- |

35. Potwierdzenie złożonego zamówienia

| CREATE PROCEDURE usp\_orderPlaced @OrderID INT AS BEGIN  IF ((SELECT OrderID FROM Orders WHERE OrderID = @OrderID) IS NULL)  BEGIN  ;THROW 60000, 'There no such order!!!', 1;  END  ELSE  IF ((SELECT StatusID FROM Orders WHERE OrderID = @OrderID) != 6)  BEGIN  ;THROW 60000, 'Order has already been placed!!!', 1;  END  ELSE  BEGIN  UPDATE Orders  SET StatusID = 1  WHERE OrderID = @OrderID  END END |
| --- |

36. Dodanie osoby do rezerwacji

| CREATE PROCEDURE usp\_addReservationPerson @VarOrderID INT,  @VarFirstName VARCHAR(50),  @VarLastName VARCHAR(50)  AS  BEGIN  IF ((@VarOrderID) NOT IN (SELECT OrderID  FROM Orders))  BEGIN  ;THROW 60000, 'That email is already used!!!', 1;  END   DECLARE @NumberOfPeopleOnOrder INT  SET @NumberOfPeopleOnOrder = (SELECT COUNT(\*)  FROM ReservationPerson  WHERE OrderID = @VarOrderID)   DECLARE @PossibleSlots INT  SET @PossibleSlots = (  SELECT SUM(TableCapacity)  FROM Tables  JOIN ReservationTable RT2  ON Tables.TableID = RT2.TableID  WHERE RT2.OrderID = @VarOrderID  )   IF (@PossibleSlots IS NOT NULL AND  @PossibleSlots >= (@NumberOfPeopleOnOrder + 1))  BEGIN   INSERT INTO ReservationPerson (OrderID, FirstName, LastName)  VALUES (@VarOrderID, @VarFirstName, @VarLastName)  END  ELSE  BEGIN  ;THROW 60000, 'There is no place on that reservation', 1;  END  END |
| --- |

1. **Indeksy**

Indeksy dodano tylko w tabelach, które mają wysoki stosunek zapytań do wstawiania nowych rekordów albo uaktualniania starych.

| CREATE INDEX MenuIndex ON Menu (StartDate, EndDate, IsReadyForDisplay)  CREATE INDEX DiscountsIndex ON Discounts (DiscountType, DiscountAcquisitionDate, CustomerID)  CREATE INDEX OrdersIndex ON Orders (OrderTime, CustomerID) |
| --- |