Podstawy baz danych

pt 9:45
nr zespołu: 21
Autorzy: Iga Szaflik, Paweł Małkowski, Mikołaj Gaweł

1. Wymagania i funkcje systemu

Zaimplementowana baza danych realizuje poniższe funkcje:

Produkcja:

- obliczanie kosztu produkcji danego towaru
- określenie czasu potrzebnego do produkcji (na podstawie wydajności)
- umożliwienie dokonywania preorderów (rezerwacja produkcji pod konkretne zamówienie)

Sprzedaż:

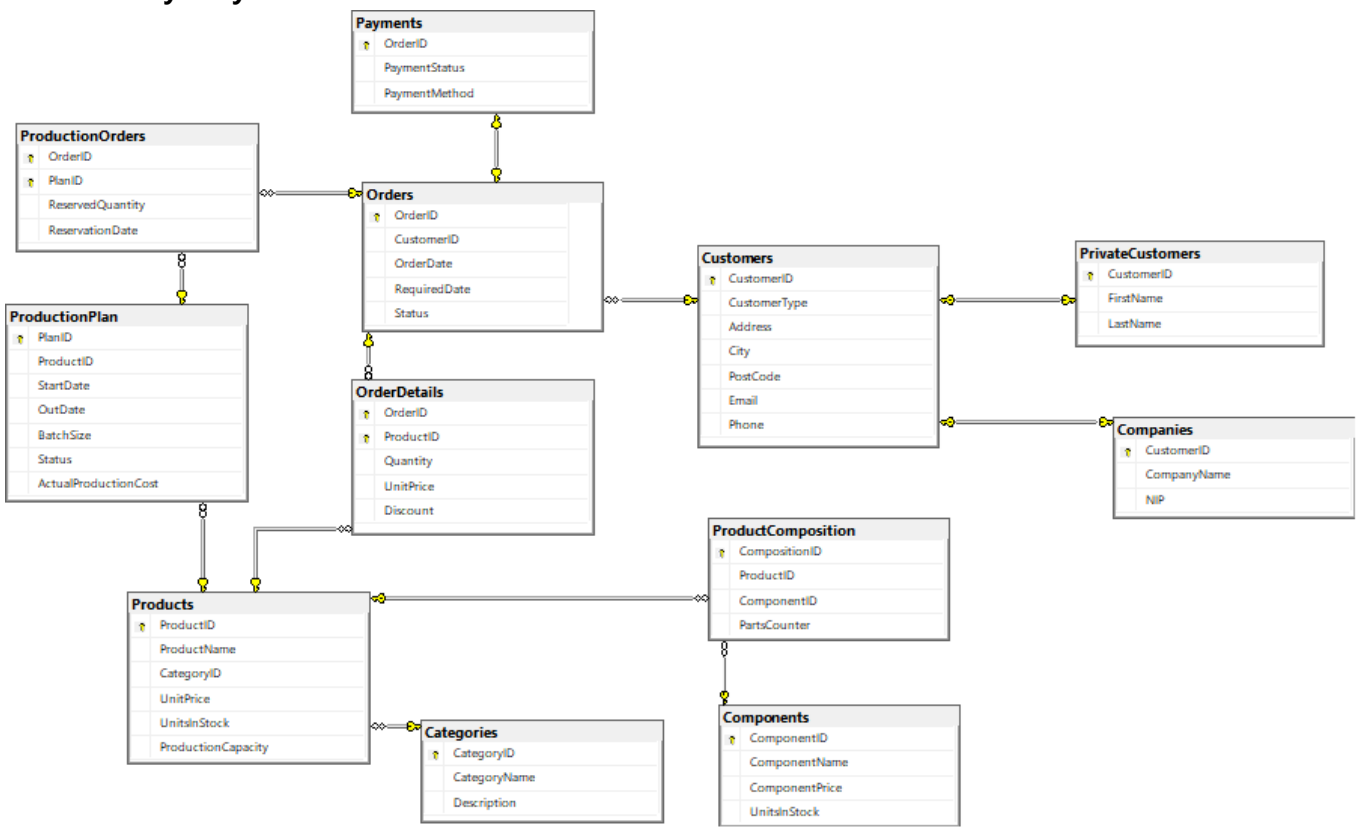
- obsługa klientów indywidualnych oraz firm (rozdzielenie struktur)
- obsługa statusów płatności
- możliwość udzielenia rabatu na konkretne pozycje zamówienia

Analiza:

- wprowadza funkcje służące analizie danych i kosztów produkcji

2. Baza danych

Schemat bazy danych



Nasza baza danych składa się z następujących tabel:

Produkty/Magazyn:

- Kategorie (`Categories`)
- Części do produkcji (`Components`)
- Produkty (`Products`)
- Skład produktu (`ProductComposition`)

Sprzedaż i Klienci:

- Klienci - tabela bazowa (`Customers`)
- Klienci Indywidualni (`PrivateCustomers`)
- Firmy (`Companies`)
- Zamówienia (`Orders`)
- Szczegóły zamówienia (`OrderDetails`)
- Płatności (`Payments`)

Produkcja:

- Plan Produkcji (`ProductionPlan`)
- Rezerwacje Produkcyjne (`ProductionOrders`)

Implementacja struktur bazy danych (Kod DDL)

Poniżej znajduje się kod tworzący tabele z uwzględnieniem warunków integralności oraz relacji.

Categories

```
CREATE TABLE Categories (
    CategoryID INT IDENTITY(1,1) PRIMARY KEY,
    CategoryName NVARCHAR(50) NOT NULL,
    Description NVARCHAR(500)
);
```

Jest to tabela służąca do definiowania kategorii produktów. Umożliwia logiczne uporządkowanie produktów w magazynie oraz ułatwia generowanie raportów sprzedaży w podziale na grupy.

Nazwa atrybutu	Typ	Opis/Uwagi
CategoryID	INT	Klucz główny (PK). Pole z autoinkrementacją (IDENTITY) – unikalny numer nadawany automatycznie przez bazę danych.
CategoryName	NVARCHAR(50)	Nazwa kategorii. Pole wymagane (NOT NULL).
Description	NVARCHAR(500)	Dodatkowy opis tekstowy wyjaśniający, co wchodzi w skład danej kategorii.

Components

```
CREATE TABLE Components (
    ComponentID INT IDENTITY(1,1) PRIMARY KEY,
    ComponentName NVARCHAR(100) NOT NULL,
    ComponentPrice DECIMAL(10,2) NOT NULL,
    UnitsInStock INT DEFAULT 0
);
```

Tabela ta pełni rolę magazynu komponentów. Przechowuje szczegółowe informacje o wszystkich elementach składowych niezbędnych do wytworzenia produktów. Służy również do inwentaryzacji (śledzenia stanu) oraz wyliczania kosztów produkcji.

Nazwa atrybutu	Typ	Opis/Uwagi
ComponentID	INT	Klucz główny (PK) z autoinkrementacją. Unikalny identyfikator danej części w systemie.
ComponentName	NVARCHAR(100)	Nazwa części. Pole wymagane.
ComponentPrice	DECIMAL(10,2)	Jednostkowy koszt danej części. Typ DECIMAL zapewnia precyzję dla wartości pieniężnych.
UnitsInStock	INT	Aktualna ilość sztuk dostępna w magazynie. Domyślnie ustawiona na 0 (DEFAULT 0).

Products

```
CREATE TABLE Products (
    ProductID INT IDENTITY(1,1) PRIMARY KEY,
    ProductName NVARCHAR(100) NOT NULL,
    CategoryID INT FOREIGN KEY REFERENCES Categories(CategoryID),
    UnitPrice DECIMAL(10,2) NOT NULL,
    UnitsInStock INT DEFAULT 0,
    ProductionCapacity INT NOT NULL
);
```

Jest to centralna tabela systemu reprezentująca katalog wyrobów gotowych przeznaczonych do sprzedaży. Oprócz podstawowych danych handlowych i stanów magazynowych, przechowuje kluczowy parametr ProductionCapacity. Jest on niezbędny do planowania produkcji, pozwalając oszacować czas potrzebny na realizację zamówienia.

Nazwa atrybutu	Typ	Opis/Uwagi
ProductID	INT	Klucz główny (PK) z autoinkrementacją. Unikalny identyfikator produktu.
ProductName	NVARCHAR(100)	Pełna nazwa handlowa produktu. Pole wymagane.
CategoryID	INT	Klucz obcy (FK) wiążący produkt z tabelą Categories.
UnitPrice	DECIMAL(10,2)	Katalogowa cena sprzedaży produktu.
UnitsInStock	INT	Aktualny stan magazynowy produktów. Domyślnie 0.
ProductionCapacity	INT	Zdolność produkcyjna (liczba sztuk na dzień). Parametr służący do wyznaczania harmonogramu.

ProductComposition

```
CREATE TABLE ProductComposition (
    CompositionID INT IDENTITY(1,1) PRIMARY KEY,
    ProductID INT FOREIGN KEY REFERENCES Products(ProductID),
    ComponentID INT FOREIGN KEY REFERENCES Components(ComponentID),
    PartsCounter DECIMAL(10,2) NOT NULL
);
```

Tabela ta realizuje funkcję listy materiałowej. Jest to tabela, która definiuje "przepis" na dany produkt. Określa relację wiele-do-wielu między produktami a komponentami, wskazując dokładnie, jakie części i w jakiej ilości są potrzebne do wyprodukowania jednej sztuki danego produktu.

Nazwa atrybutu	Typ	Opis/Uwagi
CompositionID	INT	Klucz główny (PK) z autoinkrementacją. Unikalny identyfikator wpisu w recepturze.

Nazwa atrybutu	Typ	Opis/Uwagi
ProductID	INT	Klucz obcy (FK) wskazujący na produkt z tabeli Products, którego dotyczy ten składnik.
ComponentID	INT	Klucz obcy (FK) wskazujący na część z tabeli Components.
PartsCounter	DECIMAL(10,2)	Ilość danej części niezbędna do wytworzenia jednej sztuki produktu końcowego.

Customers

```
CREATE TABLE Customers(
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
    CustomerType NVARCHAR(20) NOT NULL CHECK (CustomerType IN ('Private', 'Company')),
    Address NVARCHAR(100),
    City NVARCHAR(50),
    PostCode VARCHAR(10),
    Email VARCHAR(100),
    Phone VARCHAR(20)
);
```

Tabela bazowa dla wszystkich kontrahentów firmy. Przechowuje dane wspólne zarówno dla klientów indywidualnych, jak i firm, takie jak dane adresowe. Zastosowaliśmy model dziedziczenia, gdzie Customers jest encją nadrzędną, a PrivateCustomers i Companies są tabelami podrzędnymi. Kolumna CustomerType określa charakter klienta.

Nazwa atrybutu	Typ	Opis/Uwagi
CustomerID	INT	Klucz Główny (PK). Unikalny identyfikator klienta. Baza danych automatycznie nadaje kolejny numer przy dodawaniu nowego rekordu.
CustomerType	NVARCHAR(20)	Określa typ klienta. Wymusza jedną z dwóch wartości: 'Private' (osoba prywatna) lub 'Company' (firma). Jest to pole wymagalne (NOT NULL).
Address	NVARCHAR(100)	Ulica oraz numer domu/lokalu klienta.
City	NVARCHAR(50)	Miasto zamieszkania lub siedziby firmy.
PostCode	NVARCHAR(10)	Kod pocztowy. Używamy VARCHAR, aby obsłużyć kody zawierające myślniki lub formaty zagraniczne.
Email	VARCHAR(100)	Adres e-mail.
Phone	VARCHAR(20)	Numer telefonu kontaktowego. Typ tekstowy pozwala na zapis numeru z prefiksem kierunkowym.

PrivateCustomers

```
CREATE TABLE PrivateCustomers(
    CustomerID INT PRIMARY KEY FOREIGN KEY REFERENCES Customers(CustomerID),
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL
);
```

Tabela przechowująca szczegółowe dane dotyczące klientów indywidualnych. Jest to tabela podrzędna w relacji dziedziczenia względem tabeli Customers. Relacja między tymi tabelami to 1:1.

Nazwa atrybutu	Typ	Opis/Uwagi
CustomerID	INT	Klucz Główny (PK) i Klucz Obcy (FK). Pole to wskazuje na rekord w tabeli Customers.
FirstName	NVARCHAR(50)	Imię klienta indywidualnego. Pole wymagane (NOT NULL).
LastName	NVARCHAR(50)	Nazwisko klienta indywidualnego. Pole wymagane (NOT NULL).

Companies

```
CREATE TABLE Companies (
    CustomerID INT PRIMARY KEY FOREIGN KEY REFERENCES Customers(CustomerID),
    CompanyName NVARCHAR(100) NOT NULL,
    NIP VARCHAR(15)
);
```

Tabela dedykowana dla firm. Podobnie jak w przypadku klientów prywatnych, jest to tabela podrzędna względem Customers. Przechowuje dane identyfikacyjne podmiotu gospodarczego niezbędne np. do wystawienia faktury.

Nazwa atrybutu	Typ	Opis/Uwagi
CustomerID	INT	Klucz Główny (PK) i Klucz Obcy (FK). Identyfikator firmy, będący jednocześnie odniesieniem do ogólnego rekordu w tabeli Customers.
CompanyName	NVARCHAR(100)	Pełna nazwa rejestrowa firmy. Pole wymagane (NOT NULL).
NIP	VARCHAR(15)	Numer Identyfikacji Podatkowej. Przechowywany jako tekst, aby umożliwić zapis ewentualnych kresek, choć zalecany jest zapis ciągły.

Payments

```
CREATE TABLE Payments(
    OrderID INT PRIMARY KEY FOREIGN KEY REFERENCES Orders(OrderID),
    PaymentStatus VARCHAR(20) DEFAULT 'Unpaid' CHECK (PaymentStatus IN ('Pending', 'Paid', 'Unpaid')),
    PaymentMethod VARCHAR(50)
);
```

Tabela obsługująca statusy finansowe zamówień. Zaprojektowana w relacji 1:1 do tabeli Orders. Dzięki niej realizujemy funkcjonalność systemu płatności i możemy kontrolować, które zamówienia zostały już opłacone przez klienta.

Nazwa atrybutu	Typ	Opis/Uwagi
OrderID	INT	Klucz Główny (PK) i Klucz Obcy (FK). Odwołuje się bezpośrednio do identyfikatora zamówienia w tabeli Orders. Dzięki temu rozwiązaniu nie ma możliwości stworzenia płatności bez istniejącego zamówienia.
PaymentStatus	NVARCHAR(20)	Określa aktualny stan płatności. Dzięki zastosowaniu CHECK, pole akceptuje tylko zdefiniowane stany (oczekujące, opłacone, nieopłacone), co zapobiega błędom. Domyślnie system przypisuje tu wartość 'Unpaid'.
PaymentMethod	VARCHAR(50)	Informacja o wybranej metodzie płatności.

Orders

```
CREATE TABLE Orders (
    OrderID INT IDENTITY(1,1) PRIMARY KEY,
    CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
    OrderDate DATETIME DEFAULT GETDATE(),
    RequiredDate DATETIME,
    Status VARCHAR(20) DEFAULT 'Pending' CHECK (Status IN ('Pending', 'In Production', 'Completed', 'Cancelled'))
);
```

Tabela zawiera podstawowe informacje odnośnie złożonych zamówień. Jest łącznikiem między klientami, produkcją i magazynem.

Nazwa atrybutu	Typ	Opis/Uwagi
OrderID	INT	Identyfikator zamówienia stanowiący klucz główny tabeli (PK).
CustomerID	INT	Identyfikator klienta składającego dane zamówienie. Stanowi on klucz obcy wskazujący na tabele Customers (FK).
OrderDate	DATETIME	Data i godzina złożenia zamówienia. Domyślnie jest to aktualna data systemowa.
RequiredDate	DATETIME	Data wymaganej realizacji. Kluczowa w planowaniu zamówień produkcyjnych.

Nazwa atrybutu	Typ	Opis/Uwagi
Status	VARCHAR(20)	Określa fazę, w której aktualnie znajduje się zamówienie. `'Pending'` (oczekujące), `'In Production'` (w produkcji), `'Completed'` (zakończone), `'Cancelled'` (anulowane). Wartość domyślna to `'Pending'`.

OrderDetails

```
CREATE TABLE OrderDetails (
    OrderID INT FOREIGN KEY REFERENCES Orders(OrderID),
    ProductID INT FOREIGN KEY REFERENCES Products(ProductID),
    Quantity INT NOT NULL CHECK (Quantity > 0),
    UnitPrice DECIMAL(10,2) NOT NULL,
    Discount DECIMAL(5,2) DEFAULT 0.00,
    PRIMARY KEY (OrderID, ProductID)
);
```

Tabela zawiera szegóły dotyczące poszczególnych elementów zamówień. Pełni role tablicy łączącej zamówienia (`Orders`) oraz produkty (`Products`) w relacji wiele do wiele.

Nazwa atrybutu	Typ	Opis/Uwagi
OrderID	INT	Identyfikator zamówienia, do którego przypisana jest ta pozycja. Jest to klucz obcy wskazujący na tabelę `Orders`. Współtworzy klucz główny.
ProductID	INT	Identyfikator produktu wchodzącego w skład zamówienia. Klucz obcy wskazujący na tabelę `Products`. Współtworzy klucz główny.
Quantity	INT	Liczba sztuk danego produktu w tym konkretnym zamówieniu. Na kolumnę nałożone jest ograniczenie wymuszające wartość dodatnią (Quantity > 0).
UnitPrice	DECIMAL(10,2)	Cena jednostkowa produktu w momencie składania zamówienia. Jest to cena "zamrożona" dla transakcji – może różnić się od aktualnej ceny katalogowej w tabeli Products.
Discount	DECIMAL(5,2)	Rabat procentowy udzielony na tę konkretną pozycję zamówienia. Wartość domyślna to 0.00.

ProductionPlan

```
CREATE TABLE ProductionPlan (
    PlanID INT IDENTITY(1,1) PRIMARY KEY,
    ProductID INT FOREIGN KEY REFERENCES Products(ProductID),
    StartDate DATE,
    OutDate DATE,
    BatchSize INT,
    Status VARCHAR(20) DEFAULT 'Planned' CHECK (Status IN ('Planned', 'In Production', 'Completed')),
    ActualProductionCost DECIMAL(10,2) NULL
);
```

Tabela służy do zarządzania produkcją. Wiersze reprezentują pojedynczą partię produkcyjną, która wykonana zostanie w danym terminie. Tabela pozwala analizować przebieg produkcji.

Nazwa atrybutu	Typ	Opis/Uwagi
PlanID	INT	Identyfikator partii produkcyjnej. Stanowi klucz główny (PK)
ProductID	INT	Informacja jaki produkt wytwarzany jest w danej partii. Klucz obcy tabeli Products (FK)
StartDate	DATE	Data rozpoczęcia produkcji
OutDate	DATE	Planowana data zakończenia produkcji
BatchSize	INT	Ilość produktów w danej pratii
Status	VARCHAR(20)	Określa status produkcji `Planed` (zaplanowana), `In Production` (w trakcie produkcji) oraz `Completed` (zakończona). Domyślnie status ustawiony jest na `Planed`.
ActualProductionCost	DECIMAL(10,2)	Rzeczywisty koszt produkcji całej partii. Obliczany po zakończeniu produkcji.

ProductionOrders

```
CREATE TABLE ProductionOrders (
    OrderID INT FOREIGN KEY REFERENCES Orders(OrderID),
    PlanID INT FOREIGN KEY REFERENCES ProductionPlan(PlanID),
    ReservedQuantity INT NOT NULL CHECK (ReservedQuantity > 0),
    ReservationDate DATETIME DEFAULT GETDATE(),
    PRIMARY KEY (OrderID, PlanID)
);
```

Jest to tabela łącząca zamówienia (`Orders`) wraz z planowaną produkcją (`ProductionPlan`) umożliwia to rezerwacje produktów, które dopiero zostaną wyprodukowane.

Nazwa atrybutu	Typ	Opis/Uwagi
OrderID	INT	Identyfikator zamówienia. Jest to klucz obcy wskazujący na tabele zamówień (Orders). Składa się on na klucz główny zamówień produkcyjnych.

Nazwa atrybutu	Typ	Opis/Uwagi
PlanID	INT	Numer partii produkcyjnej będący kluczem obcym wskazującym na tabele planu produkcji (ProductionPlan). Składa się na klucz główny zamówień produkcyjnych.
ReservedQuantity	INT	Liczba zarezerwowanych sztuk produktów. Musi być dodatnia
ReservationDate	DATETIME	Data i godzina złożonej rezerwacji. Domyślnie przyjmuje datę tworzenia rekordu.

3. Widoki, procedury/funkcje, triggery

Widoki

Analiza sprzedaży z uwzględnieniem rabatów i wartości zamówień

(`v_FullSalesReport`)
Widok ten łączy dane z kilku tabel (OrderDetails, Orders, Products, Categories, Customers), aby pokazać pełną listę sprzedaży w jednym czytelnym zestawieniu. Mamy gotową listę z nazwami klientów i produktów oraz wylicza końcową cenę każdej pozycji po uwzględnieniu rabatu, co ułatwia przeglądanie historii transakcji.

```
CREATE VIEW v_FullSalesReport AS
SELECT
    o.OrderID,
    o.OrderDate,
    c.CategoryName,
    p.ProductName,
    od.Quantity,
    od.UnitPrice,
    (od.Quantity * od.UnitPrice * (1 - od.Discount/100)) AS TotalRowValue,
    cust.CustomerType
FROM OrderDetails od
JOIN Orders o ON od.OrderID = o.OrderID
JOIN Products p ON od.ProductID = p.ProductID
JOIN Categories c ON p.CategoryID = c.CategoryID
JOIN Customers cust ON o.CustomerID = cust.CustomerID;
```

Monitoring stanów magazynowych i planowanie produkcji.

(`v_InventoryStatus`)
Ten widok służy do szybkiego sprawdzania stanu magazynu. Oprócz prostej liczby sztuk, wyświetla też status, dzięki czemu od razu widać, czego brakuje. Dodatkowo, na podstawie wydajności produkcji, widok oblicza szacunkowy czas (w dniach) potrzebny na dorobienie partii 50 sztuk danego produktu.

```sql
CREATE VIEW v_InventoryStatus AS
SELECT
    p.ProductID,
    p.ProductName,
    c.CategoryName,
    p.UnitsInStock,
    p.ProductionCapacity,
    CASE
        WHEN p.UnitsInStock = 0 THEN 'Out of Stock'
        WHEN p.UnitsInStock < 10 THEN 'Low Stock - Reorder'
        ELSE 'Available'
    END AS StockStatus,
    CAST(50.0 / NULLIF(p.ProductionCapacity, 0) AS DECIMAL(5,1)) AS DaysToProduceBatch50
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID;
```

Analiza kosztów materiałowych

(v_ProductProductionCosts)
Widok oblicza dokładny koszt materiałów potrzebnych do wyprodukowania jednej sztuki mebla. Sumuje ceny wszystkich części zdefiniowanych w składzie produktu. Dzięki temu możemy porównać koszt produkcji z ceną sprzedaży i zobaczyć, ile zarabiamy na jednym produkcie (marża).

```sql
CREATE VIEW v_ProductProductionCosts AS
SELECT
    p.ProductName,
    c.CategoryName,
    SUM(comp.ComponentPrice * pc.PartsCounter) AS TotalMaterialCost,
    p.UnitPrice AS SellingPrice,
    (p.UnitPrice - SUM(comp.ComponentPrice * pc.PartsCounter)) AS EstimatedMargin
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID
JOIN ProductComposition pc ON p.ProductID = pc.ProductID
JOIN Components comp ON pc.ComponentID = comp.ComponentID
GROUP BY p.ProductName, c.CategoryName, p.UnitPrice;
```

Karta historii klienta

(v_CustomerOrderSummary)
Jest to podsumowanie historii zakupów dla każdego klienta. Widok zlicza, ile razy dany klient składał zamówienie i ile łącznie wydał pieniędzy. Pokazuje również średni procent otrzymanego rabatu.

```sql
CREATE VIEW v_CustomerOrderSummary AS
SELECT
    c.CustomerID,
    CASE
        WHEN c.CustomerType = 'Company' THEN comp.CompanyName
        ELSE pc.FirstName + ' ' + pc.LastName
    END AS CustomerName,
    COUNT(o.OrderID) AS OrderCount,
    MIN(o.OrderDate) AS FirstOrderDate,
    MAX(o.OrderDate) AS LastOrderDate,
    ISNULL(CAST(SUM(od.Quantity * od.UnitPrice * (1 - od.Discount/100)) AS DECIMAL(10,2)), 0.00) AS TotalSpent,
    CAST(AVG(od.Discount) AS DECIMAL(5,2)) AS AvgDiscountPct
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
LEFT JOIN OrderDetails od ON o.OrderID = od.OrderID
LEFT JOIN Companies comp ON c.CustomerID = comp.CustomerID
LEFT JOIN PrivateCustomers pc ON c.CustomerID = pc.CustomerID
GROUP BY c.CustomerID, c.CustomerType, comp.CompanyName, pc.FirstName, pc.LastName;
```

Funkcje

1. Obliczanie kosztu materiałowego produktu

(`CalculateProductionCost`)

Funkcja oblicza sumaryczny koszt surowców potrzebnych do wytworzenia jednej sztuki produktu na podstawie jego receptury (tabela `ProductComposition`) oraz aktualnych cen komponentów w magazynie.

```sql
CREATE FUNCTION dbo.CalculateProductionCost (@ProductID INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @TotalCost DECIMAL(10,2);

    SELECT @TotalCost = SUM(c.ComponentPrice * pc.PartsCounter)
    FROM ProductComposition pc
    JOIN Components c ON pc.ComponentID = c.ComponentID
    WHERE pc.ProductID = @ProductID;

    RETURN ISNULL(@TotalCost, 0);
END;
```

2. Szacowanie czasu produkcji

(`EstimateProductionTime`)

Funkcja służy do planowania mocy przerobowych. Szacuje czas (w godzinach) potrzebny na realizację zlecenia o zadanej wielkości (`@TargetQuantity`). Obliczenia bazują na parametrze wydajności (`ProductionCapacity`) zdefiniowanym dla każdego produktu. Funkcja posiada zabezpieczenie przed błędem dzielenia przez zero.

```sql
CREATE FUNCTION dbo.EstimateProductionTime (@ProductID INT, @TargetQuantity INT)
RETURNS DECIMAL(10,1)
AS
BEGIN
    DECLARE @Capacity INT;
    DECLARE @EstimatedHours DECIMAL(10,1);

    SELECT @Capacity = ProductionCapacity
    FROM Products
    WHERE ProductID = @ProductID;

    IF @Capacity IS NULL OR @Capacity = 0
        SET @EstimatedHours = 0;
    ELSE
        SET @EstimatedHours = CAST(@TargetQuantity AS DECIMAL) / CAST(@Capacity AS DECIMAL);

    RETURN @EstimatedHours;
END;
```

3. Całkowita wartość klienta (LTV)

(`CalculateCustomerValue`)

Funkcja oblicza całkowitą wartość przychodu wygenerowanego przez danego klienta. Sumuje ona wartość wszystkich pozycji z zamówień o statusie Completed, uwzględniając przy tym indywidualnie przyznane rabaty (Discount).

```sql
CREATE FUNCTION dbo.CalculateCustomerValue (@CustomerID INT)
RETURNS DECIMAL(12,2)
AS
BEGIN
    DECLARE @SumaWydatkow DECIMAL(12,2);

    SELECT @SumaWydatkow = SUM((od.Quantity * od.UnitPrice) * (1.00 - ISNULL(od.Discount/100, 0)))
    FROM Orders o
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    WHERE o.CustomerID = @CustomerID
      AND o.Status = 'Completed';

    RETURN ISNULL(@SumaWydatkow, 0);
END;
```

Trigger-y

1. Aktualizacja stanu magazynu części po rozpoczęciu składania produktu

(`trg_ProductionPlan_Start_ReduceComponents`)
Celem tego wyzwalacza jest sprawdzenie czy w magazynie znajduje się wystarczająca ilość częsci by złożyć produkt,

którego produkcja została zaplanowana (zmiana statusu na `in Production`). Jeżeli nie posiadamy w magazynie odpowiedniej liczby części produkcja jest blokowana. W przeciwnym wypadku pobierana jest odpowiednia ilość częsci z magazynu

```
CREATE TRIGGER trg_ProductionPlan_Start_ReduceComponents
ON ProductionPlan
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN deleted d ON i.PlanID = d.PlanID
        WHERE i.Status = 'In Production'
          AND d.Status <> 'In Production'
    )
    BEGIN
        IF EXISTS (
            SELECT 1
            FROM Components C
            JOIN ProductComposition PC ON C.ComponentID = PC.ComponentID
            JOIN inserted i ON PC.ProductID = i.ProductID
            JOIN deleted d ON i.PlanID = d.PlanID
            WHERE i.Status = 'In Production'
              AND d.Status <> 'In Production'
              AND (C.UnitsInStock - (PC.PartsCounter * i.BatchSize)) < 0
        )
        BEGIN
            RAISERROR ('Błąd: Brak wystarczającej liczby komponentów w magazynie, aby rozpocząć produkcję.', 16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END

        UPDATE C
        SET C.UnitsInStock = C.UnitsInStock - (PC.PartsCounter * i.BatchSize)
        FROM Components C
        JOIN ProductComposition PC ON C.ComponentID = PC.ComponentID
        JOIN inserted i ON PC.ProductID = i.ProductID
        JOIN deleted d ON i.PlanID = d.PlanID
        WHERE i.Status = 'In Production'
          AND d.Status <> 'In Production';
    END
END;
```

2. Aktualizacja dostępnej liczby produktów po zakończeniu produkcji

(`trg_UpdateProductStock_OnProductionComplete`)
Wyzwalacz automatycznie aktualizuje ilość dostępnych produktów w momencie ukończenia, ich składania (zmiana statusu na `Completed`).

```
CREATE TRIGGER trg_UpdateProductStock_OnProductionComplete
ON ProductionPlan
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF UPDATE(Status)
    BEGIN
        UPDATE p
        SET p.UnitsInStock = p.UnitsInStock + i.BatchSize
        FROM Products p
        INNER JOIN inserted i ON p.ProductID = i.ProductID
        INNER JOIN deleted d ON i.PlanID = d.PlanID
        WHERE i.Status = 'Completed' AND (d.Status IS NULL OR d.Status <> 'Completed');
    END
END
GO
```

3. Aktualizacja dostępnej liczby produktów po złożeniu zamówienia

(`trg_ReduceProductStock_OnOrder`)
Wyzwalacz realizuje aktualizacje dostępnej liczby produktów w momencie dokonania zamówienia (`Pending`).

```
CREATE TRIGGER trg_ReduceProductStock_OnOrder
ON OrderDetails
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE P
    SET P.UnitsInStock = P.UnitsInStock - i.Quantity
    FROM Products P
    JOIN inserted i ON P.ProductID = i.ProductID;
END;
```