UNIWERSYTET JAGIELLOŃSKI
Wydział Matematyki i Informatyki

PROJEKT

BAZA DANYCH KSIĘGARNI INTERNETOWEJ

– BAZY DANYCH 2022 –

Autorzy:

Aleksandra Prodziewicz
Paulina Purs
Szymon Majewski

CEL PROJEKTU

Realizacja bazy danych kameralnej księgarni internetowej, przechowującej informacje na temat dostępnych produktów, klientów, pracowników. W swojej mocno uproszczonej formie umożliwiająca jej użycie wraz ze stworzeniem dodatkowych podstawowych ułatwień obsługi.

MOŻLIWOŚCI

Główną możliwością bazy jest obsługa podstawowych zadań koniecznych do poprawnego funkcjonowania księgarni internetowej – zarówno od strony pracownika jak i klienta. Zawarte w niej procedury ułatwiają realizację złożenia zamówienia, założenia konta, zatrudnienia pracownika, zmiany statusu złożonego zamówienia jak i wiele innych uproszczeń funkcjonalności. Jest to ramowa forma bazy danych, pozostawiająca duże pole do rozwoju, zgodne z wybranym kierunkiem oraz dopasowane do odpowiednich przyszłych potrzeb.

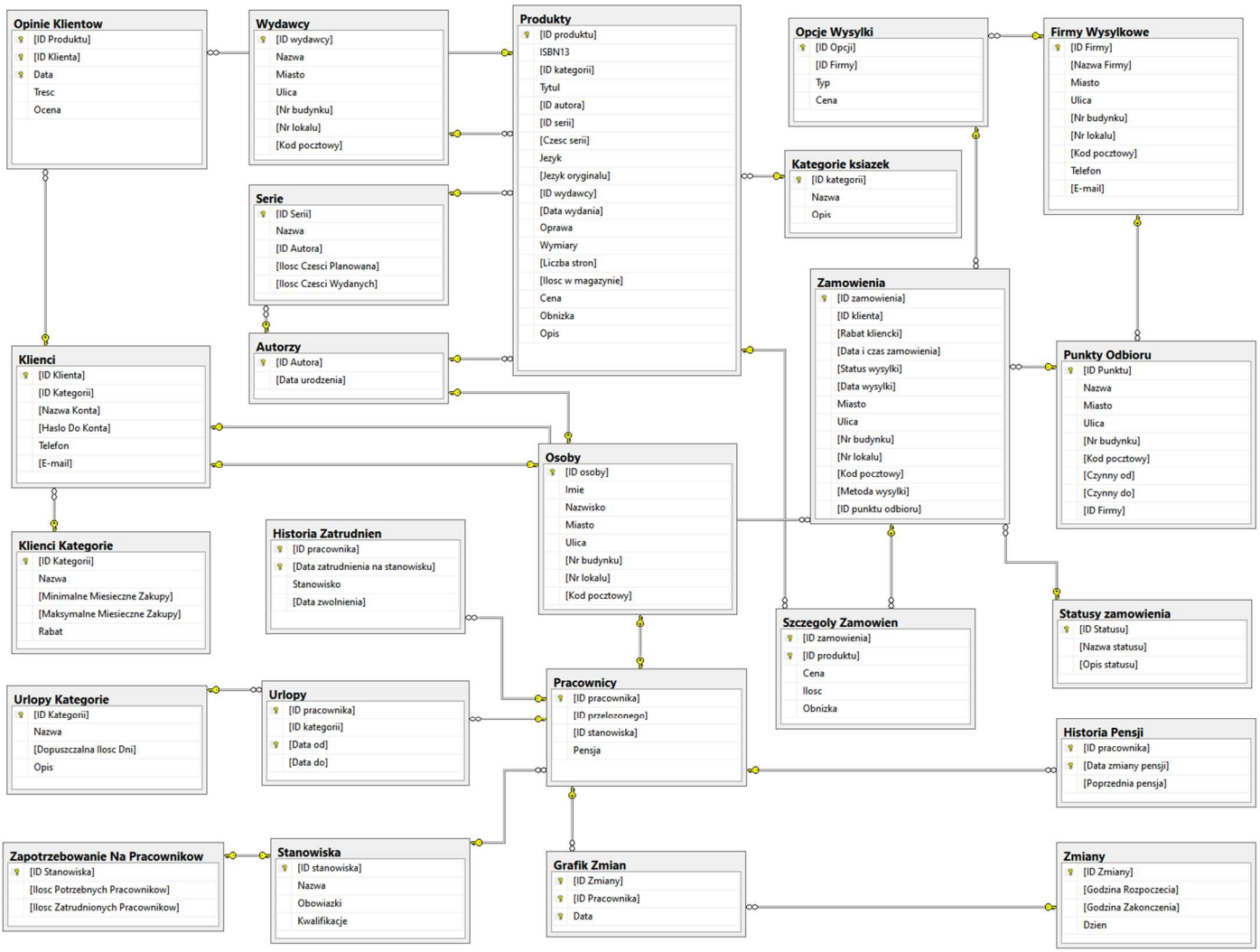
GŁÓWNE ZAŁOŻENIA I OGRANICZENIA

Ograniczyliśmy projekt do podstaw, tzw. dane przykładowe są podane jedynie w celu pokazania ich schematu i funkcjonalności z zastosowaniem różnych funkcji oraz procedur - ze względu na skalę projektu, rzeczywista baza księgarni zawierałaby tysiące produktów. Dalszymi założeniami, które przyjęliśmy na potrzeby projektu są: konieczność założenia konta, by móc zostać klientem naszej księgarni – ponadto każdemu klientowi przypisywana jest kategoria na podstawie jego miesięcznych zakupów, przypisanie tylko jednej kategorii dla jednego produktu, każdy pracownik może mieć jednego bezpośredniego przełożonego oraz każde zamówienie musi przejść przez każdy etap zamówienia, w których nie uwzględniamy możliwości anulowania.

STRATEGIA PIELĘGNACJI BAZY DANYCH

Ze względu na częste zmiany w bazie danych, codziennie, w godzinach nocnych, aby w pewnym stopniu ograniczyć obciążenie serwera, wykonywana będzie różnicowa kopia zapasowa, zaś raz w tygodniu pełny backup całej bazy. Takie działania pozwolą na ewentualne odzyskanie danych utraconych wskutek niewykluczonych awarii.

DIAGRAM ER I SCHEMAT RELACJI



Schemat relacji bazy danych

Autorzy

Column Name	Data Type	Allow Nulls
[ID Autora]	int	☐
[Data urodzenia]	date	☑
		☐

Firmy Wysylkowe

Column Name	Data Type	Allow Nulls
[ID Firmy]	int	☐
[Nazwa Firmy]	nvarchar(100)	☐
Miasto	nvarchar(50)	☐
Ulica	nvarchar(50)	☐
[Nr budynku]	int	☐
[Nr lokalu]	int	☑
[Kod pocztowy]	nvarchar(10)	☐
Telefon	int	☑
[E-mail]	nvarchar(255)	☐
		☐

Grafik Zmian

Column Name	Data Type	Allow Nulls
[ID Zmiany]	int	☐
[ID Pracownika]	int	☐
Data	date	☐
		☐

Historia Pensji

Column Name	Data Type	Allow Nulls
[ID pracownika]	int	☐
[Data zmiany pensji]	date	☐
[Poprzednia pensja]	money	☐
		☐

Historia Zatrudnien

Column Name	Data Type	Allow Nulls
[ID pracownika]	int	☐
[Data zatrudnienia na stanowisku]	date	☐
Stanowisko	int	☐
[Data zwolnienia]	date	☑
		☐

Kategorie Ksiazek

Column Name	Data Type	Allow Nulls
[ID kategorii]	int	☐
Nazwa	nvarchar(50)	☐
Opis	nvarchar(500)	☑
		☐

Klienci

Column Name	Data Type	Allow Nulls
[ID Klienta]	int	☐
[ID Kategorii]	int	☐
[Nazwa Konta]	nvarchar(50)	☐
[Haslo Do Konta]	nvarchar(50)	☐
Telefon	int	☐
[E-mail]	nvarchar(255)	☑
		☐

Klienci Kategorie

Column Name	Data Type	Allow Nulls
[ID Kategorii]	int	☐
Nazwa	nvarchar(50)	☐
[Minimalne Miesieczne ...	smallmoney	☑
[Maksymalne Miesieczn...	smallmoney	☑
Rabat	float	☐
		☐

Opcje Wysylki

Column Name	Data Type	Allow Nulls
[ID Opcji]	int	☐
[ID Firmy]	int	☐
Typ	nvarchar(50)	☐
Cena	smallmoney	☐
		☐

Opinie Klientow

Column Name	Data Type	Allow Nulls
[ID Produktu]	int	☐
[ID Klienta]	int	☐
Data	datetime	☐
Tresc	nvarchar(500)	☑
Ocena	int	☐
		☐

Osoby

Column Name	Data Type	Allow Nulls
[ID osoby]	int	☐
Imie	nvarchar(50)	☐
Nazwisko	nvarchar(50)	☐
Miasto	nvarchar(50)	☑
Ulica	nvarchar(50)	☑
[Nr budynku]	int	☑
[Nr lokalu]	int	☑
[Kod pocztowy]	nvarchar(10)	☑
		☐

Pracownicy

Column Name	Data Type	Allow Nulls
[ID pracownika]	int	☐
[ID przelozonego]	int	☑
[ID stanowiska]	int	☐
Pensja	money	☐
		☐

Produkty

Column Name	Data Type	Allow Nulls
[ID produktu]	int	☐
ISBN13	bigint	☐
[ID kategorii]	int	☐
Tytul	nvarchar(50)	☐
[ID autora]	int	☐
[ID serii]	int	☑
[Czesc serii]	int	☑
Jezyk	nvarchar(50)	☐
[Jezyk oryginalu]	nvarchar(50)	☑
[ID wydawcy]	int	☐
[Data wydania]	date	☐
Oprawa	nvarchar(50)	☐
Wymiary	nvarchar(50)	☐
[Liczba stron]	int	☐
[Ilosc w magazynie]	int	☐
Cena	smallmoney	☐
Obnizka	real	☐
Opis	nvarchar(1000)	☑
		☐

Punkty Odbioru

Column Name	Data Type	Allow Nulls
[ID Punktu]	int	☐
Nazwa	nvarchar(50)	☐
Miasto	nvarchar(50)	☐
Ulica	nvarchar(50)	☐
[Nr budynku]	int	☐
[Kod pocztowy]	nvarchar(10)	☐
[Czynny od]	time(7)	☐
[Czynny do]	time(7)	☐
[ID Firmy]	int	☐
		☐

Serie

Column Name	Data Type	Allow Nulls
[ID Serii]	int	☐
Nazwa	nvarchar(300)	☐
[ID Autora]	int	☐
[Ilosc Czesci Planowana]	int	☑
[Ilosc Czesci Wydanych]	int	☐
		☐

Stanowiska

Column Name	Data Type	Allow Nulls
[ID stanowiska]	int	☐
Nazwa	nvarchar(50)	☐
Obowiazki	nvarchar(200)	☑
Kwalifikacje	nvarchar(200)	☑
		☐

Statusy Zamowienia

Column Name	Data Type	Allow Nulls
[ID Statusu]	int	☐
[Nazwa statusu]	nvarchar(50)	☐
[Opis statusu]	nvarchar(200)	☐
		☐

Szczegoly zamowien

Column Name	Data Type	Allow Nulls
[ID zamowienia]	int	☐
[ID produktu]	int	☐
Cena	smallmoney	☐
Ilosc	int	☐
Obnizka	real	☐
		☐

Urlopy

Column Name	Data Type	Allow Nulls
[ID pracownika]	int	☐
[ID kategorii]	int	☑
[Data od]	date	☐
[Data do]	date	☐
		☐

Urlopy Kategorie

Column Name	Data Type	Allow Nulls
[ID Kategorii]	int	☐
Nazwa	nvarchar(50)	☐
[Dopuszczalna Ilosc Dni]	int	☐
Opis	nvarchar(150)	☑
		☐

Wydawcy

Column Name	Data Type	Allow Nulls
[ID wydawcy]	int	☐
Nazwa	nvarchar(50)	☐
Miasto	nvarchar(50)	☐
Ulica	nvarchar(50)	☐
[Nr budynku]	int	☐
[Nr lokalu]	int	☑
[Kod pocztowy]	nvarchar(10)	☐
		☐

Zamowienia

Column Name	Data Type	Allow Nulls
[ID zamowienia]	int	☐
[ID klienta]	int	☐
[Rabat kliencki]	real	☐
[Data i czas zamowienia]	datetime	☐
[Status wysylki]	int	☐
[Data wysylki]	date	☑
Miasto	nvarchar(50)	☑
Ulica	nvarchar(50)	☑
[Nr budynku]	int	☑
[Nr lokalu]	int	☑
[Kod pocztowy]	nvarchar(10)	☑
[Metoda wysylki]	int	☑
[ID punktu odbioru]	int	☑
		☐

Zapotrzebowanie Na Pracownikow

Column Name	Data Type	Allow Nulls
[ID Stanowiska]	int	☐
[Ilosc Potrzebnych Pracownikow]	int	☐
[Ilosc Zatrudnionych Pracownikow]	int	☐
		☐

Zmiany

Column Name	Data Type	Allow Nulls
[ID Zmiany]	int	☐
[Godzina Rozpoczecia]	time(7)	☐
[Godzina Zakonczenia]	time(7)	☐
Dzien	nvarchar(20)	☐
		☐

SKRYPT TWORZĄCY BAZĘ DANYCH

```
-- Usuniecie bazy danych, jesli istnieje
USE master
IF EXISTS( SELECT * FROM sys.databases WHERE name = 'Ksiegarnia
Internetowa' )
DROP DATABASE [Ksiegarnia Internetowa]
-- Stworzenie bazy danych
CREATE DATABASE [Ksiegarnia Internetowa]
```

Baza danych składa się z 24 tabel.

```
--Stworzenie tabeli Osoby
CREATE TABLE Osoby (
[ID osoby] INT IDENTITY(1,1) PRIMARY KEY,
Imie NVARCHAR(50) NOT NULL,
Nazwisko NVARCHAR(50) NOT NULL,
Miasto NVARCHAR(50),
Ulica NVARCHAR(50),
[Nr budynku] INT,
[Nr lokalu] INT,
[Kod pocztowy] NVARCHAR(10)
);

--przykładowe dane
INSERT INTO Osoby (Imie, Nazwisko, Miasto, Ulica, [Nr budynku], [Nr lokalu], [Kod
pocztowy]) VALUES
('Fonda', 'Lee', NULL, NULL, NULL, NULL, NULL),
('Pascal', 'Garnier', NULL, NULL, NULL, NULL, NULL),
('Robert', 'Wegner', NULL, NULL, NULL, NULL, NULL),
('Alfons', 'Kerdinger', NULL, NULL, NULL, NULL, NULL),
('Aleksandra', 'Kughn', NULL, NULL, NULL, NULL, NULL),
('Eugenia', 'Hipokamp', NULL, NULL, NULL, NULL, NULL),
('Edwart', 'Nencka', NULL, NULL, NULL, NULL, NULL),
('Kazimierz', 'Korfanty', NULL, NULL, NULL, NULL, NULL),
('Bazyli', 'Kornik', NULL, NULL, NULL, NULL, NULL),
('Katarzyna', 'Tucholska', NULL, NULL, NULL, NULL, NULL),
('Julia', 'Kot', NULL, NULL, NULL, NULL, NULL),
('Beata', 'Beddinge', NULL, NULL, NULL, NULL, NULL),
('Matylda', 'Krakers', NULL, NULL, NULL, NULL, NULL),
('Faustyna', 'Dzik', NULL, NULL, NULL, NULL, NULL),
('Beata', 'Trebel', NULL, NULL, NULL, NULL, NULL),
('Pawel', 'Kaczor', NULL, NULL, NULL, NULL, NULL);
```

```
--Stworzenie tabeli Firmy Wysylkowe
CREATE TABLE [Firmy Wysylkowe] (
[ID Firmy] INT IDENTITY(1, 1) PRIMARY KEY,
[Nazwa Firmy] NVARCHAR(100) UNIQUE NOT NULL,
Miasto NVARCHAR(50) NOT NULL,
Ulica NVARCHAR(50) NOT NULL,
[Nr budynku] INT NOT NULL,
[Nr lokalu] INT,
[Kod pocztowy] NVARCHAR(10) NOT NULL,
Telefon INT,
```

```
[E-mail] NVARCHAR(255) UNIQUE NOT NULL
);

--przykładowe dane
INSERT INTO [Firmy Wysylkowe] ([Nazwa Firmy], Miasto, Ulica, [Nr budynku], [Nr
lokalu], [Kod pocztowy], Telefon, [E-mail]) VALUES
('Paczuchy', 'Kraków', 'Lal', '15', NULL, '11-171', 123456789,
'paczuch@paczuchy.pl'),
('xx', 'Kraków', 'Lal', '15', NULL, '11-171', 123456789, 'paczxch@paczuchy.pl'),
('Paczuchyx', 'Kraków', 'Lal', '15', NULL, '11-171', 123456789,
'paczuwqch@paczuchy.pl');


--Stworzenie tabeli Autorzy
CREATE TABLE Autorzy (
[ID Autora] INT REFERENCES Osoby([ID Osoby]) ON DELETE CASCADE PRIMARY KEY,
[Data urodzenia] DATE,
);

--przykładowe dane
INSERT INTO Autorzy ([ID Autora], [Data urodzenia]) VALUES
(1, '1979-03-10'),
(2, '1949-07-04'),
(3, NULL);


--Stworzenie tabeli Statusy Zamowienia
CREATE TABLE [Statusy zamowienia](
[ID Statusu] INT IDENTITY(1, 1) PRIMARY KEY,
[Nazwa statusu] NVARCHAR(50) UNIQUE NOT NULL,
[Opis statusu] NVARCHAR(200) NOT NULL
);

--przykładowe dane
INSERT INTO [Statusy zamowienia]([Nazwa statusu], [Opis statusu]) VALUES
('przyjeto', 'oczekawinie na platnosc'),
('oplacono', 'trafilo do realizjcji'),
('do wyslania', 'oczekuje na wyslke'),
('wyslano', 'zostalo wyslane');


--Stworzenie tabeli Urlopy Kategorie
CREATE TABLE [Urlopy Kategorie] (
[ID Kategorii] INT IDENTITY(1, 1) PRIMARY KEY,
Nazwa NVARCHAR(50) UNIQUE NOT NULL,
[Dopuszczalna Ilosc Dni] INT NOT NULL,
Opis NVARCHAR(150)
);

--przykładowe dane
INSERT INTO [Urlopy Kategorie] (Nazwa, [Dopuszczalna Ilosc Dni], Opis) VALUES
('Urlop wypoczynkowy',31,'celem urlopu jest wypoczynek pracownika i regeneracja
utraconych przez niego sil'),
('Urlop okolicznosciowy',100,'celem urlopu jest zalatwienie spraw osobistych
pracownika, odpowiadajacych okreslonym okolicznosciom wskazanym w k.p.'),
('Urlop na dziecko',100,'macierzynski, tacierzynski, rodzicielski, ojcowski) (cele
urlopu jest opieka nad dzieckiem'),
('Urlop szkoleniowy',31,'celem urlopu jest podnoszenie kwalifikacji zawodowych
pracownika'),
```

```
('Urlop na zadanie',14,'Urlop na zadanie'),
('Urlop bezplatny',47,'Urlop bezplatny');


--Stworzenie tabeli Klienci Kategorie
CREATE TABLE [Klienci Kategorie] (
[ID Kategorii] INT IDENTITY(1, 1) PRIMARY KEY,
Nazwa NVARCHAR(50) UNIQUE NOT NULL,
[Minimalne Miesieczne Zakupy] SMALLMONEY,
[Maksymalne Miesieczne Zakupy] SMALLMONEY,
[Rabat] FLOAT NOT NULL,
);

--przykładowe dane
INSERT INTO [Klienci Kategorie] ( Nazwa, [Minimalne Miesieczne Zakupy],
[Maksymalne Miesieczne Zakupy], Rabat) VALUES
('Poczatkujacy', 0, 100, 0.05),
('Zaawansowany', 100, 700, 0.1),
('Ksiazkoholik', 700, 1000, 0.2),
('Psychofan', 1000, 1500, 0.35);


--Stworzenie tabeli Serie
CREATE TABLE Serie (
[ID Serii] INT IDENTITY(1, 1) PRIMARY KEY,
Nazwa NVARCHAR(300) NOT NULL,
[ID Autora] INT NOT NULL,
[Ilosc Czesci Planowana] INT,
[Ilosc Czesci Wydanych] INT NOT NULL,

FOREIGN KEY ([ID Autora]) REFERENCES Autorzy([ID Autora]) ON DELETE CASCADE
);

--przykładowe dane
INSERT INTO Serie (Nazwa, [ID Autora], [Ilosc Czesci Planowana], [Ilosc Czesci
Wydanych]) VALUES
('The Green Bone Saga', 1, 3, 3),
('Opowieści z meekhańskiego pogranicza', 3, 5, 6);


--Stworzenie tabeli Punkty Odbioru
CREATE TABLE [Punkty Odbioru] (
[ID Punktu] INT IDENTITY(1, 1) PRIMARY KEY,
Nazwa NVARCHAR(50) UNIQUE NOT NULL,
Miasto NVARCHAR(50) NOT NULL,
Ulica NVARCHAR(50) NOT NULL,
[Nr budynku] INT NOT NULL,
[Kod pocztowy] NVARCHAR(10) NOT NULL,
[Czynny od] TIME NOT NULL,
[Czynny do] TIME NOT NULL,
[ID Firmy] INT NOT NULL,

FOREIGN KEY ([ID Firmy]) REFERENCES [Firmy Wysylkowe]([ID Firmy]) ON DELETE
CASCADE,
);

--przykładowe dane
INSERT INTO [Punkty Odbioru] (Nazwa, Miasto, Ulica, [Nr budynku], [Kod pocztowy],
[Czynny od], [Czynny do], [ID Firmy]) VALUES
```

```
('A', 'Kraków', 'Lal', '15', '11-171', '08:00:00', '17:00:00', 1),
('AA', 'Kraków', 'Lal', '15', '11-171', '10:00:00', '20:00:00', 1),
('AAA', 'Kraków', 'Lal', '15', '11-171', '10:00:00', '20:00:00', 1),
('AB', 'Kraków', 'Lal', '15', '11-171', '10:00:00', '20:00:00', 2),
('AAB', 'Kraków', 'Lal', '15', '11-171', '10:00:00', '20:00:00', 3);



--Stworzenie tabeli Klienci
CREATE TABLE Klienci (
[ID Klienta] INT REFERENCES Osoby([ID Osoby]) ON DELETE CASCADE PRIMARY KEY,
[ID Kategorii] INT NOT NULL,
[Nazwa Konta] NVARCHAR(50) UNIQUE NOT NULL,
[Haslo Do Konta] NVARCHAR(50) NOT NULL,
Telefon INT NOT NULL,
[E-mail] NVARCHAR(255),

FOREIGN KEY ([ID Kategorii]) REFERENCES [Klienci Kategorie]([ID Kategorii]),
);

--przykładowe dane
INSERT INTO Klienci ([ID Klienta], [ID Kategorii], [Nazwa Konta], [Haslo Do
Konta], Telefon, [E-mail]) VALUES
(4, 1, 'wa', 'weodhdowdih', 123123123, 'sada@ssa.pl'),
(5, 2, 'ad', 'sefzaqqa', 123123123, 'sada@ssa.pl'),
(6, 3, 'fsd', 'dgzaq2nj', 123123123, 'sada@ssa.pl'),
(7, 4, 'tfhj', 'hfash&7ghv', 123123123, 'sada@ssa.pl'),
(8, 1, 'tddgf', 'dhfcjk^hgb', 123123123, 'sada@ssa.pl');


--Stworzenie tabeli Opcje Wysylki
CREATE TABLE [Opcje Wysylki] (
[ID Opcji] INT IDENTITY(1, 1) PRIMARY KEY,
[ID Firmy] INT NOT NULL,
Typ NVARCHAR(50) NOT NULL,
Cena SMALLMONEY NOT NULL,

FOREIGN KEY ([ID Firmy]) REFERENCES [Firmy Wysylkowe]([ID Firmy]) ON DELETE
CASCADE
);

--przykładowe dane
INSERT INTO [Opcje Wysylki] ([ID Firmy], Typ, Cena) VALUES
(1, 'punkt odbioru', 7),
(2, 'punkt odbioru', 9),
(3, 'punkt odbioru', 8),
(3, 'kurier', 15);


--Stworzenie tabeli Zmiany
CREATE TABLE Zmiany (
[ID Zmiany] INT IDENTITY(1, 1) PRIMARY KEY,
[Godzina Rozpoczecia] TIME NOT NULL,
[Godzina Zakonczenia] TIME NOT NULL,
Dzien NVARCHAR(20) NOT NULL
);

--przykładowe dane
```

```
INSERT INTO Zmiany ([Godzina Rozpoczecia], [Godzina Zakonczenia], Dzien) VALUES
('12:00AM','07:30AM', 'pn - czw'),
('07:00AM','01:30PM', 'pn - czw'),
('01:00PM','05:30PM', 'pn - czw'),
('05:00PM','12:00AM', 'pn - czw'),
('12:00AM','07:30AM', 'pt, sb'),
('07:00AM','01:30PM', 'pt, sb'),
('01:00PM','05:30PM', 'pt, sb'),
('05:00PM','12:00AM', 'pt, sb'),
('12:00AM','07:30AM', 'nie, swieta'),
('07:00AM','01:30PM', 'nie, swieta'),
('01:00PM','05:30PM', 'nie, swieta'),
('05:00PM','12:00AM', 'nie, swieta');


--Stworzenie tabeli Stanowiska
CREATE TABLE Stanowiska (
[ID stanowiska] INT IDENTITY(1,1) PRIMARY KEY,
Nazwa NVARCHAR(50) NOT NULL UNIQUE,
Obowiazki NVARCHAR(200),
Kwalifikacje NVARCHAR(200)
);

--przykładowe dane
INSERT INTO Stanowiska (Nazwa, Obowiazki, Kwalifikacje) VALUES
('Specjalista Do Spraw Obslugi Klienta', 'nadwzorowanie pracownikow do spraw
obslogi klienta', 'doswiadczenie w zarządzaniu ludźmi'),
('Pracownik Obslugi Klienta', 'kontakt z klientami, pomoc w rozwiazaniu
problemow', NULL),
('Prezes', 'zarzadznie firma' ,'doswiadczenie w zarzadzaniu'),
('MS SQL Database Administrator', 'zarzadzanie baza danych', 'doswiadczenie w
zarzadzaniu baza danych');


--Stworzenie tabeli Pracownicy
CREATE TABLE Pracownicy (
[ID pracownika] INT REFERENCES Osoby([ID Osoby]) ON DELETE CASCADE PRIMARY KEY,
[ID przelozonego] INT,
[ID stanowiska] INT NOT NULL,
Pensja MONEY NOT NULL

FOREIGN KEY ([ID stanowiska]) REFERENCES Stanowiska([ID Stanowiska]) ON DELETE
CASCADE
);

--przykładowe dane
INSERT INTO Pracownicy ([ID pracownika], [ID przelozonego], [ID stanowiska],
Pensja) VALUES
(9, 14, 1, 90000),
(10, 9, 2, 50000),
(11, 9, 2, 45000),
(12, 9, 2, 60000),
(13, 9, 2, 57000),
(14, NULL, 3, 120000),
(15, 14, 4, 110000),
(16, 14, 4, 110000);


--Stworzenie tabeli Urlopy
```

```sql
CREATE TABLE Urlopy (
[ID pracownika] INT,
[ID kategorii] INT,
[Data od] DATE,
[Data do] DATE NOT NULL,

PRIMARY KEY ([ID pracownika], [Data od]),
FOREIGN KEY ([ID pracownika]) REFERENCES Pracownicy([ID pracownika]) ON DELETE
CASCADE,
FOREIGN KEY ([ID kategorii]) REFERENCES [Urlopy Kategorie]([ID Kategorii]) ON
DELETE CASCADE
);

--przykładowe dane
INSERT INTO Urlopy ([ID pracownika], [ID kategorii], [Data od], [Data do]) VALUES
(9, 2, '2022-01-01', '2022-01-18'),
(10, 4, '2022-01-12', '2022-01-15');


--Stworzenie tabeli Historia Pensji
CREATE TABLE [Historia Pensji] (
[ID pracownika] INT,
[Data zmiany pensji] DATE,
[Poprzednia pensja] MONEY NOT NULL,

PRIMARY KEY ([ID pracownika], [Data zmiany pensji]),
FOREIGN KEY ([ID pracownika]) REFERENCES Pracownicy([ID pracownika]) ON DELETE
CASCADE
);

--przykładowe dane
INSERT INTO [Historia pensji] ([ID pracownika], [Data zmiany pensji], [Poprzednia
pensja]) VALUES
(9, '2021-05-13', 50000),
(10, '2021-09-13', 60000),
(11, '2022-01-13', 90000);


--Stworzenie tabeli Historia Zatrudnien
CREATE TABLE [Historia Zatrudnien] (
[ID pracownika] INT,
[Data zatrudnienia na stanowisku] DATE NOT NULL,
Stanowisko INT NOT NULL,
[Data zwolnienia] DATE,

PRIMARY KEY ([ID pracownika], [Data zatrudnienia na stanowisku]),
FOREIGN KEY ([ID pracownika]) REFERENCES Pracownicy([ID pracownika]) ON DELETE
CASCADE);

--przykładowe dane
INSERT INTO[Historia zatrudnien] ([ID pracownika], [Data zatrudnienia na
stanowisku], Stanowisko, [Data zwolnienia]) VALUES
(9, '2021-05-13', 2, '2022-01-13'),
(10, '2022-01-13', 1, NULL);


--Stworzenie tabeli Zamowienia
CREATE TABLE Zamowienia (
[ID zamowienia] INT IDENTITY(1,1) PRIMARY KEY,
```

```
[ID klienta] INT NOT NULL,
[Rabat kliencki] REAL NOT NULL,
[Data i czas zamowienia] DATETIME NOT NULL,
[Status wysylki] INT NOT NULL,
[Data wysylki] DATE,
Miasto NVARCHAR(50),
Ulica NVARCHAR(50),
[Nr budynku] INT,
[Nr lokalu] INT,
[Kod pocztowy] NVARCHAR(10),
[Metoda wysylki] INT,
[ID punktu odbioru] INT,

FOREIGN KEY ([Metoda wysylki]) REFERENCES [Opcje wysylki]([ID Opcji]),
FOREIGN KEY ([Status wysylki]) REFERENCES [Statusy zamowienia]([ID Statusu]) ON
DELETE CASCADE,
FOREIGN KEY ([ID klienta]) REFERENCES Klienci([ID klienta]) ON DELETE CASCADE,
FOREIGN KEY ([ID punktu odbioru]) REFERENCES [Punkty odbioru]([ID punktu]) ON
DELETE CASCADE,
);

--przykładowe dane
INSERT INTO Zamowienia ([ID klienta], [Rabat kliencki], [Data i czas zamowienia],
[Status Wysylki], [Data wysylki], Miasto, Ulica, [Nr budynku], [Nr lokalu], [Kod
pocztowy], [Metoda wysylki], [ID punktu odbioru]) VALUES
(4, 0.1, '2022-01-12', 4, '2022-01-14', 'Kraków', 'Lal', '15', NULL, '11-171', 4,
NULL ),
(5, 0.05, '2022-01-10', 4, '2022-01-12', 'Kraków', 'Lal', '15', NULL, '11-171', 4,
NULL ),
(6, 0.1, '2022-01-09', 1, NULL, 'Kraków', 'Lal', '15', NULL, '11-171', 4, NULL),
(5, 0.05, '2022-01-21', 3, NULL, 'Kraków', 'Lal', '15', NULL, '11-171', 4, NULL ),
(6, 0.1, '2022-01-12', 2, NULL, 'Kraków', 'Lal', '15', NULL, '11-171', 4, NULL),
(4, 0.1, '2022-01-13', 4, '2022-01-14', NULL, NULL, NULL, NULL, NULL, 1, 1 );


--Stworzenie tabeli Kategorie Ksiazek
CREATE TABLE [Kategorie ksiazek] (
[ID kategorii] INT IDENTITY(1,1) PRIMARY KEY,
Nazwa NVARCHAR(50) NOT NULL,
Opis NVARCHAR(500)
);

--przykładowe dane
INSERT INTO [Kategorie ksiazek] (Nazwa, Opis) VALUES
('fantastyka' ,'Gatunek używający magicznych i innych nadprzyrodzonych form,
motywów, jako pierwszorzędnego składnika fabuły, miejsca, czasu, akcji.'),
('science fiction', 'Literatura fantastyczna, w której konstrukcję świata
przedstawionego uzasadniają wyjaśnienia o charakterze naukowym.' ),
('klasyka', 'Dzieła z różnych okresów historycznych i stylów uznane za
doskonałe.'),
('horror', 'Odmiana fantastyki polegająca na budowaniu świata przedstawionego na
wzór rzeczywistości po to, aby wprowadzić zjawiska niedające się wytłumaczyć bez
odwoływania się do zjawisk nadprzyrodzonych.'),
('kryminal', 'Gatunek, którego fabuła zorganizowana jest wokół zbrodni,
okoliczności dojścia do niej, dochodzenia oraz ujawnienia osoby sprawcy.'),
('komedia', 'Gatunek dramatyczny o pogodnej treści, akcji obfitującej w
wydarzenia, z pomyślnym dla bohatera zakończeniem, z elementami komizmu.' );
```

```sql
--Stworzenie tabeli Wydawcy
CREATE TABLE Wydawcy (
[ID wydawcy] INT IDENTITY(1,1) PRIMARY KEY,
Nazwa NVARCHAR(50) UNIQUE NOT NULL,
Miasto NVARCHAR(50) NOT NULL,
Ulica NVARCHAR(50) NOT NULL,
[Nr budynku] INT NOT NULL,
[Nr lokalu] INT,
[Kod pocztowy] NVARCHAR(10) NOT NULL,
);

--przykładowe dane
INSERT INTO Wydawcy (Nazwa, Miasto, Ulica, [Nr budynku], [Nr lokalu], [Kod
pocztowy]) VALUES
('claroscuro', 'Kraków', 'Lal', '15', NULL, '11-171'),
('Orbit', 'Kraków', 'Lal', '15', NULL, '11-171'),
('Powergraph', 'Kraków', 'Lal', '15', NULL, '11-171');



--Stworzenie tabeli Produkty
CREATE TABLE Produkty (
[ID produktu] INT IDENTITY (1,1) PRIMARY KEY,
ISBN13 BIGINT UNIQUE NOT NULL,
[ID kategorii] INT NOT NULL,
Tytul NVARCHAR(50) NOT NULL,
[ID autora] INT NOT NULL,
[ID serii] INT,
[Czesc serii] INT,
Jezyk NVARCHAR(50) NOT NULL,
[Jezyk oryginalu] NVARCHAR(50),
[ID wydawcy] INT NOT NULL,
[Data wydania] DATE NOT NULL,
[Oprawa] NVARCHAR(50) NOT NULL,
Wymiary NVARCHAR(50) NOT NULL,
[Liczba stron] INT NOT NULL,
[Ilosc w magazynie] INT NOT NULL,
Cena SMALLMONEY NOT NULL,
Obnizka REAL NOT NULL,
Opis NVARCHAR(1000),

FOREIGN KEY ([ID autora]) REFERENCES Autorzy([ID autora]) ON DELETE CASCADE,
FOREIGN KEY ([ID serii]) REFERENCES Serie([ID serii]),
FOREIGN KEY ([ID wydawcy]) REFERENCES Wydawcy([ID wydawcy]) ON DELETE CASCADE,
FOREIGN KEY ([ID kategorii]) REFERENCES [Kategorie ksiazek]([ID kategorii]) ON
DELETE CASCADE
);

--przykładowe dane
INSERT INTO Produkty ([ID kategorii], ISBN13, Tytul, [ID autora], [ID serii],
[Czesc serii],
Jezyk, [Jezyk oryginalu],  [ID wydawcy], [Data wydania], [Oprawa], Wymiary,
[Liczba stron],
[Ilosc w magazynie], Cena, Obnizka, Opis) VALUES
(1, 9780356510514,'Jade City',1,1,1,'angielski', 'angielski',2,'2018-06-28',
'miekka',' 126 x 198 x 36mm', 560, 65, 70, 0.25,
'The Kaul family is one of two crime syndicates that control the island of Kekon.
It is the only place in the world that produces rare magical jade,
```

```
which grants those with the right training and heritage superhuman abilities. The
Green Bone clans of honorable jade-wearing warriors once protected the island from
foreign invasion
--but nowadays, in a bustling post-war metropolis full of fast cars and foreign
money, Green Bone families like the Kauls are primarily involved in commerce,
construction, and the everyday
upkeep of the districts under their protection. When the simmering tension between
the Kauls and their greatest rivals erupts into open violence in the streets, the
outcome of this clan war
will determine the fate of all Green Bones and the future of Kekon itself.'),
(1,9780356510538,'Jade War',1,1,2,'angielski', 'angielski',2,'2019-07-23',
'miekka',' 126 x 198 x 36mm', 624, 47, 70, 0.1,
'On the island of Kekon, the Kaul family is locked in a violent feud for control
of the capital city and the supply of magical jade that endows trained Green Bone
warriors with supernatural powers
they alone have possessed for hundreds of years. Beyond Kekons borders, war is
brewing. Powerful foreign governments and mercenary criminal kingpins alike turn
their eyes on the island nation.
Jade, Kekons most prized resource, could make them rich - or give them the edge
they would need to topple their rivals. Faced with threats on all sides, the Kaul
family is forced to form new and dangerous alliances,
confront enemies in the darkest streets and the tallest office towers, and put
honor aside in order to do whatever it takes to ensure their own survival - and
that of all the Green Bones of Kekon.'),
(1, 9780356510590,'Jade Legacy',1,1,3,'angielski', 'angielski',2,'2021-12-02',
'miekka',' 126 x 198 x 36mm', 752, 81, 70, 0.15,
'The Kauls have been battered by war and tragedy. They are plagued by resentments
and old wounds as their adversaries are on the ascent and their country is riven
by dangerous factions and foreign interference that
could destroy the Green Bone way of life altogether. As a new generation arises,
the clans growing empire is in danger of coming apart. The clan must discern
allies from enemies, set aside aside bloody rivalries,
 and make terrible sacrifices... but even the unbreakable bonds of blood and
loyalty may not be enough to ensure the survival of the Green Bone clans and the
nation they are sworn to protect.'),
(1, 9788361187448,'Polnoc - Poludnie',3,2,1,'polski', 'polski',3, '2012-03-07',
'twarda', '145 x 210 mm', 568, 34, 47, 0,
'Topor i skala -- oto skarby Polnocy. Wiedza o tym nieustraszeni zolnierze Szostej
Kompanii, gorskiego oddzialu, strzegacego polnocnych granic Imperium
Meekhanskiego. A jesli istnieja starcia nie do wygrania?
Jedyne, na co moze wtedy liczyc Gorska Straz, to honor gorali. Miecz i zar --
tylko tyle pozostalo zamaskowanemu wojownikowi z pustynnego Poludnia. Kiedys,
zgodnie ze zwyczajem, zaslanial twarz, by nikt nie wykradl mu duszy.
Dzis nie ma juz duszy, ktora moglby chronic. Czy z bogami mozna walczyc za pomoca
mieczy? Tak, jesli jestes Issarem i nie masz nic do stracenia.'),
(1, 9788361187455,'Wschod - Zachod',3,2,2,'polski', 'polski',3, '2012-03-07',
'twarda', '145 x 210 mm', 680, 34, 47, 0.2,
'Honor i wiernosc, wytrwalosc i zelazna wola. W ksiazkach Roberta Wegnera odzywaja
dawne wartosci, a pomiedzy barwnymi pojedynkami, intrygami, bitwami wielkich armii
i krwawymi starciami jest miejsce na emocje,
ktore potrafia skruszyc serce najwiekszego twardziela. Opowiadania z Polnocy,
Poludnia, Wschodu i Zachodu skladaja sie na epicka opowiesc o egzotycznych
swiatach roznych nacji, jezykow, wierzen i magii.'),
(1, 9788361187417,'Niebo ze stali',3,2,3,'polski', 'polski',3, '2012-03-07',
'twarda', '145 x 210 mm', 624, 34, 47, 0,
'Wozy wygnanych niegdys koczownikow stanely u stop gor, ktore oddzielaja ich od
upragnionej wolnosci. Losy Szostej Kompanii Gorskiej Strazy, dziewczyn z wolnego
czaardanu i malej dziewczynki z rodu Verdanno zaczynaja sie splatac...
 Zanim na niebie o barwie stali wzejdzie slonce, wyzyna splynie krwia.'),
```

```
(1, 9788364384240,'Pamiec wszystkich slow',3,2,4,'polski', 'polski',3, '2015-05-
06', 'twarda', '145 x 210 mm', 702, 34, 47, 0.1,
'Opowiesci z meekhanskiego pogranicza - historie z Polnocy, Poludnia, Wschodu i
Zachodu skladaja sie na egzotyczna opowiesc o swiatach roznych nacji, jezykow,
wierzen i magii.
Napisana z rozmachem "Pamiec wszystkich slow" zabiera bohaterow na niegoscinna
pustynie, w niebezpieczne uliczki pelnych przepychu wschodnich miast czy w samo
serce wyspy opanowanej przez zwasnione rody, siedziby miejscowego boga.
Pomiedzy zemsta rodowa a buntem niewolnikow, wrzuceni miedzy potegi rozgrywajace
swoja partie z Losem, bohaterowie Wegnera musza wybierac, gdy wydaje sie, ze nie
pozostal juz zaden wybor. I nawet niesmiertelni sie ugna, gdy w gre wchodzi honor,
lojalnosc i przysiegi zlozone cieniom tych, ktorzy odeszli.'),
(1, 9788364384868,'Kazde martwe marzenie',3,2,5,'polski', 'polski',3, '2018-11-
28', 'twarda', '145 x 210 mm', 744, 34, 47, 0.2,
'Deana dKllean, niegdys Piesniarka Pamieci i mistrzyni miecza, a dzis rzadzaca
pustynnym ksiestwem wybranka Boga Ognia, stoi na progu wojny. Powstanie
niewolnikow, ktore wybuchlo u poludniowych granic panstwa, zatacza coraz szersze
kregi.
Genno Laskolnyk wraz ze swoim czaardanem wolnych jezdzcow wpada w sam srodek
wojny. Czego szuka wsrod niewolnikow, ktorzy postanowili zrzucic jarzmo krwawych
panow? Tymczasem tysiace mil na polnoc Czerwone Szostki trafiaja na tajemnice,
ktora pochlonela juz niejedna ofiare.
Czy odwaga gorali ocali im zycie? Meekhan splywa krwia i wydaje sie, ze nic juz
nie powstrzyma plomienia, ktory ogarnia Imperium.'),
(3,9788362498291,'Teoria pandy',2, NULL,NULL,'polski', 'francuski',1,'2018-10-01',
'miekka', '125 x 190 mm', 196, 14, 33, 0.1,
'Dzieki swojemu talentowi kulinarnemu i niewymuszonemu luzowi, Gabriel, pojawiwszy
sie nie wiadomo skad, buduje silne wiezi z mieszkancami malego miasteczka w
Bretanii:
z recepcjonistka hotelowa, z dwoma cpunami bez grosza, a przede wszystkim z Jose,
wlascicielem baru Faro. Niczym pluszowa panda wylegujaca sie na kontuarze Faro,
Gabriel poswieca swoj czas tym,
ktorzy do niego przychodza, wiedzeni ciekawoscia, bardziej zauroczeni niz nieufni.
A jednak, gdyby tylko wiedzieli... Kolejny raz Pascal Garnier roztacza przed nami
swoj wyjatkowy czar.'),
(3,9788362498383,'Daleko, dalej',2, NULL,NULL,'polski', 'francuski',1,'2021-03-
01', 'miekka', '125 x 190 mm', 132, 25, 29, 0,
'Szescdziesieciolatek Marc ma, jak sie wydaje, wszystko, co potrzebne do
szczescia. Dusi go jednak poczucie mialkosci jego zycia - byc moze ta mialkosc
odpowiada za to, co nieudane, niezbyt chwalebne, toksyczne.
Magia wielkiej dali ciazy w jego umysle podskorna obietnica nowego poczatku dla
niego i dla jego corki Anne, przebywajacej w zakladzie psychiatrycznym. Marc, nie
informujac nikogo, zabiera wiec corke w podroz nad morze...'),
(3,9788362498154,'Jak sie ma twoj bol?',2, NULL,NULL,'polski',
'francuski',1,'2017-03-01', 'miekka', '125 x 190 mm',176, 51, 33 , 0,
'Smierc jest zajeciem Simona. Starzejacy sie tepiciel szkodnikow przygotowuje sie
do wykonania ostatniego zlecenia. Po drodze zatrzymuje sie w Vals-les-Bains, gdzie
spotyka Bernarda,
serdecznego, nieco naiwnego chlopaka o lagodnym spojrzeniu i dobrym sercu. Bernard
nigdy nie byl nad morzem, a Simon potrzebuje kierowcy. Moze przez kaprys, moze
przez rzeczywista sympatie do czlowieka,
 ktory tak rozni sie od niego, Simon proponuje Bernardowi prace.');


--Stworzenie tabeli Szczegoly Zamowien
CREATE TABLE [Szczegoly Zamowien] (
[ID zamowienia] INT,
[ID produktu] INT,
Cena SMALLMONEY NOT NULL,
```

```
Ilosc INT NOT NULL,
Obnizka REAL NOT NULL,

PRIMARY KEY ([ID zamowienia], [ID produktu]),
FOREIGN KEY ([ID zamowienia]) REFERENCES Zamowienia([ID zamowienia]),
FOREIGN KEY ([ID produktu]) REFERENCES Produkty([ID produktu]) ON DELETE CASCADE
);

--przykładowe dane
INSERT INTO [Szczegoly zamowien] ([ID zamowienia], [ID produktu], Cena, Ilosc,
Obnizka) VALUES
(1, 1, 68, 1, 0.15),
(1, 2, 63, 1, 0.34),
(2, 1, 68, 2, 0.15),
(2, 2, 63, 1, 0.34),
(3, 1, 68, 3, 0.15),
(3, 2, 63, 7, 0.34),
(4, 1, 68, 1, 0.15),
(4, 2, 63, 1, 0.34);


--Stworzenie tabeli Grafik Zmian
CREATE TABLE [Grafik Zmian] (
[ID Zmiany] INT,
[ID Pracownika] INT,
[Data] DATE NOT NULL,

FOREIGN KEY ([ID Pracownika]) REFERENCES Pracownicy([ID Pracownika]) ON DELETE
CASCADE,
FOREIGN KEY ([ID Zmiany]) REFERENCES Zmiany([ID Zmiany]) ON DELETE CASCADE,
PRIMARY KEY ([ID Zmiany], [ID Pracownika], [Data])
);

--przykładowe dane
INSERT INTO [Grafik Zmian] ([ID Zmiany], [ID Pracownika], [Data]) VALUES
(1, 9, '2022-01-13'),
(1, 10, '2022-01-13'),
(2, 11, '2022-01-13'),
(3, 12, '2022-01-13'),
(4, 13, '2022-01-13'),
(1, 9, '2022-01-14'),
(1, 10, '2022-01-14'),
(2, 11, '2022-01-14'),
(3, 12, '2022-01-14'),
(4, 13, '2022-01-14');


--Stworzenie tabeli Zapotrzebowanie Na Pracownikow
CREATE TABLE [Zapotrzebowanie Na Pracownikow] (
[ID Stanowiska] INT PRIMARY KEY,
[Ilosc Potrzebnych Pracownikow] INT NOT NULL,
[Ilosc Zatrudnionych Pracownikow] INT NOT NULL,

FOREIGN KEY ([ID Stanowiska]) REFERENCES Stanowiska([ID Stanowiska]) ON DELETE
CASCADE,
);

--przykładowe dane
```

```sql
INSERT INTO [Zapotrzebowanie Na Pracownikow] ([ID Stanowiska], [Ilosc Potrzebnych
Pracownikow], [Ilosc Zatrudnionych Pracownikow]) VALUES
(2, 10, 4),
(1, 2, 1),
(3, 1, 1),
(4, 4, 2);



--Stworzenie tabeli Opinie Klientow
CREATE TABLE [Opinie Klientow] (
[ID Produktu] INT,
[ID Klienta] INT,
Data DATETIME,
Tresc NVARCHAR(500),
Ocena INT NOT NULL,

FOREIGN KEY ([ID produktu]) REFERENCES Produkty([ID produktu]) ON DELETE CASCADE,
FOREIGN KEY ([ID Klienta]) REFERENCES Klienci([ID Klienta]),
PRIMARY KEY ([ID Klienta], [ID Produktu], Data));

--przykładowe dane
INSERT INTO [Opinie Klientow] ([ID Produktu], [ID Klienta], Data, Tresc, Ocena)
VALUES
(1, 4, '2022-01-14', 'super ksiazka', 5),
(2, 5, '2022-01-14', 'super ksiazka', 5),
(4, 6, '2022-01-14', 'super ksiazka', 5);
```

WIDOKI

W bazie danych utworzone zostało 5 przykładowych widoków.

Wyświetlenie produktów przecenionych.

```
GO
CREATE VIEW produkty_przecenione AS
SELECT * FROM Produkty
WHERE Obnizka > 0;
```

Wyświetlenie wszystkich zamówień znajdujących się w realizacji, czyli przyjętych, opłaconych i niewysłanych.

```
GO
CREATE VIEW zamownienia_realizowane AS
SELECT * FROM Zamowienia Z
      LEFT JOIN [Statusy zamowienia] S ON S.[ID Statusu] = Z.[Status
wysylki]
WHERE [Nazwa statusu] = N'oplacono' OR [Nazwa statusu] = N'do
wyslania';
```

Wyświetlenie wszystkich pracowników obsługi klienta.

```
GO
CREATE VIEW pracownicy_obslugii_klienta AS
SELECT * FROM Pracownicy P
      LEFT JOIN Osoby O ON O.[ID osoby] = P.[ID pracownika]
WHERE [ID stanowiska] IN
(SELECT [ID stanowiska] FROM Stanowiska
WHERE Nazwa = 'Pracownik Obslugi Klienta');
```

Wyświetlenie aktualnego zapotrzebowania na pracowników.

```
GO
CREATE VIEW zapotrzebowanie_pracownikow AS
SELECT Z.[ID stanowiska] AS [ID Stanowiska], (Z.[Ilosc Potrzebnych
Pracownikow] - Z.[Ilosc Zatrudnionych Pracownikow]) AS [Ilosc
pracownikow do zatrudnienia],
S.Nazwa AS Stanowisko, S.Obowiazki AS Obowiazki, S.Kwalifikacje AS
[Wymagane Kwalifikacje]
FROM [Zapotrzebowanie Na Pracownikow] Z
       JOIN Stanowiska AS S ON S.[ID stanowiska] = Z.[ID stanowiska]
WHERE Z.[Ilosc Potrzebnych Pracownikow] > Z.[Ilosc Zatrudnionych
Pracownikow];
GO
```

Wyświetlenie hierarchii pracowników.

```
GO
CREATE VIEW hierarchia_pracownikow AS
SELECT O2.Imie AS  [Imie pracowinka], O2.Nazwisko AS  [Nazwisko
pracowinka], S2.Nazwa AS [Stanowisko pracowinka], S2.Obowiazki  AS
[Obowiazki pracowinka],
O1.Imie AS [Imie przelozonego], O1.Nazwisko AS  [Nazwisko
przelozonego], S1.Nazwa AS [Stanowisko przelozonego],
P2.[ID pracownika] AS [ID pracownika], P1.[ID pracownika] AS [ID
przelozonego]
FROM Pracownicy AS P1
 JOIN Pracownicy AS P2 ON P2.[ID przelozonego] = P1.[ID pracownika]
 JOIN Osoby AS O1 ON O1.[ID osoby] = P1.[ID pracownika]
 JOIN Osoby AS O2 ON O2.[ID osoby] = P2.[ID pracownika]
 JOIN Stanowiska AS S2 ON S2.[ID stanowiska] = P2.[ID stanowiska]
 JOIN Stanowiska AS S1 ON S1.[ID stanowiska] = P1.[ID stanowiska]
GO
```

FUNKCJE

Zaimplementowane zostało również 7 przykładowych funkcji realizujących podstawowe, typowe zapytania.

Funkcja wyświetlająca informacje o pozycjach z kategorii danej argumentem.

```
GO
CREATE FUNCTION wyswietl_wszystkie_pozycje_kategorii(@kategoria AS
NVARCHAR(50))

RETURNS TABLE AS
RETURN (

      SELECT P.Tytul, P.ISBN13, CONCAT( O.Imie, ' ', O.Nazwisko ) AS
[Autor], P.[Czesc serii], P.Jezyk, P.[Jezyk oryginalu], W.Nazwa AS
[Wydawca], P.[Data wydania],
                  P.Oprawa, P.Wymiary, P.[Liczba stron], (P.Cena * (1
- P.Obnizka)) AS Cena, P.Opis
      FROM [Kategorie ksiazek] K
      JOIN Produkty P ON K.[ID kategorii] = P.[ID kategorii]
      JOIN Autorzy A ON A.[ID Autora] = P.[ID autora]
      JOIN Osoby O ON O.[ID Osoby] = A.[ID Autora]
      JOIN Wydawcy W ON W.[ID wydawcy] = P.[ID wydawcy]
      WHERE K.[Nazwa] = @kategoria
)
GO

-- przykładowe wywołanie funkcji (wypisanie informacji o pozycjach z
kategorii 'fantastyka' ):
SELECT * FROM wyswietl_wszystkie_pozycje_kategorii('fantastyka')
```

Funkcja wyświetlająca szczegóły książek należących do serii danej argumentem.

```
GO
CREATE FUNCTION wyswietl_wszystkie_pozycje_serii(@seria AS
NVARCHAR(300))
RETURNS TABLE AS
RETURN (

      SELECT P.Tytul, P.ISBN13, CONCAT( O.Imie, ' ', O.Nazwisko ) AS
[Autor], P.[Czesc serii], P.Jezyk, P.[Jezyk oryginalu], W.Nazwa AS
[Wydawca], P.[Data wydania],
                  P.Oprawa, P.Wymiary, P.[Liczba stron],(P.Cena * (1
- P.Obnizka)) AS Cena, P.Opis
      FROM [Serie] S
      JOIN Produkty P ON S.[ID Serii] = P.[ID serii]
```

```
	JOIN Autorzy A ON A.[ID Autora] = P.[ID autora]
	JOIN Osoby O ON O.[ID Osoby] = A.[ID Autora]
	JOIN Wydawcy W ON W.[ID wydawcy] = P.[ID wydawcy]
	WHERE S.[Nazwa] = @seria

)
GO

-- przykładowe wywołanie funkcji (wypisanie informacji o pozycjach z
serii 'The Green Bone Saga' ):
SELECT * FROM wyswietl_wszystkie_pozycje_serii('The Green Bone Saga')
```

Funkcja wyświetlająca informacje o wszystkich zamówieniach klienta, którego ID jest dane argumentem.

```
GO
CREATE FUNCTION wyswietl_zamowienia_klienta(@ID AS INT)
RETURNS TABLE AS
RETURN (

	SELECT  SZ.[ID zamowienia], Z.[Data i czas zamowienia],
Z.[Status wysylki], Z.[Data wysylki],
		CONCAT( Z.[Kod pocztowy], ' ', Z.[Miasto], ' ',
Z.[Ulica], ' ',  Z.[Nr budynku], ' ', Z.[Nr lokalu]) AS Adres,
		OW.[Typ] AS [Metoda wysylki], ISNULL(PO.[Nazwa],'')
AS [Punkt odbioru],
		CONCAT( PO.[Kod pocztowy], ' ', PO.[Miasto], ' ',
PO.[Ulica], ' ',  PO.[Nr budynku]) AS [Adres Punktu Odbioru],
		CAST( ( SUM( ( SZ.Cena * (1 - Obnizka) ) ) + OW.Cena)
* ( 1- Z.[Rabat kliencki] ) ) AS MONEY ) AS Kwota
	FROM [Klienci] K
	JOIN Zamowienia Z ON K.[ID Klienta] = Z.[ID klienta]
	JOIN [Szczegoly Zamowien] SZ ON Z.[ID zamowienia] = SZ.[ID
zamowienia]
	LEFT JOIN [Punkty odbioru] PO ON PO.[ID punktu] = Z.[ID punktu
odbioru]
	LEFT JOIN [Opcje wysylki] OW ON OW.[ID opcji] = Z.[Metoda
wysylki]
	WHERE K.[ID Klienta] = @ID
	GROUP BY SZ.[ID zamowienia], Z.[Data i czas zamowienia],
Z.[Status wysylki], Z.[Data wysylki],
		Z.[Miasto], Z.[Ulica], Z.[Nr budynku], Z.[Nr
lokalu], Z.[Kod pocztowy], OW.Typ, PO.[Nazwa],
		PO.[Miasto], PO.[Ulica], PO.[Nr budynku], PO.[Kod
pocztowy], OW.Cena, Z.[Rabat kliencki]
)
GO

-- przykładowe wywołanie funkcji (wypisanie informacji o zamowieniach
klienta o ID = 4):
SELECT * FROM wyswietl_zamowienia_klienta(4)
```

Funkcja wyświetlająca dane książek napisanych przez danego argumentem autora.

```
GO
CREATE FUNCTION wyswietl_ksiazki_autora(@autor_nazwisko AS
NVARCHAR(50), @autor_imie AS NVARCHAR(50))
RETURNS TABLE AS
RETURN (
      SELECT P.Tytul, P.ISBN13, CONCAT( O.Imie, ' ', O.Nazwisko ) AS
[Autor], S.Nazwa AS [Nazwa serii] ,P.[Czesc serii], P.Jezyk, P.[Jezyk
oryginalu], W.Nazwa AS [Wydawca], P.[Data wydania],
                  P.Oprawa, P.Wymiary, P.[Liczba stron], (P.Cena * (1
- P.Obnizka)) AS Cena , P.Opis
      FROM Osoby O
      JOIN Autorzy A ON O.[ID osoby] = A.[ID autora]
      JOIN Produkty P ON O.[ID osoby] = P.[ID autora]
      JOIN Serie S ON O.[ID osoby] = S.[ID autora]
      JOIN Wydawcy W ON W.[ID wydawcy] = P.[ID wydawcy]
      WHERE O.Imie =  @autor_imie AND O.Nazwisko = @autor_nazwisko

)
GO

-- przykładowe wywołanie funkcji (wypisanie informacji o pozycjach
autorstwa Fondy Lee):
SELECT * FROM wyswietl_ksiazki_autora('Lee', 'Fonda')
```

Funkcja zwraca w formie tabeli dane o pracownikach, będących na danej zmianie w danym dniu

```
IF OBJECT_ID('pracownicy_na_zmianie', 'FN') IS NOT NULL DROP FUNCTION
pracownicy_na_zmianie

GO
CREATE FUNCTION pracownicy_na_zmianie( @ID_Zmiany INT, @Data DATE )
RETURNS TABLE AS RETURN (
      SELECT P.[ID Pracownika], O.Imie, O.Nazwisko FROM [Grafik
Zmian] AS G
      JOIN Pracownicy AS P ON P.[ID Pracownika] = G.[ID Pracownika]
      JOIN Osoby AS O ON P.[ID Pracownika] = O.[ID Osoby]
      WHERE @ID_Zmiany = G.[ID Zmiany] AND @Data = G.Data
)

GO
SELECT * FROM pracownicy_na_zmianie(1, '2022-01-13')
```

Funkcja zwraca wyliczoną sumę wszystkich wydatków danego klienta w ostatnim miesiącu (okres liczony od momentu wywołania funkcji).

```
IF OBJECT_ID('miesieczne_wydatki_klienta', 'FN') IS NOT NULL DROP
FUNCTION miesieczne_wydatki_klienta

GO
CREATE FUNCTION miesieczne_wydatki_klienta( @ID_Klienta INT ) RETURNS
MONEY AS
BEGIN
	DECLARE @Wydatki MONEY

	SET @Wydatki = ( SELECT SUM(ROUND(SZ.Ilosc * CAST((SZ.Cena *
SZ.obnizka) AS MONEY ), 2)) FROM Zamowienia AS Z
				JOIN [Szczegoly Zamowien] AS SZ ON
Z.[ID zamowienia] = SZ.[ID zamowienia]
				WHERE @ID_Klienta = Z.[ID klienta] AND
DATEDIFF( month, GETDATE(), Z.[Data i czas zamowienia]) <= 1
				GROUP BY Z.[ID klienta] )
	RETURN @Wydatki
END

SELECT dbo.miesieczne_wydatki_klienta( 4 )
```

Funkcja zwraca w formie tabeli dane o pracownikach będących w którymś momencie zatrudnionych na danym stanowisku

```
IF OBJECT_ID('historia_zatrudnien_na_stanowisku', 'FN') IS NOT NULL
DROP FUNCTION historia_zatrudnien_na_stanowisku

GO
CREATE FUNCTION historia_zatrudnien_na_stanowisku( @ID_Stanowiska INT
) RETURNS TABLE AS RETURN (
	SELECT P.[ID Pracownika], O.Imie, O.Nazwisko,
	HZ.[Data zatrudnienia na stanowisku] AS [Data zatrudnienia],
HZ.[Data zwolnienia] FROM [Historia Zatrudnien] AS HZ
	JOIN Pracownicy AS P ON P.[ID Pracownika] = HZ.[ID Pracownika]
	JOIN Osoby AS O ON P.[ID Pracownika] = O.[ID Osoby]
	WHERE @ID_Stanowiska = HZ.[Stanowisko]
)

GO
SELECT dbo.historia_zatrudnien_na_stanowisku( 1 )
```

PROCEDURY SKŁADOWANE

W ramach projektu zaimplementowane zostało 9 procedur składowanych obsługujących podstawowe funkcjonalności bazy danych, jak również uwzględniających zabezpieczenia przed podstawowymi błędami.

Procedura **zmien _pensje -** zmienia pensję pracownikowi o ID danym argumentem na nową, ponadto przed zmianą dodaje obecną pensję do zarchiwizowanych (Historii Pensji).

```
GO
CREATE PROCEDURE zmien_pensje ( @ID INT = NULL, @Nowa_pensja MONEY =
NULL )
AS

DECLARE @blad AS NVARCHAR(500);

IF @ID IS NULL OR @Nowa_pensja IS NULL OR @ID NOT IN (SELECT [ID
pracownika] FROM Pracownicy)
  BEGIN
    SET @blad = 'Nieprawidłowe dane, sprawdź podane argumenty.';
    RAISERROR(@blad, 16,1);
      RETURN;
  END

IF @ID NOT IN (SELECT [ID pracownika] FROM Pracownicy)
  BEGIN
    SET @blad = 'Nie ma takiego pracownika w bazie danych.';
    RAISERROR(@blad, 16,1);
      RETURN;
  END


DECLARE @Stara_pensja AS MONEY;
SET @Stara_pensja = (SELECT Pensja FROM Pracownicy
                            WHERE [ID pracownika] = @ID)

INSERT INTO [Historia Pensji]([ID pracownika], [Data zmiany pensji],
[Poprzednia pensja])
VALUES (@ID, GETDATE(), @Stara_pensja);

UPDATE Pracownicy
SET Pensja = @Nowa_pensja
WHERE [ID pracownika] = @ID;
GO

--przykładowe wywołanie procedury - zmiana pensji pracownikowi o
ID = 9 na 15 000
EXEC zmien_pensje @ID = 9, @Nowa_pensja = 15000.00
```

Procedura **zmien _status_zamowienia -** zmienia status zamówienia o ID danym argumentem na następny w kolejności. W momencie, gdy status zmieni się na 'wysłano' aktualizowana jest data wysyłki w tabeli Zamowienia. Ponadto, gdy status zostanie już raz ustawiony jako wysłany, brak możliwości jego zmiany.

```
GO
CREATE PROCEDURE zmien_status_zamowienia ( @ID INT = NULL )
AS

DECLARE @blad AS NVARCHAR(500);

IF @ID IS NULL OR @ID NOT IN ( SELECT [ID zamowienia] FROM Zamowienia)
   BEGIN
    SET @blad = 'Nieprawidłowe dane, sprawdź podane argumenty.';
    RAISERROR(@blad, 16,1);
      RETURN;
   END

IF @ID NOT IN ( SELECT [ID zamowienia] FROM Zamowienia)
   BEGIN
    SET @blad = 'Nie ma takiego zamówienia w bazie danych.';
    RAISERROR(@blad, 16,1);
      RETURN;
   END

DECLARE @temp AS INT, @MAX AS INT;

SET @temp = (SELECT SZ.[ID Statusu] FROM Zamowienia Z
                JOIN [Statusy zamowienia] SZ ON SZ.[ID Statusu] =
Z.[Status wysylki]
                WHERE [ID zamowienia] = @ID);
SET @max = (SELECT [ID Statusu] FROM [Statusy zamowienia]
               WHERE [Nazwa statusu] = 'wyslano');

IF @temp IS NULL
   BEGIN
    UPDATE Zamowienia
   SET [Status wysylki] = 1
   WHERE [ID zamowienia] = @ID;
      RETURN;
   END

IF @temp = @max
   BEGIN
    SET @blad = 'Nie można zmienić statusu, zamówienie zostało już
wysłane.';
    RAISERROR(@blad, 16,1);
      RETURN;
   END
```

```
IF @temp + 1 = @max
   UPDATE Zamowienia
   SET [Data wysylki] = GETDATE()
   WHERE [ID zamowienia] = @ID;

UPDATE Zamowienia
SET [Status wysylki] = @temp + 1
WHERE [ID zamowienia] = @ID;
GO

--przykładowe wywołanie procedury - trzykrotne zaktualizowanie statusu
dla zamówienia o ID = 3
EXEC zmien_status_zamowienia @ID = 3
EXEC zmien_status_zamowienia @ID = 3
EXEC zmien_status_zamowienia @ID = 3
```

Procedura **dodaj_klienta -** dodaje nowego klienta tworząc nowe konto w bazie wraz ze wszystkimi danymi osobowymi, hasłem oraz loginem. Zakładamy, że hasło powinno zawierać minimum 8 znaków, każdy login unikalny, a na jeden adres e-mail można założyć tylko jedno konto.

```
GO
CREATE PROC dodaj_klienta( @Imie NVARCHAR(50) = NULL, @Nazwisko
NVARCHAR(50) = NULL, @Miasto NVARCHAR(50) = NULL, @Ulica NVARCHAR(50) =
NULL, @Budynek INT = NULL, @Lokal INT = NULL, @Kod NVARCHAR(10) = NULL,
@Login NVARCHAR(50), @Haslo VARCHAR(50), @Tel INT = NULL,
@Mail NVARCHAR(255) = NULL )
AS

DECLARE @blad AS NVARCHAR(500);

IF @Imie IS NULL OR @Nazwisko IS NULL OR @Login IS NULL OR @Haslo IS
NULL OR @Tel IS NULL
   BEGIN
         SET @blad = 'Błędne dane, sprawdź podane argumenty.';
         RAISERROR(@blad, 16,1);
         RETURN;
   END

IF LEN(@Haslo) < 8
   BEGIN
   SET @blad = 'Hasło powinno zawierać minimum 8 znaków';
         RAISERROR(@blad, 16,1);
         RETURN;
   END

IF @Login IN (SELECT [Nazwa konta] FROM Klienci)
   BEGIN
         SET @blad = 'Błędny login. Konto o podanej nazwie już
istnieje.';
```

```
        RAISERROR(@blad, 16,1);
        RETURN;
    END

IF @Mail IN (SELECT [E-mail] FROM Klienci)
    BEGIN
        SET @blad = 'Błędny e-mial. Konto na podany adres e-mail
zostało już utworzone.';
        RAISERROR(@blad, 16,1);
        RETURN;
    END

INSERT INTO Osoby(Imie, Nazwisko, Miasto, Ulica, [Nr budynku], [Nr
lokalu], [Kod pocztowy])
VALUES (@Imie, @Nazwisko, @Miasto, @Ulica, @Budynek, @Lokal, @Kod);

DECLARE @ID INT;
SET @ID= ( SELECT MAX([ID osoby]) FROM Osoby )

INSERT INTO Klienci([ID klienta], [ID kategorii], [Nazwa konta], [Haslo
do konta], Telefon, [E-mail] )
VALUES(@ID, 1, @Login, @Haslo, @Tel, @Mail );
GO

--przykładowe wywołanie procedury - dodanie klienta: Jan Kowalski wraz z
danymi, loginem i hasłem
EXEC dodaj_klienta  @Imie = 'Jan', @Nazwisko = 'Kowalski', @Login =
'jkowal', @Miasto = 'Kraków', @Ulica = 'Focha', @Budynek = '123', @Kod =
'30-111',
            @Haslo = '12asasdfgg', @Tel = 123123123, @Mail = NULL
```

Procedura **dodaj_ksiazke -** dodaje nowy produkt – książkę. Procedura nie może dodać nowej kategorii ani nowego wydawcy ani nowego autora ani nowej serii. Przy dodawaniu nowej części istniejącej już serii, procedura aktualizuję ilość książek w serii.

```
GO
CREATE PROCEDURE dodaj_ksiazke( @ISBN13 BIGINT = NULL, @tytul
NVARCHAR(50) = NULL, @id_kategorii INT = NULL, @id_autora INT = NULL,
@id_serii INT = NULL,
@czesc_serii INT = NULL, @jezyk NVARCHAR(50) = NULL, @jezyk_og
NVARCHAR(50) = NULL, @id_wydawca INT = NULL, @data_wydania DATE = NULL,
@oprawa NVARCHAR(50) = NULL,
@wymiary NVARCHAR(50) = NULL, @liczba_stron INT = NULL, @wmagazynie INT
= NULL, @cena SMALLMONEY = NULL, @obnizka REAL = NULL,  @opis
NVARCHAR(1000) = NULL)
AS

DECLARE @blad AS NVARCHAR(500);

IF @ISBN13 IS NULL OR @tytul IS NULL OR @id_autora IS NULL OR
@id_kategorii IS NULL OR @jezyk IS NULL OR  @id_wydawca IS NULL
```

```
   OR @data_wydania IS NULL OR @oprawa IS NULL OR @wymiary IS NULL OR
   @liczba_stron IS NULL OR @wmagazynie IS NULL OR @cena IS NULL OR
   @obnizka IS NULL
      BEGIN
            SET @blad = 'Błędne dane, sprawdź podane argumenty.';
            RAISERROR(@blad, 16,1);
            RETURN;
      END

   IF @ISBN13 IN (SELECT ISBN13 FROM Produkty)
      BEGIN
            SET @blad = 'Błędny ISBN13. Ksiazka o podanym ISBN13 jest juz w
   bazie.';
            RAISERROR(@blad, 16,1);
            RETURN;
      END

   INSERT INTO Produkty(ISBN13, [ID kategorii],  Tytul, [ID autora], [ID
   serii], [Czesc serii],
   Jezyk, [Jezyk oryginalu],  [ID wydawcy], [Data wydania], [Oprawa],
   Wymiary, [Liczba stron],
   [Ilosc w magazynie], Cena, Obnizka, Opis)

   VALUES (@ISBN13, @id_kategorii,  @tytul, @id_autora, @id_serii,
   @czesc_serii, @jezyk, @jezyk_og, @id_wydawca,
   @data_wydania, @oprawa, @wymiary, @liczba_stron, @wmagazynie, @cena,
   @obnizka, @opis)

   IF @id_serii IS NOT NULL
      BEGIN
            DECLARE @Stara_ilosc AS INT;
            SET @Stara_ilosc = (SELECT [Ilosc Czesci Wydanych] FROM Serie
   WHERE [ID Serii] = @id_serii)
            UPDATE Serie
            SET [Ilosc Czesci Wydanych] = @Stara_ilosc + 1
            WHERE [ID Serii] = @id_serii
      END
   GO

-- przykładowe wywołanie procedury – dodanie książki
EXEC dodaj_ksiazke @ISBN13 = 9781931363348, @id_kategorii = 1, @tytul =
'tgbn4', @id_autora = 1, @id_serii = 1, @czesc_serii = 4, @jezyk =
'angielski', @jezyk_og = 'angielski', @id_wydawca = 1,
@data_wydania = '2022-01-13', @oprawa = 'miekka', @wymiary = '10 x 20',
@liczba_stron = 674, @wmagazynie =43, @cena = 100, @obnizka = 0.02, @opis =
'dfxbgvd'
GO
```

Procedura dodająca nowego pracownika do tabeli "Pracownicy". Rekord składany na podstawie podanych danych

```
IF OBJECT_ID('dbo.dodaj_pracownika','P') IS NOT NULL DROP PROCEDURE
dbo.dodaj_pracownika

GO
CREATE PROCEDURE dbo.dodaj_pracownika( @Imie NVARCHAR(50) = NULL,
@Nazwisko NVARCHAR(50) = NULL, @Miasto NVARCHAR(50) = NULL, @Ulica
NVARCHAR(50) = NULL, @Budynek INT = NULL, @Lokal INT = NULL,
@Kod_Pocztowy NVARCHAR(10) = NULL, @ID_Przelozonego INT = NULL,
@ID_Stanowiska INT = NULL, @Pensja MONEY = NULL ) AS
BEGIN
	DECLARE @blad AS NVARCHAR(100);

	IF @Imie IS NULL OR @Nazwisko IS NULL OR @ID_Stanowiska IS NULL
OR @Pensja IS NULL
	BEGIN
		SET @blad = 'Pominieto kluczowe dane. Dodanie pracownika
nie powiodlo sie';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	IF @ID_Przelozonego NOT IN ( SELECT [ID Pracownika] FROM
Pracownicy ) AND @ID_Przelozonego IS NOT NULL
	BEGIN
		SET @blad = 'Podano bledne ID przelozonego. Dodanie
pracownika nie powiodlo sie';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	IF @ID_Stanowiska NOT IN ( SELECT [ID Stanowiska] FROM
Stanowiska )
	BEGIN
		SET @blad = 'Podano bledne ID stanowiska. Dodanie
pracownika nie powiodlo sie';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	IF ( SELECT [Ilosc Potrzebnych Pracownikow] FROM
[Zapotrzebowanie Na Pracownikow] --trzeba przetestowac
		WHERE [ID Stanowiska] = @ID_Stanowiska ) = 0
	BEGIN
		SET @blad = 'Brak zapotrzebowania pracownikow na podane
stanowisko. Dodanie pracownika nie powiodlo sie';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END
```

```
	UPDATE [Zapotrzebowanie Na Pracownikow]
	SET [Ilosc Potrzebnych Pracownikow] = ( [Ilosc Potrzebnych
Pracownikow] - 1 ),
	[Ilosc Zatrudnionych Pracownikow] = ( [Ilosc Zatrudnionych
Pracownikow] + 1 )
	WHERE [ID Stanowiska] = @ID_Stanowiska

	INSERT INTO Osoby(Imie, Nazwisko, Miasto, Ulica, [Nr budynku],
[Nr lokalu], [Kod pocztowy])
	VALUES (@Imie, @Nazwisko, @Miasto, @Ulica, @Budynek, @Lokal,
@Kod_Pocztowy);

	DECLARE @ID INT;
	SET @ID = ( SELECT MAX( [ID osoby] ) FROM Osoby )

	INSERT INTO Pracownicy( [ID Pracownika], [ID przelozonego], [ID
stanowiska], Pensja )
	VALUES (@ID, @ID_Przelozonego, @ID_Stanowiska, @Pensja)
END

GO

EXECUTE dodaj_pracownika @Imie = 'Szymon', @Nazwisko = 'Majewski',
@Miasto = 'Kraków', @Ulica = 'Focha', @Budynek = '123', @Kod_Pocztowy
= '30-111', @ID_Przelozonego = NULL, @ID_Stanowiska = 1, @Pensja =
20000.00
```

Dodanie urlopu danej kategorii danemu pracownikowi w podanym okresie

```
IF OBJECT_ID('dbo.dodaj_urlop','P') IS NOT NULL DROP PROCEDURE
dbo.dodaj_urlop

GO
CREATE PROCEDURE dbo.dodaj_urlop( @ID_Pracownika INT, @ID_Kategorii
INT, @Data_Od DATE, @Data_Do DATE ) AS
BEGIN
	DECLARE @blad AS NVARCHAR(100);

	IF @Data_Od >= @Data_Do
	BEGIN
		SET @blad = 'Podano niepoprawne daty. Urlop nie zostal
dodany';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END
```

```
	IF EXISTS ( SELECT [ID pracownika] FROM [Grafik Zmian] WHERE
([Data] BETWEEN @Data_Od AND @Data_Do) AND @ID_Pracownika = [ID
pracownika] )
	BEGIN
		SET @blad = 'Pracownik w podanym okresie ma przypisane
zmiany. Urlop nie zostal dodany';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	IF EXISTS ( SELECT [ID pracownika] FROM Urlopy WHERE
@ID_Pracownika = [ID pracownika] AND
				( ( @Data_Od <= [Data do] AND [Data od] <=
@Data_Od ) OR
				  ( @Data_Do <= [Data do] AND [Data od] <=
@Data_Do ) )
			)
	BEGIN
		SET @blad = 'Pracownik ma juz urlop w podanym terminie.
Urlop nie zostal dodany';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	DECLARE @Ilosc_Dni INT
	SET @Ilosc_Dni = DATEDIFF( day, @Data_Od, @Data_Do )

	IF @Ilosc_Dni > ( SELECT [Dopuszczalna Ilosc Dni] FROM [Urlopy
Kategorie] WHERE [ID Kategorii] = @ID_Kategorii )
	BEGIN
		SET @blad = 'Przekroczono maksymalna ilosc dni w tej
kategorii urlopu. Urlop nie zostal dodany';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	INSERT INTO Urlopy ( [ID pracownika], [ID kategorii], [Data
od], [Data do] )
	VALUES ( @ID_Pracownika, @ID_Kategorii, @Data_Od, @Data_Do )

END

SELECT * FROM Urlopy
EXECUTE dbo.dodaj_urlop @ID_Pracownika = 10, @ID_Kategorii = 1,
@Data_Od = '2022-01-17', @Data_Do = '2022-01-22'
SELECT * FROM Urlopy
EXECUTE dbo.dodaj_urlop @ID_Pracownika = 10, @ID_Kategorii = 1,
@Data_Od = '2022-01-19', @Data_Do = '2022-01-20'
EXECUTE dbo.dodaj_urlop @ID_Pracownika = 10, @ID_Kategorii = 1,
@Data_Od = '2022-02-19', @Data_Do = '2024-01-20'
EXECUTE dbo.dodaj_urlop @ID_Pracownika = 10, @ID_Kategorii = 1,
@Data_Od = '2022-02-13', @Data_Do = '2022-01-14'
EXECUTE dbo.dodaj_urlop @ID_Pracownika = 10, @ID_Kategorii = 1,
@Data_Od = '2022-01-13', @Data_Do = '2022-01-14'
```

Procedura dodająca do tabel "Zamowienia" oraz "Szczegoly Zamowien" po rekordzie zgodnym z podanymi danymi - zawartość koszyka, adres i sposób wysyłki

```
IF TYPE_ID('Pozycje_Zamowienia_TYP') IS NOT NULL DROP TYPE
Pozycje_Zamowienia_TYP

CREATE TYPE Pozycje_Zamowienia_TYP AS TABLE (
	ID_Produktu INT,
	Ilosc INT
)

IF OBJECT_ID('dbo.zloz_zamowienie','P') IS NOT NULL DROP PROCEDURE
dbo.zloz_zamowienie

GO
CREATE PROCEDURE dbo.zloz_zamowienie ( @pozycje_zamowienia
Pozycje_Zamowienia_TYP READONLY, @ID_Klienta INT,
@Miasto NVARCHAR(50) = NULL, @Ulica NVARCHAR(50) = NULL, @Nr_Budynku
INT = NULL, @Nr_Lokalu INT = NULL, @Kod_Pocztowy NVARCHAR(10) = NULL,
@Metoda_Wysylki INT = NULL, @ID_Punktu_Odbioru INT = NULL ) AS
BEGIN
	DECLARE @blad AS NVARCHAR(100);

	-- zamowienie puste
	IF NOT EXISTS ( SELECT ID_Produktu FROM @pozycje_zamowienia )
	BEGIN
		SET @blad = 'Koszyk pusty. Zamowienie nie zostalo
zlozone';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	-- brak danych do wysylki (albo punktu odbioru, albo adresu)
	IF @ID_Punktu_Odbioru IS NULL AND ( @Miasto IS NULL OR @Ulica
IS NULL OR @Nr_Budynku IS NULL OR @Kod_Pocztowy IS NULL )
	BEGIN
		SET @blad = 'Niepelne dane do wysylki. Zamowienie nie
zostalo zlozone';
		RAISERROR(@blad, 16, 1);
		RETURN;
	END

	DECLARE @ID_Prod INT
	DECLARE @Liczba INT

	DECLARE wiersz_zamowienia CURSOR FOR
	SELECT ID_Produktu, Ilosc FROM @pozycje_zamowienia

	OPEN wiersz_zamowienia

	FETCH NEXT FROM wiersz_zamowienia INTO @ID_Prod, @Liczba
```

```
	--brak towaru na stanie
	WHILE ( @@FETCH_STATUS = 0 )
	BEGIN
		IF @Liczba > ( SELECT [Ilosc w magazynie] FROM Produkty
WHERE @ID_Prod = [ID Produktu] )
		BEGIN
			SET @blad = 'Brak towaru na stanie. Zamowienie nie
zostalo zlozone';
			RAISERROR(@blad, 16, 1);
			CLOSE wiersz_zamowienia
			DEALLOCATE wiersz_zamowienia
			RETURN;
		END

		FETCH NEXT FROM wiersz_zamowienia INTO @ID_Prod, @Liczba
	END

	CLOSE wiersz_zamowienia
	DEALLOCATE wiersz_zamowienia

	--odjecie towaru ze stanu
	DECLARE wiersz_zamowienia2 CURSOR FOR
	SELECT ID_Produktu, Ilosc FROM @pozycje_zamowienia

	OPEN wiersz_zamowienia2

	FETCH NEXT FROM wiersz_zamowienia2 INTO @ID_Prod, @Liczba

	WHILE ( @@FETCH_STATUS = 0 )
	BEGIN
		UPDATE Produkty
		SET [Ilosc w magazynie] = [Ilosc w magazynie] - @Liczba
		WHERE @ID_Prod = [ID Produktu]

		FETCH NEXT FROM wiersz_zamowienia2 INTO @ID_Prod, @Liczba
	END

	CLOSE wiersz_zamowienia2
	DEALLOCATE wiersz_zamowienia2

	--nadanie klientowi kategorii
	EXECUTE dbo.przypisz_klientowi_kategorie @ID_Klienta =
@ID_Klienta
	DECLARE @ID_Kategorii_Klienta INT
	DECLARE @Rabat_Klienta FLOAT
	SET @ID_Kategorii_Klienta = ( SELECT [ID Kategorii] FROM
Klienci WHERE @ID_Klienta = [ID Klienta] )
	SET @Rabat_Klienta = ( SELECT Rabat FROM [Klienci Kategorie]
WHERE [ID Kategorii] = @ID_Kategorii_Klienta )

	--dodanie zamowienia
	INSERT INTO Zamowienia ( [ID klienta], [Rabat kliencki], [Data
i czas zamowienia], [Status wysylki], [Data wysylki], Miasto,Ulica,
```

```
[Nr budynku], [Nr lokalu], [Kod pocztowy], [Metoda Wysylki], [ID
Punktu Odbioru] )
      VALUES ( @ID_Klienta, @Rabat_Klienta, GETDATE(), 1, NULL,
@Miasto, @Ulica, @Nr_Budynku, @Nr_Lokalu, @Kod_Pocztowy,
@Metoda_Wysylki, @ID_Punktu_Odbioru )

      DECLARE @ID INT;
      SET @ID = ( SELECT MAX( [ID Zamowienia] ) FROM Zamowienia )

      --dodanie szczegolow zamowienia
      DECLARE wiersz_zamowienia3 CURSOR FOR
      SELECT ID_Produktu, Ilosc FROM @pozycje_zamowienia

      OPEN wiersz_zamowienia3

      FETCH NEXT FROM wiersz_zamowienia3 INTO @ID_Prod, @Liczba

      WHILE ( @@FETCH_STATUS = 0 )
      BEGIN
            INSERT INTO [Szczegoly Zamowien] ( [ID Zamowienia], [ID
produktu], Cena, Ilosc, Obnizka )
            VALUES ( @ID, @ID_Prod, ( SELECT Cena FROM Produkty WHERE
[ID Produktu] = @ID_Prod ), @Liczba,
            ( SELECT Obnizka FROM Produkty WHERE [ID Produktu] =
@ID_Prod ) )

            FETCH NEXT FROM wiersz_zamowienia3 INTO @ID_Prod, @Liczba
      END

      CLOSE wiersz_zamowienia3
      DEALLOCATE wiersz_zamowienia3

END

--przyklad
SELECT * FROM Zamowienia
JOIN [Szczegoly zamowien] AS S ON Zamowienia.[ID Zamowienia] = S.[ID
Zamowienia]
WHERE S.[ID Zamowienia] = ( SELECT MAX( [ID Zamowienia] ) FROM
Zamowienia )

DECLARE @Koszyk Pozycje_Zamowienia_TYP
INSERT INTO @Koszyk VALUES
( 1, 2 ),
( 2, 1 ),
( 3, 1 )

EXECUTE dbo.zloz_zamowienie @pozycje_zamowienia = @Koszyk,
@ID_Klienta = 4, @Metoda_Wysylki = 1, @ID_Punktu_Odbioru = 1

SELECT * FROM Zamowienia
JOIN [Szczegoly zamowien] AS S ON Zamowienia.[ID Zamowienia] = S.[ID
Zamowienia]
```

```
WHERE S.[ID Zamowienia] = ( SELECT MAX( [ID Zamowienia] ) FROM
Zamowienia )
```

Procedura przypisująca danemu klientowi kategorię na podstawie sumarycznych wydatków w ostatnim miesiącu

```
IF OBJECT_ID('dbo.przypisz_klientowi_kategorie','P') IS NOT NULL DROP
PROCEDURE dbo.przypisz_klientowi_kategorie

GO
CREATE PROCEDURE dbo.przypisz_klientowi_kategorie ( @ID_Klienta INT )
AS
BEGIN
	DECLARE @wydatki_w_ostatnim_miesiacu INT
	SET @wydatki_w_ostatnim_miesiacu = ( SELECT
dbo.miesieczne_wydatki_klienta( @ID_Klienta ) )

	DECLARE @Ilosc_Kategorii INT
	SET @Ilosc_Kategorii = ( SELECT COUNT( [ID kategorii] ) FROM
[Klienci Kategorie] )

	DECLARE @i INT
	SET @i = 1

	WHILE (@i <= @Ilosc_Kategorii)
	BEGIN
		IF @wydatki_w_ostatnim_miesiacu < ( SELECT [Maksymalne
Miesieczne Zakupy ] FROM [Klienci Kategorie]

	WHERE [ID Kategorii] = @i )
		BEGIN
			UPDATE Klienci
			SET [ID Kategorii] = @i
			WHERE [ID Klienta] = @ID_Klienta

			RETURN
		END

		SET @i = @i + 1
	END

	UPDATE Klienci
	SET [ID Kategorii] = ( @i - 1 )
	WHERE [ID Klienta] = @ID_Klienta

	RETURN
END

GO
EXECUTE dbo.przypisz_klientowi_kategorie @ID_Klienta = 4
```

WYZWALACZE

Baza została zaopatrzona w 5 wyzwalaczy, skupiających się głównie na wyświetlaniu informacji na temat akcji wykonanych przez użytkownika w systemie (tj. złożenie zamówienia, zmiana stanu konta itp.).

Po pomyślnym złożeniu zamówienia, zostaje wyświetlony stosowny komunikat.

```
GO
CREATE TRIGGER zlozono_zamowienie ON Zamowienia
AFTER INSERT
AS
	DECLARE @Text VARCHAR(256)
	SET @Text = FORMATMESSAGE('ZAMOWIENIE ZOSTALO ZLOZONE')
	PRINT @Text
GO
```

Po pomyślnym zatrudnieniu pracownika, zostaje wyświetlony stosowny komunikat.

```
CREATE TRIGGER zatrudniono_pracownika ON Pracownicy
AFTER INSERT
AS
	DECLARE @Text VARCHAR(256)
	SET @Text = FORMATMESSAGE('Pracownik zostal dodany do bazy
pracownikow')
	PRINT @Text
GO
```

Po pomyślnym założeniu konta zostaje wyświetlony stosowny komunikat.

```
CREATE TRIGGER dodano_klienta ON Klienci
AFTER INSERT
AS
	DECLARE @Text VARCHAR(256)
	SET @Text = FORMATMESSAGE('konto zostalo zalozone pomyslnie')
	PRINT @Text
GO
```

Po pomyślnym usunięciu konta, zostaje wyświetlony stosowny komunikat.

```
CREATE TRIGGER usuniento_konto ON Klienci
AFTER DELETE
AS
	DECLARE @Text VARCHAR(256)
```

```
	SET @Text = FORMATMESSAGE('Konto zostalo usuniete pomyslnie')
	PRINT @Text
GO
```

Po zmianie statusu zamówienia zostaje wyświetlony stosowny komunikat

```
IF OBJECT_ID('zmieniono_status_zamowienia', 'TR') IS NOT NULL
	DROP TRIGGER zmieniono_status_zamowienia
GO

CREATE TRIGGER zmieniono_status_zamowienia ON Zamowienia
AFTER UPDATE
AS
	DECLARE @Text VARCHAR(256)

	IF UPDATE( [Status wysylki] )
	BEGIN
		IF ( SELECT [Status wysylki] FROM Inserted ) = 1
		BEGIN
			SET @Text = FORMATMESSAGE('Zamowienie przyjete do
realizacji')
		END
		IF ( SELECT [Status wysylki] FROM Inserted ) = 2
		BEGIN
			SET @Text = FORMATMESSAGE('Zamowienie zostalo
oplacone')
		END
		IF ( SELECT [Status wysylki] FROM Inserted ) = 3
		BEGIN
			SET @Text = FORMATMESSAGE('Zamowienie oczekuje na
wysylke')
		END
		IF ( SELECT [Status wysylki] FROM Inserted ) = 4
		BEGIN
			SET @Text = FORMATMESSAGE('Zamowienie zostalo
wyslane')
		END
	END

	PRINT @Text
GO

SELECT * FROM Zamowienia

UPDATE Zamowienia
SET [Status wysylki] = [Status wysylki] + 1
WHERE [ID Zamowienia] = 5
```