

Hurtownia WYROBÓW CHEMICZNYCH



Uczelnia: Uniwersytet Łódzki

Wydział: Wydział ekonomiczno-socjologiczny

Kierunek: Informatyka Ekonomiczna

Rok studiów: II Rok

Semestr: 4 Semestr

Rok akademicki: 2023/2024

Skład grupy:

RODO

Spis treści

Opis tła projektowanego i implementowanego zagadnienia:	3
Opis wymagań funkcjonalnych i нефункциональных:	4
Wymagania funkcjonalne:.....	4
Wymagania нефункциональные:.....	5
MODEL FIZYCZNY	6
Skrypt tworzenia bazy danych.....	7
Wykaz utworzonych tabel fizycznych.....	8
Skrypt tworzenia tabel i referencji między nimi.....	14
Wykaz zaimplementowanych indeksów	17
Skrypt tworzenia indeksów	18
Skrypt ładujący przykładowe dane do tabel.....	19
Wykaz utworzonych widoków.....	24
Skrypt tworzenia widoków	24
Wykaz utworzonych procedur składowanych.....	27
Skrypt tworzenia procedur składowanych	27
Wykaz utworzonych funkcji użytkownika	43
Skrypt tworzenia funkcji definiowanych przez użytkownika.....	43
Wykaz utworzonych wyzwalaczy	50
Skrypt tworzenia wyzwalaczy.....	50
Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia	57
Schemat blokowy	57
Warstwa wejścia/wyjścia	58
Wnioski i podsumowanie:	61
Wykaz/spis ilustracji.....	62
Wykaz/spis tabel.....	63

Opis tła projektowanego i implementowanego zagadnienia:

Baza danych "Hurtownia wyrobów chemicznych" została stworzona w celu efektywnego zarządzania hurtownią, która specjalizuje się w sprzedaży różnego rodzaju wyrobów chemicznych. W ramach projektu powstała baza danych, która pozwala na składowanie i przetwarzanie informacji związanych z pracownikami, kontrahentami, dostawcami, produktami, sprzedażą oraz zakupami.

Opis wymagań funkcjonalnych i niefunkcjonalnych:

Wymagania funkcjonalne:

System powinien umożliwiać:

- Automatycznie odejmować liczbę sprzedanych sztuk produktów z magazynu po dodaniu nowej sprzedaży do bazy danych
- Automatycznie aktualizować stan magazynowy produktów po dodaniu nowego zakupu poprzez zwiększenie liczby sztuk w magazynie o wartość wprowadzonej w szczegółach zakupu.
- Sprawdzenie liczby dostawców zarejestrowanych w danym województwie na podstawie jego identyfikatora.
- Pobranie informacji o sprzedaży dla danego klienta.
- Obliczenie łącznej sprzedaży danego produktu
- Liczenie liczby pracowników na podstawie ich stanowiska.
- Pobranie rodzaju płatności dla określonej płatności na podstawie jej identyfikatora
- Sprawdzenie typu klienta na podstawie jego ID.
- Pobranie nazwy dostawcy na podstawie jego identyfikatora.
- Pobranie adresu kontrahenta na podstawie jego identyfikatora
- Pobranie nazwy produktu na podstawie jego identyfikatora
- Pobranie szczegółowych informacji dotyczących sprzedaży produktu na podstawie jego identyfikatora
- Wyświetlenie sprzedaży z określonego przedziału czasowego
- Wyświetlenie listy produktów z magazynu według wybranej kategorii.
- Dodawanie nowego zakupu, nowej sprzedaży do bazy danych przy użyciu procedury

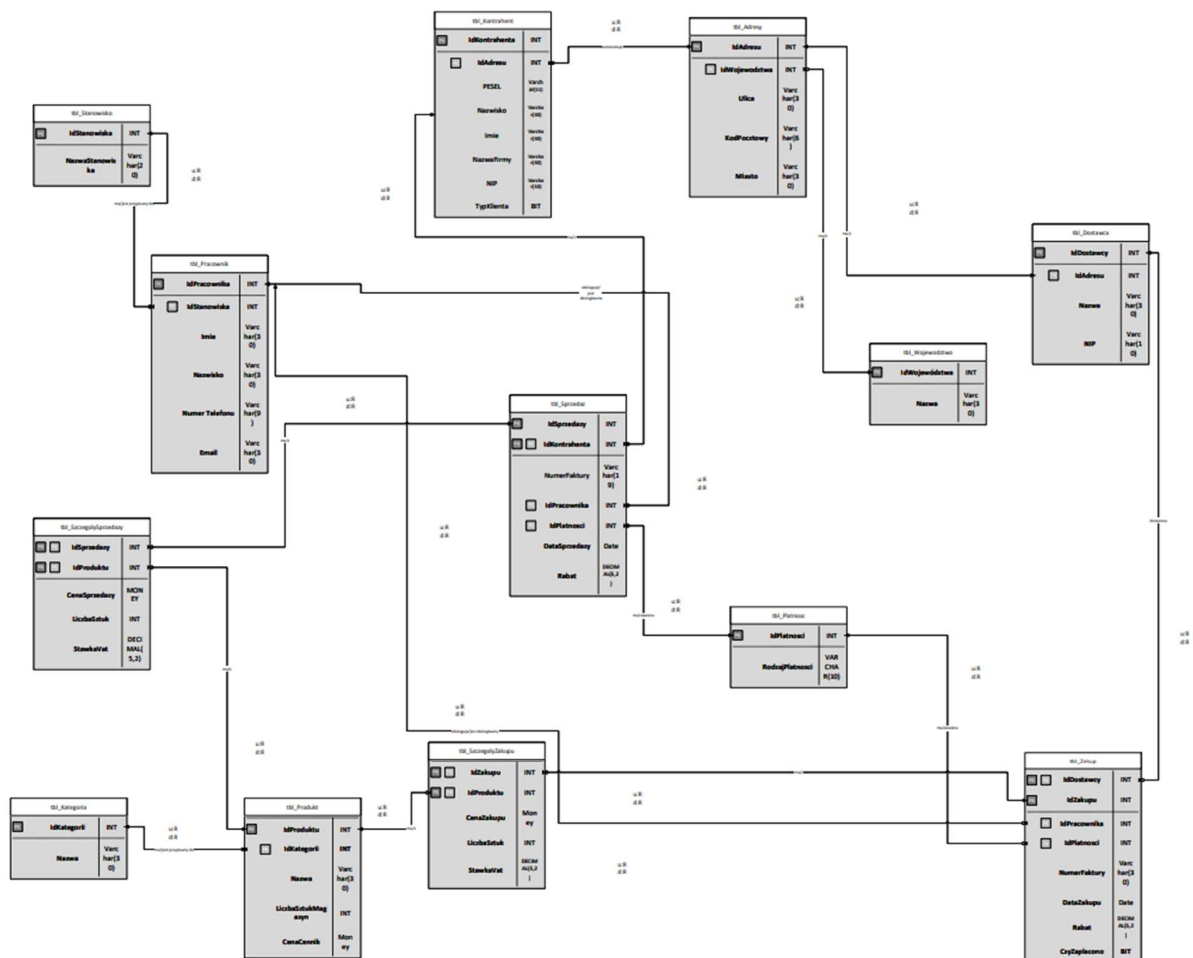
- Dodawanie, usuwanie oraz aktualizowanie informacji o pracownikach przy użyciu procedury.
- Dodawanie, usuwanie i edytowanie produktów przy użyciu procedury.
- Wyświetlenie informacji na temat produktów chemicznych dostępnych w magazynie, wraz ze stanem magazynowym, ceną, opisem i dostępnością.
- Wyświetlenie kwoty sprzedaży produktów zgrupowanej według kategorii oraz nazwy produktu
- Uzyskanie informacji o ilości produktów w magazynie według kategorii produktu.
- Wyświetlenie średniego rabatu dla wszystkich sprzedaży
- Wyświetlenie raportu zawierającego imiona i nazwiska pracowników oraz ilość dokonanych przez nich sprzedaży
- Wyświetlenie liczby produktów w każdej kategorii produktów.
- Wyświetlenie listy produktów, których liczba sztuk na magazynie jest mniejsza niż 10
- Obniżenie ceny wszystkich produktów o określony procent
- Podwyższenie ceny wszystkich produktów o określony procent

Wymagania niefunkcjonalne:

- Wysoka wydajność bazy danych, umożliwiająca szybkie przetwarzanie dużej ilości danych.
- Bezpieczeństwo i poufność danych, zapewnienie ochrony przed nieuprawnionym dostępem do bazy danych.
- Baza danych powinna zawierać indeksy na odpowiednich tabelach w celu poprawy wydajności zapytań SQL.
- System powinien informować użytkownika o dodaniu nowego pracownika do bazy danych poprzez wyświetlenie stosownego komunikatu w konsoli lub w interfejsie użytkownika
- System powinien informować użytkownika o dodaniu nowego produktu do bazy danych poprzez wyświetlenie komunikatu tekstowego zawierającego nazwę dodanego produktu.

- System powinien informować użytkowników o niskim stanie magazynowym produktów poprzez wyświetlenie odpowiedniego komunikatu w momencie, gdy liczba sztuk danego produktu spadnie poniżej 10.
- Brak możliwości usunięcia kontrahenta

MODEL FIZYCZNY



Skrypt tworzenia bazy danych

```
CREATE DATABASE Hurtownia
ON
PRIMARY
(
    NAME = 'hurtownia_wyrobow_chemicznych_dat1',
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.OPTIMA\MSSQL\DATA\hurtownia_wyrobow_chemicznych1.mdf',
    SIZE = 100MB,
    MAXSIZE = 1000MB,
    FILEGROWTH = 10MB
),
(
    NAME = 'hurtownia_wyrobow_chemicznych_dat2',
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.OPTIMA\MSSQL\DATA\hurtownia_wyrobow_chemicznych2.mdf',
    SIZE = 100MB,
    MAXSIZE = 1000MB,
    FILEGROWTH = 10MB
)
LOG ON
(
    NAME = 'hurtownia_wyrobow_chemicznych_log',
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.OPTIMA\MSSQL\DATA\hurtownia_wyrobow_chemicznych.ldf',
    SIZE = 50MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 10%
);
```

Wykaz utworzonych tabel fizycznych

Płatność					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdPłatnosci	INT	NOT NULL	Identyfikator płatności, generowany automatycznie wg. Porządku(1,1)
2		RodzajPłatnosci	VARCHAR(10)	NOT NULL	Atrybut zawierający rodzaj płatności

Stanowisko					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdStanowiska	INT	NOT NULL	Identyfikator stanowiska, generowany automatycznie wg. Porządku(1,1)
2		NazwaStanowiska	VARCHAR(20)	NOT NULL	Atrybut zawierający nazwę stanowiska

Kategoria					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdKategorii	INT	NOT NULL	Identyfikator stanowiska, generowany automatycznie wg. Porządku(1,1)
2		Nazwa	VARCHAR(20)	NOT NULL	Atrybut zawierający nazwę kategorii

Województwo					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdWojewodztwa	INT	NOT NULL	Identyfikator województwa, generowany automatycznie wg. Porządku(1,1)
2		Nazwa	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwę województwa

Adresy					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdAdresu	INT	NOT NULL	Identyfikator adresu, generowany automatycznie wg. Porządku(1,1)
2	FK	IdWojewództwa	INT	NOT NULL	Identyfikator Województwa do którego adres jest przypisany
3		Ulica	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwę ulicy
4		KodPocztowy	VARCHAR(6)	NOT NULL	Atrybut zawierający kod pocztowy
5		Miasto	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwę miasta

Pracownik					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdPracownika	INT	NOT NULL	Identyfikator pracownika, generowany automatycznie wg. Porządku(1,1)
2	FK	IdStanowiska	INT	NOT NULL	Identyfikator Stanowiska wybierany przez użytkownika
3		Imie	VARCHAR(30)	NOT NULL	Atrybut zawierający imię
4		Nazwisko	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwisko
5		Email	VARCHAR(30)	NOT NULL	Atrybut zawierający adres email
6		NumerTelefonu	VARCHAR(9)	NOT NULL	Atrybut zawierający numer telefonu

Kontrahent					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdKontrahenta	INT	NOT NULL	Identyfikator kontrahenta, generowany automatycznie wg. Porządku(1,1)
2	FK	IdAdresu	INT	NOT NULL	Identyfikator Adresu wybierany przez użytkownika
3		NazwaFirmy	VARCHAR(30)	NULL	Atrybut zawierający nazwę firmy
4		NIP	VARCHAR(10)	NULL	Atrybut zawierający numer NIP
5		Imie	VARCHAR(30)	NULL	Atrybut zawierający imię
6		Nazwisko	VARCHAR(30)	NULL	Atrybut zawierający nazwisko
7		PESEL	VARCHAR(11)	NULL	Atrybut zawierający numer PESEL
8		TypKlienta	BIT	NOT NULL	Atrybut zawierający typ klienta, gdzie 0 oznacza firmę, a 1 osobę fizyczna

Sprzedaz					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdSprzedazy	INT	NOT NULL	Identyfikator sprzedaży, generowany automatycznie wg. Porządku(1,1)
2	PK,FK	IdKontrahenta	INT	NOT NULL	Identyfikator Kontrahenta wybierany przez użytkownika
3	FK	IdPracownika	INT	NOT NULL	Identyfikator Pracownika wybierany przez użytkownika
4		DataSprzedazy	DATE	NOT NULL	Atrybut zawierający datę sprzedaży
5	FK	IdPlatnosci	INT	NOT NULL	Identyfikator Płatności wybierany przez użytkownika
6		NumerFaktury	VARCHAR(19)	NOT NULL	Atrybut zawierający numer faktury
7		Rabat	DECIMAL(5,2)	NOT NULL	Atrybut zawierający wartość rabatu

Dostawca					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdDostawcy	INT	NOT NULL	Identyfikator dostawcy, generowany automatycznie wg. Porządku(1,1)
2	FK	IdAdresu	INT	NOT NULL	Identyfikator Adresu wybierany przez użytkownika
3		Nazwa	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwę
4		NIP	VARCHAR(10)	NOT NULL	Atrybut zawierający numer NIP

Zakup					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdZakupu	INT	NOT NULL	Identyfikator zakupu, generowany automatycznie wg. Porządku(1,1)
2	PK ,FK	IdDostawcy	INT	NOT NULL	Identyfikator Dostawcy wybierany przez użytkownika
3	FK	IdPracownika	INT	NOT NULL	Identyfikator Pracownika wybierany przez użytkownika
4	FK	IdPłatnosci	INT	NOT NULL	Identyfikator Płatności wybierany przez użytkownika
5		NumerFaktury	VARCHAR(30)	NOT NULL	Atrybut zawierający adres email
6		DataZakupu	DATE	NOT NULL	Atrybut zawierający numer telefonu
7		Rabat	DECIMAL(5,2)	NOT NULL	Atrybut zawierający wartość rabatu
8		CzyZapłacono	BIT	NOT NULL	Atrybut określający czy zapłacono, gdzie 0 oznacza nie, a 1 oznacza tak.

Produkt					
LP	Atrybut Kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK	IdProduktu	INT	NOT NULL	Identyfikator produktu, generowany automatycznie wg. Porządku(1,1)
2	FK	IdKategorii	INT	NOT NULL	Identyfikator Kategorii do której produkt jest przypisany
3		Nazwa	VARCHAR(30)	NOT NULL	Atrybut zawierający nazwę
4		LiczbaSztukMagazyn	INT	NOT NULL	Atrybut zawierający liczbę sztuk na magazynie
5		CenaCennik	MONEY	NOT NULL	Atrybut zawierający cenę z cennika

SzczegolyZakupu					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK,FK	IdZakupu	INT	NOT NULL	Identyfikator zakupu do którego szczegóły są przypisane
2	PK,FK	IdProduktu	INT	NOT NULL	Identyfikator zakupionego produktu
3		CenaZakupu	MONEY	NOT NULL	Atrybut zawierający cenę zakupu
4		LiczbaSztuk	INT	NOT NULL	Atrybut zawierający liczbę sztuk zakupionego produktu
5		StawkaVat	DECIMAL(5,2)	NOT NULL	Atrybut zawierający stawkę VAT

SzczegolySprzedazy					
LP	Atrybut kluczowy	Atrybut	Typ danych	Czy wymagany	Opis
1	PK,FK	IdSprzedazy	INT	NOT NULL	Identyfikator sprzedaży do której szczegóły są przypisane
2	PK,FK	IdProduktu	INT	NOT NULL	Identyfikator sprzedanego produktu
3		CenaSprzedazy	MONEY	NOT NULL	Atrybut zawierający cenę sprzedaży
4		LiczbaSztuk	INT	NOT NULL	Atrybut zawierający liczbę sztuk sprzedanego produktu
5		StawkaVat	DECIMAL(5,2)	NOT NULL	Atrybut zawierający procentową stawkę VAT

Skrypt tworzenia tabel i referencji między nimi

```
CREATE TABLE tbl_Platnosc (  
    IdPlatnosci INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    RodzajPlatnosci VARCHAR(10) NOT NULL  
);  
  
CREATE TABLE tbl_Stanowisko (  
    IdStanowiska INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    NazwaStanowiska VARCHAR(20) NOT NULL  
);  
  
CREATE TABLE tbl_Kategoria (  
    IdKategorii INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    Nazwa VARCHAR(30) NOT NULL  
);  
  
CREATE TABLE tbl_Wojewodztwo (  
    IdWojewodztwa INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    Nazwa VARCHAR(30) NOT NULL  
);  
  
CREATE TABLE tbl_Adresy (  
    IdAdresu INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    IdWojewodztwa INT NOT NULL,  
    Ulica VARCHAR(30) NOT NULL,  
    KodPocztowy VARCHAR(6) NOT NULL,  
    Miasto VARCHAR(30) NOT NULL,  
    CONSTRAINT FK_Wojewodztwo FOREIGN KEY (IdWojewodztwa) REFERENCES  
tbl_Wojewodztwo(IdWojewodztwa)  
);  
  
CREATE TABLE tbl_Pracownik (  
    IdPracownika INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    IdStanowiska INT NOT NULL,  
    Imie VARCHAR(30) NOT NULL,  
    Nazwisko VARCHAR(30) NOT NULL,  
    Email VARCHAR(30) NOT NULL,  
    NumerTelefonu VARCHAR(9) NOT NULL,  
    CONSTRAINT FK_Pracownik_Stanowisko FOREIGN KEY (IdStanowiska) REFERENCES  
tbl_Stanowisko(IdStanowiska)  
);  
  
CREATE TABLE tbl_Kontrahent (  
    IdKontrahenta INT IDENTITY(1,1) PRIMARY KEY NOT NULL,  
    IdAdresu INT NOT NULL,  
    NazwaFirmy VARCHAR(30),  
    NIP VARCHAR(13),  
    Imie VARCHAR(30),  
    Nazwisko VARCHAR(30),  
    PESEL VARCHAR(11),  
    TypKlienta BIT NOT NULL,  
    CONSTRAINT FK_Kontrahent_Adres FOREIGN KEY (IdAdresu) REFERENCES tbl_Adresy(IdAdresu)  
);  
  
CREATE TABLE tbl_Sprzedaz (  
    IdSprzedazy INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
```

```

        IdKontrahenta INT NOT NULL,
        IdPracownika INT NOT NULL,
        DataSprzedazy DATE NOT NULL,
        IdPlatnosci INT NOT NULL,
        NumerFaktury VARCHAR(19) NOT NULL,
        Rabat DECIMAL(5,2) NOT NULL
        CONSTRAINT FK_Sprzedaz_Kontrahent FOREIGN KEY (IdKontrahenta) REFERENCES
tbl_Kontrahent(IdKontrahenta),
        CONSTRAINT FK_Sprzedaz_Pracownik FOREIGN KEY (IdPracownika) REFERENCES
tbl_Pracownik(IdPracownika),
        CONSTRAINT FK_Sprzedaz_Płatność FOREIGN KEY (IdPlatnosci) REFERENCES
tbl_Platnosc(IdPlatnosci)
);

CREATE TABLE tbl_Dostawca (
    IdDostawcy INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
    IdAdresu INT NOT NULL,
    Nazwa VARCHAR(30) NOT NULL,
    NIP VARCHAR(13) NOT NULL,
    CONSTRAINT FK_Dostawca_Adresy FOREIGN KEY (IdAdresu) REFERENCES tbl_Adresy(IdAdresu)
);

CREATE TABLE tbl_Zakup (
    IdZakupu INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
    IdDostawcy INT NOT NULL,
    IdPracownika INT NOT NULL,
    IdPlatnosci INT NOT NULL,
    NumerFaktury VARCHAR(30) NOT NULL,
    DataZakupu DATE NOT NULL,
    Rabat DECIMAL(5,2) NOT NULL,
    CzyZaplacono BIT NOT NULL,
    CONSTRAINT FK_Zakup_Dostawca FOREIGN KEY (IdDostawcy) REFERENCES
tbl_Dostawca(IdDostawcy),
    CONSTRAINT FK_Zakup_Pracownik FOREIGN KEY (IdPracownika) REFERENCES
tbl_Pracownik(IdPracownika),
    CONSTRAINT FK_Zakup_Platnosc FOREIGN KEY (IdPlatnosci) REFERENCES
tbl_Platnosc(IdPlatnosci)
);

CREATE TABLE tbl_Produkt (
    IdProduktu INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
    IdKategorii INT NOT NULL,
    Nazwa VARCHAR(30) NOT NULL,
    LiczbaSztukMagazyn INT NOT NULL,
    CenaCennik MONEY NOT NULL,
    CONSTRAINT FK_Produkt_Kategoria FOREIGN KEY (IdKategorii) REFERENCES tbl_Kategoria
(IdKategorii)
);

```

```

CREATE TABLE tbl_SzczegolyZakupu (
    IdZakupu INT NOT NULL,
    IdProduktu INT NOT NULL,
    CenaZakupu MONEY NOT NULL,
    LiczbaSztuk INT NOT NULL,
    StawkaVat DECIMAL(5,2) NOT NULL,
    CONSTRAINT PK_SzczegolyZakupu PRIMARY KEY (IdZakupu, IdProduktu),
    CONSTRAINT FK_SzczegolyZakupu_Zakup FOREIGN KEY (IdZakupu) REFERENCES
tbl_Zakup(IdZakupu),
    CONSTRAINT FK_SzczegolyZakupu_Produkt FOREIGN KEY (IdProduktu) REFERENCES
tbl_Produkt(IdProduktu)
);

CREATE TABLE tbl_SzczegolySprzedazy (
    IdSprzedazy INT NOT NULL,
    IdProduktu INT NOT NULL,
    CenaSprzedazy MONEY NOT NULL,
    LiczbaSztuk INT NOT NULL,
    StawkaVat DECIMAL(5,2) NOT NULL,
    CONSTRAINT PK_SzczegolySprzedazy PRIMARY KEY (IdSprzedazy, IdProduktu),
    CONSTRAINT FK_SzczegolySprzedazy_Sprzedaz FOREIGN KEY (IdSprzedazy) REFERENCES
tbl_Sprzedaz(IdSprzedazy),
    CONSTRAINT FK_SzczegolySprzedazy_Produkt FOREIGN KEY (IdProduktu) REFERENCES
tbl_Produkt(IdProduktu)
);

```


Wykaz zaimplementowanych indeksów

	Nazwa tabeli	Nazwa indeksu	Typ indeksu
1	tbl_Adresy	idx_adresy_id_adresu	NONCLUSTERED
2	tbl_Adresy	idx_adresy_id_województwa	NONCLUSTERED
3	tbl_Adresy	idx_adresy_kod_pocztowy	NONCLUSTERED
4	tbl_Adresy	PK__tbl_Adre__000394EE597A2...	CLUSTERED
5	tbl_Dostawca	idx_dostawca_id_adresu	NONCLUSTERED
6	tbl_Dostawca	idx_dostawca_id_dostawcy	NONCLUSTERED
7	tbl_Dostawca	idx_dostawca_nip	NONCLUSTERED
8	tbl_Dostawca	PK__tbl_Dost__1B2395A646504B...	CLUSTERED
9	tbl_KategoriaProdu...	idx_kategoria_id_kategorii	NONCLUSTERED
10	tbl_KategoriaProdu...	PK__tbl_Kate__31412B2E0904D1...	CLUSTERED
11	tbl_Kontrahent	idx_kontrahent_id_adresu	NONCLUSTERED
12	tbl_Kontrahent	idx_kontrahent_id_kontrahenta	NONCLUSTERED
13	tbl_Kontrahent	PK__tbl_Kont__2EC097F4F31009...	CLUSTERED
14	tbl_Platnosc	idx_platnosc_id_platnosci	NONCLUSTERED
15	tbl_Platnosc	PK__tbl_Plat__E53BC2CDE5E7C...	CLUSTERED
16	tbl_Pracownik	idx_pracownik_id_pracownika	NONCLUSTERED
17	tbl_Pracownik	idx_pracownik_id_stanowiska	NONCLUSTERED
18	tbl_Pracownik	idx_pracownik_numer_telefonu	NONCLUSTERED
19	tbl_Pracownik	PK__tbl_Prac__7BF2E88084A845...	CLUSTERED
20	tbl_Produkt	idx_produkt_id_kategorii	NONCLUSTERED
21	tbl_Produkt	idx_produkt_id_produktu	NONCLUSTERED
22	tbl_Produkt	PK__tbl_Prod__07856793CCB43...	CLUSTERED
23	tbl_Sprzedaz	idx_sprzedaz_id_kontrahenta	NONCLUSTERED
24	tbl_Sprzedaz	idx_sprzedaz_id_platnosci	NONCLUSTERED
25	tbl_Sprzedaz	idx_sprzedaz_id_pracownika	NONCLUSTERED
26	tbl_Sprzedaz	idx_sprzedaz_id_sprzedazy	NONCLUSTERED
27	tbl_Sprzedaz	idx_sprzedaz_numer_faktury	NONCLUSTERED
28	tbl_Sprzedaz	PK__tbl_Sprz__D7FE304A8B38F...	CLUSTERED
29	tbl_Stanowisko	idx_stanowisko_id_stanowiska	NONCLUSTERED
30	tbl_Stanowisko	PK__tbl_Stan__7EAB11DA81BC3...	CLUSTERED
31	tbl_SzczegolySprze...	idx_szczegolysprzedazy_id_produ...	NONCLUSTERED
32	tbl_SzczegolySprze...	idx_szczegolysprzedazy_id_sprze...	NONCLUSTERED
33	tbl_SzczegolySprze...	PK_SzczegolySprzedazy	CLUSTERED
34	tbl_SzczegolyZakupu	idx_szczegolyzakupu_id_produktu	NONCLUSTERED
35	tbl_SzczegolyZakupu	idx_szczegolyzakupu_id_zakupu	NONCLUSTERED
36	tbl_SzczegolyZakupu	PK_SzczegolyZakupu	CLUSTERED
37	tbl_Wojewodztwo	idx_wojewodztwo_id	NONCLUSTERED
38	tbl_Wojewodztwo	PK__tbl_Woje__4FB09E55E1AF0...	CLUSTERED
39	tbl_Zakup	idx_zakup_czy_zaplacono	NONCLUSTERED
40	tbl_Zakup	idx_zakup_id_dostawcy	NONCLUSTERED
41	tbl_Zakup	idx_zakup_id_platnosci	NONCLUSTERED
42	tbl_Zakup	idx_zakup_id_pracownika	NONCLUSTERED
43	tbl_Zakup	idx_zakup_id_zakupu	NONCLUSTERED
44	tbl_Zakup	idx_zakup_numer_faktury	NONCLUSTERED
45	tbl_Zakup	PK__tbl_Zaku__3571ABD8E31B6...	CLUSTERED

Skrypt tworzenia indeksów

```
CREATE INDEX idx_adresy_kod_pocztowy ON dbo.tbl_Adresy (KodPocztowy);
CREATE INDEX idx_adresy_id_adresu ON dbo.tbl_Adresy (IdAdresu);
CREATE INDEX idx_adresy_id_wojewodztwa ON dbo.tbl_Adresy (IdWojewodztwa);

CREATE INDEX idx_dostawca_id_adresu ON dbo.tbl_Dostawca (IdAdresu);
CREATE INDEX idx_dostawca_id_dostawcy ON dbo.tbl_Dostawca (IdDostawcy);
CREATE INDEX idx_dostawca_nip ON dbo.tbl_Dostawca (NIP);

CREATE INDEX idx_kategoria_id_kategorii ON dbo.tbl_Kategoria (IdKategorii);
CREATE INDEX idx_kontrahent_id_adresu ON dbo.tbl_Kontrahent (IdAdresu);
CREATE INDEX idx_kontrahent_id_kontrahenta ON dbo.tbl_Kontrahent (IdKontrahenta);

CREATE INDEX idx_platnosc_id_platnosci ON dbo.tbl_Platnosc (IdPlatnosci);
CREATE INDEX idx_pracownik_id_pracownika ON dbo.tbl_Pracownik (IdPracownika);
CREATE INDEX idx_pracownik_id_stanowiska ON dbo.tbl_Pracownik (IdStanowiska);
CREATE INDEX idx_pracownik_numer_telefonu ON dbo.tbl_Pracownik (NumerTelefonu);

CREATE INDEX idx_produkct_id_kategorii ON dbo.tbl_Produkt (IdKategorii);
CREATE INDEX idx_produkct_id_produkct ON dbo.tbl_Produkt (IdProdukct);

CREATE INDEX idx_sprzedaz_id_kontrahenta ON dbo.tbl_Sprzedaz (IdKontrahenta);
CREATE INDEX idx_sprzedaz_id_platnosci ON dbo.tbl_Sprzedaz (IdPlatnosci);
CREATE INDEX idx_sprzedaz_id_pracownika ON dbo.tbl_Sprzedaz (IdPracownika);
CREATE INDEX idx_sprzedaz_id_sprzedazy ON dbo.tbl_Sprzedaz (IdSprzedazy);
CREATE INDEX idx_sprzedaz_numer_faktury ON dbo.tbl_Sprzedaz (NumerFaktury);

CREATE INDEX idx_stanowisko_id_stanowiska ON dbo.tbl_Stalowisko (IdStanowiska);

CREATE INDEX idx_szczegolysprzedazy_id_produkct ON dbo.tbl_SzczegolySprzedazy
(IdProdukct);
CREATE INDEX idx_szczegolysprzedazy_id_sprzedazy ON dbo.tbl_SzczegolySprzedazy
(IdSprzedazy);

CREATE INDEX idx_szczegolyzakupu_id_produkct ON dbo.tbl_SzczegolyZakupu (IdProdukct);
CREATE INDEX idx_szczegolyzakupu_id_zakupu ON dbo.tbl_SzczegolyZakupu (IdZakupu);

CREATE INDEX idx_wojewodztwo_id ON dbo.tbl_Wojewodztwo (IdWojewodztwa);

CREATE INDEX idx_zakup_czy_zaplacono ON dbo.tbl_Zakup (CzyZaplacono);
CREATE INDEX idx_zakup_id_dostawcy ON dbo.tbl_Zakup (IdDostawcy);
CREATE INDEX idx_zakup_id_platnosci ON dbo.tbl_Zakup (IdPlatnosci);
CREATE INDEX idx_zakup_id_pracownika ON dbo.tbl_Zakup (IdPracownika);
CREATE INDEX idx_zakup_id_zakupu ON dbo.tbl_Zakup (IdZakupu);
CREATE INDEX idx_zakup_numer_faktury ON dbo.tbl_Zakup (NumerFaktury);
```

Skrypt ładujący przykładowe dane do tabel

```
INSERT INTO dbo.tbl_Platnosc (RodzajPlatnosci)
VALUES ( 'Gotówka'),
       ( 'Karta'),
       ( 'Przelew');
```

```
INSERT INTO dbo.tbl_Stanowisko (NazwaStanowiska)
VALUES ( 'Kasjer-Sprzedawca'),
       ( 'Magazynier'),
       ( 'Kierownik');
```

```
INSERT INTO dbo.tbl_Kategoria (Nazwa)
VALUES ( 'Farba'),
       ( 'Rozpuszczalnik'),
       ( 'Rozcieńczalnik'),
       ( 'Odczynnik Laboratoryjny');
```

```
INSERT INTO dbo.tbl_Wojewodztwo (Nazwa)
VALUES ( 'Łódzkie'),
       ( 'Mazowieckie'),
       ( 'Małopolskie'),
       ( 'Podlaskie'),
       ( 'Opolskie'),
       ( 'Świętokrzyskie'),
       ( 'Pomorskie'),
       ( 'Zachodniopomorskie'),
       ( 'Lubuskie'),
       ( 'Lubelskie'),
       ( 'Warmińsko-Mazurskie'),
       ( 'Wielkopolskie'),
       ( 'Podkarpackie'),
       ( 'Śląskie'),
       ( 'Kujawsko-Pomorskie'),
       ( 'Dolnośląskie');
```

```
INSERT INTO dbo.tbl_Adresy (IdWojewodztwa,Ulica,KodPocztowy,Miasto)
VALUES (8,'Krótka 12','54-645','Szczecin'),
       (2,'Mazowiecka 4/5','00-001','Warszawa'),
       (3,'Źródlana 10','03-432','Kraków'),
       (7,'Słoneczna 12','41-431','Sopot'),
       (12,'Koziołka Matołka 9','62-618','Poznań'),
       (1,'Zgierska 18','91-013','Łódź'),
       (4,'Szkolna 17','16-588','Białystok'),
       (16,'Piłsudskiego 144','07-777','Wrocław'),
       (11,'Marynarska 88','23-412','Ełk'),
       (14,'Górnicza 11/23','43-222','Katowice'),
       (6,'Ojca Mateusza 71','55-687','Sandomierz'),
       (13,'Wołodyjowskiego 99','87-999','Przemyśl'),
       (5,'Piekarska 45','23-122','Opole'),
       (10,'Zakładowa 62','11-039','Lublin'),
       (7,'Obronna 239','42-123','Gdańsk');
```

```
(14, 'Świętej Barbary 87', '21-091', 'Zabrze'),
(4, 'Konopnickiej 8/12', '16-300', 'Augustów'),
(15, 'Aleja Ojca Rydyka 1', '78-992', 'Toruń'),
(2, 'Kolikowskiego 32', '23-123', 'Pruszków'),
(8, 'Rzeczna 77', '08-767', 'Kołbaskowo'),
(6, 'Harcerska 57/3', '73-622', 'Starachowice'),
(14, 'Górna 8', '12-323', 'Gliwice'),
(9, 'Graniczna 124', '98-212', 'Gorzów Wlkp'),
(15, 'Leśna 34', '92-812', 'Bydgoszcz'),
(1, 'Motorowa 82/84', '85-101', 'Ozorków'),
(16, 'Złota 10', '44-290', 'Bogatynia'),
(3, 'Papieska 99', '87-212', 'Bochnia'),
(8, 'Bursztynowa 24', '92-123', 'Kołobrzeg'),
(7, 'Armat Helskich', '66-238', 'Puck'),
(12, 'Budowniczych 11', '37-421', 'Koziegłowy'),
(1, 'Promienista 3', '95-100', 'Zgierz'),
(10, 'Książeca 54', '81-203', 'Zamość'),
(5, 'Piwna 23/32', '67-890', 'Namysłów');
```

```
INSERT INTO dbo.tbl_Pracownik (IdStanowiska, Imie, Nazwisko, Email, NumerTelefonu)
VALUES (1, 'Łukasz', 'Dąbrowski', 'dab.lukas@wp.pl', '666777888'),
(3, 'Tomasz', 'Kowalczyk', 'tomek.kowal@gmail.com', '123456987'),
(2, 'Piotr', 'Wójcik', 'piotr.wojcik@o2.pl', '789456123'),
(1, 'Paweł', 'Rosiewicz', 'pawel.rosiewicz@onet.pl', '632193223'),
(1, 'Bogdan', 'Kosiewicz', 'kosabodzio@wp.pl', '789234211'),
(1, 'Przemysław', 'Wojdaniak', 'p.wojdaniak@gmail.com', '872892301'),
(1, 'Robert', 'Adamiak', 'adamiak.robert@o2.pl', '774329001'),
(1, 'Joanna', 'Gąbka', 'spongeasia@wp.pl', '632878212'),
(1, 'Anna', 'Nowak', 'anianowak@o2.pl', '823190783'),
(1, 'Henryk', 'Piotrowicz', 'heniu.piotrowicz@gmail.com', '835552800'),
(1, 'Bogumił', 'Banasiak', 'ban.bogus@interia.pl', '507307288'),
(1, 'Halina', 'Janiak', 'janiak.halina@gmail.com', '783219322'),
(2, 'Michał', 'Rost', 'michal.rost@interia.pl', '699126332'),
(2, 'Adrian', 'Kołodziej', 'adi.kolodziej@wp.pl', '673809201'),
(2, 'Wacław', 'Krach', 'krach.waclaw@o2.pl', '784421801'),
(2, 'Sebastian', 'Wilski', 'wilski.sebastian@wp.pl', '683225509'),
(2, 'Ryszard', 'Bogucki', 'r.bogucki@gmail.com', '701983265'),
(2, 'Sławomir', 'Borewicz', '07zglossie@interia.pl', '500617883'),
(2, 'Hubert', 'Dreman', 'h.dreman@wp.pl', '789375321'),
(2, 'Borys', 'Józefiak', 'b.jozefiak@o2.pl', '511289303'),
(2, 'Rafał', 'Strzelczyk', 'rafal.strzelczyk1@gmail.com', '832000558'),
(3, 'Adam', 'Kazmierczak', 'kazmierczak.adam@onet.pl', '599301321'),
(3, 'Krystyna', 'Izbicka', 'k.izbicka@wp.pl', '601732449'),
(3, 'Hanna', 'Pawlicka', 'pawlickahanna@o2.pl', '540032167'),
(3, 'Dagmara', 'Grzesiak', 'grzesiakdagmara2@gmail.com', '754399302');
```

```
INSERT INTO dbo.tbl_Kontrahent (IdAdresu, PESEL, Nazwisko, Imie, NazwaFirmy, NIP,
TypKlienta)
VALUES (2, '63315630567', 'Łukasz', 'Dąbrowski', '', '', 1),
(4, '64043555045', 'Tomasz', 'Kowalczyk', '', '', 1),
(5, '75209565918', 'Piotr', 'Wójcik', '', '', 1),
(6, '77980770593', 'Marek', 'Mostowiak', '', '', 1),
(7, '70005863821', 'Aleksander', 'Jabłonowski', '', '', 1),
(8, '96618584576', 'Marcin', 'Osadowski', '', '', 1),
(9, '73766214286', 'Artur', 'Lenart', '', '', 1),
(10, '64462505839', 'Marek', 'Nowak', '', '', 1),
```

```

(11, '71241345122', 'Maksymilian', 'Berent', '', '', 1),
(12, '69003187509', 'Piotr', 'Pawlak', '', '', 1),
(13, '62809312242', 'Martyna', 'Budniak', '', '', 1),
(14, '87440100518', 'Michał', 'Olczak', '', '', 1),
(15, '90154063508', 'Julia', 'Fortuna', '', '', 1),
(16, '83774884179', 'Jarosław', 'Niemiecki', '', '', 1),
(17, '67948568370', 'Oliwia', 'Łuczak', '', '', 1),
(18, '73500248709', 'Paulina', 'Dębska', '', '', 1),
(19, '68812018150', 'Marina', 'Kowalska', '', '', 1),
(20, '91469383951', 'Halina', 'Kiepska', '', '', 1),
(21, '84892890598', 'Marian', 'Paździoch', '', '', 1),
(22, '76445727748', 'Sylwester', 'Pawliczak', '', '', 1),
(23, '93710878408', 'Stanley', 'Funkel', '', '', 1),
(24, '88578742602', 'Szymon', 'Struś', '', '', 1),
(25, '82185747462', 'Milena', 'Ogórek', '', '', 1),
(26, '63348257714', 'Adam', 'Tuszyciel', '', '', 1),
(27, '81834568900', 'Aleksander', 'Pisarski', '', '', 1),
(28, '', '', '', 'Farutex sp. z.o.o.', '888-232-15-12', 0),
(29, '', '', '', 'UnionChem', '732-222-15-51', 0),
(30, '', '', '', 'Chemi-Tech', '954-167-22-37', 0),
(31, '', '', '', 'FHU Nitro', '777-854-69-22', 0),
(32, '', '', '', 'PHU Endriu', '882-001-32-89', 0),
(33, '', '', '', 'PPHU Petro', '812-456-30-44', 0),
(2, '', '', '', 'Super Chemia S.A', '881-256-88-66', 0),
(4, '', '', '', 'Anser FUFFU', '584-227-98-69', 0),
(7, '', '', '', 'Remonicista Wiesiek', '732-992-12-37', 0),
(10, '', '', '', 'STANDARD BUD', '547-850-06-98', 0),
(15, '', '', '', 'Niewiński S.C', '632-659-84-12', 0),
(18, '', '', '', 'Walczak remonty i wykończenia', '783-201-99-88', 0),
(19, '', '', '', 'Wiesł-Bud Wiesław Puchacki', '456-897-41-52', 0),
(6, '', '', '', 'Firma budowlana Joanna Kowalik', '488-878-45-66', 0),
(8, '', '', '', 'Sklep Chemiczny Malox', '721-366-97-45', 0),
(21, '', '', '', 'Szpachelka&Wiertarka', '579-451-54-74', 0),
(28, '', '', '', 'Tomtex Tomasz Król', '948-744-58-84', 0),
(11, '', '', '', 'Augustowskie Centrum Budowlane', '236-594-82-22', 0),
(10, '', '', '', 'Jarosław Wiedrzyński', '420-698-74-12', 0),
(12, '', '', '', 'Przem-Trans', '427-989-52-26', 0),
(29, '', '', '', 'Laboratorium Chemiczne LABCHEM', '369-741-85-21', 0),
(30, '', '', '', 'Pralnia Chemiczna PROX', '568-213-69-69', 0);

```

```

INSERT INTO dbo.tbl_Sprzedaz (IdKontrahenta, NumerFaktury, IdPracownika, IdPlatnosci,
DataSprzedazy, Rabat)

```

```

VALUES

```

```

('24', '2/1/2021', '19', '2', '2021-01-02 16:43:59', '0.1'),
('25', '3/1/2021', '10', '3', '2021-01-05 22:19:42', '0.0'),
('8', '5/1/2021', '14', '1', '2021-01-10 21:19:12', '0.0'),
('2', '273/1/2021', '15', '3', '2021-01-19 03:44:38', '0.0'),
('2', '274/1/2021', '5', '2', '2021-01-22 06:18:57', '0.1'),
('45', '276/1/2021', '19', '1', '2021-01-29 00:45:59', '0.0'),
('10', '277/1/2021', '13', '3', '2021-01-29 12:50:54', '0.0'),
('16', '279/1/2021', '14', '3', '2021-01-31 20:36:21', '0.1'),
('24', '280/2/2021', '17', '1', '2021-02-02 01:10:49', '0.0'),
('47', '281/2/2021', '18', '2', '2021-02-06 06:58:41', '0.1'),
('1', '282/2/2021', '7', '2', '2021-02-08 00:06:16', '0.0'),
('10', '283/2/2021', '15', '1', '2021-02-13 09:29:05', '0.0');

```

```

INSERT INTO dbo.tbl_Dostawca (IdAdresu, Nazwa, NIP)
VALUES (5, 'General Speditions', '723-431-24-14'),
(6, 'Logistic Team', '423-423-43-24'),
(7, 'Anser', '239-183-71-81'),
(11, 'Trailer Trans', '809-872-12-21'),
(31, 'DixiFlu', '900-218-92-11'),
(33, 'Petrochem SA', '762-991-23-28'),
(19, 'UniFracht', '876-329-00-22'),
(13, 'BAT S.P Z.O.O', '803-652-17-92'),
(22, 'JMP Group', '357-212-48-98'),
(24, 'PeterHurt', '809-721-74-95'),
(25, 'Victor-Max', '672-329-83-72'),
(12, 'GermanOil Polska', '761-549-76-45'),
(8, 'Quality Xpert', '641-872-10-09'),
(14, 'Natural Hydrogen', '238-910-32-32'),
(27, 'Bauman Logistics', '648-781-87-44'),
(10, 'Lau-Technics', '349-458-39-22'),
(21, 'PGH Pruszków', '354-978-48-54'),
(15, 'GreenWayCorp', '795-185-47-87'),
(20, 'Chemi-Sped', '510-239-78-56'),
(30, 'BlueWay', '842-798-54-87'),
(9, 'Union Of Chemistry', '787-159-66-42'),
(29, 'RBX Poland', '948-293-00-13');

```

```

INSERT INTO dbo.tbl_Zakup (IdDostawcy, IdPracownika, NumerFaktury, DataZakupu, Rabat,
IdPlatnosci, CzyZaplacono)
VALUES (3, 1, '1/06/2022', '2022-06-16 15:02:08', '0.05', 1, 1),
(5, 2, '', '2022-06-16 15:02:11', '0.1', 1, 1);

```

```

INSERT INTO dbo.tbl_Produkt (IdKategorii, Nazwa, LiczbaSztukMagazyn, CenaCennik)
VALUES
(1, 'DuluxWhite', 584, 25.00),
(2, 'Nitro', 1269, 9.99),
(3, 'EXTRA-BENZYL', 180, 13.45),
(1, 'Dekorol', 159, 33.85),
(1, 'Paintex', 121, 29.99),
(1, 'Ultra Spray', 197, 12.65),
(1, 'Finest Colour', 229, 39.99),
(2, 'Power Solvent', 298, 25.99),
(2, 'Kamikaze', 199, 19.99),
(2, 'Turbo', 147, 23.49),
(3, 'Benzyna Ekstrakcyjna', 95, 6.49),
(3, 'Detrixin', 180, 7.99),
(3, 'Izi', 52, 10.99),
(3, 'Harpun', 79, 5.99),
(2, 'S.T.A.L.K.E.R', 233, 12.99),
(4, 'Aldehyd Octowy', 40, 70.00),
(4, 'Krzemian Sodu', 39, 65.00),
(4, 'Perhydrol', 52, 50.00),
(4, 'Chloroform', 9, 99.99),
(4, 'Kwas Siarkowy', 4, 89.99),
(4, 'Amoniak', 45, 39.99),
(4, 'Acetylen', 29, 70.00),
(4, 'Sól Fizjologiczna', 1000, 0.50),
(4, 'Gliceryna', 499, 49.99),
(4, 'Glikol Propylenowy', 365, 69.99);

```



```

INSERT INTO dbo.tbl_SzczegolyZakupu (IdZakupu, IdProduktu, CenaZakupu, LiczbaSztuk,
StawkaVat)
VALUES ('1', '1', '20.00', '100', '0.08'),
('1', '2', '6.99', '50', '0.08'),
('1', '4', '25.00', '10', '0.08'),
('1', '9', '15.00', '3', '0.08'),
('1', '13', '5.00', '3', '0.08'),
('1', '15', '10.00', '5', '0.23'),
('1', '18', '39.99', '1', '0.23'),
('1', '21', '25.00', '5', '0.23'),
('1', '25', '13.00', '20', '0.08'),
('2', '3', '13.45', '40', '0.23'),
('2', '4', '30.40', '100', '0.23');

```

```

INSERT INTO dbo.tbl_SzczegolySprzedazy (IdSprzedazy, IdProduktu, CenaSprzedazy,
LiczbaSztuk, StawkaVat)
VALUES
('2', '1', '250.00', '10', '0.08'),
('2', '2', '499.50', '50', '0.23'),
('2', '5', '89.97', '3', '0.0'),
('2', '7', '79.98', '2', '0.0'),
('2', '8', '129.95', '5', '0.0'),
('2', '10', '70.47', '3', '0.0'),
('2', '13', '10.99', '1', '0.0'),
('2', '14', '5.99', '1', '0.23'),
('2', '15', '12.99', '1', '0.0'),
('2', '19', '99.99', '1', '0.0'),
('2', '20', '89.99', '1', '0.0'),
('2', '22', '70.00', '1', '0.23'),
('2', '24', '49.99', '1', '0.23'),
('2', '25', '139.98', '2', '0.0'),
('3', '3', '134.50', '10', '0.23'),
('3', '7', '39.99', '1', '0.0'),
('3', '8', '129.95', '5', '0.0'),
('3', '11', '64.90', '10', '0.0'),
('5', '6', '37.95', '3', '0.08'),
('5', '9', '39.98', '2', '0.08'),
('6', '5', '149.95', '5', '0.23'),
('6', '20', '1889.79', '21', '0.08'),
('6', '21', '2759.31', '69', '0.08'),
('7', '8', '2599.00', '100', '0.23'),
('7', '14', '257.57', '43', '0.0'),
('8', '20', '7019.22', '78', '0.23'),
('8', '22', '910.00', '13', '0.0'),
('9', '12', '727.09', '91', '0.0'),
('9', '25', '979.86', '14', '0.08'),
('10', '8', '1767.32', '68', '0.23'),
('11', '17', '2210.00', '34', '0.08'),
('12', '11', '642.51', '99', '0.0');

```

Wykaz utworzonych widoków

LP	Nazwa widoku	Opis
1	vw_produkty_na_wyczerpaniu	Pokazuje wszystkie produkty, które mają mniej niż 10 sztuk na magazynie
2	vw_Platnosci	Widok ten zwróci sumę wartości sprzedaży dla każdego rodzaju płatności
3	vw_liczebosc_kategorii	Pokazuje liczbę produktów w każdej kategorii
4	vw_sprzedaz_pracownika	Pokazuje imię, nazwisko i liczbę sprzedaży dla każdego pracownika
5	vw_sredni_rabat	Pokazuje średnią wartość rabatu udzielanego przy sprzedaży
6	vw_ilosc_produktow	Pokazuje nazwę kategorii, nazwę produktu i liczbę sztuk danego produktu w magazynie
7	vw_kwota_sprzedazy_produktow	Pokazuje kwotę sprzedaży danego produktu
8	vw_informacje_rozpuszczalnik	Pokazuje informacje o rozpuszczalnikach
9	vw_informacje_rozcieńczalnik	Pokazuje informacje o Rozcieńczalnik
10	vw_informacjeFarba	Pokazuje informacje o Farbach
11	vw_informacje_odczynnik_laboratoryjny	Pokazuje informacje o Odczynnikach Laboratoryjnych
12	vw_suma_platnosc_gotowka	Pokazuje sumę dokonanych zakupów przy pomocy gotówki
13	vw_suma_platnosc_karta	Pokazuje sumę dokonanych zakupów przy pomocy karty
14	vw_suma_platnosc_przelew	Pokazuje sumę dokonanych zakupów przy pomocy przelewu

Skrypt tworzenia widoków

```
CREATE VIEW vw_produkty AS
SELECT * FROM dbo.tbl_Produkt
WHERE LiczbaSztukMagazyn < 10;
GO
```

```
CREATE VIEW vw_Platnosci AS
SELECT p.RodzajPlatnosci, SUM(ss.CenaSprzedazy * ss.LiczbaSztuk) AS WartoscSprzedazy
FROM dbo.tbl_SzczegolySprzedazy ss
INNER JOIN dbo.tbl_Sprzedaz s ON ss.IdSprzedazy = s.IdSprzedazy
INNER JOIN dbo.tbl_Platnosc p ON s.IdPlatnosci = p.IdPlatnosci
GROUP BY p.RodzajPlatnosci;
GO
```

```
CREATE VIEW vw_Liczebosc_Kategorii AS
SELECT k.Nazwa, COUNT(p.IdProduktu) AS LiczbaProduktow FROM dbo.tbl_Kategoria k
LEFT JOIN dbo.tbl_Produkt p ON k.IdKategorii = p.IdKategorii
GROUP BY k.Nazwa
GO
```

```
CREATE VIEW vw_SprzedazPracownika AS
SELECT p.Imie, p.Nazwisko, COUNT(s.IdSprzedazy) AS LiczbaSprzedazy FROM dbo.tbl_Pracownik
p
LEFT JOIN dbo.tbl_Sprzedaz s ON p.IdPracownika = s.IdPracownika
GROUP BY p.Imie, p.Nazwisko
GO
```



```
CREATE VIEW vw_SredniRabat AS
SELECT AVG(Rabat) AS SredniRabat FROM dbo.tbl_Sprzedaz
GO
```

```
CREATE VIEW vw_IloscProduktow AS
SELECT k.Nazwa AS NazwaKategorii, p.Nazwa AS NazwaProduktu, p.LiczbaSztukMagazyn AS
LiczbaSztuk FROM dbo.tbl_Produkt p
INNER JOIN dbo.tbl_Kategoria k ON p.IdKategorii = k.IdKategorii
GO
```

```
CREATE VIEW vw_KwotaSprzedazyProduktow AS
SELECT k.Nazwa AS NazwaKategorii, p.Nazwa AS NazwaProduktu, SUM(s.CenaSprzedazy *
s.LiczbaSztuk) AS KwotaSprzedazy FROM dbo.tbl_Produkt p
INNER JOIN dbo.tbl_Kategoria k ON p.IdKategorii = k.IdKategorii
INNER JOIN dbo.tbl_SzczegolySprzedazy s ON p.IdProduktu = s.IdProduktu
GROUP BY k.Nazwa, p.Nazwa
GO
```

```
CREATE VIEW vw_InformacjeRozpuszczalnik
AS
SELECT p.Nazwa AS 'Nazwa Produktu',
p.LiczbaSztukMagazyn AS 'Stan Magazynowy',
p.CenaCennik AS 'Cena',
CASE WHEN p.LiczbaSztukMagazyn > 0 THEN 'Dostępny' ELSE 'Niedostępny' END AS
'Dostępność'
FROM dbo.tbl_Produkt p
WHERE p.IdKategorii IN (SELECT k.IdKategorii FROM dbo.tbl_Kategoria k WHERE k.Nazwa =
'Rozpuszczalnik')
GO
```

```
CREATE VIEW vw_InformacjeRozcienczalnik
AS
SELECT p.Nazwa AS 'Nazwa Produktu',
p.LiczbaSztukMagazyn AS 'Stan Magazynowy',
p.CenaCennik AS 'Cena',
CASE WHEN p.LiczbaSztukMagazyn > 0 THEN 'Dostępny' ELSE 'Niedostępny' END AS
'Dostępność'
FROM dbo.tbl_Produkt p
WHERE p.IdKategorii IN (SELECT k.IdKategorii FROM dbo.tbl_Kategoria k WHERE k.Nazwa =
'Rozcienczalnik')
GO
```

```
CREATE VIEW vw_InformacjeFarba
AS
SELECT p.Nazwa AS 'Nazwa Produktu',
p.LiczbaSztukMagazyn AS 'Stan Magazynowy',
p.CenaCennik AS 'Cena',
CASE WHEN p.LiczbaSztukMagazyn > 0 THEN 'Dostępny' ELSE 'Niedostępny' END AS
'Dostępność'
FROM dbo.tbl_Produkt p
WHERE p.IdKategorii IN (SELECT k.IdKategorii FROM dbo.tbl_Kategoria k WHERE k.Nazwa =
'Farba')
GO
```

```

CREATE VIEW vw_InformacjeOdczynnikLaboratoryjny
AS
SELECT p.Nazwa AS 'Nazwa Produktu',
       p.LiczbaSztukMagazyn AS 'Stan Magazynowy',
       p.CenaCennik AS 'Cena',
       CASE WHEN p.LiczbaSztukMagazyn > 0 THEN 'Dostępny' ELSE 'Niedostępny' END AS
       'Dostępność'
FROM dbo.tbl_Produkt p
WHERE p.IdKategorii IN (SELECT k.IdKategorii FROM dbo.tbl_Kategoria k WHERE k.Nazwa =
'Odczynnik Laboratoryjny')
GO

```

```

CREATE VIEW vw_SumaPlatnoscGotowka AS
SELECT SUM(ss.CenaSprzedazy*ss.LiczbaSztuk) AS WartoscSprzedazy FROM
dbo.tbl_SzczegolySprzedazy ss
INNER JOIN dbo.tbl_Sprzedaz s ON ss.IdSprzedazy = s.IdSprzedazy
INNER JOIN dbo.tbl_Platnosc p ON s.IdPlatnosci = p.IdPlatnosci
WHERE p.RodzajPlatnosci = 'Gotówka'
GO

```

```

CREATE VIEW vw_SumaPlatnoscKarta AS
SELECT SUM(ss.CenaSprzedazy*ss.LiczbaSztuk) AS WartoscSprzedazy FROM
dbo.tbl_SzczegolySprzedazy ss
INNER JOIN dbo.tbl_Sprzedaz s ON ss.IdSprzedazy = s.IdSprzedazy
INNER JOIN dbo.tbl_Platnosc p ON s.IdPlatnosci = p.IdPlatnosci
WHERE p.RodzajPlatnosci = 'Karta'
GO

```

```

CREATE VIEW vw_SumaPlatnoscPrzelew AS
SELECT SUM(ss.CenaSprzedazy*ss.LiczbaSztuk) AS WartoscSprzedazy FROM
dbo.tbl_SzczegolySprzedazy ss
INNER JOIN dbo.tbl_Sprzedaz s ON ss.IdSprzedazy = s.IdSprzedazy
INNER JOIN dbo.tbl_Platnosc p ON s.IdPlatnosci = p.IdPlatnosci
WHERE p.RodzajPlatnosci = 'Przelew'
GO

```

Wykaz utworzonych procedur składowanych

LP	Nazwa procedury	Opis
1	up_DodajProdukt	Procedura ta służy do dodawania nowych produktów
2	up_UsunProdukt	Procedura ta służy do usuwania produktów
3	up_EdytujProdukt	Procedura ta służy do edytowania informacji o produktach
4	up_DodajPracownika	Dodaje nowego pracownika
5	up_UsunPracownika	Usuwa pracownika o określonym identyfikatorze
6	up_AktualizujPracownika	Aktualizuje dane pracownika
7	up_WyswietlProduktyWedlugKategorii	Wyświetla informacje o produktach należących do określonej kategorii.
8	up_WyswietlSprzedazZPrzedzialu	Wyświetla informacje o sprzedaży dokonanej w określonym przedziale czasowym.
9	up_PobierzSzczegolySprzedazy	Wyświetla szczegółowe informacje o sprzedaży określonego produktu.
10	up_DodajZakup	Procedura dodająca zakup do bazy danych.
11	up_DodajSprzedaz	Procedura dodająca sprzedaż do bazy danych.
12	up_ObnizCenyProduktow	Umożliwia przyznanie rabatu na wszystkie produkty.
13	up_PodwyzkaCenProduktow	Umożliwia podwyższenie cen wszystkich produktów.
14	up_DodajKontrahenta	Procedura odpowiadająca za dodanie kontrahenta.
15	up_EdytujKontrahenta	Procedura pozwala na edytowanie danych kontrahenta.
16	up_DodajAdres	Procedura pozwala na dodanie adresu do bazy danych.

Skrypt tworzenia procedur składowanych

```
-----  
---  PROCEDURE DEFINITION  
---  up_DodajProdukt (@nazwa nvarchar(50), @liczbaSztukMagazyn int, @cenaCennik MONEY,  
---  @idKategorii int)  
---  CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"  
-----  
--Procedura ta dodaje nowy produkt do bazy danych.  
--  
--parametry wejściowe:  
--@nazwa - nazwa produktu  
--@liczbaSztukMagazyn - liczba sztuk produktu w magazynie  
--@cenaCennik - cena produktu w cenniku  
--@idKategorii - id kategorii, do której należy produkt  
--  
--parametry wyjściowe/zwracane wartości: Brak  
--  
--  
--Przykład użycia  
--EXEC up_DodajProdukt 'Nowy produkt', 10, 50.00, 1  
--  
--Wynik działania  
--Dodano nowy produkt do bazy danych.  
-----
```

```

CREATE PROCEDURE up_DodajProdukt
    @nazwa nvarchar(50),
    @liczbaSztukMagazyn int,
    @cenaCennik decimal(10,2),
    @idKategorii int
AS
BEGIN
    BEGIN TRY
        INSERT INTO dbo.tbl_Produkt (Nazwa, LiczbaSztukMagazyn, CenaCennik, IdKategorii)
        VALUES (@nazwa, @liczbaSztukMagazyn, @cenaCennik, @idKategorii)
    END TRY
    BEGIN CATCH
        SELECT ERROR_NUMBER() AS ErrorNumber, ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END
GO

```

```

-----
-- PROCEDURE DEFINITION
-- up_UsunProdukt (@idProduktu int)
-- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Procedura usuwa produkt o określonym identyfikatorze z tabeli dbo.tbl_Produkt
--
-- parametry wejściowe:
-- @idProduktu - identyfikator produktu do usunięcia z tabeli
--
-- parametry wyjściowe/zwracane wartości: Brak
--
--
-- Przykład użycia
-- EXEC up_UsunProdukt 16
--
-- Wynik działania
-- Procdeura usuwa wskazany produkt.
-----

```

```

CREATE PROCEDURE up_UsunProdukt
    @idProduktu int
AS
BEGIN
    BEGIN TRY
        IF EXISTS (SELECT 1 FROM dbo.tbl_Produkt WHERE IdProduktu = @idProduktu)
        BEGIN
            DELETE FROM dbo.tbl_Produkt
            WHERE IdProduktu = @idProduktu

            PRINT 'Produkt został pomyślnie usunięty.'
        END
        ELSE
        BEGIN
            RAISERROR('Nie znaleziono produktu o podanym IdProduktu.', 16, 1)
        END
    END TRY
    BEGIN CATCH
        PRINT 'Wystąpił błąd podczas usuwania produktu: ' + ERROR_MESSAGE()
    END CATCH
END
GO

```

```
-----  
--- PROCEDURE DEFINITION  
--- up_EdytujProdukt (@idProduktu int, @nazwa nvarchar(50), @liczbaSztukMagazyn int,  
@cenaCennik MONEY, @idKategorii int)  
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"  
-----
```

```
--Procedura służy do modyfikowania rekordów w tabeli "tbl_Produkt".  
--
```

```
--parametry wejściowe:  
-- Parametry wejściowe:  
-- @idProduktu - id produktu do edycji  
-- @nazwa - nowa nazwa produktu  
-- @liczbaSztukMagazyn - nowa liczba sztuk produktu w magazynie  
-- @cenaCennik - nowa cena produktu z cennika  
-- @idKategorii - id nowej kategorii produktu  
--
```

```
--parametry wyjściowe/zwracane wartości: Brak  
--
```

```
--Przykład użycia
```

```
--EXEC up_EdytujProdukt 1, 'Nowa nazwa', 50, 9.99, 2  
--
```

```
--Wynik działania
```

```
--Procedura modyfikuje wskazany produkt.  
-----
```

```
CREATE PROCEDURE up_EdytujProdukt
```

```
    @idProduktu INT,  
    @nazwa NVARCHAR(50),  
    @liczbaSztukMagazyn INT,  
    @cenaCennik MONEY,  
    @idKategorii INT
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRY
```

```
        BEGIN TRANSACTION;
```

```
        UPDATE dbo.tbl_Produkt
```

```
        SET Nazwa = @nazwa,  
            LiczbaSztukMagazyn = @liczbaSztukMagazyn,  
            CenaCennik = @cenaCennik,  
            IdKategorii = @idKategorii  
        WHERE IdProduktu = @idProduktu;
```

```
        COMMIT TRANSACTION;
```

```
    END TRY
```

```
    BEGIN CATCH
```

```
        IF @@TRANCOUNT > 0  
            ROLLBACK TRANSACTION;
```

```
        SELECT
```

```
            ERROR_NUMBER() AS ErrorNumber,  
            ERROR_MESSAGE() AS ErrorMessage,  
            ERROR_PROCEDURE() AS ErrorProcedure,  
            ERROR_LINE() AS ErrorLine;
```

```
        THROW;
```

```
    END CATCH;
```

```
END
```

```
GO
```

```

-----
--- PROCEDURE DEFINITION
--- up_DodajPracownika (@nazwisko nvarchar(50), @imie nvarchar(50), @numerTelefonu
nvarchar(15), @email nvarchar(50), @idStanowiska int)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Procedura składowana dodająca nowego pracownika do tabeli dbo.tbl_Pracownik.
--
--parametry wejściowe:
-- @nazwisko - nazwisko pracownika
-- @imie - imię pracownika
-- @numerTelefonu - numer telefonu pracownika
-- @email - adres email pracownika
-- @idStanowiska - identyfikator stanowiska pracownika
--
--parametry wyjściowe/zwracane wartości: Brak
--
--
--Przykład użycia
--EXEC up_DodajPracownika 'Kowalski', 'Jan', '123456789', 'jankowalski@example.com', 2
--
--Wynik działania
--Procdeura dodaje nowego pracownika
-----

```

```

CREATE PROCEDURE up_DodajPracownika
    @nazwisko nvarchar(50),
    @imie nvarchar(50),
    @numerTelefonu nvarchar(15),
    @email nvarchar(50),
    @idStanowiska int
AS
BEGIN
    BEGIN TRY
        INSERT INTO dbo.tbl_Pracownik (Nazwisko, Imie, NumerTelefonu, Email, IdStanowiska)
        VALUES (@nazwisko, @imie, @numerTelefonu, @email, @idStanowiska)
    END TRY
    BEGIN CATCH
        DECLARE @errorMessage nvarchar(4000);
        DECLARE @errorSeverity int;
        DECLARE @errorState int;

        SELECT
            @errorMessage = ERROR_MESSAGE(),
            @errorSeverity = ERROR_SEVERITY(),
            @errorState = ERROR_STATE();
        RAISERROR('Wystąpił błąd przy dodawaniu pracownika: %s', @errorSeverity, @errorState,
        @errorMessage);
    END CATCH
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- up_UnsunPracownika (@idPracownika int)
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Procedura ta usuwa pracownika z tabeli "dbo.tbl_Pracownik".
--
--parametry wejściowe:
-- @idPracownika - Identyfikator pracownika, który ma zostać usunięty.
--
--parametry wyjściowe/zwracane wartości: Brak
--
--/*
--Przykład użycia
--EXEC up_UnsunPracownika @idPracownika = 4
--
--Wynik działania
--Procdeura usuwa wybranego pracownika
-----

```

```

CREATE PROCEDURE up_UnsunPracownika
    @idPracownika INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        DELETE FROM dbo.tbl_Pracownik
        WHERE IdPracownika = @idPracownika;

        COMMIT TRANSACTION;

        PRINT 'Pracownik został pomyślnie usunięty.';
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;

        PRINT 'Wystąpił błąd podczas usuwania pracownika: ' + ERROR_MESSAGE();
    END CATCH;
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- up_AktualizujPracownika(@idPracownika int, @nazwisko nvarchar(50), @imie nvarchar(50),
@numerTelefonu nvarchar(15), @email nvarchar(50), @idStanowiska int)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Procedura służy do aktualizacji danych pracownika o podanym identyfikatorze.
--
--parametry wejściowe:
-- @idPracownika - identyfikator pracownika, którego dane będą aktualizowane
-- @nazwisko - nowe nazwisko pracownika
-- @imie - nowe imię pracownika
-- @numerTelefonu - nowy numer telefonu pracownika
-- @email - nowy adres e-mail pracownika
-- @idStanowiska - identyfikator stanowiska, na którym pracownik będzie pracował po
aktualizacji
--
--parametry wyjściowe/zwracane wartości: Brak
--
--Przykład użycia
--EXEC up_AktualizujPracownika 1, 'Kowalski', 'Jan', '123456789',
'jan.kowalski@example.com', 1
--
--Wynik działania
--Procdeura aktualizuje dane pracownika
-----

```

```

CREATE PROCEDURE up_AktualizujPracownika
    @idPracownika int,
    @nazwisko nvarchar(50),
    @imie nvarchar(50),
    @numerTelefonu nvarchar(15),
    @email nvarchar(50),
    @idStanowiska int
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        UPDATE dbo.tbl_Pracownik
        SET Nazwisko = @nazwisko, Imie = @imie, NumerTelefonu = @numerTelefonu, Email =
@email,
            IdStanowiska = @idStanowiska
        WHERE IdPracownika = @idPracownika;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;

        DECLARE @errorMessage NVARCHAR(1000);
        SET @errorMessage = ERROR_MESSAGE();

        RAISERROR(@errorMessage, 16, 1);
    END CATCH;
END
GO

```



```

-----
---  PROCEDURE DEFINITION
---  up_WyswietlProduktyWedlugKategorii (@idKategorii int)
---  CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----

--Procedura ta służy do wyświetlenia produktów należących do danej kategorii.
--
--parametry wejściowe:
-- @idKategorii - id kategorii, dla której mają być wyświetlone produkty
--
--parametry wyjściowe/zwracane wartości: Brak
--
--
--Przykład użycia
--EXEC up_WyswietlProduktyWedlugKategorii @idKategorii = 2

--Wynik działania
-- Wyświetla produkty z podanej kategorii
-----

CREATE PROCEDURE up_WyswietlProduktyWedlugKategorii
    @idKategorii int
AS
BEGIN
    BEGIN TRY
        SELECT IdProduktu, Nazwa, LiczbaSztukMagazyn, CenaCennik
        FROM dbo.tbl_Produkt
        WHERE IdKategorii = @idKategorii
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR_MESSAGE(),
               @ErrorSeverity = ERROR_SEVERITY(),
               @ErrorState = ERROR_STATE();

        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- up_WyswietlSprzedazZPrzedzialu (@DataPoczątkowa datetime, @DataKoncowa datetime)
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----

--Procedura służy do wyświetlenia informacji o sprzedaży z określonego przedziału czasu
--
--parametry wejściowe:
-- @DataPoczątkowa - data początkowa przedziału
-- @DataKoncowa - data końcowa przedziału
--
--parametry wyjściowe/zwracane wartości: Brak
--
--/*
--Przykład użycia
--EXEC up_WyswietlSprzedazZPrzedzialu '2020-01-01', '2021-01-31'
--
--Wynik działania
-- Wyświetla dane z tabeli Sprzedaż z określonej daty
-----
-----

CREATE PROCEDURE up_WyswietlSprzedazZPrzedzialu
    @DataPoczątkowa datetime,
    @DataKoncowa datetime
AS
BEGIN
    BEGIN TRY
        SELECT IdSprzedazy, NumerFaktury, DataSprzedazy, Rabat
        FROM dbo.tbl_Sprzedaz
        WHERE DataSprzedazy BETWEEN @DataPoczątkowa AND @DataKoncowa
    END TRY
    BEGIN CATCH
        SELECT ERROR_NUMBER() AS ErrorNumber, ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END
GO

-----
--- PROCEDURE DEFINITION
--- up_PobierzSzczegolySprzedazy (@idProduktu int)
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----

--Ta procedura zwraca szczegóły sprzedaży danego produktu.
--
--parametry wejściowe:
-- @idProduktu - identyfikator produktu, dla którego mają być pobrane szczegóły sprzedaży
--
--parametry wyjściowe/zwracane wartości: Brak
--
--
--Przykład użycia
--EXEC up_PobierzSzczegolySprzedazy @idProduktu = 1
--
--Wynik działania
--Zwraca tabele z historią sprzedaży danego produktu
-----

```

```

CREATE PROCEDURE up_PobierzSzczegolySprzedazy
    @idProduktu int
AS
BEGIN
    BEGIN TRY
        SELECT s.IdSprzedazy, s.NumerFaktury, s.DataSprzedazy, ss.CenaSprzedazy,
        ss.LiczbaSztuk, ss.StawkaVat
        FROM dbo.tbl_Sprzedaz s
        JOIN dbo.tbl_SzczegolySprzedazy ss ON s.IdSprzedazy = ss.IdSprzedazy
        WHERE ss.IdProduktu = @idProduktu
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000)
        DECLARE @ErrorSeverity INT
        DECLARE @ErrorState INT

        SELECT @ErrorMessage = ERROR_MESSAGE(), @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE()

    END CATCH
END
GO

```

```

-----
---  PROCEDURE DEFINITION
---  up_DodajZakup (@IdDostawcy int, @IdPracownika int, @NumerFaktury varchar(30),
---  @DataZakupu datetime, @Rabat decimal(5,2), @IdPlatnosci int, @CzyZaplacono bit,
---  @IdProduktu int, @CenaZakupu money, @LiczbaSztuk int, @StawkaVat decimal(5,2))
---  CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----

--Procedura dodająca informacje o nowym zakupie do bazy danych.
--
--parametry wejściowe:
--@IdDostawcy - id dostawcy, od którego kupiono produkty
--@IdPracownika - id pracownika, który dokonał zakupu
--@NumerFaktury - numer faktury związanej z zakupem
--@DataZakupu - data, w której dokonano zakupu
--@Rabat - wartość rabatu, który został naliczony na produkty
--@IdPlatnosci - id rodzaju płatności za zakupione produkty
--@CzyZaplacono - flaga informująca, czy za zakupione produkty została już zapłacona
--@IdProduktu - id produktu, który został zakupiony
--@CenaZakupu - cena jednostkowa produktu
--@LiczbaSztuk - liczba sztuk zakupionego produktu
--@StawkaVat - stawka podatku VAT, jaka została naliczona na zakupione produkty
--
--parametry wyjściowe/zwracane wartości: Brak
--
--
--Przykład użycia
--EXEC up_DodajZakup 1, 2, 'FV/2023/001', '2023-05-03 12:00:00', 0.05, 1, 1, 1, 25.00, 5,
0.23
--
--Wynik działania
--Dodanie informacji o nowym zakupie do bazy danych.
-----

```

```

CREATE PROCEDURE up_DodajZakup
    @IdDostawcy INT,
    @IdPracownika INT,
    @NumerFaktury VARCHAR(30),
    @DataZakupu DATETIME,
    @Rabat DECIMAL(5,2),
    @IdPlatnosci INT,
    @CzyZaplacono BIT,
    @IdProduktu INT,
    @CenaZakupu MONEY,
    @LiczbaSztuk INT,
    @StawkaVat DECIMAL(5,2)
AS
BEGIN
    BEGIN TRY
        DECLARE @IdZakupu INT

        INSERT INTO dbo.tbl_Zakup (IdDostawcy, IdPracownika, NumerFaktury, DataZakupu,
Rabat, IdPlatnosci, CzyZaplacono)
        VALUES (@IdDostawcy, @IdPracownika, @NumerFaktury, @DataZakupu, @Rabat,
@IdPlatnosci, @CzyZaplacono)

        SET @IdZakupu = SCOPE_IDENTITY()

        INSERT INTO dbo.tbl_SzczegolyZakupu (IdZakupu, IdProduktu, CenaZakupu,
LiczbaSztuk, StawkaVat)
        VALUES (@IdZakupu, @IdProduktu, @CenaZakupu, @LiczbaSztuk, @StawkaVat)
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_MESSAGE() AS ErrorMessage,
            ERROR_LINE() AS ErrorLine
    END CATCH
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- dbo.up_DodajSprzedaz(@IdKontrahenta INT, @NumerFaktury VARCHAR(30), @IdPracownika INT,
@IdPlatnosci INT, @DataSprzedazy DATETIME, @Rabat DECIMAL(5,2), @IdProduktu INT,
@CenaSprzedazy MONEY, @LiczbaSztuk INT, @StawkaVat DECIMAL(5,2))
--- CREATED BY: Grupa "Hurtownia Wytobow Chemicznych"
-----
--Procedura dodajaca informacje o nowej sprzedazy do bazy danych.
--
--parametry wejsciowe:
-- @IdKontrahenta - ID kontrahenta zlecenia
-- @NumerFaktury - numer faktury sprzedazy
-- @IdPracownika - ID pracownika dokonujacego sprzedazy
-- @IdPlatnosci - ID formy platnosci
-- @DataSprzedazy - data dokonania sprzedazy
-- @Rabat - wartosc rabatu
-- @IdProduktu - ID produktu sprzedawanego
-- @CenaSprzedazy - cena sprzedazy jednego produktu
-- @LiczbaSztuk - liczba sprzedanych produktow

```

```

-- @StawkaVat - wartość stawki VAT w procentach
--
--parametry wyjściowe/zwracane wartości: Brak
--
--
--Przykład użycia
--EXEC dbo.up_DodajSprzedaz 1, 'FV202205030001', 6, 2, '2022-05-03', 0.10, 7, 25.00, 2,
0.23
--
--Wynik działania
--Nowa sprzedaż zostanie dodana do bazy danych w tabeli 'tbl_Sprzedaz' i
'tbl_SzczegolySprzedazy'.
-----
-----

```

```

CREATE PROCEDURE dbo.up_DodajSprzedaz
@IdKontrahenta INT,
@NumerFaktury VARCHAR(30),
@IdPracownika INT,
@IdPlatnosci INT,
@DataSprzedazy DATETIME,
@Rabat DECIMAL(5,2),
@IdProduktu INT,
@CenaSprzedazy MONEY,
@LiczbaSztuk INT,
@StawkaVat DECIMAL(5,2)
AS
BEGIN
    BEGIN TRY
        DECLARE @IdSprzedazy INT

        INSERT INTO dbo.tbl_Sprzedaz (IdKontrahenta, NumerFaktury, IdPracownika,
IdPlatnosci, DataSprzedazy, Rabat)
        VALUES (@IdKontrahenta, @NumerFaktury, @IdPracownika, @IdPlatnosci,
@DataSprzedazy, @Rabat)

        SET @IdSprzedazy = SCOPE_IDENTITY()

        INSERT INTO dbo.tbl_SzczegolySprzedazy (IdSprzedazy, IdProduktu, CenaSprzedazy,
LiczbaSztuk, StawkaVat)
        VALUES (@IdSprzedazy, @IdProduktu, @CenaSprzedazy, @LiczbaSztuk, @StawkaVat)
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000)
        DECLARE @ErrorSeverity INT
        DECLARE @ErrorState INT

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE()

        RAISERROR('Wystąpił błąd podczas dodawania sprzedaży. Wiadomość błędu: %s',
@ErrorSeverity, @ErrorState, @ErrorMessage)
    END CATCH
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- up_ObnizCenyProduktow(@ProcentObnizki DECIMAL(5, 2))
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----

-- Procedura obniżająca ceny produktów o określony procent.
--
-- Parametry wejściowe:
-- @ProcentObnizki - procent obniżki cen produktów
--
-- Parametry wyjściowe/zwracane wartości: Brak
--
--
-- Przykład użycia:
-- EXEC up_ObnizCenyProduktow 10.00
--
-- Wynik działania:
-- Ceny cennikowe produktów zostaną obniżone o 10% w tabeli 'dbo.tbl_Produkt'.
-----

```

```

CREATE PROCEDURE up_ObnizCenyProduktow
    @ProcentObnizki DECIMAL(5, 2)
AS
BEGIN
    BEGIN TRY
        UPDATE dbo.tbl_Produkt
        SET CenaCennik = CenaCennik * (1 - (@ProcentObnizki / 100))
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000)
        DECLARE @ErrorSeverity INT
        DECLARE @ErrorState INT

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE()
        RAISERROR('Wystąpił błąd podczas obniżania cen produktów. Wiadomość błędu: %s',
            @ErrorSeverity, @ErrorState, @ErrorMessage)
    END CATCH
END
GO

```

```

-----
--- PROCEDURE DEFINITION
--- dbo.up_PodwyzkaCenProduktow(@ProcentPodwyzki DECIMAL(5,2))
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----

-- Procedura podwyższająca ceny produktów w bazie danych.
--
-- parametry wejściowe:
-- @ProcentPodwyzki - procentowa wartość podwyżki cen
--
-- parametry wyjściowe/zwracane wartości: Brak
--
--
-- Przykład użycia
-- EXEC dbo.up_PodwyzkaCenProduktow 5.00
--

```

```
-- Wynik działania
-- Ceny produktów zostaną podwyższone o 5% w tabeli 'tbl_Produkt'.
```

```
-----

CREATE PROCEDURE up_PodwyżkaCenProduktow
    @ProcentPodwyżki DECIMAL(5,2)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        UPDATE dbo.tbl_Produkt
        SET CenaCennik = CenaCennik * (1 + (@ProcentPodwyżki / 100));

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;

        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();

        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH;
END
GO
```

```
-----
--- PROCEDURE DEFINITION
--- dbo.up_DodajKontrahenta(@IdAdresu INT, @NazwaFirmy VARCHAR(30), @NIP VARCHAR(13),
--- @Imie VARCHAR(30), @Nazwisko VARCHAR(30), @PESEL VARCHAR(11), @TypKlienta BIT)
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----
```

```
--Procedura dodająca informacje o nowym kontrahencie do bazy danych.
```

```
--
--parametry wejściowe:
-- @IdAdresu - ID adresu kontrahenta
-- @NazwaFirmy - nazwa firmy kontrahenta
-- @NIP - numer NIP kontrahenta
-- @Imie - imię osoby kontaktowej
-- @Nazwisko - nazwisko osoby kontaktowej
-- @PESEL - numer PESEL osoby kontaktowej
-- @TypKlienta - typ klienta (0 - osoba fizyczna, 1 - firma)
```

```
--
--parametry wyjściowe/zwracane wartości: Brak
```

```
--
--Przykład użycia
--EXEC dbo.up_DodajKontrahenta 1, 'NazwaFirmy', '1234567890', 'Jan', 'Kowalski',
'12345678901', 1
```

```
--
```

```
--Wynik działania
--Nowy kontrahent zostanie dodany do bazy danych w tabeli 'tbl_Kontrahent'.
```

```
-----
CREATE PROCEDURE up_DodajKontrahenta
    @IdAdresu INT,
    @NazwaFirmy VARCHAR(30),
    @NIP VARCHAR(13),
    @Imie VARCHAR(30),
    @Nazwisko VARCHAR(30),
    @PESEL VARCHAR(11),
    @TypKlienta BIT
AS
BEGIN
    BEGIN TRY
        INSERT INTO tbl_Kontrahent (IdAdresu, NazwaFirmy, NIP, Imie, Nazwisko, PESEL,
TypKlienta)
        VALUES (@IdAdresu, @NazwaFirmy, @NIP, @Imie, @Nazwisko, @PESEL, @TypKlienta)
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END
GO
```

```
-----
--- PROCEDURE DEFINITION
--- dbo.up_EdytujKontrahenta(@IdKontrahenta INT, @IdAdresu INT, @NazwaFirmy VARCHAR(30),
@NIP VARCHAR(13), @Imie VARCHAR(30), - @Nazwisko VARCHAR(30), @PESEL VARCHAR(11),
@TypKlienta BIT)
--- CREATED BY: Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Procedura edytująca informacje o kontrahencie w bazie danych.
--
-- parametry wejściowe:
-- @IdKontrahenta - ID kontrahenta do edycji
-- @IdAdresu - ID adresu kontrahenta
-- @NazwaFirmy - nazwa firmy kontrahenta
-- @NIP - numer NIP kontrahenta
-- @Imie - imię kontrahenta (dla osób fizycznych)
-- @Nazwisko - nazwisko kontrahenta (dla osób fizycznych)
-- @PESEL - numer PESEL kontrahenta (dla osób fizycznych)
-- @TypKlienta - typ klienta (0 - osoba fizyczna, 1 - firma)
--
-- parametry wyjściowe/zwracane wartości: Brak
--
--
-- Przykład użycia:
-- EXEC dbo.up_EdytujKontrahenta 1, 2, 'Nazwa Firmy', '1234567890', NULL, NULL, NULL, 1
```



```
--  
-- Wynik działania:  
-- Informacje o kontrahencie zostaną zaktualizowane w tabeli 'tbl_Kontrahent' na podstawie  
podanych wartości.  
-----
```

```
CREATE PROCEDURE up_EdytujKontrahenta  
    @IdKontrahenta INT,  
    @IdAdresu INT,  
    @NazwaFirmy VARCHAR(30),  
    @NIP VARCHAR(13),  
    @Imie VARCHAR(30),  
    @Nazwisko VARCHAR(30),  
    @PESEL VARCHAR(11),  
    @TypKlienta BIT  
AS  
BEGIN  
    BEGIN TRY  
        BEGIN TRANSACTION;  
  
        UPDATE tbl_Kontrahent  
        SET IdAdresu = @IdAdresu,  
            NazwaFirmy = @NazwaFirmy,  
            NIP = @NIP,  
            Imie = @Imie,  
            Nazwisko = @Nazwisko,  
            PESEL = @PESEL,  
            TypKlienta = @TypKlienta  
        WHERE IdKontrahenta = @IdKontrahenta;  
  
        COMMIT TRANSACTION;  
    END TRY  
    BEGIN CATCH  
        IF @@TRANCOUNT > 0  
            ROLLBACK TRANSACTION;  
  
        DECLARE @ErrorMessage NVARCHAR(4000);  
        DECLARE @ErrorSeverity INT;  
        DECLARE @ErrorState INT;  
  
        SELECT  
            @ErrorMessage = ERROR_MESSAGE(),  
            @ErrorSeverity = ERROR_SEVERITY(),  
            @ErrorState = ERROR_STATE();  
        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);  
    END CATCH;  
END  
GO
```

```

-----
---  PROCEDURE DEFINITION
---  dbo.up_DodajAdres(@IdWojewodztwa INT, @Ulica VARCHAR(30), @KodPocztowy VARCHAR(6),
@Miasto VARCHAR(30))
---  CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Procedura dodająca informacje o nowym adresie do bazy danych.
--
--parametry wejściowe:
-- @IdWojewodztwa - ID województwa
-- @Ulica - nazwa ulicy
-- @KodPocztowy - kod pocztowy
-- @Miasto - nazwa miasta
--
--parametry wyjściowe/zwracane wartości: Brak
--
--/*
--Przykład użycia
--EXEC dbo.up_DodajAdres 1, 'Aleja Wojska Polskiego 1', '00-001', 'Warszawa'
--
--Wynik działania
--Nowy adres zostanie dodany do bazy danych w tabeli 'tbl_Adresy'.
-----

```

```

CREATE PROCEDURE up_DodajAdres
    @IdWojewodztwa INT,
    @Ulica VARCHAR(30),
    @KodPocztowy VARCHAR(6),
    @Miasto VARCHAR(30)
AS
BEGIN
    BEGIN TRY
        INSERT INTO tbl_Adresy (IdWojewodztwa, Ulica, KodPocztowy, Miasto)
        VALUES (@IdWojewodztwa, @Ulica, @KodPocztowy, @Miasto)
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END
GO

```

Wykaz utworzonych funkcji użytkownika

LP	Nazwa funkcji	Opis
1	uf_pobierz_nazwe_produkту	Funkcja zwraca nazwę produktu na podstawie podanego identyfikatora produktu.
2	uf_pobierz_adres_kontrahenta	Funkcja zwraca adres kontrahenta (ulicę, kod pocztowy i miasto) na podstawie podanego identyfikatora kontrahenta.
3	uf_pobierz_nazwe_dostawcy	Funkcja zwraca nazwę dostawcy na podstawie podanego identyfikatora dostawcy.
4	uf_sprawdz_typ_klienta	Funkcja zwraca typ klienta (osoba fizyczna lub firma) na podstawie podanego identyfikatora klienta.
5	uf_pobierz_typ_platnosci	Funkcja zwraca rodzaj płatności na podstawie podanego identyfikatora płatności.
6	uf_liczba_pracownikow_wg_stanowiska	Funkcja zwraca liczbę pracowników na podstawie podanego identyfikatora stanowiska.
7	uf_oblicz_łączną_sprzedaż_produkту	Funkcja zwraca łączną sprzedaż produktu na podstawie podanego identyfikatora produktu.
8	uf_pobierz_sprzedaz_dla_klienta	Funkcja zwraca tabelę z danymi o sprzedaży dla podanego identyfikatora klienta.
9	uf_liczba_dostawcow_według_województwa	Funkcja zwraca liczbę dostawców zlokalizowanych w podanym województwie na podstawie identyfikatora województwa.

Skrypt tworzenia funkcji definiowanych przez użytkownika

```
-----  
---  FUNCTION DEFINITION  
---  uf_pobierz_nazwe_produkту (@id_produkту INT)  
---  CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"  
-----  
--Funkcja zwraca nazwę produktu dla podanego identyfikatora produktu.  
--  
--parametry wejściowe:  
--@id_produkту - Identyfikator produktu, dla którego ma zostać zwrócona nazwa.  
--  
--parametry wyjściowe/zwracane wartości:  
--@nazwa_produkту - zwraca nazwę produktu.  
--  
--/*  
--Przykład użycia  
--SELECT dbo.uf_pobierz_nazwe_produkту(1)  
--  
--Wynik działania  
--Zwrócono wartość: 'DuluxWhite'  
-----
```

```

CREATE FUNCTION uf_pobierz_nazwe_produktu (@id_produktu INT)
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE @nazwa_produktu NVARCHAR(100)
    SELECT @nazwa_produktu = Nazwa
    FROM dbo.tbl_Produkt
    WHERE IdProduktu = @id_produktu

    RETURN @nazwa_produktu
END
GO

```

```

-----
--- FUNCTION DEFINITION
--- uf_pobierz_adres_kontrahenta (@idKontrahenta INT)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Funkcja zwraca adres kontrahenta o podanym ID.
--
--parametry wejściowe:
--@idKontrahenta - ID kontrahenta, dla którego ma zostać zwrócony adres
--
--parametry wyjściowe/zwracane wartości:
--@adresKontrahenta - adres kontrahenta w formacie "Ulica, Kod Pocztowy Miasto"
--
--
--Przykład użycia
--SELECT dbo.uf_pobierz_adres_kontrahenta(1)
--
--Wynik działania
--Zwrócono wartość: 'Mazowiecka 4/5, 00-001 Warszawa'
-----v

```

```

CREATE FUNCTION uf_pobierz_adres_kontrahenta (@idKontrahenta INT)
RETURNS NVARCHAR(500)
AS
BEGIN
    DECLARE @adresKontrahenta NVARCHAR(500)

    SELECT @adresKontrahenta = CONCAT(Ulica, ', ', KodPocztowy, ', ', Miasto)
    FROM dbo.tbl_Adresy
    WHERE IdAdresu = (SELECT IdAdresu FROM dbo.tbl_Kontrahent WHERE IdKontrahenta =
@idKontrahenta)

    RETURN @adresKontrahenta
END
GO

```

```

-----
--- FUNCTION DEFINITION
--- uf_pobierz_nazwe_dostawcy (@id_dostawcy INT)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Pobiera nazwę dostawcy na podstawie jego identyfikatora.
--
--parametry wejściowe:
--@id_dostawcy - identyfikator dostawcy, dla którego pobierana jest nazwa.
--
--parametry wyjściowe/zwracane wartości:
--@nazwa_dostawcy - nazwa dostawcy dla podanego identyfikatora.
--
--
--Przykład użycia
--SELECT dbo.uf_pobierz_nazwe_dostawcy(1)
--
--Wynik działania
--Zwrócono wartość: 'GeneralSpeditions'
-----

```

```

CREATE FUNCTION uf_pobierz_nazwe_dostawcy (@id_dostawcy INT)
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE @nazwa_dostawcy NVARCHAR(100)
    SELECT @nazwa_dostawcy = Nazwa
    FROM dbo.tbl_Dostawca
    WHERE IdDostawcy = @id_dostawcy

    RETURN @nazwa_dostawcy
END
GO

```

```

-----
--- FUNCTION DEFINITION
--- uf_sprawdz_typ_klienta (@id_klienta INT)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Sprawdza typ klienta o podanym IdKontrahenta
--
--parametry wejściowe:
--@id_klienta - Identyfikator klienta w tabeli dbo.tbl_Kontrahent
--
--parametry wyjściowe/zwracane wartości:
--@typ_klienta: 'os.fiz' dla klienta indywidualnego (TypKlienta = 1)
--              'Firma' dla klienta firmowego (TypKlienta = 0)
--
--/*
--Przykład użycia
--SELECT dbo.uf_sprawdz_typ_klienta(1)
--
--Wynik działania
--Zwrócono wartość: 'os.fiz'
-----

```

```

CREATE FUNCTION uf_sprawdz_typ_klienta (@id_klienta INT)
RETURNS NVARCHAR(20)
AS
BEGIN
    DECLARE @typ_klienta NVARCHAR(20)

    SELECT @typ_klienta = CASE
                                WHEN TypKlienta = 1 THEN 'os.fiz'
                                WHEN TypKlienta = 0 THEN 'Firma'
                            END
    FROM dbo.tbl_Kontrahent
    WHERE IdKontrahenta = @id_klienta

    RETURN @typ_klienta
END
GO

```

```

-----
-- FUNCTION DEFINITION
-- uf_pobierz_typ_platnosci (@id_platnosci INT)
-- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Funkcja ta służy do pobierania typu płatności na podstawie jej identyfikatora
--
--parametry wejściowe:
--@id_platnosci - identyfikator płatności
--
--parametry wyjściowe/zwracane wartości:
--@typ_platnosci - zwraca typ płatności
--
--
--Przykład użycia
--SELECT dbo.uf_pobierz_typ_platnosci(1)
--
--
--Wynik działania
--Zwrócono wartość: 'Gotówka'
-----

```

```

CREATE FUNCTION uf_pobierz_typ_platnosci (@id_platnosci INT)
RETURNS NVARCHAR(50)
AS
BEGIN
    DECLARE @typ_platnosci NVARCHAR(50)
    SELECT @typ_platnosci = RodzajPlatnosci
    FROM dbo.tbl_Platnosc
    WHERE IdPlatnosci = @id_platnosci

    RETURN @typ_platnosci
END
GO

```

```

-----
--- FUNCTION DEFINITION
--- uf_liczba_pracownikow_wg_stanowiska (@id_stanowiska INT)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Funkcja zwraca liczbę pracowników związanych z danym stanowiskiem.
--
--parametry wejściowe:
--@id_stanowiska - identyfikator stanowiska, dla którego chcemy poznać liczbę pracowników
--
--parametry wyjściowe/zwracane wartości:
--@liczba_pracownikow - liczba pracowników związanych z danym stanowiskiem
--
--
--Przykład użycia
--SELECT dbo.uf_liczba_pracownikow_wg_stanowiska(1)
--
--Wynik działania
--Zwrócono wartość: 12
-----

```

```

CREATE FUNCTION uf_liczba_pracownikow_wg_stanowiska (@id_stanowiska INT)
RETURNS INT
AS
BEGIN
    DECLARE @liczba_pracownikow INT
    SELECT @liczba_pracownikow = COUNT(*) FROM dbo.tbl_Pracownik WHERE IdStanowiska =
@id_stanowiska
    RETURN @liczba_pracownikow
END
GO

```

```

-----
--- FUNCTION DEFINITION
--- uf_oblicz_laczna_sprzedaz_produkту (@IdProdukту int)
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Funkcja ta oblicza łączną sprzedaż danego produktu na podstawie jego identyfikatora
(@IdProdukту).
--
--parametry wejściowe:
-- @IdProdukту - identyfikator produktu, dla którego chcemy obliczyć łączną sprzedaż
--
--parametry wyjściowe/zwracane wartości:
-- @LacznaSprzedaz - zwraca łączną sprzedaż produktu o podanym identyfikatorze
(@IdProdukту).
--
--
--Przykład użycia
--SELECT dbo.uf_oblicz_laczna_sprzedaz_produkту(1)
--
--Wynik działania
--Zwrócono wartość: 2565,94
-----

```

```

CREATE FUNCTION uf_oblicz_laczna_sprzedaz_produkту (@IdProdukту int)
RETURNS money
AS
BEGIN
    DECLARE @LacznaSprzedaz money
    SELECT @LacznaSprzedaz = SUM(CenaSprzedazy * LiczbaSztuk) FROM
    dbo.tbl_SzczegolySprzedazy WHERE IdProdukту = @IdProdukту
    RETURN @LacznaSprzedaz
END
GO

```

```

-----
-- FUNCTION DEFINITION
-- uf_pobierz_sprzedaz_dla_klienta (@IdKlienta int)
-- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Zwraca wszystkie sprzedaże dla danego klienta (o podanym IdKlienta) z tabeli
dbo.tbl_Sprzedaz
--
--parametry wejściowe:
-- @IdKlienta - id klienta, dla którego chcemy pobrać sprzedaż
--
--parametry wyjściowe/zwracane wartości:
--Zwracana jest tabela zawierająca wszystkie sprzedaże dla danego klienta (o podanym
IdKlienta)
--
--
--Przykład użycia
-- SELECT * FROM dbo.uf_pobierz_sprzedaz_dla_klienta(1)
--
--Wynik działania
--Zwrócono wartość: sprzedaże dla klienta o id = 1
-----

```

```

CREATE FUNCTION uf_pobierz_sprzedaz_dla_klienta (@IdKlienta int)
RETURNS TABLE
AS
RETURN (
    SELECT * FROM dbo.tbl_Sprzedaz WHERE IdKontrahenta = @IdKlienta
)
GO

```



```

-----
--- FUNCTION DEFINITION
--- uf_liczba_dostawcow_wedlug_wojewodztwa (@id_wojewodztwa INT)
--- CREATED BY:      Grupa "Hurtownia WYROBÓW Chemicznych"
-----

--Funkcja zwraca liczbę dostawców zarejestrowanych w danym województwie.
--
--parametry wejściowe:
-- @id_wojewodztwa - identyfikator województwa, dla którego ma zostać zwrócona liczba
dostawców.
--
--parametry wyjściowe/zwracane wartości:
-- @liczbaDostawcow - Zwracana jest liczba dostawców zarejestrowanych w danym
województwie.
--
--
--Przykład użycia
--SELECT dbo.uf_liczba_dostawcow_wedlug_wojewodztwa(1)
--
--Wynik działania
--Zwrócono wartość: 3
-----

CREATE FUNCTION uf_liczba_dostawcow_wedlug_wojewodztwa (@id_wojewodztwa INT)
RETURNS INT
AS
BEGIN
    DECLARE @liczbaDostawcow INT
    SELECT @liczbaDostawcow = COUNT(*) FROM dbo.tbl_Dostawca d JOIN dbo.tbl_Adresy a ON
d.IdAdresu = a.IdAdresu WHERE a.IdWojewodztwa = @id_wojewodztwa
    RETURN @liczbaDostawcow
END
GO

```

Wykaz utworzonych wyzwalaczy

LP	Nazwa wyzwalacza	Opis
1	trg_dodaj_liczbesztuk	Po zakupie aktualizuje liczbę sztuk zakupionego produktu w magazynie
2	trg_odejmij_liczbesztuk	Po sprzedaży aktualizuje liczbę sztuk produktu w magazynie
3	trg_informuj_o_stanie_magazynowym	Wyświetla informację o niskim stanie magazynowym produktu, jeśli liczba sztuk produktu spadnie poniżej 10.
4	trg_dodano_produkt	Wyświetla informację o dodaniu nowego produktu.
5	trg_dodano_pracownika	Wyświetla informację o dodaniu nowego pracownika.
6	trg_BlokujUsuniecieKontrahenta	Uniemożliwia usunięcie kontrahenta

Skrypt tworzenia wyzwalaczy

```
-----  
--- TRIGGER DEFINITION  
--- trg_dodaj_liczbesztuk  
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"  
-----  
-- Trigger wywoływany po wstawieniu nowych rekordów do tabeli dbo.tbl_ZakupSzczegoly.  
-- Aktualizuje pole LiczbaSztukMagazyn w tabeli dbo.tbl_Produkt, dodając wartość nowej  
-- liczby sztuk.  
-- Parametry wejściowe: Brak  
-- Parametry wyjściowe/zwracane wartości: Brak  
--  
--  
--Przykład użycia:  
--      INSERT INTO tbl_SzczegolyZakupu (IdZakupu, IdProduktu, CenaZakupu, LiczbaSztuk,  
--StawkaVat) VALUES (1, 3, 8.99, 5, 0.23);  
--Wynik działania  
--      dbo.tbl_Produkt zostanie zaktualizowane o wartość nowej liczby sztuk.  
-----
```

```

CREATE TRIGGER trg_dodaj_liczbesztuk
ON dbo.tbl_SzczegolyZakupu
AFTER INSERT
AS
BEGIN
    BEGIN TRY
        UPDATE dbo.tbl_Produkt
        SET LiczbaSztukMagazyn = LiczbaSztukMagazyn + inserted.LiczbaSztuk
        FROM inserted
        WHERE dbo.tbl_Produkt.IdProduktu = inserted.IdProduktu
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000)
        DECLARE @ErrorSeverity INT
        DECLARE @ErrorState INT

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE()
        RAISERROR('Wystąpił błąd podczas aktualizacji LiczbaSztukMagazyn: %s',
@ErrorSeverity, @ErrorState, @ErrorMessage)
    END CATCH
END
GO

```

```

-----
--- TRIGGER DEFINITION
--- trg_odejmij_liczbesztuk
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Trigger wywoływany po wstawieniu nowych rekordów do tabeli dbo.tbl_SzczegolySprzedazy.
-- Aktualizuje pole LiczbaSztukMagazyn w tabeli dbo.tbl_Produkt, odejmując wartość nowej
-- liczby sztuk.
-- Parametry wejściowe: Brak
-- Parametry wyjściowe/zwracane wartości: Brak
--
--Przykład użycia:
--     INSERT INTO tbl_SzczegolySprzedazy (IdSprzedazy, IdProduktu, CenaSprzedazy,
--     LiczbaSztuk, StawkaVat) VALUES (10, 1, 10.99, 2, 0.23);

--Wynik działania
--     dbo.tbl_Produkt zostanie zaktualizowane o wartość nowej liczby sztuk.

```

```

CREATE TRIGGER trg_odejmij_liczbesztuk
ON dbo.tbl_SzczegolySprzedazy
AFTER INSERT
AS
BEGIN
    BEGIN TRY
        UPDATE dbo.tbl_Produkt
        SET LiczbaSztukMagazyn = LiczbaSztukMagazyn - inserted.LiczbaSztuk
        FROM inserted
        WHERE dbo.tbl_Produkt.IdProduktu = inserted.IdProduktu;
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH;
END
GO

```

```

-----
--- TRIGGER DEFINITION
--- trg_informuj_o_stanie_magazynowym
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
-- Informuje o niskim stanie magazynowym produktu, gdy liczba sztuk wynosi poniżej 10.
-- parametry wejściowe: brak
-- parametry wyjściowe/zwracane wartości: Brak
--
--
-- Przykład użycia
--     UPDATE dbo.tbl_Produkt SET LiczbaSztukMagazyn = 5 WHERE IdProduktu = 1
--
-- Wynik działania:
--     Uwaga! Niski stan magazynowy produktu Produkt1.
-----

```

```

CREATE TRIGGER trg_informuj_o_stanie_magazynowym
ON dbo.tbl_Produkt
AFTER UPDATE
AS
BEGIN
    IF UPDATE(LiczbaSztukMagazyn)
    BEGIN
        IF (SELECT LiczbaSztukMagazyn FROM inserted) < 10
        BEGIN
            BEGIN TRY
                DECLARE @IdProduktu INT
                SET @IdProduktu = (SELECT IdProduktu FROM inserted)
                DECLARE @NazwaProduktu VARCHAR(50)
                SET @NazwaProduktu = (SELECT p.Nazwa FROM dbo.tbl_Produkt p WHERE
IdProduktu = @IdProduktu)
                PRINT 'Uwaga! Niski stan magazynowy produktu ' + @NazwaProduktu + '.'
            END TRY
            BEGIN CATCH
                PRINT 'Wystąpił błąd podczas wykonywania triggera.'
                PRINT 'Opis błędu: ' + ERROR_MESSAGE()
            END CATCH
        END
    END
END
GO

```

```

-----
--- TRIGGER DEFINITION
--- trg_dodano_produkt
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----

-- Wyświetla informację o nazwie dodanego produktu.
-- Trigger wywoływany po dodaniu nowego produktu do tabeli dbo.tbl_Produkt.
--
-- parametry wejściowe: brak
--
-- parametry wyjściowe/zwracane wartości: brak
--
--
-- Przykład użycia
--     INSERT INTO tbl_Produkt (IdKategorii, Nazwa, LiczbaSztukMagazyn, CenaCennik)
--     VALUES (1, 'Nowy produkt', 10, 9.99);
--
--
-- Wynik działania
--     Dodano produkt: Nowy produkt.
-----

```

```

CREATE TRIGGER trg_dodano_produkt
ON dbo.tbl_Produkt
AFTER INSERT
AS
BEGIN
    BEGIN TRY
        DECLARE @NazwaProduktu VARCHAR(50)
        SET @NazwaProduktu = (SELECT Nazwa FROM inserted)

        PRINT 'Dodano produkt: ' + @NazwaProduktu + '.'

    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END
GO

```

```

-----
--- TRIGGER DEFINITION
--- trg_dodano_pracownika
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-----
--Trigger wywoływany po dodaniu nowego pracownika do tabeli tbl_Pracownik.
--Wyświetla informację o dodanym pracowniku.
--
-- parametry wejściowe: brak
--
-- parametry wyjściowe/zwracane wartości: brak
--
--
--Przykład użycia
--    INSERT INTO tbl_Pracownik (IdStanowiska, Imie, Nazwisko, Email, NumerTelefonu)
--    VALUES (1, 'Jan', 'Kowalski', 'jkwalski@example.com', '123456789');
--
--
--Wynik działania
--    Dodano nowego pracownika: Jan Kowalski.
-----

```

```

CREATE TRIGGER trg_dodano_pracownika
ON dbo.tbl_Pracownik
AFTER INSERT
AS
BEGIN
    BEGIN TRY
        DECLARE @Imie VARCHAR(50), @Nazwisko VARCHAR(50)
        SELECT @Imie = Imie, @Nazwisko = Nazwisko FROM inserted

        PRINT 'Dodano nowego pracownika: ' + @Imie + ' ' + @Nazwisko + ' .'
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000)
        SET @ErrorMessage = ERROR_MESSAGE()

        PRINT 'Wystąpił błąd podczas dodawania pracownika:'
        PRINT @ErrorMessage
    END CATCH
END
GO

```

```

-----
--- TRIGGER DEFINITION
--- trg_BlokujUsuniecieKontrahenta
--- CREATED BY:      Grupa "Hurtownia Wyrobów Chemicznych"
-- Trigger blokujący usunięcie kontrahenta.
-- Uniemożliwia usunięcie rekordu z tabeli dbo.tbl_Kontrahent.
-- Parametry wejściowe: Brak

-- Parametry wyjściowe/zwracane wartości: Brak
--
--Przykład użycia:
-- DELETE FROM dbo.tbl_Kontrahent WHERE IdKontrahenta = 1
--Wynik działania
--Usunięcie kontrahenta jest niedozwolone. Transakcja zostanie cofnięta.
-----

```

```

CREATE TRIGGER trg_BlokujUsuniecieKontrahenta
ON tbl_Kontrahent
INSTEAD OF DELETE
AS
BEGIN
    BEGIN TRY
        IF EXISTS (SELECT * FROM deleted)
        BEGIN
            RAISERROR('Usunięcie kontrahenta jest niedozwolone.', 16, 1)
        END
        ELSE
        BEGIN
            DELETE FROM tbl_Kontrahent
            WHERE EXISTS (SELECT * FROM deleted WHERE deleted.IdKontrahenta =
tbl_Kontrahent.IdKontrahenta)
        END
    END TRY
    BEGIN CATCH
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        SELECT @ErrorMessage = ERROR_MESSAGE(),
               @ErrorSeverity = ERROR_SEVERITY(),
               @ErrorState = ERROR_STATE();

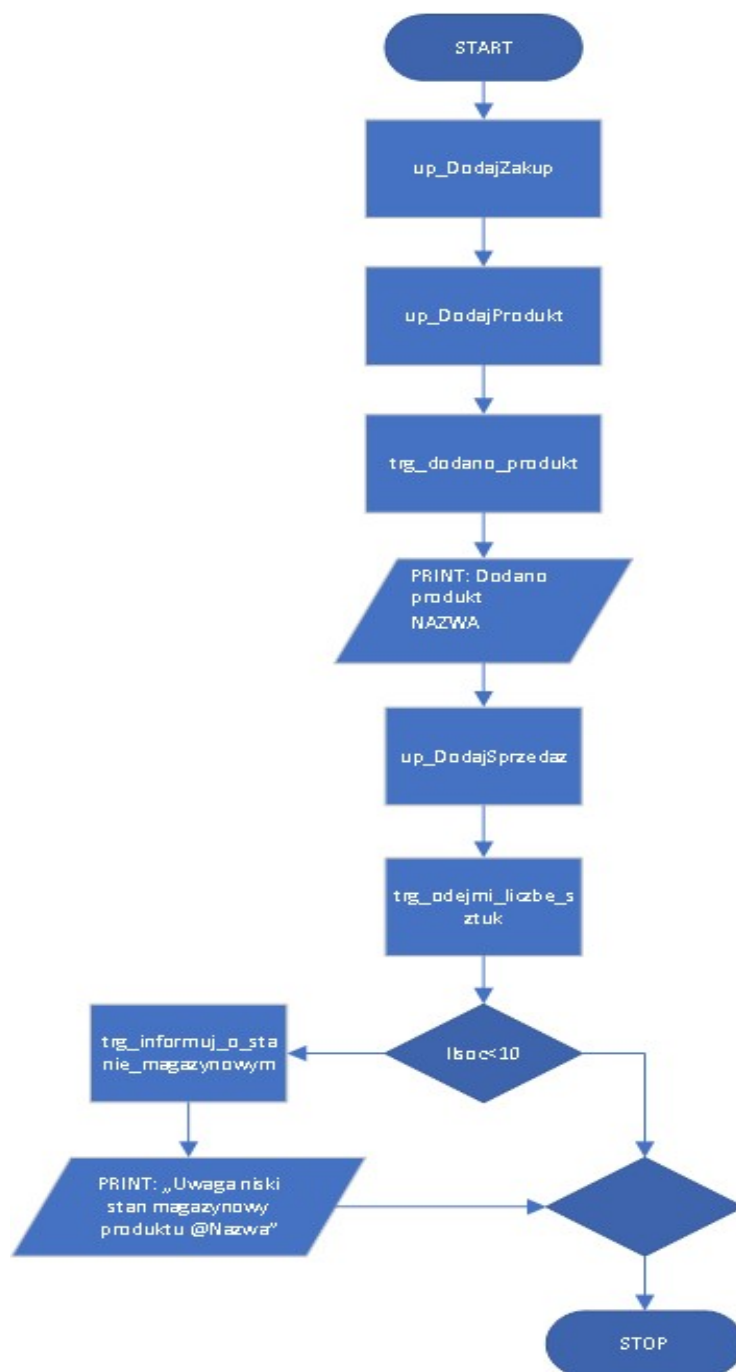
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

        ROLLBACK TRANSACTION;
    END CATCH;
END;
GO

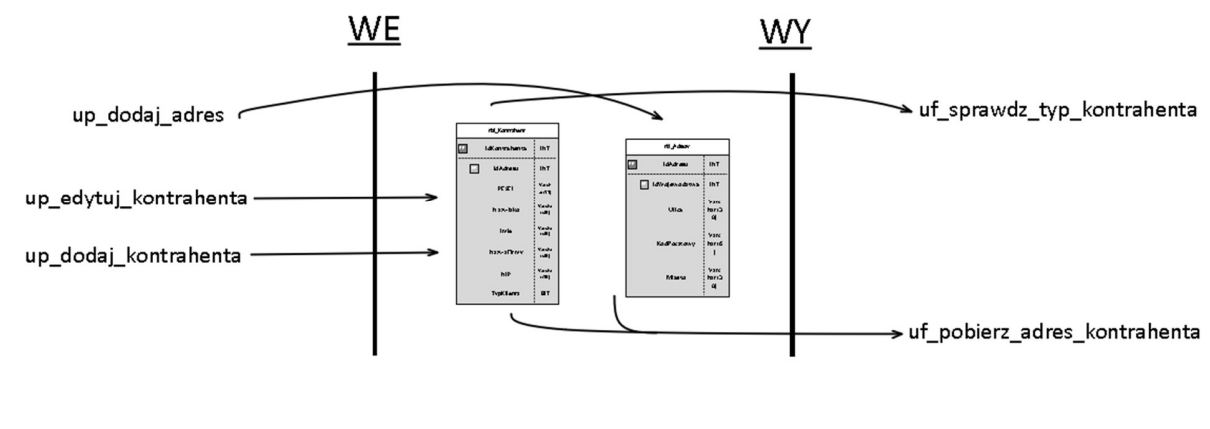
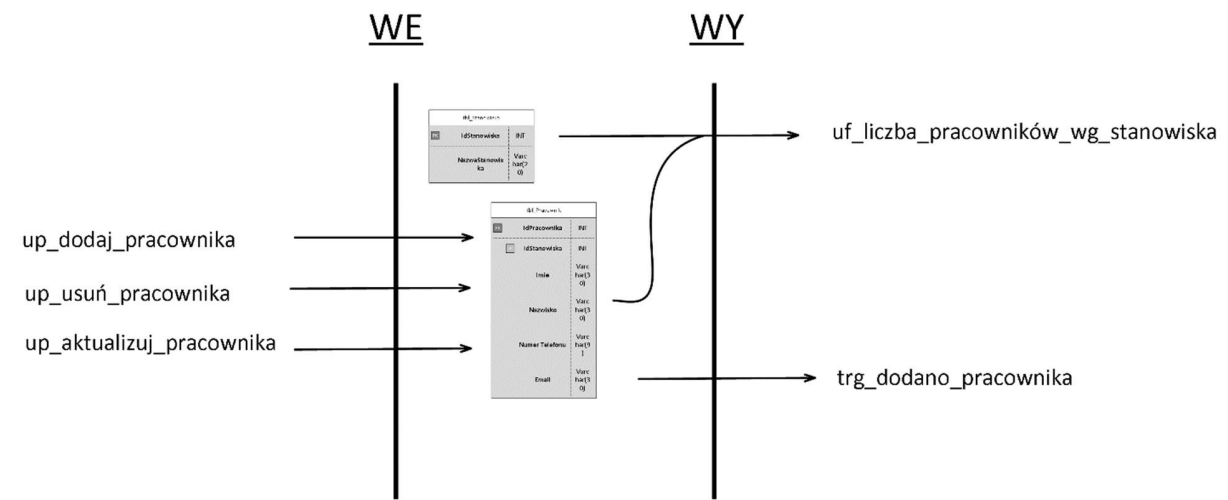
```


Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia

Schemat blokowy



Warstwa wejścia/wyjścia



WE

WY

tbl_Dostawca	
<input checked="" type="checkbox"/> PK	IdDostawcy INT
<input type="checkbox"/>	MAdressa INT
Nazwa	
NIP	
	Varc n=3 01
	Varc n=11 01

uf_pobierz_nazwe_dostawcy

uf_liczba_dostawców_wg_województwa

up_dodaj_zakup

WE

WY

tbl_ZestawienieZakupow	
<input checked="" type="checkbox"/> PK	IdZakupow INT
<input checked="" type="checkbox"/>	IdProduktow INT
CenaZakupow	
LiczbaSztuk	
StanekWlasci	
	VarChar(3)


trg_dodaj_liczbe_sztuk

tbl_Produkt	
<input checked="" type="checkbox"/> PK	IdProduktow INT
<input type="checkbox"/>	IdKategorie INT
Nazwa	
LiczbaSztukMagazynu	
CenaCenaKup	
	VarChar(3)

tbl_Zakup	
<input checked="" type="checkbox"/> PK	IdDostawcy INT
<input checked="" type="checkbox"/>	IdZakupow INT
<input type="checkbox"/>	IdPlacowilka INT
<input type="checkbox"/>	IdPlacoweci INT
NazwaFaktury	
DataZakupow	
Rabat	
Cryfowanie	
	VarChar(3)
	BIT

WE

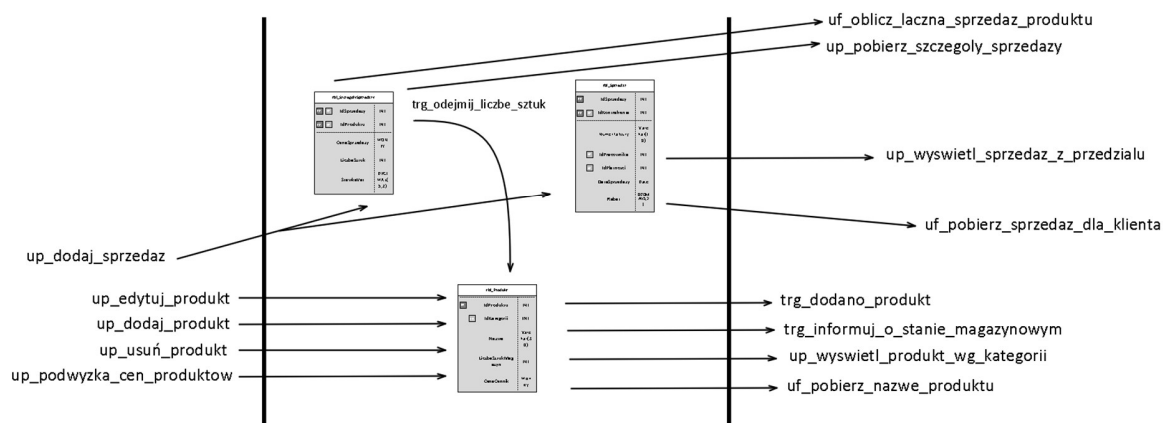
WY

tbl_Platnosci	
 IdPlatnosci	INT
RodzajPlatnosci	VARCHAR(10)

uf_pobierz_typ_platnosci

WE

WY



Wnioski i podsumowanie:

Po przeprowadzeniu projektu implementacji bazy danych "Hurtownia wyrobów chemicznych" można wyciągnąć kilka ważnych wniosków.

Po pierwsze, dobrze zaprojektowana baza danych to klucz do efektywnego zarządzania informacjami w przedsiębiorstwie. Dzięki odpowiedniemu modelowaniu danych, możliwe jest składowanie i przetwarzanie informacji w sposób, który umożliwia łatwe wyszukiwanie i analizowanie danych.

Po drugie, implementacja bazy danych wymaga czasu i dokładności. Ważne jest, aby zrozumieć potrzeby przedsiębiorstwa oraz specyfikę branży, w której działa. W trakcie projektowania bazy danych należy uwzględnić różne scenariusze i przypadki użycia, aby zapewnić optymalne działanie bazy danych.

Po trzecie, projektowanie i implementacja bazy danych to proces dynamiczny, który wymaga ciągłego monitorowania i dostosowywania do zmieniających się potrzeb przedsiębiorstwa. Wprowadzanie nowych funkcjonalności oraz rozwijanie bazy danych powinno być realizowane w sposób kontrolowany i planowy.

Podsumowując, projekt implementacji bazy danych "Hurtownia wyrobów chemicznych" był ważnym krokiem w procesie efektywnego zarządzania informacjami w przedsiębiorstwie. Przeprowadzenie tego projektu pozwoliło na poznanie specyfiki branży oraz potrzeb przedsiębiorstwa. Umożliwiło także dostosowanie bazy danych do wymagań klientów oraz stworzenie narzędzia umożliwiającego łatwe zarządzanie informacjami w firmie.

Wykaz/spis ilustracji

Ilustracja 1: Zdjęcie magazynu
Strona: 1

Ilustracja 2: Model fizyczny
Strona: 6

Ilustracja 3: Schemat blokowy
Strona: 37

Ilustracja 4, 5: Warstwa wejścia/wyjścia
Strona: 38

Ilustracja 6, 7: Warstwa wejścia/wyjścia
Strona: 39

Ilustracja 8, 9: Warstwa wejścia/wyjścia
Strona: 40

Wykaz/spis tabel

Tabela 1: Platnosc

Strona: 8

Tabela 2: Stanowisko

Strona: 8

Tabela 3: Kategoria

Strona: 8

Tabela 4: Wojewodztwo

Strona: 9

Tabela 5: Adresy

Strona: 9

Tabela 6: Pracownik

Strona: 10

Tabela 7: Kontrahent

Strona: 10

Tabela 8: Sprzedaz

Strona: 11

Tabela 9: Dostawca

Strona: 11

Tabela 10: Zakup

Strona: 12

Tabela 11: Produkt

Strona: 12

Tabela 12: SzczegolyZakupu

Strona: 13

Tabela 13: SzczegolySprzedazy

Strona: 13