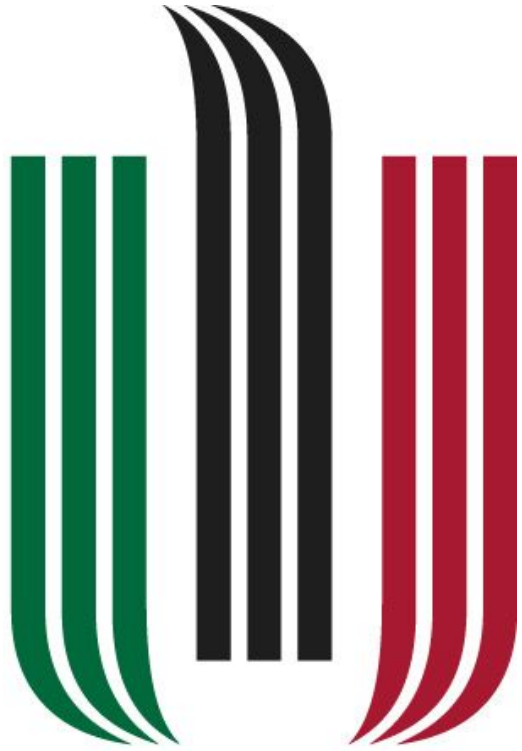


AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Podstawy Baz Danych

Projekt Konferencja

Wykonali:	Kacper Janda i Jakub Kołoczek
Wydział:	WIEiT
Kierunek:	Informatyka
Semestr:	3

Spis treści

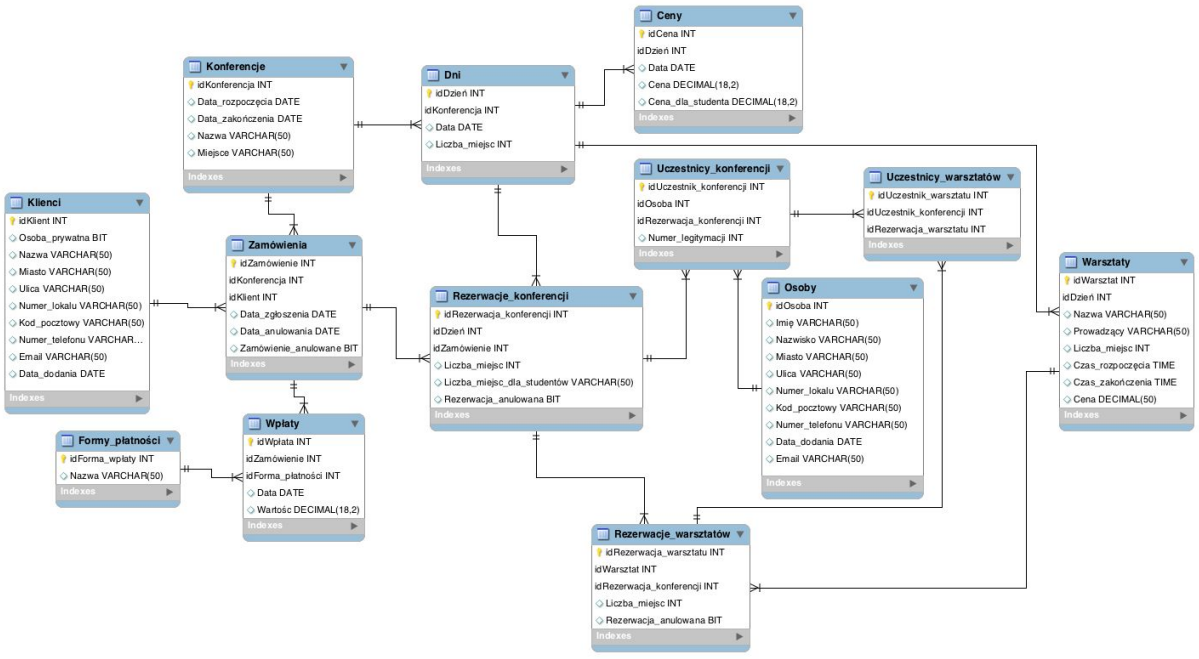
1. Opis problemu

Firma organizuje konferencje, które mogą być jedno lub kilkudniowe. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są osoby (firma nie musi podawać od razu przy rejestracji listy uczestników może zarezerwować odpowiednią liczbę miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić). Dla konferencji kilkudniowych, klienci mogą rejestrować na dowolne z tych dni, dowolną liczbę osób. Klient może zmienić liczbę osób na rezerwacji, lub całkiem ją anulować (do 2 tygodni przed konferencją).

2. Wymagania projektowe

- zwroty są realizowane za pomocą wpłat o ujemnej wartości
- klient otrzymuje zwrot w przypadku anulowania zamówienia bądź nadpłaty
- blokada możliwości składania zamówienia i rezerwacji na dwa tygodnie przed wydarzeniem
- analogicznie w przypadku dodawania progów cenowych
- zakładamy, że mogą istnieć konferencje i warsztaty o tej samej nazwie
- w rezerwacjach konferencji rozdzielamy miejsca ulgowe dla studentów od zwykłych, pełnopłatnych

3. Schemat bazy



4. Tabele

4.1. Ceny

```
CREATE TABLE Ceny
(
  idCena            INT IDENTITY PRIMARY KEY,
  idDzień           INT NOT NULL
                    CONSTRAINT FK_Ceny_Dni REFERENCES Dni,
  Data              DATE  NOT NULL,
  Cena              DECIMAL(18, 2) NOT NULL
                    CONSTRAINT Cena_większa_od_0 CHECK ([Cena] >= 0),
  Cena_dla_studenta DECIMAL(18, 2) NOT NULL
                    CONSTRAINT CK_Ceny_dla_studenta CHECK ([Cena_dla_studenta] >= 0),
                    CONSTRAINT Cena_dla_studenta_mniejsza CHECK ([Cena_dla_studenta] <= [Cena])
)
GO
```

4.2. Dni

```
CREATE TABLE Dni
(
  idDzień       INT IDENTITY PRIMARY KEY,
  idKonferencja INT  NOT NULL
                CONSTRAINT FK_Dni_Konferencje REFERENCES Konferencje,
  Data          DATE NOT NULL,
  Liczba_miejsc INT  NOT NULL
                CONSTRAINT d.Liczba_miejsc_większa_od_0 CHECK ([Liczba_miejsc] > 0)
)
GO
```

4.3. Formy_płatności

```
CREATE TABLE Formy_płatności
(
  idForma_płatności INT IDENTITY
                    CONSTRAINT PK_Formy_płatności PRIMARY KEY,
  Nazwa             VARCHAR(50) NOT NULL
)
GO
```

4.4. Klienci

```
CREATE TABLE Klienci
(
  idKlient       INT IDENTITY PRIMARY KEY,
  Osoba_prywatna BIT         NOT NULL,
  Nazwa          VARCHAR(50) NOT NULL,
  Miasto         VARCHAR(50) NOT NULL,
  Ulica          VARCHAR(50) NOT NULL,
  Numer_lokalu   VARCHAR(50) NOT NULL,
  Kod_pocztowy   VARCHAR(50) NOT NULL,
  Numer_telefonu VARCHAR(50) NOT NULL
                 CONSTRAINT CK_Klienci CHECK ([Numer_telefonu] LIKE '[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
  Email          VARCHAR(50) NOT NULL
                 CONSTRAINT Email CHECK ([Email] LIKE '%@%.%'),
  Data_dodania   DATE        NOT NULL
)
GO
CREATE INDEX indeks_nazwa_klienci ON Klienci (Nazwa)
GO
CREATE INDEX indeks_email_klienci ON Klienci (Email)
GO
```

4.5. Konferencje

```
CREATE TABLE Konferencje
(
  idKonferencja    INT IDENTITY PRIMARY KEY,
  Data_rozpoczęcia DATE NOT NULL,
  Data_zakończenia DATE        NOT NULL,
  Nazwa            VARCHAR(50) NOT NULL,
  Miejsce          VARCHAR(50) NOT NULL,
                   CONSTRAINT CK_Data_zakończenia CHECK ([Data_zakończenia] >= [Data_rozpoczęcia])
)
GO
CREATE INDEX indeks_data_rozpoczęcia_konferencje ON Konferencje (Data_rozpoczęcia)
GO
```

4.6. Osoby

```
CREATE TABLE Osoby
(
  idOsoba        INT IDENTITY PRIMARY KEY,
  Imię           VARCHAR(50) NOT NULL,
  Nazwisko       VARCHAR(50) NOT NULL,
  Miasto         VARCHAR(50) NOT NULL,
  Ulica          VARCHAR(50) NOT NULL,
  Numer_telefonu VARCHAR(50) NOT NULL
                 CONSTRAINT Numer_telefonu CHECK ([Numer_telefonu] LIKE '[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
  Data_dodania   DATE        NOT NULL,
  Email          VARCHAR(50)
                 CONSTRAINT CK_Osoby CHECK ([Email] LIKE '%@%.%'),
  Kod_pocztowy   VARCHAR(50) NOT NULL,
  Numer_lokalu   VARCHAR(50) NOT NULL
)
GO
CREATE INDEX indeks_nazwisko_osoby ON Osoby (Nazwisko)
GO
CREATE INDEX indeks_email_osoby ON Osoby (Email)
GO
```

4.7. Rezerwacje_konferencji

```
CREATE TABLE Rezerwacje_konferencji
(
  idRezerwacja_konferencji    INT IDENTITY
                              CONSTRAINT PK__Rezerwac__8D2920C677F5D907 PRIMARY KEY,
  idDzień                     INT NOT NULL
                              CONSTRAINT FK_Rezerwacje_konferencji_Dni REFERENCES Dni,
  idZamówienie                INT NOT NULL
                              CONSTRAINT FK_Rezerwacje_konferencji_Zamówienia REFERENCES Zamówienia,
  Liczba_miejsc               INT NOT NULL,
  Liczba_miejsc_dla_studentów INT NOT NULL,
  Rezerwacja_anulowana        BIT NOT NULL,
                              CONSTRAINT rk.Liczba_miejsc_większa_od_0 CHECK (([Liczba_miejsc]+[liczba_miejsc_dla_studentów])>0)
)
GO
```

4.8. Rezerwacje_warsztatów

```
CREATE TABLE Rezerwacje_warsztatów
(
  idRezerwacja_warsztatu   INT IDENTITY PRIMARY KEY,
  idWarsztat               INT NOT NULL
                           CONSTRAINT FK_Rezerwacje_warsztatów_Warsztaty REFERENCES Warsztaty,
  idRezerwacja_konferencji INT NOT NULL
                           CONSTRAINT FK_Rezerwacje_warsztatów_Rezerwacje_konferencji REFERENCES Rezerwacje_konferencji,
  Liczba_miejsc            INT NOT NULL
                           CONSTRAINT rw.Liczba_miejsc_większa_od_0 CHECK ([Liczba_miejsc] > 0),
  Rezerwacja_anulowana     BIT NOT NULL
)
GO
```

4.9. Uczestnicy_konferencji

```
CREATE TABLE Uczestnicy_konferencji
(
  idUczestnik_konferencji  INT IDENTITY PRIMARY KEY,
  idOsoba                  INT NOT NULL
                           CONSTRAINT FK_Uczestnicy_konferencji_Osoby REFERENCES Osoby,
  idRezerwacja_konferencji INT NOT NULL
                           CONSTRAINT FK_Uczestnicy_konferencji_Rezerwacje_konferencji REFERENCES Rezerwacje_konferencji,
  Numer_legitymacji        INT
                           CONSTRAINT Numer_legitymacji CHECK ([Numer_legitymacji] IS NULL
                           OR CONVERT([varchar]50, [Numer_legitymacji]) LIKE '[1-9][0-9][0-9][0-9][0-9][0-9]')
)
GO
```

4.10. Uczestnicy_warsztatów

```
CREATE TABLE Uczestnicy_warsztatów
(
  idUczestnik_warsztatu   INT IDENTITY PRIMARY KEY,
  idRezerwacja_warsztatu  INT NOT NULL
                          CONSTRAINT FK_Uczestnicy_warsztatów_Rezerwacje_warsztatów REFERENCES Rezerwacje_warsztatów,
  idUczestnik_konferencji INT NOT NULL
                          CONSTRAINT FK_Uczestnicy_warsztatów_Uczestnicy_konferencji REFERENCES Uczestnicy_konferencji
)
GO
```

4.11. Warsztaty

```
CREATE TABLE Warsztaty
(
  idWarsztat       INT IDENTITY
                   CONSTRAINT PK__Warsztat__4BA2FB8A4A9F450F PRIMARY KEY,
  idDzień          INT NOT NULL
                   CONSTRAINT FK_Warsztaty_Dni
                   REFERENCES Dni,
  Nazwa            VARCHAR(50) NOT NULL,
  Prowadzący       VARCHAR(50) NOT NULL,
  Liczba_miejsc    INT NOT NULL
                   CONSTRAINT w.Liczba_miejsc_większa_od_0 CHECK ([Liczba_miejsc] > 0),
  Czas_rozpoczęcia TIME NOT NULL,
  Czas_zakończenia TIME NOT NULL,
  Cena             DECIMAL(18, 2) NOT NULL,
                   CONSTRAINT w.CzasRozpoczęcia_mniejszy_niż_zakończenia CHECK ([Czas_rozpoczęcia] < [Czas_zakończenia])
)
GO
CREATE INDEX indeks_iddzień_warsztaty ON Warsztaty (idDzień)
GO
```

4.12. Wpłaty

```
CREATE TABLE Wpłaty
(
  idWpłata          INT IDENTITY PRIMARY KEY,
  idZamówienie      INT NOT NULL
                    CONSTRAINT FK_Wpłaty_Zamówienia REFERENCES Zamówienia,
  Wartość           DECIMAL(18, 2) NOT NULL,
  Data              DATE           NOT NULL,
  idForma_płatności INT            NOT NULL
                    CONSTRAINT FK_Wpłaty_Formy_płatności REFERENCES Formy_płatności
)
GO
CREATE INDEX indeks_data_wpłaty ON Wpłaty (Data)
GO
```

4.13. Zamówiena

```
CREATE TABLE Zamówienia
(
  idZamówienie         INT IDENTITY
                       CONSTRAINT PK__Zamówien__048742FADD3B0509 PRIMARY KEY,
  idKlient             INT  NOT NULL
                       CONSTRAINT FK_Zamówienia_Klienci REFERENCES Klienci,
  idKonferencja        INT
                       CONSTRAINT FK_Zamówienia_Konferencje REFERENCES Konferencje,
  Data_zgłoszenia      DATE NOT NULL,
  Data_anulowania      DATE,
  Zamówienie_anulowane BIT NOT NULL,
                       CONSTRAINT Data_anulowania_późniejsza_niż_zgłoszenia
                       CHECK ([Data_anulowania] IS NULL OR [Data_anulowania] >= [Data_zgłoszenia]),
                       CONSTRAINT Data_anulowania_a_zamówienie_anulowane
                       CHECK ([Zamówienie_anulowane] = 1 AND [Data_anulowania] IS NOT NULL
                       OR [Zamówienie_anulowane] = 0 AND [Data_anulowania] IS NULL)
)
GO
CREATE INDEX indeks_data_zgłoszenia_zamówienia ON Zamówienia (Data_zgłoszenia)
GO
```

5. Triggery

5.1. Dodanie_progu_cenowego

- uniemożliwia dodanie progu cenowego po zamknięciu zamówień (na 2 tygodnie przed konferencją)

```
ALTER TRIGGER Dodanie_progu_cenowego
ON Ceny
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT 'TAK'
                              FROM inserted i
                                        INNER JOIN Dni d ON i.idDzień = d.idDzień
                              WHERE i.data > DATEADD(DAY, -14, d.Data))
                    BEGIN
                              RAISERROR ('Nie można już dodać progu cenowego', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.2. Dodanie_dnia

- uniemożliwia dodania po raz drugi tego samego dnia tej samej konferencji

```
ALTER TRIGGER Dodanie_dnia
ON Dni
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT d.Data
                              FROM inserted i
                                        INNER JOIN Dni d ON i.data = d.Data AND i.idKonferencja = d.idKonferencja
                              GROUP BY d.data
                              HAVING count(*) > 1)
                    BEGIN
                              RAISERROR ('Nie można dodać tego dnia, ponieważ on już istnieje.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.3. Poza_dniami_konferencji

- uniemożliwia dodanie dnia będącego poza czasem trwania konferencji

```
CREATE TRIGGER [dbo].[Poza_dniami_konferencji]
          ON [dbo].[Dni]
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                             SELECT i.idDzień
                              FROM inserted i
                                        INNER JOIN Konferencje k ON k.idKonferencja = i.idKonferencja
                              WHERE i.Data < k.Data_rozpoczęcia OR i.Data > k.Data_zakończenia)
                    BEGIN
                              RAISERROR ('Dzień nie zawiera sie w przedziale czasowym konferencji', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.4. Zmniejszenie_liczby_miejsc_dnia_konferencji

- uniemożliwia zmniejszenie liczby miejsc na dniu konferencji w przypadku gdy zmniejszona liczba byłaby mniejsza niż liczba zapisanych na ten dzień osób

```
ALTER TRIGGER .[dbo].[Zmniejszenie_liczby_miejsc_dnia_konferencji]
ON [dbo].[Dni]
AFTER UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM inserted i
                                        INNER JOIN Rezerwacje_konferencji rk ON rk.idDzień = i.idDzień
                              GROUP BY rk.idDzień, i.Liczba_miejsc
                              HAVING sum(rk.Liczba_miejsc + rk.Liczba_miejsc_dla_studentów) > i.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można zmniejszyć ilości miejsc, bo braknie dla zarezerwowanych.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.5. Limit_osób_na_dzień_konferencji

- uniemożliwia dodanie na dzień konferencji więcej osób niż ilość miejsc

```
ALTER TRIGGER [dbo].[Limit_osób_na_dzień_konferencji]
ON [dbo].[Rezerwacje_konferencji]
AFTER INSERT, UPDATE
AS
BEGIN
          IF exists(
                              SELECT *
                              FROM Rezerwacje_konferencji rk
                                        INNER JOIN inserted i ON i.idDzień = rk.idDzień
                                        INNER JOIN Dni d ON d.idDzień = rk.idDzień
                              WHERE rk.Rezerwacja_anulowana = 0
                              GROUP BY d.idDzień, d.Liczba_miejsc
                              HAVING sum(rk.Liczba_miejsc + rk.Liczba_miejsc_dla_studentów) > d.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można dodać więcej osób na konferencję', 16, 1);
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.6. Rezerwacja_konferencji_przy_anulowanym_zamówieniu

- uniemożliwia zarezerwowanie konferencji w przypadku gdy zamówienie zostało anulowane

```
CREATE TRIGGER [dbo].[Rezerwacja_konferencji_przy_anulowanym_zamówieniu]
          ON [dbo].[Rezerwacje_konferencji]
AFTER INSERT
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT rk.idRezerwacja_konferencji
                              FROM inserted i
                                        INNER JOIN Rezerwacje_konferencji rk ON rk.idZamówienie = i.idZamówienie
                                        INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
                              WHERE rk.rezerwacja_anulowana = 1)
                    BEGIN
                             RAISERROR ('Nie można dodać rezerwacji konferencji, ponieważ zamówienie zostało anulowane.',16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.7. Rezerwacja_konferencji_zgodna_z_zamówieniem

- uniemożliwia zarezerwowanie dnia konferencji, który nie zawiera się w dniach konferencji z zamówienia

```
ALTER TRIGGER [dbo].[Rezerwacja_konferencji_zgodna_z_zamówieniem]
ON [dbo].[Rezerwacje_konferencji]
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;
	IF NOT exists(
				SELECT DISTINCT
					d.idDzień,
					d.idKonferencja,
					d.data
				FROM inserted i
					INNER JOIN Dni d ON i.idDzień = d.idDzień
					INNER JOIN Konferencje k ON k.idKonferencja = d.idKonferencja
					INNER JOIN Zamówienia z ON k.idKonferencja = z.idKonferencja
				WHERE z.idKonferencja = d.idKonferencja)
		BEGIN
				RAISERROR ('Nie można zarezerwować dnia konferencji, który nie wchodzi w skład konferencji z
			zamówienia.', 16, 1)
			ROLLBACK TRANSACTION
		END
END
GO
```

5.8. Zmniejszenie_liczby_miejsc_rezerwacji_dnia_konferencji

- uniemożliwia zmniejszenie liczby miejsc w rezerwacji dnia konferencji w przypadku gdy zmniejszona liczba byłaby mniejsza niż liczba uczestników związanych z tą rezerwacją

```
CREATE TRIGGER Zmniejszenie_liczby_miejsc_rezerwacji_dnia_konferencji
	ON Rezerwacje_konferencji
AFTER UPDATE
AS
BEGIN
	SET NOCOUNT ON;
	IF exists(
				SELECT *
				FROM inserted i
					INNER JOIN Rezerwacje_konferencji rk
						ON rk.idRezerwacja_konferencji = i.idRezerwacja_konferencji
					INNER JOIN Uczestnicy_konferencji uk
						ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
				GROUP BY rk.idRezerwacja_konferencji, i.Liczba_miejsc, i.Liczba_miejsc_dla_studentów,
					uk.Numer_legitymacji
				HAVING (count(*) > i.Liczba_miejsc_dla_studentów AND uk.Numer_legitymacji IS NOT NULL)
					OR  (count(*) > i.Liczba_miejsc AND uk.Numer_legitymacji IS NULL))
		BEGIN
				RAISERROR ('Nie można zmniejszyć liczby miejsc w rezerwacji konferencji ponieważ zapisałeś już
					więcej osób.', 16, 1)
				ROLLBACK TRANSACTION
		END
END
GO
```

5.9. Liczba_rezerwowanych_miejsc_na_warsztat

- uniemożliwia zarezerwowanie większej liczby miejsc na warsztat, niż zarezerwowało się na dzień konferencji

```
CREATE TRIGGER [dbo].[Liczba_rezerwowanych_miejsc_na_warsztat]
          ON [dbo].[Rezerwacje_warsztatów]
AFTER INSERT, UPDATE
AS
BEGIN
                    SET NOCOUNT ON;
                    IF exists(
                                        SELECT *
                                        FROM inserted i
                                                  INNER JOIN Rezerwacje_konferencji rk
                                                      ON rk.idRezerwacja_konferencji = i.idRezerwacja_konferencji
                                        WHERE (rk.Liczba_miejsc_dla_studentów + rk.Liczba_miejsc) < i.Liczba_miejsc)
                               BEGIN
                                        RAISERROR ('Chcesz zarezerwować więcej miejsc na warsztat niż na konferencję.', 16, 1)
                                        ROLLBACK TRANSACTION
                               END
END
GO
```

5.10. Limit_osób_na_warsztat

- uniemożliwia zapisanie większej liczby miejsc na warsztat niż liczba miejsc

```
ALTER TRIGGER Limit_osób_na_warsztat
          ON Rezerwacje_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          IF exists(SELECT *
                                        FROM inserted i
                                                  JOIN warsztaty w ON w.idWarsztat = i.idWarsztat
                                                  JOIN Rezerwacje_warsztatów rw ON w.idWarsztat = rw.idWarsztat
                                        WHERE rw.Rezerwacja_anulowana = 0
                                        GROUP BY rw.idWarsztat, w.Liczba_miejsc
                                        HAVING sum(rw.Liczba_miejsc) > w.Liczba_miejsc)
                               BEGIN
                                        RAISERROR ('Nie można zapisać więcej osób na ten warsztat.', 16, 1);
                                        ROLLBACK TRANSACTION;
                               END
END
GO
```

5.11. Rezerwacja_warsztatu_przy_anulowanym_zamówieniu

- uniemożliwia rezerwację warsztatu przy anulowanym zamówieniu

```
CREATE TRIGGER [dbo].[Rezerwacja_warsztatu_przy_anulowanym_zamówieniu]
          ON [dbo].[Rezerwacje_warsztatów]
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                                        SELECT i.idRezerwacja_warsztatu
                                        FROM inserted i
                                                  INNER JOIN Rezerwacje_konferencji rk
                                                      ON rk.idRezerwacja_konferencji = i.idRezerwacja_konferencji
                                        WHERE rk.rezerwacja_anulowana = 1)
                               BEGIN
                                        RAISERROR ('Nie można dodać rezerwacji warsztatu, ponieważ rezerwacja konferencji
                                                  została anulowana.', 16, 1)
                                        ROLLBACK TRANSACTION
                               END
          END
GO
```

5.12. Rezerwacja_warsztatu_zgodna_z_rezerwacją_konferencji

- uniemożliwia zarezerwowanie warsztatu, który nie wchodzi w skład konferencji z rezerwacji konferencji związanej z rezerwowanym warsztatem

```
CREATE TRIGGER [dbo].[Rezerwacja_warsztatu_zgodna_z_rezerwacją_konferencji]
          ON [dbo].[Rezerwacje_warsztatów]
AFTER INSERT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(
                              SELECT DISTINCT i.idRezerwacja_warsztatu
                              FROM inserted i
                                        INNER JOIN warsztaty w ON w.idWarsztat = i.idWarsztat
                                        INNER JOIN rezerwacje_konferencji rk
                                           ON rk.idRezerwacja_konferencji = i.idRezerwacja_konferencji
                              WHERE rk.idDzień = w.idDzień)
                    BEGIN
                              RAISERROR ('Nie można zarezerwować warsztatu, który nie wchodzi w skład dnia konferencji z
                                        rezerwacji konferencji.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.13. Zmniejszenie_liczby_miejsc_rezerwacji_warsztatu

- uniemożliwia zmniejszenie liczby miejsc w rezerwacji warsztatu w przypadku, gdy zmniejszona liczba byłaby mniejsza niż liczba osób zapisanych na ten warsztat z tej rezerwacji

```
CREATE TRIGGER [dbo].[Zmniejszenie_liczby_miejsc_rezerwacji_warsztatu]
          ON [dbo].[Rezerwacje_warsztatów]
AFTER UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM inserted i
                                        INNER JOIN Uczestnicy_warsztatów uw
                                             ON uw.idRezerwacja_warsztatu = i.idRezerwacja_warsztatu
                              GROUP BY i.idRezerwacja_warsztatu, i.Liczba_miejsc
                              HAVING COUNT(*) > i.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można zmniejszyć liczby miejsc w rezerwacji warsztatu ponieważ zapisałeś już
                                        więcej osób.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.14. Ostatnie_wykorzystane_normalne_miejsca_z_rezerwacji_konferencji

- informuje o wykorzystaniu ostatniego miejsca z rezerwacji konferencji

```
CREATE TRIGGER Ostatnie_wykorzystane_normalne_miejsca_z_rezerwacji_konferencji
          ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM Uczestnicy_konferencji uk
                                        INNER JOIN inserted i
                                             ON i.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                             ON rk.idrezerwacja_konferencji = uk.idRezerwacja_konferencji
                              WHERE uk.Numer_legitymacji IS NULL AND rk.Rezerwacja_anulowana = 0
                              GROUP BY uk.idRezerwacja_konferencji, rk.Liczba_miejsc
                              HAVING count(*) = rk.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Właśnie wykorzystałeś ostatnie normalne miejsce z danej rezerwacji.', 1, 1)
                    END
END
GO
```

5.15. Ostatnie_wykorzystane_studenckie_miejsca_z_rezerwacji_konferencji

- informuje o wykorzystaniu ostatniego studenckiego miejsca z rezerwacji konferencji

```
ALTER TRIGGER Ostatnie_wykorzystane_studenckie_miejsca_z_rezerwacji_konferencji
         ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                             SELECT *
                             FROM Uczestnicy_konferencji uk
                                        INNER JOIN inserted i
                                             ON i.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                             ON rk.idrezerwacja_konferencji = uk.idRezerwacja_konferencji
                              WHERE uk.Numer_legitymacji IS NOT NULL AND rk.Rezerwacja_anulowana = 0
                              GROUP BY uk.idRezerwacja_konferencji, rk.Liczba_miejsc_dla_studentów
                              HAVING count(*) = rk.Liczba_miejsc_dla_studentów)
                    BEGIN
                             RAISERROR ('Właśnie wykorzystałeś ostatnie studenckie miejsce z danej rezerwacji.', 1, 1)
                    END
END
GO
```

5.16. Podwójne_wykorzystanie_numeru_legitymacji

- informuje o sytuacji, gdy dwaj uczestnicy konferencji pochodzący z tej samej rezerwacji konferencji mają ten sam numer legitymacji

```
ALTER TRIGGER Podwójne_wykorzystanie_numeru_legitymacji
          ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
      SET NOCOUNT ON;
      IF exists(
                              SELECT count(*)
                              FROM inserted i
                                        INNER JOIN Uczestnicy_konferencji uk
                                                ON i.Numer_legitymacji = uk.Numer_legitymacji
                                        INNER JOIN rezerwacje_konferencji rk
                                                ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
                              HAVING count(*) > 2)
                        BEGIN
                              RAISERROR ('Ktoś korzystał już z tego numeru albumu, czy jest on na pewno Twój?', 16, 1);
                        END
END
GO
```

5.17. Podwójny_zapis_na_dzień

- uniemożliwia zapisanie danej osoby z rezerwacji konferencji dwukrotnie na tej sam dzień konferencji

```
CREATE TRIGGER [dbo].[Podwójny_zapis_na_dzień]
          ON [dbo].[Uczestnicy_konferencji]
AFTER INSERT, UPDATE
AS
BEGIN
      SET NOCOUNT ON;
      IF exists(
                               SELECT *
                               FROM Uczestnicy_konferencji uk
                                         INNER JOIN inserted i ON i.idOsoba = uk.idOsoba
                               GROUP BY uk.idRezerwacja_konferencji, uk.idOsoba
                               HAVING count(uk.idOsoba) > 1)
                     BEGIN
                               RAISERROR ('Nie można zapisać danej osoby więcej niż raz', 16, 1)
                               ROLLBACK TRANSACTION
                     END
END
GO
```

5.18. Wykorzystane_normalne_miejsca_z_rezerwacji_konferencji

- uniemożliwia zapisanie osoby na dzień konferencji w przypadku, gdy nie ma już miejsc normalnych, a dana osoba nie jest studentem

```
CREATE TRIGGER Wykorzystane_normalne_miejsca_z_rezerwacji_konferencji
          ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM Uczestnicy_konferencji uk
                                        INNER JOIN inserted i
                                              ON i.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                              ON rk.idrezerwacja_konferencji = uk.idRezerwacja_konferencji
                              WHERE uk.Numer_legitymacji IS NULL AND rk.Rezerwacja_anulowana = 0
                              GROUP BY uk.idRezerwacja_konferencji, rk.Liczba_miejsc
                              HAVING count(*) > rk.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można zapisać osoby na konferencję, ponieważ wszystkie normalne miejsca dla
                                        danego zamówienia zostały wykorzystane.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.19. Wykorzystane_studenckie_miejsca_z_rezerwacji_konferencji

- uniemożliwia wpisanie studenta na dzień konferencji w przypadku, gdy nie ma już miejsc studenckich

```
 TRIGGER Wykorzystane_studenckie_miejsca_z_rezerwacji_konferencji
          ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM Uczestnicy_konferencji uk
                                        INNER JOIN inserted i
                                              ON i.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                              ON rk.idrezerwacja_konferencji = uk.idRezerwacja_konferencji
                              WHERE uk.Numer_legitymacji IS NOT NULL AND rk.Rezerwacja_anulowana = 0
                              GROUP BY uk.idRezerwacja_konferencji, rk.Liczba_miejsc_dla_studentów
                              HAVING count(*) > rk.Liczba_miejsc_dla_studentów)
                    BEGIN
                              RAISERROR ('Nie można zapisać osoby na konferencję, ponieważ wszystkie studenckie miejsca dla
                                        danego zamówienia zostały wykorzystane.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.20. Zapis_uczestnika_dnia_konferencji_przy_anulowanej_rezerwacji

- uniemożliwia dodanie uczestnika konferencji w przypadku anulowanej rezerwacji dnia konferencji

```
CREATE TRIGGER Zapis_uczestnika_dnia_konferencji_przy_anulowanej_rezerwacji
          ON Uczestnicy_konferencji
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT uk.idUczestnik_konferencji
                              FROM inserted i
                                        INNER JOIN Uczestnicy_konferencji uk
                                              ON i.idUczestnik_konferencji = uk.idUczestnik_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                              ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                              WHERE rk.rezerwacja_anulowana = 1)
                    BEGIN
                              RAISERROR ('Nie można dodać osoby do dnia konferencji, ponieważ rezerwacja konferencji została
                                        anulowana.', 16, 1)
                              ROLLBACK TRANSACTION
                     END
END
GO
```

5.21. Czy_uczestnik_dnia_konferencji_przy_zapisie_na_warsztat

- uniemożliwia dodanie uczestnika konferencji w przypadku, gdy nie bierze on udziału w danym dniu konferencji

```
ALTER TRIGGER Czy_uczestnik_dnia_konferencji_przy_zapisie_na_warsztat
         ON Uczestnicy_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                               SELECT
                                        rk.iddzień,
                                        uk.idUczestnik_konferencji,
                                        uk.idOsoba,
                                        rk.idRezerwacja_konferencji,
                                        rw.idRezerwacja_konferencji,
                                        rw.idRezerwacja_warsztatu,
                                        w.idWarsztat,
                                        w.idDzień
                              FROM inserted i
                                        INNER JOIN uczestnicy_konferencji uk
                                               ON i.idUczestnik_konferencji = uk.idUczestnik_konferencji
                                        INNER JOIN Rezerwacje_konferencji rk
                                               ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
                                        INNER JOIN rezerwacje_warsztatów rw
                                               ON rk.idRezerwacja_konferencji = rw.idRezerwacja_konferencji
                                        INNER JOIN warsztaty w
                                               ON rw.idWarsztat = w.idWarsztat
                              WHERE w.idDzień = rk.idDzień AND rw.idRezerwacja_warsztatu = i.idRezerwacja_warsztatu)
                    BEGIN
                              RAISERROR ('Nie można dodać uczestnika warsztatu, który nie bierze udziału w danym dniu
                                        konferencji.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.22. Nachodzenie_się_warsztatów

- uniemożliwia dodanie uczestnika na dwa warsztaty nachodzące na siebie

```
ALTER TRIGGER Nachodzenie_się_warsztatów
          ON Uczestnicy_warsztatów
AFTER UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM inserted i
                                        INNER JOIN Rezerwacje_warsztatów rw1
                                               ON i.idRezerwacja_warsztatu = rw1.idRezerwacja_warsztatu
                                        INNER JOIN Warsztaty w1 ON rw1.idWarsztat = w1.idWarsztat
                                        INNER JOIN Warsztaty w2 ON w1.idDzień = w2.idDzień
                                        INNER JOIN Rezerwacje_warsztatów rw2 ON w2.idWarsztat = rw2.idWarsztat
                                        INNER JOIN Uczestnicy_warsztatów uw
                                               ON rw2.idRezerwacja_warsztatu = uw2.idRezerwacja_warsztatu
                              WHERE (w1.Czas_rozpoczęcia < w2.Czas_zakończenia AND w1.Czas_zakończenia > w2.Czas_rozpoczęcia))
                    BEGIN
                              RAISERROR ('Ten uczestnik bierze już udział w warsztatach w tym czasie.', 16, 1)
                              ROLLBACK TRANSACTION
                     END
END
GO
```

5.23. Ostatnie_wykorzystane_miejsce_z_rezerwacji_warsztatu

- informuje o wykorzystaniu ostatniego miejsca w rezerwacji warsztatu

```
ALTER TRIGGER Ostatnie_wykorzystane_miejsce_z_rezerwacji_warsztatu
          ON Uczestnicy_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT uw.idRezerwacja_warsztatu
                              FROM Uczestnicy_warsztatów uw
                                        INNER JOIN inserted i
                                              ON i.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
                                        INNER JOIN Rezerwacje_warsztatów rw
                                              ON uw.idRezerwacja_warsztatu = rw.idRezerwacja_warsztatu
                              WHERE rw.Rezerwacja_anulowana = 0
                              GROUP BY uw.idRezerwacja_warsztatu, rw.Liczba_miejsc
                              HAVING count(*) = rw.Liczba_miejsc)
                     BEGIN
                              RAISERROR ('Właśnie wykorzystałeś ostatnie miejsce z danej rezerwacji.', 1, 1)
                     END
END
GO
```

5.24. Podwójny_zapis_na_warsztat

- uniemożliwia zapisanie tej samej osoby dwukrotnie na ten sam warsztat

```
CREATE TRIGGER Podwójny_zapis_na_warsztat
          ON Uczestnicy_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT
                                        w.idWarsztat,
                                        uk.idOsoba
                              FROM Uczestnicy_warsztatów uw
                                        INNER JOIN Uczestnicy_konferencji uk
                                               ON uk.idUczestnik_konferencji = uw.idUczestnik_konferencji
                                        INNER JOIN Rezerwacje_warsztatów rw
                                               ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
                                        INNER JOIN Warsztaty w ON w.idWarsztat = rw.idWarsztat
                              GROUP BY w.idWarsztat, uk.idOsoba
                              HAVING count(*) > 1)
                    BEGIN
                              RAISERROR ('Dana osoba jest już zapisana na ten warsztat', 16, 1);
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.25. Wykorzystane_miejsca_z_rezerwacji_warsztatu

- uniemożliwia dodanie osoby na warsztat po wykorzystaniu wszystkich miejsc z rezerwacji

```
ALTER TRIGGER Wykorzystane_miejsca_z_rezerwacji_warsztatu
          ON Uczestnicy_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT uw.idRezerwacja_warsztatu
                              FROM Uczestnicy_warsztatów uw
                                        INNER JOIN inserted i
                                               ON i.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
                                        INNER JOIN Rezerwacje_warsztatów rw
                                               ON uw.idRezerwacja_warsztatu = rw.idRezerwacja_warsztatu
                              WHERE rw.Rezerwacja_anulowana = 0
                              GROUP BY uw.idRezerwacja_warsztatu, rw.Liczba_miejsc
                              HAVING count(*) > rw.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można zapisać osoby na warsztat, ponieważ wszystkie miejsca dla danego zamówienia
                                        zostały wykorzystane.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.26. Zapis_uczestnika_warsztatu_przy_anulowanej_rezerwacji

- uniemożliwia zapisanie osoby na warsztat, gdy rezerwacji warsztatu jest anulowana

```
CREATE TRIGGER Zapis_uczestnika_warsztatu_przy_anulowanej_rezerwacji
          ON Uczestnicy_warsztatów
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT uw.idUczestnik_warsztatu
                              FROM inserted i
                                        INNER JOIN Uczestnicy_warsztatów uw
                                              ON i.idUczestnik_warsztatu = uw.idUczestnik_warsztatu
                                        INNER JOIN Rezerwacje_warsztatów rw
                                              ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
                              WHERE rw.rezerwacja_anulowana = 1)
                    BEGIN
                              RAISERROR ('Nie można dodać osoby do warsztatu, ponieważ rezerwacja warsztatu została
                                        anulowana.', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.27. Zmniejszenie_liczby_miejsc_warsztatu

- uniemożliwia zmniejszenie liczby miejsc warsztatu w przypadku, gdy zmniejszona liczba byłaby mniejsza niż liczba osób zapisanych na ten warsztat

```
CREATE TRIGGER Zmniejszenie_liczby_miejsc_warsztatu
          ON Warsztaty
AFTER UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM inserted i
                                        INNER JOIN Rezerwacje_warsztatów rw ON rw.idWarsztat = i.idWarsztat
                              GROUP BY rw.idWarsztat, i.Liczba_miejsc
                              HAVING sum(rw.Liczba_miejsc) > i.Liczba_miejsc)
                    BEGIN
                              RAISERROR ('Nie można zmniejszyć ilości miejsc', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.28. Opłata_za_anulowane_zamówienie

- uniemożliwia opłacenie anulowanego zamówienia

```
CREATE TRIGGER Opłata_za_anulowane_zamówienie
          ON Wpłaty
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT *
                              FROM Zamówienia z
                                        INNER JOIN inserted i ON i.idZamówienie = z.idZamówienie
                              WHERE z.Zamówienie_anulowane = 1)
                    BEGIN
                              RAISERROR ('Nie można opłacić anulowanego zamówienia', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.29. Wpłacono_za_dużo

- informuje użytkownika o zbyt wysokiej wpłacie oraz zwraca nadwyżkę

```
ALTER TRIGGER [dbo].[Wpłacono_za_dużo]
          ON [dbo].[Wpłaty]
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT z.idZamówienie
                              FROM dbo.Zamówienia AS z
                                        INNER JOIN inserted i ON i.idZamówienie = z.idZamówienie
                              WHERE (
                                        SELECT ile
                                         FROM Ile_zostało_do_zapłaty(z.idZamówienie)) < 0)
          BEGIN
                    DECLARE @idZamówienie INT = (SELECT TOP 1 idZamówienie
                                                  FROM inserted)

                    INSERT INTO Wpłaty (idZamówienie, Wartość, Data, idForma_płatności)
                    VALUES (@idZamówienie, (SELECT ile
                                             FROM Ile_zostało_do_zapłaty(@idZamówienie)), getdate(), 4)
                    RAISERROR ('Wpłaciłeś za dużo, zrobiliśmy Ci wypłatę.', 1, 1)
          END
END
GO
```

5.30. Anulowanie_po_zakończeniu

- uniemożliwia anulowanie zamówienia po zakończeniu okresu przed konferencją (po rozpoczęciu konferencji)

```
CREATE TRIGGER Anulowanie_po_zakończeniu
          ON Zamówienia
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(           SELECT *
                                 FROM inserted i
                                 WHERE (i.Data_anulowania IS NOT NULL)
                                      AND i.Data_anulowania >=
                                                                (SELECT k.Data_rozpoczęcia
                                                           FROM konferencje k
                                                                     WHERE k.idKonferencja = i.idKonferencja)))
                    BEGIN
                              RAISERROR ('Nie można anulować zamówienia po rozpoczęciu konferencji', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

5.31. Zbyt_późne_zgłoszenie

- uniemożliwia dodanie zamówienia na 14 dni przed konferencją

```
CREATE TRIGGER .Zbyt_późne_zgłoszenie
          ON Zamówienia
AFTER INSERT, UPDATE
AS
BEGIN
          SET NOCOUNT ON;
          IF exists(
                              SELECT 'TAK'
                              FROM inserted i
                                        INNER JOIN dbo.Konferencje k ON k.idKonferencja = i.idKonferencja
                              WHERE i.Data_zgłoszenia > DATEADD(DAY, -14, k.Data_rozpoczęcia))
                    BEGIN
                              RAISERROR ('Nie można już dodać zamówienia na tę konferencję', 16, 1)
                              ROLLBACK TRANSACTION
                    END
END
GO
```

6. Procedury

6.1. Anuluj_jeśli_nieopłacone

```
ALTER PROCEDURE Anuluj_jeśli_nieopłacone
AS
  BEGIN
    SET NOCOUNT ON;
    DECLARE @Ile_zamówień_do_anulowania INT = (SELECT count(*)
                                               FROM Zamówienia AS z
                                               WHERE (datediff(DAY, z.Data_zgłoszenia, getdate()) > 7)
                                                     AND z.Data_anulowania IS NULL
                                                     AND ((SELECT Ile
                                                           FROM Ile_zostało_do_zapłaty(z.idZamówienie)) > 0))
    DECLARE @Licznik INT = 0
    WHILE (@Licznik < @Ile_zamówień_do_anulowania)
      BEGIN
        DECLARE @idZamówienia INT = (SELECT idZamówienie
                                     FROM (SELECT ROW_NUMBER() OVER (ORDER BY idZamówienie ASC ) AS wiersz,
                                                  idZamówienie
                                           FROM Zamówienia z
                                           WHERE (datediff(DAY, z.Data_zgłoszenia, getdate()) > 7)
                                             AND z.Data_anulowania IS NULL
                                             AND ((SELECT Ile
                                                   FROM Ile_zostało_do_zapłaty(z.idZamówienie)) > 0)) AS chrzan
                                     WHERE wiersz = 1)
        EXEC Anuluj_zamówienie @idZamówienia;
        SET @Licznik = @Licznik + 1
      END
END
GO
```

6.2. Anuluj_zamówienie

```
CREATE PROCEDURE [dbo].[Anuluj_zamówienie]
                    @idZamówienia INT
AS
BEGIN
          SET NOCOUNT ON;

                    UPDATE Zamówienia
                    SET Zamówienie_anulowane = 1
                    WHERE idZamówienie = @idZamówienia

                    UPDATE Zamówienia
                    SET Data_anulowania = getdate()
                    WHERE idZamówienie = @idZamówienia

                    UPDATE Rezerwacje_konferencji
                    SET Rezerwacja_anulowana = 1
                    WHERE idZamówienie = @idZamówienia

                    UPDATE Rezerwacje_warsztatów
                    SET Rezerwacja_anulowana = 1
                    WHERE idRezerwacja_konferencji IN (SELECT rk.idRezerwacja_konferencji
                                                       FROM Rezerwacje_konferencji rk
                                                       WHERE rk.idZamówienie = @idZamówienia)
END
GO
```

6.3. Dodaj_cenę

```
CREATE PROCEDURE [dbo].[Dodaj_cenę]
                    @idDzień           INT,
                    @Data              DATE,
                    @Cena              DECIMAL(18, 2),
                    @Cena_dla_studenta DECIMAL(18, 2)
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dni
                     WHERE idDzień = @idDzień))
            BEGIN
                    INSERT INTO Ceny (idDzień, Data, Cena, Cena_dla_studenta)
                    VALUES (@idDzień, @Data, @Cena, @Cena_dla_studenta)
            END
          ELSE
            BEGIN
                    RAISERROR ('Dzień o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
            END
END
GO
```

6.4. Dodaj_formę_płatności

```
CREATE PROCEDURE Dodaj_formę_płatności
                    @Nazwa VARCHAR(50)
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                    FROM dbo.Formy_płatności fp
                     WHERE fp.Nazwa = @Nazwa))
            BEGIN
                    RAISERROR ('Forma płatności o takiej nazwie już istnieje', 16, 1)
                    ROLLBACK TRANSACTION
            END
          ELSE
            BEGIN
                    INSERT INTO Formy_płatności (Nazwa)
                    VALUES (@Nazwa)
            END
END
GO
```

6.5. Dodaj_klienta

```
ALTER PROCEDURE Dodaj_klienta (
  @Osoba_prywatna BIT,
  @Nazwa          VARCHAR(50),
  @Miasto         VARCHAR(50),
  @Ulica          VARCHAR(50),
  @Numer_lokalu   VARCHAR(50),
  @Kod_pocztowy   VARCHAR(50),
  @Numer_telefonu VARCHAR(50),
  @Email          VARCHAR(50)
) AS
  SET NOCOUNT ON;
  IF exists(SELECT *
            FROM Klienci k
            WHERE k.Email = @Email)
      BEGIN
            RAISERROR ('Klient o podanym emailu już istnieje w bazie.', 16, 1)
            ROLLBACK TRANSACTION
      END
  ELSE
      BEGIN
             INSERT INTO klienci (Osoba_prywatna, Nazwa, Miasto, Ulica, Numer_lokalu, Kod_pocztowy, Numer_telefonu, Email)
             VALUES (@Osoba_prywatna, @Nazwa, @Miasto, @Ulica, @Numer_lokalu, @Kod_pocztowy, @Numer_telefonu, @Email)
      END
GO
```

6.6. Dodaj_konferencję

```
ALTER PROCEDURE [dbo].[Dodaj_konferencję]

                    @Data_rozpoczęcia DATE,
                    @Ilość_dni        INT,
                    @Nazwa            VARCHAR(50),
                    @Miejsce          VARCHAR(50),
                    @Liczba_miejsc    INT
AS
BEGIN
          SET NOCOUNT ON;

                    DECLARE @idKonferencja AS INT
                    DECLARE @Data_zakończenia AS DATE = dateadd(DAY, @Ilość_dni - 1, @Data_rozpoczęcia)
                         INSERT INTO Konferencje (Data_rozpoczęcia, Data_zakończenia, Nazwa, Miejsce)
                         VALUES (@Data_rozpoczęcia, @Data_zakończenia, @Nazwa, @Miejsce)
                    SELECT @idKonferencja = SCOPE_IDENTITY();
                   DECLARE @cnt DATE = @Data_rozpoczęcia
                    DECLARE @end DATE = @Data_zakończenia
                         WHILE CAST(@cnt AS DATE) <= CAST(@end AS DATE)
                               BEGIN
                                         INSERT INTO Dni (idKonferencja, [Data], Liczba_miejsc)
                                         VALUES (@idKonferencja, @cnt, @Liczba_miejsc)
                                         SET @cnt = dateadd(DAY, 1, @cnt);
                               END;
END
GO
```

6.7. Dodaj_osobę

```
ALTER PROCEDURE [dbo].[Dodaj_osobę] (
  @Imię           VARCHAR(50),
  @Nazwisko       VARCHAR(50),
  @Miasto         VARCHAR(50),
  @Ulica          VARCHAR(50),
  @Kod_pocztowy   VARCHAR(50),
  @Numer_lokalu   VARCHAR(50),
  @Numer_telefonu VARCHAR(50),
  @Email          VARCHAR(50)
) AS

  SET NOCOUNT ON;

  IF exists(SELECT *
            FROM Osoby o
            WHERE o.Email = @Email)
    BEGIN
      RAISERROR ('Osoba o podanym emailu już istnieje w bazie.', 16, 1)
      ROLLBACK TRANSACTION
    END
  ELSE
    BEGIN
      INSERT INTO osoby (Imię, Nazwisko, Miasto, Ulica, Numer_telefonu, Data_dodania, Email, Kod_pocztowy, Numer_lokalu)
      VALUES (@Imię, @Nazwisko, @Miasto, @Ulica, @Numer_telefonu, GETDATE(), @Email, @Kod_pocztowy, @Numer_lokalu)
    END
GO
```

6.8. Dodaj_uczestnika_konferencji

```
CREATE PROCEDURE [dbo].[Dodaj_uczestnika_konferencji]
                    @idOsoba                  INT,
                    @idRezerwacja_konferencji INT,
                    @Numer_legitymacji        INT = NULL
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dbo.Osoby
                     WHERE idOsoba = @idOsoba))
            BEGIN
                    IF (EXISTS(SELECT *
                               FROM dbo.Rezerwacje_konferencji
                               WHERE idRezerwacja_konferencji = @idRezerwacja_konferencji))
                       BEGIN
                              INSERT INTO Uczestnicy_konferencji (idOsoba, idRezerwacja_konferencji, Numer_legitymacji)
                              VALUES (@idOsoba, @idRezerwacja_konferencji, @Numer_legitymacji)
                       END
                    ELSE
                       BEGIN
                              RAISERROR ('Rezerwacja konferencji o podanym id nie istnieje', 16, 1)
                              ROLLBACK TRANSACTION
                       END
                    END
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.9. Dodaj_uczestnika_warsztatu

```
CREATE PROCEDURE [dbo].[Dodaj_uczestnika_warsztatu]
                    @idRezerwacja_warsztatu  INT,
                    @idUczestnik_konferencji INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dbo.Uczestnicy_konferencji
                     WHERE idUczestnik_konferencji = @idUczestnik_konferencji))
          BEGIN
                    IF (EXISTS(SELECT *
                               FROM dbo.Rezerwacje_warsztatów
                               WHERE idRezerwacja_warsztatu = @idRezerwacja_warsztatu))
                       BEGIN
                              INSERT INTO Uczestnicy_warsztatów (idRezerwacja_warsztatu, idUczestnik_konferencji)
                              VALUES (@idRezerwacja_warsztatu, @idUczestnik_konferencji)
                       END
                    ELSE
                       BEGIN
                              RAISERROR ('Rezerwacja warsztatu o podanym id nie istnieje', 16, 1)
                              ROLLBACK TRANSACTION
                       END
          END
          BEGIN
                    RAISERROR ('Uczestnik konferencji o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
          END
END
GO
```

6.10. Dodaj_warsztat

```
CREATE PROCEDURE [dbo].[Dodaj_warsztat](
  @idDzień          INT,
  @Nazwa            VARCHAR(50),
  @Prowadzący       VARCHAR(50),
  @Liczba_miejsc    INT,
  @Czas_rozpoczęcia TIME(7),
  @Czas_zakończenia TIME(7),
  @Cena             DECIMAL(18, 2)
)
AS

  SET NOCOUNT ON;
BEGIN
    SET NOCOUNT ON;
    IF (EXISTS(SELECT *
               FROM dni
               WHERE idDzień = @idDzień))
      BEGIN
        INSERT INTO Warsztaty (idDzień, Nazwa, Prowadzący, Liczba_miejsc, Czas_rozpoczęcia, Czas_zakończenia, Cena)
        VALUES (@idDzień, @Nazwa, @Prowadzący, @Liczba_miejsc, @Czas_rozpoczęcia, @Czas_zakończenia, @Cena)
      END
    ELSE
      BEGIN
        RAISERROR ('Dzień o podanym id nie istnieje', 16, 1)
        ROLLBACK TRANSACTION
      END
END
GO
```

6.11. Dodaj_wpłatę

```
CREATE PROCEDURE [dbo].[Dodaj_wpłatę](
          @idZamówienie      INT,
          @Wartość           DECIMAL(18, 2),
          @idForma_płatności INT
)
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM zamówienia
                     WHERE idZamówienie = @idZamówienie))
             BEGIN
                    IF (EXISTS(SELECT *
                               FROM dbo.Formy_płatności
                               WHERE idForma_płatności = @idForma_płatności))
                       BEGIN
                              INSERT INTO Wpłaty (idZamówienie, Wartość, Data, idForma_płatności)
                              VALUES (@idZamówienie, @Wartość, GETDATE(), @idForma_płatności)
                       END
                    ELSE
                       BEGIN
                              RAISERROR ('Forma płatności o podanym id nie istnieje', 16, 1)
                              ROLLBACK TRANSACTION
                       END
             END
          ELSE
            BEGIN
                    RAISERROR ('Zamówienie o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
                    END
END
GO
```

6.12. Edycja_danych_klienta

```
CREATE PROCEDURE Edycja_danych_klienta
    @idKlient       INT,
    @Osoba_prywatna BIT = NULL,
    @Nazwa          NCHAR(10) = NULL,
    @Miasto         NCHAR(10) = NULL,
    @Ulica          NCHAR(10) = NULL,
    @Numer_lokalu   NCHAR(10) = NULL,
    @Kod_pocztowy   NCHAR(10) = NULL,
    @Numer_telefonu NCHAR(10) = NULL,
    @Email          NCHAR(20) = NULL
AS
BEGIN
    SET NOCOUNT ON;
    IF NOT exists(SELECT *
                  FROM Klienci k
                  WHERE k.idKlient = @idKlient)
      BEGIN
        RAISERROR ('Klient o danym id nie istnieje', 16, 1)
        ROLLBACK TRANSACTION
      END
    ELSE
      BEGIN
        IF @Osoba_prywatna IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Osoba_prywatna = @Osoba_prywatna
            WHERE Klienci.idKlient = @idKlient
          END
        IF @Nazwa IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Nazwa = @Nazwa
            WHERE Klienci.idKlient = @idKlient
          END
        IF @Miasto IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Miasto = @Miasto
            WHERE Klienci.idKlient = @idKlient
          END

        IF @Ulica IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Ulica = @Ulica
            WHERE Klienci.idKlient = @idKlient
          END

        IF @Numer_lokalu IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Numer_lokalu = @Numer_lokalu
            WHERE Klienci.idKlient = @idKlient
          END

        IF @Kod_pocztowy IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Kod_pocztowy = @Kod_pocztowy
            WHERE Klienci.idKlient = @idKlient
          END

        IF @Numer_telefonu IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Numer_telefonu = @Numer_telefonu
            WHERE Klienci.idKlient = @idKlient
          END

        IF @Email IS NOT NULL
          BEGIN
            UPDATE Klienci
            SET Klienci.Email = @Email
          WHERE Klienci.idKlient = @idKlient
          END
      END
END
GO
```

6.13. Edycja_danych_osoby

```
CREATE PROCEDURE Edycja_danych_osoby
    @idOsoba        INT,
    @Imię           NCHAR(10) = NULL,
    @Nazwisko       NCHAR(10) = NULL,
    @Miasto         NCHAR(10) = NULL,
    @Ulica          NCHAR(10) = NULL,
    @Numer_telefonu NCHAR(10) = NULL,
    @Email          NCHAR(20) = NULL
AS
  BEGIN
    SET NOCOUNT ON;
    IF NOT exists(SELECT *
                  FROM Osoby o
                  WHERE o.idOsoba = @idOsoba)
      BEGIN
        RAISERROR ('Osoba o danym id nie istnieje', 16, 1)
        ROLLBACK TRANSACTION
      END
    ELSE
      BEGIN
        IF @Imię IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Imię = @Imię
            WHERE Osoby.idOsoba = @idOsoba
          END

        IF @Nazwisko IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Nazwisko = @Nazwisko
            WHERE Osoby.idOsoba = @idOsoba
          END

        IF @Miasto IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Miasto = @Miasto
            WHERE Osoby.idOsoba = @idOsoba
          END

        IF @Ulica IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Ulica = @Ulica
            WHERE Osoby.idOsoba = @idOsoba
          END

        IF @Numer_telefonu IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Numer_telefonu = @Numer_telefonu
            WHERE Osoby.idOsoba = @idOsoba
          END

        IF @Email IS NOT NULL
          BEGIN
            UPDATE Osoby
            SET Osoby.Email = @Email
            WHERE Osoby.idOsoba = @idOsoba
          END
      END
END
GO
```

6.14. Edycja_liczby_miejsc_na_dniu

```
CREATE PROCEDURE [dbo].[Edycja_liczby_miejsc_na_dniu]
          @idDzień            INT,
          @Nowa_liczba_miejsc INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT *
                        FROM Dni d
                        WHERE d.idDzień = @idDzień)
             BEGIN
                    RAISERROR ('Dzień o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          ELSE
             BEGIN
                    UPDATE Dni
                    SET Liczba_miejsc = @Nowa_liczba_miejsc
                    WHERE idDzień = @idDzień
             END
END
GO
```

6.15. Edycja_liczby_miejsc_w_rezerwacji_konferencji

```
ALTER PROCEDURE Edycja_liczby_miejsc_w_rezerwacji_konferencji
          @idRezerwacja_konferencji         INT,
          @Nowa_liczba_miejsc               INT = NULL,
          @Nowa_liczba_miejsc_dla_studentów INT = NULL
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT *
                        FROM Rezerwacje_konferencji rk
                        WHERE rk.idRezerwacja_konferencji = @idRezerwacja_konferencji)
             BEGIN
                    RAISERROR ('Rezerwacja konferencji o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          ELSE
             BEGIN
                    IF NOT exists(SELECT *
                                  FROM Rezerwacje_konferencji rk
                                  WHERE rk.Rezerwacja_anulowana = 0 AND rk.idRezerwacja_konferencji = @idRezerwacja_konferencji)
                       BEGIN
                              RAISERROR ('Rezerwacja konferencji o danym id została anulowana więc zmiana liczby miejsc jest
                                          niemożliwa', 16, 1)
                              ROLLBACK TRANSACTION
                       END
       BEGIN
          IF @Nowa_liczba_miejsc IS NOT NULL
               BEGIN
                    UPDATE Rezerwacje_konferencji
                    SET Liczba_miejsc = @Nowa_liczba_miejsc
                    WHERE idRezerwacja_konferencji = @idRezerwacja_konferencji
                END

          IF @Nowa_liczba_miejsc_dla_studentów IS NOT NULL
             BEGIN
                    UPDATE Rezerwacje_konferencji
                    SET Liczba_miejsc_dla_studentów = @Nowa_liczba_miejsc_dla_studentów
                    WHERE idRezerwacja_konferencji = @idRezerwacja_konferencji
             END
        END
    END
END
GO
```

6.16. Edycja_liczby_miejsc_w_rezerwacji_warsztatu

```
ALTER PROCEDURE Edycja_liczby_miejsc_w_rezerwacji_warsztatu
          @idRezerwacja_warsztatu INT,
          @Nowa_liczba_miejsc     INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT *
                        FROM Rezerwacje_warsztatów rw
                        WHERE rw.idRezerwacja_warsztatu = @idRezerwacja_warsztatu)
             BEGIN
                    RAISERROR ('Rezerwacja warsztatu o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          BEGIN
            IF NOT exists(SELECT *
                          FROM Rezerwacje_warsztatów rw
                          WHERE rw.Rezerwacja_anulowana = 0 AND rw.idRezerwacja_warsztatu = @idRezerwacja_warsztatu)
              BEGIN
                RAISERROR ('Rezerwacja warsztatu o danym id została anulowana więc zmiana liczby miejsc jest niemożliwa', 16, 1)
                ROLLBACK TRANSACTION
               END
          ELSE
              BEGIN
                    UPDATE Rezerwacje_warsztatów
                    SET Liczba_miejsc = @Nowa_liczba_miejsc
                    WHERE idRezerwacja_warsztatu = @idRezerwacja_warsztatu
              END
          END
END
GO
```

6.17. Edytuj_liczbę_miejsc_na_warsztacie

```
CREATE PROCEDURE Edytuj_liczbę_miejsc_na_warsztacie
          @idWarsztat         INT,
          @Nowa_liczba_miejsc INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT *
                        FROM Warsztaty w
                        WHERE w.idWarsztat = @idWarsztat)
             BEGIN
                    RAISERROR ('Warsztat o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          ELSE
             BEGIN
                    UPDATE Warsztaty
                    SET Liczba_miejsc = @Nowa_liczba_miejsc
                    WHERE idWarsztat = @idWarsztat
             END
END
GO
```

6.18. Liczba_uczestników_danego_dnia

```
CREATE PROCEDURE Liczba_uczestników_danego_dnia
          @idDzień INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT d.idDzień
                        FROM Dni d
                        WHERE d.idDzień = @idDzień)
            BEGIN
                    RAISERROR ('Dzień o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
            END
          ELSE
                    SELECT count(*)
                    FROM Uczestnicy_danego_dnia_bez_anulowanych(@idDzień)
END
GO
```

6.19. Liczba_uczestników_danego_warsztatu

```
CREATE PROCEDURE Liczba_uczestników_danego_warsztatu
          @idWarsztatu INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT w.idWarsztat
                        FROM Warsztaty w
                        WHERE w.idWarsztat = @idWarsztatu)
             BEGIN
                    RAISERROR ('Warsztat o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          ELSE
                    SELECT count(*)
                    FROM Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych(@idWarsztatu)
END
GO
```

6.20. Liczba_uczestników_studentów_danego_dnia

```
CREATE PROCEDURE Liczba_uczestników_studentów_danego_dnia
          @idDzień INT
AS
BEGIN
          SET NOCOUNT ON;
          IF NOT exists(SELECT d.idDzień
                        FROM Dni d
                        WHERE d.idDzień = @idDzień)
             BEGIN
                    RAISERROR ('Dzień o danym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
          ELSE
                    SELECT count(*)
                    FROM Studenci_w_danym_dniu_bez_anulowanych(@idDzień)
END
GO
```

6.21. _Ile_zostało_do_zapłaty

```
CREATE PROCEDURE [dbo].[_Ile_zostało_do_zapłaty]
          @idZamówienie INT
AS
          IF (EXISTS(SELECT *
                    FROM zamówienia
                     WHERE idZamówienie = @idZamówienie))
             SELECT *
             FROM dbo.Ile_zostało_do_zapłaty(@idZamówienie)
          ELSE
             BEGIN
                    RAISERROR ('Zamówienie o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
GO
```

6.22. _Jaka_cena_obowiązuje

```
CREATE PROCEDURE [dbo].[_Jaka_cena_obowiązuje]
          @idDzień INT,
          @Data    DATE
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dni
                     WHERE idDzień = @idDzień))
             BEGIN
                    IF ((SELECT Data
                           FROM DNI
                           WHERE idDzień = @idDzień) >= @Data)
                       SELECT *
                       FROM dbo.Jaka_cena_obowiązuje(@idDzień, @Data)
                    ELSE
                       BEGIN
                              RAISERROR ('Podana data jest po dacie konferencji', 16, 1)
                              ROLLBACK TRANSACTION
                        END
                     END
          ELSE
             BEGIN
                    RAISERROR ('Dzień o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.23. _Koszty_za_osoby_z_danego_zamówienia

```
CREATE PROCEDURE [dbo].[_Koszty_za_osoby_z_danego_zamówienia]
          @idZamówienie INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM zamówienia
                     WHERE idZamówienie = @idZamówienie))
             SELECT *
             FROM dbo.Koszty_za_osoby_z_danego_zamówienia(@idZamówienie)
          ELSE
             BEGIN
                    RAISERROR ('Zamówienie o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.24. _Koszty_za_uczestników_z_danego_zamówienia

```
CREATE PROCEDURE [dbo].[_Koszty_za_uczestników_z_danego_zamówienia]
          @idZamówienie INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM zamówienia
                     WHERE idZamówienie = @idZamówienie))
             SELECT *
             FROM dbo.Koszty_za_uczestników_z_danego_zamówienia(@idZamówienie)
          ELSE
             BEGIN
                    RAISERROR ('Zamówienie o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.25. _Koszty_za_uczestników_z_danej_rezerwacji

```
CREATE PROCEDURE [dbo].[_Koszty_za_uczestników_z_danej_rezerwacji]
          @idRezerwacji_konferencji INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dbo.Rezerwacje_konferencji
                     WHERE idRezerwacja_konferencji = @idRezerwacji_konferencji))
              SELECT *
              FROM dbo.Koszty_za_uczestników_z_danej_rezerwacji(@idRezerwacji_konferencji)
          ELSE
             BEGIN
                    RAISERROR ('Rezerwacja konferencji o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
          END
END
GO
```

6.26. _Moje_nadchodzące_dni_konferncji

```
CREATE PROCEDURE [dbo].[_Moje_nadchodzące_dni_konferncji]
          @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
             SELECT *
             FROM dbo.Moje_nadchodzące_dni_konferencji(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.27. _Moje_nadchodzące_dni_konferncji_i_warsztaty

```
CREATE PROCEDURE [dbo].[_Moje_nadchodzące_dni_konferncji_i_warsztaty]
          @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
             SELECT *
             FROM dbo.Moje_nadchodzące_dni_konferencji_i_warsztaty(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.28. _Moje_nadchodzące_warsztaty

```
CREATE PROCEDURE [dbo].[_Moje_nadchodzące_warsztaty]
                    @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
             SELECT *
             FROM dbo.Moje_nadchodzące_warsztaty(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.29. _Na_co_jestem_zapisany

```
CREATE PROCEDURE [dbo].[_Na_co_jestem_zapisany]
          @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
               SELECT *
               FROM dbo.Na_co_jestem_zapisany(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.30. _Na_jakie_dni_konferencji_jestem_zapisany

```
CREATE PROCEDURE [dbo].[_Na_jakie_dni_konferencji_jestem_zapisany]
          @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
             SELECT *
             FROM dbo.Na_jakie_dni_konferencji_jestem_zapisany(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
          END
END
GO
```

6.31. _Na_jakie_warsztaty_jestem_zapisany

```
CREATE PROCEDURE [dbo].[_Na_jakie_warsztaty_jestem_zapisany]
          @idOsoba INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM osoby
                     WHERE idosoba = @idOsoba))
              SELECT *
              FROM dbo.Na_jakie_warsztaty_jestem_zapisany(@idOsoba)
          ELSE
             BEGIN
                    RAISERROR ('Osoba o podanym id nie istnieje', 16, 1)
                    ROLLBACK TRANSACTION
             END
END
GO
```

6.32. _Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych

```
CREATE PROCEDURE _Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych
         @idWarsztat INT
AS
BEGIN
        SET NOCOUNT ON;
        IF (EXISTS(SELECT *
                   FROM Warsztaty
                   WHERE idWarsztat = @idWarsztat))
            SELECT *
            FROM dbo.Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych(@idWarsztat);
        ELSE
            BEGIN
                RAISERROR ('Warsztat o podanym id nie istnieje', 16, 1);
                ROLLBACK TRANSACTION;
            END;
END;
GO
```

6.33. _Płatności_danego_klienta

```
CREATE PROCEDURE _Płatności_danego_klienta
    @idKlient INT
AS
BEGIN
    SET NOCOUNT ON;
    IF (EXISTS(SELECT *
               FROM klienci
               WHERE idKlient = @idKlient))
      SELECT *
      FROM dbo.Płatności_danego_klienta(@idKlient)
    ELSE
      BEGIN
        RAISERROR ('Klient o podanym id nie istnieje', 16, 1)
        ROLLBACK TRANSACTION
      END
END
GO
```

6.34. _Szczegóły_zamówienia

```
CREATE PROCEDURE [dbo].[_Szczegóły_zamówienia]
          @idZamówienie INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dbo.Zamówienia
                     WHERE idZamówienie = @idZamówienie))
             SELECT *
             FROM dbo.Szczegóły_zamówienia(@idZamówienie)
          ELSE
                     BEGIN
                               RAISERROR ('Zamówienie o podanym id nie istnieje', 16, 1)
                               ROLLBACK TRANSACTION
                     END
END
GO
```

6.35. _Uczestnicy_danego_dnia_bez_anulowanych

```
CREATE PROCEDURE _Uczestnicy_danego_dnia_bez_anulowanych
          @idDzień INT
AS
BEGIN
          SET NOCOUNT ON;
          IF (EXISTS(SELECT *
                     FROM dni
                     WHERE idDzień = @idDzień))
             SELECT *
             FROM dbo.Uczestnicy_danego_dnia_bez_anulowanych(@idDzień)
          ELSE
                     BEGIN
                               RAISERROR ('Nie istnieje dzień o podanym id', 16, 1)
                               ROLLBACK TRANSACTION
                     END
END
GO
```

7. Funkcje

7.1. Dni_danej_konferencji

```
CREATE FUNCTION Dni_danej_konferencji
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT *
	FROM Dni d
	WHERE d.idKonferencja = @idKonferencja
)
GO
```

7.2. Identyfikatory_na_dany_dzień

```
ALTER FUNCTION Identyfikatory_na_dany_dzień
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.Imię,
		o.Nazwisko,
		CASE WHEN k.Osoba_prywatna = 0
			THEN k.Nazwa
			ELSE '' END AS Firma,
		ko.Nazwa,
		d.Data
	FROM osoby o
		INNER JOIN Uczestnicy_konferencji uk ON uk.idOsoba = o.idOsoba
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
		INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
		INNER JOIN dni d ON rk.idDzień = d.idDzień
		INNER JOIN Konferencje ko ON d.idKonferencja = ko.idKonferencja
		INNER JOIN Klienci k ON k.idKlient = z.idKlient
	WHERE rk.idDzień = @idDzień
)
GO
```

7.3. Ile_zostało_do_zapłaty

```
CREATE FUNCTION [dbo].[Ile_zostało_do_zapłaty]
(
    @idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
  SELECT
    z.idZamówienie,
    CASE WHEN (z.Zamówienie_anulowane = 1) THEN 0
        ELSE ((isnull((SELECT SUM(w.Cena * rw.Liczba_miejsc) AS Expr1
                    FROM dbo.Rezerwacje_konferencji AS rk
                       INNER JOIN dbo.Rezerwacje_warsztatów AS rw ON rw.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
                       INNER JOIN dbo.Warsztaty AS w ON w.idWarsztat = rw.idWarsztat
                    WHERE (rk.idZamówienie = z.idZamówienie)), 0)
            +isnull((SELECT SUM(c.Cena * rk.Liczba_miejsc + c.Cena_dla_studenta * rk.Liczba_miejsc_dla_studentów) AS Expr1
                    FROM dbo.Rezerwacje_konferencji AS rk
                       INNER JOIN dbo.Ceny AS c ON c.idDzień = rk.idDzień
                                       AND c.idCena = (SELECT idCena
                                                       FROM Jaka_cena_obowiązuje(rk.idDzień, z.Data_zgłoszenia))
                                                        WHERE rk.idZamówienie = z.idZamówienie), 0))
            -isnull((SELECT SUM(wp.Wartość)
                          FROM dbo.Wpłaty AS wp
                          WHERE (idZamówienie = z.idZamówienie)), 0)) END AS Ile
  FROM Zamówienia z
  WHERE z.idZamówienie = @idZamówienie
)
GO
```

7.4. Jaka_cena_obowiązuje

```
CREATE FUNCTION [dbo].[Jaka_cena_obowiązuje]
(
          @idDzień INT,
          @Data    DATE
)
RETURNS TABLE
AS
RETURN
(
          SELECT TOP (1)
                    idCena,
                    Cena,
                    Cena_dla_studenta
          FROM Ceny AS c1
          WHERE (idDzień = @idDzień) AND (Data <= @Data)
          ORDER BY Data DESC
)
GO
```

7.5. Klienci_zgłaszający_daną_osobę

```
CREATE FUNCTION [dbo].[Klienci_zgłaszający_daną_osobę]
(
	@idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		k.idKlient,
		k.Nazwa
	FROM klienci k
		INNER JOIN zamówienia z ON z.idKlient = k.idKlient
		INNER JOIN Rezerwacje_konferencji rk ON rk.idZamówienie = z.idZamówienie
		INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
	WHERE uk.idOsoba = @idOsoba
)
GO
```

7.6. Koszt_danego_zamówienia

```
CREATE FUNCTION Koszt_danego_zamówienia
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		z.idZamówienie,
		z.idKlient,
		ko.idKonferencja,
		ko.nazwa AS Nazwa_konferencji,
		k.nazwa AS Nazwa_klienta,
		z.Data_zgłoszenia,
		(SELECT Koszt_konferencji
		    FROM Koszt_konferencji_w_danym_zamówieniu(z.idZamówienie))  AS Koszt_konferencji,
		(SELECT Koszt_warsztatów
		    FROM Koszt_warsztatów_w_danym_zamówieniu(z.idZamówienie))   AS Koszt_warsztatów,
		((SELECT Koszt_konferencji
			FROM Koszt_konferencji_w_danym_zamówieniu(z.idZamówienie))
		   + (SELECT Koszt_warsztatów
			FROM Koszt_warsztatów_w_danym_zamówieniu(z.idZamówienie))) AS Koszt_całkowity
	FROM Zamówienia z
		INNER JOIN Klienci k ON k.idKlient = z.idKlient
		INNER JOIN Konferencje ko ON ko.idKonferencja = z.idKonferencja
	WHERE z.idZamówienie = @idZamówienie
	)
GO
```

7.7. Koszt_konferencji_w_danym_zamówieni

```
CREATE FUNCTION [dbo].[Koszt_konferencji_w_danym_zamówieniu]
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		z.idZamówienie,
		z.idKlient,
		ko.idKonferencja,
		ko.nazwa AS Nazwa_konferencji,
		k.nazwa AS Nazwa_klienta,
		z.Data_zgłoszenia,
		(isnull((SELECT SUM(c.Cena * rk.Liczba_miejsc + c.Cena_dla_studenta * rk.Liczba_miejsc_dla_studentów) AS a
			FROM Rezerwacje_konferencji AS rk
			  INNER JOIN Ceny AS c ON c.idDzień = rk.idDzień AND c.idCena = (SELECT TOP (1) idCena
			                                                                 FROM Ceny AS c1
			                                                                 WHERE (rk.idZamówienie = z.idZamówienie)
			                                                                     AND (idDzień = rk.idDzień)
			                                                                     AND (Data <= z.Data_zgłoszenia)
			                                                                   ORDER BY Data DESC)
			WHERE z.Zamówienie_anulowane = 0), 0)) AS Koszt_konferencji
	FROM Zamówienia z
		INNER JOIN Klienci k ON k.idKlient = z.idKlient
		INNER JOIN Konferencje ko ON ko.idKonferencja = z.idKonferencja
	WHERE z.idZamówienie = @idZamówienie
)
GO
```

7.8. Koszt_warsztatów_w_danym_zamówieniu

```
CREATE FUNCTION [dbo].[Koszt_warsztatów_w_danym_zamówieniu]
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		z.idZamówienie,
		z.idKlient,
		ko.idKonferencja,
		ko.nazwa AS Nazwa_konferencji,
		k.nazwa AS Nazwa_klienta,
		z.Data_zgłoszenia,
		(isnull((SELECT SUM(w.Cena * rw.Liczba_miejsc) AS b
			FROM Rezerwacje_konferencji AS rk
			     INNER JOIN Rezerwacje_warsztatów AS rw
			           ON rw.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
			      INNER JOIN Warsztaty AS w ON w.idWarsztat = rw.idWarsztat
			WHERE (rk.idZamówienie = z.idZamówienie AND z.Zamówienie_anulowane = 0)), 0)) AS Koszt_warsztatów
	FROM Zamówienia z
		INNER JOIN Klienci k ON k.idKlient = z.idKlient
		INNER JOIN Konferencje ko ON ko.idKonferencja = z.idKonferencja
	WHERE z.idZamówienie = @idZamówienie
)
GO
```

7.9. Koszt_za_danego_uczestnika

```
CREATE FUNCTION [dbo].[Koszt_za_danego_uczestnika]
(
	@idUczestnik_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		uk.idUczestnik_konferencji,
		rk.IdRezerwacja_konferencji,
		rk.idDzień,
		z.idZamówienie,
		o.imię,
		o.nazwisko,
		uk.Numer_legitymacji,
		CASE WHEN (z.Zamówienie_anulowane = 1) THEN 0
		    ELSE (isnull((SELECT Koszt
		                  FROM Koszt_za_konferencję_danego_uczestnika(uk.idUczestnik_konferencji)), 0)
		          + isnull((SELECT Koszt_warsztatów
		                    FROM Koszt_za_warsztaty_danego_uczestnika(uk.idUczestnik_konferencji)), 0))
		   END AS Koszt
	FROM Uczestnicy_konferencji uk
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
		INNER JOIN Osoby o ON o.idOsoba = uk.idOsoba
		INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
	WHERE uk.idUczestnik_konferencji = @idUczestnik_konferencji
)
GO
```

7.10. Koszt_za_konferencję_danego_uczestnika

```
CREATE FUNCTION [dbo].[Koszt_za_konferencję_danego_uczestnika]
(
	@idUczestnik_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		uk.idUczestnik_konferencji,
		o.idOsoba,
		o.Imię,
		o.Nazwisko,
		uk.Numer_legitymacji,
		ISNULL((CASE WHEN uk.Numer_legitymacji IS NULL
			THEN (SELECT Cena
				FROM Jaka_cena_obowiązuje(rk.idDzień, z.Data_zgłoszenia))
			ELSE (SELECT Cena_dla_studenta
				FROM Jaka_cena_obowiązuje(rk.idDzień, z.Data_zgłoszenia)) END), 0) AS Koszt
	FROM Uczestnicy_konferencji uk
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
		INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
		INNER JOIN Osoby o ON o.idOsoba = uk.idOsoba
	WHERE uk.idUczestnik_konferencji = @idUczestnik_konferencji
)
GO
```

7.11. Koszt_za_warsztaty_danego_uczestnika

```
CREATE FUNCTION [dbo].[Koszt_za_warsztaty_danego_uczestnika]
(
	@idUczestnik_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		uk.idUczestnik_konferencji,
		o.idOsoba,
		o.Imię,
		o.Nazwisko,
		uk.Numer_legitymacji,
		isnull(sum(w.cena), 0) AS Koszt_warsztatów
	FROM Uczestnicy_warsztatów uw
		INNER JOIN Rezerwacje_warsztatów rw ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
		INNER JOIN Warsztaty w ON w.idWarsztat = rw.idWarsztat
		INNER JOIN Uczestnicy_konferencji uk ON uk.idUczestnik_konferencji = uw.idUczestnik_konferencji
		INNER JOIN Osoby o ON o.idOsoba = uk.idOsoba
	WHERE uw.idUczestnik_konferencji = @idUczestnik_konferencji
	GROUP BY uk.idUczestnik_konferencji, o.idOsoba, o.Imię, o.Nazwisko, uk.Numer_legitymacji
)
GO
```

7.12. Koszty_za_osoby_z_danego_zamówienia

```
CREATE FUNCTION [dbo].[Koszty_za_osoby_z_danego_zamówienia]
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		uk.idOsoba,
		ku.Imię,
		ku.Nazwisko,
		isnull(SUM(ku.Koszt), 0) AS Koszt
	FROM Koszty_za_uczestników_z_danego_zamówienia(@idZamówienie) ku
		INNER JOIN Uczestnicy_konferencji uk ON uk.idUczestnik_konferencji = ku.idUczestnik_konferencji
	GROUP BY uk.idOsoba, ku.Imię, ku.Nazwisko
)
GO
```

7.13. Koszty_za_uczestników_z_danego_zamówienia

```
CREATE FUNCTION [dbo].[Koszty_za_uczestników_z_danego_zamówienia]
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		uk.idUczestnik_konferencji,
		rk.idRezerwacja_konferencji,
		d.Data AS Dzień,
		(SELECT Imię
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Imię,
		(SELECT Nazwisko
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Nazwisko,
		(SELECT Numer_legitymacji
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Numer_legitymacji,
		(SELECT Koszt
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Koszt
	FROM Zamówienia z
		INNER JOIN Rezerwacje_konferencji rk ON rk.idZamówienie = z.idZamówienie
		INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
		INNER JOIN Dni d ON d.idDzień = rk.idDzień
	WHERE z.idZamówienie = @idZamówienie
)
GO
```

7.14. Koszty_za_uczestników_w_danym_dniu_danego_zamówienia

```
CREATE FUNCTION [dbo].[Koszty_za_uczestników_w_danym_dniu_danego_zamówienia]
(
	@idRezerwacja_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		(SELECT idUczestnik_konferencji
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS idUczestnik_konferencji,
		(SELECT imię
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Imię,
		(SELECT nazwisko
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Nazwisko,
		(SELECT Numer_legitymacji
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Numer_legitymacji,
		(SELECT Koszt
		    FROM Koszt_za_danego_uczestnika(uk.idUczestnik_konferencji)) AS Koszt
	FROM Rezerwacje_konferencji rk
		INNER JOIN dni d ON d.idDzień = rk.idDzień
		INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
	WHERE rk.idRezerwacja_konferencji = @idRezerwacja_konferencji
)
GO
```

7.15. Liczba_brakujących_uczestników_dnia_konferencji_z_danej_rezerwacji

```
CREATE FUNCTION Liczba_brakujących_uczestników_dnia_konferencji_z_danej_rezerwacji
(
	@idRezerwacja_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		rk.Liczba_miejsc - (SELECT count(*)
				FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(@idRezerwacja_konferencji)
				WHERE Student = 'NIE') AS Normalne,
		rk.Liczba_miejsc_dla_studentów - (SELECT count(*)
				FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(@idRezerwacja_konferencji)
				WHERE Student = 'TAK') AS Studenckie
	FROM Rezerwacje_konferencji rk
	WHERE rk.idRezerwacja_konferencji = @idRezerwacja_konferencji AND rk.Rezerwacja_anulowana = 0
)
GO
```

7.16. Liczba_brakujących_uczestników_warsztatu_z_danej_rezerwacji

```
CREATE FUNCTION Liczba_brakujących_uczestników_warsztatu_z_danej_rezerwacji
(
	@idRezerwacja_warsztatu INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT rw.Liczba_miejsc - (SELECT isnull(count(*), 0)
				FROM Uczestnicy_warsztatu_z_danej_rezerwacji(@idRezerwacja_warsztatu)) AS Normalne

	FROM Rezerwacje_warsztatów rw
	WHERE rw.idRezerwacja_warsztatu = @idRezerwacja_warsztatu AND rw.Rezerwacja_anulowana = 0
)
GO
```

7.17. Liczba_brakujących_uczestników_z_danej_rezerwacji

```
CREATE FUNCTION Liczba_brakujących_uczestników_z_danej_rezerwacji
(
    @idZamówienie INT
)
  RETURNS TABLE
  AS
  RETURN
  (
SELECT
    rk.idDzień,
    d.Data,
    rk.Liczba_miejsc,
    rk.Liczba_miejsc - (SELECT count(*)
                        FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(rk.idRezerwacja_konferencji)
                        WHERE Student = 'NIE') AS 'Brak uczestników normalnych',

    rk.Liczba_miejsc_dla_studentów,
    rk.Liczba_miejsc_dla_studentów - (SELECT count(*)
                                      FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(rk.idRezerwacja_konferencji)
                                      WHERE Student = 'TAK') AS 'Brak uczestników studenckich',
    'Konferencja ' + k.Nazwa AS Rodzaj_rezerwacji,
    rk.Rezerwacja_anulowana
FROM Rezerwacje_konferencji rk
    INNER JOIN Dni d ON d.idDzień = rk.idDzień
    INNER JOIN Konferencje k ON d.idKonferencja = k.idKonferencja
WHERE rk.idZamówienie = @idZamówienie AND rk.Rezerwacja_anulowana = 0
UNION
SELECT
    w.idDzień,
    d2.Data,
    rw.Liczba_miejsc,
    rw.Liczba_miejsc - (SELECT isnull(count(*), 0)
                        FROM Uczestnicy_warsztatu_z_danej_rezerwacji(rw.idRezerwacja_warsztatu)),
    0,
    0,
    'Warsztat ' + w.Nazwa AS Rodzaj_rezerwacji,
    rw.Rezerwacja_anulowana
FROM rezerwacje_warsztatów rw
    INNER JOIN rezerwacje_konferencji rk2 ON rk2.idRezerwacja_konferencji = rw.idRezerwacja_konferencji
    INNER JOIN dni d2 ON d2.idDzień = rk2.idDzień
    INNER JOIN Warsztaty w ON rw.idWarsztat = w.idWarsztat
WHERE rk2.idZamówienie = @idZamówienie AND rw.Rezerwacja_anulowana = 0
)
GO
```

7.18. Moje_nadchodzące_dni_konferencji

```
CREATE FUNCTION [dbo].[Moje_nadchodzące_dni_konferencji]
(
          @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT *
          FROM Na_jakie_dni_konferencji_jestem_zapisany(@idOsoba)
          WHERE Data >= getdate()
)
GO
```

7.19. Moje_nadchodzące_dni_konferencji_i_warsztaty

```
CREATE FUNCTION Moje_nadchodzące_dni_konferencji_i_warsztaty
(
          @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT *
          FROM Na_co_jestem_zapisany(@idOsoba)
          WHERE Data >= getdate()
)
GO
```

7.20. Moje_nadchodzące_warsztaty

```
CREATE FUNCTION Moje_nadchodzące_warsztaty
(
          @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT *
          FROM Na_jakie_warsztaty_jestem_zapisany(@idOsoba)
          WHERE Data >= getdate()

)
GO
```

7.21. Na_co_jestem_zapisany

```
CREATE FUNCTION Na_co_jestem_zapisany
(
    @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
SELECT
    k.idkonferencja,
    k.nazwa AS Nazwa,
    d.data,
    k.miejsce,
    'Dzień konferencji' AS Typ,
    kl.Nazwa AS 'Nazwa klienta'
FROM Dni d
    INNER JOIN konferencje k ON k.idKonferencja = d.idKonferencja
    INNER JOIN Rezerwacje_konferencji rk ON rk.idDzień = d.idDzień
    INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
    INNER JOIN zamówienia z ON rk.idZamówienie = z.idZamówienie
    INNER JOIN Klienci kl ON z.idKlient = kl.idKlient
WHERE uk.idOsoba = @idOsoba
UNION
SELECT
    k.idkonferencja,
    w.nazwa AS Nazwa,
    d.data,
    k.miejsce,
    'Warsztat AS Typ,
    kl.Nazwa AS 'Nazwa klienta'
FROM Dni d
    INNER JOIN konferencje k ON k.idKonferencja = d.idKonferencja
    INNER JOIN Rezerwacje_konferencji rk ON rk.idDzień = d.idDzień
    INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
    INNER JOIN zamówienia z ON rk.idZamówienie = z.idZamówienie
    INNER JOIN Klienci kl ON z.idKlient = kl.idKlient
    INNER JOIN Uczestnicy_warsztatów uw ON uk.idUczestnik_konferencji = uw.idUczestnik_konferencji
    INNER JOIN Warsztaty w ON d.idDzień = w.idDzień
WHERE uk.idOsoba = @idOsoba
)
GO
```

7.22. Na_jakie_dni_konferencji_jestem_zapisany

```
CREATE FUNCTION Na_jakie_dni_konferencji_jestem_zapisany
(
          @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT DISTINCT
                    k.idkonferencja,
                    k.nazwa S Nazwa,
                    d.data,
                    k.miejsce,
                    'Dzień konferencji' AS Typ,
                    kl.Nazwa AS 'Nazwa klienta'
          FROM Dni d
                    INNER JOIN konferencje k ON k.idKonferencja = d.idKonferencja
                    INNER JOIN Rezerwacje_konferencji rk ON rk.idDzień = d.idDzień
                    INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
                    INNER JOIN zamówienia z ON rk.idZamówienie = z.idZamówienie
                    INNER JOIN Klienci kl ON z.idKlient = kl.idKlient
          WHERE uk.idOsoba = @idOsoba
)
GO
```

7.23. Na_jakie_warsztaty_jestem_zapisany

```
CREATE FUNCTION Na_jakie_warsztaty_jestem_zapisany
(
    @idOsoba INT
)
RETURNS TABLE
AS
RETURN
(
SELECT DISTINCT
    k.idkonferencja,
    w.nazwa    AS Nazwa,
    d.data,
    w.Czas_rozpoczęcia,
    w.Czas_zakończenia,
    k.miejsce,
    'Warsztat' AS Typ,
    kl.Nazwa   AS 'Nazwa klienta'
FROM Dni d
    INNER JOIN konferencje k ON k.idKonferencja = d.idKonferencja
    INNER JOIN Rezerwacje_konferencji rk ON rk.idDzień = d.idDzień
    INNER JOIN Uczestnicy_konferencji uk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
    INNER JOIN zamówienia z ON rk.idZamówienie = z.idZamówienie
    INNER JOIN Klienci kl ON z.idKlient = kl.idKlient
    INNER JOIN Uczestnicy_warsztatów uw ON uk.idUczestnik_konferencji = uw.idUczestnik_konferencji
    INNER JOIN Warsztaty w ON d.idDzień = w.idDzień
WHERE uk.idOsoba = @idOsoba
)
GO
```

7.24. Osoby_od_danego_klienta

```
CREATE FUNCTION Osoby_od_danego_klienta
(
          @idKlinet INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT
                    o.idOsoba,
                    o.Imię,
                    o.Nazwisko
          FROM Osoby o
                    INNER JOIN Uczestnicy_konferencji uk ON uk.idOsoba = o.idOsoba
                    INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                    INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
                    INNER JOIN klienci k ON k.idKlient = z.idKlient
          WHERE k.idKlient = @idKlinet
)
GO
```

7.25. Osoby_od_danego_klienta_bez_anulowanych

```
CREATE FUNCTION Osoby_od_danego_klienta_bez_anulowanych
(
          @idKlinet INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT
                    o.idOsoba,
                    o.Imię,
                    o.Nazwisko
          FROM Osoby o
                    INNER JOIN Uczestnicy_konferencji uk ON uk.idOsoba = o.idOsoba
                    INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
                    INNER JOIN Zamówienia z ON z.idZamówienie = rk.idZamówienie
                    INNER JOIN klienci k ON k.idKlient = z.idKlient
          WHERE k.idKlient = @idKlinet AND rk.Rezerwacja_anulowana = 0
)
GO
```

7.26. Osoby_zapisane_z_danego_zamówienia

```
CREATE FUNCTION Osoby_zapisane_z_danego_zamówienia
(
    @idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
SELECT
    rk.idDzień,
    d.Data,
    o.Imię,
    o.Nazwisko,
    'Konferencja ' + k.Nazwa AS Rodzaj_rezerwacji,
FROM Rezerwacje_konferencji rk
    INNER JOIN Dni d ON d.idDzień = rk.idDzień
    INNER JOIN Konferencje k ON d.idKonferencja = k.idKonferencja
    INNER JOIN uczestnicy_konferencji uk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
    INNER JOIN Osoby o ON uk.idOsoba = o.idOsoba
WHERE rk.idZamówienie = @idZamówienie
UNION
SELECT
    w.idDzień,
    d2.Data,
    o.Imię,
    o.Nazwisko,
    'Warsztat ' + w.Nazwa AS Rodzaj_rezerwacji,
FROM rezerwacje_warsztatów rw
    INNER JOIN rezerwacje_konferencji rk2 ON rk2.idRezerwacja_konferencji = rw.idRezerwacja_konferencji
    INNER JOIN dni d2 ON d2.idDzień = rk2.idDzień
    INNER JOIN Warsztaty w ON rw.idWarsztat = w.idWarsztat
    INNER JOIN uczestnicy_warsztatów uw ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
    INNER JOIN Uczestnicy_konferencji ik ON rk2.idRezerwacja_konferencji = ik.idRezerwacja_konferencji
    INNER JOIN osoby o ON ik.idOsoba = o.idOsoba
WHERE rk2.idZamówienie = @idZamówienie
)
GO
```

7.27. Osoby_zgłoszone_na_dany_warsztat

```
CREATE FUNCTION [dbo].[Osoby_zgłoszone_na_dany_warsztat]
(
	@idWarsztat INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.idosoba,
		o.Imię,
		o.Nazwisko,
		o.Miasto,
		o.Ulica,
		o.Numer_telefonu,
		o.Email,
		o.Data_dodania
	FROM Osoby o
		INNER JOIN Uczestnicy_konferencji uk ON uk.idOsoba = o.idOsoba
		INNER JOIN Uczestnicy_warsztatów uw ON uw.idUczestnik_konferencji = uk.idUczestnik_konferencji
		INNER JOIN Rezerwacje_warsztatów rw ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
	WHERE rw.idWarsztat = @idWarsztat
)
GO
```

7.28. Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych

```
CREATE FUNCTION Osoby_zgłoszone_na_dany_warsztat_bez_anulowanych
(
	@idWarsztat INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.idosoba,
		o.Imię,
		o.Nazwisko,
		o.Miasto,
		o.Ulica,
		o.Numer_telefonu,
		o.Email,
		o.Data_dodania
	FROM Osoby o
		INNER JOIN Uczestnicy_konferencji uk ON uk.idOsoba = o.idOsoba
		INNER JOIN Uczestnicy_warsztatów uw ON uw.idUczestnik_konferencji = uk.idUczestnik_konferencji
		INNER JOIN Rezerwacje_warsztatów rw ON rw.idRezerwacja_warsztatu = uw.idRezerwacja_warsztatu
	WHERE rw.idWarsztat = @idWarsztat AND rw.Rezerwacja_anulowana = 0
)
GO
```

7.29. Płatności_danego_klienta

```
CREATE FUNCTION [dbo].[Płatności_danego_klienta]
(
	@idKlient INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWpłata,
		w.idZamówienie,
		ko.idKonferencja,
		ko.Nazwa AS Nazwa_konferencji,
		w.Wartość,
		w.Data,
		fp.nazwa AS Forma_płatności
	FROM Wpłaty w
		INNER JOIN Zamówienia z ON z.idZamówienie = w.idZamówienie
		INNER JOIN konferencje ko ON ko.idKonferencja = z.idKonferencja
		INNER JOIN klienci k ON k.idKlient = z.idKlient
		INNER JOIN Formy_płatności fp ON w.idForma_płatności = fp.idForma_płatności
	WHERE k.idKlient = @idKlient
)
GO
```

7.30. Płatności_dla_danej_konferencji

```
CREATE FUNCTION [dbo].[Płatności_dla_danej_konferencji]
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWpłata,
		w.idZamówienie,
		w.Wartość,
		w.Data,
		fp.nazwa AS Forma_płatności
	FROM Wpłaty w
		INNER JOIN Zamówienia z ON z.idZamówienie = w.idZamówienie
		INNER JOIN Formy_płatności fp ON w.idForma_płatności = fp.idForma_płatności
	WHERE z.idKonferencja = @idKonferencja
)
GO
```

7.31. Płatności_w_danej_formie

```
CREATE FUNCTION [dbo].[Płatności_w_danej_formie]
(
	@idForma_płatności INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWpłata,
		w.idZamówienie,
		k.idKonferencja,
		k.nazwa AS Nazwa_konferencji,
		w.Wartość,
		w.Data
	FROM Wpłaty w
		INNER JOIN Zamówienia z ON z.idZamówienie = w.idZamówienie
		INNER JOIN Konferencje k ON k.idKonferencja = z.idKonferencja
	WHERE w.idForma_płatności = @idForma_płatności
)
GO
```

7.32. Progi_cenowe_danego_dnia_konferencji

```
ALTER FUNCTION Progi_cenowe_danego_dnia_konferencji
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		c.idDzień,
		c.idCena,
		c.Data,
		c.Cena,
		c.Cena_dla_studenta
	FROM Ceny c
	WHERE c.idDzień = @idDzień
)
GO
```

7.33. Przywróć_anulowane_zamówienie

```
ALTER PROCEDURE Przywróć_anulowane_zamówienie
    @idZamówienia INT
AS
BEGIN
    SET NOCOUNT ON;

    IF NOT exists(SELECT *
                  FROM Zamówienia z
                  WHERE z.idZamówienie = @idZamówienia)
      BEGIN
        RAISERROR ('Zamówienie o podanym id nie istnieje.', 16, 1)
        ROLLBACK TRANSACTION
      END
    ELSE
      BEGIN
       IF NOT exists(SELECT *
                     FROM Anulowane_zamówienia az
                     WHERE az.idZamówienie = @idZamówienia)
          BEGIN
            RAISERROR ('Zamówienie o podanym id nie było anulowane.', 16, 1)
            ROLLBACK TRANSACTION
          END
      ELSE
        UPDATE Zamówienia
        SET Zamówienie_anulowane = 0
        WHERE idZamówienie = @idZamówienia

        UPDATE Zamówienia
        SET Data_anulowania = NULL
        WHERE idZamówienie = @idZamówienia

        UPDATE Rezerwacje_konferencji
        SET Rezerwacja_anulowana = 0
        WHERE idZamówienie = @idZamówienia

        UPDATE Rezerwacje_warsztatów
        SET Rezerwacja_anulowana = 0
        WHERE idRezerwacja_konferencji IN (SELECT rk.idRezerwacja_konferencji
                                           FROM Rezerwacje_konferencji rk
                                           WHERE rk.idZamówienie = @idZamówienia)
      END
END
GO
```

7.34. Studenci_w_danym_dniu

```
CREATE FUNCTION Studenci_w_danym_dniu
(
          @idDzień INT
)
RETURNS TABLE
AS
RETURN
(
          SELECT
                    o.Imię,
                    o.Nazwisko,
                    o.Miasto,
                    o.Ulica,
                    o.Numer_telefonu,
                    o.Email,
                    o.Data_dodania
          FROM Osoby o
                    INNER JOIN Uczestnicy_konferencji uk ON o.idOsoba = uk.idOsoba
                    INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
          WHERE rk.idDzień = @idDzień AND uk.Numer_legitymacji IS NOT NULL
)
GO
```

7.35. Studenci_w_danym_dniu_bez_anulowanych

```
CREATE FUNCTION Studenci_w_danym_dniu_bez_anulowanych
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.Imię,
		o.Nazwisko,
		o.Miasto,
		o.Ulica,
		o.Numer_telefonu,
		o.Email,
		o.Data_dodania
	FROM Osoby o
		INNER JOIN Uczestnicy_konferencji uk ON o.idOsoba = uk.idOsoba
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
	WHERE rk.idDzień = @idDzień AND uk.Numer_legitymacji IS NOT NULL AND rk.Rezerwacja_anulowana = 0
)
GO
```

7.36. Szczegóły_zamówienia

```
ALTER FUNCTION Szczegóły_zamówienia
(
	@idZamówienie INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		rk.idDzień,
		d.Data,
		rk.Liczba_miejsc,
		rk.Liczba_miejsc_dla_studentów,
		'Konferencja ' + k.Nazwa AS Rodzaj_rezerwacji,
		rk.Rezerwacja_anulowana
	FROM Rezerwacje_konferencji rk
		INNER JOIN Dni d ON d.idDzień = rk.idDzień
		INNER JOIN Konferencje k ON d.idKonferencja = k.idKonferencja
	WHERE rk.idZamówienie = @idZamówienie
	UNION
	SELECT
		w.idDzień,
		d2.Data,
		rw.Liczba_miejsc,
		0,
		'Warsztat ' + w.Nazwa AS Rodzaj_rezerwacji,
		rw.Rezerwacja_anulowana
	FROM rezerwacje_warsztatów rw
		INNER JOIN rezerwacje_konferencji rk2 ON rk2.idRezerwacja_konferencji = rw.idRezerwacja_konferencji
		INNER JOIN dni d2 ON d2.idDzień = rk2.idDzień
		INNER JOIN Warsztaty w ON rw.idWarsztat = w.idWarsztat
	WHERE rk2.idZamówienie = @idZamówienie
)
GO
```

7.37. Uczestnicy_danego_dnia

```
CREATE FUNCTION [dbo].[Uczestnicy_danego_dnia]
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.idOsoba,
		o.Imię,
		o.Nazwisko,
		o.Miasto,
		o.Ulica,
		o.Numer_telefonu,
		o.Email,
		o.Data_dodania
	FROM Osoby o
		INNER JOIN Uczestnicy_konferencji uk ON o.idOsoba = uk.idOsoba
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
	WHERE rk.idDzień = @idDzień
)
GO
```

7.38. Uczestnicy_danego_dnia_bez_anulowanych

```
CREATE FUNCTION Uczestnicy_danego_dnia_bez_anulowanych
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.idOsoba,
		o.Imię,
		o.Nazwisko,
		o.Miasto,
		o.Ulica,
		o.Numer_telefonu,
		o.Email,
		o.Data_dodania
	FROM Osoby o
		INNER JOIN Uczestnicy_konferencji uk ON o.idOsoba = uk.idOsoba
		INNER JOIN Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = uk.idRezerwacja_konferencji
	WHERE rk.idDzień = @idDzień AND rk.Rezerwacja_anulowana = 0
	)
GO
```

7.39. Uczestnicy_dnia_konferencji_z_danej_rezerwacji

```
ALTER FUNCTION Uczestnicy_dnia_konferencji_z_danej_rezerwacji
(
	@idRezerwacja_konferencji INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.Imię,
		o.Nazwisko,
		CASE WHEN (uk.Numer_legitymacji IS NOT NULL)
			THEN 'TAK'
		ELSE 'NIE' END AS Student
	FROM Uczestnicy_konferencji uk
		INNER JOIN Rezerwacje_konferencji rk ON uk.idRezerwacja_konferencji = rk.idRezerwacja_konferencji
		INNER JOIN Osoby o ON uk.idOsoba = o.idOsoba
		INNER JOIN dni d ON rk.idDzień = d.idDzień
		INNER JOIN konferencje k ON d.idKonferencja = k.idKonferencja
	WHERE rk.idRezerwacja_konferencji = @idRezerwacja_konferencji AND rk.Rezerwacja_anulowana = 0
)
GO
```

7.40. Uczestnicy_warsztatu_z_danej_rezerwacji

```
FUNCTION Uczestnicy_warsztatu_z_danej_rezerwacji
(
	@idRezerwacja_warsztatu INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		o.Imię,
		o.Nazwisko
	FROM Uczestnicy_warsztatów uw
		INNER JOIN Rezerwacje_warsztatów rw ON uw.idRezerwacja_warsztatu = rw.idRezerwacja_warsztatu
		INNER JOIN warsztaty w ON rw.idWarsztat = w.idWarsztat
		INNER JOIN Uczestnicy_konferencji uk ON uw.idUczestnik_konferencji = uk.idUczestnik_konferencji
		INNER JOIN osoby o ON uk.idOsoba = o.idOsoba
	WHERE rw.idRezerwacja_warsztatu = @idRezerwacja_warsztatu AND rw.Rezerwacja_anulowana = 0
)
GO
```

7.41. Warsztaty_danego_dnia

```
CREATE FUNCTION Warsztaty_danego_dnia
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWarsztat,
		w.idDzień,
		w.Nazwa,
		w.Prowadzący,
		w.Czas_rozpoczęcia,
		w.Czas_zakończenia,
		w.Cena,
		w.Liczba_miejsc
	FROM Dni d
		INNER JOIN Warsztaty w ON w.idDzień = d.idDzień
	WHERE d.idDzień = @idDzień
)
GO
```

7.42. Warsztaty_danej_konferencji

```
CREATE FUNCTION [dbo].[Warsztaty_danej_konferencji]
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWarsztat,
		w.idDzień,
		w.Nazwa,
		w.Prowadzący,
		w.Czas_rozpoczęcia,
		w.Czas_zakończenia,
		w.Cena,
		w.Liczba_miejsc
	FROM Dni d
		INNER JOIN Warsztaty w ON w.idDzień = d.idDzień
	WHERE d.idKonferencja = @idKonferencja
)
GO
```

7.43. Wolne_miejsca_na_warsztatach_danego_dnia

```
CREATE FUNCTION Wolne_miejsca_na_warsztatach_danego_dnia
(
	@idDzień INT
)
RETURNS TABLE
AS
RETURN
(	SELECT
		w.idWarsztat,
		d.Data,
		w.Liczba_miejsc - isnull(sum(rw.Liczba_miejsc), 0) AS wolne_miejsca
	FROM Warsztaty_danego_dnia(@idDzień) AS w
		LEFT JOIN Rezerwacje_warsztatów rw ON rw.idWarsztat = w.idWarsztat
		INNER JOIN Dni d ON d.idDzień = w.idDzień
	WHERE (rw.Rezerwacja_anulowana = 0 OR rw.idRezerwacja_konferencji IS NULL)
	GROUP BY w.idWarsztat, w.Liczba_miejsc, d.Data
)
GO
```

7.44. Wolne_miejsca_na_warsztatach_danej_konferencji

```
ALTER FUNCTION Wolne_miejsca_na_warsztatach_danej_konferencji
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		w.idWarsztat,
		d.Data,
		w.Liczba_miejsc - isnull(sum(rw.Liczba_miejsc), 0) AS wolne_miejsca
	FROM Warsztaty_danej_konferencji(@idKonferencja) AS w
		LEFT JOIN Rezerwacje_warsztatów rw ON rw.idWarsztat = w.idWarsztat
		INNER JOIN Dni d ON d.idDzień = w.idDzień
	WHERE (rw.Rezerwacja_anulowana = 0 OR rw.idRezerwacja_konferencji IS NULL)
	GROUP BY w.idWarsztat, w.Liczba_miejsc, d.Data
)
GO
```

7.45. Wolne_miejsca_w_dniach_danej_konferencji

```
ALTER FUNCTION Wolne_miejsca_w_dniach_danej_konferencji
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		d.idDzień,
		d.Data,
		d.Liczba_miejsc - isnull(sum(rk.Liczba_miejsc), 0) - isnull(sum(rk.Liczba_miejsc_dla_studentów), 0)
			AS wolne_miejsca
	FROM Dni_danej_konferencji(@idKonferencja) d
		LEFT JOIN Rezerwacje_konferencji rk ON rk.idDzień = d.idDzień
	WHERE (rk.Rezerwacja_anulowana = 0 OR rk.idRezerwacja_konferencji IS NULL)
	GROUP BY d.idDzień, d.Data, d.Liczba_miejsc
)
GO
```

7.46. Zamówienia_nieopłacone_danego_klienta

```
CREATE FUNCTION [dbo].[Zamówienia_nieopłacone_danego_klienta]
(
	@idKlient INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		zn.idZamówienie,
		zn.idKlient,
		zn.idKonferencja,
		zn.Nazwa_klienta,
		zn.Nazwa_konferencji,
		zn.Data_zgłoszenia,
		zn.Data_anulowania,
		zn.Zamówienie_anulowane,
		zn.Ile_do_zapłaty
	FROM Zamówienia_nieopłacone zn
	WHERE zn.idKlient = @idKlient
)
GO
```

7.47. Zamówienia_nieopłacone_danej_konferencji

```
CREATE FUNCTION Zamówienia_nieopłacone_danej_konferencji
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT *
	FROM dbo.Zamówienia_nieopłacone zn
	WHERE zn.idKonferencja = @idKonferencja
)
GO
```

7.48. Zamówienia_opłacone_danego_klienta

```
CREATE FUNCTION [dbo].[Zamówienia_opłacone_danego_klienta]
(
	@idKlient INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT
		zo.idZamówienie,
		zo.idKlient,
		zo.idKonferencja,
		zo.Nazwa_klienta,
		zo.Nazwa_konferencji,
		zo.Data_zgłoszenia
	FROM Zamówienia_opłacone zo
	WHERE zo.idKlient = @idKlient
)
GO
```

7.49. Zamówienia_opłacone_danej_konferencji

```
CREATE FUNCTION Zamówienia_opłacone_danej_konferencji
(
	@idKonferencja INT
)
RETURNS TABLE
AS
RETURN
(
	SELECT *
	FROM Zamówienia_opłacone zo
	WHERE zo.idKonferencja = @idKonferencja
)
GO
```

7.50. Zarezerwuj_konferencję

```
CREATE PROCEDURE [dbo].[Zarezerwuj_konferencję]
          @idDzień                    INT,
          @idZamówienia               INT,
          @Liczba_miejsc              INT,
          @Liczba_miejsc_dla_studentów INT
AS
BEGIN
          SET NOCOUNT ON;
          INSERT INTO Rezerwacje_konferencji (idDzień, idZamówienie, Liczba_miejsc, Liczba_miejsc_dla_studentów,
                    Rezerwacja_anulowana)
          VALUES (@idDzień, @idZamówienia, @Liczba_miejsc, @Liczba_miejsc_dla_studentów, 0)
END
GO
```

7.51. Zarezerwuj_warsztat

```
CREATE PROCEDURE [dbo].[Zarezerwuj_warsztat](
          @idWarsztat               INT,
          @idRezerwacja_konferencji INT,
          @Liczba_miejsc            INT

)
AS
BEGIN
          SET NOCOUNT ON;
          INSERT INTO Rezerwacje_warsztatów (idWarsztat, idRezerwacja_konferencji, Liczba_miejsc, Rezerwacja_anulowana)
          VALUES (@idWarsztat, @idRezerwacja_konferencji, @Liczba_miejsc, 0)
END
GO
```

7.52. Złóż_zamówienie

```
CREATE PROCEDURE Złóż_zamówienie(
          @idKlient      INT,
          @idKonferencja INT
)
AS
BEGIN
          SET NOCOUNT ON;
          INSERT INTO Zamówienia (idKlient, idKonferencja, Data_zgłoszenia, Data_anulowania, Zamówienie_anulowane)
          VALUES (@idKlient, @idKonferencja, GETDATE(), NULL, 0)
END
GO
```

8. Widoki

8.1. Anulowane_rezerwacje_dni_konferencji

```
CREATE VIEW [dbo].[Anulowane_rezerwacje_dni_konferencji]
  AS
    SELECT
      rk.idRezerwacja_konferencji,
      rk.idDzień
    FROM Rezerwacje_konferencji rk
    WHERE rk.Rezerwacja_anulowana = 1
GO
```

8.2. Anulowane_warsztaty

```
CREATE VIEW [dbo].[Anulowane_warsztaty]
  AS
    SELECT
      rw.idRezerwacja_warsztatu,
      rw.idWarsztat
    FROM Rezerwacje_warsztatów rw
    WHERE rw.Rezerwacja_anulowana = 1
GO
```

8.3. Anulowane_zamówienia

```
CREATE VIEW [dbo].[Anulowane_zamówienia]
  AS
    SELECT
      z.idZamówienie,
      z.idKlient,
      z.Data_anulowania
    FROM Zamówienia z
    WHERE z.Zamówienie_anulowane = 1
GO
```

8.4. Konferencje_nadchodzące

```
CREATE VIEW Konferencje_nadchodzące
  AS
    SELECT *
    FROM Konferencje k
    WHERE k.Data_rozpoczęcia > getdate()
GO
```

8.5. Konferencje_trwające

```
CREATE VIEW Konferencje_trwające
  AS
    SELECT *
    FROM Konferencje k
    WHERE k.Data_rozpoczęcia <= getdate() AND k.Data_zakończenia >= getdate()
GO
```

8.6. Liczba_brakujących_uczestników_w_rezerwacjach_konferencji

```
CREATE VIEW [dbo].[Liczba_brakujących_uczestników_w_rezerwacjach_konferencji]
  AS
    SELECT
           rk.idRezerwacja_konferencji,
           rk.Liczba_miejsc - (SELECT COUNT(*)
                               FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(rk.idRezerwacja_konferencji)
                               WHERE Student = 'NIE') AS Normalne,
           rk.Liczba_miejsc_dla_studentów - (SELECT COUNT(*)
                                             FROM Uczestnicy_dnia_konferencji_z_danej_rezerwacji(rk.idRezerwacja_konferencji)
                                             WHERE Student = 'TAK') AS Studenckie,
           rk.idDzień
   FROM Rezerwacje_konferencji rk
   WHERE rk.Rezerwacja_anulowana = 0
GO
```

8.7. Liczba_brakujących_uczestników_w_rezerwacjach_warsztatach

```
CREATE VIEW [dbo].[Liczba_brakujących_uczestników_we_wszystkich_warsztatach]
  AS
   SELECT
           rw.idRezerwacja_warsztatu,
           rw.Liczba_miejsc - (SELECT ISNULL(COUNT(*), 0)
                                          FROM Uczestnicy_warsztatu_z_danej_rezerwacji(rw.idRezerwacja_warsztatu)) AS Normalne,
           rk.idDzień
  FROM Rezerwacje_warsztatów rw
           INNER JOIN dbo.Rezerwacje_konferencji rk ON rk.idRezerwacja_konferencji = rw.idRezerwacja_konferencji
  WHERE rw.Rezerwacja_anulowana = 0
GO
```

8.8. Popularność_dni_konferencji

```
CREATE VIEW [dbo].[Popularność_dni_konferencji]
  AS
    SELECT
      d.idDzień,
      d.idKonferencja,
      k.Nazwa,
      d.Data,
      (ISNULL((SELECT SUM(rk.Liczba_miejsc)
               FROM Rezerwacje_konferencji rk
               WHERE rk.idDzień = d.idDzień AND rk.Rezerwacja_anulowana = 0), 0)) AS Liczba_zapisanych,
      d.Liczba_miejsc,
      CAST(ISNULL((SELECT SUM(rk.Liczba_miejsc)
                   FROM Rezerwacje_konferencji rk
                   WHERE rk.idDzień = d.idDzień AND rk.Rezerwacja_anulowana = 0), 0) * 1.0 / d.Liczba_miejsc AS DECIMAL(18, 2))
                   AS Popularność
   FROM Dni d
      INNER JOIN Konferencje k ON d.idKonferencja = k.idKonferencja
GO
```

8.9. Popularność_konferencji

```
CREATE VIEW [dbo].[Popularność_konferencji]
  AS
   SELECT
      k.idKonferencja,
      k.Nazwa,
      k.Data_rozpoczęcia,
      k.Data_zakończenia,
      CAST((isnull((SELECT SUM(d.Liczba_miejsc)
                    FROM Dni d
                    WHERE d.idKonferencja = k.idKonferencja), 0)
           + ISNULL((SELECT SUM(w.Liczba_miejsc)
                       FROM Dni d
                         INNER JOIN Warsztaty w ON w.idDzień = d.idDzień), 0)
           - (ISNULL((SELECT SUM(wm.wolne_miejsca)
                      FROM Wolne_miejsca_w_dniach_danej_konferencji(k.idKonferencja) wm), 0)
           + ISNULL((SELECT SUM(wmw.wolne_miejsca)
                       FROM Wolne_miejsca_na_warsztatach_danej_konferencji(k.idKonferencja) wmw), 0))) * 1.0
           / (ISNULL((SELECT SUM(d.Liczba_miejsc)
                      FROM Dni d
                      WHERE d.idKonferencja = k.idKonferencja), 0)
      +     ISNULL((SELECT SUM(w.Liczba_miejsc)
                    FROM Dni d
                      INNER JOIN Warsztaty w ON w.idDzień = d.idDzień), 0)) AS DECIMAL(18, 2)) AS Popularność
FROM Konferencje k
GO
```

8.10. Popularność_warsztatów

```
CREATE VIEW [dbo].[Popularność_warsztatów]
  AS
    SELECT
      w.idWarsztat,
      k.idKonferencja,
      d.idDzień,
      k.Nazwa AS Nazwa_konferencji,
      w.Nazwa AS Nazwa_warsztatu,
      w.Prowadzący,
      w.Cena,
      w.Liczba_miejsc,
      w.Czas_rozpoczęcia,
      w.Czas_zakończenia,
      d.Data,
      isnull(cast((SELECT sum(rw.Liczba_miejsc)
                   FROM Rezerwacje_warsztatów rw
                   WHERE rw.idWarsztat = w.idWarsztat AND rw.Rezerwacja_anulowana = 0)*1.0/w.Liczba_miejsc AS DECIMAL(18, 2)), 0)
              AS Popularność
    FROM dbo.Warsztaty w
      INNER JOIN dbo.Dni d ON w.idDzień = d.idDzień
      INNER JOIN dbo.Konferencje k ON d.idKonferencja = k.idKonferencja
GO
```

8.11. Suma_wpłat

```
CREATE VIEW [dbo].[Suma_wpłat]
  AS
    SELECT ISNULL(SUM(w.Wartość), 0) AS Suma_wpłat
    FROM Wpłaty w
GO
```

8.12. Suma_wpłat_klientów

```
CREATE VIEW [dbo].[Suma_wpłat_klientów]
  AS
    SELECT
           k.idKlient,
           k.Nazwa,
          isnull((SELECT sum(w.Wartość)
                  FROM Zamówienia z
                    INNER JOIN Wpłaty w ON w.idZamówienie = z.idZamówienie
                  WHERE z.idKlient = k.idKlient), 0) AS Suma_wpłat
    FROM Klienci k
GO
```

8.13. Sumy_wpłat_z_poszczególnych_form_płatności

```
CREATE VIEW [dbo].[Sumy_wpłat_z_poszczególnych_form_płatności]
  AS
   SELECT
      w.idForma_płatności,
      fp.nazwa AS Forma_płatności,
      isnull(sum(w.Wartość), 0) AS Suma_wpłat
    FROM Wpłaty w
      INNER JOIN Formy_płatności fp ON fp.idForma_płatności = w.idForma_płatności
    GROUP BY w.idForma_płatności, fp.nazwa
GO
```

8.14. Uczestnictwo_klientów_w_konferencjach

```
CREATE VIEW [dbo].[Uczestnictwo_klientów_w_konferencjach]
  AS
   SELECT
      k.idKlient,
      k.Nazwa,
      count(DISTINCT z.idKonferencja) AS Ilość_konferencji
    FROM Zamówienia z
      INNER JOIN Klienci k ON k.idKlient = z.idKlient
    GROUP BY k.idKlient, k.Nazwa
GO
```

8.15. Wolne_miejsca_na_dniach_konferencji

```
ALTER VIEW [dbo].[Wolne_miejsca_na_dniach_konferencji]
  AS
   SELECT
    d.idDzień,
    d.idKonferencja,
    isnull((SELECT wol.wolne_miejsca
            FROM Wolne_miejsca_w_dniach_danej_konferencji(d.idKonferencja) wol
            WHERE wol.idDzień = d.idDzień), 0) AS Wolne_miejsca
  FROM Dni d
GO
```

8.16. Wyświetl_firmy

```
ALTER VIEW Wyświetl_firmy
 AS
  SELECT TOP (100) PERCENT
    Nazwa,
    Miasto,
    Ulica,
    Numer_lokalu,
    Kod_pocztowy,
    Numer_telefonu,
    Email
  FROM Klienci
  WHERE (Osoba_prywatna = 0)
GO
```

8.17. Wyświetl_osoby_prywatne

```
CREATE VIEW dbo.Wyświetl_osoby_prywatne
  AS
    SELECT TOP (100) PERCENT
      Nazwa,
      Miasto,
      Ulica,
      Numer_lokalu,
      Kod_pocztowy,
      Numer_telefonu,
      Email
    FROM dbo.Klienci
    WHERE (Osoba_prywatna = 1)
GO
```

8.18. Zamówienia_nieopłacone

```
CREATE VIEW [dbo].[Zamówienia_nieopłacone]
 AS
   SELECT
          z.idZamówienie,
          z.idKlient,
          z.idKonferencja,
          k.nazwa AS Nazwa_klienta,
          ko.nazwa AS Nazwa_konferencji,
          z.Data_zgłoszenia,
          z.Data_anulowania,
          z.Zamówienie_anulowane,
          (SELECT ile
                 FROM Ile_zostało_do_zapłaty(z.idZamówienie)) AS Ile_do_zapłaty
          FROM Zamówienia AS z
            INNER JOIN Klienci k ON k.idKlient = z.idKlient
           INNER JOIN Konferencje ko ON ko.idKonferencja = z.idKonferencja
          WHERE (SELECT ile
                 FROM Ile_zostało_do_zapłaty(z.idZamówienie)) > 0 AND z.Zamówienie_anulowane = 0
GO
```

8.19. Zamówienia_opłacone

```
CREATE VIEW [dbo].[Zamówienia_opłacone]
  AS
    SELECT
           z.idZamówienie,
           z.idKlient,
           z.idKonferencja,
           k.Nazwa  AS Nazwa_klienta,
           ko.nazwa AS Nazwa_konferencji,
           z.Data_zgłoszenia
     FROM Zamówienia AS z
       INNER JOIN klienci k ON k.idKlient = z.idKlient
       INNER JOIN konferencje ko ON ko.idKonferencja = z.idKonferencja
     WHERE (SELECT ile
              FROM Ile_zostało_do_zapłaty(z.idZamówienie)) = 0 AND z.Zamówienie_anulowane = 0
GO
```

9. Generator

```
CREATE PROCEDURE [dbo].[Generuj_dane]
    @Data_startowa                                    DATE = '01-01-2015',
    @Data_końcowa                                     DATE = '01-01-2020',
    @Ile_konferencji                                  INT = 100,
    @Maksymalna_długość_konferencji                   INT = 5,
    @Minimalna_ilość_miejsc_na_dniu                   INT = 10,
    @Maksymalna_ilość_miejsc_na_dniu                  INT = 200,
    @Maksymalna_ilość_progów_cenowych                 INT = 5,
    @Ilość_dni_na_zapisy                              INT = 90,
    @Minimalna_cena_dnia                              DECIMAL(18, 2) = 20,
    @Maksymalna_cena_dnia                             DECIMAL(18, 2) = 200,
    @Maksymalny_procent_ceny_dla_studenta             DECIMAL(18, 2) = 1.0,
    @Minimalny_procent_ceny_dla_studenta              DECIMAL(18, 2) = 0.4,
    @Minimalny_wzrost_ceny_z_każdym_progiem           DECIMAL(18, 2) = 1.2,
    @Maksymalny_wzrost_ceny_z_każdym_progiem          DECIMAL(18, 2) = 2.0,
    @Minimalna_ilość_warsztatów_jednego_dnia          INT = 0,
    @Maksymalna_ilość_warsztatów_jednego_dnia         INT = 6,
    @Minimalna_cena_warsztatu                         DECIMAL(18, 2) = 0,
    @Maksymalna_cena_warsztatu                        DECIMAL(18, 2) = 50,
    @Minimalna_ilość_miejsc_na_warsztacie             INT = 10,
    @Maksymalna_ilość_miejsc_na_warsztacie            INT = 50,
    @Ilość_firm                                       INT = 100,
    @Ilość_osób_prywatnych                            INT = 400,
    @Maksymalna_ilość_zamówień                        INT = 20,
    @Maksymalna_ilość_zarezerwowanych_miejsc          INT = 10,
    @Maksymalna_ilość_zarezerwowanych_miejsc_dla_studentów INT = 10,
    @Jaka_część_osób_z_recyklingu                     DECIMAL(18, 2) = 0.2,
    @Maksymalna_ilość_zarezerwowanych_miejsc_na_warsztat   INT = 10,
    @Podróże_w_czasie_nie_istnieją                    BIT = 1,
    @Wirtualna_data_dzisiejsza                        DATE = '12-12-2017'


AS
  BEGIN
    SET NOCOUNT ON;

    CREATE TABLE Firmy (
      id    INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
      Nazwa VARCHAR(50)         NOT NULL,
      Mail  VARCHAR(50)         NOT NULL,
    )

    CREATE TABLE Adres (
      id     INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
      ulica  VARCHAR(50)         NULL,
      numer  VARCHAR(50)         NULL,
      miasto VARCHAR(50)         NULL,
      kod    VARCHAR(6)          NULL,
      kraj   VARCHAR(50)         NULL,
    )

    CREATE TABLE Imiona (
```

```
  id   INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
  imie VARCHAR(50)         NOT NULL,
)

CREATE TABLE Nazwiska (
  id       INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
  nazwisko VARCHAR(50)         NOT NULL,
)

INSERT INTO Firmy (Nazwa, Mail) VALUES ('2K Czech', 'contact@2KCzech.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('3D Realms', 'contact@3DRealms.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('343 Studios', 'contact@343Studios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('3DO', 'contact@3DO.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('38 Studios', 'contact@38Studios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('4A Games', 'contact@4AGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Acclaim Entertainment', 'contact@AcclaimEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Accolade', 'contact@Accolade.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Activision', 'contact@Activision.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Arc System Works', 'contact@ArcSystemWorks.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Arcane Studios', 'contact@ArcaneStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('ArenaNet', 'contact@ArenaNet.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Ascaron', 'contact@Ascaron.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Atari', 'contact@Atari.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Atlus', 'contact@Atlus.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Avalanche Studios', 'contact@AvalancheStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bethesda Softworks', 'contact@BethesdaSoftworks.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Big Huge Games', 'contact@BigHugeGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('BioWare', 'contact@BioWare.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bizzare Creations', 'contact@BizzareCreations.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Blizzard Entertainment', 'contact@BlizzardEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Blue Byte Software', 'contact@BlueByteSoftware.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Blue Fang Games', 'contact@BlueFangGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bohemia Interactive', 'contact@BohemiaInteractive.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bullfrog Productions', 'contact@BullfrogProductions.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bugbear Entertainment', 'contact@BugbearEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Bungie Studios', 'contact@BungieStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Capcom', 'contact@Capcom.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('CCP Games', 'contact@CCPGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('CD Projekt Red', 'contact@CDProjektRed.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Clap Hanz', 'contact@ClapHanz.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Codemasters', 'contact@Codemasters.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Core Design', 'contact@CoreDesign.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Creative Assembly', 'contact@CreativeAssembly.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Criterion Games', 'contact@CriterionGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Crystal Dynamics', 'contact@CrystalDynamics.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Crytek', 'contact@Crytek.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('CyberConnect2', 'contact@CyberConnect2.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Deadline Games', 'contact@DeadlineGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Digital Illusions CE', 'contact@DigitalIllusionsCE.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Double Fine Productions', 'contact@DoubleFineProductions.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Double Helix Games', 'contact@DoubleHelixGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Egosoft', 'contact@Egosoft.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Eidos Interactive', 'contact@EidosInteractive.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Electronic Arts', 'contact@ElectronicArts.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Epic Games', 'contact@EpicGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Eurocom', 'contact@Eurocom.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Firaxis Games', 'contact@FiraxisGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Firefly Studios', 'contact@FireflyStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Frictional Games', 'contact@FrictionalGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('From Software', 'contact@FromSoftware.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Frozenbyte', 'contact@Frozenbyte.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Gameloft', 'contact@Gameloft.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Game Freak', 'contact@GameFreak.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Gearbox Software', 'contact@GearboxSoftware.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Guerilla Games', 'contact@GuerillaGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('HAL Laboratory', 'contact@HALLaboratory.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Hothead Games', 'contact@HotheadGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('id Software', 'contact@idSoftware.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Ignition Entertainment', 'contact@IgnitionEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Infinity Ward', 'contact@InfinityWard.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Insomaniac Games', 'contact@InsomaniacGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Intelligent Systems', 'contact@IntelligentSystems.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('IO Interactive', 'contact@IOInteractive.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Irrational Games', 'contact@IrrationalGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('JoWood Entertainment AG', 'contact@JoWoodEntertainmentAG.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Konami', 'contact@Konami.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Level5', 'contact@Level5.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Lionhead Studios', 'contact@LionheadStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Looking Glass Studios', 'contact@LookingGlassStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('LucasArts', 'contact@LucasArts.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Maxis Software', 'contact@MaxisSoftware.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Media Molecule', 'contact@MediaMolecule.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Midway Games', 'contact@MidwayGames.pl')
```

```
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Naughty Dog', 'contact@NaughtyDog.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('NCsoft', 'contact@NCsoft.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('NetherRealm Studios', 'contact@NetherRealmStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Neversoft', 'contact@Neversoft.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Nintendo', 'contact@Nintendo.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Oddworld Inhabitants', 'contact@OddworldInhabitants.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Obsidian Entertainment', 'contact@ObsidianEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Paradox Interactive', 'contact@ParadoxInteractive.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Pandemic Studios', 'contact@PandemicStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('People Can Fly', 'contact@PeopleCanFly.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Piranha Bytes', 'contact@PiranhaBytes.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Platinum Games', 'contact@PlatinumGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Polyphony Digital', 'contact@PolyphonyDigital.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('PopCap Games', 'contact@PopCapGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Quantic Dream', 'contact@QuanticDream.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Radical Entertainment', 'contact@RadicalEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Rare Limited', 'contact@RareLimited.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Reality Pump Studios', 'contact@RealityPumpStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Remedy Entertainment', 'contact@RemedyEntertainment.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Rockstar Games', 'contact@RockstarGames.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Sega', 'contact@Sega.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Snowblind Studios', 'contact@SnowblindStudios.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Square Enix', 'contact@SquareEnix.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Sucker Punch Productions', 'contact@SuckerPunchProductions.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Team17', 'contact@Team17.pl')
INSERT INTO Firmy (Nazwa, Mail) VALUES ('Techland', 'contact@Techland.pl')
INSERT INTO Adres VALUES ('ul. Bruska', '64', 'Łódź', '94223', 'Polska')
INSERT INTO Adres VALUES ('ul. Michałowicza Jerzego', '114', 'Łódź', '94217', 'Polska')
INSERT INTO Adres VALUES ('ul. Jeziorko', '5', 'Kraków', '31987', 'Polska')
INSERT INTO Adres VALUES ('ul. Komety', '113', 'Szczecin', '70700', 'Polska')
INSERT INTO Adres VALUES ('ul. Leszno', '71', 'Warszawa', '01175', 'Polska')
INSERT INTO Adres VALUES ('ul. Pocztowa', '38', 'Darłowo', '76150', 'Polska')
INSERT INTO Adres VALUES ('ul. Droga Golfowa', '94', 'Warszawa', '04865', 'Polska')
INSERT INTO Adres VALUES ('ul. Krótka', '90', 'Warszawa', '02293', 'Polska')
INSERT INTO Adres VALUES ('ul. Krótka', '122', 'Warszawa', '02293', 'Polska')
INSERT INTO Adres VALUES ('ul. Mokra', '95', 'Rzeszów', '35111', 'Polska')
INSERT INTO Adres VALUES ('ul. Odkrywców', '27', 'Wrocław', '53212', 'Polska')
INSERT INTO Adres VALUES ('ul. Beskidzka', '101', 'Rybnik', '44200', 'Polska')
INSERT INTO Adres VALUES ('ul. Robocza', '150', 'Poznań', '61517', 'Polska')
INSERT INTO Adres VALUES ('ul. Księdza Knosały', '107', 'Radzionków', '41922', 'Polska')
INSERT INTO Adres VALUES ('ul. Bluszczowa', '50', 'Łódź', '91858', 'Polska')
INSERT INTO Adres VALUES ('ul. Tracka', '110', 'Olsztyn', '10365', 'Polska')
INSERT INTO Adres VALUES ('ul. Puławska', '118', 'Warszawa', '02740', 'Polska')
INSERT INTO Adres VALUES ('ul. Kossaka Juliusza', '118', 'Olsztyn', '10349', 'Polska')
INSERT INTO Adres VALUES ('ul. Werbeny', '78', 'Warszawa', '04997', 'Polska')
INSERT INTO Adres VALUES ('ul. Parysa', '96', 'Lublin', '20712', 'Polska')
INSERT INTO Adres VALUES ('ul. Perłowa', '115', 'Lublin', '20574', 'Polska')
INSERT INTO Adres VALUES ('ul. Nowoursynowska', '19', 'Warszawa', '02776', 'Polska')
INSERT INTO Adres VALUES ('ul. Chmielewskiego Gracjana', '54', 'Lublin', '20620', 'Polska')
INSERT INTO Adres VALUES ('ul. Jaworowa', '130', 'KędzierzynKoźle', '47220', 'Polska')
INSERT INTO Adres VALUES ('ul. Dworcowa', '54', 'Bytów', '77100', 'Polska')
INSERT INTO Adres VALUES ('ul. Kiemliczów', '87', 'Poznań', '60174', 'Polska')
INSERT INTO Adres VALUES ('ul. Raginisa Władysława', '1', 'Białystok', '15161', 'Polska')
INSERT INTO Adres VALUES ('ul. Kurniki', '74', 'Kraków', '31156', 'Polska')
INSERT INTO Adres VALUES ('ul. Szczypiorkowa', '148', 'Warszawa', '02752', 'Polska')
INSERT INTO Adres VALUES ('ul. Sasanek', '62', 'Katowice', '40750', 'Polska')
INSERT INTO Adres VALUES ('ul. Piastów', '142', 'Rzeszów', '35077', 'Polska')
INSERT INTO Adres VALUES ('ul. Przejazd', '53', 'Białystok', '15430', 'Polska')
INSERT INTO Adres VALUES ('ul. Balladyny', '145', 'Zielona Góra', '65323', 'Polska')
INSERT INTO Adres VALUES ('ul. Jeleniogórska', '47', 'Poznań', '60179', 'Polska')
INSERT INTO Adres VALUES ('ul. Murarzy', '18', 'Bydgoszcz', '85804', 'Polska')
INSERT INTO Adres VALUES ('ul. Wenedów', '40', 'Koszalin', '75847', 'Polska')
INSERT INTO Adres VALUES ('ul. Włościańska', '97', 'Szczecin', '70021', 'Polska')
INSERT INTO Adres VALUES ('ul. Przesmyk', '130', 'Bydgoszcz', '85035', 'Polska')
INSERT INTO Adres VALUES ('ul. Skarżyńskiego Stanisława', '45', 'Warszawa', '02377', 'Polska')
INSERT INTO Adres VALUES ('ul. Dziwnowska', '97', 'Poznań', '60456', 'Polska')
INSERT INTO Adres VALUES ('ul. Ciasna', '70', 'Szczecin', '70747', 'Polska')
INSERT INTO Adres VALUES ('ul. Dworzec PKP Żabieniec', '118', 'Łódź', '91340', 'Polska')
INSERT INTO Adres VALUES ('ul. Kasztanowa', '8', 'Katowice', '40670', 'Polska')
INSERT INTO Adres VALUES ('ul. Abrahama Antoniego', '105', 'Gdynia', '81366', 'Polska')
INSERT INTO Adres VALUES ('ul. Mostowa', '50', 'Murowana Goślina', '62095', 'Polska')
INSERT INTO Adres VALUES ('ul. Kostrzyńska', '27', 'Warszawa', '02979', 'Polska')
INSERT INTO Adres VALUES ('ul. Zakręt', '93', 'Szczecin', '70754', 'Polska')
INSERT INTO Adres VALUES ('ul. Chełmońskiego Józefa', '4', 'Racibórz', '47400', 'Polska')
INSERT INTO Adres VALUES ('ul. Izabelli', '66', 'Warszawa', '01738', 'Polska')
INSERT INTO Adres VALUES ('ul. Wigury Stanisława', '150', 'Łódź', '90319', 'Polska')
INSERT INTO Adres VALUES ('ul. Poznańska', '32', 'Krośniewice', '99340', 'Polska')
INSERT INTO Adres VALUES ('ul. Wróblewskiego Walerego', '114', 'Koszalin', '75076', 'Polska')
INSERT INTO Adres VALUES ('ul. Czerska', '140', 'Poznań', '60446', 'Polska')
INSERT INTO Adres VALUES ('ul. Melisy', '3', 'Warszawa/Wesoła', '05077', 'Polska')
INSERT INTO Adres VALUES ('ul. Spławie', '51', 'Poznań', '61312', 'Polska')
INSERT INTO Adres VALUES ('ul. Wał Zawadowski', '15', 'Warszawa', '02986', 'Polska')
INSERT INTO Adres VALUES ('ul. Kartuska', '35', 'Gdańsk', '80298', 'Polska')
```

```
INSERT INTO Adres VALUES ('ul. Opatowicka', '95', 'Tarnowskie Góry', '4261', 'Polska')
INSERT INTO Adres VALUES ('ul. Balicka', '10', 'Kraków', '30199', 'Polska')
INSERT INTO Adres VALUES ('ul. Błękitna', '67', 'Łódź', '93322', 'Polska')
INSERT INTO Adres VALUES ('ul. ŻegiestówZdrój', '91', 'Żegiestów', '33360', 'Polska')
INSERT INTO Adres VALUES ('ul. Spokojna', '53', 'Siemianowice Śląskie', '41100', 'Polska')
INSERT INTO Adres VALUES ('ul. Nawrot', '134', 'Łódź', '90008', 'Polska')
INSERT INTO Adres VALUES ('ul. Gdańska', '136', 'Olsztyn', '10254', 'Polska')
INSERT INTO Adres VALUES ('Pl. Lipowy', '100', 'Poznań', '61478', 'Polska')
INSERT INTO Adres VALUES ('ul. Ukryta', '129', 'Wrocław', '50334', 'Polska')
INSERT INTO Adres VALUES ('ul. Zielińskiego doktora', '71', 'Zabrze', '41800', 'Polska')
INSERT INTO Adres VALUES ('ul. Lasek Brzozowy', '142', 'Warszawa', '02792', 'Polska')
INSERT INTO Adres VALUES ('ul. Jesienna', '52', 'Lublin', '20337', 'Polska')
INSERT INTO Adres VALUES ('ul. Alsa Roderyka', '24', 'Rzeszów', '35030', 'Polska')
INSERT INTO Adres VALUES ('ul. Marusarzówny Heleny', '75', 'Łódź', '94041', 'Polska')
INSERT INTO Adres VALUES ('ul. Cmentarna', '126', 'KędzierzynKoźle', '47200', 'Polska')
INSERT INTO Adres VALUES ('ul. Leszka Białego', '112', 'KędzierzynKoźle', '47232', 'Polska')
INSERT INTO Adres VALUES ('ul. Lęborska', '24', 'Łódź', '92713', 'Polska')
INSERT INTO Adres VALUES ('Pl. Obrońców Katowic', '90', 'Katowice', '40091', 'Polska')
INSERT INTO Adres VALUES ('ul. Nałkowskiej Zofii', '144', 'Siemianowice Śląskie', '41100', 'Polska')
INSERT INTO Adres VALUES ('ul. Głęboka', '137', 'Łódź', '92201', 'Polska')
INSERT INTO Adres VALUES ('ul. Strusia', '85', 'Białystok', '15539', 'Polska')
INSERT INTO Adres VALUES ('ul. Afrykańska', '90', 'Gdynia', '81118', 'Polska')
INSERT INTO Adres VALUES ('ul. Generała Hallera Józefa', '94', 'Gdynia', '81427', 'Polska')
INSERT INTO Adres VALUES ('ul. Błogosławionego Ładysława z Gielniowa', '135', 'Warszawa', '02066', 'Polska')
INSERT INTO Adres VALUES ('ul. Podchorążych', '127', 'Lublin', '20811', 'Polska')
INSERT INTO Adres VALUES ('ul. Szamarzewskiego Augustyna', '94', 'Poznań', '60531', 'Polska')
INSERT INTO Adres VALUES ('ul. Potulicka', '39', 'Bydgoszcz', '85376', 'Polska')
INSERT INTO Adres VALUES ('ul. Władysława Łokietka', '6', 'Sopot', '81735', 'Polska')
INSERT INTO Adres VALUES ('ul. Adwokacka', '39', 'Łódź', '91305', 'Polska')
INSERT INTO Adres VALUES ('ul. Sierakowskiego Zygmunta', '36', 'Wrocław', '51678', 'Polska')
INSERT INTO Adres VALUES ('ul. Mosdorfa Jana', '129', 'Kraków', '31618', 'Polska')
INSERT INTO Adres VALUES ('Al. Zjednoczenia', '135', 'Zielona Góra', '65109', 'Polska')
INSERT INTO Adres VALUES ('ul. Rejtana Tadeusza', '142', 'Kraków', '30510', 'Polska')
INSERT INTO Adres VALUES ('ul. Dąbrowskiego Jarosława', '105', 'Gliwice', '44100', 'Polska')
INSERT INTO Adres VALUES ('ul. Bernardyńska', '10', 'Gliwice', '44102', 'Polska')
INSERT INTO Adres VALUES ('ul. Skwarczyńskiej Stefanii', '29', 'Łódź', '91175', 'Polska')
INSERT INTO Adres VALUES ('ul. Okrzei Stefana', '40', 'Mysłowice', '41406', 'Polska')
INSERT INTO Adres VALUES ('ul. Matejki Jana', '87', 'Koszalin', '75544', 'Polska')
INSERT INTO Adres VALUES ('ul. Czarnkowska', '2', 'Bydgoszcz', '85411', 'Polska')
INSERT INTO Adres VALUES ('ul. Wigury Stanisława', '50', 'Dąbrowa Górnicza', '41300', 'Polska')
INSERT INTO Adres VALUES ('ul. Hutnicza', '49', 'Jaworzno', '43602', 'Polska')
INSERT INTO Adres VALUES ('ul. Maciejewicza Konstantego', '1', 'Szczecin', '71004', 'Polska')
INSERT INTO Adres VALUES ('ul. Jeżynowa', '86', 'Szczecin', '70892', 'Polska')
INSERT INTO Adres VALUES ('ul. Kołosa Włodzimierza', '131', 'Warszawa', '02093', 'Polska')
INSERT INTO Adres VALUES ('ul. Marszałkowska', '101', 'Warszawa', '00517', 'Polska')
INSERT INTO Adres VALUES ('ul. Gliniana', '79', 'Racibórz', '47406', 'Polska')
INSERT INTO Adres VALUES ('Al. Wojska Polskiego', '150', 'Warszawa', '01515', 'Polska')
INSERT INTO Adres VALUES ('ul. Słowiańska', '71', 'Szczecin', '71463', 'Polska')
INSERT INTO Adres VALUES ('ul. Długa', '119', 'Białystok', '15765', 'Polska')
INSERT INTO Adres VALUES ('ul. Pinokia', '117', 'Zielona Góra', '65012', 'Polska')
INSERT INTO Adres VALUES ('ul. Rumińskiego Bolesława', '49', 'Bydgoszcz', '85030', 'Polska')
INSERT INTO Adres VALUES ('ul. Przyszkole', '132', 'Łódź', '93549', 'Polska')
INSERT INTO Adres VALUES ('ul. Hetmańska', '145', 'Chorzów', '41516', 'Polska')
INSERT INTO Adres VALUES ('ul. Syreny', '43', 'Kraków', '31216', 'Polska')
INSERT INTO Adres VALUES ('ul. Czwartaków', '74', 'Płock', '09403', 'Polska')
INSERT INTO Adres VALUES ('ul. Sienkiewicza Henryka', '127', 'Poznań', '60818', 'Polska')
INSERT INTO Adres VALUES ('ul. Krzewowa', '131', 'Rzeszów', '35232', 'Polska')
INSERT INTO Adres VALUES ('ul. Piłsudskiego', '103', 'Zawiercie', '42403', 'Polska')
INSERT INTO Adres VALUES ('ul. Wróblewskiego Walerego', '120', 'Opole', '45760', 'Polska')
INSERT INTO Adres VALUES ('ul. Wierzbięcice', '125', 'Poznań', '61548', 'Polska')
INSERT INTO Adres VALUES ('ul. Krzemieniecka', '130', 'Lublin', '20130', 'Polska')
INSERT INTO Adres VALUES ('Al. Tysiąclecia Państwa Polskiego', '74', 'Białystok', '15111', 'Polska')
INSERT INTO Adres VALUES ('ul. Batorego Stefana', '51', 'Bydgoszcz', '85104', 'Polska')
INSERT INTO Adres VALUES ('ul. Warzywna', '127', 'Katowice', '40545', 'Polska')
INSERT INTO Adres VALUES ('ul. Pocztowa', '132', 'Nakło nad Notecią', '89100', 'Polska')
INSERT INTO Adres VALUES ('ul. Morgowa', '42', 'Warszawa', '04224', 'Polska')
INSERT INTO Adres VALUES ('ul. PawlikowskiejJasnorzewskiej Marii', '51', 'Poznań', '60461', 'Polska')
INSERT INTO Adres VALUES ('ul. Profesora Musiała Włodzimierza', '127', 'Łódź', '93365', 'Polska')
INSERT INTO Adres VALUES ('ul. Chodkiewicza Karola Jana', '69', 'Bydgoszcz', '85065', 'Polska')
INSERT INTO Adres VALUES ('ul. Czeremchowa', '124', 'Szczecin', '70763', 'Polska')
INSERT INTO Adres VALUES ('ul. Grunwaldzka', '74', 'Rybnik', '44210', 'Polska')
INSERT INTO Adres VALUES ('ul. Śliska', '112', 'Poznań', '61369', 'Polska')
INSERT INTO Adres VALUES ('ul. Ceglana', '87', 'Opole', '45811', 'Polska')
INSERT INTO Adres VALUES ('ul. Długoszyńska', '71', 'Jaworzno', '43602', 'Polska')
INSERT INTO Adres VALUES ('ul. Grodziska', '104', 'Warszawa', '01255', 'Polska')
INSERT INTO Adres VALUES ('ul. Szczytnowska', '73', 'Olsztyn', '10629', 'Polska')
INSERT INTO Adres VALUES ('ul. Grzybowa', '51', 'Ruda Śląska', '41707', 'Polska')
INSERT INTO Adres VALUES ('ul. Trygłowa', '38', 'Szczecin', '70780', 'Polska')
INSERT INTO Adres VALUES ('ul. Hajduki', '113', 'Świętochłowice', '41600', 'Polska')
INSERT INTO Adres VALUES ('ul. Iłżecka', '84', 'Warszawa', '02135', 'Polska')
INSERT INTO Adres VALUES ('ul. Wolborska', '114', 'Warszawa', '02196', 'Polska')
INSERT INTO Adres VALUES ('ul. Generała Pełczyńskiego Tadeusza', '92', 'Warszawa', '01471', 'Polska')
INSERT INTO Adres VALUES ('ul. Nawrot', '69', 'Łódź', '90039', 'Polska')
```

```
INSERT INTO Adres VALUES ('ul. Sokoła', '67', 'Szczecin', '71691', 'Polska')
INSERT INTO Adres VALUES ('ul. Babimojska', '116', 'Wrocław', '54426', 'Polska')
INSERT INTO Adres VALUES ('ul. Lawinowa', '137', 'Łódź', '92010', 'Polska')
INSERT INTO Adres VALUES ('ul. Wirskiego', '52', 'Chełm', '22116', 'Polska')
INSERT INTO Adres VALUES ('ul. Marynarska', '94', 'Warszawa', '02674', 'Polska')
INSERT INTO Adres VALUES ('ul. Księdza Ligudy Alojzego', '88', 'Opole', '45109', 'Polska')
INSERT INTO Adres VALUES ('ul. Opalenicka', '39', 'Poznań', '60358', 'Polska')
INSERT INTO Adres VALUES ('ul. Menonitów', '27', 'Gdańsk', '80805', 'Polska')
INSERT INTO Adres VALUES ('ul. Aldony', '107', 'Gdynia', '81524', 'Polska')
INSERT INTO Adres VALUES ('ul. Berwińskiego Ryszarda', '45', 'Poznań', '60765', 'Polska')
INSERT INTO Adres VALUES ('ul. Rzecha', '96', 'Rzeszów', '35322', 'Polska')
INSERT INTO Adres VALUES ('ul. Królewska', '86', 'Lublin', '20109', 'Polska')
INSERT INTO Adres VALUES ('ul. Skrajna', '147', 'Rzeszów', '35231', 'Polska')
INSERT INTO Adres VALUES ('Pl. Wolności', '7', 'Dobromierz', '58170', 'Polska')
INSERT INTO Adres VALUES ('ul. Rybnicka', '12', 'JastrzębieZdrój', '44335', 'Polska')
INSERT INTO Adres VALUES ('ul. Sobótki', '14', 'Rzeszów', '35302', 'Polska')
INSERT INTO Adres VALUES ('ul. Kolektorska', '109', 'Warszawa', '01692', 'Polska')
INSERT INTO Adres VALUES ('ul. Ebro', '48', 'Warszawa', '01490', 'Polska')
INSERT INTO Adres VALUES ('ul. Brzozowa', '149', 'Gdynia', '81515', 'Polska')
INSERT INTO Adres VALUES ('ul. Szuwarowa', '49', 'Łódź', '91356', 'Polska')
INSERT INTO Adres VALUES ('ul. Smosarskiej Jadwigi', '86', 'Rzeszów', '35602', 'Polska')
INSERT INTO Adres VALUES ('ul. Grenadierów', '37', 'Kraków', '30085', 'Polska')
INSERT INTO Adres VALUES ('ul. Koszalińska', '48', 'Olsztyn', '10622', 'Polska')
INSERT INTO Adres VALUES ('Pl. 11 Listopada', '27', 'Goniądz', '19110', 'Polska')
INSERT INTO Adres VALUES ('ul. Chętnika Adama', '45', 'Białystok', '15166', 'Polska')
INSERT INTO Adres VALUES ('ul. Mikołowska', '62', 'Racibórz', '47400', 'Polska')
INSERT INTO Adres VALUES ('ul. Bieczyńskiego Feliksa', '50', 'Lublin', '20073', 'Polska')
INSERT INTO Adres VALUES ('ul. Winiary', '75', 'Poznań', '60665', 'Polska')
INSERT INTO Adres VALUES ('ul. Biasa Wojciecha', '35', 'Opole', '45751', 'Polska')
INSERT INTO Adres VALUES ('ul. Minogi', '16', 'Gdańsk', '80840', 'Polska')
INSERT INTO Adres VALUES ('ul. Borysławska', '93', 'Opole', '45316', 'Polska')
INSERT INTO Adres VALUES ('ul. Ramułta Stefana', '27', 'Gdańsk', '80061', 'Polska')
INSERT INTO Adres VALUES ('ul. Irysów', '99', 'KędzierzynKoźle', '47224', 'Polska')
INSERT INTO Adres VALUES ('ul. Odrodzenia', '44', 'Lubin', '59300', 'Polska')
INSERT INTO Adres VALUES ('ul. Dąbrowskiego', '130', 'Poznań', '60908', 'Polska')
INSERT INTO Adres VALUES ('ul. Bałtycka', '13', 'Warszawa', '03237', 'Polska')
INSERT INTO Adres VALUES ('ul. Konwaliowa', '85', 'Białystok', '15674', 'Polska')
INSERT INTO Adres VALUES ('ul. Kawalerii', '99', 'Warszawa', '00468', 'Polska')
INSERT INTO Adres VALUES ('ul. Bohaterów', '31', 'Poznań', '61852', 'Polska')
INSERT INTO Adres VALUES ('ul. Wiatraczna', '117', 'Kraków', '31987', 'Polska')
INSERT INTO Adres VALUES ('ul. Bacciarellego Marcelego', '59', 'Warszawa', '00591', 'Polska')
INSERT INTO Adres VALUES ('ul. Bielicha', '41', 'Radom', '26601', 'Polska')
INSERT INTO Adres VALUES ('ul. Bałtycka', '83', 'Olsztyn', '11041', 'Polska')
INSERT INTO Adres VALUES ('ul. Główna', '12', 'Ruda Śląska', '41710', 'Polska')
INSERT INTO Adres VALUES ('ul. Kościuszki Tadeusza', '52', 'Mikołów', '43190', 'Polska')
INSERT INTO Adres VALUES ('ul. Zgórska', '50', 'Kielce', '25827', 'Polska')
INSERT INTO Adres VALUES ('ul. Bernardyńska', '55', 'Rzeszów', '35069', 'Polska')
INSERT INTO Adres VALUES ('ul. Tkaczy', '18', 'Warszawa', '01346', 'Polska')
INSERT INTO Adres VALUES ('ul. Minkusa Krystiana', '46', 'Opole', '45592', 'Polska')
INSERT INTO Adres VALUES ('ul. Malczewskiego Jacka', '130', 'Kraków', '30207', 'Polska')
INSERT INTO Adres VALUES ('ul. Żniwna', '79', 'Lublin', '20621', 'Polska')
INSERT INTO Adres VALUES ('ul. Malachitowa', '22', 'Bydgoszcz', '85369', 'Polska')
INSERT INTO Adres VALUES ('ul. Gęsia', '117', 'Łódź', '91457', 'Polska')
INSERT INTO Adres VALUES ('ul. Drabowe Bagno', '25', 'Jaworzno', '43608', 'Polska')
INSERT INTO Adres VALUES ('ul. Grzegorza Zarugiewicza', '117', 'Zielona Góra', '65518', 'Polska')
INSERT INTO Adres VALUES ('ul. Zajęcza', '109', 'Warszawa', '00356', 'Polska')
INSERT INTO Adres VALUES ('ul. Belwederska', '62', 'Warszawa', '00762', 'Polska')
INSERT INTO Adres VALUES ('ul. Ulanowska', '8', 'Warszawa', '04859', 'Polska')
INSERT INTO Adres VALUES ('ul. Biwakowa', '135', 'Poznań', '60480', 'Polska')
INSERT INTO Adres VALUES ('ul. Piotrowskiego Maksymiliana', ' 60', 'Bydgoszcz', '85098', 'Polska')
INSERT INTO Imiona (Imie) VALUES ('Bogusław')
INSERT INTO Imiona (Imie) VALUES ('Zachariasz')
INSERT INTO Imiona (Imie) VALUES ('Urszula')
INSERT INTO Imiona (Imie) VALUES ('Izaak')
INSERT INTO Imiona (Imie) VALUES ('Bartosz')
INSERT INTO Imiona (Imie) VALUES ('Korneli')
INSERT INTO Imiona (Imie) VALUES ('Dobrogost')
INSERT INTO Imiona (Imie) VALUES ('Makary')
INSERT INTO Imiona (Imie) VALUES ('Władysław')
INSERT INTO Imiona (Imie) VALUES ('Klaudia')
INSERT INTO Imiona (Imie) VALUES ('Fryderyka')
INSERT INTO Imiona (Imie) VALUES ('Czesław')
INSERT INTO Imiona (Imie) VALUES ('Aleksy')
INSERT INTO Imiona (Imie) VALUES ('Barbara')
INSERT INTO Imiona (Imie) VALUES ('Maurycy')
INSERT INTO Imiona (Imie) VALUES ('Grażyna')
INSERT INTO Imiona (Imie) VALUES ('Aureliusz')
INSERT INTO Imiona (Imie) VALUES ('Jozafat')
INSERT INTO Imiona (Imie) VALUES ('Sergiusz')
INSERT INTO Imiona (Imie) VALUES ('Julek')
INSERT INTO Imiona (Imie) VALUES ('Mirosława')
INSERT INTO Imiona (Imie) VALUES ('Jaropełk')
INSERT INTO Imiona (Imie) VALUES ('Grzegorz')
```

```
INSERT INTO Imiona (Imie) VALUES ('Henryka')
INSERT INTO Imiona (Imie) VALUES ('Jerzy')
INSERT INTO Imiona (Imie) VALUES ('Wiktoria')
INSERT INTO Imiona (Imie) VALUES ('Miłogost')
INSERT INTO Imiona (Imie) VALUES ('Roman')
INSERT INTO Imiona (Imie) VALUES ('Bronisława')
INSERT INTO Imiona (Imie) VALUES ('Celestyna')
INSERT INTO Imiona (Imie) VALUES ('Katarzyna')
INSERT INTO Imiona (Imie) VALUES ('Friderich')
INSERT INTO Imiona (Imie) VALUES ('Ania')
INSERT INTO Imiona (Imie) VALUES ('Basia')
INSERT INTO Imiona (Imie) VALUES ('Idzi')
INSERT INTO Imiona (Imie) VALUES ('Jarek')
INSERT INTO Imiona (Imie) VALUES ('Wojtek')
INSERT INTO Imiona (Imie) VALUES ('Julita')
INSERT INTO Imiona (Imie) VALUES ('Antoni')
INSERT INTO Imiona (Imie) VALUES ('Wiga')
INSERT INTO Imiona (Imie) VALUES ('Alojzy')
INSERT INTO Imiona (Imie) VALUES ('Wera')
INSERT INTO Imiona (Imie) VALUES ('Marceli')
INSERT INTO Imiona (Imie) VALUES ('Rajmund')
INSERT INTO Imiona (Imie) VALUES ('Eustachy')
INSERT INTO Imiona (Imie) VALUES ('Krzyś')
INSERT INTO Imiona (Imie) VALUES ('Gustaw')
INSERT INTO Imiona (Imie) VALUES ('Zygfryd')
INSERT INTO Imiona (Imie) VALUES ('Szymon')
INSERT INTO Imiona (Imie) VALUES ('Krysia')
INSERT INTO Imiona (Imie) VALUES ('Eliasz')
INSERT INTO Imiona (Imie) VALUES ('Mikolaj')
INSERT INTO Imiona (Imie) VALUES ('Ryszard')
INSERT INTO Imiona (Imie) VALUES ('Bogusław')
INSERT INTO Imiona (Imie) VALUES ('Alfons')
INSERT INTO Imiona (Imie) VALUES ('Urjasz')
INSERT INTO Imiona (Imie) VALUES ('Krystyn')
INSERT INTO Imiona (Imie) VALUES ('Feliks')
INSERT INTO Imiona (Imie) VALUES ('Metody')
INSERT INTO Imiona (Imie) VALUES ('Angelika')
INSERT INTO Imiona (Imie) VALUES ('Lesław')
INSERT INTO Imiona (Imie) VALUES ('Anastazy')
INSERT INTO Imiona (Imie) VALUES ('Anastazja')
INSERT INTO Imiona (Imie) VALUES ('Felicyta')
INSERT INTO Imiona (Imie) VALUES ('Irenka')
INSERT INTO Imiona (Imie) VALUES ('Władysław')
INSERT INTO Imiona (Imie) VALUES ('Hanna')
INSERT INTO Imiona (Imie) VALUES ('Karolina')
INSERT INTO Imiona (Imie) VALUES ('Wiga')
INSERT INTO Imiona (Imie) VALUES ('Henryka')
INSERT INTO Imiona (Imie) VALUES ('Wisława')
INSERT INTO Imiona (Imie) VALUES ('Walentyna')
INSERT INTO Imiona (Imie) VALUES ('Przemek')
INSERT INTO Imiona (Imie) VALUES ('Katarzyna')
INSERT INTO Imiona (Imie) VALUES ('Krystiana')
INSERT INTO Imiona (Imie) VALUES ('Krystian')
INSERT INTO Imiona (Imie) VALUES ('Idzi')
INSERT INTO Imiona (Imie) VALUES ('Franciszek')
INSERT INTO Imiona (Imie) VALUES ('Malwina')
INSERT INTO Imiona (Imie) VALUES ('Dyta')
INSERT INTO Imiona (Imie) VALUES ('Michał')
INSERT INTO Imiona (Imie) VALUES ('Miłosz')
INSERT INTO Imiona (Imie) VALUES ('Bogumił')
INSERT INTO Imiona (Imie) VALUES ('Maryla')
INSERT INTO Imiona (Imie) VALUES ('Lidia')
INSERT INTO Imiona (Imie) VALUES ('Justyn')
INSERT INTO Imiona (Imie) VALUES ('Malina')
INSERT INTO Imiona (Imie) VALUES ('Dominik')
INSERT INTO Imiona (Imie) VALUES ('Alicja')
INSERT INTO Imiona (Imie) VALUES ('Radzimierz')
INSERT INTO Imiona (Imie) VALUES ('Gracja')
INSERT INTO Imiona (Imie) VALUES ('Lesława')
INSERT INTO Imiona (Imie) VALUES ('Gabryś')
INSERT INTO Imiona (Imie) VALUES ('Dobrosława')
INSERT INTO Imiona (Imie) VALUES ('Ludwika')
INSERT INTO Imiona (Imie) VALUES ('Tobiasz')
INSERT INTO Imiona (Imie) VALUES ('Teodozja')
INSERT INTO Imiona (Imie) VALUES ('Jolanta')
INSERT INTO Imiona (Imie) VALUES ('Izaak')
INSERT INTO Imiona (Imie) VALUES ('Kunegunda')
INSERT INTO Imiona (Imie) VALUES ('Hipolit')
INSERT INTO Imiona (Imie) VALUES ('Władysława')
INSERT INTO Imiona (Imie) VALUES ('Gabryjel')
INSERT INTO Imiona (Imie) VALUES ('Przemysław')
INSERT INTO Imiona (Imie) VALUES ('Mikolaj')
INSERT INTO Imiona (Imie) VALUES ('Malwina')
```

```
INSERT INTO Imiona (Imie) VALUES ('Weronika')
INSERT INTO Imiona (Imie) VALUES ('Karol')
INSERT INTO Imiona (Imie) VALUES ('Hainrich')
INSERT INTO Imiona (Imie) VALUES ('Gertruda')
INSERT INTO Imiona (Imie) VALUES ('Józefa')
INSERT INTO Imiona (Imie) VALUES ('Jarosława')
INSERT INTO Imiona (Imie) VALUES ('Wiktoria')
INSERT INTO Imiona (Imie) VALUES ('Gabrjel')
INSERT INTO Imiona (Imie) VALUES ('Maurycy')
INSERT INTO Imiona (Imie) VALUES ('Aureliusz')
INSERT INTO Imiona (Imie) VALUES ('Andrzej')
INSERT INTO Imiona (Imie) VALUES ('Luiza')
INSERT INTO Imiona (Imie) VALUES ('Klara')
INSERT INTO Imiona (Imie) VALUES ('Drugi')
INSERT INTO Imiona (Imie) VALUES ('Irenka')
INSERT INTO Imiona (Imie) VALUES ('Ksenia')
INSERT INTO Imiona (Imie) VALUES ('Wiga')
INSERT INTO Imiona (Imie) VALUES ('Zygmunt')
INSERT INTO Imiona (Imie) VALUES ('Julianna')
INSERT INTO Imiona (Imie) VALUES ('Kacper')
INSERT INTO Imiona (Imie) VALUES ('Miłosław')
INSERT INTO Imiona (Imie) VALUES ('Zosia')
INSERT INTO Imiona (Imie) VALUES ('Eliasz')
INSERT INTO Imiona (Imie) VALUES ('Beatrycze')
INSERT INTO Imiona (Imie) VALUES ('Zdzisława')
INSERT INTO Imiona (Imie) VALUES ('Dorota')
INSERT INTO Imiona (Imie) VALUES ('Juliusz')
INSERT INTO Imiona (Imie) VALUES ('Świętopełk')
INSERT INTO Imiona (Imie) VALUES ('Przemysł')
INSERT INTO Imiona (Imie) VALUES ('Alicja')
INSERT INTO Imiona (Imie) VALUES ('Bogumiła')
INSERT INTO Imiona (Imie) VALUES ('Konstanty')
INSERT INTO Imiona (Imie) VALUES ('Henio')
INSERT INTO Imiona (Imie) VALUES ('Mateusz')
INSERT INTO Imiona (Imie) VALUES ('Celina')
INSERT INTO Imiona (Imie) VALUES ('Serafina')
INSERT INTO Imiona (Imie) VALUES ('Ludwika')
INSERT INTO Imiona (Imie) VALUES ('Dorota')
INSERT INTO Imiona (Imie) VALUES ('Lechosław')
INSERT INTO Imiona (Imie) VALUES ('Henryka')
INSERT INTO Imiona (Imie) VALUES ('Benedykt')
INSERT INTO Imiona (Imie) VALUES ('Hieronim')
INSERT INTO Imiona (Imie) VALUES ('Anastazy')
INSERT INTO Imiona (Imie) VALUES ('Miłosław')
INSERT INTO Imiona (Imie) VALUES ('Justyna')
INSERT INTO Imiona (Imie) VALUES ('Tytus')
INSERT INTO Imiona (Imie) VALUES ('Waleria')
INSERT INTO Imiona (Imie) VALUES ('Mieczysław')
INSERT INTO Imiona (Imie) VALUES ('Patrycja')
INSERT INTO Imiona (Imie) VALUES ('Julita')
INSERT INTO Imiona (Imie) VALUES ('Bożydar')
INSERT INTO Imiona (Imie) VALUES ('Lubomir')
INSERT INTO Imiona (Imie) VALUES ('Miron')
INSERT INTO Imiona (Imie) VALUES ('Fryderyk')
INSERT INTO Imiona (Imie) VALUES ('Sylwester')
INSERT INTO Imiona (Imie) VALUES ('Sulisław')
INSERT INTO Imiona (Imie) VALUES ('Teodozja')
INSERT INTO Imiona (Imie) VALUES ('Metody')
INSERT INTO Imiona (Imie) VALUES ('Klimek')
INSERT INTO Imiona (Imie) VALUES ('Waleria')
INSERT INTO Imiona (Imie) VALUES ('Ludwik')
INSERT INTO Imiona (Imie) VALUES ('Janek')
INSERT INTO Imiona (Imie) VALUES ('Dobrosława')
INSERT INTO Imiona (Imie) VALUES ('Donat')
INSERT INTO Imiona (Imie) VALUES ('Wiola')
INSERT INTO Imiona (Imie) VALUES ('Lidia')
INSERT INTO Imiona (Imie) VALUES ('Luiza')
INSERT INTO Imiona (Imie) VALUES ('Konstanty')
INSERT INTO Imiona (Imie) VALUES ('Konstanty')
INSERT INTO Imiona (Imie) VALUES ('Grażyna')
INSERT INTO Imiona (Imie) VALUES ('Sobiesław')
INSERT INTO Imiona (Imie) VALUES ('Zdzisława')
INSERT INTO Imiona (Imie) VALUES ('Hajnrich')
INSERT INTO Imiona (Imie) VALUES ('Czcibor')
INSERT INTO Imiona (Imie) VALUES ('Wielisław')
INSERT INTO Imiona (Imie) VALUES ('Juliusz')
INSERT INTO Imiona (Imie) VALUES ('Edyta')
INSERT INTO Imiona (Imie) VALUES ('Michalina')
INSERT INTO Imiona (Imie) VALUES ('Gertruda')
INSERT INTO Imiona (Imie) VALUES ('Przemko')
INSERT INTO Imiona (Imie) VALUES ('Mieczysława')
INSERT INTO Imiona (Imie) VALUES ('Janusz')
INSERT INTO Imiona (Imie) VALUES ('Klimek')
```

```
INSERT INTO Imiona (Imie) VALUES ('Drugi')
INSERT INTO Imiona (Imie) VALUES ('Augustyn')
INSERT INTO Imiona (Imie) VALUES ('Bolek')
INSERT INTO Imiona (Imie) VALUES ('Dominik')
INSERT INTO Imiona (Imie) VALUES ('Cezar')
INSERT INTO Imiona (Imie) VALUES ('Klimek')
INSERT INTO Imiona (Imie) VALUES ('Bartosz')
INSERT INTO Imiona (Imie) VALUES ('Przemek')
INSERT INTO Imiona (Imie) VALUES ('Kasia')
INSERT INTO Imiona (Imie) VALUES ('Augustyna')
INSERT INTO Imiona (Imie) VALUES ('Bogumiła')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Duda')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kwiatkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Walczak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Rakoczy')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Zając')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Rutkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Pawlak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Maciejewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Gorska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Michalska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Sokołowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Olszewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wysocka')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Majewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Majewska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Zając')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wieczorek')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Adamczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Woźniak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kalinowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Majewska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wysocki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Jasińska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kowalski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Szewczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wiśniewska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Olszewska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Grabowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kowalczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Maciejewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Zielinski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Olszewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Adamska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Pawłowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kucharska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Symanski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wysocka')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Piotrowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Król')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kwiatkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Symanski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowicki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kowalczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kalinowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Czarnecka')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Zawadzki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wieczorek')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Grusza')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Sawicki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wysocki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Płot')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Olszewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowakowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Borkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Zielinska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowakowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Grabowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kucharska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Piotrowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Rutkowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Rutkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Walczak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Borkowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Królik')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Majewska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Michalska')
```

```
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kowalczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kaczmarek')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Tomaszewski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kurczak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kaczmarek')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kamińska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Kwiatkowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Jasiński')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Adamczyk')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Sobczak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wysocki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowakowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Duda')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Ostrowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowicki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Michalska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Borkowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Nowicki')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Pawłowska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Walczak')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Wojciechowski')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Szczepańska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Gorska')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Dąbrowsi')
INSERT INTO Nazwiska (Nazwisko) VALUES ('Jasińska')


DELETE FROM Uczestnicy_warsztatów
DBCC CHECKIDENT (Uczestnicy_warsztatów, RESEED, 0)
DELETE FROM Uczestnicy_konferencji
DBCC CHECKIDENT (Uczestnicy_konferencji, RESEED, 0)
DELETE FROM Osoby
DBCC CHECKIDENT (Osoby, RESEED, 0)
DELETE FROM Rezerwacje_warsztatów
DBCC CHECKIDENT (Rezerwacje_warsztatów, RESEED, 0)
DELETE FROM Warsztaty
DBCC CHECKIDENT (Warsztaty, RESEED, 0)
DELETE FROM Rezerwacje_konferencji
DBCC CHECKIDENT (Rezerwacje_konferencji, RESEED, 0)
DELETE FROM Ceny
DBCC CHECKIDENT (Ceny, RESEED, 0)
DELETE FROM Dni
DBCC CHECKIDENT (Dni, RESEED, 0)
DELETE FROM Wpłaty
DBCC CHECKIDENT (Wpłaty, RESEED, 0)
DELETE FROM Zamówienia
DBCC CHECKIDENT (Zamówienia, RESEED, 0)
DELETE FROM Konferencje
DBCC CHECKIDENT (Konferencje, RESEED, 0)
DELETE FROM Klienci
DBCC CHECKIDENT (Klienci, RESEED, 0)
DELETE FROM Formy_płatności
DBCC CHECKIDENT (Formy_płatności, RESEED, 0)


INSERT INTO Formy_płatności (Nazwa) VALUES ('Gotówka')
INSERT INTO Formy_płatności (Nazwa) VALUES ('Karta')
INSERT INTO Formy_płatności (Nazwa) VALUES ('Przelew')
INSERT INTO Formy_płatności (Nazwa) VALUES ('Do łapy')


DECLARE @IKlient INT = 1
WHILE (@IKlient <= 100)
  BEGIN
    INSERT INTO Klienci (Osoba_prywatna, Nazwa, Miasto, Ulica, Numer_lokalu, Kod_pocztowy, Numer_telefonu, Email)
    VALUES (0, (SELECT nazwa
                FROM Firmy
                WHERE id = @IKlient), (SELECT miasto
                                       FROM adres
                                       WHERE id = cast(rand() * 100 + 1 AS INT)),
            (SELECT ulica
             FROM adres
             WHERE id = cast(rand() * 100 + 1 AS INT)), (SELECT numer
                                                         FROM adres
                                                         WHERE id = cast(rand() * 100 + 1 AS INT)),
            (SELECT kod
             FROM adres
             WHERE id = cast(rand() * 100 + 1 AS INT)), cast(rand() * 888888888 + 111111111 AS INT),
            (SELECT replace(nazwa, ' ', '')
             FROM Firmy
```

```
            WHERE id = @IKlient) + '@gmail.com')
    SET @IKlient = @IKlient + 1
  END
SET @IKlient = 0
WHILE (@IKlient < @Ilość_osób_prywatnych)
  BEGIN
    DECLARE @Imię VARCHAR(50) = (SELECT imie
                                 FROM imiona
                                 WHERE id = cast(rand() * 100 + 1 AS INT))
    DECLARE @Nazwisko VARCHAR(50) = (SELECT nazwisko
                                     FROM nazwiska
                                     WHERE id = cast(rand() * 100 + 1 AS INT))
    INSERT INTO Klienci (Osoba_prywatna, Nazwa, Miasto, Ulica, Numer_lokalu, Kod_pocztowy, Numer_telefonu, Email)
    VALUES (1, @Imię + ' ' + @Nazwisko,
            (SELECT miasto
             FROM adres
             WHERE id = cast(rand() * 100 + 1 AS INT)),
            (SELECT ulica
             FROM adres
             WHERE id = cast(rand() * 100 + 1 AS INT)), (SELECT numer
                                                         FROM adres
                                                         WHERE id = cast(rand() * 100 + 1 AS INT)),
            (SELECT kod
             FROM adres
             WHERE id = cast(rand() * 100 + 1 AS INT)), cast(rand() * 888888888 + 111111111 AS INT),
           @Imię + '.' + @Nazwisko + '@gmail.com')
    SET @IKlient = @IKlient + 1
  END


DECLARE @IKonferencja INT = 0
WHILE (@IKonferencja < @Ile_konferencji)
  BEGIN
    DECLARE @Różnica_w_dniach INT = datediff(DAY, @Data_startowa, @Data_końcowa)
    DECLARE @idMiejsca_konferencji INT = cast(rand() * 100 + 1 AS INT)
    DECLARE @Długość INT = cast(rand() * (@Maksymalna_długość_konferencji - 1) + 1 AS INT)

    DECLARE @Dzień_rozpoczęcia INT = rand() * @Różnica_w_dniach + 1
    DECLARE @Początek_zapisów DATE = dateadd(DAY, @Dzień_rozpoczęcia - @Ilość_dni_na_zapisy, @Data_startowa)


    INSERT INTO Konferencje (Data_rozpoczęcia, Data_zakończenia, Nazwa, Miejsce)
    VALUES (dateadd(DAY, @Dzień_rozpoczęcia, @Data_startowa),
            dateadd(DAY, @Dzień_rozpoczęcia + @Długość, @Data_startowa), 'To Do',
            (SELECT Miasto
             FROM Adres
             WHERE id = @idMiejsca_konferencji))

    DECLARE @idKonferencji INT = @@Identity
    DECLARE @IDzień INT = 0

    WHILE (@IDzień <= @Długość)
      BEGIN
        DECLARE @Ilość_miejsc_w_dniu INT
        SET @Ilość_miejsc_w_dniu = cast(
            rand() * (@Maksymalna_ilość_miejsc_na_dniu - @Minimalna_ilość_miejsc_na_dniu) +
            @Minimalna_ilość_miejsc_na_dniu AS INT)
        INSERT INTO Dni (idKonferencja, [Data], Liczba_miejsc)
        VALUES (@idKonferencji, dateadd(DAY, @IDzień, dateadd(DAY, @Dzień_rozpoczęcia, @Data_startowa)),
                @Ilość_miejsc_w_dniu)

        DECLARE @idDnia INT = @@identity
        DECLARE @Cena_początkowa DECIMAL(18, 2) = rand() * (@Maksymalna_cena_dnia - @Minimalna_cena_dnia) +
                                                  @Minimalna_cena_dnia
        DECLARE @Zniżka DECIMAL(18, 2) = rand() *
                                         (@Maksymalny_procent_ceny_dla_studenta -
                                          @Minimalny_procent_ceny_dla_studenta) +
                                         @Minimalny_procent_ceny_dla_studenta

        INSERT INTO Ceny (idDzień, Data, Cena, Cena_dla_studenta)
        VALUES (@idDnia, @Początek_zapisów, @Cena_początkowa, @Cena_początkowa * @Zniżka)

        DECLARE @ICena INT = 0
        DECLARE @Ilość_progów INT = cast(rand() * (@Maksymalna_ilość_progów_cenowych - 1) AS INT)
        WHILE (@ICena < @Ilość_progów)
          BEGIN
            DECLARE @Co_ile_nowy_próg INT =
            datediff(DAY, @Początek_zapisów, dateadd(DAY, @Dzień_rozpoczęcia, @Data_startowa)) / @ilość_progów
            SET @Cena_początkowa = @Cena_początkowa * (rand() *
                                                       (@Maksymalny_wzrost_ceny_z_każdym_progiem -
                                                        @Minimalny_wzrost_ceny_z_każdym_progiem) +
                                                       @Minimalny_wzrost_ceny_z_każdym_progiem)
            BEGIN TRY
```

```
        INSERT INTO Ceny (idDzień, Data, Cena, Cena_dla_studenta)
        VALUES (@idDnia, dateadd(DAY, (@ICena + 1) * @Co_ile_nowy_próg, @Początek_zapisów), @Cena_początkowa,
                @Cena_początkowa * @Zniżka)
        END TRY
        BEGIN CATCH
        END CATCH

        SET @ICena = @ICena + 1
      END

    DECLARE @Ile_warsztatów INT = cast(rand() *
                                       (@Maksymalna_ilość_warsztatów_jednego_dnia -
                                        @Minimalna_ilość_warsztatów_jednego_dnia) +
                                       @Minimalna_ilość_warsztatów_jednego_dnia AS INT)
    DECLARE @IWarsztat INT = 0
    WHILE (@IWarsztat < @Ile_warsztatów)
      BEGIN
        DECLARE @Rand_imię INT = cast(rand() * 99 + 1 AS INT)
        DECLARE @Rand_nazwisko INT = cast(rand() * 99 + 1 AS INT)
        DECLARE @Godzina_początkowa INT = cast(rand() * 13 + 8 AS INT)
        DECLARE @Czas_trwania INT = cast(rand() * 3 + 1 AS INT)
        BEGIN TRY
        INSERT INTO Warsztaty (idDzień, Nazwa, Prowadzący, Liczba_miejsc, Czas_rozpoczęcia, Czas_zakończenia, Cena)
        VALUES (@idDnia, 'To Do',
                (SELECT imie
                 FROM imiona
                 WHERE id = @Rand_imię) + ' ' + (SELECT nazwisko
                                                 FROM nazwiska
                                                 WHERE id = @Rand_nazwisko),
                cast(rand() * (@Maksymalna_ilość_miejsc_na_warsztacie - @Minimalna_ilość_miejsc_na_warsztacie) +
                     @Minimalna_ilość_miejsc_na_warsztacie AS INT),
                cast(right('0' + cast(@Godzina_początkowa AS VARCHAR(50)), 2) + ':00:00' AS DATETIME2(7)),
                cast(right('0' + cast(@Godzina_początkowa + @Czas_trwania AS VARCHAR(50)), 2) + ':00:00' AS
                     DATETIME2(7)),
                rand() * (@Maksymalna_cena_warsztatu - @Minimalna_cena_warsztatu) + @Minimalna_cena_warsztatu)
        END TRY
        BEGIN CATCH
        END CATCH

        SET @IWarsztat = @IWarsztat + 1
      END


    SET @IDzień = @IDzień + 1
  END

DECLARE @Ilość_zamówień INT = cast(rand() * (@Maksymalna_ilość_zamówień + 1) AS INT)
DECLARE @IZamówienie INT = 0
WHILE (@IZamówienie < @Ilość_zamówień)
  BEGIN
    DECLARE @Data_zgłoszenia DATE = dateadd(DAY, rand() * (@Ilość_dni_na_zapisy - 14), @Początek_zapisów)
    IF (@Podróże_w_czasie_nie_istnieją = 1 AND
        cast(@Data_zgłoszenia AS DATE) < cast(@Wirtualna_data_dzisiejsza AS DATE))
      BEGIN
        INSERT INTO Zamówienia (idKlient, idKonferencja, Data_zgłoszenia, Data_anulowania, Zamówienie_anulowane)
        VALUES (rand() * (@Ilość_firm + @Ilość_osób_prywatnych) + 1,
                @IdKonferencji,
                @Data_zgłoszenia, NULL, 0)
        DECLARE @idZamówienia INT = @@identity

        ----------------------------------------------------------------------
        DECLARE @Ile_wpłat INT
        DECLARE @Rand DECIMAL(18, 2) = rand()

        IF @Rand > 0.6
          BEGIN SET @Ile_wpłat = 1 END
        ELSE IF @Rand > 0.15
          BEGIN SET @Ile_wpłat = 2 END
        ELSE IF @Rand > 0.05
          BEGIN SET @Ile_wpłat = 3 END
        ELSE BEGIN SET @Ile_wpłat = 0 END

        DECLARE @IWpłata INT = 0
        ----------------------------------------------------------------------

        DECLARE @Dzień_pewniak INT = cast(rand() * (@Długość + 1) AS INT)
        DECLARE @IDzień2 INT = 0
        WHILE (@IDzień2 <= @Długość)
          BEGIN
            IF (@IDzień = @Dzień_pewniak OR rand() > 0.5)
              BEGIN
                DECLARE @Ile_normalnych INT = cast(rand() * (@Maksymalna_ilość_zarezerwowanych_miejsc + 1) AS
                                                   INT)
```

```
DECLARE @Ile_ulgowych INT = cast(
    rand() * (@Maksymalna_ilość_zarezerwowanych_miejsc_dla_studentów + 1) AS INT)

DECLARE @idTego_dnia INT = (SELECT idDzień
                            FROM Dni
                            WHERE idKonferencja = @idKonferencji AND
                                  Data =
                                  dateadd(DAY, @Dzień_rozpoczęcia + @IDzień2, @Data_startowa))

BEGIN TRY
INSERT INTO Rezerwacje_konferencji (idDzień, idZamówienie, Liczba_miejsc, Liczba_miejsc_dla_studentów,
                                    Rezerwacja_anulowana)
VALUES (@idTego_dnia, @idZamówienia, @Ile_normalnych,
        CASE WHEN @Ile_normalnych > 0
          THEN @Ile_ulgowych
        ELSE @Ile_ulgowych + 1 END, 0)

DECLARE @idRezerwacji_konferencji INT = @@identity
IF (@Ile_normalnych = 0)
  BEGIN SET @Ile_ulgowych = @Ile_ulgowych + 1 END
DECLARE @IUk INT = 0
DECLARE @Ile_w_sumie INT = @Ile_normalnych + @Ile_ulgowych
DECLARE @Ile_osób_dostępnych INT = 0
WHILE (@IUk < @Ile_w_sumie)
  BEGIN
    DECLARE @Rand2 DECIMAL(18, 2) = rand()
    IF (@Rand2 < @Jaka_część_osób_z_recyklingu AND @Ile_osób_dostępnych > 0)
      BEGIN
        DECLARE @Wybrana_osoba INT = cast(rand() * @Ile_osób_dostępnych + 1 AS INT)
        IF exists(
            SELECT uk.idOsoba
            FROM Uczestnicy_konferencji uk
            WHERE uk.idOsoba = @Wybrana_osoba AND
                  uk.idRezerwacja_konferencji = @idRezerwacji_konferencji)
          BEGIN
            SET @Rand2 = @Rand2 + 1
          END
        ELSE
          BEGIN
            INSERT INTO Uczestnicy_konferencji (idOsoba, idRezerwacja_konferencji, Numer_legitymacji)
            VALUES (@Wybrana_osoba, @idRezerwacji_konferencji,
                    (CASE WHEN @Ile_normalnych = 0
                      THEN cast(rand() * (899999 + 1) + 100000 AS INT)
                     ELSE NULL END))
          END
      END

    IF (@Rand >= @Jaka_część_osób_z_recyklingu OR @Ile_osób_dostępnych = 0)
      BEGIN
        DECLARE @Imię2 NCHAR(10) = (SELECT i.imie
                                    FROM imiona i
                                    WHERE i.id = cast(rand() * 199 + 1 AS INT))
        DECLARE @Nazwisko2 NCHAR(10) = (SELECT n.nazwisko
                                        FROM nazwiska n
                                        WHERE n.id = cast(rand() * 99 + 1 AS INT))


        INSERT INTO Osoby (Imię, Nazwisko, Miasto, Ulica, Numer_telefonu, Data_dodania, Email)
        VALUES (@Imię2, @Nazwisko2,
                cast((SELECT miasto
                      FROM adres
                      WHERE id = cast(rand() * 99 + 1 AS INT)) AS NCHAR(10)),
                cast((SELECT ulica
                      FROM adres
                      WHERE id = cast(rand() * 99 + 1 AS INT)) AS NCHAR(10)),
                cast(rand() * 888888888 + 111111111 AS INT),
                dateadd(DAY, cast(rand() * (
                  @Ilość_dni_na_zapisy - datediff(DAY, @Początek_zapisów, @Data_zgłoszenia) -
                  14) AS INT),
                       @Data_zgłoszenia), cast(cast(rand() * 10000000 AS INT) AS NCHAR(10)))


        DECLARE @idOsoby INT = @@identity
        SET @Ile_osób_dostępnych = @Ile_osób_dostępnych + 1
        INSERT INTO Uczestnicy_konferencji (idOsoba, idRezerwacja_konferencji, Numer_legitymacji)
        VALUES (@idOsoby, @idRezerwacji_konferencji,
                CASE WHEN @Ile_normalnych = 0
                  THEN cast(rand() * (899999 + 1) + 100000 AS INT)
                ELSE NULL END)
      END

    IF (@Ile_normalnych = 0)
      BEGIN SET @Ile_ulgowych = @Ile_ulgowych - 1 END
```

```
                ELSE BEGIN SET @Ile_normalnych = @Ile_normalnych - 1 END
                SET @IUk = @IUk + 1
              END

            DECLARE @Ile_warsztatów_tego_dnia INT = (SELECT count(*)
                                                     FROM Warsztaty
                                                     WHERE idDzień = @idTego_dnia)
            DECLARE @IWarsztat2 INT = 0
            WHILE (@IWarsztat2 < @Ile_warsztatów_tego_dnia)
              BEGIN
                IF (rand() > 0.5)
                  BEGIN
                    DECLARE @idWarsztatu INT = (
                      SELECT idWarsztat
                      FROM (SELECT
                              ROW_NUMBER()
                              OVER (
                                ORDER BY idWarsztat ASC ) AS wiersz,
                              idWarsztat
                            FROM Warsztaty
                            WHERE idDzień = @idTego_dnia
                           ) AS chrzan
                      WHERE wiersz = @IWarsztat2 + 1)
                    BEGIN TRY
                    INSERT INTO Rezerwacje_warsztatów (idWarsztat, idRezerwacja_konferencji, Liczba_miejsc,
                                                       Rezerwacja_anulowana)
                    VALUES (@idWarsztatu, @idRezerwacji_konferencji,
                            cast(rand() * @Maksymalna_ilość_zarezerwowanych_miejsc_na_warsztat + 1 AS INT),
                            0)
                    END TRY
                    BEGIN CATCH
                    END CATCH

                  END

                SET @IWarsztat2 = @IWarsztat2 + 1
              END

            END TRY
            BEGIN CATCH
            END CATCH
          END


        SET @IDzień2 = @IDzień2 + 1
      END


    END
    SET @IZamówienie = @IZamówienie + 1
  END

  SET @IKonferencja = @IKonferencja + 1

END

DROP TABLE Adres
DROP TABLE Firmy
DROP TABLE Imiona
DROP TABLE Nazwiska


END
GO
```

10. Role

W naszej bazie danych powinny zostać zdefiniowane następujące role:
- administrator -pełny dostęp do bazy
- obsługa - dostęp do widoków i procedur informacyjnych, przede wszystkim do widoków związanych z płatnościami oraz brakującymi zgłoszeniami, dostęp do funkcji, obsługa ma udzielać pomocy właścicielowi firmy, może go wyręczyć w niektórych zadaniach

- właściciel firmy - może tworzyć konferencje i warsztaty, usuwać/anulować własne wydarzenia, modyfikować ich dane, dostęp do wszystkich widoków i procedur związanych z utworzonymi konferencjami + wszystko to co klient
- klient - może robić/anulować rezerwacje na konferencje i warsztaty, zgłaszać dowolną liczbę osób na nie, korzystać z funkcji mówiącej o jego płatnościach, użyć funkcji zwracającej informację ile jeszcze pozostało do zapłaty za daną konferencję + klient może być uczestnikiem
- uczestnik - może sprawdzać na co jest zapisany, może stać się klientem, korzystać z procedury do edycji własnych danych

11. Job

- Dodatkowo: funkcja wykonująca się automatycznie co 24h i anulująca zamówienia, które nie zostały opłacone w terminie

```
USE msdb ;
GO

EXEC msdb.dbo.sp_add_job
@job_name = N'Sprawdź czy opłacono zamówienia czy anulować';
GO

EXEC sp_add_jobstep
@job_name = N'Sprawdź czy opłacono zamówienia czy anulować',
@step_name = N'Sprawdź warunek i wykonaj procedurę',
@subsystem = N'TSQL',
@command =
N'declare @id int
declare cur CURSOR LOCAL for

 SELECT *
FROM Zamówienia as z
where (datediff(DAY, z.Data_zgłoszenia, getdate()) > 7)
   and z.Data_anulowania is null
   and ((select Ile from Ile_zostało_do_zapłaty (@id)) > 0)

open cur
fetch next from cur into @id

while @@FETCH_STATUS = 0 BEGIN

exec Anuluj_zamówienie @id
fetch next from cur into @id
END

close cur
deallocate cur';
GO

EXEC dbo.sp_add_schedule
 @schedule_name = N'Codziennie',
 @freq_type = 4,
 @active_start_time = 000001 ,
 @freq_interval = 1;

USE msdb ;
GO
EXEC sp_attach_schedule
@job_name = N'Sprawdź czy opłacono zamówienia czy anulować',
@schedule_name = N'Codziennie';
GO

EXEC dbo.sp_add_jobserver
@job_name = N'Sprawdzanie czy anulowac rezerwacje';
GO
```