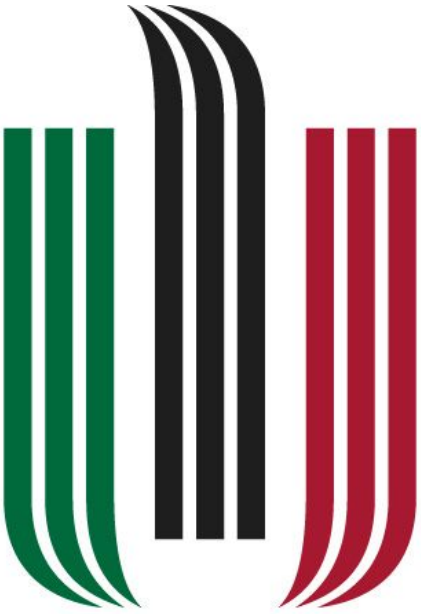

AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Podstawy baz danych

Projekt: Konferencje

Krzysztof Bieniasz, Mateusz Nabywaniec

Spis treści

1. System zarządzania konferencjami

1.1 Opis problemu

Projekt dotyczy systemu wspomagania działalności firmy organizującej konferencje:

1.2 Ogólne informacje

Firma organizuje konferencje, które mogą być jedno- lub kilkudniowe. Klienci
powinni móc rejestrować się na konferencje za pomocą systemu www. 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 ilość miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić - a jeśli sama nie uzupełni do tego czasu, to pracownicy
dzwonią do firmy i ustalają takie informacje). Każdy uczestnik konferencji otrzymuje identyfikator imienny (+ ew. informacja o firmie na nim). Dla konferencji kilkudniowych, uczestnicy mogą rejestrować się na dowolne z tych dni.
Warsztaty
Ponadto z konferencją związane są warsztaty, na które uczestnicy także mogą się zarejestrować - muszą być jednak zarejestrowani tego dnia na konferencję, aby móc w nich uczestniczyć. Kilka warsztatów może trwać równocześnie, ale uczestnik nie może zarejestrować się na więcej niż jeden warsztat, który trwa w tym samym czasie. Jest także ograniczona ilość miejsc na każdy warsztat i na każdy dzień konferencji. Część warsztatów może być płatna, a część jest darmowa.

1.3 Opłaty

Opłata za udział w konferencji zależy nie tylko od zarezerwowanych usług, ale także od terminu ich rezerwacji - jest kilka progów ceny (progi ceny dotyczą tylko udziału w konferencji, cena warsztatów jest stała) i im bliżej rozpoczęcia konferencji, tym cena jest wyższa (jest także zniżka procentowa dla studentów i w takim wypadku przy rezerwacji trzeba podać nr legitymacji studenckiej). Na zapłatę klienci mają tydzień od rezerwacji na konferencję - jeśli do tego czasu nie pojawi się opłata, rezerwacja jest anulowana.
Raporty
Dla organizatora najbardziej istotne są listy osobowe uczestników na każdy dzień konferencji i na każdy warsztat, a także informacje o płatnościach klientów. Ponadto organizator chciałby mieć informację o klientach, którzy najczęściej korzystają z jego usług.

1.4 Specyfika firmy

Firma organizuje średnio 2 konferencje w miesiącu, każda z nich trwa zwykle 2-3 dni, w tym średnio w każdym dniu są 4 warsztaty. Na każdą konferencję średnio rejestruje się 200 osób. Stworzona baza danych powinna zostać wypełniona w takim stopniu, aby odpowiadała 3-letniej działalności firmy.

2. Schemat bazy

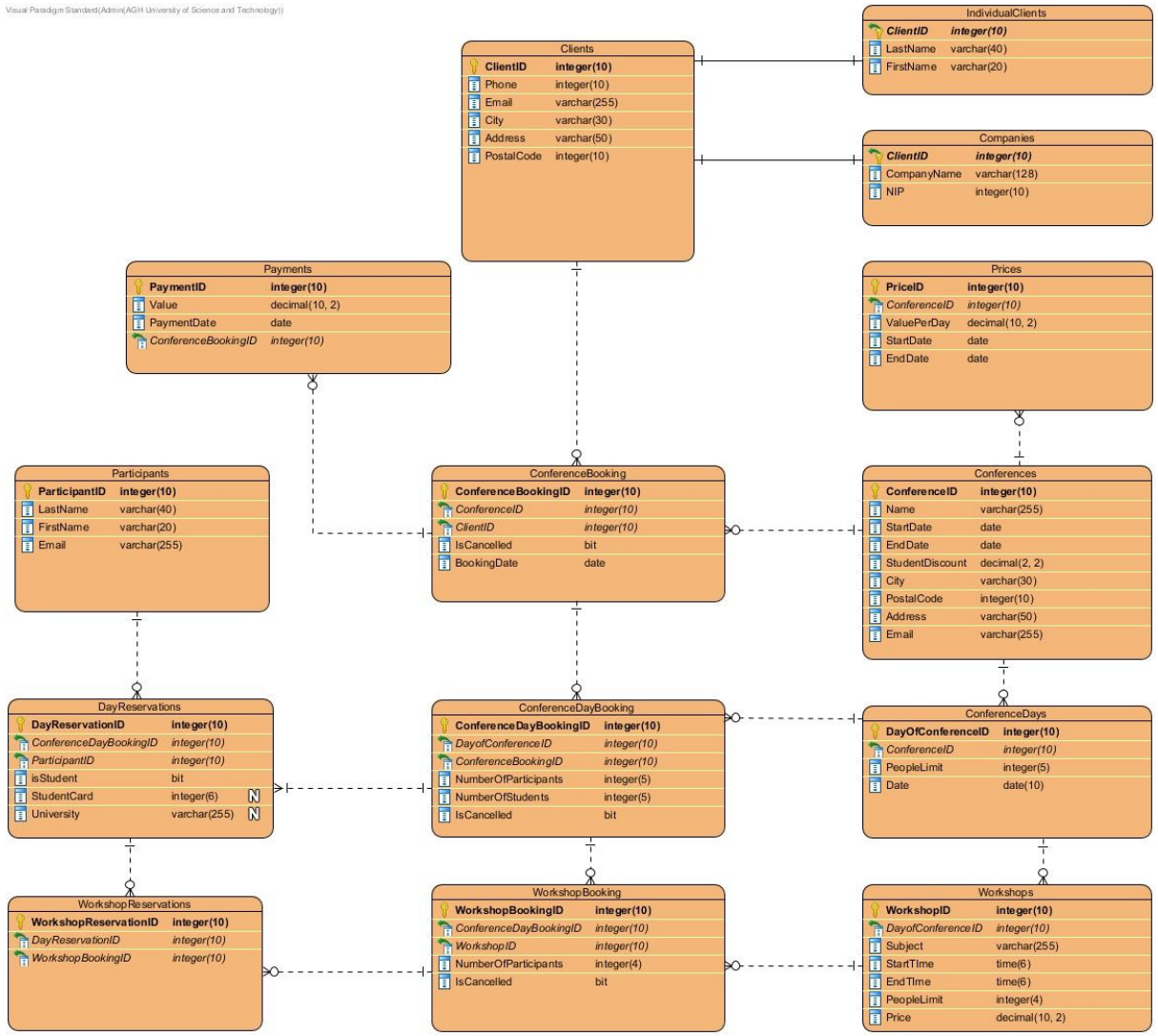

Tabela Conferences

Zawiera informacji o wszystkich organizowanych konferencjach

ConferenceID – Identyfikator konferencji
Name – nazwa konferencji
StartDate – data początku konferencji
EndDate – data końca konferencji
StudentDiscount – wartość zniżki dla studentów (wartość od 0 do 1)
City – miasto, w którym odbywa się konferencja
PostalCode - kod pocztowy
Address – ulica, na której znajduje się budynek, w którym odbywa się konferencja
Email – adres mailowy firmy, dla której organizowana jest konferencja

Tabela Prices

Zawiera informację o cenie za pojedynczy dzień konferencji w poszczególnych przedziałach czasowych

PriceID – identyfikator ceny konferencji
ConferenceID – identyfikator konferencji, którego dotyczy dana cena
ValuePerDay – cena za jeden dzień konferencji (przyjmujemy, że każdy dzień konferencji kosztuje tyle samo)
StartDate – data początku obowiązywania danej ceny
EndDate – data końca obowiązywania danej ceny

Tabela Clients

Zawiera informację o klientach

ClientID – identyfikator klienta
Phone – numer telefonu klienta
Email – adress mailowy klienta
Address – ulica i nr posesji klienta
City – miasto
PostalCode – kod pocztowy

Tabela IndividualClients

Zawiera dane klientów indywidualnych, które nie zostały ujęte w tabeli Clients

ClientID - identyfikator klienta
LastName – nazwisko klienta
FirstName – imię klienta

Tabela Companies

Zawiera dane firm będących klientami organizacji organizującej szkolenia, które nie zostały ujęte w tabeli clients

ClientID - identyfikator klienta
CompanyName – nazwa firmy
NIP – numer identyfikacji podatkowej firmy

Tabela ConferenceDays

Zawiera informacje na temat poszczególnych dni konferencji

DayOfConferenceID – identyfikator dnia konferencji
ConferenceID – identyfikator konferencji, która zawiera w sobie konkretny dzień konferencji
PeopleLimit – limit uczestników mogących się zapisać na dany dzień konferencji
Date – data konkretnego dnia konferencji

Tabela Workshops

Tabela zawierająca informację o warsztatach przeprowadzanych podczas konferencji

WorkshopID – identyfikator warszatu
DayOfConference ID – identyfikat dnia konferencji, w trakcie którego odbywa się dany warsztat
Subject – nazwa warszatu
StartTime – godzina startu warsztatu
EndTime – godzina końca warszatu
PeopleLimit - limit uczstników w warsztacie
Price – cena za warszat

Tabela ConferenceBooking

Tabela zawierające rezerwację konferencję

ConferenceBookingID – identyfikator rezerwacji na poszczególną konferencję
ConferenceID – identyfikator konferencji, której dotyczy rezerwacja
ClientID – identyfikator klienta, który dokonał rezerwacji

IsCancelled – wartość bitowa informująca o tym, czy rezerwacja jest anulowana. Rezerwacja może zostać anulowana przez klienta lub w przypadku przekroczenia daty płatności
BookingDate – data rezerwacji

Tabela ConferenceDayBooking

Tabela zawierające rezerwację na konkretne dni konferencji.

ConferenceDayBookingID – identyfikator rezerwacji na konkretny dzień konferencji
DayOfConferenceID – identyfikator dnia konferencji, którego dotyczy dana rezerwacja
ConferenceBookingID – identyfikator rezerwacji konferencji, w której rezerwujemy szczególny dzień
NumberOfParticipants – liczba osób, których dotyczy rezerwacja (zawiera w sobie liczbę studentów)
NumberOfStudent – liczba studentów, których dotyczy rezerwacja
IsCancelled – wartość bitowa, która określa czy nie anulowano rezerwacji

Tabela WorkshopBooking

Zawiera rezerwację na warsztaty

WorkshopBookingID – identyfikator rezerwacji warsztatu
ConferenceDayBookingID - identyfikator rezerwacji na konkretny dzień konferencji, w trakcie którego zarezerwowano warsztat
WorkshopID – identyfikator warsztatu, którego dotyczy rezerwacja
NumberOfParticipants – liczba osób, których dotyczy rezerwacja
IsCancelled – wartość bitowa, która określa czy nie anulowano rezerwacji

Tabela Payments

Tabela płatności za usługi.

PaymentsID – identyfikator płatności
Value – wartość płatności
PaymentDate – data płatności
ConferenceBookingID – identyfikator rezerwacji, z którym powiązana jest dana płatność

Tabela Participants

Tabela wszystkich osobowych uczestników konferencji.

ParticipantID– identyfikator uczestnika
LastName – nazwisko uczestnika
FirstName – imię uczestnika

E-mail - adres mailowy uczestnika

DayReservations

Tabela zarezerwowanych miejsc dla konkretnych osób na konferencję
danego dnia

DayReservationID – identyfikator rezerwacji dla konkretnej osoby
ConferenceBookingDayID – identyfikator rezerwacji ogólnej na konkretny dzień
ParticipantID – identyfikator uczestnika konferencji
isStudent - informacja czy osoba która zarezerwowała miejsce jest studentem
StudentCard – numer legitymacji studenckiej (w przypadku nie posiadania statusu studenta wartość null)
University - nazwa uczelni powiązanej z legitymacją studenckich

Tabela WorkshopReservations

Tabela zarezerwowanych miejsc dla konkretnych osób na warszaty.

WorkshopReservationID – identyfikator rezerwacji dla konkretnej osoby
DayReservationID – identyfikator rezerwacji dla konkretnej osoby danego dnia
WorkshopBookingID – identyfikator rezerwacji ogólnej na dany warsztat

3. Tabele

3.1 Klienci

```
CREATE TABLE [dbo].[Clients](
	[ClientID] [int] IDENTITY (1,1) NOT NULL,
	[Phone] [int] NOT NULL,
	[Email] [varchar](50) NOT NULL,
	[Address] [varchar](50) NOT NULL,
	[PostalCode] [varchar](10) NOT NULL,
	[City] [varchar](30) NOT NULL,
	[Country] [varchar](30) NOT NULL,
 CONSTRAINT [PK_Clients] PRIMARY KEY CLUSTERED
(
	[ClientID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Clients]  WITH CHECK ADD  CONSTRAINT [CK_Clients_Email] CHECK  (([Email]
like ''_%@_%._%''))
GO

ALTER TABLE [dbo].[Clients] CHECK CONSTRAINT [CK_Clients_Email]
GO

ALTER TABLE [dbo].[Clients]  WITH CHECK ADD  CONSTRAINT [CK_Clients_PostalCode_In_Poland]
CHECK  ((NOT [Country] like 'Poland' OR [PostalCode] like '[0-9][0-9]-[0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Clients] CHECK CONSTRAINT [CK_Clients_PostalCode_In_Poland]
GO
```

3.2 Companies

```
CREATE TABLE [dbo].[Companies](
	[ClientID] [int] NOT NULL,
	[CompanyName] [varchar](128) NOT NULL,
	[NIP] [varchar](20) NULL,
 CONSTRAINT [PK_Companies] PRIMARY KEY CLUSTERED
(
	[ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[NIP] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Companies]  WITH CHECK ADD  CONSTRAINT [FK_Companies_Clients]
FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[Companies] CHECK CONSTRAINT [FK_Companies_Clients]
GO
```

3.3 ConferenceBooking

```
CREATE TABLE [dbo].[ConferenceBooking](
	[ConferenceBookingID] [int] IDENTITY(1,1) NOT NULL,
```

```
	[ConferenceID] [int] NOT NULL,
	[ClientID] [int] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
	[BookingDate] [date] NOT NULL,
 CONSTRAINT [PK_Table_1] PRIMARY KEY CLUSTERED
(
	[ConferenceBookingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConferenceBooking] ADD  CONSTRAINT [DF_IsCancelled]  DEFAULT ((0)) FOR
[IsCancelled]
GO

ALTER TABLE [dbo].[ConferenceBooking]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceBooking_Clients] FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[ConferenceBooking] CHECK CONSTRAINT [FK_ConferenceBooking_Clients]
GO

ALTER TABLE [dbo].[ConferenceBooking]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceBooking_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[ConferenceBooking] CHECK CONSTRAINT [FK_ConferenceBooking_Conferences]
GO
```

3.4 ConferenceDayBooking

```
CREATE TABLE [dbo].[ConferenceDayBooking](
	[ConferenceDayBookingID] [int] IDENTITY(1,1) NOT NULL,
	[DayOfConferenceID] [int] NOT NULL,
	[ConferenceBookingID] [int] NOT NULL,
	[NumberOfParticipants] [int] NOT NULL,
	[NumberOfStudents] [int] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_ConferenceDayBooking] PRIMARY KEY CLUSTERED
(
	[ConferenceDayBookingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConferenceDayBooking] ADD  CONSTRAINT
[DF_ConferenceDayBooking_IsCancelled]  DEFAULT ((0)) FOR [IsCancelled]
```

```
GO

ALTER TABLE [dbo].[ConferenceDayBooking]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceDayBooking_ConferenceBooking] FOREIGN KEY([ConferenceBookingID])
REFERENCES [dbo].[ConferenceBooking] ([ConferenceBookingID])
GO

ALTER TABLE [dbo].[ConferenceDayBooking] CHECK CONSTRAINT
[FK_ConferenceDayBooking_ConferenceBooking]
GO

ALTER TABLE [dbo].[ConferenceDayBooking]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceDayBooking_ConferenceDays] FOREIGN KEY([DayOfConferenceID])
REFERENCES [dbo].[ConferenceDays] ([DayOfConferenceID])
GO

ALTER TABLE [dbo].[ConferenceDayBooking] CHECK CONSTRAINT
[FK_ConferenceDayBooking_ConferenceDays]
GO

ALTER TABLE [dbo].[ConferenceDayBooking]  WITH CHECK ADD  CONSTRAINT
[CK_ConferenceDayBooking_NumberOfParticipants_not_less_than_NumberOfStudents] CHECK
(([NumberOfParticipants]>=[NumberOfStudents]))
GO

ALTER TABLE [dbo].[ConferenceDayBooking] CHECK CONSTRAINT
[CK_ConferenceDayBooking_NumberOfParticipants_not_less_than_NumberOfStudents]
GO

ALTER TABLE [dbo].[ConferenceDayBooking]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceDayBooking_NotNegative_NumberOfStudents] CHECK  (([NumberOfStudents]>=(0)))
GO

ALTER TABLE [dbo].[ConferenceDayBooking] CHECK CONSTRAINT
[FK_ConferenceDayBooking_NotNegative_NumberOfStudents]
GO
```

3.5 ConferenceDays

```
CREATE TABLE [dbo].[ConferenceDays](
        [DayOfConferenceID] [int] IDENTITY(1,1) NOT NULL,
        [ConferenceID] [int] NOT NULL,
        [PeopleLimit] [int] NOT NULL,
        [Date] [date] NOT NULL,
 CONSTRAINT [PK_ConferenceDays] PRIMARY KEY CLUSTERED
(
        [DayOfConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[ConferenceDays]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceDays_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT [FK_ConferenceDays_Conferences]
GO

ALTER TABLE [dbo].[ConferenceDays]  WITH CHECK ADD  CONSTRAINT
[CK_ConferenceDays_Positive_PeopleLimit] CHECK  (([PeopleLimit]>=(0)))
GO

ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT
[CK_ConferenceDays_Positive_PeopleLimit]
GO
```

3.6 Conferences

```
CREATE TABLE [dbo].[Conferences](
        [ConferenceID] [int] IDENTITY(1,1) NOT NULL,
        [Name] [varchar](50) NOT NULL,
        [StartDate] [date] NOT NULL,
        [StudentDiscount] [decimal](2, 2) NOT NULL,
        [City] [varchar](50) NOT NULL,
        [PostalCode] [int] NOT NULL,
        [Address] [nchar](10) NOT NULL,
        [Email] [nvarchar](10) NOT NULL,
        [EndDate] [date] NOT NULL,
 CONSTRAINT [PK_Conferences] PRIMARY KEY CLUSTERED
(
        [ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT [CK_Conferences_Email] CHECK
(([Email] like '_%@_%._%'))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences_Email]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT
[CK_Conferences_EndDate_after_StartDate] CHECK  (([EndDate]>=[StartDate]))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences_EndDate_after_StartDate]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT
[CK_Conferences_StudentDiscount] CHECK  (([StudentDiscount]>=(0) AND [StudentDiscount]<(1)))
```

```
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CK_Conferences_StudentDiscount]
GO
```

3.7 DayReservations

```
CREATE TABLE [dbo].[DayReservations](
	[DayReservationID] [int] IDENTITY(1,1) NOT NULL,
	[ConferenceDayBookingID] [int] NOT NULL,
	[ParticipantID] [int] NOT NULL,
	[IsStudent] [bit] NOT NULL,
	[StudentCard] [varchar](6) NULL,
	[University] [varchar](255) NULL,
 CONSTRAINT [PK_DayReservations] PRIMARY KEY CLUSTERED
(
	[DayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT
[FK_DayReservations_ConferenceDayBooking] FOREIGN KEY([ConferenceDayBookingID])
REFERENCES [dbo].[ConferenceDayBooking] ([ConferenceDayBookingID])
GO

ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT
[FK_DayReservations_ConferenceDayBooking]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT
[FK_DayReservations_Participants] FOREIGN KEY([ParticipantID])
REFERENCES [dbo].[Participants] ([ParticipantID])
GO

ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT [FK_DayReservations_Participants]
GO

ALTER TABLE [dbo].[DayReservations]  WITH CHECK ADD  CONSTRAINT
[CK_DayReservations_isStudent_Card_And_University] CHECK  (([IsStudent]=(0) OR [StudentCard] IS
NOT NULL AND [University] IS NOT NULL))
GO

ALTER TABLE [dbo].[DayReservations] CHECK CONSTRAINT
[CK_DayReservations_isStudent_Card_And_University]
GO
```

3.8 IndividualClients

```
CREATE TABLE [dbo].[IndividualClients](
	[ClientID] [int] NOT NULL,
	[LastName] [varchar](40) NOT NULL,
	[FirstName] [varchar](20) NOT NULL,
 CONSTRAINT [PK_IndividualClients] PRIMARY KEY CLUSTERED
(
	[ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[IndividualClients]  WITH CHECK ADD  CONSTRAINT [FK_IndividualClients_Clients]
FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[IndividualClients] CHECK CONSTRAINT [FK_IndividualClients_Clients]
GO
```

3.9 Participants

```
CREATE TABLE [dbo].[Participants](
	[ParticipantID] [int] IDENTITY(1,1) NOT NULL,
	[LastName] [varchar](40) NOT NULL,
	[FirstName] [varchar](20) NOT NULL,
	[Email] [varchar](50) NULL,
 CONSTRAINT [PK_Participants] PRIMARY KEY CLUSTERED
(
	[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Participants]  WITH CHECK ADD  CONSTRAINT [CK_Participants_Email] CHECK
(([Email] like '_%@_%._%'))
GO

ALTER TABLE [dbo].[Participants] CHECK CONSTRAINT [CK_Participants_Email]
GO
```

3.10 Payments

```
CREATE TABLE [dbo].[Payments](
	[PaymentID] [int] IDENTITY(1,1) NOT NULL,
```

```
	[Value] [money] NOT NULL,
	[PaymentDate] [date] NOT NULL,
	[ConferenceBookingID] [int] NOT NULL,
 CONSTRAINT [PK_Payments] PRIMARY KEY CLUSTERED
(
	[PaymentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT [FK_Payments_ConferenceBooking]
FOREIGN KEY([ConferenceBookingID])
REFERENCES [dbo].[ConferenceBooking] ([ConferenceBookingID])
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [FK_Payments_ConferenceBooking]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT [CK_Payments_Positive_Value]
CHECK  (([Value]>(0)))
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [CK_Payments_Positive_Value]
GO
```

3.11 Prices

```
CREATE TABLE [dbo].[Prices](
	[PriceID] [int] IDENTITY(1,1) NOT NULL,
	[ConferenceID] [int] NOT NULL,
	[ValuePerDay] [money] NOT NULL,
	[StartDate] [date] NOT NULL,
	[EndDate] [date] NOT NULL,
 CONSTRAINT [PK_Prices] PRIMARY KEY CLUSTERED
(
	[PriceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [FK_Prices_Conferences] FOREIGN
KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [FK_Prices_Conferences]
GO
```

```
ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [CK_Prices_EndDate_After_StartDate]
CHECK  (([StartDate]<=[EndDate]))
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [CK_Prices_EndDate_After_StartDate]
GO
```

3.12 WorkshopBooking

```
CREATE TABLE [dbo].[WorkshopBooking](
	[WorkshopBookingID] [int] IDENTITY(1,1) NOT NULL,
	[ConferenceDayBookingID] [int] NOT NULL,
	[WorkshopID] [int] NOT NULL,
	[NumberOfParticipants] [int] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_WorkshopBooking] PRIMARY KEY CLUSTERED
(
	[WorkshopBookingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopBooking]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopBooking_ConferenceDayBooking] FOREIGN KEY([ConferenceDayBookingID])
REFERENCES [dbo].[ConferenceDayBooking] ([ConferenceDayBookingID])
GO

ALTER TABLE [dbo].[WorkshopBooking] CHECK CONSTRAINT
[FK_WorkshopBooking_ConferenceDayBooking]
GO

ALTER TABLE [dbo].[WorkshopBooking]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopBooking_Workshops] FOREIGN KEY([WorkshopID])
REFERENCES [dbo].[Workshops] ([WorkshopID])
GO

ALTER TABLE [dbo].[WorkshopBooking] CHECK CONSTRAINT [FK_WorkshopBooking_Workshops]
GO

ALTER TABLE [dbo].[WorkshopBooking]  WITH CHECK ADD  CONSTRAINT
[CK_WorkshopBooking_Positive_NumberOfParticipants] CHECK  (([NumberOfParticipants]>(0)))
GO

ALTER TABLE [dbo].[WorkshopBooking] CHECK CONSTRAINT
[CK_WorkshopBooking_Positive_NumberOfParticipants]
GO
```

3.13 WorkshopReservations

```
CREATE TABLE [dbo].[WorkshopReservations](
	[WorkshopReservationID] [int] IDENTITY(1,1) NOT NULL,
	[DayReservationID] [int] NOT NULL,
	[WorkshopBookingID] [int] NOT NULL,
 CONSTRAINT [PK_WorkshopReservations] PRIMARY KEY CLUSTERED
(
	[WorkshopReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopReservations_DayReservations] FOREIGN KEY([DayReservationID])
REFERENCES [dbo].[DayReservations] ([DayReservationID])
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT
[FK_WorkshopReservations_DayReservations]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopReservations_WorkshopBooking] FOREIGN KEY([WorkshopBookingID])
REFERENCES [dbo].[WorkshopBooking] ([WorkshopBookingID])
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT
[FK_WorkshopReservations_WorkshopBooking]
GO
```

3.14 Workshops

```
CREATE TABLE [dbo].[Workshops](
	[WorkshopID] [int] IDENTITY(1,1) NOT NULL,
	[DayOfConferenceID] [int] NULL,
	[Subject] [varchar](50) NULL,
	[StartTime] [time](7) NULL,
	[EndTime] [time](7) NULL,
	[PeopleLimit] [int] NULL,
	[Price] [money] NULL,
 CONSTRAINT [PK_Workshops] PRIMARY KEY CLUSTERED
(
	[WorkshopID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [FK_Workshops_ConferenceDays]
FOREIGN KEY([DayOfConferenceID])
REFERENCES [dbo].[ConferenceDays] ([DayOfConferenceID])
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [FK_Workshops_ConferenceDays]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT
[CK_Workshops_Positive_PeopleLimit] CHECK  (([PeopleLimit]>=(0)))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CK_Workshops_Positive_PeopleLimit]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CK_Workshops_Price] CHECK
(([Price] IS NULL OR [Price]>=(0)))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CK_Workshops_Price]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT
[CK_Workshops_EndTime_After_StartTime] CHECK  (([StartTime]<[EndTime]))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CK_Workshops_EndTime_After_StartTime]
GO
```

4. Funkcje

4.1 BookingValue

Zwaca sumaryczną wartość do zapłaty za konferencję

```
CREATE FUNCTION [dbo].[BookingValue] (@ConferenceBookingID int)
RETURNS money
AS
BEGIN
	RETURN (
	(select isnull(sum(dbo.DayOfConferenceWorkshopBookingValue
(cdb.ConferenceDayBookingID)),0.00)
	from ConferenceBooking as cb
	inner join ConferenceDayBooking as cdb
```

```
        on cdb.ConferenceBookingID = cb.ConferenceBookingID
        where cb.ConferenceBookingID = @ConferenceBookingID)
        +
        (select isnull(sum(tmp.tmpVal),0.00)
        from
        (select
((cdb.NumberOfParticipants-cdb.NumberOfStudents)*(dbo.ConferenceDayPrice(cb.ConferenceID,cb.Booki
ngDate))

+cdb.NumberOfStudents*(dbo.ConferenceDayPrice(cb.ConferenceID,cb.BookingDate))*
                                    (select 1-c.StudentDiscount from Conferences as c where
c.ConferenceID = cb.ConferenceID)) as tmpVal
        from ConferenceBooking as cb
        inner join ConferenceDayBooking as cdb
        on cdb.ConferenceBookingID = cb.ConferenceBookingID
        where cb.ConferenceBookingID = @ConferenceBookingID
        ) as tmp)
        );
END
GO
```

4.2 ConferenceParticipants

Funkcja, która zwraca listę uczestników danej konferencji.

```
QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[ConferenceParticipants] ( @ConferenceID Int)
RETURNS @ConferenceParticipants TABLE
(
        PartcipantID Int,
        FirstName varchar (20),
        LastName varchar (40),
        Email varchar (50)
)
AS
BEGIN
        INSERT INTO @ConferenceParticipants
        select p.ParticipantID, p.FirstName, p.LastName, p.Email
        FROM Participants AS p
        JOIN DayReservations AS dr
        ON dr.ParticipnantID = p.ParticipantID
        JOIN ConferenceDayBooking AS cdb
        ON cdb.ConferenceDayBookingID = dr.ConferenceDayBookingID
        JOIN ConferenceBooking AS cb
        ON cb.ConferenceBookingID = cdb.ConferenceBookingID
        WHERE cb.ConferenceID = @ConferenceID
        RETURN;
END
```

```
GO
```

4.3 ConferenceDayPrice

Funkcja zwracająca cenę za dzień konferencji w zależności od daty rezerwacji (różne progi cenowe)

```
CREATE FUNCTION [dbo].[ConferenceDayPrice] (@ConferenceID int, @Date date)
RETURNS money
AS
BEGIN
	RETURN (
	SELECT p.ValuePerDay
	FROM Prices AS p
	JOIN Conferences AS c
	ON c.ConferenceID = p.ConferenceID
	WHERE (@ConferenceID = p.ConferenceID AND
		p.EndDate >= @Date AND
		p.StartDate <= @Date)
	);
END
GO
```

4.4 ConferencePrice

Funkcja zwracająca koszt udział w danej konferencji w zależności od daty rezerwacji (progi cenowe).

```
CREATE FUNCTION [dbo].[ConferencePrice] (@ConferenceID int, @Date date)
RETURNS money
AS
BEGIN
	RETURN (
	SELECT p.ValuePerDay* (DATEDIFF(day, c.StartDate, c.EndDate)+1)
	FROM Prices AS p
	JOIN Conferences AS c
	ON c.ConferenceID = p.ConferenceID
	WHERE (@ConferenceID = p.ConferenceID AND
	p.EndDate >= @Date AND
	p.StartDate <= @Date)
	);
END
GO
```

4.5 DayOfConferenceParticipants

Funkcja zwracająca tabelę uczestników danego dnia konferencji

```
CREATE FUNCTION [dbo].[DayOfConferenceParticipants] ( @DayOfConferenceID Int)
RETURNS @ConferenceParticipants TABLE
(
	PartcipantID Int,
	FirstName varchar (20),
	LastName varchar (40),
	Email varchar (50)
)
AS
BEGIN
	INSERT INTO @ConferenceParticipants
	select p.ParticipantID, p.FirstName, p.LastName, p.Email
	FROM Participants AS p
	JOIN DayReservations AS dr
	ON dr.ParticipantID = p.ParticipantID
	JOIN ConferenceDayBooking AS cdb
	ON cdb.ConferenceDayBookingID = dr.ConferenceDayBookingID
	WHERE cdb.DayOfConferenceID = @DayOfConferenceID
	RETURN;
END
```

4.6 DaysForConference

Funkcja zwracająca tabelę zawierające poszczególne dni konferencji wraz z limitami osób

```
CREATE Function [dbo].[DaysForConference] ( @ConferenceID Int)
returns @days TABLE
(
	DayOfConferenceID Int,
	PeopleLimit Int,
	Date date
)
as
Begin
	Insert Into @days
	select DayOfConferenceID, PeopleLimit, Date
	from ConferenceDays
	where @ConferenceID = ConferenceID
	return;
End
GO
```

4.7 DayOfConferenceWorkshopBookingValue

Funkcja zwracająca koszt rezerwacji warsztatu

```
CREATE FUNCTION [dbo].[DayOfConferenceWorkshopBookingValue] (@ConferenceDayBookingID int)
RETURNS money
AS
BEGIN
	RETURN (
	SELECT isnull(sum(dbo.WorkshopBookingValue (wb.WorkshopBookingID)),0.00)
	FROM ConferenceDayBooking as cdb
	inner join WorkshopBooking as wb
	on wb.ConferenceDayBookingID = cdb.ConferenceDayBookingID
	where cdb.ConferenceDayBookingID = @ConferenceDayBookingID
	AND wb.IsCancelled = 0
	);
END
GO
```

4.8 FreePlacesForConferenceDay

Zwraca ilość wolnych miejsc na dany dzień konferencji

```
CREATE FUNCTION [dbo].[FreePlacesForConferenceDay] (@ConferenceDayID int)
RETURNS int
AS
BEGIN
	RETURN (
	SELECT cd.PeopleLimit -
	(select sum(NumberOfParticipants) from ConferenceDayBooking as cdb
	where cdb.DayOfConferenceID = @ConferenceDayID and cdb.IsCancelled = 0) as
NumberOfFreePlaces
	FROM ConferenceDays as cd
	where cd.DayOfConferenceID = @ConferenceDayID

	);
END
GO
```

4.9 FreePlacesForWorkshop

Zwraca ilość wolnych miejsc na dany warsztat

```
CREATE FUNCTION [dbo].[FreePlacesForWorkshop] (@WorkshopID int)
RETURNS int
AS
BEGIN
	RETURN (
	SELECT w.PeopleLimit -
	(select sum(wb.NumberOfParticipants) from WorkshopBooking as wb
	where wb.WorkshopID = @WorkshopID and wb.IsCancelled = 0) as NumberOfFreePlaces
	FROM Workshops as w
	where w.WorkshopID = @WorkshopID

	);
END
GO
```

4.10 PaymentsSum

Zwraca sumę wpłat klienta na daną konferencję

```
CREATE FUNCTION [dbo].[PaymentsSum] (@ClientID int, @ConferenceID int)
RETURNS money
AS
BEGIN
	RETURN(
	SELECT ISNULL(SUM(p.Value),0)
	FROM Payments as p
	JOIN ConferenceBooking as cb
	ON cb.ConferenceBookingID = p.ConferenceBookingID
	WHERE (cb.ClientID = @ClientID AND cb.ConferenceID = @ConferenceID)
	);
END
GO
```

4.11 PaymentsSumOfClient

Funkcja zwracająca sumę wpłat klienta na konferencje

```
CREATE FUNCTION [dbo].[PaymentsSumOfClient] (@ClientID int)
RETURNS money
AS
BEGIN
	RETURN(
	SELECT ISNULL(SUM(p.Value),0.00)
```

```
        FROM Payments as p
        JOIN ConferenceBooking as cb
        ON cb.ConferenceBookingID = p.ConferenceBookingID
        WHERE (cb.ClientID = @ClientID)
        );
END
GO
```

4.12 TotalPaymentsForConference

Zwraca sumę wpłat na daną konferencję przez wszystkich uczestników

```
CREATE FUNCTION [dbo].[TotalPaymentsForConference] (@ConferenceID int)
RETURNS money
AS
BEGIN
        RETURN (
        SELECT sum(p.value) as SumOfPayments
        FROM Payments as p
        inner join ConferenceBooking as cb
        on p.ConferenceBookingID = cb.ConferenceBookingID
        inner join Conferences as c
        on c.ConferenceID = cb.ConferenceID
        where cb.ConferenceID = @ConferenceID

        );
END
GO
```

4.13 WorkshopBookingValue

Zwraca sumę do zapłaty za daną rezerwację warsztatu

```
CREATE FUNCTION [dbo].[WorkshopBookingValue] (@WorkshopBookingID int)
RETURNS money
AS
BEGIN
        RETURN (
        SELECT wb.NumberOfParticipants*(SELECT w.Price FROM Workshops as w where
w.WorkshopID = wb.WorkshopID)
        FROM WorkshopBooking as wb
        WHERE wb.WorkshopBookingID = @WorkshopBookingID
        AND wb.IsCancelled = 0
        );
END
GO
```

4.14 WorkshopsInConferenceDay

Zwraca listę warsztatów odbywających się w danym dniu.

```
create Function [dbo].[WorkshopsInConferenceDay] ( @ConferenceDayID Int)
returns @days TABLE
(
	Workshop Int,
	Subject varchar(50),
	StartimeTime time(7),
	EndTime time(7),
	PeopleLimit Int,
	Price money

)
as
Begin
	Insert Into @days
	select WorkshopID, Subject, StartTime, EndTime, PeopleLimit, Price
	from Workshops
	where @ConferenceDayID = DayOfConferenceID
	return;
End
GO
```

4.15 WorkshopParticipants

Zwraca listę uczestników danego warsztatu

```
CREATE FUNCTION [dbo].[WorkshopParticipants] ( @WorkshopID Int)
RETURNS @WorkshopParticipants TABLE
(
	PartcipantID Int,
	FirstName varchar (20),
	LastName varchar (40),
	Email varchar (50)
)
AS
BEGIN
	INSERT INTO @WorkshopParticipants
	SELECT p.ParticipantID, p.FirstName, p.LastName, p.Email
	FROM Participants AS p
	JOIN DayReservations AS dr
	ON dr.ParticipnantID = p.ParticipantID
	JOIN WorkshopReservations AS wr
	ON wr.DayReservationID = dr.DayReservationID
	JOIN WorkshopBooking AS wb
	ON wb.WorkshopBookingID = wr.WorkshopBookingID
	WHERE wb.WorkshopID = @WorkshopID
	RETURN;
END
GO
```

4.16 ConferencesForParticipant

Zwraca konferencje, na które jest zapisany dany uczestnik

```
CREATE FUNCTION [dbo].[ConferencesForParticipant] (@ParticipantID Int)
RETURNS @ConferencesForParticipant TABLE
(
	ConferenceID Int,
	ConferenceName varchar(255),
	StartDate Date,
	EndDate Date
	)
AS
BEGIN
INSERT INTO @ConferencesForParticipant
	SELECT c.ConferenceID, c.Name, c.StartDate, c.EndDate
	FROM Conferences as c
	JOIN ConferenceBooking as cb
	ON cb.ConferenceID = c.ConferenceID AND cb.IsCancelled = '0'
	JOIN ConferenceDayBooking as cdb
	ON cdb.ConferenceBookingID = cb.ConferenceBookingID AND cdb.IsCancelled = '0'
	JOIN  DayReservations as dr
	ON dr.ConferenceDayBookingID = cdb.ConferenceDayBookingID
	WHERE dr.ParticipantID = @ParticipantID
	RETURN;
END
```

4.17 WorkshopsForParticipant

Zwraca listę warsztatów na które uczestnik jest zapisany

```
CREATE FUNCTION [dbo].[WorkshopsForParticipant] (@ParticipantID Int)
RETURNS @WorkshopsForParticipant TABLE
(
	WorkshopID Int,
	WorkshopSubject varchar(255),
	WorkshopDay Date,
	StartTime Time,
	EndDate Time
	)
AS
BEGIN
INSERT INTO @WorkshopsForParticipant
	SELECT w.WorkshopID, w.Subject, cd.Date, w.StartTime, w.EndTime
	FROM Workshops as w
	JOIN ConferenceDays as cd
	ON cd.DayOfConferenceID = w.DayOfConferenceID
	JOIN WorkshopBooking as wb
	ON wb.WorkshopID = w.WorkshopID AND wb.isCancelled= '0'
	JOIN WorkshopReservations as wr
	ON wr.WorkshopBookingID = wb.WorkshopBookingID
```

```
        JOIN DayReservations as dr
        ON dr.DayReservationID = wr.DayReservationID
        WHERE dr.ParticipantID = @ParticipantID
        RETURN;
END
```

5. Procedury

5.1 AddClient

Procedura dodająca klienta

```
CREATE PROCEDURE [dbo].[PROC_AddClient]
(
        @Phone int,
        @Email varchar(50),
        @Address varchar(50),
        @PostalCode varchar(10),
        @City varchar(30),
        @Country varchar(30)
)
AS
BEGIN
SET NOCOUNT ON;
BEGIN TRY
        INSERT INTO Clients
        (Phone,
        Email,
        Address,
        PostalCode,
        City,
        Country)

        VALUES(
        @Phone,
        @Email,
        @Address,
        @PostalCode,
        @City,
        @Country)
        END TRY
        BEGIN CATCH
```

```
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add client' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
END
GO
```

5.2 AddCompany

Procedura dodająca firmę

```
CREATE PROCEDURE [dbo].[PROC_AddCompany]
(
	@ClientID int,
	@CompanyName varchar(128),
	@NIP varchar(20)
)
AS
BEGIN
SET NOCOUNT ON;
BEGIN TRY
	IF NOT EXISTS
		(SELECT * FROM Clients
		WHERE ClientID = @ClientID)
	BEGIN
	;THROW 52000, 'There is no client with that number',1
	END


	INSERT INTO Companies
	(ClientID,
	CompanyName,
	NIP)

	VALUES(
	@ClientID,
	@CompanyName,
	@NIP)
	END TRY
	BEGIN CATCH
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add company' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
END
GO
```

5.3 AddConference

Procedura dodająca konferencję

```
CREATE PROCEDURE [dbo].[PROC_AddConference]
	(@Name varchar(50),
	@StartDate date,
	@StudentDiscount decimal (2,2),
	@City varchar(50),
	@PostalCode int,
	@Address nchar(10),
	@Email nvarchar(10),
	@EndDate date)
AS
BEGIN
SET NOCOUNT ON;
	IF(@StartDate > @EndDate)
		BEGIN;
		THROW 52000, 'EndDate should not be earlier than StartDate.',1
		END

	IF(@StudentDiscount < 0 OR @StudentDiscount > 1)
		BEGIN
		;THROW 52000, 'The discount must be between 0 and 1.',1
		END

	INSERT INTO Conferences
		(Name,
		StartDate,
		StudentDiscount,
		City,
		PostalCode,
		Address,
		Email,
		EndDate)

	VALUES(
		@Name,
		@StartDate,
		@StudentDiscount,
		@City,
		@PostalCode,
		@Address,
		@Email,
		@EndDate)
END
GO
```

5.4 AddConferenceBooking

Procedura dodająca rezerwację konferencji

```
CREATE PROCEDURE [dbo].[PROC_AddConferenceBooking]
	(@ConferenceID int,
	@ClientID int,
	@BookingDate date)
AS
BEGIN
SET NOCOUNT ON
	BEGIN TRY
	IF NOT EXISTS (SELECT * FROM Clients
			WHERE ClientID = @ClientID)
	BEGIN;
	THROW 52000, 'This client does not exist',1
	END

	IF NOT EXISTS (SELECT * FROM Conferences
			WHERE ConferenceID = @ConferenceID)
	BEGIN;
	THROW 52000, 'This conference does not exist',1
	END

	INSERT INTO ConferenceBooking
		(ConferenceID,
		ClientID,
		BookingDate)
	VALUES
		(@ConferenceID,
		@ClientID,
		@BookingDate)

	END TRY

	BEGIN CATCH
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add conference booking' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
END


GO
```

5.5 AddConferenceDay

procedura dodająca dzień konferencji

```
CREATE PROCEDURE [dbo].[PROC_AddConferenceDay]
	(@ConferenceID int,
	@PeopleLimit int,
	@Date date)
AS
BEGIN
SET NOCOUNT ON
```

```
	BEGIN TRY
	IF NOT EXISTS
		(SELECT * FROM Conferences
		WHERE ConferenceID = @ConferenceID)
	BEGIN
	;THROW 52000, 'There is no conference with that number',1
	END

	IF EXISTS
		(SELECT * FROM ConferenceDays AS cd
		WHERE @Date = cd.Date AND cd.ConferenceID = @ConferenceID)
	BEGIN
	; THROW 52000, 'Cannot add that day, it already exists' ,1
	END

	INSERT INTO ConferenceDays
		(ConferenceID,
		PeopleLimit,
		Date)

	VALUES
		(@ConferenceID,
		@PeopleLimit,
		@Date)
	END TRY

	BEGIN CATCH
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add day' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
END
GO
```

5.6 AddConferenceDayBooking

Procedura dodająca rezerwacją na dzień konferencji

```
CREATE PROCEDURE [dbo].[PROC_AddConferenceDayBooking]
	(@DayOfConferenceID int,
	@ConferenceBookingID int,
	@NumberOfParticpiants int,
	@NumberOfStudents int)

AS
BEGIN
SET NOCOUNT ON
	BEGIN TRY
	IF NOT EXISTS (SELECT * FROM ConferenceBooking
			WHERE ConferenceBookingID = @ConferenceBookingID)
		BEGIN;
		THROW 52000, 'Conference booking does not exist', 1
		END
```

```
	IF NOT EXISTS (SELECT * FROM ConferenceDays
				WHERE DayOfConferenceID = @DayOfConferenceID)
		BEGIN;
		THROW 52000, 'This day does not exist',1
		END

	DECLARE @Day int = (SELECT DayOfConferenceID
					FROM ConferenceDayBooking
					WHERE DayOfConferenceID = @DayOfConferenceID AND
ConferenceBookingID = @ConferenceBookingID
					)

	IF(@Day IS NOT NULL)
		BEGIN;
		THROW 52000, 'This conferece booking has already been added',1
		END

	IF (@NumberOfParticpiants <= 0)
		BEGIN;
		THROW 52000, 'Number of particpants must be positive',1
		END

	IF (@NumberOfParticpiants < @NumberOfStudents)
		BEGIN;
		THROW 52000, 'Number of participants must be greater or at least equal to number of
students',1
		END

	DECLARE @FreePlaces int = (
					SELECT dbo.FreePlacesForConferenceDay(@Day)
					)
	IF (@FreePlaces = 0)
		BEGIN;
		THROW 52000, 'No free places for this day', 1
		END

	IF (@FreePlaces < @NumberOfParticpiants)
		BEGIN;
		THROW 52000, 'This day has not enough free places', 1
		END

	INSERT INTO ConferenceDayBooking
		(DayOfConferenceID,
		ConferenceBookingID,
		NumberOfParticipants,
		NumberOfStudents,
		IsCancelled)

	VALUES
		(@DayOfConferenceID,
		@ConferenceBookingID,
		@NumberOfParticpiants,
		@NumberOfStudents,
```

```
                0)

        END TRY
        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot add conference day booking' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.7 AddDayReservation

Procedura dodająca rezerwację konkretnego uczestnika na dzień konferencji

```
CREATE PROCEDURE [dbo].[PROC_AddDayReservation]
        (@ConferenceDayBookingID int,
        @ParticipantID int,
        @IsStudent bit,
        @StudentCard varchar(6),
        @University varchar(255))

AS
BEGIN
SET NOCOUNT ON;
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM Participants
                        WHERE ParticipantID = @ParticipantID)
        BEGIN;
        THROW 52000, 'This participant does not exist', 1
        END

        IF NOT EXISTS (SELECT * FROM ConferenceDayBooking
                        WHERE ConferenceDayBookingID = @ConferenceDayBookingID)
        BEGIN;
        THROW 52000, 'This conference day booking does not exist', 1
        END

        IF (@IsStudent = 1 AND (@StudentCard IS NULL OR @University IS NULL))
        BEGIN;
        THROW 52000, 'You have to add student card number and university', 1
        END

        IF EXISTS (SELECT * FROM DayReservations
                        WHERE ConferenceDayBookingID = @ConferenceDayBookingID AND
                                ParticipantID = @ParticipantID)
        BEGIN;
        THROW 52000, 'The participant has been already added for this day',1
        END

        INSERT INTO DayReservations
                (ConferenceDayBookingID,
```

```
                ParticipantID,
                IsStudent,
                StudentCard,
                University)

        VALUES
                (@ConferenceDayBookingID,
                @ParticipantID,
                @IsStudent,
                @StudentCard,
                @University)

        END TRY
        BEGIN CATCH
                DECLARE @errorMessage nvarchar (2048)
                SET @errorMessage = 'Cannot add reservation' + ERROR_MESSAGE();
                THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.8 AddIndividualClient

Procedura dodająca nowego klienta indywidualnego

```
CREATE PROCEDURE [dbo].[PROC_AddIndividualClient]
        (@ClientID int,
        @LastName varchar(40),
        @Firstname varchar (20))
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS
                (SELECT * FROM Clients
                WHERE ClientID = @ClientID)
        BEGIN
        ;THROW 52000, 'There is no client with that number',1
        END

        INSERT INTO IndividualClients
                (ClientID,
                FirstName,
                LastName)

        VALUES
                (@ClientID,
                @Firstname,
                @LastName)
        END TRY
```

```
	BEGIN CATCH
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add individual client' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
END
GO
```

5.9 AddParticipant

procedura dodająca nowego uczestnika

```
CREATE PROCEDURE [dbo].[PROC_AddParticipant]
	(@LastName varchar(40),
	@Firstname varchar(20),
	@Email varchar(50)
	)
AS
BEGIN
SET NOCOUNT ON;
	INSERT INTO Participants
	(LastName,
	FirstName,
	Email)

	VALUES(
	@LastName,
	@Firstname,
	@Email)
END
GO
```

5.10 AddPayment

Procedura dodająca wpłatę na daną konferencję

```
CREATE PROCEDURE [dbo].[PROC_AddPayment]
  (@Value money,
  @PaymentDate date,
  @ConferenceBookingID int)
AS
BEGIN
SET NOCOUNT ON
	BEGIN TRY
	IF(@Value = 0)
		BEGIN;
		THROW 52000, 'Value cannot be 0',1
		END

	IF NOT EXISTS (SELECT * FROM ConferenceBooking
```

```
                WHERE ConferenceBookingID = @ConferenceBookingID)
            BEGIN;
            THROW 52000, 'This conference booking does not exist', 1
            END

        IF 1 = (SELECT IsCancelled FROM ConferenceBooking
            WHERE ConferenceBookingID = @ConferenceBookingID)
            BEGIN;
            THROW 52000, 'Cannot add payment to cancelled booking',1
            END
        IF 7 > DATEDIFF(DAY,@PaymentDate,(SELECT cb.BookingDate
                FROM ConferenceBooking as cb
                WHERE cb.ConferenceBookingID = @ConferenceBookingID))
            BEGIN;
            THROW 52000, 'Cannot add payment, it is too late',1
            END


        INSERT INTO Payments
        (Value,
        PaymentDate,
        ConferenceBookingID)

        VALUES
        (@Value,
        @PaymentDate,
        @ConferenceBookingID)

        END TRY
        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot add payment' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.11 AddPrice

Procedura dodająca nowy próg cenowy na daną konferencję

```
CREATE PROCEDURE [dbo].[PROC_AddPrice]
        (
        @ConferenceID int,
        @ValuePerDay money,
        @StartDate date,
        @EndDate date)


AS
BEGIN
```

```
SET NOCOUNT ON;
	IF(@StartDate > @EndDate)
	BEGIN;
	THROW 52000, 'EndDate should not be earlier than StartDate.',1
	END

	IF(@ValuePerDay < 0)
	BEGIN;
	THROW 52000, 'The ValuePerDay must not be less than 0',1
	END
	IF NOT EXISTS (SELECT * FROM Conferences WHERE ConferenceID = @ConferenceID)
	BEGIN;
	THROW 52000, 'Conference does not exist',1
	END
	IF EXISTS (SELECT * FROM Conferences where ConferenceID = @ConferenceID
		AND EndDate >= @StartDate AND EndDate <= @EndDate)
	BEGIN;
	THROW 52000, 'Dates must not overlap, already exists such Price whose EndDate is between
StartDate and EndDate',1
	END


	INSERT INTO Prices
	(ConferenceID,
	ValuePerDay,
	StartDate,
	EndDate)

	VALUES(
	@ConferenceID,
	@ValuePerDay,
	@StartDate,
	@EndDate)
END
GO
```

5.12 AddWorkshop

Procedura dodająca nowy warsztat

```
CREATE PROCEDURE [dbo].[PROC_AddWorkshop]
	(@DayOfConferenceID int,
	@Subject varchar(50),
	@StartTime time(7),
	@EndTime time(7),
	@PeopleLimit int,
	@Price money)
AS
BEGIN
SET NOCOUNT ON
	BEGIN TRY
	IF(@StartTime > @EndTime)
	BEGIN;
```

```
        THROW 52000, 'EndTime should not be earlier than StartTime.',1
        END

        IF NOT EXISTS(SELECT * FROM ConferenceDays
                        WHERE DayOfConferenceID = @DayOfConferenceID)

        BEGIN;
        THROW 52000, 'Conference day does not exist', 1
        END


        INSERT INTO Workshops
                (DayOfConferenceID,
                Subject,
                StartTime,
                EndTime,
                PeopleLimit,
                Price)

        VALUES
                (@DayOfConferenceID,
                @Subject,
                @StartTime,
                @EndTime,
                @PeopleLimit,
                @Price)

        END TRY
        BEGIN CATCH
                DECLARE @errorMessage nvarchar (2048)
                SET @errorMessage = 'Cannot add day' + ERROR_MESSAGE();
                THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.13 AddWorkshopBooking

Procedura dodająca rezerwację na warsztat

```
CREATE PROCEDURE [dbo].[PROC_AddWorkshopBooking]
        (@ConferenceDayBookingID int,
        @WorkshopID int,
        @NumberOfParticipants int)

AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM ConferenceDayBooking
                        WHERE ConferenceDayBookingID = @ConferenceDayBookingID)
        BEGIN;
```

```
        THROW 52000, 'This conferece day booking does not exist',1
        END

        IF NOT EXISTS (SELECT * FROM Workshops
                        WHERE WorkshopID = @WorkshopID)
        BEGIN;
        THROW 52000, 'This workshop does not exist',1
        END

        IF EXISTS (SELECT * FROM WorkshopBooking
                        WHERE ConferenceDayBookingID = @ConferenceDayBookingID
                        AND WorkshopID = @WorkshopID
                        AND IsCancelled = 0)
        BEGIN;
        THROW 52000, 'Workshop has been already booked', 1
        END

        IF 1 = (SELECT IsCancelled FROM ConferenceDayBooking
                WHERE ConferenceDayBookingID = @ConferenceDayBookingID)
        BEGIN;
        THROW 52000, 'Cannot book workshop for cancelled day reservation', 1
        END

        DECLARE @FreePlaces int = (SELECT dbo.FreePlacesForWorkshop(@WorkshopID))

        IF (@FreePlaces = 0)
        BEGIN;
        THROW 52000, 'No free places for workshop',1
        END

        IF (@FreePlaces < @NumberOfParticipants)
        BEGIN;
        THROW 52000, 'There is not enough places',1
        END

        INSERT INTO WorkshopBooking
        (ConferenceDayBookingID,
        WorkshopID,
        NumberOfParticipants,
        IsCancelled)

        VALUES
        (@ConferenceDayBookingID,
        @WorkshopID,
        @NumberOfParticipants,
        0)

        END TRY
        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot add workshop booking' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
```

```
GO
```

5.14 AddWorkshopReservation

Procedura dodająca rezerwację konkretnej osoby na warsztat. Istotnym elementem tej procedury jest sprawdzenie czy dana osoba nie uczestniczy już w jakimś innym warsztacie w danym czasie

```
CREATE PROCEDURE [dbo].[PROC_AddWorkshopReservation]
	(@DayReservationID int,
	@WorkshopBookingID int
	)

AS
BEGIN
SET NOCOUNT ON;
	BEGIN TRY
	IF NOT EXISTS (SELECT * FROM DayReservations
			WHERE DayReservationID = @DayReservationID)
	BEGIN;
	THROW 52000, 'This day reservation does not exist', 1
	END

	IF NOT EXISTS (SELECT * FROM WorkshopBooking
			WHERE WorkshopBookingID = @WorkshopBookingID)
	BEGIN;
	THROW 52000, 'This workshop booking booking does not exist', 1
	END

	IF (SELECT p.ParticipantID
		from DayReservations as dr
		inner join Participants as p
		on p.ParticipantID = dr.ParticipantID AND dr.DayReservationID = @DayReservationID)
		IN
		(select p.ParticipantID from Participants as p
		inner join DayReservations as dr
		on p.ParticipantID = dr.ParticipantID
		inner join WorkshopReservations as wr
		on wr.DayReservationID = dr.DayReservationID
		inner join WorkshopBooking as wb
		on wr.WorkshopBookingID = wb.WorkshopBookingID
		where wb.IsCancelled = 0 AND wb.WorkshopID in
			(select w1.WorkshopID from Workshops as w1
			where
			(w1.EndTime > (select tmp.StartTime from
			(select w.StartTime, w.EndTime,
			(select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
	w.DayOfConferenceID) as DateOfWorkshop
			from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
	WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp)
```

```
                    AND w1.EndTime < (select tmp.EndTime from
                    (select w.StartTime, w.EndTime,
                    (select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
w.DayOfConferenceID) as DateOfWorkshop
                    from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp)
                    AND
                    (select cd1.Date from ConferenceDays as cd1 where cd1.DayOfConferenceID =
w1.DayOfConferenceID) = (select tmp.DateOfWorkshop from
                    (select w.StartTime, w.EndTime,
                    (select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
w.DayOfConferenceID) as DateOfWorkshop
                    from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp))
                    OR
                    (w1.StartTime > (select tmp.StartTime from
                    (select w.StartTime, w.EndTime,
                    (select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
w.DayOfConferenceID) as DateOfWorkshop
                    from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp)
                    AND w1.StartTime < (select tmp.EndTime from
                    (select w.StartTime, w.EndTime,
                    (select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
w.DayOfConferenceID) as DateOfWorkshop
                    from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp)
                    AND
                    (select cd1.Date from ConferenceDays as cd1 where cd1.DayOfConferenceID =
w1.DayOfConferenceID) = (select tmp.DateOfWorkshop from
                    (select w.StartTime, w.EndTime,
                    (select cd.Date from ConferenceDays as cd where cd.DayOfConferenceID =
w.DayOfConferenceID) as DateOfWorkshop
                    from  workshops as w where w.WorkshopID =(select wb.WorkshopID from
WorkshopBooking as wb where wb.WorkshopBookingID=@WorkshopBookingID)) as tmp))))
BEGIN;
THROW 52000, 'This participant already takes part in other workshop at the same time',1
END

INSERT INTO WorkshopReservations
(DayReservationID,
WorkshopBookingID)

VALUES
(@DayReservationID,
@WorkshopBookingID
)

END TRY
BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot add workshop reservation' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
END CATCH
```

```
END
GO
```

AddWorkshop

Procedura dodająca nowy warsztat

```
CREATE PROCEDURE [dbo].[PROC_AddWorkshop]
	(@DayOfConferenceID int,
	@Subject varchar(50),
	@StartTime time(7),
	@EndTime time(7),
	@PeopleLimit int,
	@Price money)
AS
BEGIN
SET NOCOUNT ON
	BEGIN TRY
	IF(@StartTime > @EndTime)
	BEGIN;
	THROW 52000, 'EndTime should not be earlier than StartTime.',1
	END

	IF NOT EXISTS(SELECT * FROM ConferenceDays
			WHERE DayOfConferenceID = @DayOfConferenceID)

	BEGIN;
	THROW 52000, 'Conference day does not exist', 1
	END


	INSERT INTO Workshops
	(DayOfConferenceID,
	Subject,
	StartTime,
	EndTime,
	PeopleLimit,
	Price)

	VALUES
	(@DayOfConferenceID,
	@Subject,
	@StartTime,
	@EndTime,
	@PeopleLimit,
	@Price)

	END TRY
	BEGIN CATCH
		DECLARE @errorMessage nvarchar (2048)
		SET @errorMessage = 'Cannot add day' + ERROR_MESSAGE();
		THROW 5200, @errorMessage, 1;
	END CATCH
```

```
END
GO
```

5.15 CancelConferenceBooking

Procedura anulująca rezerwację na konferencję

```
CREATE PROCEDURE [dbo].[PROC_CancelConferenceBooking]
        (@ConferenceBookingID int)
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM ConferenceBooking
                        WHERE ConferenceBookingID = @ConferenceBookingID)
                BEGIN;
                THROW 52000, 'Conference booking does not exist', 1
                END

        IF 1 = (SELECT IsCancelled FROM ConferenceBooking
                WHERE ConferenceBookingID = @ConferenceBookingID)
                BEGIN;
                THROW 52000, 'Booking has already been cancelled',1
                END


        UPDATE ConferenceBooking
        SET IsCancelled = 1
        WHERE ConferenceBookingID = @ConferenceBookingID
        END TRY

        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot cancel conference booking' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

CancelConferenceDayBooking - procedura anulująca rezerwację na dany dzień konferencji

```
CREATE PROCEDURE [dbo].[PROC_CancelConferenceDayBooking]
        (@ConferenceDayBookingID int)
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM ConferenceDayBooking
```

```
                WHERE ConferenceDayBookingID = @ConferenceDayBookingID)
            BEGIN;
            THROW 52000, 'Conference day booking does not exist', 1
            END

        IF 1 = (SELECT IsCancelled FROM ConferenceDayBooking
            WHERE ConferenceDayBookingID = @ConferenceDayBookingID)
            BEGIN;
            THROW 52000, 'Booking has already been cancelled',1
            END


        UPDATE ConferenceDayBooking
        SET IsCancelled = 1
        WHERE @ConferenceDayBookingID = ConferenceDayBookingID
        END TRY

        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot cancel conference day booking' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.16 CancelWorkshopBooking

Procedura anulująca rezerwację na warsztat

```
CREATE PROCEDURE [dbo].[PROC_CancelWorkshopBooking]
        (@WorkshopBookingID int)
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM WorkshopBooking
                WHERE WorkshopBookingID = @WorkshopBookingID)
            BEGIN;
            THROW 52000, 'Workshop booking does not exist', 1
            END

        IF 1 = (SELECT IsCancelled FROM WorkshopBooking
            WHERE WorkshopBookingID = @WorkshopBookingID)
            BEGIN;
            THROW 52000, 'Booking has already been cancelled',1
            END


        UPDATE WorkshopBooking
        SET IsCancelled = 1
        WHERE @WorkshopBookingID = WorkshopBookingID
        END TRY
```

```
        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot cancel workshop booking' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.17 ChangeConferenceDayFreePlaces

Procedura zmieniająca limit miejsc na dany dzień konferencji

```
CREATE PROCEDURE [dbo].[PROC_ChangeConferenceDayFreePlaces]
        (@DayOfConferenceID int,
        @NewPeopleLimit int )
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM ConferenceDays
                        WHERE DayOfConferenceID = @DayOfConferenceID)
                BEGIN;
                THROW 52000, 'Such DayOfConferenceID does not exist', 1
                END
        IF (@NewPeopleLimit <=0)
        BEGIN;
        THROW 52000, 'People limit must be positive',1
        END

        IF (@NewPeopleLimit < (SELECT sum(NumberOfParticipants ) FROM ConferenceDayBooking
        WHERE DayOfConferenceID = @DayOfConferenceID  AND IsCancelled = 0))
        BEGIN;
        THROW 52000, 'Is impossible to reduce the people limit because places are booked',1
        END
        UPDATE ConferenceDays
        SET PeopleLimit = @NewPeopleLimit
        WHERE @DayOfConferenceID = DayOfConferenceID
        END TRY

        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot change free places ' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.18 ChangeWorkshopFreePlaces

Procedura zmieniająca limit osób na dany warsztat

```
CREATE PROCEDURE [dbo].[PROC_ChangeWorkshopFreePlaces]
        ( @WorkshopID int,
        @NewPeopleLimit int )
AS
BEGIN
SET NOCOUNT ON
        BEGIN TRY
        IF NOT EXISTS (SELECT * FROM Workshops
                        WHERE WorkshopID = @WorkshopID)
                BEGIN;
                THROW 52000, 'Workshop does not exist', 1
                END
        IF (@NewPeopleLimit <=0)
        BEGIN;
        THROW 52000, 'People limit must be positive',1
        END

        IF (@NewPeopleLimit < (SELECT sum(NumberOfParticipants ) FROM WorkshopBooking
        WHERE WorkshopID = @WorkshopID  AND IsCancelled = 0))
        BEGIN;
        THROW 52000, 'Is impossible to reduce the people limit because places are booked',1
        END
        UPDATE Workshops
        SET PeopleLimit = @NewPeopleLimit
        WHERE @WorkshopID = WorkshopID
        END TRY

        BEGIN CATCH
        DECLARE @errorMessage nvarchar (2048)
        SET @errorMessage = 'Cannot change free places ' + ERROR_MESSAGE();
        THROW 5200, @errorMessage, 1;
        END CATCH
END
GO
```

5.19 MostActiveClients

Procedura zwracająca listę klientów posortowaną wg ilości zarezerwowanych konferencji

```
CREATE PROCEDURE [dbo].[PROC_MostActiveClients] AS
BEGIN
SELECT c.ClientID, com.CompanyName as "Comapny name or client name",
cs.NumberOfConferenceBooking, '1' as "IsCompany"
FROM [dbo].[VIEW_ClientsStats] as cs
JOIN Clients AS c ON c.ClientID = cs.ClientID
JOIN Companies AS com on com.ClientID = c.ClientID

UNION
```

```
SELECT c.ClientID, ic.FirstName + ' ' + ic.LastName as "Comapny name or client name",
cs.NumberOfConferenceBooking, '0' as "IsCompany"
FROM [dbo].[VIEW_ClientsStats] as cs
JOIN Clients AS c ON c.ClientID = cs.ClientID
JOIN IndividualClients AS ic on ic.ClientID = c.ClientID

ORDER BY cs.NumberOfConferenceBooking DESC
END
GO
```

5.20 CancelUnpaidBookingAfter7Days

Procedura mająca na celu anulowanie nieopłaconych rezerwacji po czasie 7 dni od jej dokonania. Może być wywoływana okresowo w zależności od tego jak często chcemy uaktualniać dane.

```
CREATE PROCEDURE PROC_CancelUnpaidBookingAfter7Days
AS
UPDATE ConferenceBooking
SET IsCancelled = 1
WHERE 7 > DATEDIFF(DAY,GETDATE(),BookingDate)
        AND dbo.BookingValue (ConferenceBookingID)> dbo.PaymentsSum (ClientID,
ConferenceBookingID)
```

6. Widoki

6.1 MostPopularWorkshopInTheFuture

Zwraca najpopularniejsze warsztaty, które mają się odbyć w przyszłości

```
CREATE VIEW [dbo].[VIEW_MostPopularWorkshopsInTheFuture]
AS
(SELECT TOP 100 w.WorkshopID, w.Subject, w.PeopleLimit,
(select sum(wb.NumberOfParticipants)
from WorkshopBooking as wb where wb.IsCancelled = 0 and wb.WorkshopID = w.WorkshopID) as
Participants,
((isnull((select sum(wb.NumberOfParticipants)
from WorkshopBooking as wb where wb.IsCancelled = 0 and wb.WorkshopID = w.WorkshopID),0)*100)/
w.PeopleLimit)  as OccupancyRate
FROM Workshops AS w
WHERE (SELECT cd.Date FROM  ConferenceDays as cd where cd.DayOfConferenceID  =
w.DayOfConferenceID) > GETDATE()
order by 5 desc)
GO
```

6.2 MostPopularWorkshops

Widok pokazujące najpopularniejsze warsztaty wg współczynnika zapełnienia

```
CREATE VIEW [dbo].[MostPopularWorkshops]
AS
(SELECT TOP 100 w.WorkshopID, w.Subject, w.PeopleLimit,
(select sum(wb.NumberOfParticipants)
from WorkshopBooking as wb where wb.IsCancelled = 0 and wb.WorkshopID = w.WorkshopID) as
Participants,
(isnull((select sum(wb.NumberOfParticipants)
from WorkshopBooking as wb where wb.IsCancelled = 0 and wb.WorkshopID = w.WorkshopID),0)/
w.PeopleLimit) *100 as OccupancyRate
FROM Workshops AS w
order by 5 desc)
GO
```

6.3 Clients

Widok pokazujący wszystkich klientów wraz z nazwą i informacją czy jest firmą czy klientem indywidualnym.

```
CREATE VIEW [dbo].[VIEW_Clients]
AS
	SELECT c.ClientID, (SELECT(CompanyName) FROM Companies AS co
			WHERE co.ClientID = c.ClientID) AS ClientName,
			c.Country, c.City, c.Address, c.PostalCode, c.Email, c.Phone, '1' as IsCompany
	FROM Clients as c
	UNION
	SELECT c.ClientID, (SELECT(FirstName +' ' +LastName) FROM IndividualClients AS ic
			WHERE ic.ClientID = c.ClientID) AS ClientName,
			c.Country, c.City, c.Address, c.PostalCode, c.Email, c.Phone, '0' as IsCompany
	FROM Clients as c

GO
```

6.4 ClientsBalanceSummary

Widok podsumowujący, ile każdy klient powinien zapłacić za swoje rezerwacje i ile rzeczywiście zapłacił

	CompanyName or FirstName and Lastname	ClientID	SummaryValueOfAllBooking	SummaryOfAllPayments	Balance
1	Abate Hilliary	2692	6408,50	2958,00	-3450,50
2	Abate Wilhelm	2831	1665,00	1323,00	-342,00
3	Abrahamoff Jolyn	2395	879,00	569,00	-310,00
4	Abrahamoff Kenna	1206	4585,70	2471,00	-2114,70
5	Abrahamoff Laurie	1412	3238,00	1059,00	-2179,00
6	Abrahamoff Rolph	1280	1381,00	549,00	-832,00
7	Abrashkin Gregoor	2454	1744,60	547,00	-1197,60
8	Acutt Noby	2421	420,00	726,00	306,00
9	Adamo Octavia	2825	5984,00	2361,00	-3623,00
10	Aizik Dagny	2817	840,00	626,00	-214,00
11	Alanbrooke Gennifer	2600	3301,00	3170,00	-131,00

```
CREATE VIEW [dbo].[VIEW_ClientsBalanceSummary]
as
(
select (select com.CompanyName from Companies as com where com.ClientID = c.ClientID) as
"CompanyName or FirstName and Lastname ",
        c.ClientID as ClientID, sum(dbo.BookingValue(cb.ConferenceBookingID)) as
SummaryValueOfAllBooking, dbo.PaymentsSumOfClient (c.ClientID) as SummaryOfAllPayments,
        dbo.PaymentsSumOfClient (c.ClientID) - sum(dbo.BookingValue(cb.ConferenceBookingID))  as
Balance
from Clients as c
inner join ConferenceBooking as cb
on c.ClientID = cb.ClientID
where c.ClientID in (select com.ClientID from Companies as com)
group by c.ClientID

union
select (select ic.Firstname + ' ' + ic.LastName from IndividualClients as ic where ic.ClientID = c.ClientID) as
"CompanyName or FirstName and Lastname ",
        c.ClientID as ClientID, sum(dbo.BookingValue(cb.ConferenceBookingID)) as
SummaryValueOfAllBooking, dbo.PaymentsSumOfClient (c.ClientID) as SummaryOfAllPayments,
        dbo.PaymentsSumOfClient (c.ClientID) - sum(dbo.BookingValue(cb.ConferenceBookingID)) as
Balance
from Clients as c
inner join ConferenceBooking as cb
on c.ClientID = cb.ClientID
where c.ClientID in (select ic.ClientID from IndividualClients as ic)
group by c.ClientID

)
GO
```

6.5 ClientsThatPayToMuchForConferenceBooking

Wypisuje dane klientów oraz numer rezerwacji, na którą zostały wpłacone większe środki niż były wymagane

	CompanyName or FirstName and Lastname	ClientID	ConferenceBookingID	ExtraPaid
553	Capital One Financial Corporation	665	665	6,00
554	Carolina Trust BancShares, Inc.	716	716	163,00
555	Cathay General Bancorp	572	572	197,80
556	Cavco Industries, Inc.	163	163	244,00
557	CBOE Holdings, Inc.	205	205	356,00
558	CBS Corporation	130	130	175,50
559	Celgene Corporation	402	402	40,00
560	Cemtrex Inc.	102	102	181,80
561	Centene Corporation	448	448	161,00
562	Cesca Therapeutics Inc.	116	116	277,00
563	Chemung Financial Corp	155	155	330,00

```
CREATE VIEW [dbo].[VIEW_ClientsThatPayToMuchForConferenceBooking]
as
(
	select (select com.CompanyName from Companies as com where com.ClientID = c.ClientID) as
"CompanyName or FirstName and Lastname ",
	c.ClientID, cb.ConferenceBookingID,
	(select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) -
dbo.BookingValue (cb.ConferenceBookingID) as ExtraPaid
	from Clients as c
	inner join ConferenceBooking as cb
	on c.ClientID = cb.ConferenceBookingID
	where c.ClientID in (select com.ClientID from Companies as com where com.ClientID =
c.ClientID)
	AND
	dbo.BookingValue (cb.ConferenceBookingID) -
	(select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) <
0.00
	union
	select (select ic.FirstName + ' ' + ic.LastName from IndividualClients as ic where ic.ClientID =
c.ClientID) as "CompanyName or FirstName and Lastname ",
	c.ClientID, cb.ConferenceBookingID,
	(select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) -
dbo.BookingValue (cb.ConferenceBookingID) as ExtraPaid
	from Clients as c
	inner join ConferenceBooking as cb
	on c.ClientID = cb.ConferenceBookingID
	where c.ClientID in (select ic.ClientID from IndividualClients as ic where ic.ClientID = c.ClientID)
	AND
	dbo.BookingValue (cb.ConferenceBookingID) -
	(select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) <
0.00
	 )
GO
```

6.6 ClientsStats

Widok pokazujący ile konferencji zarezerwował dany klient oraz ile było osobowych obecności

```
CREATE VIEW VIEW_ClientsStats AS
	SELECT c.ClientID, count(*) AS NumberOfBooking,
	sum(tmp1.SingleParticipationsInDays) AS SingleParticipationsInDays
	FROM (SELECT cb.ClientID, (select isnull(sum(cdb.NumberOfParticipants),0)
                                                    AS SingleParticipationsInDays
		FROM ConferenceDayBooking AS cdb
		WHERE cdb.ConferenceBookingID = cb.ConferenceBookingID
			AND cdb.IsCancelled =0)AS SingleParticipationsInDays FROM
		ConferenceBooking AS cb ) AS tmp1
	inner join Clients AS c
	ON c.ClientID = tmp1.ClientID
	GROUP BY c.ClientID
GO
```

6.7 ClientsThatNotPayEnoughForConferenceBooking

Wypisuje dane klientów oraz numer rezerwacji, na którą zostały wpłacone mniejsze środki niż były wymagane

```
CREATE VIEW [dbo].[VIEW_ClientsThatNotPayEnoughForConferenceBooking]
as
(
	select (select com.CompanyName from Companies as com where com.ClientID = c.ClientID) as
"CompanyName or FirstName and Lastname ",
	c.ClientID, cb.ConferenceBookingID,
	(-1)*(select p.Value from Payments as p where p.ConferenceBookingID =
cb.ConferenceBookingID) + dbo.BookingValue (cb.ConferenceBookingID) as Deficit
	from Clients as c
	inner join ConferenceBooking as cb
	on c.ClientID = cb.ConferenceBookingID
	where c.ClientID in (select com.ClientID from Companies as com where com.ClientID =
c.ClientID)
	AND
	dbo.BookingValue (cb.ConferenceBookingID) -
	(select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) >
0.00
	union
	select (select ic.FirstName + ' ' + ic.LastName from IndividualClients as ic where ic.ClientID =
c.ClientID) as "CompanyName or FirstName and Lastname ",
	c.ClientID, cb.ConferenceBookingID,
	(-1)*(select p.Value from Payments as p where p.ConferenceBookingID =
cb.ConferenceBookingID) + dbo.BookingValue (cb.ConferenceBookingID) as Deficit
	from Clients as c
```

```
        inner join ConferenceBooking as cb
        on c.ClientID = cb.ConferenceBookingID
        where c.ClientID in (select ic.ClientID from IndividualClients as ic where ic.ClientID = c.ClientID)
        AND
        dbo.BookingValue (cb.ConferenceBookingID) -
        (select p.Value from Payments as p where p.ConferenceBookingID = cb.ConferenceBookingID) >
0.00
         )
GO
```

6.8 CompanyClientsToCall

Widok pokazujący dane kontaktowe klientów firmowych, którzy nie podali wszystkich uczestników dnia konferencji lub warsztatu który zaczyna się za 2 tygodnie

```
CREATE VIEW VIEW_CompanyClientsToCall as
select c.ClientID, c.Email, c.Phone, cb.ConferenceBookingID, cdb.ConferenceDayBookingID, -isnull((select
count(*) from DayReservations as dr
                where dr.ConferenceDayBookingID = cdb.ConferenceDayBookingID group by
dr.ConferenceDayBookingID),0)+ cdb.NumberOfParticipants as NumberOfDayReservationToComplete
from Clients as c
inner join ConferenceBooking as cb
on c.ClientID = cb.ClientID AND cb.IsCancelled = 0
inner join Conferences as con
on con.ConferenceID  = cb.ConferenceID
inner join ConferenceDayBooking as cdb
on cdb.ConferenceBookingID = cb.ConferenceBookingID
where DATEDIFF(day, GETDATE(), con.StartDate) > 0
        AND DATEDIFF(day, GETDATE(), con.StartDate) < 14
        AND c.ClientID  in (select com.ClientID from Companies as com)

        AND isnull((select count(*) from DayReservations as dr
                where dr.ConferenceDayBookingID = cdb.ConferenceDayBookingID group by
dr.ConferenceDayBookingID),0)
                < cdb.NumberOfParticipants
```

Widok pokazujący na jakie dni organizator niewłaściwie zaplanował liczbę miejsc na dzień konferencji oraz na warsztaty. Pokazywane są takie dni konferencji, w których prawdopodobna jest sytuacja, w której będą jeszcze dostępne miejsca na warsztaty, ale nie będzie już możliwości zapisania się na dzień konferencji.

```
CREATE VIEW [dbo].[VIEW_DaysOfConferenceThatDemandChangesLimits] as
        select cd.ConferenceID, cd.DayOfConferenceID, cd.Date, cd.PeopleLimit,
        (select sum(tmp.FreePlaces) from
                (select ([dbo].[FreePlacesForWorkshop] (w.WorkshopID )) as FreePlaces
                from Workshops as w
                where w.DayOfConferenceID = cd.DayOfConferenceID) as tmp)
```

```
        as NumberOfAvaibleWorkshopPlaces
    from ConferenceDays as cd
    where ([dbo].[FreePlacesForConferenceDay] (cd.DayOfConferenceID)) <
    (select sum(tmp.FreePlaces) from
        (select ([dbo].[FreePlacesForWorkshop] (w.WorkshopID )) as FreePlaces
        from Workshops as w
        where w.DayOfConferenceID = cd.DayOfConferenceID) as tmp)
GO
```

7. Triggery

7.1 CancellingConferenceBooking

Trigger anulujący rezerwację dni konferencji, gdy anulowano rezerwację konferencji.

```
ALTER TRIGGER [dbo].[CancellingConferenceBooking] on [dbo].[ConferenceBooking]
    AFTER UPDATE
    AS
    BEGIN
    SET NOCOUNT ON
    UPDATE ConferenceDayBooking SET IsCancelled = 1
        WHERE ConferenceBookingID IN (
            SELECT I.conferenceBookingID from inserted as I, deleted as d
            where i.IsCancelled = 1 AND D.IsCancelled = 0)
    END
```

7.2 CancellingConferenceDayBooking

Trigger anulujący rezerwacje warsztatu, gdy rezerwacja na dzień konferencji została anulowana.

```
ALTER TRIGGER [dbo].[CancellingConferenceDayBooking] on [dbo].[ConferenceDayBooking]
    AFTER UPDATE
    AS
    BEGIN
    SET NOCOUNT ON
    UPDATE WorkshopBooking SET IsCancelled = 1
        WHERE ConferenceDayBookingID IN
```

```
                ( SELECT I.ConferenceDayBookingID from inserted as I, deleted as d
                where i.IsCancelled = 1 AND d.IsCancelled  = 0 )
END
```

8. Indeksy

Oprócz indeksów zakładanych automatycznie przez klucze główne zdecydowaliśmy się założyć indeksy na wszystkich kluczach obcych. Charakterystyka danych w naszej bazie jest taka, iż w większości przypadków selektywność klucza obcego w tabelach jest bardzo dosyć znaczącą. Przykładowo przeciętnie tylko kilka z około 19000 rekordów w tabeli DayReservations zawiera ten sam klucz obcy ConferenceBookingID. Dzięki założeniu indeksu na tej kolumnie odpowiednie rekordy szybko zostaną namierzone w sensie odnalezienia klucza klastrowanego, który doczepiany jest do najniższego poziomu wszystkich indeksów nieklastrowanych. Za pomocą tego klucza klastrowanego wyszukane zostaną pozostałe atrybuty tabeli w znacznie szybszym czasie niż przeszukiwanie całej tabeli.

```
CREATE INDEX INDEX_ConferenceBooking_FK_ConferenceID  on  ConferenceBooking(ConferenceID);

CREATE INDEX INDEX_ConferenceBooking_FK_ClientID  on ConferenceBooking(ClientID);

CREATE INDEX INDEX_Payments_FK_ConferenceBookingID on Payments(ConferenceBookingID);

CREATE INDEX INDEX_ConferencesDays_FK_ConferenceID  on  ConferenceDays(ConferenceID);

CREATE INDEX INDEX_Workshops_FK_ConferenceDayID  on Workshops(DayOfConferenceID);

CREATE INDEX INDEX_ConferenceDayBooking_FK_DayOfConferenceID on
ConferenceDayBooking(DayOfConferenceID);

CREATE INDEX INDEX_ConferenceDayBooking_FK_ConferenceDayBookingID on
ConferenceDayBooking(ConferenceDayBookingID);

CREATE INDEX INDEX_WorkshopBooking_FK_ConferenceDayBookingID  on
WorkshopBooking(ConferenceDayBookingID);

CREATE INDEX INDEX_WorkshopBooking_FK_WorkshopID on WorkshopBooking(WorkshopID);

CREATE INDEX INDEX_DayReservations_FK_ConferenceDayBookingID on
DayReservations(ConferenceDayBookingID);

CREATE INDEX INDEX_DayReservations_FK_ParticipantID on DayReservations(ParticipantID);
```

```
CREATE INDEX INDEX_WorkshopReservations_FK_DayReservationID  on
WorkshopReservations(DayReservationID);

CREATE INDEX INDEX_WorkshopReservations_FK_WorkshopBookingID on
WorkshopReservations(WorkshopBookingID);
```

Zdecydowaliśmy się też założyć indeksy na polach LastName w tabeli IndividualClients oraz CompanyName w tabeli Companies, żeby przyspieszyć proces obsługi klienta, który może korzystać z podstawowych danych klienta takich jak imię, nazwisko i nazwa firmy.

```
CREATE INDEX INDEX_Composite_IndividualClients_FirstName_LastName
ON IndividualClients
(
  FirstName,
  LastName
)
CREATE INDEX INDEX_Companies_CompanyName on Companies(CompanyName);
```

9. Role i funkcje dla ról

9.1 Administrator

Nadzoruje całą bazę. Ma dostęp do wszystkich funkcji, widoków, procedur.

9.2 Pracownik firmy organizującej konferencje

Osoby odpowiedzialne za działalność firmy zajmującą się organizowaniem konferencji
Uprawnienia:

- dodawanie konferencji (AddConference),
- dodawanie dnia konferencji (AddConferenceDay),
- dodawanie warsztatu (AddWorkshop),
- najpopularniejsze warsztaty (MostPopularWorkshops),
- najbardziej aktywni klienci (MostActiveClients),
- zmiana wolnych miejsc (ChangeConferenceDayFreePlaces, ChangeWorkshopFreePlaces),
- statystyki klientów (ClientsStats),
- dodawanie progu cenowego (AddPrice),
- bilans wpłat klienta (ClientBalanceSummary),
- klienci którzy wpłacili za dużo/ za mało (ClientsThatPayToMuchForConferenceBooking, ClientsThatNotPayEnoughForConferenceBooking),
- dni które wymagają zmiany limiu (DaysOfConferenceThatDemandChangesLimits),

9.3 Klient

Uprawnienia:

- dodawanie klienta (AddClient, AddInividualClient, AddCompany),
- dodawanie uczestnika (AddParticipant),
- lista dni dla konferencji (DaysForConference),
- koszt konferencji (ConferecncePrice),
- koszt dnia konferencji (ConferenceDayPrice),
- koszt rezerwacji warsztatu (DayOfConferenceWorkshopBookingValue),
- wolne miejsca na dzień konferencoji (FreePlacesForConferenceDay),
- wolnie miejsca na warsztat (FreePlacesForWorkshop),
- dostępne warsztatów (WorkshopsInConferenceDay),
- bilans wpłat dla siebie (PaymentsSum, PaymentsSumOfClient),
- dodanie płatności (AddPayment),

9.4 Uczestnik

Uprawnienia:

- konferencje na które jest zapisany (ConferencesForParticipant),
- warsztaty na które jest zapisany (WorkshopsForParticipant).

10. Generator danych

Dane są generowane przez program napisany w Javie. Niepodzielne nazwy własne takie jak imiona, nazwiska, nazwy ulic, uniwersytetów zostało wygenerowane za pomocą serwisu Mackoroo (https://mockaroo.com/), a następnie wprowadzone do programu w Javie. Wszystkie wygenerowane nazwy konferencji i tematy warsztatów dotyczą języków programowania i zostały stworzone za pomocą prostego schematu „wzbudzające zainteresowanie hasło” + nazwa języka programowania. Następnie tworzone są kaskadowo rekordy do poszczególnych tabel w ten sposób, iż najpierw tworzone są dane do tabel bez kluczy obcych, a w dalszej kolejności do tych tabel, które zawierają klucze obce powiązane z tabeli, które mają już wygenerowane dane. Poprzez takie podejście jesteśmy w stanie wygenerować takie dane, które odpowiadają zależnościami między poszczególnymi tabelami z naszej bazy danych.
Wprowadziliśmy dane 2000 klientów indywidualnych, 1000 firm, oraz 5000 unikalnych uczestników. W bazie znajduje się 75 unikalnych konferencji, które odpowiadają mniej

więcej trzyletniej działalności firmy, która organizuje średnio 2 konferencje w miesiącu. Zgodnie z wymaganiami każda konferencja trwa średnio 2-3 dni oraz podczas każdego dnia odbywa się średnio 4 warsztaty.

Klasa z główną metodą, która generuje dane oraz zapisuje polecenia w języku sql do 6 plików tekstowych. Przy dodawaniu danych do bazy korzystaliśmy z utworzonych wcześniej procedur dodających.

10.1 TestProgram

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.sql.SQLOutput;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class TestProgram {

	public static void main(String[] args) throws ParseException,
IndexOutOfBoundsException, FileNotFoundException {
	PrintWriter outWriter1 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane1.txt");
	PrintWriter outWriter2 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane2.txt");
	PrintWriter outWriter3 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane3.txt");
	PrintWriter outWriter4 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane4.txt");
	PrintWriter outWriter5 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane5.txt");
	PrintWriter outWriter6 = new
PrintWriter("C:\\Users\\Admin\\Desktop\\Bazy\\Dane6.txt");

	RandomMaker r = new RandomMaker();

	String [] tabFirstNames =  {"Dagny" ,"Ashley",  "Zack",  "Wilhelm",
"Karry",  "Branden",  "Yevette",  "Melamie",  "Ema",  "Augustine",
"Gennifer",  "Christiano",  "Wilma",  "Louisette",  "Peadar",  "Sigrid",
"Kenna",  "Selig",  "Tulley",  "Adrian",  "Neysa",  "Berna",  "Amye",
"Harcourt",  "Harper",  "Gae",  "Francisca",  "Loutitia",  "Siward",
"Aubrette",  "Gabbie",  "Adham",  "Fin",  "Maximilien",  "Melissa",
"Aliza",  "Scarlett",  "Savina",  "Ruprecht",  "Bertha",  "Idaline",
```

```
"Wilt", "Michaelina", "Rozelle", "Cele", "Honoria", "Rip", "Luisa",
"Trudi", "Weidar", "Leodora", "Fawnia", "Susie", "Mollee", "Gregoor",
"Matthieu", "Enrique", "Sandro", "Gusty", "Sheela", "Yul",
"Friedrick", "Jeramie", "Vale", "Hilliary", "Odessa", "Neale",
"Octavia", "Desiree", "Decca", "Rianon", "Fred", "Cathrine",
"Roosevelt", "Rolph", "Jolyn", "Florina", "Devonne", "Gerry",
"Gaspard", "Noll", "Kevon", "Randie", "Ailee", "Timothea", "Darbie",
"Symon", "Monroe", "Nikoletta", "Dorine", "Happy", "Annabel",
"Charlene", "Lyndy", "Laurie", "Spense", "Engelbert", "Nisse",
"Noby", "Trenton"};
      List<String> listFirstNames = new
ArrayList<>(Arrays.asList(tabFirstNames));
      String [] tabLastNames = {"Bonefant", "Macklam", "Surgener",
"Romanin", "Itzcovich", "Ghirigori", "Gipp", "Botterman", "Caplan",
"Milbank", "Henriques", "Joost", "Beale", "Enderwick", "Muscat",
"Noir", "Middlemass", "Fonzone", "Billett", "Moralas", "Shurey",
"Twallin", "Maneylaws", "Peartree", "Kits", "Cuell", "Dallin",
"Kornes", "Niemiec", "Harner", "Tilt", "Simond", "Pawels",
"Hoodlass", "Papworth", "Fetherby", "Brucker", "Lukock",
"MacCallister", "Szach", "Gallant", "Quinane", "Temlett", "Marritt",
"Jirick", "Rushmare", "Bauckham", "McGinney", "Lukes", "Kield",
"Belderson", "Stanger", "Hollingby", "Sieur", "Pelos", "Du Pre",
"Dotson", "Stitch", "Wimbridge", "Banbury", "Pedroli", "Handscombe",
"Gianolo", "Marzelli", "Abrahamoff", "Bridgman", "Farrey", "Ginnane",
"Pattemore", "Mazin", "Muller", "Kinkaid", "Copnell", "Rawsthorne",
"Troke", "Goaks", "Madre", "Flippini", "Sergison", "Ley", "Blasdale",
"Temperton", "Alessandone", "Weddup", "Litt", "Terzza", "Cadogan",
"Ipplett", "Ismail", "Layne", "Franzonello", "Cappineer", "Muffen",
"Perrins", "Carwithan", "Simonaitis", "Girth", "Lanfranconi",
"Shurmer", "Flight", "McTrustie", "Stitle", "Wrixon", "Greenier",
"Benne", "Kamena", "Dani", "Leggate", "Bradly", "Wakely", "McGlew",
"Abrashkin", "Abate", "Dearle", "Docherty", "Dwyr", "Dennis",
"Jurasek", "Emanueli", "Dehn", "Raittie", "Codi", "Dacre",
"Somersett", "Goodread", "Walsham", "Leggett", "Lyttle", "Corran",
"Becke", "Carette", "Eadmeades", "Grushin", "Neylan", "Schermick",
"Tolle", "Labbet", "Shouler", "McRobert", "Shaw", "Goldstein",
"Itzchaky", "Dunhill", "Rraundl", "Shaw", "Groome", "De Beauchamp",
"MacQuarrie", "Hartnell", "Naisbet", "Pisculli", "Yewdall", "Alster",
"Brissenden", "Pudge", "Crace", "Gillmor", "Dunkerk", "Acutt",
"Pledger", "Hoffner", "Kaser", "Cristofori", "Lennarde", "Dalley",
"Demaid", "Onyon", "Sollas", "Jeffcoat", "Kellar", "Craigg",
"Worrell", "Gallaher", "Meachen", "Biford", "Bowkley", "Leverton",
"Alenin", "Renahan", "Moine", "O'Breen", "Budgeon", "Messager",
"Spyby", "Heap", "McComiskey", "Perassi", "Smout", "Hewkin",
"Mallock", "Hartman", "Kliement", "Stenning", "Kirvin", "Keelinge",
"Rawstron", "Breddy", "ducarme", "Tattersill", "Kauscher", "Currie",
"Demare", "Gerasch", "Winyard", "Paullin", "Wagstaff", "Whitmell",
"Kidby", "Davenhall", "Humphrey", "Fawssett", "Erbe", "Wallace",
"Wrightam", "Blade", "Merner", "Postans", "Bretton", "Kenelin",
"Wardle", "Cudworth", "Tembridge", "Audus", "Hatrey", "Skurray", "De
Domenici", "Eady", "Bleakman", "Plevey", "Ivain", "Pantridge",
```

"Lotterington", "Vallerine", "Wagg", "Sugg", "Ruffler", "Belone",
"Rosenhaupt", "Feitosa", "Billham", "Waith", "Ranklin", "Wilber",
"Kunze", "Lasty", "Filippi", "Vanyakin", "Yabsley", "McRavey",
"Roakes", "Nutting", "Leward", "Bage", "Pervew", "Isaksen",
"Ferrero", "Jee", "Risborough", "De Robertis", "Code", "Pybus",
"Stabler", "Cleaves", "Walder", "Rands", "Hardage", "Clowney",
"Farra", "Husher", "Bowie", "Chedzoy", "Gerbl", "Klasen", "Tigwell",
"Copes", "Hugnot", "Monnoyer", "Langdon", "Halgarth", "Maingot",
"Lehmann", "Dilks", "Sicha", "Sands", "Hayhoe", "Warre", "Goldie",
"Vannacci", "Ambrois", "Blackboro", "Dixie", "Tebbe", "Martinson",
"Sushams", "Gero", "Scripture", "Eakin", "Ilson", "Tisor", "Kybbye",
"Moubray", "Bowie", "Ferri", "McNaughton", "Tabour", "Holdren",
"Huntingdon", "McCleary", "Maggiori", "Bernucci", "Eger", "Sidnall",
"Carlill", "Burrows", "Dewsnap", "Lillegard", "Whitters", "Teasey",
"Midford", "Klimowicz", "Lovatt", "Parnby", "Flobert", "Recher",
"Wingham", "Poupard", "Giampietro", "Brickner", "Aleksahkin",
"Lovewell", "Spiteri", "Bleasdale", "Lathleiff", "Glasard", "Semmens",
"Swadon", "Slade", "Pock", "Frankom", "Myhan", "Schneidar",
"Grinyov", "Minithorpe", "Chamley", "Ephson", "Joice", "Partener",
"Robertacci", "Chelam", "Croisdall", "McOrkill", "Vasilic", "Allawy",
"Lorenzetto", "Sevior", "Craze", "Elldred", "Phoebe", "Odda",
"Kingsly", "MacInnes", "Hellis", "Pettiford", "Proffer", "Semiraz",
"Douse", "Potte", "Lownds", "Pendlington", "Wall", "Piccop",
"Cissell", "Noteyoung", "Jerdon", "Ricardon", "Gyurkovics", "Hugill",
"Rittmeyer", "Imort", "Mapis", "Pellett", "Lovekin", "Cloke",
"Ochterlony", "Garretts", "Bilverstone", "Champain", "Flohard",
"Morey", "Bryett", "Radeliffe", "Dellit", "Brice", "Toler", "Tovey",
"Huskisson", "Deavall", "Kerry", "Meran", "Portman", "Stiegars",
"Alanbrooke",
"Eastcott", "Andrelli", "Carrodus", "Kings", "Kilcoyne",
"Brewett", "Berkely", "Greasley", "Holtaway", "Smalcombe", "Bligh",
"Cattermoul", "Bonicelli", "Eeles", "Norres", "Gurnell", "Audsley",
"Hillitt", "Legrave", "Brookz", "Grosier", "Glowacha", "Blaker",
"Lamminam", "Roydon", "Thompstone", "Western", "O'Leagham", "Oaker",
"Tapson", "Tankin", "Karlolak", "Seegar", "Sennett", "Mauditt",
"Derisley", "Grieg", "McNeigh", "Lambarton", "Govier", "Bimson",
"Duerden", "Foote", "Lobley", "Rowbotham", "Le Claire", "Dargie",
"Richold", "Stadding", "Cankett", "Bernhard", "Graalman", "Yuryichev",
"de Mullett", "Bratt", "Faas", "Sarle", "Collomosse", "Stiff",
"Candish", "Gary", "Prudham", "Dulany", "Whapples", "Iddiens",
"Cruz", "Gorman", "Gut", "MacCumiskey", "Bollans", "Liebmann",
"Aspinal", "Midgley", "Gorini", "Glading", "Doghartie", "Aliberti",
"Coxon", "Gniewosz", "Oddey", "De Ferrari", "Sivill", "Thecham",
"Laffan", "Zupo", "Marking", "Panyer", "Jowle", "Pickless",
"Deverock", "Fennessy", "Worner", "O'Deoran", "Jacobowicz", "Askam",
"Brinkley", "Parffrey", "Weild", "Levinge", "Woolstenholmes",
"Bolgar", "Farenden", "Frude", "Broek", "MacPhaden", "Blazek",
"Vedekhin", "Widocks", "Baistow", "Atack", "Scholard", "Drydale", "Le
febre", "Huxham", "Billiard", "Cartmail", "Menhci", "Ropars",
"Whyard", "Rumin", "Stanbridge", "Trotter", "Stanbury", "Joslyn",
"Bayne", "Pudner", "Glazyer", "Rammell", "Cheatle", "Milesap",

"Naulls", "Axworthy", "Choldcroft", "Flack", "Dellenbroker",
"Demangel", "Levene", "Leban", "Padillo", "Yakovlev", "Grouer",
"Shutle", "Lindstedt", "Tivolier", "Marquet", "Lindfors",
"Braffington", "Crimpe", "Dysert", "Grunwall", "Avrasin", "Habbert",
"Clifton", "Lathleiffure", "Skipton", "Butter", "Finicj", "Igounet",
"Blincoe", "Kamena", "Badini", "Jeroch", "Jaume", "Cino", "Faley",
"Grubbe", "Royal", "Yakobowitz", "Weighell", "Possa", "Hodcroft",
"Locket", "Gianettini", "Curmi", "Cribbins", "Skeates", "Hollyard",
"Dy", "Cleary", "Blowfield", "Balderstone", "Vasnetsov", "Copping",
"Willmont", "Kerford", "Brendel", "Kenewel", "Kilshaw", "Micklem",
"Soan", "Chaves", "Amiable", "Hardisty", "Vigors", "Floweth",
"Ricciardi", "Annies", "Lanfranconi", "Kennealy", "Fedder",
"Champkins", "Gannicleff", "Vaughten", "Mollett", "Elmore", "Tod",
"Sidary", "Beany", "Hought", "Iredale", "Siddle", "Rockall",
"Prandini", "Westman", "Mansuer", "Struttman", "Blackborn", "Arkill",
"Merricks", "Etter", "Richardsson", "Darkins", "Caudle", "Riddall",
"Tylor", "Keam", "Harmson", "Bauman", "Whitrod", "Mainland",
"Gavigan", "Sharphouse", "Wikey", "Balassi", "Georgeon", "Gouge",
"Manis", "Girdler", "Elener", "Craik", "Butterly", "Denington",
"Flowith", "Paradise", "Tribe", "Sackey", "Van der Kruijs", "Wimmers",
"Giannazzo", "Radley", "Whitewood", "Morstatt", "Capstake",
"Rentcome", "Funnell", "Ordemann", "Wherry", "Sherwen", "Taffley",
"Scoggan", "McPhater", "Ioselev", "Kliment", "Brammer", "Fritter",
"Eatherton", "Mowen", "Deerr", "Kermath", "Cops", "Sultan",
"Lukovic", "Ector", "Ruck", "Ervin", "Dovidaitis", "Varden",
"Carslake", "Axton", "Sarfatti", "Bryett", "Meadmore", "Druery",
"Wingatt", "Betz", "Shawel", "Gynni", "Philimore", "Scoyne", "Kiffe",
"Ainscow", "Rosberg", "Sharples", "Rudge", "Gudeman", "Rowlinson",
"MacCarrane", "Nardi", "Bradane", "Gumn", "Spinnace", "Gaiter",
"Pirkis", "Spinige", "Boshers", "Cockarill", "Renshaw", "Jamrowicz",
"Hentze", "Bodle", "Dudderidge", "Docherty", "Northcote", "Isoldi",
"Wollrauch", "Maren", "Uman", "Cinderey", "Kliemann", "Mattiassi",
"Showering", "Lunck", "Portugal", "Blaszczak", "Dadge", "Connar",
"MacGlory", "Boecke", "Victor", "Kinneir", "De Witt", "Aizik",
"Trigwell", "Keighly", "De Winton", "Keavy", "Shrimpton", "Bowmaker",
"Schimann", "Willmer", "Attrill", "Sifflett", "Roaf", "Furmenger",
"Antao", "Melby", "Impy", "Sennett", "Duly", "Galle", "Fayerman",
"Fossitt", "Wegener", "Petherick", "Stonelake", "Mathwin", "Sturch",
"Labrom", "Neem", "Beavers", "Lebbon", "Carr", "Blune", "Pawfoot",
"Djurkovic", "Hawk", "Bonnaire", "Deakin", "Fritche", "Rentalll",
"Jenkerson", "Megroff", "Lapenna", "Evenden", "Marmon", "Durban",
"Grelka", "Murrow", "Seydlitz", "Blincow", "Haulkham", "Bracknall",
"Ivett", "Jakel", "Sibille", "Pepon", "Matisse", "Perring",
"McClancy", "Badland", "Florey", "Severs", "Pugsley", "Giacovazzo",
"Tomkin", "Imrie", "Ferrini", "Caldairou", "Di Biagio", "Sherlock",
"Yankov", "Livezey", "Westhofer", "Bispo", "Durrance", "Dick",
"Lawther", "Sarchwell", "Catonne", "Skeene", "Pounder", "Pressdee",
"Olivetti", "Tinline", "Stanyan", "Conti", "Rolph", "Skeath", "Boxe",
"Spurrier", "Gidman", "Kikke", "Breslau", "Spurrett", "Schimek",
"Digges", "Cammacke", "Scroyton", "Denslow", "Charnick", "Lopes",
"Dumini", "Joiris", "Bockman", "Fraczek", "Skivington", "Boosey",

```
"Cowderay", "Simenot", "Chaytor", "Berrie", "Cookley", "Graalmans",
"Yuryaev", "Delacourt",
            "Jelleman", "McQuie", "Jaskiewicz", "Scholes", "Ramsey",
"Hakonsson", "Noury", "Usmar", "Santhouse", "Dils", "Manilo",
"Pasticznyk", "Breed", "Holhouse", "Ghione", "Rumin", "Fassman", "Van
Hault", "Lynnett", "Muttock", "Poultney", "Dodswell", "Duckitt",
"Musgrave", "Phillipson", "Krollmann", "Hughf", "Duffit", "Malthouse",
"Dabell", "Puleston", "McClements", "Hencke", "Revett", "Szymoni",
"Ewbanche", "Erik", "Sothern", "Larrad", "Jewis", "Dubble",
"Clampton", "Maliffe", "Paley", "Sidaway", "Likely", "Kempson",
"Beavon", "Cowing", "Whitefoot", "Simoni", "Truse", "Stuchburie",
"Armour", "Slewcock", "Gallie", "Lang", "Gwinnel", "Doidge",
"Wilcockes", "Wakerley", "Dutton", "Leney", "Element", "Matches",
"Rosina", "Pittford", "Piggford", "Ringe", "Howse", "Choules",
"Duckinfield", "Heinzler", "Rolston", "Kiffe", "Mackinder", "Cisar",
"Reece", "Headrick", "Fancott", "Castagna", "Dolman", "Lisle",
"Dakin", "Edmonson", "Fawltey", "Riddles", "Scourgie", "Ellson",
"Hoggin", "Thomlinson", "Rowland", "Ramstead", "Andrewartha",
"Bardill", "Stegers", "Dellar", "Rhys", "Grills", "Murfill",
"Biffin", "Courteney", "Heersema", "Holyard", "Osboldstone",
"Overall", "MacTrustey", "Corradini", "Bellsham", "Tytler",
"Caldecott", "Bradshaw", "Diggle", "Rogerot", "Sandland", "Edson",
"Elsop", "Booker", "Blabber", "Spink", "Ansell", "Molson",
"Littleton", "Horsefield", "Lorek", "Lawland", "Tregona",
"Meiklejohn", "Ritchie", "MacNeish", "Antrim", "Salery", "Bramah",
"Coppen", "Adamo", "Vardey", "Stockney", "Denkel", "Faltskog",
"Mayler", "Boribal", "Attrill", "Evennett", "Hazelden", "McGowing",
"Chadburn", "Coase", "Shippard", "Sandiford", "Snowsill", "Balden",
"Lidgertwood", "VanBrugh"};
      List<String> listLastNames = new
ArrayList<>(Arrays.asList(tabLastNames));
      String [] tabStreets = {"Magnolia Avenue","JFK Street","West Street",
"Amsterdam Avenue","East Side","5th Avenue", "Baker Road"};
      List <String> listStreets = new
ArrayList<>(Arrays.asList(tabStreets));

      String [] tabCities = {"Boston", "Los Angeles", "Oklahoma City",
"Washington", "Miami", "Denver", "San Jose", "Atlanta",
            "Seatle","Cansas City","Dover","Chicago","Detroit","San
Antonio", "Charleston"};
         List <String> listCities = new
ArrayList<>(Arrays.asList(tabCities));
      String [] tabCompanies = {"First Trust RBA Quality Income ETF",
"KBR, Inc.", "Golar LNG Limited", "Cerecor Inc.", "HSBC Holdings plc",
"J P Morgan Chase & Co", "Cincinnati Bell Inc", "Blueknight Energy
Partners L.P., L.L.C.", "Natl Westminster Pfd", "Gilead Sciences, Inc.",
"First United Corporation", "BWX Technologies, Inc.", "Fluor
Corporation", "GTx, Inc.", "Cass Information Systems, Inc", "Vicor
Corporation", "Nano Dimension Ltd.", "Cohen & Steers Limited Duration
Preferred and Income Fund, Inc", "ClearSign Combustion Corporation",
"ProPetro Holding Corp.", "E.I. du Pont de Nemours and Company", "General
```

Motors Company", "VictoryShares US EQ Income Enhanced Volatility Wtd ETF", "Goldman Sachs Group, Inc. (The)", "Nuveen Ohio Quality Municipal Income Fund", "Everi Holdings Inc.", "Seres Therapeutics, Inc.", "Booz Allen Hamilton Holding Corporation", "Cobalt International Energy, Inc.", "Almost Family Inc", "Vanguard International High Dividend Yield ETF", "First Hawaiian, Inc.", "ClearBridge Energy MLP Fund Inc.", "RealNetworks, Inc.", "Rayonier Inc.", "China Cord Blood Corporation", "Pennsylvania Real Estate Investment Trust", "Saratoga Investment Corp", "Safe Bulkers, Inc", "UNIVERSAL INSURANCE HOLDINGS INC", "Pulse Biosciences, Inc", "Liberty Media Corporation", "Teekay LNG Partners L.P.", "SandRidge Energy, Inc.", "iShares MSCI Europe Financials Sector Index Fund", "Trillium Therapeutics Inc.", "Restaurant Brands International Inc.", "StealthGas, Inc.", "LaSalle Hotel Properties", "Nuveen Tax-Advantaged Total Return Strategy Fund", "Seabridge Gold, Inc.", "Open Text Corporation", "Wells Fargo & Company", "Esperion Therapeutics, Inc.", "Voya Global Advantage and Premium Opportunity Fund", "NCS Multistage Holdings, Inc.", "Red Lion Hotels Corporation", "DLH Holdings Corp.", "Hercules Capital, Inc.", "First Trust Emerging Markets Small Cap AlphaDEX Fund", "Yingli Green Energy Holding Company Limited", "Wheeler Real Estate Investment Trust, Inc.", "Five Below, Inc.", "Iron Mountain Incorporated", "Great Plains Energy Inc", "BlackRock Income Trust Inc. (The)", "Ascent Capital Group, Inc.", "Calamos Global Total Return Fund", "Ramco-Gershenson Properties Trust", "Total S.A.", "VALE S.A.", "Rosetta Genomics Ltd.", "Second Sight Medical Products, Inc.", "Aberdeen Greater China Fund, Inc.", "Buffalo Wild Wings, Inc.", "U.S. Bancorp", "MFS Special Value Trust", "VictoryShares Emerging Market Volatility Wtd ETF", "Edge Therapeutics, Inc.", "Eastman Chemical Company", "Lantheus Holdings, Inc.", "Toll Brothers Inc.", "XG Technology, Inc", "Quest Resource Holding Corporation.", "PartnerRe Ltd.", "Neovasc Inc.", "Legg Mason, Inc.", "Lilis Energy, Inc.", "Washington Federal, Inc.", "Five Oaks Investment Corp.", "JELD-WEN Holding, Inc.", "Christopher & Banks Corporation", "Meridian Waste Solutions, Inc", "Global X FinTech ETF", "Severn Bancorp Inc", "NETGEAR, Inc.", "Loxo Oncology, Inc.", "LM Funding America, Inc.", "Pacific Special Acquisition Corp.", "Accuray Incorporated", "Bunge Limited", "Cemtrex Inc.", "W.R. Berkley Corporation", "Marten Transport, Ltd.", "The First Bancshares, Inc.", "Phibro Animal Health Corporation", "Ryanair Holdings plc", "Sussex Bancorp", "Aware, Inc.", "Discover Financial Services", "Integra LifeSciences Holdings Corporation", "MarketAxess Holdings, Inc.", "Royal Bank Scotland plc (The)", "ZAGG Inc", "Cigna Corporation", "Cesca Therapeutics Inc.", "Bank of Hawaii Corporation", "Arlington Asset Investment Corp", "Berkshire Hathaway Inc.", "Eldorado Resorts, Inc.", "Energizer Holdings, Inc.", "Barclays Inverse US Treasury Composite ETN", "Sage Therapeutics, Inc.", "Mercury General Corporation", "NF Energy Saving Corporation", "Invacare Corporation", "B. Riley Financial, Inc.", "Guaranty Federal Bancshares, Inc.", "Equity Bancshares, Inc.", "CBS Corporation", "Ross Stores, Inc.", "Lawson Products, Inc.", "Terreno Realty Corporation", "Endologix, Inc.", "PIMCO California Municipal Income Fund", "Monster Digital, Inc.", "Computer Programs and Systems, Inc.", "Nuveen Senior Income Fund", "Choice Hotels International, Inc.", "First Community

Corporation", "Hecla Mining Company", "AtriCure, Inc.", "VanEck Vectors Pharmaceutical ETF", "Genie Energy Ltd.", "Old Second Bancorp, Inc.", "Grifols, S.A.", "AAON, Inc.",
"PC-Tel, Inc.", "China Petroleum & Chemical Corporation", "NextEra Energy, Inc.", "First Trust Senior Loan Fund ETF", "Aspen Insurance Holdings Limited", "Proto Labs, Inc.", "iPath US Treasury Long Bond Bull ETN", "Chemung Financial Corp", "Otelco Inc.", "Merrill Lynch Depositor, Inc.", "John Wiley & Sons, Inc.", "ZAIS Group Holdings, Inc.", "Whiting Petroleum Corporation", "HealthSouth Corporation", "NI Holdings, Inc.", "Cavco Industries, Inc.", "SK Telecom Co., Ltd.", "Kennedy-Wilson Holdings Inc.", "Protagonist Therapeutics, Inc.", "State Bank Financial Corporation.", "Kroger Company (The)", "Horizon Global Corporation", "RiverNorth Opportunities Fund, Inc.", "Nortel Inversora SA", "IQ Chaikin U.S. Small Cap ETF", "ServiceSource International, Inc.", "MACOM Technology Solutions Holdings, Inc.", "Eaton Vance Risk-Managed Diversified Equity Income Fund", "Yext, Inc.", "Envision Healthcare Corporation", "Acxiom Corporation", "EXFO Inc", "Federated National Holding Company", "Manitowoc Company, Inc. (The)", "Entertainment Gaming Asia Incorporated", "Gabelli Equity Trust, Inc. (The)", "Twenty-First Century Fox, Inc.", "Amtech Systems, Inc.", "bebe stores, inc.", "Atlas Air Worldwide Holdings", "Beazer Homes USA, Inc.", "Deutsche Bank AG", "Pacific Ethanol, Inc.", "Nordic American Offshore Ltd", "Highwoods Properties, Inc.", "Oncobiologics, Inc.", "UNIVERSAL INSURANCE HOLDINGS INC", "Prosperity Bancshares, Inc.", "Keryx Biopharmaceuticals, Inc.", "SunTrust Banks, Inc.", "BlackRock Investment Quality Municipal Trust Inc. (The)", "Entergy Louisiana, Inc.", "PowerShares Global Agriculture Portfolio", "NV5 Global, Inc.", "Innocoll Holdings", "EMCORE Corporation", "Seabridge Gold, Inc.", "CBOE Holdings, Inc.", "PNC Financial Services Group, Inc. (The)", "Concord Medical Services Holdings Limited", "Mitsubishi UFJ Financial Group Inc", "National Holdings Corporation", "Bemis Company, Inc.", "PartnerRe Ltd.", "Sears Canada Inc. ", "Panhandle Royalty Company", "Tejon Ranch Co", "Soligenix, Inc.", "Synovus Financial Corp.", "Merus N.V.", "Murphy USA Inc.", "Terex Corporation", "P.A.M. Transportation Services, Inc.", "Warrior Met Coal, Inc.", "First Trust Rising Dividend Achievers ETF", "Nuveen California Municipal Value Fund 2", "Pope Resources", "Rightside Group, Ltd.", "Mack-Cali Realty Corporation", "The Gabelli Healthcare & Wellness Trust", "FIRST REPUBLIC BANK", "Grupo Financiero Galicia S.A.", "AmTrust Financial Services, Inc.", "BlackRock Capital Investment Corporation", "PartnerRe Ltd.", "Avinger, Inc.", "Petrobras Argentina S.A.", "Sterling Bancorp", "FuelCell Energy, Inc.", "iShares MSCI EM ESG Optimized ETF", "Synthetic Fixed-Income Securities, Inc.", "PowerShares DWA Energy Momentum Portfolio", "Fortress Investment Group LLC", "Golden Entertainment, Inc.", "Gridsum Holding Inc.", "Kinder Morgan, Inc.", "Lincoln National Corporation", "Edge Therapeutics, Inc.", "Avista Healthcare Public Acquisition Corp.", "Okta, Inc.", "First Trust United Kingdom AlphaDEX Fund", "ClearBridge Large Cap Growth ESG ETF", "Och-Ziff Capital Management Group LLC", "Invesco Plc", "Randolph Bancorp, Inc.", "Primerica, Inc.", "Parke Bancorp, Inc.", "Jack Henry & Associates, Inc.", "Apache Corporation", "Scorpio Bulkers Inc.", "Regeneron Pharmaceuticals, Inc.", "Bellicum Pharmaceuticals, Inc.", "Westlake

Chemical Corporation", "Glu Mobile Inc.", "Olin Corporation", "Saia, Inc.", "Banco Bradesco Sa", "SecureWorks Corp.", "AmTrust Financial Services, Inc.", "Dime Community Bancshares, Inc.", "Calamos Strategic Total Return Fund", "Canadian Natural Resources Limited", "Neenah Paper, Inc.", "Protective Life Corporation", "City Office REIT, Inc.", "Bemis Company, Inc.", "Dominion Energy, Inc.", "Urstadt Biddle Properties Inc.", "Blackrock MuniYield Quality Fund III, Inc.", "Ardmore Shipping Corporation", "TOR Minerals International Inc", "Lennar Corporation", "TopBuild Corp.", "Denbury Resources Inc.", "McDonald's Corporation", "Kingstone Companies, Inc", "Vident Core U.S. Bond Strategy Fund", "Flaherty & Crumrine Total Return Fund Inc", "Brooks Automation, Inc.", "NuStar Logistics, L.P.", "Five Prime Therapeutics, Inc.", "Barings Corporate Investors", "Texas Capital Bancshares, Inc.", "e.l.f. Beauty, Inc.", "KalVista Pharmaceuticals, Inc.", "Cinemark Holdings Inc", "Nuveen New York Municipal Value Fund 2", "BHP Billiton Limited", "YuMe, Inc.", "TEGNA Inc.", "M&T Bank Corporation", "Barclays PLC", "XBiotech Inc.", "Celgene Corporation", "NGL ENERGY PARTNERS LP", "Natural Resource Partners LP", "Dunkin' Brands Group, Inc.", "Matador Resources Company", "Sprouts Farmers Market, Inc.",
"Cambrex Corporation", "Wheeler Real Estate Investment Trust, Inc.", "BioLineRx Ltd.", "CNH Industrial N.V.", "MCBC Holdings, Inc.", "MGM Growth Properties LLC", "MacroGenics, Inc.", "Scientific Games Corp", "Quinpario Acquisition Corp. 2", "Adams Diversified Equity Fund, Inc.", "First Trust NASDAQ-100 Equal Weighted Index Fund", "Southern Copper Corporation", "Credit Suisse AG", "E*TRADE Financial Corporation", "Sonic Foundry, Inc.", "Matlin & Partners Acquisition Corporation", "Hess Corporation", "Energy Focus, Inc.", "Benefitfocus, Inc.", "Gap, Inc. (The)", "Primo Water Corporation", "Landmark Infrastructure Partners LP", "DHX Media Ltd.", "Advaxis, Inc.", "NeuroDerm Ltd.", "Neonode Inc.", "CBOE Holdings, Inc.", "Stanley Black & Decker, Inc.", "SPX FLOW, Inc.", "Voya Emerging Markets High Income Dividend Equity Fund", "Prudential Financial, Inc.", "Customers Bancorp, Inc", "Wells Fargo & Company", "United Bancshares, Inc.", "National Research Corporation", "MidSouth Bancorp", "Pebblebrook Hotel Trust", "Connecticut Water Service, Inc.", "Honda Motor Company, Ltd.", "InnerWorkings, Inc.", "Alpha and Omega Semiconductor Limited", "BlackRock Resources", "X-Links Gold Shares Covered Call ETN", "M.D.C. Holdings, Inc.", "TPG Specialty Lending, Inc.", "Emclaire Financial Corp", "Opko Health, Inc.", "Protective Life Corporation", "Bassett Furniture Industries, Incorporated", "Mesa Laboratories, Inc.", "Asure Software Inc", "Tyson Foods, Inc.", "MannKind Corporation", "Papa Murphy's Holdings, Inc.", "Hersha Hospitality Trust", "Intersections, Inc.", "Xenon Pharmaceuticals Inc.", "Jack Henry & Associates, Inc.", "Ameris Bancorp", "Biolase, Inc.", "Infinity Pharmaceuticals, Inc.", "Fulgent Genetics, Inc.", "FibroGen, Inc", "XBiotech Inc.", "Pimco New York Municipal Income Fund II", "Bank Of New York Mellon Corporation (The)", "UMB Financial Corporation", "Alphabet Inc.", "Zix Corporation", "OUTFRONT Media Inc.", "Moxian, Inc.", "Jewett-Cameron Trading Company", "Ohr Pharmaceuticals, Inc.", "Brooks Automation, Inc.", "Adamis Pharmaceuticals Corporation", "BlackRock New York Investment Quality Municipal Trust Inc. (Th", "MFA Financial, Inc.", "Arena Pharmaceuticals, Inc.", "Capital City Bank

Group", "Iridium Communications Inc", "Forum Merger Corporation", "Eaton Vance NextShares Trust", "W.R. Berkley Corporation", "Accenture plc", "Forward Pharma A/S", "Dime Community Bancshares, Inc.", "Daseke, Inc.", "Steel Partners Holdings LP", "MercadoLibre, Inc.", "Zayo Group Holdings, Inc.", "Pzena Investment Management Inc", "Axon Enterprise, Inc.", "Expeditors International of Washington, Inc.", "Papa Murphy's Holdings, Inc.", "Occidental Petroleum Corporation", "Celgene Corporation", "United Community Banks, Inc.", "NGL ENERGY PARTNERS LP", "Changyou.com Limited", "Teledyne Technologies Incorporated", "Tesla, Inc. ", "Statoil ASA", "PowerShares DWA Basic Materials Momentum Portfolio", "Landmark Bancorp Inc.", "LSC Communications, Inc.", "CenterPoint Energy, Inc.", "Avista Healthcare Public Acquisition Corp.", "Verint Systems Inc.", "DTE Energy Company", "First Defiance Financial Corp.", "Liberty Tax, Inc.", "Esperion Therapeutics, Inc.", "Liberty Broadband Corporation", "Computer Task Group, Incorporated", "Willbros Group, Inc.", "Medley Capital Corporation", "Eleven Biotherapeutics, Inc.", "DDR Corp.", "The GDL Fund", "Neurotrope, Inc.", "AMC Networks Inc.", "Echo Global Logistics, Inc.", "Lake Shore Bancorp, Inc.", "The Travelers Companies, Inc.", "Mimecast Limited", "Global Partner Acquisition Corp.", "FinTech Acquisition Corp. II", "Nuveen AMT-Free Municipal Credit Income Fund", "NxStage Medical, Inc.", "American Financial Group, Inc.", "MSG Networks Inc.", "Asure Software Inc", "PNM Resources, Inc. (Holding Co.)", "Gap, Inc. (The)", "Spark Therapeutics, Inc.", "MSA Safety Incorporporated", "Scudder Global High Income Fund, Inc.", "Gabelli Dividend", "Molson Coors Brewing Company", "Ross Stores, Inc.", "The Hain Celestial Group, Inc.", "Centene Corporation", "Zynga Inc.", "Accenture plc", "Federated National Holding Company", "Woodward, Inc.", "GoPro, Inc.", "Arbor Realty Trust", "Korea Equity Fund, Inc.", "Zions Bancorporation", "MyoKardia, Inc.", "American River Bankshares", "Loews Corporation", "Dynex Capital, Inc.", "Fidelity Nasdaq Composite Index Tracking Stock", "Alaska Communications Systems Group, Inc.", "Monro Muffler Brake, Inc.", "Willis Towers Watson Public Limited Company", "CBL & Associates Properties, Inc.", "iShares PHLX SOX Semiconductor Sector Index Fund", "Sun Bancorp, Inc.", "Northern Trust Corporation", "iPath US Treasury Steepener ETN", "Varian Medical Systems, Inc.", "Trimble Inc.", "P.A.M. Transportation Services, Inc.", "Acadia Realty Trust", "KBL Merger Corp. IV", "BlackRock, Inc.", "Gladstone Commercial Corporation", "Wells Fargo & Company", "H&E Equipment Services, Inc.", "Donegal Group, Inc.", "Donaldson Company, Inc.", "Harmony Gold Mining Company Limited", "First Community Corporation", "Telecom Italia S.P.A.",
"VelocityShares VIX Short-Term ETN", "Eaton Vance High Income 2021 Target Term Trust", "Triple-S Management Corporation", "Forward Industries, Inc.", "Wellesley Bancorp, Inc.", "Rice Midstream Partners LP", "L Brands, Inc.", "RiceBran Technologies", "NI Holdings, Inc.", "AT&T Inc.", "Foresight Energy LP", "Southern Company (The)", "Ingredion Incorporated", "Saul Centers, Inc.", "Vail Resorts, Inc.", "Monotype Imaging Holdings Inc.", "Morgan Stanley", "Janus Henderson Group plc", "Herc Holdings Inc.", "Signature Bank", "Carbo Ceramics, Inc.", "OFG Bancorp", "Telefonica SA", "Arrow Financial Corporation", "Diageo plc", "BioTelemetry, Inc.", "Coty Inc.", "First Trust Managed Municipal ETF",

"Select Bancorp, Inc.", "Freshpet, Inc.", "RLJ Entertainment, Inc.", "Dreyfus Strategic Municipals, Inc.", "Northrop Grumman Corporation", "VanEck Vectors Generic Drugs ETF", "TOR Minerals International Inc", "Zynerba Pharmaceuticals, Inc.", "Pilgrim Pride Corporation", "ReWalk Robotics Ltd", "Home Bancorp, Inc.", "PNC Financial Services Group, Inc. (The)", "Chesapeake Utilities Corporation", "Commercial Vehicle Group, Inc.", "Regions Financial Corporation", "Energy Transfer Partners, L.P.", "Comstock Holding Companies, Inc.", "Hawaiian Holdings, Inc.", "Apple Inc.", "ManTech International Corporation", "Kindred Healthcare, Inc.", "Waters Corporation", "Tableau Software, Inc.", "Citrix Systems, Inc.", "ORBCOMM Inc.", "Zix Corporation", "Aralez Pharmaceuticals Inc.", "General Dynamics Corporation", "Arcos Dorados Holdings Inc.", "Highwoods Properties, Inc.", "RiceBran Technologies", "Crown Crafts, Inc.", "Tyler Technologies, Inc.", "Altria Group", "Ecolab Inc.", "Patriot National, Inc.", "OpGen, Inc.", "Dorian LPG Ltd.", "Rosehill Resources Inc.", "Royce Global Value Trust, Inc.", "TRI Pointe Group, Inc.", "FTD Companies, Inc.", "Nuveen Preferred and Income Term Fund", "WisdomTree Barclays Interest Rate Hedged U.S. Aggregate Bond F", "VOC Energy Trust", "Goldman Sachs Group, Inc. (The)", "SiteOne Landscape Supply, Inc.", "Harvest Capital Credit Corporation", "Delphi Automotive plc", "Tenneco Inc.", "Ringcentral, Inc.", "Galmed Pharmaceuticals Ltd.", "Ultrapar Participacoes S.A.", "CNH Industrial N.V.", "Lazard Ltd.", "Goldman Sachs Group, Inc. (The)", "HollyFrontier Corporation", "Alexandria Real Estate Equities, Inc.", "IDACORP, Inc.", "Dollar General Corporation", "Cathay General Bancorp", "Sophiris Bio, Inc.", "FairPoint Communications, Inc.", "Eleven Biotherapeutics, Inc.", "Astoria Financial Corporation", "T-Mobile US, Inc.", "IBERIABANK Corporation", "Eaton Vance Corporation", "MKS Instruments, Inc.", "Nuveen All Cap Energy MLP Opportunities Fund", "Ashford Hospitality Trust Inc", "Eros International PLC", "Oaktree Capital Group, LLC", "Access National Corporation", "Blackstone GSO Senior Floating Rate Term Fund", "Kirkland's, Inc.", "Colony NorthStar, Inc.", "MRC Global Inc.", "Idera Pharmaceuticals, Inc.", "InfoSonics Corp", "Southwestern Energy Company", "Nova Measuring Instruments Ltd.", "Synthesis Energy Systems, Inc.", "Brown & Brown, Inc.", "Alphatec Holdings, Inc.", "Del Taco Restaurants, Inc.", "Seaspan Corporation", "Nova Lifestyle, Inc", "Community Bankers Trust Corporation.", "Whitestone REIT", "Sun Communities, Inc.", "America Movil, S.A.B. de C.V.", "Murphy Oil Corporation", "First Financial Bankshares, Inc.", "ZAIS Group Holdings, Inc.", "BT Group plc", "Superior Uniform Group, Inc.", "1st Source Corporation", "iShares 20+ Year Treasury Bond ETF", "Manchester United Ltd.", "Town Sports International Holdings, Inc.", "Eiger BioPharmaceuticals, Inc.", "TE Connectivity Ltd.", "Ingredion Incorporated", "Chimerix, Inc.", "Bemis Company, Inc.", "Neos Therapeutics, Inc.", "Abercrombie & Fitch Company", "Brookdale Senior Living Inc.", "Credicorp Ltd.", "TE Connectivity Ltd.", "Danaos Corporation", "Blackrock Muni Intermediate Duration Fund Inc", "Brandywine Realty Trust", "JMP Group LLC", "Ecology and Environment, Inc.", "Tabula Rasa HealthCare, Inc.", "Eaton Vance NextShares Trust", "Bank of America Corporation", "MEI Pharma, Inc.", "Adobe Systems Incorporated", "Flowserve Corporation", "Telefonica SA", "Spring Bank

Pharmaceuticals, Inc.", "Telephone and Data Systems, Inc.", "Camping World Holdings, Inc.", "McDermott International, Inc.", "Magal Security Systems Ltd.", "Fairmount Santrol Holdings Inc.", "Mednax, Inc", "Apptio, Inc.", "Marcus Corporation (The)", "VictoryShares International High Div Volatility Wtd ETF", "Pain Therapeutics, Inc.", "Arc Logistic Partners LP", "Fastenal Company", "Ambac Financial Group, Inc.", "Ally Financial Inc.", "PAR Technology Corporation", "Enduro Royalty Trust", "TPG Pace Energy Holdings Corp.", "Abeona Therapeutics Inc.", "Eleven Biotherapeutics, Inc.", "Calamos Convertible and High Income Fund", "Pitney Bowes Inc.", "GigaMedia Limited", "Rand Capital Corporation", "Mesa Laboratories, Inc.", "Balchem Corporation", "Shenandoah Telecommunications Co", "Mercury General Corporation", "Meritage Corporation", "Resolute Forest Products Inc.", "Capital One Financial Corporation", "Guess?, Inc.", "TriMas Corporation", "Consolidated Edison Inc", "Public Storage", "Broadway Financial Corporation", "FirstService Corporation", "Kindred Biosciences, Inc.", "PIMCO Commercial Mortgage Securities Trust, Inc.", "Veeva Systems Inc.", "Brookdale Senior Living Inc.", "World Point Terminals, LP", "Crocs, Inc.", "Federal Agricultural Mortgage Corporation", "China Lodging Group, Limited", "Dynagas LNG Partners LP", "PowerShares S&P SmallCap Utilities Portfolio", "Landmark Infrastructure Partners LP", "Ascent Capital Group, Inc.", "Kinder Morgan, Inc.", "Valmont Industries, Inc.", "United States Cellular Corporation", "JMP Group LLC", "Horizon Technology Finance Corporation", "Xperi Corporation", "Playa Hotels & Resorts N.V.", "Telephone and Data Systems, Inc.", "Simulations Plus, Inc.", "Cytokinetics", "Incorporated", "New York REIT, Inc.", "ENGlobal Corporation", "Colgate-Palmolive Company", "VASCO Data Security International, Inc.", "Solar Capital Ltd.", "PTC Therapeutics, Inc.", "Zendesk, Inc.", "Boise Cascade, L.L.C.", "Oxbridge Re Holdings Limited", "NASDAQ TEST STOCK", "Transdigm Group Incorporated", "Sanofi", "Broadcom Limited", "OncoGenex Pharmaceuticals Inc.", "Versum Materials, Inc.", "Daseke, Inc.", "Magyar Bancorp, Inc.", "CPFL Energia S.A.", "CBS Corporation", "Comcast Corporation", "UNITIL Corporation", "Carolina Trust BancShares, Inc.", "Cardiovascular Systems, Inc.", "Dime Community Bancshares, Inc.", "BCE, Inc.", "NetSol Technologies Inc.", "Scorpio Tankers Inc.", "Deluxe Corporation", "The GDL Fund", "Great Plains Energy Inc", "First Trust High Yield Long/Short ETF", "Ivy NextShares", "First Financial Bankshares, Inc.", "Physicians Realty Trust", "The Hanover Insurance Group, Inc.", "Rayonier Advanced Materials Inc.", "VelocityShares Daily 2x VIX Medium-Term ETN", "Cimpress N.V", "Golden Ocean Group Limited", "Cara Therapeutics, Inc.", "FS Bancorp, Inc.", "Heritage Insurance Holdings, Inc.", "FARO Technologies, Inc.", "Advantage Oil & Gas Ltd", "Global X Health & Wellness Thematic ETF", "The Health and Fitness ETF", "Xcel Brands, Inc", "Westpac Banking Corporation", "The Charles Schwab Corporation", "CM Finance Inc", "BB&T Corporation", "Farmers National Banc Corp.", "Grupo Aeroportuario del Centro Norte S.A.B. de C.V.", "Managed Duration Investment Grade Municipal Fund", "Silgan Holdings Inc.", "Urban Edge Properties", "Aberdeen Japan Equity Fund, Inc. ", "Remark Holdings, Inc.", "Cabot Oil & Gas Corporation", "IRIDEX Corporation", "Hennessy Advisors, Inc.", "FuelCell Energy, Inc.", "First Internet Bancorp", "NanoString Technologies, Inc.",

"Goldman Sachs Group, Inc. (The)", "Clifton Bancorp Inc.", "Redhill Biopharma Ltd.", "Seneca Foods Corp.", "WisdomTree Emerging Markets Quality Dividend Growth Fund", "PowerShares DWA Tactical Multi-Asset Income Portfolio", "Entergy Texas Inc", "Atossa Genetics Inc.", "Capital One Financial Corporation", "NewLink Genetics Corporation", "P & F Industries, Inc.", "Boulevard Acquisition Corp. II", "Walgreens Boots Alliance, Inc.", "Yulong Eco-Materials Limited", "Euronav NV", "Celestica, Inc.", "Unity Bancorp, Inc.", "Ventas, Inc.",
"SeaChange International, Inc.", "SM Energy Company", "Barclays Inverse US Treasury Composite ETN", "Waste Connections, Inc.", "JetBlue Airways Corporation", "Tidewater Inc.", "Estee Lauder Companies, Inc. (The)", "Alexandria Real Estate Equities, Inc.", "Enterprise Bancorp Inc", "PowerShares Russell 1000 Low Beta Equal Weight Portfolio", "OraSure Technologies, Inc.", "iShares iBoxx $ High Yield ex Oil & Gas Corporate Bond ETF", "Fitbit, Inc.", "First Business Financial Services, Inc.", "Kratos Defense & Security Solutions, Inc.", "Envestnet, Inc", "RAIT Financial Trust", "NOW Inc.", "Tesoro Logistics LP", "Entergy Arkansas, Inc.", "ZIOPHARM Oncology Inc", "The Hanover Insurance Group, Inc.", "Jabil Inc.", "Avangrid, Inc.", "PFSweb, Inc.", "Orion Energy Systems, Inc.", "Liberty Media Corporation", "SORL Auto Parts, Inc.", "THL Credit, Inc.", "J. Alexander's Holdings, Inc.", "Qualstar Corporation", "Silicon Laboratories, Inc.", "RadNet, Inc.", "Asta Funding, Inc.", "BalckRock Taxable Municipal Bond Trust", "Cato Corporation (The)", "American Homes 4 Rent", "Legg Mason, Inc.", "ScanSource, Inc.", "Streamline Health Solutions, Inc.", "Virtusa Corporation", "CBIZ, Inc.", "OMNOVA Solutions Inc.", "Kingsway Financial Services, Inc.", "OFG Bancorp", "Sears Canada Inc. ", "TerraForm Power, Inc.", "Bank Of New York Mellon Corporation (The)", "Model N, Inc.", "Applied Genetic Technologies Corporation", "Ruby Tuesday, Inc.", "Equity Commonwealth", "Hewlett Packard Enterprise Company", "Nomura Holdings Inc ADR", "Dynegy Inc.", "MarketAxess Holdings, Inc.", "MFC Bancorp Ltd.", "Cancer Genetics, Inc.", "NextEra Energy, Inc.", "VelocityShares Daily Inverse VIX Short-Term ETN", "Natus Medical Incorporated", "Sensient Technologies Corporation", "MDU Resources Group, Inc.", "FIRST REPUBLIC BANK", "Home Federal Bancorp, Inc. of Louisiana", "BlackLine, Inc.", "Air Transport Services Group, Inc", "RPC, Inc.", "Steel Partners Holdings LP", "DoubleLine Income Solutions Fund",
"United Bankshares, Inc.", "Cliffs Natural Resources Inc.", "Horizon Technology Finance Corporation", "iShares FTSE EPRA/NAREIT Europe Index Fund", "Oxbridge Re Holdings Limited", "Hannon Armstrong Sustainable Infrastructure Capital, Inc.", "Eversource Energy", "Utah Medical Products, Inc.", "Telefonica SA", "RLJ Entertainment, Inc.", "Sunworks, Inc.", "Marlin Business Services Corp.", "Yum! Brands, Inc.", "Cohen & Steers Total Return Realty Fund, Inc.", "Fate Therapeutics, Inc.", "Mylan N.V.", "eBay Inc.", "Consolidated Communications Holdings, Inc.", "Portland General Electric Company", "lululemon athletica inc.", "Bridge Bancorp, Inc.", "Impax Laboratories, Inc.", "Targa Resources Partners LP", "American Tower Corporation (REIT)", "Pzena Investment Management Inc", "BT Group plc", "HC2 Holdings, Inc.", "Global Partners LP", "KMG Chemicals, Inc.", "Globus Medical, Inc.", "Assembly Biosciences, Inc.", "Therapix Biosciences Ltd.", "Datawatch Corporation",

"Scorpio Tankers Inc.", "Flexsteel Industries, Inc.", "Pzena Investment Management Inc", "Century Communities, Inc.", "Hanmi Financial Corporation", "Government Properties Income Trust", "Unifirst Corporation", "Glacier Bancorp, Inc.", "Atrion Corporation", "Insulet Corporation", "Goldman Sachs MLP Income Opportunities Fund", "Monmouth Real Estate Investment Corporation", "Vantiv, Inc.", "Cohen & Steers Total Return Realty Fund, Inc.", "SunTrust Banks, Inc.", "Southwestern Energy Company", "Aviat Networks, Inc.", "MidWestOne Financial Group, Inc.", "Old Second Bancorp, Inc.", "Ocwen Financial Corporation", "Cincinnati Bell Inc", "Aspen Insurance Holdings Limited", "Logitech International S.A.", "MGM Growth Properties LLC",
"Farmer Brothers Company", "Kaman Corporation", "Coty Inc.", "Modine Manufacturing Company", "Lions Gate Entertainment Corporation", "Dycom Industries, Inc.", "Chembio Diagnostics, Inc.", "Gabelli Global Small and Mid Cap Value Trust (The)", "Senior Housing Properties Trust", "iShares Nasdaq Biotechnology Index Fund", "Royal Bank Scotland plc (The)", "Univest Corporation of Pennsylvania", "Vornado Realty Trust", "Nuveen New York Municipal Value Fund, Inc.", "PowerShares DWA Developed Markets Momentum Portfolio", "General Electric Company", "Tenneco Inc.", "Hancock Holding Company", "Polar Power, Inc.", "Asia Pacific Wire & Cable Corporation Limited", "Wipro Limited", "The Meet Group, Inc.", "Investors Real Estate Trust", "Norwood Financial Corp.", "China Unicom (Hong Kong) Ltd", "CRA International,Inc.", "Maximus, Inc.", "Nuveen Connecticut Quality Municipal Income Fund", "EnteroMedics Inc.", "RAIT Financial Trust", "RAIT Financial Trust", "ROBO Global Robotics and Automation Index ETF", "New York Times Company (The)", "INVESCO MORTGAGE CAPITAL INC", "Salem Media Group, Inc.", "PPlus Trust", "HNI Corporation", "Flotek Industries, Inc.", "Bright Horizons Family Solutions Inc.", "BlackRock New York Investment Quality Municipal Trust Inc. (Th", "Yield10 Bioscience, Inc.", "Capital Product Partners L.P.", "New Ireland Fund, Inc. (The)", "Magic Software Enterprises Ltd.", "Invesco Senior Income Trust", "Thai Fund, Inc. (The)", "CSRA Inc.", "CSP Inc.", "Algonquin Power & Utilities Corp.", "City Holding Company", "Blackrock MuniHoldings New Jersey Insured Fund, Inc.", "Huntsman Corporation", "EQT GP Holdings, LP", "Pro-Dex, Inc.", "Alliance World Dollar Government Fund II", "Eastman Kodak Company", "Amec Plc Ord", "Alliance One International, Inc.", "DXC Technology Company", "Live Nation Entertainment, Inc.", "Kinsale Capital Group, Inc.", "Sucampo Pharmaceuticals, Inc.", "Nordic American Tankers Limited", "Eaton Vance Corporation", "New Senior Investment Group Inc.", "Lilis Energy, Inc.", "Templeton Global Income Fund, Inc.", "Insignia Systems, Inc.", "Group 1 Automotive, Inc.", "Delta Technology Holdings Limited", "Gladstone Land Corporation", "HFF, Inc.", "Columbus McKinnon Corporation", "Federal Agricultural Mortgage Corporation", "Francesca's Holdings Corporation", "CVR Energy Inc.", "Aemetis, Inc", "First Trust Large Cap Core AlphaDEX Fund", "Ampco-Pittsburgh Corporation", "Taylor Devices, Inc.", "PowerShares 1-30 Laddered Treasury Portfolio", "Oasmia Pharmaceutical AB", "United Community Bancorp", "Array BioPharma Inc.", "Enzo Biochem, Inc.", "Great Elm Capital Group, Inc. ", "Digital Turbine, Inc.", "Liberty Interactive Corporation", "Valhi, Inc.", "SI Financial Group, Inc.", "Biomerica, Inc.", "Realogy Holdings Corp.", "Century

```
Casinos, Inc.",  "Rayonier Advanced Materials Inc.",  "United Financial,
Inc.",  "CafePress Inc.",  "Federal Realty Investment Trust"
      };
      List <String> listCompanyNames = new
ArrayList<>(Arrays.asList(tabCompanies));

      List<Company> listCompanies = new ArrayList<>();
      for (int i = 0; i<1000;i++)
      {
            String companyName = listCompanyNames.get(i);
            String  nip = r.createNip();
            String email = r.createCompanyEmail(companyName);
            String phone = r.createPhoneNumber();
            String addres = r.randomString(listStreets) +" " +
r.smallNumber();
            String postal = r.createPostalCode();
            String  city = r.randomString (listCities);
            listCompanies.add(new Company
                  (i+1,companyName,nip,phone,email,addres,postal,city,
"USA"));

      }

      List<IndividualClient> listOfIndividualClients = new ArrayList<>();
      for (int i = 0; i<2000;i++)
      {
            String firstName = r.randomString (listFirstNames);
            String  lastName = r.randomString (listLastNames);
            String email = r.createEmail(firstName,lastName);
            String phone = r.createPhoneNumber();
            String addres = r.randomString(listStreets) +" " +
r.smallNumber();
            String postal = r.createPostalCode();
            String  city = r.randomString (listCities);
            listOfIndividualClients.add(new IndividualClient
                  (i+1001,firstName,lastName,phone,email,addres,postal,city,
"USA"));
      }


      List<Conference> listConferences = new ArrayList<>();
      List<Price> listPrices = new ArrayList<>();
      List<ConferenceDay> listConferenceDays = new ArrayList<>();
      List<Workshop> listWorkshops = new ArrayList<>();
      List <ConferenceBooking> listConferenceBooking = new ArrayList<>();
      List<ConferenceDayBooking> listConferenceDayBooking = new
ArrayList<>();
      List <WorkshopBooking> listWorkshopBooking = new ArrayList<>();
      List <Participant> listParticipant = new ArrayList<>();
      List <Payment> listPayment = new ArrayList<>();
      List<DayReservation> listDayReservation = new ArrayList<>();
```

```
        List<WorkshopReservation> listWorkshopReservation = new ArrayList<>();

        int workshopReservationCounter=0;
        int dayReservationCounter = 0;
        int paymentCounter = 0;
        int workshopCounter = 0;
        int priceCounter= 0;
        int conferenceDayCounter= 0;

        SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd/yyyy");
        String dateFirstString = "01/01/2015";
        Date iterDate = dateFormat.parse(dateFirstString);
        int k = 0;
        SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");

          while(k < 75)    // w tym warunku możemy ustalić dowolną ilość
konferencji
        {
              Random rand = new Random();
              String language = r.chooseLanduage();
              String conferenceName = r.createConferenceName(language);
              Calendar calendar = Calendar.getInstance();
              calendar.setTime(iterDate);
              int n = rand.nextInt (15) + 5;
              calendar.add(Calendar.DATE,n);
              iterDate = calendar.getTime();
              String startDate = dateFormat.format(iterDate);
              n = rand.nextInt (99) + 1;
              double studentDiscount = n*1.0/100;
              String city = r.randomString(listCities);
              String postalCode = r.createPostalCode();
              String address = r.randomString(listStreets) +" " +
r.smallNumber();
              String email = r.createCompanyEmail(language);
              calendar.setTime(iterDate);
              n = rand.nextInt (4) + 0;
              calendar.add(Calendar.DATE,n);
              iterDate = calendar.getTime();
              String endDate = dateFormat.format(iterDate);
              listConferences.add(new Conference
                    (language,k + 1,conferenceName,startDate,studentDiscount,
                              city,postalCode,address,email,endDate,n));
              n = rand.nextInt(30)+ 20;
              int value =n*10;
              for(int tmp = 0; tmp < 5; tmp++)
              {
              // progi cenowe są określone na 100 dni przed konferencją
              double valuePerDay = value*1.00 - tmp*20;
              Date priceDate = iterDate;
                  calendar.setTime(priceDate);
              calendar.add(calendar.DATE,-101 + tmp*20);
```

```
            priceDate = calendar.getTime();
            String priceStartDate = dateFormat.format(priceDate);
            calendar.add(calendar.DATE,19);
            priceDate = calendar.getTime();
            String priceEndDate = dateFormat.format(priceDate);
            Price cena = new Price (priceCounter+1,
                        k+1,valuePerDay,priceStartDate,priceEndDate);
            listPrices.add(cena);
                calendar.setTime(iterDate);
            priceCounter ++;
            }


            // dodawanie poszczególnych dni

            for(int tmp=0;tmp<=listConferences.get(k).getLenght();tmp++)
            {
            String beginning = listConferences.get(k).getStartDate();
            Date beginningDate = dateFormat.parse(beginning);
                calendar.setTime(beginningDate);
               calendar.add(Calendar.DATE,tmp);
            beginningDate = calendar.getTime();
            beginning = dateFormat.format(beginningDate);
            int peopleLimit = r.chooseConferenceDayPeopleLimit();

            ConferenceDay cd = new ConferenceDay(conferenceDayCounter+1,k+1,
                        peopleLimit,beginning);
                listConferenceDays.add(cd);


            // generowanie warszatów dla poszczególnych dni
            n = rand.nextInt(4) + 2;
                 String hour0 = "9:00:00";
            Date hourToIter = timeFormat.parse(hour0);
            Calendar timeCalendar = Calendar.getInstance();


            for(int i = 0; i<n;i++)
            {
                  int confID =
listConferenceDays.get(conferenceDayCounter).getConferenceID();
                  String languageWorkshop =
listConferences.get(confID-1).getLanguage();
                  String subject =
r.chooseWorkshopSubject(languageWorkshop);
                  String workshopStart = timeFormat.format(hourToIter);
                    timeCalendar.setTime(hourToIter);
                    timeCalendar.add(Calendar.MINUTE, 60);
                  hourToIter = timeCalendar.getTime();
                  String workshopEnd = timeFormat.format(hourToIter);
                    timeCalendar.setTime(hourToIter);
```

```
                    timeCalendar.add(Calendar.MINUTE,5);
                  hourToIter = timeCalendar.getTime();
                  Double workshopMoney = (rand.nextInt(20) + 10)*1.00;
                  int workshopPeopleLimit = rand.nextInt(12) + 10;
                  if(workshopPeopleLimit > 15)
                  {
                        workshopMoney = 0.00;
                  }

                  Workshop w = new
Workshop(workshopCounter+1,conferenceDayCounter+1,

subject,workshopStart,workshopEnd,workshopPeopleLimit,workshopMoney);
                  listWorkshops.add(w);
                  workshopCounter++;

                }

            conferenceDayCounter++;
                calendar.setTime(iterDate);
            }
            k++;
      }

      // generowanie rezerwacji ogólnych na konferencję
      //rezerwacji dokonują zarówną klienci indywidualni jak i grupowi
      Calendar bookingCalendar = Calendar.getInstance();
      int conferenceBookingCounter = 0;
      int conferenceBookingDayCounter= 0;
      int workshopBookingCounter=0;
      for(int i= 0; i<listConferences.size();i++)
      {
             Conference con = listConferences.get(i);
            Random rand = new Random();
            int n = rand.nextInt(960) + 30;
            int step = rand.nextInt (5) + 7;
            List<IndividualClient> selectedIndividual = new ArrayList<>();
              List <Company> selectedCompany = new ArrayList<>();
            List <ConferenceDay> selectedConferenceDay = new ArrayList<>();
            for (int icd = 0 ; icd<listConferenceDays.size();icd++)
            {

if(listConferenceDays.get(icd).getConferenceID()==con.getConferenceID())
            {
                  selectedConferenceDay.add(listConferenceDays.get(icd));
            }
            }

            for(int j = 0; j<10;j++)
            {
            selectedCompany.add(listCompanies.get(((n+j*step)%999)));
```

```
            }
            int kj = 0;
            while(kj<50)
            {

selectedIndividual.add(listOfIndividualClients.get(((n+1234+kj*step)%1999)));
            kj++;
            }

            kj = 0;
            while(kj<selectedCompany.size())
            {
            int selectedCompanyID = selectedCompany.get(kj).getCompanyID();
            String tmpStartDate = con.getStartDate();
            Date tmpStartDateDate = dateFormat.parse(tmpStartDate);
                bookingCalendar.setTime(tmpStartDateDate);
            n = rand.nextInt(50) + 15;
                bookingCalendar.add(Calendar.DATE,n);
            tmpStartDateDate = bookingCalendar.getTime();
                 tmpStartDate = dateFormat.format(tmpStartDateDate);


            ConferenceBooking cb = new
ConferenceBooking(conferenceBookingCounter+1,
                        con.getConferenceID(), selectedCompanyID,
tmpStartDate);
            listConferenceBooking.add(cb);
            kj++;
                conferenceBookingCounter++;

            for(int q = 0; q<selectedConferenceDay.size();q++)
            {
                  int dayOfConferenceeID =
selectedConferenceDay.get(q).getConferenceDayID();
                  n = rand.nextInt(4)+1;
                  int nprim = rand.nextInt(n);
                  ConferenceDayBooking cdb  =
                        new
ConferenceDayBooking(conferenceBookingDayCounter+1,

dayOfConferenceeID,conferenceBookingCounter,n,nprim);
                    listConferenceDayBooking.add(cdb);
                  List <Workshop> selectedWorkshops = new ArrayList<>();
                  for (int icd = 0 ; icd<listWorkshops.size();icd++)
                  {

                        if(listWorkshops.get(icd).getDayOfConferenceID() ==
cdb.getDayOfConferenceID())
                        {
                            selectedWorkshops.add(listWorkshops.get(icd));
```

```
                        }
                    }
                    for(int qwe = 0; qwe<selectedWorkshops.size();qwe++)
                    {
                        int selectedWorkshopID =
selectedWorkshops.get(qwe).getWorkshopID();
                       int possibleAttendees =
cdb.getNumberOfParticipants();
                        possibleAttendees = rand.nextInt(possibleAttendees);
                        if(possibleAttendees > 0)  possibleAttendees =
rand.nextInt(possibleAttendees);


if(listWorkshops.get(selectedWorkshopID-1).getCurrentOccupancy() +
possibleAttendees <=

listWorkshops.get(selectedWorkshopID-1).getPeopleLimit() &&
possibleAttendees>0)
                        {
                           WorkshopBooking wb =
                                    new
WorkshopBooking(workshopBookingCounter+1,

cdb.getConferenceDayBookingID(),selectedWorkshopID,possibleAttendees);
                        listWorkshopBooking.add(wb);
                        double actualWorkshopSummancy =
cdb.getPriceOfWorkshops();
                            cdb.setPriceOfWorkshops(actualWorkshopSummancy +

possibleAttendees*(listWorkshops.get(wb.getWorkshopID()-1).getPeopleLimit()))
;

                            workshopBookingCounter++;
                        int attendeesBefore =
listWorkshops.get(selectedWorkshopID-1).getCurrentOccupancy();
                        attendeesBefore = attendeesBefore +
possibleAttendees;

listWorkshops.get(selectedWorkshopID-1).setCurrentOccupancy(attendeesBefore);
                        }
                    }
                      conferenceBookingDayCounter++;
                   }

             }
             kj= 0;
             while(kj<selectedIndividual.size())
             {
             int selectedIndividualID =
selectedIndividual.get(kj).getIndividualClientID();
```

```
            String tmpStartDate = con.getStartDate();
            Date tmpStartDateDate = dateFormat.parse(tmpStartDate);
                bookingCalendar.setTime(tmpStartDateDate);
            n = rand.nextInt(50) + 15;
                bookingCalendar.add(Calendar.DATE,n);
            tmpStartDateDate = bookingCalendar.getTime();
            tmpStartDate = dateFormat.format(tmpStartDateDate);


            ConferenceBooking cb = new
ConferenceBooking(conferenceBookingCounter+1,
                        con.getConferenceID(), selectedIndividualID,
tmpStartDate);
                listConferenceBooking.add(cb);
            kj++;
                conferenceBookingCounter++;

            for(int q = 0; q<selectedConferenceDay.size();q++)
            {
                  int dayOfConferenceeID =
selectedConferenceDay.get(q).getConferenceDayID();
                  n = rand.nextInt(2)+1;
                  int nprim = rand.nextInt(n);
                  ConferenceDayBooking cdb  =
                        new
ConferenceDayBooking(conferenceBookingDayCounter+1,

dayOfConferenceeID,conferenceBookingCounter,n,nprim);
                    listConferenceDayBooking.add(cdb);
                  List <Workshop> selectedWorkshops = new ArrayList<>();

                  for (int icd = 0 ; icd<listWorkshops.size();icd++)
                  {

                        if(listWorkshops.get(icd).getDayOfConferenceID() ==
cdb.getDayOfConferenceID())
                        {
                        selectedWorkshops.add(listWorkshops.get(icd));

                        }
                  }
                  for(int qwe = 0; qwe<selectedWorkshops.size();qwe++)
                  {
                        int selectedWorkshopID =
selectedWorkshops.get(qwe).getWorkshopID();
                        int possibleAttendees =
cdb.getNumberOfParticipants();
                        possibleAttendees = rand.nextInt(possibleAttendees);

if(listWorkshops.get(selectedWorkshopID-1).getCurrentOccupancy() +
```

```
                                    possibleAttendees <=
listWorkshops.get(selectedWorkshopID-1).getPeopleLimit()
                                    && possibleAttendees > 0)
                            {
                             WorkshopBooking wb =
                                          new
WorkshopBooking(workshopBookingCounter+1,

cdb.getConferenceDayBookingID(),selectedWorkshopID,possibleAttendees);
                            listWorkshopBooking.add(wb);
                            double actualWorkshopSummancy =
cdb.getPriceOfWorkshops();
                               cdb.setPriceOfWorkshops(actualWorkshopSummancy +

possibleAttendees*(listWorkshops.get(wb.getWorkshopID()-1).getPeopleLimit()))
;
                               workshopBookingCounter++;
                            int attendeesBefore =
listWorkshops.get(selectedWorkshopID-1).getCurrentOccupancy();
                               attendeesBefore = attendeesBefore +
possibleAttendees;

listWorkshops.get(selectedWorkshopID-1).setCurrentOccupancy(attendeesBefore);
                            }
                        }


                          conferenceBookingDayCounter++;
                    }
              }
        }
        // wygenerowanie danych dla tabeli Participants
        for(int i=0 ; i<5000;i++)
        {
              String firstName = r.randomString (listFirstNames);
              String  lastName = r.randomString (listLastNames);
              String mail = r.createEmail(firstName,lastName);
              Participant p = new Participant(i+1,firstName, lastName, mail);
              listParticipant.add(p);
              System.out.println(p);
        }
        //generowanie rekordów w tabeli Payments
        for(int i=0;i<listConferenceBooking.size();i++)
        {
              ConferenceBooking cb = listConferenceBooking.get(i);
              Random rn = new Random();
              double valueOfBooking = 1.00*(rn.nextInt(234)+ 500);
          double discount =
listConferences.get(cb.getConferenceID()-1).getStudentDiscount();
              double pricePerDay = 0;
              for(int j=0; j<listPrices.size();j++)
```

```
            {
            Date d1 = dateFormat.parse(listPrices.get(j).getStartdate());
            Date d2 = dateFormat.parse(listPrices.get(j).getEndDate());
            Date dd = dateFormat.parse(cb.getBookingDate());
            if(dd.compareTo(d1)>=0 && dd.compareTo(d2) <= 0 &&
                        listPrices.get(j).getConferenceID() ==
cb.getConferenceID())
            {
                  pricePerDay = listPrices.get(j).getValuePerDay();
            }
            }
            List <ConferenceDayBooking> selectedConferenceDayBooking = new
ArrayList<>();
            for (int icd = 0 ;
icd<selectedConferenceDayBooking.size();icd++)
            {

            if(listConferenceDayBooking.get(icd).getConferenceBookingID() ==
cb.getConferenceBookingID());
            {

selectedConferenceDayBooking.add(listConferenceDayBooking.get(icd));
            }
            }
            for (int icd = 0 ;
icd<selectedConferenceDayBooking.size();icd++)
            {
            valueOfBooking = valueOfBooking +

(selectedConferenceDayBooking.get(icd).getNumberOfParticipants()-
selectedConferenceDayBooking.get(icd).getNumberOfStudents())*pricePerDay +

selectedConferenceDayBooking.get(icd).getNumberOfStudents()*pricePerDay*(1-di
scount);
            }
            Date lolo = dateFormat.parse(cb.getBookingDate());
            Calendar tmpCalendar = Calendar.getInstance();
            tmpCalendar.setTime(lolo);
            tmpCalendar.add(Calendar.DATE,2);
            lolo = tmpCalendar.getTime();
            String bookingDateToInsert = dateFormat.format(lolo);
            Payment p = new Payment(paymentCounter+1,valueOfBooking,
                    bookingDateToInsert,cb.getConferenceBookingID());
            listPayment.add(p);
            paymentCounter ++;
      }

      // wygenerowanie poszczególnych rekordów w DayReservation i
WorkshopReservation
      for(int i = 0; i<listConferenceDayBooking.size();i++)
      {
```

```
            ConferenceDayBooking cdb = listConferenceDayBooking.get(i);
            List <Integer> idiki = new ArrayList<>();

            List<WorkshopBooking> selectedWorkshopBooking = new
ArrayList<>();
            for(int j = 0 ; j<listWorkshopBooking.size(); j++)
            {
            if(listWorkshopBooking.get(j).getConferenceDayBookingID() ==
cdb.getConferenceDayBookingID())
            {
                    selectedWorkshopBooking.add(listWorkshopBooking.get(j));
            }
            }
            int normal = cdb.getNumberOfParticipants() -
cdb.getNumberOfStudents();
            int amaountOfStudents = cdb.getNumberOfStudents();

            for(int qaz = 0; qaz < normal; qaz++)
            {
            Random rand = new Random();
            int n = rand.nextInt(4889) + 1;

                DayReservation dr = new
DayReservation(dayReservationCounter+1,
                        cdb.getConferenceDayBookingID(),n,0,"NULL","NULL");
                listDayReservation.add(dr);
                idiki.add(dr.getDayReservationID());

                 dayReservationCounter++;
            }
            for(int qaz = 0; qaz < amaountOfStudents; qaz++)
            {
            Random rand = new Random();
            int n = rand.nextInt(4889) + 1;

            DayReservation dr =
                        new DayReservation(dayReservationCounter+1,

cdb.getConferenceDayBookingID(),n,1,r.createStudentCard(),
                                r.createUniversity());
                listDayReservation.add(dr);
                idiki.add(dr.getDayReservationID());
            dayReservationCounter++;
            }
            for(int wri = 0; wri<selectedWorkshopBooking.size();wri++)
            {
            WorkshopBooking wb = selectedWorkshopBooking.get(wri);
            List <Integer> alreadyUsed = new ArrayList<>();
            for(int edc = 0; edc < wb.getNumberOfParticipnats(); edc++)
            {
                  int quasiBool = 0;
```

```
                    int dayRervationUser = -1;
                        while(quasiBool == 0)
                    {
                          Random randWR = new Random();
                          int nridik = randWR.nextInt(idiki.size());
                          if(!alreadyUsed.contains(idiki.get(nridik)))
                          {
                              dayRervationUser = idiki.get(nridik);
                              alreadyUsed.add(dayRervationUser);
                          quasiBool = 1;
                          }
                    }
                  WorkshopReservation wr =
                          new
WorkshopReservation(workshopReservationCounter+1,

dayRervationUser,wb.getWorkshopBookingID());
                     listWorkshopReservation.add(wr);
                      workshopReservationCounter++;
             }
             }
      }

      // zapisanie danych do plików tekstowych

      for (int i = 0 ;i<listCompanies.size(); i++)
      {
            outWriter1.println(listCompanies.get(i));
       }

      for (int i = 0 ;i<listOfIndividualClients.size(); i++)
      {
            outWriter1.println(listOfIndividualClients.get(i));
      }

      for (int i = 0; i < listParticipant.size();i++)
      {
            outWriter1.println(listParticipant.get(i));
      }

      for (int i = 0 ;i<listConferences.size(); i++)
      {
            outWriter2.println(listConferences.get(i));
      }

      for (int i = 0 ;i<listPrices.size(); i++)
      {
            outWriter2.println(listPrices.get(i));
      }
      for (int i = 0 ;i<listConferenceDays.size(); i++)
      {
```

```
            outWriter2.println(listConferenceDays.get(i));
        }
        for (int i = 0 ;i<listWorkshops.size(); i++)
        {
             outWriter2.println(listWorkshops.get(i));
        }
        for (int i = 0 ;i<listConferenceBooking.size(); i++)
        {
            outWriter2.println(listConferenceBooking.get(i));
        }

        for (int i = 0 ;i<listConferenceDayBooking.size(); i++)
        {
            outWriter3.println(listConferenceDayBooking.get(i));
        }
        for (int i = 0 ;i<listWorkshopBooking.size(); i++)
        {
            outWriter3.println(listWorkshopBooking.get(i));
        }

        for (int i = 0 ;i<listDayReservation.size(); i++)
        {
            outWriter4.println(listDayReservation.get(i));
        }

        for (int i = 0 ;i<listWorkshopReservation.size(); i++)
        {
            outWriter5.println(listWorkshopReservation.get(i));
        }

        for (int i = 0 ;i<listPayment.size(); i++)
        {
            outWriter6.println(listPayment.get(i));
        }

        System.out.println("KONIEC");
        outWriter1.close();
        outWriter2.close();
        outWriter3.close();
         outWriter4.close();
        outWriter5.close();
        outWriter6.close();

        }
}
```

10.2 Klasa RandomMaker

Klasa pomocnicza RandomMaker - klasa pomocnicza służąca do wprowadzenia czynnika losowego

```
import java.sql.Date;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class RandomMaker {
   public String createEmail (String s1, String s2)
   {
       String [] tabEndings =
{"github.io","gmail.com","gov.com","microsoft.com","dailmail.com","onet.eu"
,"twiiter.com"};
       List<String> endings = new ArrayList<>(Arrays.asList(tabEndings));
       Random rand = new Random();
       int n = rand.nextInt(endings.size());
       s1 = s1 + "qwerty";
       s2 = s2 + "qwerty";
       String email = s1.substring(0,2) +
s2.substring(0,6)+"@"+endings.get(n);
       return email.toLowerCase();
   }

   public String createCompanyEmail (String s1)
   {
       String s2 =s1.replaceAll("\\s+","")+"language";
       String email = s2.substring(0,8)+"@gmail.com";
       return email.toLowerCase();

   }


   public String createUniversity()
   {
       String [] uniTab = {"Arizona State University, Downtown Phoenix
Campus",
               "American International College",
               "Columbia University",
               "Wilberforce University",
               "Universidade para o Desenvolvimento do Alto Vale do
Itajaí",
               "University of Dongola",
               "Cape Coast Polytechnic ",
               "Fundación Universitaria San Martín",
               "Oxford Brookes University",
               "University of Otago",
```

```
                "Wollega University",
            "London Business School",
            "Minhaj University Lahore",
            "Universidad Nacional de Ingeniería",
            "Medical University of South Carolina",
            "St. Joseph College",
            "National Kaohsiung First University of Science and Technology",
            "Universidad Nacional de Santa - Chimbote" };
        Random rand = new Random();
        int n = rand.nextInt(uniTab.length-2);
        return uniTab[n];


    }


    public String createPhoneNumber ()
    {
        Random rand = new Random();

        int n1 = rand.nextInt(809) + 101;
        int n2 =  rand.nextInt(809) + 101;
        int n3 = rand.nextInt(804) + 101;
        String phone = Integer.toString(n1) + Integer.toString(n2) +
Integer.toString(n3);
        return phone;

    }

    public String createStudentCard ()
    {
        Random rand = new Random();

        int n1 = rand.nextInt(808) + 101;
        int n2 =  rand.nextInt(817) + 101;
        String phone = Integer.toString(n1) + Integer.toString(n2);
        return phone;

    }




    public String createPostalCode ()
    {
        Random rand = new Random();

        int n1 = rand.nextInt(99999) + 10001;
        String p = Integer.toString(n1);
        return p;
```

```
    }

    public String smallNumber ()
    {
        Random rand = new Random();
        int n = rand.nextInt (100) + 1;
        String p = Integer.toString(n);
        return p;
    }


    public int chooseConferenceDayPeopleLimit ()
    {
        Random rand = new Random();
        int n = rand.nextInt (20) + 140;
        return n;
    }



    public  String randomString (List<String> list)
    {
        Random rand = new Random();
        int n = rand.nextInt(list.size());
        return list.get(n);
    }
    public String createNip ()
    {
        Random rand = new Random();
        int n = rand.nextInt (9) + 2;
        String nip = Integer.toString(n);
        nip = nip +createPhoneNumber();
        return nip;

    }
    public String chooseLanduage () {
        String[] languages = {"ActionScript", "Ada", "Agilent VEE", "Algol",
"Alice", "Angelscript", "Apex", "APL", "AppleScript", "Arc", "Arduino",
"ASP", "AspectJ", "Assembly", "ATLAS", "Augeas", "AutoHotkey", "AutoIt",
"AutoLISP", "Automator", "Avenue", "Awk", "Bash", "(Visual) Basic", "bc",
"BCPL", "BETA", "BlitzMax", "Boo", "Bourne Shell", "Bro", "C", "C Shell",
"C#", "C++", "C++/CLI", "C-Omega", "Caml", "Ceylon", "CFML", "cg", "Ch",
"CHILL", "CIL", "CL (OS/400)", "Clarion", "Clean", "Clipper", "Clojure",
"CLU", "COBOL", "Cobra", "CoffeeScript", "ColdFusion", "COMAL", "Common
Lisp", "Coq", "cT", "Curl", "D", "Dart", "DCL", "DCPU-16 ASM",
"Delphi/Object Pascal", "DiBOL", "Dylan", "E", "eC", "Ecl", "ECMAScript",
"EGL", "Eiffel", "Elixir", "Emacs Lisp", "Erlang", "Etoys", "Euphoria",
"EXEC", "F#", "Factor", "Falcon", "Fancy", "Fantom", "Felix", "Forth",
"Fortran", "Fortress", "(Visual) FoxPro", "Gambas", "GNU Octave", "Go",
"Google AppsScript", "Gosu", "Groovy", "Haskell", "haXe", "Heron", "HPL",
"HyperTalk", "Icon", "IDL", "Inform", "Informix-4GL", "INTERCAL", "Io",
```

```
"Ioke", "J", "J#", "JADE", "Java", "Java FX Script", "JavaScript",
"JScript", "JScript.NET", "Julia", "Korn Shell", "Kotlin", "LabVIEW",
"Ladder Logic", "Lasso", "Limbo", "Lingo", "Lisp",
               "Logo", "Logtalk", "LotusScript", "LPC", "Lua", "Lustre",
"M4", "MAD", "Magic", "Magik", "Malbolge", "MANTIS", "Maple",
"Mathematica", "MATLAB", "Max/MSP", "MAXScript", "MEL", "Mercury", "Mirah",
"Miva", "ML", "Monkey", "Modula-2", "Modula-3", "MOO", "Moto", "MS-DOS
Batch", "MUMPS", "NATURAL", "Nemerle", "Nimrod", "NQC", "NSIS", "Nu",
"NXT-G", "Oberon", "Object Rexx", "Objective-C", "Objective-J", "OCaml",
"Occam", "ooc", "Opa", "OpenCL", "OpenEdge ABL", "OPL", "Oz", "Paradox",
"Parrot", "Pascal", "Perl", "PHP", "Pike", "PILOT", "PL/I", "PL/SQL",
"Pliant", "PostScript", "POV-Ray", "PowerBasic", "PowerScript",
"PowerShell", "Processing", "Prolog", "Puppet", "Pure Data", "Python", "Q",
"R", "Racket", "REALBasic", "REBOL", "Revolution", "REXX", "RPG (OS/400)",
"Ruby", "Rust", "S", "S-PLUS", "SAS", "Sather", "Scala", "Scheme",
"Scilab", "Scratch", "sed", "Seed7", "Self", "Shell", "SIGNAL", "Simula",
"Simulink", "Slate", "Smalltalk", "Smarty", "SPARK", "SPSS", "SQR",
"Squeak", "Squirrel", "Standard ML", "Suneido", "SuperCollider", "TACL",
"Tcl", "Tex", "thinBasic", "TOM", "Transact-SQL", "Turing", "TypeScript",
"Vala/Genie", "VBScript", "Verilog", "VHDL", "VimL", "Visual Basic .NET",
"WebDNA", "Whitespace", "X10", "xBase", "XBase++", "Xen", "XPL", "XSLT",
"XQuery", "yacc", "Yorick", "Z shell"};
       List<String> listLanguages = new
ArrayList<>(Arrays.asList(languages));
       Random rand = new Random();
       int n1 = rand.nextInt(listLanguages.size());
       return listLanguages.get(n1);

   }



   public String createConferenceName (String language)
   {
       String [] prompts = {"10th anniversary of ", "The future of ", "Key
to carrer succes - ", "15th anniversary of ",
       "4th international conference on ", "Annual conference on "};

       List<String> listPrompts = new ArrayList<>(Arrays.asList(prompts));
       Random rand = new Random();

       int n2 = rand.nextInt(listPrompts.size());
       String name =  listPrompts.get(n2) + language + " programming
language";
       return name;


   }

   public Date addDays (Date starting)
   {
```

```
        int dzien = starting.getDay();
        return starting;
    }
    public String chooseWorkshopSubject (String language)
    {
        String [] tabPrompts = {"Business usage of ",
                "Improve your programming skills in ",
                "The secrets of ",
                "Questions You always want to ask about ",
                "Tips and Hints about ",
                "Changes in newest version of ",
                "Object Oriented Programming and ",
                "Procedural Programming and ",
                "Function Programming and "};
        List<String> prompts = new ArrayList<>(Arrays.asList(tabPrompts));
        Random rand = new Random();

        int n2 = rand.nextInt(prompts.size());
        String subject = prompts.get(n2) + language;
        return subject;
    }

}
```

10.3 Klasy odpowiadające poszczególnym tabelom

10.3.1 Klasa Company

```
public class Company {
      private int companyID;
      private String companyName;
      private String nip;
      private String phone;
      private String email;
      private String address;
      private String postalCode;
      private String city;
      private String country;

      public Company(int companyID, String companyName, String nip, String
phone, String email, String address, String postalCode, String city, String
country) {
      this.companyID = companyID;
      this.companyName = companyName;
      this.nip = nip;
```

```
        this.phone = phone;
        this.email = email;
        this.address = address;
        this.postalCode = postalCode;
        this.city = city;
        this.country = country;
        }

        public int getCompanyID() {
        return companyID;
        }

        public String getCompanyName() {
        return companyName;
        }

        public String getNip() {
        return nip;
        }

        public String getPhone() {
        return phone;
        }

        public String getEmail() {
        return email;
        }

        public String getAddress() {
        return address;
        }

        public String getPostalCode() {
        return postalCode;
        }

        public String getCity() {
        return city;
        }

        public String getCountry() {
        return country;
        }

        @Override
        public String toString() {
        return "exec dbo.Proc_AddClient " +
              "'" + phone + "', '" + email.replaceAll("
","").replaceAll("\'","")
              + "' ,'" + address.replaceAll("\'","") + "', '" +
```

```
            postalCode  + "', '" +  city.replaceAll("\'","") + "', '" +
country + "'" + '\n'
            + "exec dbo.Proc_AddCompany '" + companyID + "', '" +
companyName.replaceAll("\'","")
            + "', '" + nip  + "'";
      }
}
```

10.3.2 Klasa Conference

```
public class Conference {

      private String language;
      private int conferenceID;
      private String name;
      private String startDate;
      private double studentDiscount;
      private String city;
      private String postalCode;
      private String address;
      private String email;
      private String  endDate;
      private int lenght;


      public String getLanguage() {

      return language;

      }



      public double getStudentDiscount() {

      return studentDiscount;

      }



      public String getCity() {

      return city;

      }
```

```
public String getPostalCode() {

return postalCode;

}



public String getAddress() {

return address;

}



public String getEmail() {

return email;

}



public String getEndDate() {

return endDate;

}



public int getLenght() {

return lenght;

}

public int getConferenceID() {
return conferenceID;
}

public String getName() {
return name;
}

public String getStartDate() {
return startDate;
}
public Conference(String language, int conferenceID, String name,
```

```
String startDate, double studentDiscount, String city, String postalCode,
String address, String email, String endDate, int lenght) {

	this.language = language;
	this.conferenceID = conferenceID;
	this.name = name;
	this.startDate = startDate;
	this.studentDiscount = studentDiscount;
	this.city = city;
	this.postalCode = postalCode;
	this.address = address;
	this.email = email;
	this.endDate = endDate;
	this.lenght = lenght;



	}

	@Override

	public String toString() {
	return "exec dbo.PROC_AddConference '"+ name + "', '" + startDate +
"', '" + studentDiscount + "', '"+ city + "', '" + postalCode + "', '" +
address + "', '"

		+ email +

		"', '" + endDate + "'";

	}
}
```

10.3.3 Klasa ConferenceBooking

```
public class ConferenceBooking {

	private int conferenceBookingID;
	private int conferenceID;
	private int clientID;
	private String bookingDate;

	public ConferenceBooking(int conferenceBookingID, int conferenceID,
int clientID, String bookingDate) {
	this.conferenceBookingID = conferenceBookingID;
	this.conferenceID = conferenceID;
	this.clientID = clientID;
	this.bookingDate = bookingDate;
	}
```

```
	public int getConferenceBookingID() {
	return conferenceBookingID;
	}



	public int getConferenceID() {
	return conferenceID;
	}



	public int getClientID() {
	return clientID;
	}

	public String getBookingDate() {
	return bookingDate;
	}

	@Override

	public String toString() {

	return "exec dbo.Proc_AddConferenceBooking '" + conferenceID + "', '"
+ clientID + "', '" + bookingDate + "'";
	}
}
```

10.3.4 Klasa ConferenceDay

```
public class ConferenceDay {

	private int conferenceDayID;
	private int conferenceID;
	private int peopleLimit;
	private String dateOfConferenceDay;

	public int getConferenceDayID() {
	return conferenceDayID;
	}

	public int getConferenceID() {
	return conferenceID;
	}

	public int getPeopleLimit() {
	return peopleLimit;
	}
```

```
	public String getDateOfConferenceDay() {
	return dateOfConferenceDay;
	}



	public ConferenceDay(int conferenceDayID, int conferenceID, int
peopleLimit, String dateOfConferenceDay) {

	this.conferenceDayID = conferenceDayID;
	this.conferenceID = conferenceID;
	this.peopleLimit = peopleLimit;
	this.dateOfConferenceDay = dateOfConferenceDay;
	}
	@Override
	public String toString() {
	  return "exec dbo.Proc_AddConferenceDay '" + conferenceID + "', '" +
peopleLimit + "', '" + dateOfConferenceDay + "'";
	}
}
```

10.3.5 Klasa ConferenceDayBooking

```
public class ConferenceDayBooking {

	private int conferenceDayBookingID;
	private int dayOfConferenceID;
	private int conferenceBookingID;
	private int numberOfParticipants;
	private int numberOfStudents;
	private double priceOfWorkshops;



	public ConferenceDayBooking(int conferenceDayBookingID, int
dayOfConferenceID, int conferenceBookingID, int numberOfParticipants, int
numberOfStudents) {

	this.conferenceDayBookingID = conferenceDayBookingID;
	this.dayOfConferenceID = dayOfConferenceID;
	this.conferenceBookingID = conferenceBookingID;
	this.numberOfParticipants = numberOfParticipants;
	this.numberOfStudents = numberOfStudents;
	this.priceOfWorkshops = 0;

	}
```

```
	public double getPriceOfWorkshops() {
	return priceOfWorkshops;
	}

	public void setPriceOfWorkshops(double priceOfWorkshops) {
	this.priceOfWorkshops = priceOfWorkshops;
	}

	public int getDayOfConferenceID() {
	return dayOfConferenceID;
	}

	public int getConferenceDayBookingID() {
	return conferenceDayBookingID;
	}



	public int getConferenceBookingID() {

	return conferenceBookingID;

	}



	public int getNumberOfParticipants() {

	return numberOfParticipants;

	}



	public int getNumberOfStudents() {

	return numberOfStudents;

	}



	@Override

	public String toString() {

	return "exec dbo.Proc_AddConferenceDayBooking '" + dayOfConferenceID
+"', '" + conferenceBookingID + "', '"

			+ numberOfParticipants +
```

```
            "', '" + numberOfStudents + "'";
      }


      }
```

10.3.6 Klasa DayReservation

```
public class DayReservation {


      private int dayReservationID;

      private int conferenceDayBookingID;

      private int participantID;

      private int isStudent;

      private String studentCard;

      private String university;


      @Override

      public String toString() {

      return "exec [dbo].[PROC_AddDayReservation] '"  +
conferenceDayBookingID +

            "', '" + participantID +  "', '"

                  + isStudent +  "', '"

            + studentCard +  "', '"

            + university + "'";

      }


      public DayReservation(int dayReservationID, int
```

```
conferenceDayBookingID, int participantID, int isStudent, String
studentCard, String university) {

        this.dayReservationID = dayReservationID;
        this.conferenceDayBookingID = conferenceDayBookingID;
        this.participantID = participantID;
        this.isStudent = isStudent;
        this.studentCard = studentCard;
        this.university = university;
        }
        public int getDayReservationID() {
        return dayReservationID;
        }
        public int getConferenceDayBookingID() {
        return conferenceDayBookingID;

        }

        public int getParticipantID() {
        return participantID
        }

        public int getIsStudent() {
        return isStudent;

        }

        public String getStudentCard() {
        return studentCard;

        }



        public String getUniversity() {

        return university;

        }

}
```

10.3.7 Klasa IndividualClient

```
public class IndividualClient {
   private int individualClientID;
   private String firstanme;
   private String lastname;
   private String phone;
   private String email;
   private String address;
```

```
    private String postalCode;
    private String city;
    private String country;

    public IndividualClient(int individualClientID, String firstanme, String
lastname, String phone, String email, String address, String postalCode,
String city, String country) {
        this.individualClientID = individualClientID;
        this.firstanme = firstanme;
        this.lastname = lastname;
        this.phone = phone;
        this.email = email;
        this.address = address;
        this.postalCode = postalCode;
        this.city = city;
        this.country = country;
    }

    public int getIndividualClientID() {
        return individualClientID;
    }


    @Override
    public String toString() {
        return "exec dbo.Proc_AddClient " +
                "'" + phone + "', '" + email.replaceAll("\'","").replaceAll("
","") + "' ,'" + address.replaceAll("\'","") + "', '" +  postalCode  + "',
'" +  city.replaceAll("\'","") + "', '" + country + "'" + '\n'
                + "exec dbo.Proc_AddIndividualClient '"+ individualClientID
+ "', '" + firstanme.replaceAll("\'","") + "', ' " +
lastname.replaceAll("\'","") + "'";
    }
}
```

10.3.8 Klasa Participant

```
public class Participant {

      private int participantID;
      private String lastName;
      private String firstName;
      private String email;



      public Participant(int participantID, String lastName, String
firstName, String email) {

      this.participantID = participantID;
      this.lastName = lastName;
```

```
        this.firstName = firstName;
        this.email = email;

        }

        @Override

        public String toString() {

         return "exec [dbo].[PROC_AddParticipant] '" +
lastName.replaceAll("\'","") + "', '" +

              firstName.replaceAll("\'","") + "','" +
email.replaceAll("\'","").replaceAll(" ","") + "'";
        }
}
```

10.3.9 Klasa Payment

```
public class Payment {

        private int paymentID;
        private double valueOfPayment;
        private String paymentDate;
        private int conferenceBookingID;



        public Payment(int paymentID, double valueOfPayment, String
paymentDate, int conferenceBookingID) {

        this.paymentID = paymentID;
        this.valueOfPayment = valueOfPayment;
        this.paymentDate = paymentDate;
        this.conferenceBookingID = conferenceBookingID;

        }



        @Override

        public String toString() {

        return "exec [dbo].[PROC_AddPayment] '" + valueOfPayment + "', '"+
paymentDate + "', '"

              + conferenceBookingID + "'";
        }
}
```

10.3.10 Klasa Price

```
public class Price {

	private int priceID;
	private int conferenceID;
	private double valuePerDay;
	private String startdate;
	private String endDate;



	public Price(int priceID, int conferenceID, double valuePerDay, String
startdate, String endDate) {

	this.priceID = priceID;
	this.conferenceID = conferenceID;
	this.valuePerDay = valuePerDay;
	this.startdate = startdate;
	this.endDate = endDate;

	}



	public int getPriceID() {

	return priceID;

	}



	public int getConferenceID() {

	return conferenceID;

	}



	public double getValuePerDay() {

	return valuePerDay;

	}


	public String getStartdate() {

	return startdate;
```

```
	}

	public String getEndDate() {

	return endDate;

	}

	@Override
	public String toString() {

	return "exec dbo.PROC_AddPrice '" + conferenceID +"', '" +
valuePerDay +"', '" + startdate + "', '" + endDate + "'";

	}

}



public class Price {

	private int priceID;

	private int conferenceID;

	private double valuePerDay;

	private String startdate;

	private String endDate;


	public Price(int priceID, int conferenceID, double valuePerDay, String
startdate, String endDate) {

	this.priceID = priceID;

	this.conferenceID = conferenceID;

	this.valuePerDay = valuePerDay;

	this.startdate = startdate;

	this.endDate = endDate;

	}
```

```
	public int getPriceID() {
	return priceID;
	}

	public int getConferenceID() {
	return conferenceID;
	}

	public double getValuePerDay() {
	return valuePerDay;
	}

	public String getStartdate() {
	return startdate;
	}

	public String getEndDate() {
	return endDate;
	}

	@Override
	public String toString() {
	return "exec dbo.PROC_AddPrice '" + conferenceID +"', '" +
valuePerDay +"', '" + startdate + "', '" + endDate + "'";
	}
```

```
}
```

10.3.11 Klasa Workshop

```
public class Workshop {

	private int workshopID;
	private int dayOfConferenceID;
	private String subject;
	private String startTime;
	private String endTime;
	private int peopleLimit;
	private Double money;
	private int currentOccupancy;



	public Workshop(int workshopID, int dayOfConferenceID, String subject,
String startTime, String endTime, int peopleLimit, Double money) {

	this.workshopID = workshopID;
	this.dayOfConferenceID = dayOfConferenceID;
	this.subject = subject;
	this.startTime = startTime;
	this.endTime = endTime;
	this.peopleLimit = peopleLimit;
	this.money = money;
	this.currentOccupancy = 0;

	}



	public int getPeopleLimit() {

	return peopleLimit;

	}



	public int getCurrentOccupancy() {
	return currentOccupancy;
	}



	public void setCurrentOccupancy(int currentOccupancy) {
	this.currentOccupancy = currentOccupancy;
	}
```

```
        public int getWorkshopID() {
        return workshopID;
        }



        public int getDayOfConferenceID() {
        return dayOfConferenceID;
        }



        @Override
        public String toString() {
        return "exec dbo.PROC_AddWorkshop '" + dayOfConferenceID + "', '" +
subject + "', '" + startTime + "', '" +
                    endTime + "', '" + peopleLimit + "', '" + money + "'";
        }
}
```

10.3.12 Klasa WorkshopBooking

```
public class WorkshopBooking {
   private int workshopBookingID;
   private int conferenceDayBookingID;
   private int workshopID;
   private int numberOfParticipnats;

   public WorkshopBooking(int workshopBookingID, int conferenceDayBookingID,
int workshopID, int numberOfParticipnats) {
        this.workshopBookingID = workshopBookingID;
        this.conferenceDayBookingID = conferenceDayBookingID;
        this.workshopID = workshopID;
        this.numberOfParticipnats = numberOfParticipnats;
   }

   public int getWorkshopBookingID() {
        return workshopBookingID;
   }

   public int getConferenceDayBookingID() {
        return conferenceDayBookingID;
   }

   public int getWorkshopID() {
        return workshopID;
   }
```

```
    public int getNumberOfParticipnats() {
        return numberOfParticipnats;
    }

    @Override
    public String toString() {
        return "exec [dbo].[PROC_AddWorkshopBooking] '" +
conferenceDayBookingID +
                "', '" + workshopID + "', '" + numberOfParticipnats + "'";
    }

}
```

10.3.13 Klasa WorkshopReservation

```
public class WorkshopReservation {
    private int workshopReservationID;
    private  int dayReservationID;
    private int workshopBookingID;

    public WorkshopReservation(int workshopReservationID, int
dayReservationID, int workshopBookingID) {
        this.workshopReservationID = workshopReservationID;
        this.dayReservationID = dayReservationID;
        this.workshopBookingID = workshopBookingID;
    }

    @Override
    public String toString() {
        return "exec [dbo].[PROC_AddWorkshopReservation] '" +
dayReservationID + "', '" + workshopBookingID + "'";
    }
}
```

Przed wprowadzeniem poleceń wygenerowanych przez generator należy usunąć poprzednie rekordy i zresetować licznik Identity dla każdej tabeli, w której wartość klucza głównego jest autoikrementowana.

```
DELETE from IndividualClients
DELETE from Companies
DBCC CHECKIDENT('WorkshopReservations', RESEED, 0)
delete from DayReservations
DBCC CHECKIDENT('DayReservations', RESEED, 0)
Delete from WorkshopBooking
DBCC CHECKIDENT('WorkshopBooking', RESEED, 0)
```

```
Delete from Payments
DBCC CHECKIDENT('Payments', RESEED, 0)
delete from ConferenceDayBooking
DBCC CHECKIDENT('ConferenceDayBooking', RESEED, 0)
Delete from ConferenceBooking
DBCC CHECKIDENT('ConferenceBooking', RESEED, 0)
DELETE FROM Prices
DBCC CHECKIDENT('Prices', RESEED, 0)
DELETE from Workshops
DBCC CHECKIDENT('Workshops', RESEED, 0)
DELETE from ConferenceDays
DBCC CHECKIDENT('ConferenceDays', RESEED, 0)
DELETE FROM Conferences
DBCC CHECKIDENT('Conferences', RESEED, 0)
DELETE from Clients
DBCC CHECKIDENT('Clients', RESEED, 0)
DELETE from Participants
DBCC CHECKIDENT('Participants', RESEED, 0)
```