# Spis treści

[Wprowadzenie 4](#_Toc535005459)

[Cel 4](#_Toc535005460)

[Założenia 4](#_Toc535005461)

[Implementacja 4](#_Toc535005462)

[Wnioski 4](#_Toc535005463)

[Schemat bazy 5](#_Toc535005464)

[Tabele 6](#_Toc535005465)

[Conferences 6](#_Toc535005466)

[ConferenceDays 7](#_Toc535005467)

[ConferenceCosts 8](#_Toc535005468)

[Workshops 9](#_Toc535005469)

[ConferenceBooking 10](#_Toc535005470)

[ConferenceDayBooking 11](#_Toc535005471)

[WorkshopBooking 12](#_Toc535005472)

[Payments 13](#_Toc535005473)

[Participants 14](#_Toc535005474)

[DayParticipants 15](#_Toc535005475)

[WorkshopParticipants 16](#_Toc535005476)

[Clients 17](#_Toc535005477)

[Indeksy 18](#_Toc535005478)

[Widoki 19](#_Toc535005479)

[MostPopularWorkshops 19](#_Toc535005480)

[MostPopularConferences 20](#_Toc535005481)

[MostPopularConferencesByStudents 21](#_Toc535005482)

[MostPopularWorkshopsByStudents 22](#_Toc535005483)

[MostProfitableConference 23](#_Toc535005484)

[MostProfitableWorkshops 24](#_Toc535005485)

[WorkshopsFreePlaces 25](#_Toc535005486)

[ConferenceFreePlaces 26](#_Toc535005487)

[AvailableConferenceDays 27](#_Toc535005488)

[AvailableWorkshops 28](#_Toc535005489)

[ClientsActivity 29](#_Toc535005490)

[MostProfitableClients 30](#_Toc535005491)

[Procedury 31](#_Toc535005492)

[Dodające 31](#_Toc535005493)

[AddConferenceDay 31](#_Toc535005494)

[AddWorkshop 32](#_Toc535005495)

[AddConferenceCost 33](#_Toc535005496)

[AddConferenceBooking 34](#_Toc535005497)

[AddConferenceDayBooking 35](#_Toc535005498)

[AddWorkshopBooking 36](#_Toc535005499)

[AddDayParticipant 37](#_Toc535005500)

[AddWorkshopParticipant 38](#_Toc535005501)

[AddParticipant 39](#_Toc535005502)

[AddClient 40](#_Toc535005503)

[Aktualizujące 41](#_Toc535005504)

[RemoveConference 41](#_Toc535005505)

[UpdateConferenceDetails 42](#_Toc535005506)

[UpdateWorkshopDetails 44](#_Toc535005507)

[CancelConferenceBooking 46](#_Toc535005508)

[CancelConferenceDayBooking 47](#_Toc535005509)

[UpdateWorkShopNumberOfParticipants 48](#_Toc535005510)

[UpdateConferenceDayNumberOfParticipants 49](#_Toc535005511)

[Wyświetlające 50](#_Toc535005512)

[ShowConferenceDaysAmountOfParticipants 50](#_Toc535005513)

[ShowListOfEventsOfConference 51](#_Toc535005514)

[Funkcje 52](#_Toc535005515)

[FreeDayPlacesForParticipants 52](#_Toc535005516)

[FreeDayPlacesForStudents 53](#_Toc535005517)

[FreeDayPlaces 54](#_Toc535005518)

[FreeWorkshopPlaces 55](#_Toc535005519)

[DaysOfConference 56](#_Toc535005520)

[ConferenceDayParticipants 57](#_Toc535005521)

[WorkshopsPerConference 58](#_Toc535005522)

[WorkshopDate 59](#_Toc535005523)

[BookingDaysCost 60](#_Toc535005524)

[BookingWorkshopCost 61](#_Toc535005525)

[TotalBookingCost 62](#_Toc535005526)

[WorkshopListForParticipant 63](#_Toc535005527)

[ConferencesDaysListForParticipant 64](#_Toc535005528)

[ClientsOrdersList 65](#_Toc535005529)

[Triggery 66](#_Toc535005530)

[TooFewFreePlacesForDayBooking 66](#_Toc535005531)

[TooFewFreePlacesForWorkshopBooking 67](#_Toc535005532)

[LessPlacesForDayThanForWorkshop 68](#_Toc535005533)

[NotEnoughBookedPlacesForDay 69](#_Toc535005534)

[NotEnoughBookedPlacesForWorkshop 70](#_Toc535005535)

[TooFewPlacesAfterDecreasingDayCapacity 71](#_Toc535005536)

[TooFewPlacesAfterDecreasingWorkshopCapacity 72](#_Toc535005537)

[BookingDayInDifferentConference 73](#_Toc535005538)

[BookingDayAlreadyExists 74](#_Toc535005539)

[BookingWorkshopInDifferentDay 75](#_Toc535005540)

[ArePriceThresholdsMonotonous 76](#_Toc535005541)

# Wprowadzenie

## Cel

…

## Założenia

..

## Implementacja

..

## Wnioski

…

# Schemat bazy

# Tabele

## Conferences

**CREATE TABLE Conferences (**

**ConferenceID INT NOT NULL IDENTITY,**

**Name VARCHAR(100) NOT NULL,**

**Place VARCHAR(100) NOT NULL,**

**DiscountForStudents FLOAT NOT NULL DEFAULT 0,**

**Description VARCHAR(200) NOT NULL,**

**CONSTRAINT ProperDiscountForStudents CHECK (**

**DiscountForStudents >= 0**

**AND DiscountForStudents <= 100**

**),**

**CONSTRAINT ProperDescription CHECK (LEN(Description) > 5),**

**CONSTRAINT Conferences\_pk PRIMARY KEY (ConferenceID)**

**);**

## ConferenceDays

**CREATE TABLE ConferenceDays (**

**ConferenceDayID INT NOT NULL IDENTITY,**

**Conferences\_ConferenceID INT NOT NULL,**

**DATE DATE NOT NULL,**

**NumberOfParticipants INT NOT NULL,**

**CONSTRAINT PositiveNumberOfConferenceParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT ConferenceDays\_pk PRIMARY KEY (ConferenceDayID)**

**);**

**-- Reference: Conferences\_ConferenceDays (table: ConferenceDays)**

**ALTER TABLE ConferenceDays ADD CONSTRAINT Conferences\_ConferenceDays FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

## ConferenceCosts

**CREATE TABLE ConferenceCosts (**

**ConferenceCostID INT NOT NULL IDENTITY,**

**Conferences\_ConferenceID INT NOT NULL,**

**Cost DECIMAL(9, 2) NOT NULL,**

**DateFrom DATE NOT NULL,**

**DateTo DATE NOT NULL,**

**CONSTRAINT ProperDayDifferance CHECK (DateFrom <= DateTo),**

**CONSTRAINT NonnegativeCostConf CHECK (Cost >= 0),**

**CONSTRAINT ConferenceCosts\_pk PRIMARY KEY (ConferenceCostID)**

**);**

**-- Reference: Conferences\_ConferenceDay (table: ConferenceCosts)**

**ALTER TABLE ConferenceCosts ADD CONSTRAINT Conferences\_ConferenceDay FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

## Workshops

**CREATE TABLE Workshops (**

**WorkshopID INT NOT NULL IDENTITY,**

**ConferenceDays\_ConferenceDayID INT NOT NULL,**

**Name VARCHAR(100) NOT NULL,**

**StartTime DATE NOT NULL,**

**EndTime DATE NOT NULL,**

**Cost DECIMAL(9, 2) NOT NULL DEFAULT 0,**

**NumberOfParticipants INT NOT NULL DEFAULT 10,**

**CONSTRAINT NonnegativeCostWorkshop CHECK (Cost >= 0),**

**CONSTRAINT ProperTimeDifferance CHECK (StartTime <= EndTime),**

**CONSTRAINT PosiviteWorkshopParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT Workshops\_pk PRIMARY KEY (WorkshopID)**

**);**

**-- Reference: Workshops\_ConferenceDays (table: Workshops)**

**ALTER TABLE Workshops ADD CONSTRAINT Workshops\_ConferenceDays FOREIGN KEY (ConferenceDays\_ConferenceDayID) REFERENCES ConferenceDays (ConferenceDayID)**

## ConferenceBooking

**CREATE TABLE ConferenceBooking (**

**ConferenceBookingID INT NOT NULL IDENTITY,**

**Clients\_ClientID INT NOT NULL,**

**Conferences\_ConferenceID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**IsCanceled BIT NOT NULL DEFAULT 0,**

**CONSTRAINT ConferenceBooking\_pk PRIMARY KEY (ConferenceBookingID)**

**);**

**-- Reference: BookingConference\_Clients (table: ConferenceBooking)**

**ALTER TABLE ConferenceBooking ADD CONSTRAINT BookingConference\_Clients FOREIGN KEY (Clients\_ClientID) REFERENCES Clients (ClientID)**

**-- Reference: BookingConference\_Conferences (table: ConferenceBooking)**

**ALTER TABLE ConferenceBooking ADD CONSTRAINT BookingConference\_Conferences FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

## ConferenceDayBooking

**CREATE TABLE ConferenceDayBooking (**

**ConferenceDayBookingID INT NOT NULL IDENTITY,**

**ConferenceBooking\_ConferenceBookingID INT NOT NULL,**

**ConferenceDays\_ConferenceDayID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**IsCancelled BIT NOT NULL DEFAULT 0,**

**NumberOfParticipants INT NOT NULL,**

**NumberOfStudents INT NOT NULL DEFAULT 0,**

**CONSTRAINT PositiveNumberOfParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT NonnegativeNumberOfStudents CHECK (NumberOfStudents >= 0),**

**CONSTRAINT ProperNumberOfStudents CHECK (NumberOfParticipants >= NumberOfStudents),**

**CONSTRAINT ConferenceDayBooking\_pk PRIMARY KEY (ConferenceDayBookingID)**

**);**

**-- Reference: ConferenceDayBooking\_ConferenceBooking (table: ConferenceDayBooking)**

**ALTER TABLE ConferenceDayBooking ADD CONSTRAINT ConferenceDayBooking\_ConferenceBooking FOREIGN KEY (ConferenceBooking\_ConferenceBookingID) REFERENCES ConferenceBooking (ConferenceBookingID)**

**-- Reference: ConferenceDayBooking\_ConferenceDays (table: ConferenceDayBooking)**

**ALTER TABLE ConferenceDayBooking ADD CONSTRAINT ConferenceDayBooking\_ConferenceDays FOREIGN KEY (ConferenceDays\_ConferenceDayID) REFERENCES ConferenceDays (ConferenceDayID)**

## WorkshopBooking

**CREATE TABLE WorkshopBooking (**

**WorkshopBookingID INT NOT NULL IDENTITY,**

**Workshops\_WorkshopID INT NOT NULL,**

**ConferenceDayBooking\_ConferenceDayBookingID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**NumberOfParticipants INT NOT NULL,**

**CONSTRAINT PositiveWorkshopNumberOfParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT WorkshopBooking\_pk PRIMARY KEY (WorkshopBookingID)**

**);**

**-- Reference: WorkshopBooking\_ConferenceDayBooking (table: WorkshopBooking)**

**ALTER TABLE WorkshopBooking ADD CONSTRAINT WorkshopBooking\_ConferenceDayBooking FOREIGN KEY (ConferenceDayBooking\_ConferenceDayBookingID) REFERENCES ConferenceDayBooking (ConferenceDayBookingID)**

**-- Reference: WorkshopBooking\_Workshops (table: WorkshopBooking)**

**ALTER TABLE WorkshopBooking ADD CONSTRAINT WorkshopBooking\_Workshops FOREIGN KEY (Workshops\_WorkshopID) REFERENCES Workshops (WorkshopID)**

## Payments

**CREATE TABLE Payments (**

**PaymentID INT NOT NULL IDENTITY,**

**ConferenceBooking\_ConferenceBookingID INT NOT NULL,**

**Amount DECIMAL(9, 2) NOT NULL,**

**PayDate DATE NOT NULL DEFAULT getdate(),**

**CONSTRAINT PositiveValue CHECK (Amount > 0),**

**CONSTRAINT Payments\_pk PRIMARY KEY (PaymentID)**

**);**

**-- Reference: Payments\_ConferenceBooking (table: Payments)**

**ALTER TABLE Payments ADD CONSTRAINT Payments\_ConferenceBooking FOREIGN KEY (ConferenceBooking\_ConferenceBookingID) REFERENCES ConferenceBooking (ConferenceBookingID)**

## Participants

**CREATE TABLE Participants (**

**ParticipantID INT NOT NULL IDENTITY,**

**FirstName VARCHAR(40) NOT NULL,**

**LastName VARCHAR(40) NOT NULL,**

**Email VARCHAR(100) NOT NULL,**

**Street VARCHAR(40) NOT NULL,**

**City VARCHAR(40) NOT NULL,**

**PostalCode VARCHAR(10) NOT NULL,**

**County VARCHAR(40) NOT NULL,**

**CONSTRAINT Participants\_pk PRIMARY KEY (ParticipantID)**

**);**

## DayParticipants

**CREATE TABLE DayParticipants (**

**DayParticipantID INT NOT NULL IDENTITY,**

**ConferenceDayBooking\_ConferenceDayBookingID INT NOT NULL,**

**Participants\_ParticipantID INT NOT NULL,**

**StudentID VARCHAR(6) NULL DEFAULT NULL,**

**CONSTRAINT DayParticipants\_pk PRIMARY KEY (DayParticipantID)**

**);**

**-- Reference: DayParticipants\_ConferenceDayBooking (table: DayParticipants)**

**ALTER TABLE DayParticipants ADD CONSTRAINT DayParticipants\_ConferenceDayBooking FOREIGN KEY (ConferenceDayBooking\_ConferenceDayBookingID) REFERENCES ConferenceDayBooking (ConferenceDayBookingID)**

**-- Reference: Participants\_DayParticipants (table: DayParticipants)**

**ALTER TABLE DayParticipants ADD CONSTRAINT Participants\_DayParticipants FOREIGN KEY (Participants\_ParticipantID) REFERENCES Participants (ParticipantID)**

## WorkshopParticipants

**CREATE TABLE WorkshopParticipants (**

**WorkshopParticipantID INT NOT NULL IDENTITY,**

**WorkshopBooking\_WorkshopBookingID INT NOT NULL,**

**DayParticipants\_DayParticipantID INT NOT NULL,**

**CONSTRAINT WorkshopParticipants\_pk PRIMARY KEY (WorkshopParticipantID)**

**);**

**-- Reference: DayParticipants\_WorkshopParticipants (table: WorkshopParticipants)**

**ALTER TABLE WorkshopParticipants ADD CONSTRAINT DayParticipants\_WorkshopParticipants FOREIGN KEY (DayParticipants\_DayParticipantID) REFERENCES DayParticipants (DayParticipantID)**

**-- Reference: WorkshopParticipants\_WorkshopBooking (table: WorkshopParticipants)**

**ALTER TABLE WorkshopParticipants ADD CONSTRAINT WorkshopParticipants\_WorkshopBooking FOREIGN KEY (WorkshopBooking\_WorkshopBookingID) REFERENCES WorkshopBooking (WorkshopBookingID)**

## Clients

**CREATE TABLE Clients (**

**ClientID INT NOT NULL IDENTITY,**

**IsCompany BIT NOT NULL,**

**Name VARCHAR(100) NOT NULL,**

**Surname VARCHAR(100) NULL,**

**Email VARCHAR(100) NOT NULL,**

**CONSTRAINT Clients\_pk PRIMARY KEY (ClientID)**

**);**

# Indeksy

**CREATE INDEX Client ON Clients (ClientID ASC)**

**CREATE INDEX Client\_ConferenceBooking ON ConferenceBooking (Clients\_ClientID ASC)**

**CREATE INDEX ConferenceBook ON ConferenceDayBooking (ConferenceDayBookingID ASC)**

**CREATE INDEX ConferenceDay ON ConferenceDayBooking (ConferenceDays\_ConferenceDayID ASC)**

**CREATE INDEX NumberOfParticipantsInConferenceDay ON ConferenceDays (NumberOfParticipants ASC)**

**CREATE INDEX Conference ON ConferenceDays (Conferences\_ConferenceID ASC)**

**CREATE INDEX Participant ON DayParticipants (DayParticipantID ASC)**

**CREATE INDEX ConferenceDayBook ON WorkshopBooking (ConferenceDayBooking\_ConferenceDayBookingID ASC)**

**CREATE INDEX DayParticipant ON WorkshopParticipants (DayParticipants\_DayParticipantID ASC)**

**CREATE INDEX NumberOfParticipantsInWorkshop ON Workshops (NumberOfParticipants ASC)**

**CREATE INDEX ConferenceDayID ON Workshops (ConferenceDays\_ConferenceDayID ASC)**

# Widoki

## MostPopularWorkshops

**CREATE VIEW view\_MostPopularWorkshops**

**AS**

**SELECT TOP 10 conf.name AS ConferenceName,**

**w.workshopid,**

**w.name,**

**isnull(sum(wp.workshopparticipantid), 0) AS popularnosc**

**FROM Workshops AS w**

**INNER JOIN conferencedays AS cd ON cd.conferencedayid = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN conferences AS conf ON conf.conferenceid = cd.Conferences\_ConferenceID**

**INNER JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**INNER JOIN WorkshopParticipants AS wp ON wp.WorkshopBooking\_WorkshopBookingID = wb.workshopbookingid**

**GROUP BY w.workshopid,**

**w.name,**

**conf.name**

**ORDER BY popularnosc DESC**

## MostPopularConferences

**CREATE VIEW view\_MostPopularConferences**

**AS**

**SELECT TOP 10 c.ConferenceID,**

**c.Name,**

**isnull(sum(dp.DayParticipantID), 0) AS popularity**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**INNER JOIN DayParticipants AS dp ON dp.ConferenceDayBooking\_ConferenceDayBookingID = cdb.ConferenceDayBookingID**

**GROUP BY c.ConferenceID,**

**c.Name**

**ORDER BY popularity DESC**

## MostPopularConferencesByStudents

**CREATE VIEW view\_MostPopularConferencesByStudents**

**AS**

**SELECT TOP 10 c.ConferenceID,**

**c.Name,**

**isnull(sum(dp.DayParticipantID), 0) AS popularity**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**INNER JOIN DayParticipants AS dp ON dp.ConferenceDayBooking\_ConferenceDayBookingID = cdb.ConferenceDayBookingID**

**AND dp.StudentID IS NOT NULL**

**GROUP BY c.ConferenceID,**

**c.Name**

**ORDER BY popularity DESC**

## MostPopularWorkshopsByStudents

**CREATE VIEW view\_MostPopularWorkshopsByStudents**

**AS**

**SELECT TOP 10 conf.name AS ConferenceName,**

**w.workshopid,**

**w.name,**

**isnull(sum(wp.workshopparticipantid), 0) AS popularnosc**

**FROM Workshops AS w**

**INNER JOIN conferencedays AS cd ON cd.conferencedayid = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN conferences AS conf ON conf.conferenceid = cd.Conferences\_ConferenceID**

**INNER JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**INNER JOIN WorkshopParticipants AS wp ON wp.WorkshopBooking\_WorkshopBookingID = wb.workshopbookingid**

**INNER JOIN DayParticipants AS dp ON dp.DayParticipantID = wp.DayParticipants\_DayParticipantID**

**AND dp.StudentID IS NOT NULL**

**GROUP BY w.workshopid,**

**w.name,**

**conf.name**

**ORDER BY popularnosc DESC**

## MostProfitableConference

**CREATE VIEW view\_MostProfitableConference**

**AS**

**SELECT TOP 10 c.ConferenceID,**

**c.Name,**

**cc.Cost + a.koszt AS cena**

**FROM Conferences AS c**

**INNER JOIN ConferenceCosts AS cc ON cc.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN (**

**SELECT con.ConferenceID,**

**(**

**SELECT isnull(sum(w.Cost), 0) AS koszta**

**FROM conferencedays AS cd**

**INNER JOIN Workshops AS w ON w.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**WHERE cd.Conferences\_ConferenceID = con.ConferenceID**

**) AS koszt**

**FROM Conferences AS con**

**) AS a ON c.ConferenceID = a.ConferenceID**

**ORDER BY cena**

## MostProfitableWorkshops

**CREATE VIEW view\_MostProfitableWorkshops**

**AS**

**SELECT TOP 10 c.name AS NazwaKonferencji,**

**w.name,**

**w.cost**

**FROM Workshops AS w**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c ON c.ConferenceID = cd.Conferences\_ConferenceID**

**ORDER BY w.cost DESC**

## WorkshopsFreePlaces

**CREATE VIEW view\_WorkshopsFreePlaces**

**AS**

**SELECT c.name AS ConferenceName,**

**w.workshopid,**

**w.name,**

**w.numberofparticipants AS Miejsca,**

**isnull(SUM(wp.WorkshopParticipantID), 0) AS Zajete,**

**w.numberofparticipants - isnull(SUM(wb.NumberOfParticipants), 0) AS WolneMiejsca**

**FROM workshops AS w**

**INNER JOIN WorkshopBooking AS wb ON wb.workshops\_workshopid = w.workshopid**

**INNER JOIN WorkshopParticipants AS wp ON wp.WorkshopBooking\_WorkshopBookingID = wb.WorkshopBookingID**

**INNER JOIN conferencedays AS cd ON cd.conferencedayid = w.conferencedays\_conferencedayid**

**INNER JOIN conferences AS c ON c.conferenceid = cd.conferences\_conferenceid**

**GROUP BY w.workshopid,**

**w.name,**

**w.numberofparticipants,**

**c.name**

## ConferenceFreePlaces

**CREATE VIEW view\_ConferenceFreePlaces**

**AS**

**SELECT c.name,**

**cd.DATE,**

**isnull(sum(dp.DayParticipantID), 0) AS Zajete,**

**cdb.NumberOfParticipants AS Miejsca,**

**cdb.NumberOfParticipants - isnull(sum(dp.DayParticipantID), 0) AS WolneMiejsca**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**INNER JOIN DayParticipants AS dp ON dp.ConferenceDayBooking\_ConferenceDayBookingID = cdb.ConferenceDayBookingID**

**GROUP BY c.Name,**

**cd.DATE,**

**cdb.NumberOfParticipants**

## AvailableConferenceDays

**CREATE VIEW view\_AvailableConferenceDays**

**AS**

**SELECT cdp.\***

**FROM VIEW\_ConferenceFreePlaces cdp**

**WHERE WolneMiejsca > 0**

## AvailableWorkshops

**CREATE VIEW view\_AvailableWorkshops**

**AS**

**SELECT w.\***

**FROM view\_WorkshopsFreePlaces w**

**WHERE WolneMiejsca > 0**

## ClientsActivity

**CREATE VIEW view\_ClientsActivity**

**AS**

**SELECT c.\*,**

**isnull((**

**SELECT COUNT(\*)**

**FROM conferencebooking AS cb**

**WHERE cb.clients\_clientid = c.ClientID**

**), 0) AS Bookings,**

**isnull((**

**SELECT SUM(p.amount)**

**FROM conferencebooking AS cb**

**INNER JOIN payments AS p ON cb.ConferenceBookingID = p.ConferenceBooking\_ConferenceBookingID**

**WHERE cb.clients\_clientid = c.ClientID**

**), 0) AS Payments**

**FROM Clients AS c**

## MostProfitableClients

**CREATE VIEW view\_MostProfitableClients**

**AS**

**SELECT TOP 10 c.\***

**FROM view\_ClientsActivity c**

**ORDER BY Payments**

# Procedury

## Dodające

### AddConferenceDay

**CREATE PROCEDURE PROCEDURE\_AddConferenceDay (**

**@ConferenceID INT,**

**@Date DATE,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO ConferenceDays (**

**Conferences\_ConferenceID,**

**DATE,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceID,**

**@Date,**

**@NumberOfParticipants**

**);**

**END**

### AddWorkshop

**CREATE PROCEDURE PROCEDURE\_AddWorkshop (**

**@ConferenceDayID INT,**

**@Name VARCHAR(100),**

**@StartTime DATE,**

**@EndTime DATE,**

**@Cost DECIMAL(9, 2),**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO Workshops (**

**ConferenceDays\_ConferenceDayID,**

**Name,**

**StartTime,**

**EndTime,**

**Cost,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceDayID,**

**@Name,**

**@StartTime,**

**@EndTime,**

**@Cost,**

**@NumberOfParticipants**

**);**

**END**

### AddConferenceCost

**CREATE PROCEDURE PROCEDURE\_AddConferenceCost (**

**@ConferenceID INT,**

**@Cost DECIMAL(9, 2),**

**@DateFrom DATE,**

**@DateTo DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO ConferenceCosts (**

**Conferences\_ConferenceID,**

**Cost,**

**DateFrom,**

**DateTo**

**)**

**VALUES (**

**@ConferenceID,**

**@Cost,**

**@DateFrom,**

**@DateTo**

**);**

**END**

### AddConferenceBooking

**CREATE PROCEDURE PROCEDURE\_AddConferenceBooking (**

**@ConferenceID INT,**

**@ClientID INT,**

**@BookingDate DATE**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**IF @BookingDate IS NULL**

**BEGIN**

**INSERT INTO ConferenceBooking (**

**Conferences\_ConferenceID,**

**Clients\_ClientID**

**)**

**VALUES (**

**@ConferenceID,**

**@ClientID**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO ConferenceBooking (**

**Conferences\_ConferenceID,**

**BookingDate,**

**Clients\_ClientID**

**)**

**VALUES (**

**@ConferenceID,**

**@BookingDate,**

**@ClientID**

**);**

**END**

**END**

**END**

### AddConferenceDayBooking

**CREATE PROCEDURE PROCEDURE\_AddConferenceDayBooking (**

**@ConferenceDayID INT,**

**@ConferenceBookingID INT,**

**@NumberOfParticipants INT,**

**@NumberOfStudents INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**OR NOT EXISTS (**

**SELECT \***

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID lub ConferenceBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF @NumberOfStudents IS NULL**

**BEGIN**

**INSERT INTO ConferenceDayBooking (**

**ConferenceDays\_ConferenceDayID,**

**ConferenceBooking\_ConferenceBookingID,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceDayID,**

**@ConferenceBookingID,**

**@NumberOfParticipants**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO ConferenceDayBooking (**

**ConferenceDays\_ConferenceDayID,**

**ConferenceBooking\_ConferenceBookingID,**

**NumberOfParticipants,**

**NumberOfStudents**

**)**

**VALUES (**

**@ConferenceDayID,**

**@ConferenceBookingID,**

**@NumberOfParticipants,**

**@NumberOfStudents**

**);**

**END**

**END**

**END**

### AddWorkshopBooking

**CREATE PROCEDURE PROCEDURE\_AddWorkshopBooking (**

**@WorkshopID INT,**

**@ConferenceDayBookingID INT,**

**@BookingDate DATE,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**OR NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopID lub ConferenceDayBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF @BookingDate IS NULL**

**BEGIN**

**INSERT INTO WorkshopBooking (**

**Workshops\_WorkshopID,**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**NumberOfParticipants**

**)**

**VALUES (**

**@WorkshopID,**

**@ConferenceDayBookingID,**

**@NumberOfParticipants**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO WorkshopBooking (**

**Workshops\_WorkshopID,**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**BookingDate,**

**NumberOfParticipants**

**)**

**VALUES (**

**@WorkshopID,**

**@ConferenceDayBookingID,**

**@BookingDate,**

**@NumberOfParticipants**

**);**

**END**

**END**

**END**

### AddDayParticipant

**CREATE PROCEDURE PROCEDURE\_AddDayParticipant (**

**@ConferenceDayBookingID INT,**

**@ParticipantID INT,**

**@StudentID VARCHAR(6)**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Participants**

**WHERE @ParticipantID = ParticipantID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ParticipantID', 1**

**END**

**ELSE**

**BEGIN**

**INSERT INTO DayParticipants (**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**Participants\_ParticipantID,**

**StudentID**

**)**

**VALUES (**

**@ConferenceDayBookingID,**

**@ParticipantID,**

**@StudentID**

**);**

**END**

**END**

**END**

### AddWorkshopParticipant

**CREATE PROCEDURE PROCEDURE\_AddWorkshopParticipant (**

**@WorkshopBookingID INT,**

**@DayParticipantID INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM WorkshopBooking**

**WHERE @WorkshopBookingID = WorkshopBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Participants**

**WHERE @DayParticipantID = ParticipantID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ParticipantID', 1**

**END**

**ELSE**

**BEGIN**

**INSERT INTO WorkshopParticipants (**

**WorkshopBooking\_WorkshopBookingID,**

**DayParticipants\_DayParticipantID**

**)**

**VALUES (**

**@WorkshopBookingID,**

**@DayParticipantID**

**);**

**END**

**END**

**END**

### AddParticipant

**CREATE PROCEDURE PROCEDURE\_AddParticipant (**

**@FirstName VARCHAR(40),**

**@LastName VARCHAR(40),**

**@Email VARCHAR(100),**

**@Street VARCHAR(40),**

**@City VARCHAR(40),**

**@PostalCode VARCHAR(10),**

**@Country VARCHAR(40)**

**)**

**AS**

**BEGIN**

**INSERT INTO Participants (**

**FirstName,**

**LastName,**

**Email,**

**Street,**

**City,**

**PostalCode,**

**County**

**)**

**VALUES (**

**@FirstName,**

**@LastName,**

**@Email,**

**@Street,**

**@City,**

**@PostalCode,**

**@Country**

**);**

**END**

### AddClient

**CREATE PROCEDURE PROCEDURE\_AddClient (**

**@IsCompany BIT,**

**@Name VARCHAR(100),**

**@Surname VARCHAR(100),**

**@Email VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO Clients (**

**IsCompany,**

**Name,**

**Surname,**

**Email**

**)**

**VALUES (**

**@IsCompany,**

**@Name,**

**@Surname,**

**@Email**

**);**

**END**

## Aktualizujące

### RemoveConference

**CREATE PROCEDURE PROCEDURE\_RemoveConference (@ConferenceID INT)**

**AS**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**DELETE**

**FROM Conferences**

**WHERE Conferences.ConferenceID = @ConferenceID;**

**END**

### UpdateConferenceDetails

**CREATE PROCEDURE PROCEDURE\_UpdateConferenceDetails (**

**@ConferenceID INT,**

**@Name VARCHAR(100),**

**@Place VARCHAR(100),**

**@DiscountForStudents FLOAT,**

**@Description VARCHAR(200)**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**IF @Name IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Name = @Name**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'ConferenceID is null', 1**

**END**

**IF @DiscountForStudents IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET DiscountForStudents = @DiscountForStudents**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'DiscountForStudents is null', 1**

**END**

**IF @Place IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Place = @Place**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Place is null', 1**

**END**

**IF @Description IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Description = @Description**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Description is null', 1**

**END**

**END**

**END**

### UpdateWorkshopDetails

**CREATE PROCEDURE PROCEDURE\_UpdateWorkshopDetails (**

**@WorkshopID INT,**

**@Name VARCHAR(100),**

**@StartTime DATE,**

**@EndTime DATE,**

**@Cost DECIMAL(9, 2),**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono warsztatu', 1**

**END**

**ELSE**

**BEGIN**

**IF @Name IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET Name = @Name**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'WorkshopID is null', 1**

**END**

**IF @StartTime IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET StartTime = @StartTime**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'StartTime is null', 1**

**END**

**IF @EndTime IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET EndTime = @EndTime**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'EndTime is null', 1**

**END**

**IF @Cost IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET Cost = @Cost**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Cost is null', 1**

**END**

**IF @NumberOfParticipants IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'NumberOfParticipants is null', 1**

**END**

**END**

**END**

### CancelConferenceBooking

**CREATE PROCEDURE PROCEDURE\_CancelConferenceBooking (@ConferenceBookingID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**)**

**OR NOT (**

**SELECT IsCanceled**

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**) = 0**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceBookingID lub rezerwacja została już wcześniej anulowana', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceBooking**

**SET IsCanceled = 1**

**WHERE @ConferenceBookingID = ConferenceBookingID;**

**END**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**WHERE @ConferenceBookingID = ConferenceBooking\_ConferenceBookingID;**

**END**

**END**

### CancelConferenceDayBooking

**CREATE PROCEDURE PROCEDURE\_CancelConferenceDayBooking (@ConferenceDayBookingID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**OR NOT (**

**SELECT IsCancelled**

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**) = 0**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayBookingID lub rezerwacja została już wcześniej anulowana', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID;**

**END**

**END**

### UpdateWorkShopNumberOfParticipants

**CREATE PROCEDURE PROCEDURE\_UpdateWorkShopNumberOfParticipants (**

**@WorkshopID INT,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopID', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE Workshops**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE @WorkshopID = WorkshopID;**

**END**

**END**

### UpdateConferenceDayNumberOfParticipants

**CREATE PROCEDURE PROCEDURE\_UpdateConferenceDayNumberOfParticipants (**

**@ConferenceDayID INT,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceDays**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE @ConferenceDayID = ConferenceDayID;**

**END**

**END**

### CancelConferenceBookingWithoutPayingAfterSevenDays

**CREATE PROCEDURE PROCEDURE\_CancelConferenceBookingWithoutPayingAfterSevenDays**

**AS**

**BEGIN**

**UPDATE ConferenceBooking**

**SET IsCanceled = 1**

**FROM (**

**SELECT \***

**FROM ConferenceBooking**

**LEFT JOIN Payments ON ConferenceBookingID = ConferenceBooking\_ConferenceBookingID**

**WHERE IsCanceled = 0**

**AND PaymentID IS NULL**

**AND DATEDIFF(DAY, BookingDate, getdate()) > 7**

**) AS a**

**WHERE ConferenceBooking.ConferenceBookingID = a.ConferenceBookingID**

**END**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**FROM (**

**SELECT ConferenceDayBookingID AS ID**

**FROM ConferenceBooking**

**INNER JOIN ConferenceDayBooking ON ConferenceBookingID = ConferenceBooking\_ConferenceBookingID**

**WHERE ConferenceBooking.IsCanceled = 1**

**AND ConferenceDayBooking.IsCancelled = 0**

**) AS b**

**WHERE ConferenceDayBookingID = b.ID**

**END**

### 

## Wyświetlające

### ShowConferenceDaysAmountOfParticipants

**CREATE PROCEDURE PROCEDURE\_ShowConferenceDaysAmountOfParticipants (@ConferenceID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceID', 1**

**END**

**ELSE**

**BEGIN**

**SELECT ConferenceDayID,**

**DATE,**

**SUM(ConferenceDayBooking.NumberOfParticipants) AS Participants,**

**SUM(NumberOfStudents) AS Students**

**FROM ConferenceDays**

**INNER JOIN ConferenceDayBooking ON Conferences\_ConferenceID = ConferenceDays\_ConferenceDayID**

**WHERE Conferences\_ConferenceID = @ConferenceID**

**AND IsCancelled = 0**

**GROUP BY ConferenceDayID,**

**DATE**

**END**

**END**

### ShowListOfEventsOfConference

**CREATE PROCEDURE PROCEDURE\_ShowListOfEventsOfConference (@ConferenceDayID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID', 1**

**END**

**ELSE**

**BEGIN**

**SELECT WorkshopID,**

**Name,**

**StartTime,**

**EndTime,**

**Cost,**

**NumberOfParticipants**

**FROM Workshops**

**WHERE ConferenceDays\_ConferenceDayID = @ConferenceDayID**

**END**

**END**

# Funkcje

## FreeDayPlacesForParticipants

**CREATE FUNCTION FUNCTION\_FreeDayPlacesForParticipants (@ConferenceDayID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT cdb.numberofparticipants - isnull(SUM(dp.DayParticipantID), 0)**

**FROM conferencedays AS cd**

**LEFT JOIN ConferenceDayBooking AS cdb ON cd.conferencedayid = cdb.conferencedays\_conferencedayid**

**LEFT JOIN DayParticipants AS dp ON dp.ConferenceDayBooking\_ConferenceDayBookingID = cdb.ConferenceDayBookingID**

**WHERE cd.conferencedayid = @ConferenceDayID**

**AND dp.StudentID IS NULL**

**GROUP BY cd.conferencedayid,**

**cdb.numberofparticipants**

**);**

**END**

## FreeDayPlacesForStudents

**CREATE FUNCTION FUNCTION\_FreeDayPlacesForStudents (@ConferenceDayID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT cdb.NumberOfStudents - isnull(SUM(dp.DayParticipantID), 0)**

**FROM conferencedays AS cd**

**LEFT JOIN ConferenceDayBooking AS cdb ON cd.conferencedayid = cdb.conferencedays\_conferencedayid**

**LEFT JOIN DayParticipants AS dp ON dp.ConferenceDayBooking\_ConferenceDayBookingID = cdb.ConferenceDayBookingID**

**WHERE cd.conferencedayid = @ConferenceDayID**

**AND dp.StudentID IS NOT NULL**

**GROUP BY cd.conferencedayid,**

**cdb.NumberOfStudents**

**);**

**END**

## FreeDayPlaces

**CREATE FUNCTION FUNCTION\_FreeDayPlaces (@ConferenceDayID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (dbo.FUNCTION\_FreeDayPlacesForParticipants(@ConferenceDayID) + dbo.FUNCTION\_FreeDayPlacesForStudents(@ConferenceDayID));**

**END**

## FreeWorkshopPlaces

**CREATE FUNCTION FUNCTION\_FreeWorkshopPlaces (@WorkShopID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT w.numberofparticipants - isnull(SUM(wp.workshopparticipantid), 0)**

**FROM Workshops AS w**

**LEFT JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**LEFT JOIN WorkshopParticipants AS wp ON wp.WorkshopBooking\_WorkshopBookingID = wb.WorkshopBookingID**

**WHERE w.WorkshopID = @WorkShopID**

**GROUP BY w.WorkshopID,**

**w.numberofparticipants**

**);**

**END**

## DaysOfConference

**CREATE FUNCTION FUNCTION\_DaysOfConference (@ConferenceID INT)**

**RETURNS @days TABLE (ConferenceDayID INT)**

**AS**

**BEGIN**

**INSERT INTO @days**

**SELECT cd.ConferenceDayID**

**FROM ConferenceDays AS cd**

**WHERE cd.Conferences\_ConferenceID = @ConferenceID**

**RETURN;**

**END**

## ConferenceDayParticipants

**CREATE FUNCTION FUNCTION\_ConferenceDayParticipants (@ConferenceDayID INT)**

**RETURNS @Participants TABLE (**

**firstname VARCHAR(40),**

**lastname VARCHAR(40),**

**email VARCHAR(100),**

**county VARCHAR(40),**

**city VARCHAR(40),**

**street VARCHAR(40),**

**postalcode VARCHAR(10)**

**)**

**AS**

**BEGIN**

**INSERT INTO @Participants**

**SELECT p.firstname,**

**p.lastname,**

**p.email,**

**p.county,**

**p.city,**

**p.Street,**

**p.PostalCode**

**FROM Participants AS p**

**INNER JOIN DayParticipants AS dp ON dp.Participants\_ParticipantID = p.ParticipantID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = dp.ConferenceDayBooking\_ConferenceDayBookingID**

**WHERE cdb.ConferenceDays\_ConferenceDayID = @ConferenceDayID**

**AND cdb.IsCancelled = 0**

**RETURN**

**END**

## WorkshopsPerConference

**CREATE FUNCTION FUNCTION\_WorkshopsPerConference (@ConferenceID INT)**

**RETURNS @Workshop TABLE (**

**workshopID INT,**

**name VARCHAR(100),**

**starttime DATE,**

**endtime DATE,**

**cost DECIMAL(9, 2),**

**numberofparticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO @Workshop**

**SELECT w.workshopid,**

**w.name,**

**w.StartTime,**

**w.EndTime,**

**w.cost,**

**w.NumberOfParticipants**

**FROM Workshops AS w**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**WHERE cd.Conferences\_ConferenceID = @ConferenceID**

**RETURN**

**END**

## WorkshopDate

**CREATE FUNCTION FUNCTION\_WorkshopDate (@WorkshopID INT)**

**RETURNS @WShop TABLE (**

**workshopID INT,**

**name VARCHAR(100),**

**startime DATE,**

**endtime DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO @WShop**

**SELECT workshopid,**

**name,**

**StartTime,**

**EndTime**

**FROM Workshops**

**WHERE WorkshopID = @WorkshopID**

**RETURN**

**END**

## BookingDaysCost

**CREATE FUNCTION FUNCTION\_BookingDaysCost (@ConferenceBookingID INT)**

**RETURNS DECIMAL(9, 2)**

**AS**

**BEGIN**

**RETURN (**

**SELECT SUM(cdb.numberofparticipants \* cc.cost + cdb.numberofstudents \* cc.cost \* (1.00 - c.discountforstudents))**

**FROM ConferenceDayBooking cdb**

**INNER JOIN conferencebooking cb ON cdb.ConferenceBooking\_ConferenceBookingID = cb.ConferenceBookingID**

**INNER JOIN conferences c ON cb.conferences\_conferenceid = c.conferenceid**

**INNER JOIN conferencecosts cc ON c.conferenceid = cc.conferences\_conferenceid**

**WHERE cb.conferencebookingid = @ConferenceBookingID**

**GROUP BY cb.conferencebookingid**

**);**

**END**

## BookingWorkshopCost

**CREATE FUNCTION FUNCTION\_BookingWorkshopCost (@ConferenceBookID INT)**

**RETURNS DECIMAL(9, 2)**

**AS**

**BEGIN**

**RETURN (**

**SELECT isnull(SUM(wb.numberofparticipants \* w.cost), 0)**

**FROM ConferenceBooking cb**

**LEFT JOIN conferencedaybooking cdb ON cb.conferencebookingid = cdb.ConferenceBooking\_ConferenceBookingID**

**LEFT JOIN workshopbooking wb ON cdb.conferencedaybookingid = wb.conferencedaybooking\_conferencedaybookingid**

**LEFT JOIN workshops w ON wb.workshops\_workshopid = w.workshopid**

**WHERE cb.conferencebookingid = @ConferenceBookID**

**GROUP BY cb.conferencebookingid**

**);**

**END**

## TotalBookingCost

**CREATE FUNCTION FUNCTION\_TotalBookingCost (@ConferenceBookID INT)**

**RETURNS DECIMAL(9, 2)**

**AS**

**BEGIN**

**RETURN (**

**SELECT dbo.FUNCTION\_BookingDaysCost(bs.conferencebookingid) + dbo.FUNCTION\_BookingWorkshopCost(bs.conferencebookingid)**

**FROM conferencebooking bs**

**WHERE bs.conferencebookingid = @ConferenceBookID**

**);**

**END**

## WorkshopListForParticipant

**CREATE FUNCTION FUNCTION\_WorkshopListForParticipant (@Participant INT)**

**RETURNS @table TABLE (**

**workshopid INT,**

**Name VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT w.workshopid,**

**w.name**

**FROM participants AS p**

**INNER JOIN dayparticipants AS dp ON p.participantid = dp.participants\_participantid**

**INNER JOIN workshopparticipants AS wp ON dp.dayparticipantid = wp.dayparticipants\_dayparticipantid**

**INNER JOIN WorkshopBooking AS wb ON wp.WorkshopBooking\_WorkshopBookingID = wb.workshopbookingid**

**INNER JOIN workshops AS w ON wb.workshops\_workshopid = w.workshopid**

**WHERE p.participantid = @Participant**

**RETURN**

**END**

## ConferencesDaysListForParticipant

**CREATE FUNCTION FUNCTION\_ConferencesDaysListForParticipant (@Participant INT)**

**RETURNS @table TABLE (**

**name VARCHAR(100),**

**place VARCHAR(100),**

**DATE DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT c.name,**

**c.place,**

**cd.DATE**

**FROM participants AS p**

**INNER JOIN dayparticipants AS dp ON p.participantid = dp.participants\_participantid**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = dp.ConferenceDayBooking\_ConferenceDayBookingID**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = cdb.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c ON c.ConferenceID = cd.Conferences\_ConferenceID**

**WHERE p.participantid = @Participant**

**RETURN**

**END**

## ClientsOrdersList

**CREATE FUNCTION FUNCTION\_ClientsOrdersList (@ClientID INT)**

**RETURNS @table TABLE (**

**conferenceid INT,**

**name VARCHAR(100),**

**place VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT c.ConferenceID,**

**c.name,**

**c.place**

**FROM Conferences AS c**

**INNER JOIN ConferenceBooking AS cb ON cb.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN Clients AS cl ON cl.ClientID = cb.Clients\_ClientID**

**WHERE cl.ClientID = @ClientID**

**RETURN**

**END**

# Triggery

## TooFewFreePlacesForDayBooking

Sprawdza, czy jest wystarczająca liczba miejsc w dniu konferencji.

**CREATE TRIGGER TRIGGER\_TooFewFreePlacesForDayBooking ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeDayPlaces(a.ConferenceDays\_ConferenceDayID) < 0**

**)**

**BEGIN**

**SELECT 'Brak wystarczajacej liczby miejsc w dniu konferencji'**

**END**

**END**

## TooFewFreePlacesForWorkshopBooking

Sprawdza, czy jest wystarczająca liczba miejsc na warsztacie.

**CREATE TRIGGER TRIGGER\_TooFewFreePlacesForWorkshopBooking ON WorkshopBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeWorkshopPlaces(a.Workshops\_WorkshopID) < 0**

**)**

**BEGIN**

**SELECT 'Brak wystarczajacej liczby miejsc w warsztacie'**

**END**

**END**

## LessPlacesForDayThanForWorkshop

Blokuje rezerwację na warsztat, jeżeli klient zarezerwował mniej miejsc na dzień niż warsztat.

**CREATE TRIGGER TRIGGER\_LessPlacesForDayThanForWorkshop ON ConferenceDayBooking**

**AFTER INSERT,**

**UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeDayPlaces(a.ConferenceDays\_ConferenceDayID) < 0**

**)**

**BEGIN**

**SELECT 'Klient zarezerwował mniej miejsc na dzień niż na warsztat'**

**END**

**END**

## NotEnoughBookedPlacesForDay

Blokuje zapis uczestnika na dzień konferencji, jeżeli wszystkie miejsca od klienta są już zajęte.

**CREATE TRIGGER TRIGGER\_NotEnoughBookedPlacesForDay ON DayParticipants**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE (**

**a.StudentID IS NULL**

**AND dbo.FUNCTION\_FreeDayPlacesForStudents(a.ConferenceDayBooking\_ConferenceDayBookingID) < 0**

**)**

**OR (**

**a.StudentID IS NULL**

**AND dbo.FUNCTION\_FreeDayPlacesForParticipants(a.ConferenceDayBooking\_ConferenceDayBookingID) < 0**

**)**

**)**

**BEGIN**

**SELECT 'Wszystkie miejsca klienta zostały już zarezerwowane'**

**END**

**END**

## NotEnoughBookedPlacesForWorkshop

Blokuje zapis uczestnika na warsztat, jeżeli wszystkie zarezerwowane miejsca są już zajęte.

**CREATE TRIGGER TRIGGER\_NotEnoughBookedPlacesForWorkshop ON WorkshopParticipants**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeWorkshopPlaces(a.WorkshopBooking\_WorkshopBookingID) < 0**

**)**

**BEGIN**

**SELECT 'Wszystkie zarezerwowane miejsca są już zajęte'**

**END**

**END**

## TooFewPlacesAfterDecreasingDayCapacity

Sprawdza, czy po zmniejszeniu liczby miejsc na dzień konferencji zarezerwowane miejsca mieszczą się w nowym limicie.

**CREATE TRIGGER TRIGGER\_TooFewPlacesAfterDecreasingDayCapacity ON ConferenceDays**

**AFTER UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = a.ConferenceDayID**

**GROUP BY a.ConferenceDayID,**

**a.NumberOfParticipants**

**HAVING a.NumberOfParticipants < SUM(cdb.NumberOfParticipants) + SUM(cdb.NumberOfStudents)**

**)**

**BEGIN**

**SELECT 'Po zmniejszeniu liczby miejsc na dzień konferencji zarezerwowane miejsca nie mieszczą się w nowym limicie'**

**END**

**END**

## TooFewPlacesAfterDecreasingWorkshopCapacity

Sprawdza, czy po zmniejszeniu liczby miejsc na warsztat zarezerwowane miejsca mieszczą się w nowym limicie.

**CREATE TRIGGER TRIGGER\_TooFewPlacesAfterDecreasingWorkshopCapacity ON Workshops**

**AFTER UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = a.WorkshopID**

**GROUP BY a.WorkshopID,**

**a.NumberOfParticipants**

**HAVING a.NumberOfParticipants < SUM(wb.NumberOfParticipants)**

**)**

**BEGIN**

**SELECT 'Po zmniejszeniu liczby miejsc na warsztat zarezerwowane miejsca nie mieszczą się w nowym limicie'**

**END**

**END**

## BookingDayInDifferentConference

Sprawdza, czy rezerwowany jest dzień z konferencji odpowiadającej rezerwacji na konferencję.

**CREATE TRIGGER TRIGGER\_BookingDayInDifferentConference ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = a.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c1 ON c1.ConferenceID = cd.Conferences\_ConferenceID**

**INNER JOIN ConferenceBooking AS cb ON cb.ConferenceBookingID = a.ConferenceBooking\_ConferenceBookingID**

**INNER JOIN Conferences AS c2 ON c2.ConferenceID = cb.Conferences\_ConferenceID**

**WHERE c1.ConferenceID != c2.ConferenceID**

**)**

**BEGIN**

**SELECT 'Klient próbuje przepisać do konferencji rezerwację dnia z innej konferencji'**

**END**

**END**

## BookingDayAlreadyExists

Sprawdza, czy rezerwacja danego dnia konferencji już istnieje.

**CREATE TRIGGER TRIGGER\_BookingDayAlreadyExists ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceDayBooking AS cbd ON a.ConferenceBooking\_ConferenceBookingID = cbd.ConferenceBooking\_ConferenceBookingID**

**AND a.ConferenceDays\_ConferenceDayID = cbd.ConferenceDays\_ConferenceDayID**

**WHERE a.ConferenceBooking\_ConferenceBookingID != cbd.ConferenceBooking\_ConferenceBookingID**

**)**

**BEGIN**

**SELECT 'Rezerwacja danego dnia konferencji już istnieje'**

**END**

**END**

## BookingWorkshopInDifferentDay

Sprawdza, czy rezerwowany jest warsztat z dnia odpowiadającemu rezerwacji.

**CREATE TRIGGER TRIGGER\_BookingWorkshopInDifferentDay ON WorkshopBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**INNER JOIN Workshops AS w ON w.WorkshopID = a.Workshops\_WorkshopID**

**INNER JOIN ConferenceDays AS cd1 ON cd1.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = a.ConferenceDayBooking\_ConferenceDayBookingID**

**INNER JOIN ConferenceDays AS cd2 ON cd2.ConferenceDayID = cdb.ConferenceDays\_ConferenceDayID**

**WHERE cd1.Conferences\_ConferenceID != cd2.Conferences\_ConferenceID**

**)**

**BEGIN**

**SELECT 'Klient próbuje zapisać się do warsztatu z innego dnia niż jego rezerwacja'**

**END**

**END**

## ArePriceThresholdsMonotonous

Sprawdza, czy progi cenowe konferencji są ułożone w porządku rosnącym w stosunku do czasu pozostałego do konferencji.

**CREATE TRIGGER TRIGGER\_ArePriceThresholdsMonotonous ON ConferenceCosts**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**DECLARE @Cost DECIMAL(9, 2) = (**

**SELECT a.Cost**

**FROM inserted AS a**

**);**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceCosts AS cc ON a.Conferences\_ConferenceID = cc.Conferences\_ConferenceID**

**WHERE (**

**(**

**cc.DateFrom < a.DateFrom**

**AND a.DateFrom < cc.dateto**

**)**

**OR (**

**a.DateFrom < cc.DateFrom**

**AND cc.DateTo < a.DateTo**

**)**

**OR (**

**cc.DateFrom >= a.DateFrom**

**AND a.DateTo >= cc.DateTo**

**)**

**OR (**

**a.DateFrom >= cc.DateFrom**

**AND cc.DateTo >= a.DateTo**

**)**

**)**

**AND cc.ConferenceCostID != a.ConferenceCostID**

**)**

**BEGIN**

**SELECT 'Koszt pokrywa się z istniejącymi kosztami'**

**END**

**ELSE**

**BEGIN**

**DECLARE @PreviousCost DECIMAL(9, 2) = (**

**SELECT TOP 1 a.Cost**

**FROM inserted AS a**

**INNER JOIN ConferenceCosts AS cc ON cc.ConferenceCostID = a.ConferenceCostID**

**WHERE cc.Conferences\_ConferenceID = a.Conferences\_ConferenceID**

**AND cc.DateTo < a.DateFrom**

**ORDER BY cc.DateFrom DESC**

**)**

**DECLARE @NextCost DECIMAL(9, 2) = (**

**SELECT TOP 1 a.Cost**

**FROM inserted AS a**

**INNER JOIN ConferenceCosts AS cc ON cc.ConferenceCostID = a.ConferenceCostID**

**WHERE cc.Conferences\_ConferenceID = a.Conferences\_ConferenceID**

**AND cc.DateFrom > a.DateTo**

**ORDER BY cc.DateFrom**

**)**

**IF (**

**(**

**@PreviousCost IS NOT NULL**

**AND @PreviousCost >= @Cost**

**)**

**OR (**

**@NextCost IS NOT NULL**

**AND @NextCost <= @Cost**

**)**

**)**

**BEGIN**

**SELECT 'Cena nie jest w poprawnej kolejności z poprzednimi (PreviousCost = %, NextCost = %.',**

**@PreviousCost,**

**@NextCost;**

**END**

**END**

**END**