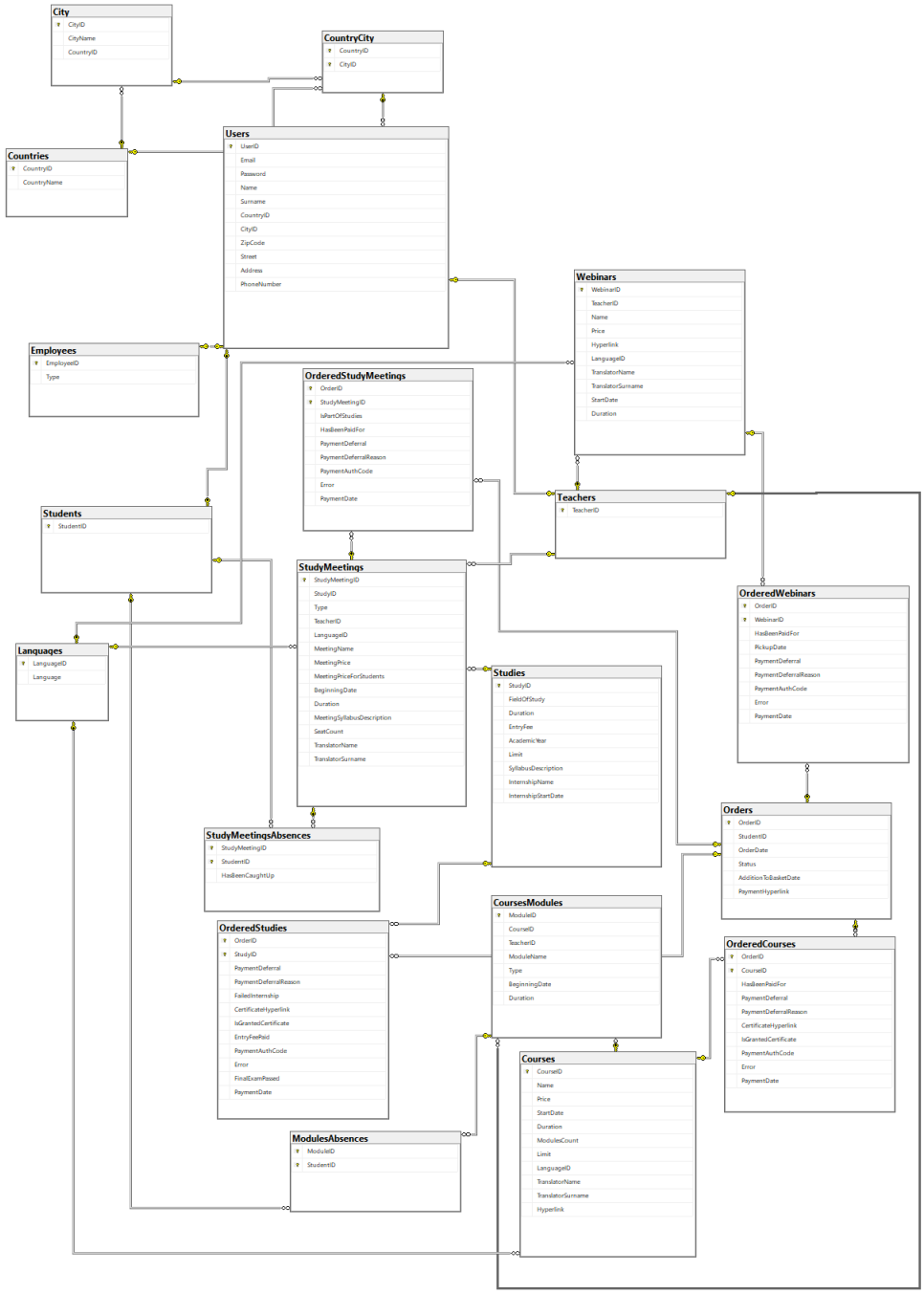
pbd_<14>_raport5 | Piotr Albiński, Adam Konior, Mateusz Maciaszczyk

Identyfikacja użytkowników:

- pracownik biura obsługi dydaktyki:
  - wprowadzenie informacji o użytkownikach, pracownikach dodawanie i usuwanie użytkowników z systemu,
  - zarządzanie danymi np. usuwanie dostępu do webinarów, ustalenia cen produktów,
  - wprowadzanie harmonogramów (również ich zmiana),
  - przypisywanie kursom/webinarium/studium wykładowców/nauczycieli,
  - odroczenie płatności (decyzją Dyrektora Szkoły),
  - generowanie raportów:
    - finansowych - zestawienie przychodów z różnych form nauczania,
    - listy dłużników,
    - ogólny raport dotyczący liczby zapisanych osób na przyszłe wydarzenia,
    - lista obecności,
    - lista osób z kolizjami w terminach zajęć,
    - bilokacji wszystkich nauczycieli, uczniów
  - dodawanie produktów do sklepu(całościowych webinarów/kursów/studium),
  - usuwanie produktów ze sklepu,
  - wprowadzanie sylabusa do systemu
  - generowanie listy kursantów, którzy ukończyli kurs,
  - Funkcje do naprawy błędów/dokonywania zmian:
    - modyfikowanie listy uczestników danego kursu/studium/webinaru(np. dodawanie uczestników po rozpoczęciu webinaru, usuwanie uczestników, którzy zrezygnowali),
- Dyrektor:
  - decyduje o odroczeniu płatności
  - weryfikuje ukończenie kursów/studium i podejmuje decyzję o wysłaniu dyplomów (np. generowanie listy absolwentów),
  - generowanie listy kursantów, którzy ukończyli kurs,
- klient firmy/ student:
  - zakładanie konta w systemie,
  - logowanie do konta w systemie,
  - wyświetlanie i zarządzanie profilem,
  - dodawanie produktów do koszyka,
  - opłacanie wybranych produktów (samą płatność stanowi zewnętrzny system, którego nie mamy implementować),
  - generowanie własnych kolizji w planie zajęć,
  - sprawdzenie własnego długu,
  - dostęp do informacji o poszczególnych webinarach:
    - język wykładu,
    - dane prowadzącego,
  - możliwość zapisania się do odrobienia zajęć,
  - weryfikacja postępu w kursie (obecność, zaliczenie obejrzenia materiału),
  - generowanie raportu własnej frekwencji,
  - ogólny raport dotyczący liczby zapisanych osób na przyszłe wydarzenia,
  - ogólny raport dotyczący frekwencji
  - raport bilokacji własnych zajęć
- nauczyciel
  - udostępnianie webinarów(dodawanie do bazy rekordów z linkami),
  - generowanie raportów:
    - lista obecności (na zajęciach, prowadzonych przez siebie),
    - bilokacji (raport bilokacji własnych uczniów),
    - dot. frekwencji (raporty frekwencji własnych zajęć),
    - dot. osób zapisanych na przyszłe wydarzenia (raporty na temat osób zapisanych na zajęcia prowadzone przez siebie),
  - wprowadzenia frekwencji do systemu,
- system:
  - generowanie linku do płatności,
  - informacja zwrotna o statusie transakcji i dodanie dostępu do produktu do konta,
  - automatycznie sprawdzenie obecności,
  - weryfikacja obejrzenia materiału,
  - weryfikowanie warunków ukończenia kursów/studium,
  - ustalenie limitu miejsc,
  - weryfikowanie przekroczenia limitu miejsc: kursy hybrydowe i stacjonarne.



Skrypty tworzenia tabel:

Tabela City:

- lista wszystkich miast

```
CREATE TABLE [dbo].[City](
    [CityID] [int] IDENTITY(1,1) NOT NULL,
    [CityName] [nvarchar](50) NOT NULL,
    [CountryID] [int] NOT NULL,
 CONSTRAINT [PK_City] PRIMARY KEY CLUSTERED
(
    [CityID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[City]  WITH CHECK ADD  CONSTRAINT [FK_City_Countries] FOREIGN KEY([CountryID])
REFERENCES [dbo].[Countries] ([CountryID])
GO

ALTER TABLE [dbo].[City] CHECK CONSTRAINT [FK_City_Countries]
GO
```

Tabela Countries:

- lista wszystkich państw

```
CREATE TABLE [dbo].[Countries](
    [CountryID] [int] IDENTITY(1,1) NOT NULL,
    [CountryName] [nchar](50) NOT NULL,
 CONSTRAINT [PK_Countries] PRIMARY KEY CLUSTERED
(
    [CountryID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Countries]  WITH CHECK ADD  CONSTRAINT [NotEmptyCountryName] CHECK  (([CountryName]<>''))
GO

ALTER TABLE [dbo].[Countries] CHECK CONSTRAINT [NotEmptyCountryName]
GO
```

Tabela CountryCity:

- tabela która łączy kraje z miastami, używamy do walidacji czy dane miasto znajduje się w danym państwie

```
CREATE TABLE [dbo].[CountryCity](
    [CountryID] [int] NOT NULL,
    [CityID] [int] NOT NULL,
 CONSTRAINT [PK_CountryCity] PRIMARY KEY CLUSTERED
(
    [CountryID] ASC,
    [CityID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CountryCity]  WITH CHECK ADD  CONSTRAINT [FK_CountryCity_City] FOREIGN KEY([CityID])
REFERENCES [dbo].[City] ([CityID])
GO

ALTER TABLE [dbo].[CountryCity] CHECK CONSTRAINT [FK_CountryCity_City]
GO

ALTER TABLE [dbo].[CountryCity]  WITH CHECK ADD  CONSTRAINT [FK_CountryCity_Countries] FOREIGN KEY([CountryID])
REFERENCES [dbo].[Countries] ([CountryID])
GO

ALTER TABLE [dbo].[CountryCity] CHECK CONSTRAINT [FK_CountryCity_Countries]
GO
```

Tabela Courses:

- tabela zawiera informacje na temat wszystkich kursów
- duration: czas trwania kursu
- modulesCount: liczba modułów, z których składa się kurs
- limit: ile osób może maksymalnie uczestniczyć w kursie

```
CREATE TABLE [dbo].[Courses](
    [CourseID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [Price] [money] NOT NULL,
    [StartDate] [datetime] NOT NULL,
    [Duration] [int] NOT NULL,
    [ModulesCount] [int] NOT NULL,
    [Limit] [int] NOT NULL,
    [LanguageID] [int] NOT NULL,
    [TranslatorName] [nvarchar](50) NULL,
    [TranslatorSurname] [nvarchar](50) NULL,
    [Hyperlink] [nvarchar](100) NOT NULL,
 CONSTRAINT [PK_Courses] PRIMARY KEY CLUSTERED
(
    [CourseID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [FK_Courses_Languages] FOREIGN KEY([LanguageID])
REFERENCES [dbo].[Languages] ([LanguageID])
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [FK_Courses_Languages]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [C_TranslatorName] CHECK  (([TranslatorName]<>'' AND [TranslatorSurname]<>''))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [C_TranslatorName]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [Duration] CHECK  (([Duration]>(0)))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [Duration]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [Limit] CHECK  (([Limit]>(0)))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [Limit]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [ModulesCount] CHECK  (([ModulesCount]>(0)))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [ModulesCount]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [Name] CHECK  (([Name]<>''))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [Name]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [Price] CHECK  (([Price]>(0)))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [Price]
GO
```

Tabela CoursesModules:

- tabela zawiera informacje na temat modułów, z których składa się kurs(courseID identyfikator kursu, w którym zawiera się dany moduł)
- type: typ modułu np. stacjonarne, online...
- BeginningDate, EndingDate: data rozpoczęcia i zakończenia kursu

```
CREATE TABLE [dbo].[CoursesModules](
    [ModuleID] [int] IDENTITY(1,1) NOT NULL,
    [CourseID] [int] NOT NULL,
    [TeacherID] [int] NOT NULL,
    [ModuleName] [nvarchar](50) NOT NULL,
    [Type] [nvarchar](50) NOT NULL,
    [BeginningDate] [datetime] NOT NULL,
    [Duration] [time](7) NOT NULL,
 CONSTRAINT [PK_CoursesModules] PRIMARY KEY CLUSTERED
(
    [ModuleID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CoursesModules]  WITH CHECK ADD  CONSTRAINT [FK_CoursesModules_Courses] FOREIGN KEY([CourseID])
```

```
REFERENCES [dbo].[Courses] ([CourseID])
GO

ALTER TABLE [dbo].[CoursesModules] CHECK CONSTRAINT [FK_CoursesModules_Courses]
GO

ALTER TABLE [dbo].[CoursesModules]  WITH CHECK ADD  CONSTRAINT [FK_CoursesModules_Teachers] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Teachers] ([TeacherID])
GO

ALTER TABLE [dbo].[CoursesModules] CHECK CONSTRAINT [FK_CoursesModules_Teachers]
GO

ALTER TABLE [dbo].[CoursesModules]  WITH CHECK ADD  CONSTRAINT [Type] CHECK  (([Type]='Online Asynchroniczny' OR [Type]='Online Synchroniczny' OR [Type]='Stacjonarny' OR [Type]='Hybrydowy'))
GO

ALTER TABLE [dbo].[CoursesModules] CHECK CONSTRAINT [Type]
GO
```

Tabela Employees:

- tabela zawiera osoby, które są pracownikami
- type określa czy jest to pracownik biura czy dyrektor

```
CREATE TABLE [dbo].[Employees](
    [EmployeeID] [int] NOT NULL,
    [Type] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Employees] PRIMARY KEY CLUSTERED
(
    [EmployeeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Employees]  WITH CHECK ADD  CONSTRAINT [FK_Employees_Users] FOREIGN KEY([EmployeeID])
REFERENCES [dbo].[Users] ([UserID])
GO

ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [FK_Employees_Users]
GO

ALTER TABLE [dbo].[Employees]  WITH CHECK ADD  CONSTRAINT [E_Type] CHECK  (([Type]='Secretary' OR [Type]='Headmaster'))
GO

ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [E_Type]
GO
```

Tabela Languages

- słownik języków

```
CREATE TABLE [dbo].[Languages](
    [LanguageID] [int] IDENTITY(1,1) NOT NULL,
    [Language] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Languages] PRIMARY KEY CLUSTERED
(
    [LanguageID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Tabela ModulesAbsences:

- tabela zawiera informacje, który student nie był na którym module z kursów

```
CREATE TABLE [dbo].[ModulesAbsences](
    [ModuleID] [int] NOT NULL,
    [StudentID] [int] NOT NULL,
 CONSTRAINT [PK_ModulesAbsences] PRIMARY KEY CLUSTERED
(
    [ModuleID] ASC,
    [StudentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ModulesAbsences]  WITH CHECK ADD  CONSTRAINT [FK_ModulesAbsences_CoursesModules] FOREIGN KEY([ModuleID])
REFERENCES [dbo].[CoursesModules] ([ModuleID])
GO

ALTER TABLE [dbo].[ModulesAbsences] CHECK CONSTRAINT [FK_ModulesAbsences_CoursesModules]
GO

ALTER TABLE [dbo].[ModulesAbsences]  WITH CHECK ADD  CONSTRAINT [FK_ModulesAbsences_Students] FOREIGN KEY([StudentID])
REFERENCES [dbo].[Students] ([StudentID])
GO

ALTER TABLE [dbo].[ModulesAbsences] CHECK CONSTRAINT [FK_ModulesAbsences_Students]
GO
```

Tabela OrderedCourses:

- tabela zawiera informacje na temat zamówionych kursów
- IsGrantedCertificate: czy został przyznany certyfikat
- CertificateHyperlink: link do certyfikatu

```
CREATE TABLE [dbo].[OrderedCourses](
    [OrderID] [nvarchar](50) NOT NULL,
    [CourseID] [int] NOT NULL,
    [HasBeenPaidFor] [bit] NOT NULL,
    [PaymentDeferral] [bit] NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [CertificateHyperlink] [nvarchar](100) NULL,
    [IsGrantedCertificate] [bit] NULL,
    [PaymentAuthCode] [nvarchar](50) NULL,
    [Error] [nvarchar](max) NULL,
    [PaymentDate] [datetime] NULL,
 CONSTRAINT [PK_OrderedCourses] PRIMARY KEY CLUSTERED
(
    [OrderID] ASC,
    [CourseID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[OrderedCourses] ADD  CONSTRAINT [DF_OrderedCourses_PaymentDeferral]  DEFAULT ((0)) FOR [PaymentDeferral]
GO

ALTER TABLE [dbo].[OrderedCourses] ADD  CONSTRAINT [DF_OrderedCourses_IsGrantedCertificate]  DEFAULT ((0)) FOR [IsGrantedCertificate]
GO

ALTER TABLE [dbo].[OrderedCourses]  WITH CHECK ADD  CONSTRAINT [FK_OrderedCourses_Courses] FOREIGN KEY([CourseID])
REFERENCES [dbo].[Courses] ([CourseID])
GO
```

```
ALTER TABLE [dbo].[OrderedCourses] CHECK CONSTRAINT [FK_OrderedCourses_Courses]
GO

ALTER TABLE [dbo].[OrderedCourses]  WITH CHECK ADD  CONSTRAINT [FK_OrderedCourses_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO

ALTER TABLE [dbo].[OrderedCourses] CHECK CONSTRAINT [FK_OrderedCourses_Orders]
GO

ALTER TABLE [dbo].[OrderedCourses]  WITH CHECK ADD  CONSTRAINT [OC_Certificates] CHECK  (([IsGrantedCertificate]=(0) AND [CertificateHyperlink] IS NULL OR [CertificateHyperlink] IS NOT NULL))
GO

ALTER TABLE [dbo].[OrderedCourses] CHECK CONSTRAINT [OC_Certificates]
GO

ALTER TABLE [dbo].[OrderedCourses]  WITH CHECK ADD  CONSTRAINT [OC_PaymentDeferral] CHECK  (([PaymentDeferral]=(0) AND [PaymentDeferralReason] IS NULL OR [PaymentDeferral]=(1)))
GO

ALTER TABLE [dbo].[OrderedCourses] CHECK CONSTRAINT [OC_PaymentDeferral]
GO
```

Tabela OrderedStudies:

- tabela zawiera informacje na temat zamówionych studiów
- FailedInternship: czy praktyki zostały zaliczone
- EntryFeePaid: czy opłata rekrutacyjna została opłacona

```
CREATE TABLE [dbo].[OrderedStudies](
    [OrderID] [nvarchar](50) NOT NULL,
    [StudyID] [int] NOT NULL,
    [PaymentDeferral] [bit] NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [FailedInternship] [bit] NULL,
    [CertificateHyperlink] [nvarchar](100) NULL,
    [IsGrantedCertificate] [bit] NULL,
    [EntryFeePaid] [bit] NOT NULL,
    [PaymentAuthCode] [nvarchar](50) NULL,
    [Error] [nvarchar](max) NULL,
    [FinalExamPassed] [bit] NULL,
    [PaymentDate] [datetime] NULL,
 CONSTRAINT [PK_OrderedStudies_1] PRIMARY KEY CLUSTERED
(
    [OrderID] ASC,
    [StudyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[OrderedStudies] ADD  CONSTRAINT [DF_OrderedStudies_PaymentDeferral]  DEFAULT ((0)) FOR [PaymentDeferral]
GO

ALTER TABLE [dbo].[OrderedStudies] ADD  CONSTRAINT [DF_OrderedStudies_IsGrantedCertificate]  DEFAULT ((0)) FOR [IsGrantedCertificate]
GO

ALTER TABLE [dbo].[OrderedStudies]  WITH CHECK ADD  CONSTRAINT [FK_OrderedStudies_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO

ALTER TABLE [dbo].[OrderedStudies] CHECK CONSTRAINT [FK_OrderedStudies_Orders]
GO

ALTER TABLE [dbo].[OrderedStudies]  WITH CHECK ADD  CONSTRAINT [FK_OrderedStudies_Studies] FOREIGN KEY([StudyID])
REFERENCES [dbo].[Studies] ([StudyID])
GO

ALTER TABLE [dbo].[OrderedStudies] CHECK CONSTRAINT [FK_OrderedStudies_Studies]
GO

ALTER TABLE [dbo].[OrderedStudies]  WITH CHECK ADD  CONSTRAINT [OS_Certificates] CHECK  (([IsGrantedCertificate]=(0) AND [CertificateHyperlink] IS NULL OR [CertificateHyperlink] IS NOT NULL OR [FailedInternship]=(0) AND
[CertificateHyperlink] IS NULL))
GO

ALTER TABLE [dbo].[OrderedStudies] CHECK CONSTRAINT [OS_Certificates]
GO

ALTER TABLE [dbo].[OrderedStudies]  WITH CHECK ADD  CONSTRAINT [OS_PaymentDeferral] CHECK  (([PaymentDeferral]=(0) AND [PaymentDeferralReason] IS NULL OR [PaymentDeferral]=(1)))
GO

ALTER TABLE [dbo].[OrderedStudies] CHECK CONSTRAINT [OS_PaymentDeferral]
GO
```

Tabela OrderedStudyMeetings:

- tabela zawiera informacje na temat zamówionych pojedynczych spotkań z toku studiów
- IsPartOfStudies: czy osoba która zamówiła spotkanie bierze udział w studiach
- LeftPayment: ile zostało do zapłacenia

```
CREATE TABLE [dbo].[OrderedStudyMeetings](
    [OrderID] [nvarchar](50) NOT NULL,
    [StudyMeetingID] [int] NOT NULL,
    [IsPartOfStudies] [bit] NOT NULL,
    [HasBeenPaidFor] [bit] NOT NULL,
    [PaymentDeferral] [bit] NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [PaymentAuthCode] [nvarchar](50) NULL,
    [Error] [nvarchar](max) NULL,
    [PaymentDate] [datetime] NULL,
 CONSTRAINT [PK_OrderedStudyMeetings_1] PRIMARY KEY CLUSTERED
(
    [StudyMeetingID] ASC,
    [OrderID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[OrderedStudyMeetings] ADD  CONSTRAINT [DF_OrderedStudyMeetings_PaymentDeferral]  DEFAULT ((0)) FOR [PaymentDeferral]
GO

ALTER TABLE [dbo].[OrderedStudyMeetings]  WITH CHECK ADD  CONSTRAINT [FK_OrderedStudyMeetings_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO

ALTER TABLE [dbo].[OrderedStudyMeetings] CHECK CONSTRAINT [FK_OrderedStudyMeetings_Orders]
GO

ALTER TABLE [dbo].[OrderedStudyMeetings]  WITH CHECK ADD  CONSTRAINT [FK_OrderedStudyMeetings_StudyMeetings] FOREIGN KEY([StudyMeetingID])
REFERENCES [dbo].[StudyMeetings] ([StudyMeetingID])
GO

ALTER TABLE [dbo].[OrderedStudyMeetings] CHECK CONSTRAINT [FK_OrderedStudyMeetings_StudyMeetings]
GO

ALTER TABLE [dbo].[OrderedStudyMeetings]  WITH CHECK ADD  CONSTRAINT [OSM_PaymentDeferral] CHECK  (([PaymentDeferral]=(0) AND [PaymentDeferralReason] IS NULL OR [PaymentDeferral]=(1)))
GO

ALTER TABLE [dbo].[OrderedStudyMeetings] CHECK CONSTRAINT [OSM_PaymentDeferral]
GO
```

Tabela OrderedWebinars:

- tabela zawiera informacje na temat zamówionych webinariów
- OrderID: klucz obcy, który wskazuje na tabele Orders, do którego zamówienia należy dany webinar
- LeftPayment: ile zostało do zapłacenia
- PickupDate: okres, na który został zakupiony webinar
- PaymentDeferral, PaymentDeferralReasson: czy płatność została odroczona oraz powód

```
CREATE TABLE [dbo].[OrderedWebinars](
    [OrderID] [nvarchar](50) NOT NULL,
    [WebinarID] [int] NOT NULL,
    [HasBeenPaidFor] [bit] NOT NULL,
    [PickupDate] [datetime] NULL,
    [PaymentDeferral] [bit] NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [PaymentAuthCode] [nvarchar](50) NULL,
    [Error] [nvarchar](max) NULL,
    [PaymentDate] [datetime] NULL,
 CONSTRAINT [PK_OrderedWebinars] PRIMARY KEY CLUSTERED
(
    [OrderID] ASC,
    [WebinarID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[OrderedWebinars] ADD  CONSTRAINT [DF_OrderedWebinars_PaymentDeferral]  DEFAULT ((0)) FOR [PaymentDeferral]
GO

ALTER TABLE [dbo].[OrderedWebinars]  WITH CHECK ADD  CONSTRAINT [FK_OrderedWebinars_Orders] FOREIGN KEY([OrderID])
REFERENCES [dbo].[Orders] ([OrderID])
GO

ALTER TABLE [dbo].[OrderedWebinars] CHECK CONSTRAINT [FK_OrderedWebinars_Orders]
GO

ALTER TABLE [dbo].[OrderedWebinars]  WITH CHECK ADD  CONSTRAINT [FK_OrderedWebinars_Webinars] FOREIGN KEY([WebinarID])
REFERENCES [dbo].[Webinars] ([WebinarID])
GO

ALTER TABLE [dbo].[OrderedWebinars] CHECK CONSTRAINT [FK_OrderedWebinars_Webinars]
GO

ALTER TABLE [dbo].[OrderedWebinars]  WITH CHECK ADD  CONSTRAINT [OW_PaymentDeferral] CHECK  (([PaymentDeferral]=(0) AND [PaymentDeferralReason] IS NULL OR [PaymentDeferral]=(1)))
GO

ALTER TABLE [dbo].[OrderedWebinars] CHECK CONSTRAINT [OW_PaymentDeferral]
GO
```

Tabela Orders:

- tabela pełni rolę koszyka, zapisuje dane, który student co ma w koszyku oraz kiedy to zamówił
- status: informacja czy produkt jest w koszyku, czy płatność jest przetwarzana oraz czy produkt już jest zamówiony

```
CREATE TABLE [dbo].[Orders](
    [OrderID] [nvarchar](50) NOT NULL,
    [StudentID] [int] NOT NULL,
    [OrderDate] [datetime] NULL,
    [Status] [nvarchar](50) NOT NULL,
    [AdditionToBasketDate] [datetime] NULL,
    [PaymentHyperlink] [nvarchar](max) NULL,
 CONSTRAINT [PK_Orders] PRIMARY KEY CLUSTERED
(
    [OrderID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Orders]  WITH CHECK ADD  CONSTRAINT [O_Status] CHECK  (([Status]='Delivered' OR [Status]='Pending'))
GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [O_Status]
GO
```

Tabela Students:

- tabela zawiera wszystkie osoby, które są uczniami/wykupiły jakiś kurs/webinar

```
CREATE TABLE [dbo].[Students](
    [StudentID] [int] NOT NULL,
 CONSTRAINT [PK_Students] PRIMARY KEY CLUSTERED
(
    [StudentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Students]  WITH CHECK ADD  CONSTRAINT [FK_Students_Users1] FOREIGN KEY([StudentID])
REFERENCES [dbo].[Users] ([UserID])
GO

ALTER TABLE [dbo].[Students] CHECK CONSTRAINT [FK_Students_Users1]
GO
```

Tabela Studies:

- tabela zawiera informacje na temat wszystkich studiów
- duration: ile semestrów trwają studia
- entryFee: opłata rekrutacyjna
- SyllabusDescription: opis toku studiów

```
CREATE TABLE [dbo].[Studies](
    [StudyID] [int] IDENTITY(1,1) NOT NULL,
    [FieldOfStudy] [nvarchar](50) NOT NULL,
    [Duration] [int] NOT NULL,
    [EntryFee] [money] NOT NULL,
    [AcademicYear] [int] NOT NULL,
    [Limit] [int] NOT NULL,
    [SyllabusDescription] [nvarchar](max) NOT NULL,
    [InternshipName] [nvarchar](50) NOT NULL,
    [InternshipStartDate] [datetime] NOT NULL,
 CONSTRAINT [PK_Studies] PRIMARY KEY CLUSTERED
(
    [StudyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [FieldOfStudy] UNIQUE NONCLUSTERED
(
    [FieldOfStudy] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Studies]  WITH CHECK ADD  CONSTRAINT [S_Duration] CHECK  (([Duration]>(0)))
GO
```

```
ALTER TABLE [dbo].[Studies] CHECK CONSTRAINT [S_Duration]
GO

ALTER TABLE [dbo].[Studies]  WITH CHECK ADD  CONSTRAINT [S_EntryFee] CHECK  (([EntryFee]>=(0)))
GO

ALTER TABLE [dbo].[Studies] CHECK CONSTRAINT [S_EntryFee]
GO

ALTER TABLE [dbo].[Studies]  WITH CHECK ADD  CONSTRAINT [S_Limit] CHECK  (([Limit]>(0)))
GO

ALTER TABLE [dbo].[Studies] CHECK CONSTRAINT [S_Limit]
GO

ALTER TABLE [dbo].[Studies]  WITH CHECK ADD  CONSTRAINT [S_NotEmpty] CHECK  (([SyllabusDescription]<>'' AND [InternshipName]<>''))
GO

ALTER TABLE [dbo].[Studies] CHECK CONSTRAINT [S_NotEmpty]
GO
```

Tabela StudyMeetings:

- tabela zawiera informacje na temat wszystkich spotkań w ramach studiów
- type: typ spotkania np. stacjonarne, zdalne, hybrydowe
- MeetingPrice, MeetingPriceForStudents: cena za pojedyncze spotkanie dla osoby spoza studiów oraz dla osoby zapisanej już na studia

```
CREATE TABLE [dbo].[StudyMeetings](
    [StudyMeetingID] [int] IDENTITY(1,1) NOT NULL,
    [StudyID] [int] NOT NULL,
    [Type] [nvarchar](50) NOT NULL,
    [TeacherID] [int] NOT NULL,
    [LanguageID] [int] NULL,
    [MeetingName] [nvarchar](50) NOT NULL,
    [MeetingPrice] [money] NOT NULL,
    [MeetingPriceForStudents] [money] NOT NULL,
    [BeginningDate] [datetime] NOT NULL,
    [Duration] [time](7) NULL,
    [MeetingSyllabusDescription] [nvarchar](1000) NOT NULL,
    [SeatCount] [int] NULL,
    [TranslatorName] [nvarchar](50) NULL,
    [TranslatorSurname] [nvarchar](50) NULL,
 CONSTRAINT [PK_StudyMeetings] PRIMARY KEY CLUSTERED
(
    [StudyMeetingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [FK_StudyMeetings_Languages] FOREIGN KEY([LanguageID])
REFERENCES [dbo].[Languages] ([LanguageID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Languages]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [FK_StudyMeetings_Studies] FOREIGN KEY([StudyID])
REFERENCES [dbo].[Studies] ([StudyID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Studies]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [FK_StudyMeetings_Teachers] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Teachers] ([TeacherID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Teachers]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [SM_Duration] CHECK  (([Duration]='01:30' OR [Duration]='00:45'))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [SM_Duration]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [SM_MeetingPrice] CHECK  (([MeetingPrice]>(0) AND [MeetingPriceForStudents]>(0)))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [SM_MeetingPrice]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [SM_MeetingSyllabus] CHECK  (([MeetingSyllabusDescription]<>''))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [SM_MeetingSyllabus]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [SM_SeatCount] CHECK  (([SeatCount]>(0)))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [SM_SeatCount]
GO
```

Tabela StudyMeetingsAbsences:

- tabela zawiera informacje, który student nie był na którym spotkaniu ze studiów
- HasBeenCaughtUp: informacja czy odrobił tę nieobecność

```
CREATE TABLE [dbo].[StudyMeetingsAbsences](
    [StudyMeetingID] [int] NOT NULL,
    [StudentID] [int] NOT NULL,
    [HasBeenCaughtUp] [bit] NOT NULL,
 CONSTRAINT [PK_StudyMeetingsAbsences_1] PRIMARY KEY CLUSTERED
(
    [StudyMeetingID] ASC,
    [StudentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[StudyMeetingsAbsences]  WITH CHECK ADD  CONSTRAINT [FK_StudyMeetingsAbsences_Students] FOREIGN KEY([StudentID])
REFERENCES [dbo].[Students] ([StudentID])
GO

ALTER TABLE [dbo].[StudyMeetingsAbsences] CHECK CONSTRAINT [FK_StudyMeetingsAbsences_Students]
GO

ALTER TABLE [dbo].[StudyMeetingsAbsences]  WITH CHECK ADD  CONSTRAINT [FK_StudyMeetingsAbsences_StudyMeetings1] FOREIGN KEY([StudyMeetingID])
REFERENCES [dbo].[StudyMeetings] ([StudyMeetingID])
GO

ALTER TABLE [dbo].[StudyMeetingsAbsences] CHECK CONSTRAINT [FK_StudyMeetingsAbsences_StudyMeetings1]
GO
```

Tabela Teachers:

- tabela zawiera wszystkie osoby, które są nauczycielami

```
CREATE TABLE [dbo].[Teachers](
    [TeacherID] [int] NOT NULL,
 CONSTRAINT [PK_Teachers] PRIMARY KEY CLUSTERED
(
    [TeacherID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Teachers]  WITH CHECK ADD  CONSTRAINT [FK_Teachers_Users1] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Users] ([UserID])
GO

ALTER TABLE [dbo].[Teachers] CHECK CONSTRAINT [FK_Teachers_Users1]
GO
```

Tabela Users:

- tabela, w której znajdują się wszyscy użytkownicy i ich dane

```
CREATE TABLE [dbo].[Users](
    [UserID] [int] IDENTITY(1,1) NOT NULL,
    [Email] [nvarchar](320) NOT NULL,
    [Password] [nvarchar](50) NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [Surname] [nvarchar](50) NOT NULL,
    [CountryID] [int] NOT NULL,
    [CityID] [int] NOT NULL,
    [ZipCode] [nvarchar](50) NULL,
    [Street] [nvarchar](50) NOT NULL,
    [Address] [nvarchar](50) NOT NULL,
    [PhoneNumber] [nvarchar](50) NULL,
 CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
(
    [UserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Users]  WITH CHECK ADD  CONSTRAINT [FK_Users_CountryCity] FOREIGN KEY([CountryID], [CityID])
REFERENCES [dbo].[CountryCity] ([CountryID], [CityID])
GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK_Users_CountryCity]
GO

ALTER TABLE [dbo].[Users]  WITH CHECK ADD  CONSTRAINT [U_Names] CHECK  (([Name]<>'' AND [Surname]<>''))
GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [U_Names]
GO

ALTER TABLE [dbo].[Users]  WITH CHECK ADD  CONSTRAINT [U_NotEmpty] CHECK  (([Email]<>'' AND [Password]<>'' AND [ZipCode]<>'' AND [Street]<>'' AND [Address]<>'' AND [PhoneNumber]<>''))
GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [U_NotEmpty]
GO
```

Tabela Webinars:

- tabela zawiera informacje na temat wszystkich webinarów
- hyperlink: link do webinaru
- language: język, w którym są prowadzone webinary
- translatorName, translatorSurname: imię i nazwisko translatora

```
CREATE TABLE [dbo].[Webinars](
    [WebinarID] [int] IDENTITY(1,1) NOT NULL,
    [TeacherID] [int] NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [Price] [money] NOT NULL,
    [Hyperlink] [nvarchar](100) NOT NULL,
    [LanguageID] [int] NOT NULL,
    [TranslatorName] [nvarchar](50) NULL,
    [TranslatorSurname] [nvarchar](50) NULL,
    [StartDate] [datetime] NOT NULL,
    [Duration] [time](7) NOT NULL,
 CONSTRAINT [PK_Webinars] PRIMARY KEY CLUSTERED
(
    [WebinarID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [FK_Webinars_Languages] FOREIGN KEY([LanguageID])
REFERENCES [dbo].[Languages] ([LanguageID])
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [FK_Webinars_Languages]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [FK_Webinars_Teachers] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Teachers] ([TeacherID])
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [FK_Webinars_Teachers]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [W_Hyperlink] CHECK  (([Hyperlink]<>''))
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [W_Hyperlink]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [W_Name] CHECK  (([Name]<>''))
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [W_Name]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [W_Price] CHECK  (([Price]>(0)))
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [W_Price]
GO

ALTER TABLE [dbo].[Webinars]  WITH CHECK ADD  CONSTRAINT [W_Translator] CHECK  (([TranslatorName]<>'' AND [TranslatorSurname]<>''))
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [W_Translator]
GO
```

Widoki

Raporty dłużników

Dłużnikiem jest osoba, która dany produkt zamówiła, nie spełnia odpowiednich dla danego typu nauki terminów płatności, nie ma odroczenia płatności.

Raport dłużników Courses

```
CREATE VIEW [dbo].[n_1_CoursesDebtorReport]
AS
SELECT dbo.Courses.Name, dbo.Courses.CourseID, dbo.Students.StudentID, dbo.Users.Name AS Expr1, dbo.Users.Surname, dbo.OrderedCourses.HasBeenPaidFor, dbo.Courses.StartDate AS CourseStartDate,
dbo.OrderedCourses.PaymentDeferral, dbo.Courses.Price AS Money
FROM   dbo.Courses INNER JOIN
            dbo.OrderedCourses ON dbo.Courses.CourseID = dbo.OrderedCourses.CourseID INNER JOIN
            dbo.Orders ON dbo.OrderedCourses.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
WHERE (dbo.OrderedCourses.PaymentDeferral = 0) AND (DATEDIFF(day, GETDATE(), dbo.Courses.StartDate) <= 3) AND (dbo.Orders.Status = 'Delivered') AND (dbo.OrderedCourses.HasBeenPaidFor = 0)
```

	Name	CourseID	StudentID	Expr1	Surname	HasBeenPaidFor	CourseStartDate	PaymentDeferral	Money
1	Podstawy Analizy Danych	6	53	Berkeley	Manjin	0	2024-01-07 00:00:00.000	0	319,74
2	Jezyk Angielski dla Poczatkujacych	7	58	Osborn	Cartmale	0	2024-01-08 00:00:00.000	0	371,05
3	Marketing Cyfrowy	8	56	Padriac	Mowsley	0	2024-01-08 00:00:00.000	0	362,32
4	Marketing Cyfrowy	8	67	Laurette	Sjostrom	0	2024-01-08 00:00:00.000	0	362,32
5	Jezyk Angielski dla Poczatkujacych	7	60	Jillane	Kipping	0	2024-01-08 00:00:00.000	0	371,05
6	Podstawy Analizy Danych	6	55	Saundra	Pachmann	0	2024-01-07 00:00:00.000	0	319,74

Raport dłużników Studies

```
CREATE VIEW [dbo].[n_1_StudiesDebtorReport]
AS
SELECT dbo.Orders.StudentID, dbo.Users.Name, dbo.Users.Surname, dbo.Studies.EntryFee, dbo.OrderedStudies.EntryFeePaid, dbo.OrderedStudies.PaymentDeferral, dbo.Studies.AcademicYear
FROM   dbo.OrderedStudies INNER JOIN
            dbo.Orders ON dbo.OrderedStudies.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID INNER JOIN
            dbo.Studies ON dbo.OrderedStudies.StudyID = dbo.Studies.StudyID
WHERE (dbo.OrderedStudies.PaymentDeferral = 0) AND (dbo.OrderedStudies.EntryFeePaid = 0) AND (dbo.Orders.Status = 'Delivered')
```

	StudentID	Name	Surname	EntryFee	EntryFeePaid	PaymentDeferral	AcademicYear
1	54	Peggie	Abrahamsohn	40,90	0	0	2023
2	53	Berkeley	Manjin	40,90	0	0	2023
3	57	Abdul	O'Longain	40,90	0	0	2023

Raport dłużników StudyMeetings bez studium

```
CREATE VIEW [dbo].[n_1_MeetingsNoStudiesDebtorReport]
AS
SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.StudyMeetingID, dbo.Students.StudentID, dbo.Users.Name, dbo.Users.Surname, dbo.OrderedStudyMeetings.HasBeenPaidFor, dbo.StudyMeetings.BeginningDate AS
StudyMeetingsBeginningDate,
            dbo.OrderedStudyMeetings.PaymentDeferral, dbo.StudyMeetings.MeetingPrice AS Money
FROM   dbo.Orders INNER JOIN
            dbo.OrderedStudyMeetings ON dbo.Orders.OrderID = dbo.OrderedStudyMeetings.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID INNER JOIN
            dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
WHERE (dbo.OrderedStudyMeetings.IsPartOfStudies = 0) AND (dbo.OrderedStudyMeetings.PaymentDeferral = 0) AND (DATEDIFF(day, GETDATE(), dbo.StudyMeetings.BeginningDate) <= 3) AND (dbo.Orders.Status = 'Delivered') AND
(dbo.OrderedStudyMeetings.HasBeenPaidFor = 0)
```

	MeetingName	StudyMeetingID	StudentID	Name	Surname	HasBeenPaidFor	StudyMeetingsBeginningDate	PaymentDeferral	Money
1	Sesja Szkoleniowa	13	54	Peggie	Abrahamsohn	0	2023-11-25 00:00:00.000	0	14,05
2	Spotkanie Organizacyjne	14	55	Saundra	Pachmann	0	2023-02-08 00:00:00.000	0	19,68

Raport dłużników Webinars

```
CREATE VIEW [dbo].[n_1_WebinarsDebtorReport]
AS
SELECT dbo.Webinars.WebinarID, dbo.Webinars.Name, dbo.Students.StudentID, dbo.Users.Name AS StudentName, dbo.Users.Surname, dbo.OrderedWebinars.HasBeenPaidFor, dbo.Webinars.StartDate AS WebinarStartDate,
dbo.OrderedWebinars.PaymentDeferral,
            dbo.Webinars.Price AS Money
FROM   dbo.OrderedWebinars INNER JOIN
            dbo.Orders ON dbo.OrderedWebinars.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Webinars ON dbo.OrderedWebinars.WebinarID = dbo.Webinars.WebinarID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
WHERE (dbo.Webinars.StartDate < GETDATE()) AND (dbo.OrderedWebinars.PaymentDeferral = 0) AND (dbo.Orders.Status = 'Delivered') AND (dbo.OrderedWebinars.HasBeenPaidFor = 0)
```

	WebinarID	Name	StudentID	StudentName	Surname	HasBeenPaidFor	WebinarStartDate	PaymentDeferral	Money
1	2	Taktyka Czasu	57	Abdul	O'Longain	0	2023-01-10 01:20:14.000	0	94,68
2	19	Inteligencja Emocjonalna	60	Jillane	Kipping	0	2024-01-07 04:25:34.000	0	105,83
3	5	Dynamika Zespolu	55	Saundra	Pachmann	0	2023-02-23 20:14:37.000	0	112,76
4	21	Wizja Przyszlosci	59	Willy	Stickney	0	2023-09-25 06:57:29.000	0	141,61
5	2	Taktyka Czasu	69	Cass	Maxwaile	0	2023-01-10 01:20:14.000	0	94,68
6	18	Mentalnosc Sukcesu	58	Osborn	Cartmale	0	2024-01-07 04:55:17.000	0	93,34
7	19	Inteligencja Emocjonalna	58	Osborn	Cartmale	0	2024-01-07 04:25:34.000	0	105,83
8	12	Sztuczki Produktywnosci	57	Abdul	O'Longain	0	2023-07-01 16:09:57.000	0	102,79

Raport dłużników StudyMeetings ze studium

```
CREATE VIEW [dbo].[n_1_MeetingsStudiesDebtorReport]
AS
SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.StudyMeetingID, dbo.Students.StudentID, dbo.Users.Name, dbo.Users.Surname, dbo.OrderedStudyMeetings.HasBeenPaidFor, dbo.StudyMeetings.BeginningDate AS
StudyMeetingsBeginningDate,
            dbo.OrderedStudyMeetings.PaymentDeferral, dbo.StudyMeetings.MeetingPriceForStudents AS Money
FROM   dbo.Orders INNER JOIN
            dbo.OrderedStudyMeetings ON dbo.Orders.OrderID = dbo.OrderedStudyMeetings.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID INNER JOIN
            dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
WHERE (dbo.OrderedStudyMeetings.IsPartOfStudies = 1) AND (dbo.OrderedStudyMeetings.PaymentDeferral = 0) AND (DATEDIFF(day, GETDATE(), dbo.StudyMeetings.BeginningDate) <= 3) AND (dbo.Orders.Status = 'Delivered') AND
(dbo.OrderedStudyMeetings.HasBeenPaidFor = 0)
```

	MeetingName	StudyMeetingID	StudentID	Name	Surname	HasBeenPaidFor	StudyMeetingsBeginningDate	PaymentDeferral	Money
1	Warsztaty Rozwojowe	2	54	Peggie	Abrahamsohn	0	2023-01-19 00:00:00.000	0	10,00
2	Sesja Brainstormingu	3	57	Abdul	O'Longain	0	2023-12-07 00:00:00.000	0	6,00
3	Panel Dyskusyjny	6	57	Abdul	O'Longain	0	2023-03-24 00:00:00.000	0	7,00
4	Spotkanie Projektowe	9	57	Abdul	O'Longain	0	2023-01-11 00:00:00.000	0	5,00
5	Warsztaty Kreatywne	12	57	Abdul	O'Longain	0	2023-02-18 00:00:00.000	0	10,00
6	Konwent Badawczy	15	57	Abdul	O'Longain	0	2023-04-21 00:00:00.000	0	10,00
7	Forum Edukacyjne	18	57	Abdul	O'Longain	0	2023-12-07 00:00:00.000	0	5,00
8	Spotkanie Konsultacyjne	21	57	Abdul	O'Longain	0	2023-10-18 00:00:00.000	0	9,00
9	Sesja Inspiracyjna	24	57	Abdul	O'Longain	0	2023-02-24 00:00:00.000	0	8,00

Raporty zapisanych.

To, że ktoś jest zapisany na dany typ spotkania oznacza, że zamówił go i jego status to 'Delivered'.

Raport zapisanych osób na CoursesModules

```
CREATE VIEW [dbo].[n_1_CoursesModulesPeopleCount]
AS
SELECT dbo.CoursesModules.Type, COUNT(dbo.OrderedCourses.CourseID) AS [Liczba osob], dbo.CoursesModules.ModuleName, dbo.CoursesModules.BeginningDate
FROM   dbo.Orders INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
            dbo.OrderedCourses ON dbo.Orders.OrderID = dbo.OrderedCourses.OrderID INNER JOIN
            dbo.Courses ON dbo.OrderedCourses.CourseID = dbo.Courses.CourseID INNER JOIN
            dbo.CoursesModules ON dbo.Courses.CourseID = dbo.CoursesModules.CourseID
WHERE (dbo.Orders.Status = 'Delivered')
GROUP BY dbo.OrderedCourses.CourseID, dbo.CoursesModules.Type, dbo.CoursesModules.ModuleName, dbo.CoursesModules.BeginningDate
```

	Type	Liczba osob	ModuleName	BeginningDate
1	HY	1	Health Care	2023-10-28 00:00:00.000
2	OS	1	Capital Goods	2024-02-07 00:00:00.000
3	OS	1	n/a	2024-02-20 00:00:00.000
4	ST	1	Consumer Services	2023-10-08 00:00:00.000
5	HY	3	Consumer Non-Durables	2023-10-20 00:00:00.000
6	OA	3	Finance	2023-12-30 00:00:00.000
7	OA	3	Math	2024-01-20 00:00:00.000
8	ST	3	Consumer Services	2023-10-13 00:00:00.000
9	ST	3	Finance	2024-01-25 00:00:00.000
10	OA	4	Finance	2024-02-11 00:00:00.000
11	OA	4	Physics	2024-01-20 00:00:00.000
12	OA	4	Transportation	2023-12-23 00:00:00.000
13	OS	4	Health Care	2024-03-20 00:00:00.000
14	ST	4	n/a	2024-01-15 00:00:00.000
15	HY	3	Finance	2023-09-20 00:00:00.000
16	OS	3	Consumer Durables	2024-02-15 00:00:00.000
17	ST	3	Basic Industries	2023-11-19 00:00:00.000
18	ST	3	Health Care	2024-01-24 00:00:00.000
19	HY	5	Finance	2024-01-20 00:00:00.000
20	OA	5	Finance	2023-09-30 00:00:00.000
21	OS	5	n/a	2023-10-27 00:00:00.000
22	ST	5	Miscellaneous	2023-09-03 00:00:00.000
23	HY	1	Health Care	2024-02-13 00:00:00.000
24	OA	1	Energy	2023-12-11 00:00:00.000
25	OS	1	Finance	2023-09-08 00:00:00.000
26	ST	1	Finance	2023-11-23 00:00:00.000
27	HY	1	Consumer Services	2023-11-28 00:00:00.000

Raport zapisanych osób na Meetings

```
CREATE VIEW [dbo].[n_1_StudyMeetingsPeopleCount]
AS
SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.Type, COUNT(dbo.OrderedStudyMeetings.StudyMeetingID) AS [Liczba osob], dbo.StudyMeetings.BeginningDate
FROM   dbo.OrderedStudyMeetings INNER JOIN
            dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID INNER JOIN
            dbo.Orders ON dbo.OrderedStudyMeetings.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID
WHERE (dbo.Orders.Status = 'Delivered')
GROUP BY dbo.OrderedStudyMeetings.StudyMeetingID, dbo.StudyMeetings.Type, dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.BeginningDate
```

	MeetingName	Type	Liczba osob	BeginningDate
1	Spotkanie Integracyjne	OA	1	2023-06-10 00:00:00.000
2	Warsztaty Rozwojowe	HY	1	2023-01-19 00:00:00.000
3	Sesja Brainstormingu	ST	1	2023-12-07 00:00:00.000
4	Debata Akademicka	ST	4	2023-07-04 00:00:00.000
5	Zjazd Naukowy	HY	1	2023-02-08 00:00:00.000
6	Panel Dyskusyjny	OS	1	2023-03-24 00:00:00.000
7	Konferencja Metodyczna	HY	1	2023-09-01 00:00:00.000
8	Symposium Wiedzy	OA	1	2023-05-11 00:00:00.000
9	Spotkanie Projektowe	ST	1	2023-01-11 00:00:00.000
10	Forum Innowacji	ST	1	2023-02-06 00:00:00.000
11	Dzień Otwarty	HY	2	2023-06-09 00:00:00.000
12	Warsztaty Kreatywne	ST	1	2023-02-18 00:00:00.000
13	Sesja Szkoleniowa	ST	2	2023-11-25 00:00:00.000

Raport zapisanych osób na Webinars

```
CREATE VIEW [dbo].[n_1_WebinarsPeopleCount]
AS
SELECT dbo.Webinars.Name, COUNT(dbo.OrderedWebinars.WebinarID) AS [Liczba osob], 'zdalnie' AS tryb, dbo.Webinars.StartDate
FROM   dbo.OrderedWebinars INNER JOIN
            dbo.Orders ON dbo.OrderedWebinars.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Webinars ON dbo.OrderedWebinars.WebinarID = dbo.Webinars.WebinarID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID
WHERE (dbo.Orders.Status = 'Delivered')
GROUP BY dbo.Webinars.Name, dbo.OrderedWebinars.WebinarID, dbo.Webinars.StartDate
```

	Name	Liczba osob	tryb	StartDate
1	Moc Rozwoju	2	zdalnie	2023-09-18 03:25:16.000
2	Taktyka Czasu	3	zdalnie	2023-01-10 01:20:14.000
3	Iskra Kreatywnosci	1	zdalnie	2023-08-24 16:23:39.000
4	Klucze Perswazji	1	zdalnie	2023-12-27 09:50:05.000
5	Dynamika Zespolu	2	zdalnie	2023-02-23 20:14:37.000
6	Mistrzostwo Celów	1	zdalnie	2023-05-13 20:46:41.000
7	Rzemioslo Pewnosci	1	zdalnie	2023-04-01 14:06:27.000
8	Harmonia Zycia	1	zdalnie	2023-02-25 19:51:42.000
9	Innowacyjny Umysl	1	zdalnie	2023-06-24 13:02:46.000
10	Cyfrowa Magia	1	zdalnie	2023-03-02 13:01:19.000
11	Swiadome Strategie	1	zdalnie	2023-10-17 02:09:18.000
12	Sztuczki Produktywnosci	3	zdalnie	2023-07-01 16:09:57.000
13	Skok Kariery	1	zdalnie	2023-01-06 23:36:23.000
14	Krawedz Przywództwa	2	zdalnie	2023-01-03 15:19:16.000
15	Sieciowanie Pro	3	zdalnie	2023-02-05 03:00:26.000
16	Mentalnosc Sukcesu	1	zdalnie	2024-01-07 04:55:17.000
17	Inteligencja Emocjonalna	2	zdalnie	2024-01-07 04:25:34.000
18	Wizja Przyszlosci	3	zdalnie	2023-09-25 06:57:29.000
19	Funkcja czasu	2	zdalnie	2023-09-25 07:00:00.000

Raporty Finansowe

Raporty są tworzone w następujące sposób, patrzymy do odpowiadających tabel ordered. Następnie łącząc z tabelą odpowiadającą typowi nauczania, grupujemy po ID i podajemy kwoty. Dodatkowo sprawdzamy czy produkt został już zamówiony (nie jest w koszyku) i został opłacony. Każdy raport uwzględnia wyniki finansowe z ostatniego roku.

Raport finansowy Courses

```
CREATE VIEW [dbo].[n_1_CoursesFinancialReport]
AS
WITH t1 AS (SELECT dbo.Courses.CourseID, COUNT(*) * dbo.Courses.Price AS moneyMade, dbo.Courses.Price, COUNT(*) AS quantity
                   FROM    dbo.Orders INNER JOIN
                                dbo.OrderedCourses ON dbo.Orders.OrderID = dbo.OrderedCourses.OrderID RIGHT OUTER JOIN
                                dbo.Courses ON dbo.OrderedCourses.CourseID = dbo.Courses.CourseID
                   WHERE (dbo.Orders.Status = 'Delivered') AND (dbo.OrderedCourses.HasBeenPaidFor = 1) AND (DATEDIFF(DAY, dbo.OrderedCourses.PaymentDate, GETDATE()) <= 365)
                   GROUP BY dbo.Courses.CourseID, dbo.Courses.Price)
    SELECT Courses_1.CourseID, Courses_1.Name, Courses_1.Price AS priceForCourse, ISNULL(t1_1.quantity, 0) AS quantity, ISNULL(t1_1.moneyMade, 0) AS CoursesIncome
   FROM    dbo.Courses AS Courses_1 LEFT OUTER JOIN
             t1 AS t1_1 ON t1_1.CourseID = Courses_1.CourseID
```

	WebinarID	Name	priceforwebinar	quantity	webinarsIncome
1	1	Moc Rozwoju	89,80	2	179,60
2	2	Taktyka Czasu	94,68	1	94,68
3	3	Iskra Kreatywnosci	94,70	1	94,70
4	4	Klucze Perswazji	140,55	1	140,55
5	5	Dynamika Zespolu	112,76	1	112,76
6	6	Mistrzostwo Celów	49,43	1	49,43
7	7	Rzemioslo Pewnosci	119,36	1	119,36
8	8	Harmonia Zycia	62,73	1	62,73
9	9	Innowacyjny Umysl	83,48	1	83,48
10	10	Cyfrowa Magia	83,31	1	83,31
11	11	Swiadome Strategie	127,66	1	127,66
12	12	Sztuczki Produktywnosci	102,79	1	102,79
13	13	Skok Kariery	126,48	1	126,48
14	14	Krawedz Przywództwa	40,06	1	40,06
15	15	Sieciowanie Pro	70,12	3	210,36
16	16	Ty jako Marka	96,86	0	0,00
17	17	Finansowa Sprawnosc	140,35	0	0,00
18	18	Mentalnosc Sukcesu	93,34	0	0,00
19	19	Inteligencja Emocjonalna	105,83	0	0,00
20	20	Madre Decyzje	110,82	0	0,00
21	21	Wizja Przyszlosci	141,61	0	0,00
22	28	Funkcja czasu	150,00	0	0,00

Raport finansowy Studies

```
CREATE view [dbo].[n_1_StudiesFinancialReport] as
    WITH t1 AS (SELECT dbo.StudyMeetings.StudyMeetingID,
                       COUNT(*) * dbo.StudyMeetings.MeetingPriceForStudents AS zarobionasuma
                FROM dbo.Orders
                         INNER JOIN
                     dbo.OrderedStudyMeetings ON dbo.Orders.OrderID = dbo.OrderedStudyMeetings.OrderID
                        RIGHT OUTER JOIN
                     dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID
                WHERE (dbo.Orders.Status = 'Delivered')
                  AND (dbo.OrderedStudyMeetings.HasBeenPaidFor = 1)
                  AND (dbo.OrderedStudyMeetings.IsPartOfStudies = 1)
                  AND (DATEDIFF(DAY, PaymentDate, GETDATE()) <= 365)
                GROUP BY dbo.StudyMeetings.StudyMeetingID, dbo.StudyMeetings.MeetingPriceForStudents),
         t2 AS
             (SELECT StudyMeetings_1.StudyID        AS idstudiow,
                     StudyMeetings_1.StudyMeetingID AS idspotkania,
                     ISNULL(t1_1.zarobionasuma, 0)  AS zarobionasuma
              FROM dbo.StudyMeetings AS StudyMeetings_1
                       LEFT OUTER JOIN
                   t1 AS t1_1 ON t1_1.StudyMeetingID = StudyMeetings_1.StudyMeetingID),
         t3 AS
             (SELECT idstudiow, SUM(zarobionasuma) AS zarobionasuma
              FROM t2 AS t2_1
              GROUP BY idstudiow),
         t4 AS
             (SELECT dbo.OrderedStudies.StudyID AS idstudiow, COUNT(dbo.OrderedStudies.EntryFeePaid) AS liczbaoplacenfee
              FROM dbo.OrderedStudies
                       INNER JOIN
                   dbo.Orders AS Orders_1 ON dbo.OrderedStudies.OrderID = Orders_1.OrderID
              WHERE (Orders_1.Status = 'Delivered')
                AND (dbo.OrderedStudies.EntryFeePaid = 1)
                AND (DATEDIFF(DAY, PaymentDate, GETDATE()) <= 365)
              GROUP BY dbo.OrderedStudies.StudyID),
         t5 AS
             (SELECT dbo.Studies.StudyID,
                     Studies.FieldOfStudy                                    as name,
                     ISNULL(t4_1.liczbaoplacenfee * dbo.Studies.EntryFee, 0) AS zarobekzentryfee
              FROM dbo.Studies
                       LEFT OUTER JOIN
                   t4 AS t4_1 ON dbo.Studies.StudyID = t4_1.idstudiow)
    SELECT t5_1.StudyID,
           t5_1.name,
           t3_1.zarobionasuma                        as meetingsMoney,
           t5_1.zarobekzentryfee                     as entryFeeMoney,
           t3_1.zarobionasuma + t5_1.zarobekzentryfee AS studySum
    FROM t5 AS t5_1
             LEFT OUTER JOIN
         t3 AS t3_1 ON t5_1.StudyID = t3_1.idstudiow
```

	StudyID	name	meetingsMoney	entryFeeMoney	studySum
1	11	Computer Science	15,00	40,90	55,90
2	12	Electronics	43,00	56,19	99,19
3	13	Cybersecurity	8,00	0,00	8,00

Raport finansowy StudyMeetings poza studium

```
CREATE VIEW [dbo].[n_1_MeetingsNoStudiesFinancialReport]
AS
WITH t1 AS (SELECT dbo.StudyMeetings.StudyMeetingID, COUNT(*) * dbo.StudyMeetings.MeetingPrice AS moneyMade, COUNT(*) AS quantity
                   FROM    dbo.Orders INNER JOIN
                                dbo.OrderedStudyMeetings ON dbo.Orders.OrderID = dbo.OrderedStudyMeetings.OrderID RIGHT OUTER JOIN
                                dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID
                   WHERE (dbo.Orders.Status = 'Delivered') AND (dbo.OrderedStudyMeetings.HasBeenPaidFor = 1) AND (dbo.OrderedStudyMeetings.IsPartOfStudies = 0) AND (DATEDIFF(DAY, dbo.OrderedStudyMeetings.PaymentDate,
GETDATE()) <= 365)
                   GROUP BY dbo.StudyMeetings.StudyMeetingID, dbo.StudyMeetings.MeetingPrice)
    SELECT StudyMeetings_1.StudyMeetingID, StudyMeetings_1.MeetingName, StudyMeetings_1.MeetingPrice AS priceForMeeting, ISNULL(t1_1.quantity, 0) AS quantity, ISNULL(t1_1.moneyMade, 0) AS StudyMeetingsIncome
   FROM    dbo.StudyMeetings AS StudyMeetings_1 LEFT OUTER JOIN
             t1 AS t1_1 ON t1_1.StudyMeetingID = StudyMeetings_1.StudyMeetingID
```

	StudyMeetingID	MeetingName	priceForMeeting	quantity	StudyMeetingsIncom
1	1	Spotkanie Integracyjne	13,61	1	13,61
2	2	Warsztaty Rozwojowe	11,22	0	0,00
3	3	Sesja Brainstormingu	18,85	1	18,85
4	4	Debata Akademicka	11,51	3	34,53
5	5	Zjazd Naukowy	11,42	0	0,00
6	6	Panel Dyskusyjny	16,72	0	0,00
7	7	Konferencja Metodyczna	13,90	0	0,00
8	8	Symposium Wiedzy	18,44	1	18,44
9	9	Spotkanie Projektowe	13,73	0	0,00
10	10	Forum Innowacji	19,52	0	0,00
11	11	Dzień Otwarty	18,56	2	37,12
12	12	Warsztaty Kreatywne	11,52	0	0,00
13	13	Sesja Szkoleniowa	14,05	0	0,00
14	14	Spotkanie Organizacyjne	19,68	0	0,00
15	15	Konwent Badawczy	14,81	0	0,00
16	16	Sesja Networkingowa	15,94	0	0,00
17	17	Warsztaty Specjalistyczne	11,14	0	0,00
18	18	Forum Edukacyjne	17,46	0	0,00
19	19	Zgromadzenie Naukowe	18,30	0	0,00
20	20	Konferencja Interdyscypl...	14,05	0	0,00

Raport finansowy Webinars

```
CREATE VIEW [dbo].[n_1_WebinarsFinancialReport]
AS
WITH t1 AS (SELECT dbo.Webinars.WebinarID, COUNT(*) * dbo.Webinars.Price AS moneyMade, dbo.Webinars.Price AS price, COUNT(*) AS quantity
                   FROM    dbo.Orders INNER JOIN
                                dbo.OrderedWebinars ON dbo.Orders.OrderID = dbo.OrderedWebinars.OrderID RIGHT OUTER JOIN
                                dbo.Webinars ON dbo.OrderedWebinars.WebinarID = dbo.Webinars.WebinarID
                   WHERE (dbo.Orders.Status = 'Delivered') AND (dbo.OrderedWebinars.HasBeenPaidFor = 1) AND (DATEDIFF(DAY, dbo.OrderedWebinars.PaymentDate, GETDATE()) <= 365)
                   GROUP BY dbo.Webinars.WebinarID, dbo.Webinars.Price)
    SELECT TOP (100) PERCENT Webinars_1.WebinarID, Webinars_1.Name, Webinars_1.Price AS priceforwebinar, ISNULL(t1_1.quantity, 0) AS quantity, ISNULL(t1_1.moneyMade, 0) AS webinarsIncome
   FROM    dbo.Webinars AS Webinars_1 LEFT OUTER JOIN
             t1 AS t1_1 ON t1_1.WebinarID = Webinars_1.WebinarID
```

	WebinarID	Name	priceforwebinar	quantity	webinarsIncome
1	1	Moc Rozwoju	89,80	2	179,60
2	2	Taktyka Czasu	94,68	1	94,68
3	3	Iskra Kreatywnosci	94,70	1	94,70
4	4	Klucze Perswazji	140,55	1	140,55
5	5	Dynamika Zespolu	112,76	1	112,76
6	6	Mistrzostwo Celów	49,43	1	49,43
7	7	Rzemioslo Pewnosci	119,36	1	119,36
8	8	Harmonia Zycia	62,73	1	62,73
9	9	Innowacyjny Umysl	83,48	1	83,48
10	10	Cyfrowa Magia	83,31	1	83,31
11	11	Swiadome Strategie	127,66	1	127,66
12	12	Sztuczki Produktywnosci	102,79	1	102,79
13	13	Skok Kariery	126,48	1	126,48
14	14	Krawedz Przywództwa	40,06	1	40,06
15	15	Sieciowanie Pro	70,12	3	210,36
16	16	Ty jako Marka	96,86	0	0,00
17	17	Finansowa Sprawnosc	140,35	0	0,00
18	18	Mentalnosc Sukcesu	93,34	0	0,00
19	19	Inteligencja Emocjonalna	105,83	0	0,00
20	20	Madre Decyzje	110,82	0	0,00
21	21	Wizja Przyszlosci	141,61	0	0,00
22	28	Funkcja czasu	150,00	0	0,00

Raporty frekwencji

Raporty frekwencji powstają w taki sposób, że od osób zapisanych na dany typ spotkania odejmujemy liczbę osób nieobecnych.

Raport frekwencji Meetings

```
CREATE VIEW [dbo].[n_1_MeetingsPresencesReport]
AS
WITH t2 AS (SELECT StudyMeetingID, COUNT(StudentID) AS absencje
                   FROM    dbo.StudyMeetingsAbsences
                   GROUP BY StudyMeetingID), t1 AS
    (SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.Type, COUNT(dbo.OrderedStudyMeetings.StudyMeetingID) AS [Liczba osob], dbo.StudyMeetings.StudyMeetingID, dbo.StudyMeetings.BeginningDate
    FROM    dbo.OrderedStudyMeetings INNER JOIN
                 dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID INNER JOIN
                 dbo.Orders ON dbo.OrderedStudyMeetings.OrderID = dbo.Orders.OrderID INNER JOIN
                 dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID
    WHERE (dbo.Orders.Status = 'Delivered')
    GROUP BY dbo.OrderedStudyMeetings.StudyMeetingID, dbo.StudyMeetings.Type, dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.StudyMeetingID, dbo.StudyMeetings.BeginningDate)
    SELECT t1_1.Type, t1_1.StudyMeetingID, t1_1.MeetingName, t1_1.[Liczba osob] - ISNULL(t2_1.absencje, 0) AS presenceCount, t1_1.BeginningDate
   FROM    t1 AS t1_1 LEFT OUTER JOIN
             t2 AS t2_1 ON t2_1.StudyMeetingID = t1_1.StudyMeetingID
```

	Type	StudyMeetingID	MeetingName	presenceCount	BeginningDate
1	OA	1	Spotkanie Integracyjne	1	2023-06-10 00:00:00.000
2	HY	2	Warsztaty Rozwojowe	1	2023-01-19 00:00:00.000
3	ST	3	Sesja Brainstormingu	2	2023-12-07 00:00:00.000
4	ST	4	Debata Akademicka	4	2023-07-04 00:00:00.000
5	HY	5	Zjazd Naukowy	1	2023-02-08 00:00:00.000
6	OS	6	Panel Dyskusyjny	2	2023-03-24 00:00:00.000
7	HY	7	Konferencja Metodyczna	1	2023-09-01 00:00:00.000
8	OA	8	Symposium Wiedzy	1	2023-05-11 00:00:00.000
9	ST	9	Spotkanie Projektowe	1	2023-01-11 00:00:00.000
10	ST	10	Forum Innowacji	1	2023-02-06 00:00:00.000
11	HY	11	Dzień Otwarty	2	2023-06-09 00:00:00.000
12	ST	12	Warsztaty Kreatywne	2	2023-02-18 00:00:00.000
13	ST	13	Sesja Szkoleniowa	2	2023-11-25 00:00:00.000
14	OA	14	Spotkanie Organizacyjne	1	2023-02-08 00:00:00.000
15	ST	15	Konwent Badawczy	1	2023-04-21 00:00:00.000
16	HY	18	Forum Edukacyjne	1	2023-12-07 00:00:00.000
17	HY	21	Spotkanie Konsultacyjne	1	2023-10-18 00:00:00.000
18	HY	24	Sesja Inspiracyjna	1	2023-02-24 00:00:00.000
19	Zdalne	28	triggertest	1	2024-06-10 00:00:00.000
20	HY	30	ExampleName	1	2024-06-10 00:00:00.000

Raport frekwencji Modules

```
CREATE VIEW [dbo].[n_1_ModulesPresencesReport]
AS
WITH t2 AS (SELECT ModuleID, COUNT(StudentID) AS absencje
                   FROM    dbo.ModulesAbsences
                   GROUP BY ModuleID), t1 AS
    (SELECT dbo.CoursesModules.Type, COUNT(dbo.OrderedCourses.CourseID) AS [Liczba osob], dbo.CoursesModules.ModuleName, dbo.CoursesModules.ModuleID, dbo.CoursesModules.BeginningDate
    FROM    dbo.Orders INNER JOIN
                 dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
                 dbo.OrderedCourses ON dbo.Orders.OrderID = dbo.OrderedCourses.OrderID INNER JOIN
                 dbo.Courses ON dbo.OrderedCourses.CourseID = dbo.Courses.CourseID INNER JOIN
                 dbo.CoursesModules ON dbo.Courses.CourseID = dbo.CoursesModules.CourseID
    WHERE (dbo.Orders.Status = 'Delivered')
    GROUP BY dbo.OrderedCourses.CourseID, dbo.CoursesModules.Type, dbo.CoursesModules.ModuleName, dbo.CoursesModules.ModuleID, dbo.CoursesModules.BeginningDate)
    SELECT t1_1.Type, t1_1.ModuleID, t1_1.ModuleName, t1_1.[Liczba osob] - ISNULL(t2_1.absencje, 0) AS presenceCount, t1_1.BeginningDate
   FROM    t1 AS t1_1 LEFT OUTER JOIN
                t2 AS t2_1 ON t2_1.ModuleID = t1_1.ModuleID
```

	Type	ModuleID	ModuleName	presenceCount	BeginningDate
1	OA	2	Math	2	2024-01-20 00:00:00.000
2	OA	6	Physics	4	2024-01-20 00:00:00.000
3	ST	7	Consumer Services	1	2023-10-08 00:00:00.000
4	ST	8	Finance	3	2024-01-25 00:00:00.000
5	OA	9	Finance	4	2024-02-11 00:00:00.000
6	OS	10	Consumer Durables	3	2024-02-15 00:00:00.000
7	OS	11	n/a	5	2023-10-27 00:00:00.000
8	OS	12	Finance	1	2023-09-08 00:00:00.000
9	HY	13	Consumer Services	1	2023-11-28 00:00:00.000
10	OA	14	Finance	1	2024-03-09 00:00:00.000
11	OS	15	Health Care	1	2023-12-05 00:00:00.000
12	OA	16	Finance	1	2023-09-07 00:00:00.000
13	ST	17	n/a	1	2023-09-16 00:00:00.000
14	OS	18	Finance	1	2023-09-01 00:00:00.000
15	OS	22	Capital Goods	1	2024-02-07 00:00:00.000
16	OA	23	Finance	3	2023-12-30 00:00:00.000
17	OA	24	Transportation	4	2023-12-23 00:00:00.000
18	HY	25	Finance	3	2023-09-20 00:00:00.000
19	OA	26	Finance	5	2023-09-30 00:00:00.000
20	ST	27	Finance	1	2023-11-23 00:00:00.000

Szczegółowa frekwencja na Meetings

```
CREATE VIEW [dbo].[n_1_MeetingsDetailsPresences]
AS
WITH t1 AS (SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.Type, dbo.StudyMeetings.BeginningDate, dbo.Students.StudentID, dbo.Users.Name, dbo.Users.Surname
                   FROM    dbo.OrderedStudyMeetings INNER JOIN
                                dbo.StudyMeetings ON dbo.OrderedStudyMeetings.StudyMeetingID = dbo.StudyMeetings.StudyMeetingID INNER JOIN
                                dbo.Orders ON dbo.OrderedStudyMeetings.OrderID = dbo.Orders.OrderID INNER JOIN
                                dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
                                dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
                   WHERE (dbo.StudyMeetings.BeginningDate < GETDATE()) AND (dbo.Orders.Status = 'Delivered'))
    SELECT t1_1.Name, t1_1.Surname, t1_1.StudentID, t1_1.BeginningDate, t1_1.MeetingName, CASE WHEN StudyMeetingID IS NULL THEN 1 ELSE 0 END AS obecność
   FROM    dbo.StudyMeetingsAbsences RIGHT OUTER JOIN
                t1 AS t1_1 ON t1_1.StudentID = dbo.StudyMeetingsAbsences.StudentID
```

	Name	Surname	StudentID	BeginningDate	MeetingName	obecność
1	Kathryn	MacNeil	52	2023-06-10 00:00:00.000	Spotkanie Integracyjne	1
2	Peggie	Abrahamsohn	54	2023-01-19 00:00:00.000	Warsztaty Rozwojowe	1
3	Saundra	Pachmann	55	2023-12-07 00:00:00.000	Sesja Brainstormingu	1
4	Saundra	Pachmann	55	2023-07-04 00:00:00.000	Debata Akademicka	1
5	Willy	Stickney	59	2023-07-04 00:00:00.000	Debata Akademicka	1
6	Jackie	Goncalo	62	2023-07-04 00:00:00.000	Debata Akademicka	1
7	Tracee	Calvert	70	2023-07-04 00:00:00.000	Debata Akademicka	1
8	Lilah	Worham	61	2023-02-08 00:00:00.000	Zjazd Naukowy	1
9	Jillane	Kipping	60	2023-03-24 00:00:00.000	Panel Dyskusyjny	1
10	Marillin	Goodfield	65	2023-09-01 00:00:00.000	Konferencja Metodyczna	1
11	Jackie	Goncalo	62	2023-05-11 00:00:00.000	Symposium Wiedzy	1
12	Lynn	Orbell	63	2023-01-11 00:00:00.000	Spotkanie Projektowe	0
13	Carmel	Willers	66	2023-02-06 00:00:00.000	Forum Innowacji	1
14	Lynn	Orbell	63	2023-06-09 00:00:00.000	Dzień Otwarty	0
15	Tracee	Calvert	70	2023-06-09 00:00:00.000	Dzień Otwarty	1
16	Cass	Maxwaile	69	2023-02-18 00:00:00.000	Warsztaty Kreatywne	1
17	Osborn	Cartmale	58	2023-11-25 00:00:00.000	Sesja Szkoleniowa	0
18	Peggie	Abrahamsohn	54	2023-11-25 00:00:00.000	Sesja Szkoleniowa	1

Szczegółowa frekwencja na Modules

```
CREATE VIEW [dbo].[n_1_ModulesDetailsPresences]
AS
WITH t1 AS (SELECT dbo.CoursesModules.ModuleName, dbo.CoursesModules.Type, dbo.CoursesModules.BeginningDate, dbo.Students.StudentID, dbo.Users.Name, dbo.Users.Surname
                   FROM    dbo.Courses INNER JOIN
                                dbo.CoursesModules ON dbo.Courses.CourseID = dbo.CoursesModules.CourseID INNER JOIN
                                dbo.OrderedCourses ON dbo.Courses.CourseID = dbo.OrderedCourses.CourseID INNER JOIN
                                dbo.Orders ON dbo.OrderedCourses.OrderID = dbo.Orders.OrderID INNER JOIN
                                dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID INNER JOIN
                                dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
                   WHERE (dbo.Orders.Status = 'Delivered'))
    SELECT t1_1.Name, t1_1.Surname, t1_1.StudentID, t1_1.BeginningDate, t1_1.ModuleName, CASE WHEN ModulesAbsences.ModuleID IS NULL THEN 1 ELSE 0 END AS obecność
   FROM    dbo.ModulesAbsences RIGHT OUTER JOIN
                t1 AS t1_1 ON t1_1.StudentID = dbo.ModulesAbsences.StudentID
```

	Name	Surname	StudentID	BeginningDate	ModuleName	obecność
1	Peggie	Abrahamsohn	54	2024-01-20 00:00:00.000	Math	0
2	Jillane	Kipping	60	2024-01-20 00:00:00.000	Math	1
3	Saundra	Pachmann	55	2024-01-20 00:00:00.000	Math	1
4	Berkeley	Manjin	53	2024-01-20 00:00:00.000	Physics	1
5	Berkeley	Manjin	53	2024-01-20 00:00:00.000	Physics	1
6	Abdul	O'Longain	57	2024-01-20 00:00:00.000	Physics	1
7	Saundra	Pachmann	55	2024-01-20 00:00:00.000	Physics	1
8	Lilah	Worham	61	2023-10-08 00:00:00.000	Consumer Services	1
9	Peggie	Abrahamsohn	54	2024-01-25 00:00:00.000	Finance	0
10	Jillane	Kipping	60	2024-01-25 00:00:00.000	Finance	1
11	Saundra	Pachmann	55	2024-01-25 00:00:00.000	Finance	1
12	Berkeley	Manjin	53	2024-02-11 00:00:00.000	Finance	1
13	Berkeley	Manjin	53	2024-02-11 00:00:00.000	Finance	1
14	Abdul	O'Longain	57	2024-02-11 00:00:00.000	Finance	1
15	Saundra	Pachmann	55	2024-02-11 00:00:00.000	Finance	1
16	Osborn	Cartmale	58	2024-02-15 00:00:00.000	Consumer Durables	1
17	Willy	Stickney	59	2024-02-15 00:00:00.000	Consumer Durables	1
18	Jillane	Kipping	60	2024-02-15 00:00:00.000	Consumer Durables	1
19	Saundra	Pachmann	55	2023-10-27 00:00:00.000	n/a	1
20	Padriac	Mowsley	56	2023-10-27 00:00:00.000	n/a	1

Raporty bilokacji

Raporty pokazują jeżeli jakiejś osobie pokrywają się spotkania.

Raport bilokacji Webinary

```
CREATE VIEW [dbo].[n_1_WebinarsBilocation]
AS
SELECT aw.WebinarID AS firstWebinarID, bw.WebinarID AS secondWebinarID, aw.Name AS firstWebinarName, bw.Name AS secondWebinarName, dbo.Students.StudentID, dbo.Users.Name AS UserName, dbo.Users.Surname, aw.StartDate AS
firstStartDate, aw.Duration AS firstDuration,
             bw.StartDate AS secondStartDate, bw.Duration AS secondDuration
FROM   dbo.Webinars AS aw INNER JOIN
             dbo.OrderedWebinars AS a ON aw.WebinarID = a.WebinarID INNER JOIN
             dbo.OrderedWebinars AS b ON a.OrderID = b.OrderID INNER JOIN
             dbo.Webinars AS bw ON b.WebinarID = bw.WebinarID AND aw.WebinarID < bw.WebinarID AND DATEADD(SECOND, DATEDIFF(SECOND, '00:00:00', aw.Duration), aw.StartDate) > bw.StartDate AND aw.StartDate < DATEADD(SECOND,
DATEDIFF(SECOND, '00:00:00', bw.Duration),
             bw.StartDate) INNER JOIN
             dbo.Orders ON a.OrderID = dbo.Orders.OrderID INNER JOIN
             dbo.Students ON dbo.Students.StudentID = dbo.Orders.StudentID INNER JOIN
             dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
```

	firstWebinarID	secondWebinarID	firstWebinarName	secondWebinarName	StudentID	UserName	Surname	firstStartDate	firstDuration	secondStartDate	secondDuration
1	18	19	Mentalnosc Sukcesu	Inteligencja Emocjonalna	58	Osborn	Cartmale	2024-01-07 04:55:17.000	02:00:00.0000000	2024-01-07 04:25:34.000	02:00:00.0000000
2	21	28	Wizja Przyszlosci	Funkcja czasu	54	Peggie	Abrahamsohn	2023-09-25 06:57:29.000	01:30:00.0000000	2023-09-25 07:00:00.000	01:30:00.0000000

Raport bilokacji StudyMeetings

```
CREATE VIEW [dbo].[n_1_StudyMeetingsBilocations]
AS
SELECT asm.StudyMeetingID AS firstStudyMeetingID, bsm.StudyMeetingID AS secondStudyMeetingID, asm.MeetingName AS firstMeetingName, bsm.MeetingName AS secondMeetingName, dbo.Students.StudentID, dbo.Users.Name AS UserName,
dbo.Users.Surname,
             asm.BeginningDate AS firstBeginningDate, asm.Duration AS firstDuration, bsm.BeginningDate AS secondBeginningDate, bsm.Duration AS secondDuration
FROM   dbo.StudyMeetings AS asm INNER JOIN
             dbo.OrderedStudyMeetings AS a ON asm.StudyMeetingID = a.StudyMeetingID INNER JOIN
             dbo.OrderedStudyMeetings AS b ON a.OrderID = b.OrderID INNER JOIN
             dbo.StudyMeetings AS bsm ON b.StudyMeetingID = bsm.StudyMeetingID AND asm.StudyMeetingID < bsm.StudyMeetingID AND DATEADD(SECOND, DATEDIFF(SECOND, '00:00:00', asm.Duration), asm.BeginningDate) >
bsm.BeginningDate AND
             asm.BeginningDate < DATEADD(SECOND, DATEDIFF(SECOND, '00:00:00', bsm.Duration), bsm.BeginningDate) INNER JOIN
             dbo.Orders ON a.OrderID = dbo.Orders.OrderID INNER JOIN
             dbo.Students ON dbo.Students.StudentID = dbo.Orders.StudentID INNER JOIN
             dbo.Users ON dbo.Students.StudentID = dbo.Users.UserID
```

	firstStudyMeetingID	secondStudyMeetingID	firstMeetingName	secondMeetingName	StudentID	UserName	Surname	firstBeginningDate	firstDuration	secondBeginningDate	secondDuration
1	3	18	Sesja Brainstormingu	Forum Edukacyjne	57	Abdul	O'Longain	2023-12-07 00:00:00.000	01:30:00.0000000	2023-12-07 00:00:00.000	01:30:00.0000000
2	28	30	triggertest	ExampleName	55	Saundra	Pachmann	2024-06-10 00:00:00.000	01:30:00.0000000	2024-06-10 00:00:00.000	01:30:00.0000000

Procedury

AddCourseToBasket, AddStudyMeetingToBasket, AddStudyToBasket, AddWebinarToBasket

Procedura polega na dodaniu kursu do koszyka. Na poziomie bazy danych dodaje ona do tabeli Orders nowy rekord z zamówieniem, które znajduje się w koszyku. Tabela OrderedCourses pełni tutaj rolę takiego Order Details(jednemu zamówieniu w tabeli Orders może odpowiadać kilka produltów w tabelach OrderedCourses, OrderedStudies, OrderedStudyMeetings, OrderedWebinars). Połączone są one z tabelą Orders przez OrderID. Po dodaniu pierwszego produktu do koszyka, generowany jest link do płatności i wstawiany do tabeli Orders dla odpowiedniego zamówienia. Walidacja przy dodawaniu produktu do koszyka(w tym wypadku kursu) polega na sprawdzeniu, czy nie został przekroczony limit uczestników, czy są jeszcze 3 dni przed rozpoczęciem kursu, czy dany student i dany kurs istnieją oraz przy tworzeniu koszyka, czy przypadkiem nie został on już stworzony. Podobnie działają poniższe procedury AddStudyMeetingToBasket, AddStudyToBasket, AddWebinarToBasket.

27	99b848bc-712f-4e41-a6d8-afd15e822ad3	55	NULL	Pending	2024-01-06 18:44:38.973	example.com
28	9a4def8b-6828-4059-9901-52d32483fa58	56	2024-01-02 18:29:54.963	Delivered	2024-01-02 18:28:48.177	example1.com

Na tym przykładowym zrzucie ekranu z tabeli Orders kolejne kolumny to:

- OrderID
- StudentID
- OrderDate
- Status
- AdditionToBasketDate
- PaymentHyperlink

13	99b848bc-712f-4e41-a6d8-afd15e822ad3	10	0	NULL	1	whatever	239c747e-3ff7-4ff2-a3c9-1cd9e410bcd0	NULL

A to zrzut ekranu z tabeli OrderedWebinars. Kolejne kolumny to:

- OrderID
- WebinarID
- HasBeenPaidFor
- PickupDate
- PaymentDeferral
- PaymentDeferralReason
- PaymentAuthCode
- Error

```
ALTER PROCEDURE [dbo].[AddCourseToBasket]
    @OrderID nvarchar(50),
    @StudentID int,
    @CourseID int,
    @PaymentHyperlink nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON; -- !!!
    -- weryfikujemy, czy podany kurs i student istnieją
BEGIN TRY
    IF NOT EXISTS (SELECT 1 FROM Courses WHERE CourseID = @CourseID) OR
        NOT EXISTS (select 1 from Students where StudentID=@StudentID)
        THROW 50002,'Nie istnieje Kurs albo Student o takim ID',1;
    -- weryfikujemy, czy zamówienie tego kursu nie spowodowałoby przekroczenia limitu uczestników danego kursu
    DECLARE @limit_exceeded bit = 0;
    DECLARE @max_limit int;
```

```
    SELECT @max_limit = Limit FROM Courses WHERE CourseID=@CourseID;

    IF (SELECT COUNT(*)
        FROM OrderedCourses
        WHERE CourseID = @CourseID) >= @max_limit
        SET @limit_exceeded = 1

    IF (@limit_exceeded=1)
        THROW 50001,'Limit uczestnikow przekroczony',1;
    DECLARE @date_ok bit; -- weryfikujemy, czy na pewno nadal można kupić kurs(u nas kurs można dodać do koszyka najpóźniej
    -- 3 dni przed jego rozpoczęciem, można opłacić go później, ale wtedy jest się dłużnikiem)
    SELECT @date_ok = CASE
                        WHEN DATEDIFF(second, GETDATE(), StartDate) > 259200 THEN 1 -- 259200s = 72h = 3 dni
                        ELSE 0
                        END
                    FROM
                    Courses
                    WHERE CourseID=@CourseID;

    IF (@date_ok=0)
    BEGIN
        THROW 50003,'Za późno na kupienie tego kursu',1;
    END;

    DECLARE @order_exists bit = 0;
    SET @order_exists = dbo.DoesOrderExist(@OrderID);
    IF (@order_exists=0)
    BEGIN
        INSERT INTO Orders (OrderID, StudentID, Status, AdditionToBasketDate, PaymentHyperlink)
        VALUES (@OrderID, @StudentID, 'Pending', GETDATE(), @PaymentHyperlink);
    END
    INSERT INTO OrderedCourses (OrderID, CourseID, HasBeenPaidFor)
    VALUES (@OrderID, @CourseID, 0);
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() as ErrorNumber,
           ERROR_MESSAGE() as ErrorMessage
END CATCH
END
```

```
ALTER PROCEDURE [dbo].[AddStudyMeetingToBasket]
    @OrderID nvarchar(50),
    @StudentID int,
    @StudyMeetingID int,
    @isPartOfStudies bit,
    @PaymentHyperlink nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON
    -- Insert statements for procedure here
    -- sprawdzamy czy takie studium i taki student istnieją
BEGIN TRY
    IF (NOT EXISTS (select 1 from StudyMeetings where StudyMeetingID=@StudyMeetingID )) OR
        (NOT EXISTS (select 1 from Students where StudentID=@StudentID))
        THROW 50001,'Nie istnieje Zjazd albo Student o takim ID',1;

    DECLARE @limit_exceeded bit;
    DECLARE @max_limit int;
    SELECT @max_limit = SeatCount FROM StudyMeetings WHERE StudyMeetingID=@StudyMeetingID;

    IF (SELECT COUNT(*)
        FROM OrderedStudyMeetings
        WHERE StudyMeetingID = @StudyMeetingID) >= @max_limit
        SET @limit_exceeded = 1
    ELSE
        SET @limit_exceeded = 0

    IF (@limit_exceeded=1)
        THROW 50001,'Limit uczestnikow przekroczony',1;

    DECLARE @date_ok bit; -- weryfikujemy, czy na pewno nadal można kupić studium(trzeba dokonać wpłaty na 3 dni przed
    -- rozpoczęciem)
    SELECT @date_ok = CASE
                        WHEN DATEDIFF(second, BeginningDate, GETDATE()) > 259200 THEN 1 -- 259200s = 72h = 3 dni
                        ELSE 0
                        END
                    FROM
                    StudyMeetings
                    WHERE StudyMeetingID=@StudyMeetingID;

    IF (@date_ok=0)
    BEGIN
        THROW 50001,'Za późno na kupienie tego kursu',1;
    END;

    -- sprawdzamy, czy koszyk już istnieje, jeśli nie, tworzymy go
    DECLARE @order_exists bit = 0;
    SET @order_exists = dbo.DoesOrderExist(@OrderID);
    IF (@order_exists=0)
    BEGIN
        INSERT INTO Orders (OrderID, StudentID, Status, AdditionToBasketDate, PaymentHyperlink)
        VALUES (@OrderID, @StudentID, 'Pending', GETDATE(), @PaymentHyperlink);
    END
    INSERT INTO OrderedStudyMeetings(OrderID, StudyMeetingID, HasBeenPaidFor, IsPartOfStudies)
    VALUES (@OrderID, @StudyMeetingID, 0, @isPartOfStudies);
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() as ErrorNumber,
           ERROR_MESSAGE() as ErrorMessage
END CATCH
END
```

```
ALTER PROCEDURE [dbo].[AddStudyToBasket]
    @OrderID nvarchar(50),
    @StudentID int,
    @StudyID int,
    @PaymentHyperlink nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON
    -- Insert statements for procedure here
BEGIN TRY
    IF (NOT EXISTS (select 1 from Studies where StudyID=@StudyID)) OR
        (NOT EXISTS (select 1 from Students where StudentID=@StudentID))
        THROW 50001,'Nie istnieje Studium albo Student o takim ID',1;

    DECLARE @limit_exceeded bit;
    DECLARE @max_limit int;
    SELECT @max_limit = Limit FROM Studies WHERE StudyID=@StudyID;

    IF (SELECT COUNT(*)
        FROM OrderedStudies
        WHERE StudyID = @StudyID) >= @max_limit
        SET @limit_exceeded = 1
    ELSE
        SET @limit_exceeded = 0

    IF (@limit_exceeded=1)
```

```
        THROW 50001,'Limit uczestnikow przekroczony',1;

    -- sprawdzamy, czy takie studium i taki student istnieją
    DECLARE @order_exists bit = 0;
    SET @order_exists = dbo.DoesOrderExist(@OrderID);
    IF (@order_exists=0)
    BEGIN
        INSERT INTO Orders (OrderID, StudentID, Status, AdditionToBasketDate, PaymentHyperlink)
        VALUES (@OrderID, @StudentID, 'Pending', GETDATE(), @PaymentHyperlink);
    END
    INSERT INTO OrderedStudies (OrderID, StudyID, EntryFeePaid)
    VALUES (@OrderID, @StudyID, 0);
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() as ErrorNumber,
           ERROR_MESSAGE() as ErrorMessage
END CATCH
END
```

```
ALTER PROCEDURE [dbo].[AddWebinarToBasket]
    @WebinarID int,
    @StudentID int,
    @OrderID nvarchar(50),
    @PaymentHyperlink nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON
    -- Insert statements for procedure here
    -- sprawdzamy, czy taki webinar i taki student istnieją
BEGIN TRY
    IF (EXISTS (select 1 from Webinars where WebinarID=@WebinarID)) AND
        (EXISTS (select 1 from Students where StudentID=@StudentID))
    BEGIN
        DECLARE @order_exists bit = 0;
        SET @order_exists = dbo.DoesOrderExist(@OrderID);
        IF (@order_exists=0)
        BEGIN
            INSERT INTO Orders (OrderID, StudentID, Status, AdditionToBasketDate, PaymentHyperlink)
            VALUES (@OrderID, @StudentID, 'Pending', GETDATE(), @PaymentHyperlink);
        END
        INSERT INTO OrderedWebinars (OrderID, WebinarID, HasBeenPaidFor)
        VALUES (@OrderID, @WebinarID, 0);
    END
    ELSE
    BEGIN
        THROW 50001,'Nie istnieje Webinar albo Student o takim ID',1;
    END
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() as ErrorNumber,
           ERROR_MESSAGE() as ErrorMessage
END CATCH
END
```

DeliverTheOrder

Procedura polega na dostarczeniu zamówionych produktów znajdujących się uprzednio w koszyku. Osobna procedura niżej, "PayForProduct" rejestruje wpłatę za poszczególne produkty zamówienia. Dostarczenie produktu polega na zmianie statusu zamówienia z "Pending" (w koszyku) na "Delivered" (dostarczone) oraz ustawieniu daty zamówienia na obecną datę.

```
ALTER PROCEDURE [dbo].[DeliverTheOrder]
    @order_id nvarchar(50)
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    DECLARE @current_status nvarchar(50);
    SELECT @current_status = Status
                             from Orders
                             where OrderID=@order_id;
    IF (@current_status='Pending')
    BEGIN
        UPDATE Orders
        SET Status='Delivered', OrderDate=GETDATE()
        WHERE OrderID=@order_id;
        INSERT INTO OrderedStudyMeetings (OrderID, StudyMeetingID, IsPartOfStudies)
        (select OrderID, StudyMeetingID, 1
                FROM Studies
                JOIN OrderedStudies on Studies.StudyID=OrderedStudies.StudyID
                JOIN StudyMeetings on Studies.StudyID = StudyMeetings.StudyID
                WHERE OrderID=@order_id);
    END;
    ELSE
    BEGIN
        RAISERROR ('Podane zamówienie zostało już dostarczone',16,1);
    END;
END
```

PayForProduct

Procedura polega na zarejestrowaniu wpłaty za dany produkt. Ustawiany jest klucz autoryzacyjny płatności w razie wystąpienia błędu oraz pole w tabeli "Error" jest uzupełniane w razie wystąpienia błędu, który możemy przechwycić. Jeśli błąd nie wystąpił, powinniśmy przekazać NULL jako @error.

```
ALTER PROCEDURE [dbo].[PayForProduct]
    @product_type nvarchar(50),
    @product_id int,
    @student_id int,
    @payment_auth_code nvarchar(50),
    @error nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    -- Ta procedura jest wywoływana w momencie zatwierdzenia płatności przez naszą aplikację.

    IF (@product_type='webinar')
    BEGIN
        UPDATE OrderedWebinars
        SET HasBeenPaidFor=1, PaymentAuthCode=@payment_auth_code, Error=@error, PaymentDate=GETDATE()
        FROM OrderedWebinars
        JOIN Orders on OrderedWebinars.OrderID=Orders.OrderID
        WHERE WebinarID=@product_id and StudentID=@student_id;

    END;
    ELSE IF (@product_type='course')
    BEGIN
        UPDATE OrderedCourses
        SET HasBeenPaidFor=1, PaymentAuthCode=@payment_auth_code, Error=@error, PaymentDate=GETDATE()
        FROM OrderedCourses
        JOIN Orders on OrderedCourses.OrderID=Orders.OrderID
        WHERE CourseID=@product_id and StudentID=@student_id;

    END;
```

```
    ELSE IF (@product_type='study')
    BEGIN
        UPDATE OrderedStudies
        SET EntryFeePaid=1, PaymentAuthCode=@payment_auth_code, Error=@error,PaymentDate=GETDATE()
        FROM OrderedStudies
        JOIN Orders on OrderedStudies.OrderID=Orders.OrderID
        WHERE StudyID=@product_id and StudentID=@student_id;

    END;
    ELSE IF (@product_type='study_meeting')
    BEGIN
        UPDATE OrderedStudyMeetings
        SET HasBeenPaidFor=1, PaymentAuthCode=@payment_auth_code, Error=@error, PaymentDate=GETDATE()
        FROM OrderedStudyMeetings
        JOIN Orders on OrderedStudyMeetings.OrderID=Orders.OrderID
        WHERE StudyMeetingID=@product_id and StudentID=@student_id;

    END;
    ELSE
    BEGIN
        RAISERROR ('Podano bledny typ produktu',16,1);
    END
END
```

RegisterCaughtUpStudyMeeting

Procedura polega na zarejestrowaniu odrobienia nieobecności na zjeździe(StudyMeeting) przez studenta. Po prostu ustawia ona wartość w odpowiednim rekordzie kolumny HasBeenCaughtUp(czy została odrobiona) na True. Wtedy nie jest ona liczona jako nieobecność np. w raporcie frekwencji. Walidacja polega na sprawdzeniu, czy student faktycznie posiada nieobecność na tym zjeździe. Jeśli jej nie posiada, procedura zwraca błąd.

```
ALTER PROCEDURE [dbo].[RegisterCaughtUpStudyMeetingAbsence]
    -- Add the parameters for the stored procedure here
    @study_meeting_id int,
    @student_id int
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    IF (exists (select 1 from StudyMeetingsAbsences where StudentID=@student_id and StudyMeetingID=@study_meeting_id))
        UPDATE StudyMeetingsAbsences
        SET HasBeenCaughtUp=1
        where StudentID=@student_id and StudyMeetingID=@study_meeting_id;
    ELSE
        RAISERROR ('Student nie zakupil tego zjazdu',16,1);
END
```

ApplyPaymentDeferralToOrderedProduct

Przypisuje danemu produktowi z zamówienia odroczenie płatności(PaymentDeferral). Walidacja polega na sprawdzeniu, czy dany produkt nie został już przypadkiem opłacony - wtedy nie nadajemy odroczenia płatności.

```
ALTER PROCEDURE [dbo].[ApplyPaymentDeferralToOrderedProduct]
    @product_type nvarchar(50),
    @order_id nvarchar(50),
    @product_id int,
    @payment_deferral_reason nvarchar(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON; -- !!
    -- Insert statements for procedure here
    BEGIN TRY
        IF (@product_type='webinar')
        BEGIN
            IF ((select HasBeenPaidFor from OrderedWebinars where OrderID=@order_id and WebinarID=@product_id)=0)
            BEGIN
                UPDATE OrderedWebinars
                SET PaymentDeferral=1, PaymentDeferralReason=@payment_deferral_reason
                WHERE OrderID=@order_id and WebinarID=@product_id;
            END
            ELSE
            BEGIN
                THROW 50001, 'Ten produkt zostal juz oplacony', 1;
            END
        END
        IF (@product_type='course')
        BEGIN
            IF ((select HasBeenPaidFor from OrderedCourses where OrderID=@order_id and CourseID=@product_id)=0)
            BEGIN
                UPDATE OrderedCourses
                SET PaymentDeferral=1, PaymentDeferralReason=@payment_deferral_reason
                WHERE OrderID=@order_id and CourseID=@product_id;
            END
            ELSE
            BEGIN
                THROW 50002, 'Ten produkt zostal juz oplacony', 1;
            END
        END
        IF (@product_type='study')
        BEGIN
            IF ((select EntryFeePaid from OrderedStudies where OrderID=@order_id and StudyID=@product_id)=0)
            BEGIN
                UPDATE OrderedStudies
                SET PaymentDeferral=1, PaymentDeferralReason=@payment_deferral_reason
                WHERE OrderID=@order_id and StudyID=@product_id;
            END
            ELSE
            BEGIN
                THROW 50003, 'Ten produkt zostal juz oplacony', 1;
            END
        END
        IF (@product_type='study_meeting')
        BEGIN
            IF ((select HasBeenPaidFor from OrderedStudyMeetings where OrderID=@order_id and StudyMeetingID=@product_id)=0)
            BEGIN
                UPDATE OrderedStudyMeetings
                SET PaymentDeferral=1, PaymentDeferralReason=@payment_deferral_reason
                WHERE OrderID=@order_id and StudyMeetingID=@product_id;
            END
            ELSE
            BEGIN
                THROW 50004, 'Ten produkt zostal juz oplacony', 1;
            END
        END
    END TRY
    BEGIN CATCH
        SELECT ERROR_NUMBER() as ErrorNumber,
               ERROR_MESSAGE() as ErrorMessage
    END CATCH
END
```

GrantStudentCertificate

Procedura polega na przyznaniu studentowi certyfikatu ukończenia studiów. Najpierw przechodzimy przez następujące walidacje:

- czy student w ogóle zamówił dane studium
- czy osiągnął 80% obecności
- czy zdał egzamin końcowy
- czy zaliczył praktyki
- czy dokonał wszelkich należności pieniężnych Jeśli tak, wstawiamy do bazy podany link do certyfikatu dla tego studenta.

```
ALTER PROCEDURE [dbo].[GrantStudentCertificate]
    @student_id int,
    @certificate_hyperlink nvarchar(MAX),
    @study_id int
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON
    -- Insert statements for procedure here
    -- Najpierw sprawdzamy, czy student zamówił studium o takim @study_id
    DECLARE @ordered_study_exists bit;
    SELECT @ordered_study_exists = CASE
                                    WHEN count(StudyID) > 0 then 1
                                    else 0
                                   END
                                    from OrderedStudies
                                    join Orders on Orders.OrderID=OrderedStudies.OrderID
                                    where Orders.StudentID=@student_id and OrderedStudies.StudyID=@study_id;
    -- Teraz liczymy ilość wszystkich spotkań i ilość niedrobionych absencji danego studenta na danym studium
    DECLARE @total_study_meetings_count decimal;
    SELECT @total_study_meetings_count = count(StudyMeetingID) from StudyMeetings
                                    where StudyID=@study_id;
    DECLARE @absences_count decimal;
    SELECT @absences_count = count(StudyMeetingsAbsences.StudyMeetingID)
                                from StudyMeetingsAbsences
                                join StudyMeetings on StudyMeetings.StudyMeetingID = StudyMeetingsAbsences.StudyMeetingID
                                where StudentID=@student_id and StudyID=@study_id and HasBeenCaughtUp=0;
    -- Sprawdzamy, czy student zdał egzamin końcowy z danego studium i czy zaliczył praktyki
    DECLARE @final_exam_passed bit;
    SELECT @final_exam_passed = FinalExamPassed
                                from OrderedStudies
                                join Orders on Orders.OrderID = OrderedStudies.OrderID
                                where StudyID=@study_id and StudentID=@student_id;
    DECLARE @failed_internship bit;
    SELECT @failed_internship = FailedInternship
                                from OrderedStudies
                                join Orders on Orders.OrderID = OrderedStudies.OrderID
                                where StudyID=@study_id and StudentID=@student_id;
    -- Sprawdzamy dodatkowo, czy student opłacił wszelkie spotkania studyjne
    DECLARE @everything_paid bit;
    SELECT @everything_paid = CASE
                                WHEN COUNT(*) > 0 THEN 0
                                ELSE 1
                              END
                                from OrderedStudies
                                join Orders on Orders.OrderID = OrderedStudies.OrderID
                                join OrderedStudyMeetings on OrderedStudyMeetings.OrderID = Orders.OrderID
                                where StudentID=@student_id and StudyID=@study_id and HasBeenPaidFor=0;
    -- Trzeba jeszcze sprawdzić, czy student uiścił wpisowe
    DECLARE @entry_fee_paid bit;
    SELECT @entry_fee_paid = CASE
                                WHEN EntryFeePaid=1 THEN 1
                                ELSE 0
                             END
                                from OrderedStudies
                                join Orders on Orders.OrderID=OrderedStudies.OrderID
                                where StudentID=@student_id and StudyID=@study_id;
    -- Ostateczna walidacja
    IF (@ordered_study_exists=1 and @total_study_meetings_count > 0 and @absences_count/@total_study_meetings_count<=0.2
        and @final_exam_passed=1 and @failed_internship=0 and @everything_paid=1 and @entry_fee_paid=1)
    BEGIN
        UPDATE OrderedStudies
        SET IsGrantedCertificate=1,
            CertificateHyperlink=@certificate_hyperlink
        FROM OrderedStudies
        join Orders on Orders.OrderID = OrderedStudies.OrderID
        where StudentID=@student_id and StudyID=@study_id;
    END;
    ELSE
    BEGIN
        RAISERROR ('Student nie spełnia kryteriów otrzymania certyfikatu z tego studium',16,1);
    END;
END
```

InsertEmployees

Procedura wstawiania pracowników

```
ALTER PROCEDURE [dbo].[InsertEmployees]
    @Email nvarchar(50),
    @Password nvarchar(50),
    @Name nvarchar(50),
    @Surname nvarchar(50),
    @Country nvarchar(50),
    @City nvarchar(50),
    @ZipCode nvarchar(50),
    @Street nvarchar(50),
    @Address nvarchar(50),
    @Phone nvarchar(50),
    @type nvarchar(50)
AS
BEGIN
    insert into Users(Email, Password, Name, Surname, CountryID, CityID, ZipCode, Street, Address, PhoneNumber)
    values (@Email, @Password, @Name, @Surname, (select CountryID from Countries where CountryName = @Country), (select CityID from City where CityName = @City), @ZipCode, @Street, @Address, @Phone);
    insert into Employees(EmployeeID, type) values ((select UserID from Users where Email = @Email), @type);
END
```

InsertStudents

Procedura wstawiania studentów

```
ALTER PROCEDURE [dbo].[InsertStudents]
    @Email nvarchar(50),
    @Password nvarchar(50),
    @Name nvarchar(50),
    @Surname nvarchar(50),
    @Country nvarchar(50),
    @City nvarchar(50),
    @ZipCode nvarchar(50),
    @Street nvarchar(50),
    @Address nvarchar(50),
    @Phone nvarchar(50)
AS
BEGIN
    insert into Users(Email, Password, Name, Surname, CountryID, CityID, ZipCode, Street, Address, PhoneNumber)
    values (@Email, @Password, @Name, @Surname, (select CountryID from Countries where CountryName = @Country), (select CityID from City where CityName = @City), @ZipCode, @Street, @Address, @Phone);
    insert into Students(StudentID) values ((select UserID from Users where Email = @Email));
END
```

InsertTeachers

Procedura wstawiania nauczycieli

```
ALTER PROCEDURE [dbo].[InsertTeachers]
    @Email nvarchar(50),
    @Password nvarchar(50),
    @Name nvarchar(50),
    @Surname nvarchar(50),
    @Country nvarchar(50),
    @City nvarchar(50),
    @ZipCode nvarchar(50),
    @Street nvarchar(50),
    @Address nvarchar(50),
    @Phone nvarchar(50)
AS
BEGIN
    insert into Users(Email, Password, Name, Surname, CountryID, CityID, ZipCode, Street, Address, PhoneNumber)
    values (@Email, @Password, @Name, @Surname, (select CountryID from Countries where CountryName = @Country), (select CityID from City where CityName = @City), @ZipCode, @Street, @Address, @Phone);
    insert into Teachers(TeacherID) values ((select UserID from Users where Email = @Email));
END
```

RegisterModuleAbsence

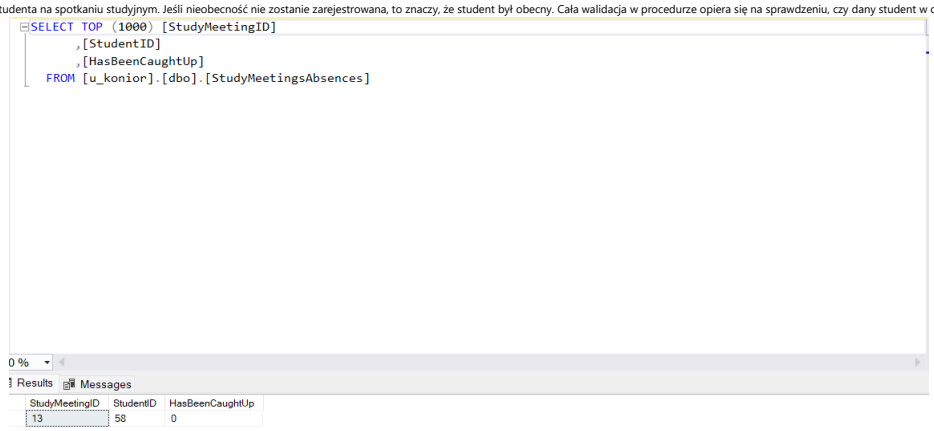
Procedura polega na zarejestrowaniu nieobecności studenta na module poprzez dodanie rekordu do tabeli ModulesAbsences. Walidacja polega jedynie na sprawdzeniu, czy dany student w ogóle zamówił kurs z dnaym modułem.

```
ALTER PROCEDURE [dbo].[RegisterModuleAbsence]
    @module_id int,
    @student_id int
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    -- Sprawdzamy, czy dany student zamówił kurs z danym modułem
    IF (EXISTS (SELECT 1 FROM OrderedCourses
                        JOIN Courses on Courses.CourseID=OrderedCourses.CourseID
                        JOIN CoursesModules on Courses.CourseID=CoursesModules.CourseID
                        JOIN Orders on Orders.OrderID = OrderedCourses.OrderID
                        WHERE ModuleID=@module_id AND StudentID=@student_id))
    BEGIN
        INSERT INTO ModulesAbsences (ModuleID, StudentID)
        VALUES (@module_id, @student_id);
    END;
    ELSE
        RAISERROR ('Student nie posiada kursu z tym modulem',16,1);
END
```

RegisterStudyMeetingAbsence

Procedura ma na celu zarejestrowanie nieobecności studenta na spotkaniu studyjnym. Jeśli nieobecność nie zostanie zarejestrowana, to znaczy, że student był obecny. Cała walidacja w procedurze opiera się na sprawdzeniu, czy dany student w ogóle zapisał się na dane spotkanie



studyjne. Jeśli tak, nieobecność zostaje zarejestrowana.

```
ALTER PROCEDURE [dbo].[RegisterStudyMeetingAbsence]
    @student_id int,
    @study_meeting_id int
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    -- sprawdzamy, czy dany student zamowil w ogole dany zjazd
    IF (EXISTS (SELECT 1 FROM OrderedStudyMeetings
                        JOIN Orders on Orders.OrderID = OrderedStudyMeetings.OrderID
                        WHERE StudyMeetingID=@study_meeting_id AND StudentID=@student_id))
    BEGIN
        INSERT INTO StudyMeetingsAbsences (StudyMeetingID, StudentID, HasBeenCaughtUp)
        VALUES (@study_meeting_id, @student_id, 0);
    END
    ELSE
        RAISERROR ('Student nie zamowil tego zjazdu',16,1);
END
```

InsertCourses

Procedura służąca do wstawiania kursów do bazy

```
ALTER PROCEDURE [dbo].[InsertCourses]
    @Name NVARCHAR(50),
    @Price MONEY,
    @StartDate DATETIME,
    @Duration INT,
```

```
    @ModulesCount INT,
    @Limit INT,
    @Language NVARCHAR(50),
    @TranslatorName NVARCHAR(50),
    @TranslatorSurname NVARCHAR(50),
    @Hyperlink NVARCHAR(100)
AS
BEGIN
    -- sprawdzamy czy jezyk istnieje
    if not exists (select * from Languages where Language = @Language)
        begin
            raiserror('Język nie istnieje', 16, 1)
        end
    else
        begin
            -- sprawdzamy czy data jest wieksza od dzisiejszej
            if @StartDate <= GETDATE()
                begin
                    raiserror('Data startu musi byc po dniu dzisiajeszym', 16, 1)
                end
            else
                begin
                    insert into Courses (Name, Price, StartDate, Duration, ModulesCount, Limit, LanguageID, TranslatorName, TranslatorSurname, Hyperlink)
                    values (@Name, @Price, @StartDate, @Duration, @ModulesCount, @Limit, (select LanguageID from Languages where Language = @Language), @TranslatorName, @TranslatorSurname, @Hyperlink)
                end
        end
END
```

InsertStudyMeetings

Procedura służąca do wstawiania spotkań w ramach studiów do bazy

```
ALTER PROCEDURE [dbo].[InsertStudieMeetings]
    @Study nvarchar(50),
    @Type nvarchar(50),
    @TeacherID int,
    @Language nvarchar(50),
    @MeetingName nvarchar(50),
    @MeetingPrice money,
    @MeetingPriceForStudents money,
    @BeginningDate datetime,
    @Duration time,
    @Syllabus nvarchar(max),
    @Limit int,
    @TranslatorName nvarchar(50),
    @TranslatorSurname nvarchar(50)
AS
BEGIN
    if not exists (select * from Teachers where TeacherId = @TeacherID)
        begin
            raiserror('Nauczyciel nie istnieje', 16, 1)
        end
    else if not exists (select * from Languages where Language = @Language)
        begin
            raiserror('Język nie istnieje', 16, 1)
        end
    else if not exists (select * from Studies where FieldOfStudy = @Study)
        begin
            raiserror('Kierunek nie istnieje', 16, 1)
        end
    else if @BeginningDate <= getdate()
        begin
            raiserror('Data rozpoczęcia musi być późniejsza niż dzisiejsza', 16, 1)
        end
    else
        begin
            insert into StudyMeetings (StudyID, Type, TeacherID, LanguageID, MeetingName, MeetingPrice, MeetingPriceForStudents, BeginningDate, Duration, MeetingSyllabusDescription, SeatCount, TranslatorName,
TranslatorSurname)
            values ((select StudyID from Studies where FieldOfStudy = @Study), @Type, @TeacherID, (select LanguageID from Languages where Languages.LanguageID = @Language), @MeetingName, @MeetingPrice,
@MeetingPriceForStudents, @BeginningDate, @Duration, @Syllabus, @Limit, @TranslatorName, @TranslatorSurname)
        end
END
```

#InsertStudies Procedura służąca do wstawiania studiów do bazy

```
ALTER PROCEDURE [dbo].[InsertStudies]
    @FieldOfStudy nvarchar(50),
    @Duration int,
    @EntryFee money,
    @AcademicYear int,
    @Limit int,
    @Syllabus nvarchar(max),
    @InternshipName nvarchar(50),
    @InternshipStartDate datetime
AS
BEGIN
    if @AcademicYear >= YEAR(getdate())
        begin
            insert into Studies (FieldOfStudy, Duration, EntryFee, AcademicYear, Limit, SyllabusDescription, InternshipName, InternshipStartDate)
            values (@FieldOfStudy, @Duration, @EntryFee, @AcademicYear, @Limit, @Syllabus, @InternshipName, @InternshipStartDate)
        end
END
```

InsertWebinars

Procedura służąca do wstawiania webinarów do bazy

```
ALTER PROCEDURE [dbo].[InsertWebinars]
    @TeacherId INT,
    @Name NVARCHAR(50),
    @Price MONEY,
    @Hyperlink NVARCHAR(100),
    @Language NVARCHAR(50),
    @TranslatorName NVARCHAR(50),
    @TranslatorSurname NVARCHAR(50),
    @StartDate DATETIME,
    @Duration TIME
AS
BEGIN
    -- sprawdzamy czy nauczyciel istnieje
    if not exists (select * from Teachers where TeacherId = @TeacherId)
        begin
            raiserror('Nauczyciel nie istnieje', 16, 1)
        end
    else
        begin
            -- sprawdzamy czy jezyk istnieje
            if not exists (select * from Languages where Language = @Language)
                begin
                    raiserror('Język nie istnieje', 16, 1)
                end
            else
                begin
                    -- sprawdzamy czy data jest wieksza od dzisiejszej
                    if @StartDate <= GETDATE()
                        begin
```

```
                        raiserror('Data startu musi byc po dniu dzisiajeszym', 16, 1)
                    end
                else
                    begin
                        insert into Webinars (TeacherId, Name, Price, Hyperlink, LanguageID, TranslatorName, TranslatorSurname, StartDate, Duration)
                        values (@TeacherId, @Name, @Price, @Hyperlink, (select LanguageID from Languages where Language = @Language), @TranslatorName, @TranslatorSurname, @StartDate, @Duration)
                    end
            end
        end
END
```

Triggery:

UpdateStudyLimit

Po dodaniu nowego spotkania w ramach studiów trigger aktualizuje limit miejsc na studiach, żeby limit na studia był mniejszy lub równy niż każdy z pośród spotkań, z których się składa

```
CREATE TRIGGER UpdateStudyLimit
ON StudyMeetings
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Studies
    SET Limit = i.SeatCount
    FROM inserted i
    INNER JOIN Studies s ON i.StudyID = s.StudyID
    WHERE s.Limit > i.SeatCount;
END
GO
```

Role:

Teacher (nauczyciel)

- Rejestruje nieobecności studentów
- Rejestruje odrobienie nieobecności przez studenta

```
create role Teacher
grant execute on RegisterStudyMeetingAbsence to Teacher
grant execute on RegisterModuleAbsence to Teacher
grant execute on RegisterCaughtUpStudyMeetingAbsence to Teacher
```

Student

- Zamawia produkty
- Otrzymuje je
- Płaci za produkty

```
create role Student
grant execute on AddCourseToBasket to Student
grant execute on AddStudyMeetingToBasket to Student
grant execute on AddStudyToBasket to Student
grant execute on AddWebinarToBasket to Student
grant execute on DeliverTheOrder to Student
grant execute on PayForProduct to Student
```

Employee (pracownik biura)

- Generuje certyfikaty
- Wstawia kursy, webinary i studia do bazy
- Wstawia nowych pracowników, studentów i nauczycieli do bazy
- Generuje raporty finansowe
- Generuje raporty dłużników
- Sprawdza frekwencje studentów

```
create role employee
grant execute on GrantStudentCertificate to employee
grant execute on InsertCourses to employee
grant execute on InsertStudies to employee
grant execute on InsertWebinars to employee
grant execute on InsertStudent to employee
grant execute on InsertEmployees to employee
grant execute on InsertTeachers to employee
grant execute on InsertStudyMeetings to employee
grant execute on SelectAllCustomers to employee
grant select on n1_CoursesFinancialReport to employee
grant select on n1_WebinarsFinancialReport to employee
grant select on n1_MeetingsNoStudiesFinancialReport to employee
grant select on n_1_StudiesFinancialReport to employee
grant select on n_FrekwencjaSzczegółowaMeetings to employee
grant select on n_RaportDotyczącyLiczbyOsóbNaCoursesModules to employee
grant select on n_RaportDotyczącyLiczbyOsóbNaWebinars to employee
grant select on n_RaportDotyczącyLiczbyOsóbNaMeetings to employee
grant select on RaportDotyczącyLiczbyOsóbNaWebinars to employee
grant select on n_RaportFrekwencjiMeetings to employee
grant select on n_RaportFrekwencjiModules to employee
grant select on n_RaportDłużnikówCourses to employee
grant select on n_RaportDłużnikówStudies to employee
grant select on n_RaportDłużnikówWebinars to employee
grant select on n_RaportDłużnikówStudyMeetingsNieStudium to employee
grant select on n_RaportFinansowyCourses to employee
grant select on n_RaportFinansowyStudies to employee
grant select on n_RaportFinansowyWebinars to employee
```

Director (dyrektor)

- Odroczenie płatności
- Generowanie raportów finansowych
- Generowanie raportów dłużników

```
create role Director
grant execute on ApplyPaymentDeferralToOrderedProduct to Director
grant execute on SelectAllCustomers to Director
grant select on n1_CoursesFinancialReport to Director
grant select on n1_WebinarsFinancialReport to Director
grant select on n1_MeetingsNoStudiesFinancialReport to Director
grant select on n_1_StudiesFinancialReport to Director
grant select on n_RaportDłużnikówCourses to Director
grant select on n_RaportDłużnikówStudies to Director
grant select on n_RaportDłużnikówWebinars to Director
grant select on n_RaportDłużnikówStudyMeetingsNieStudium to Director
```