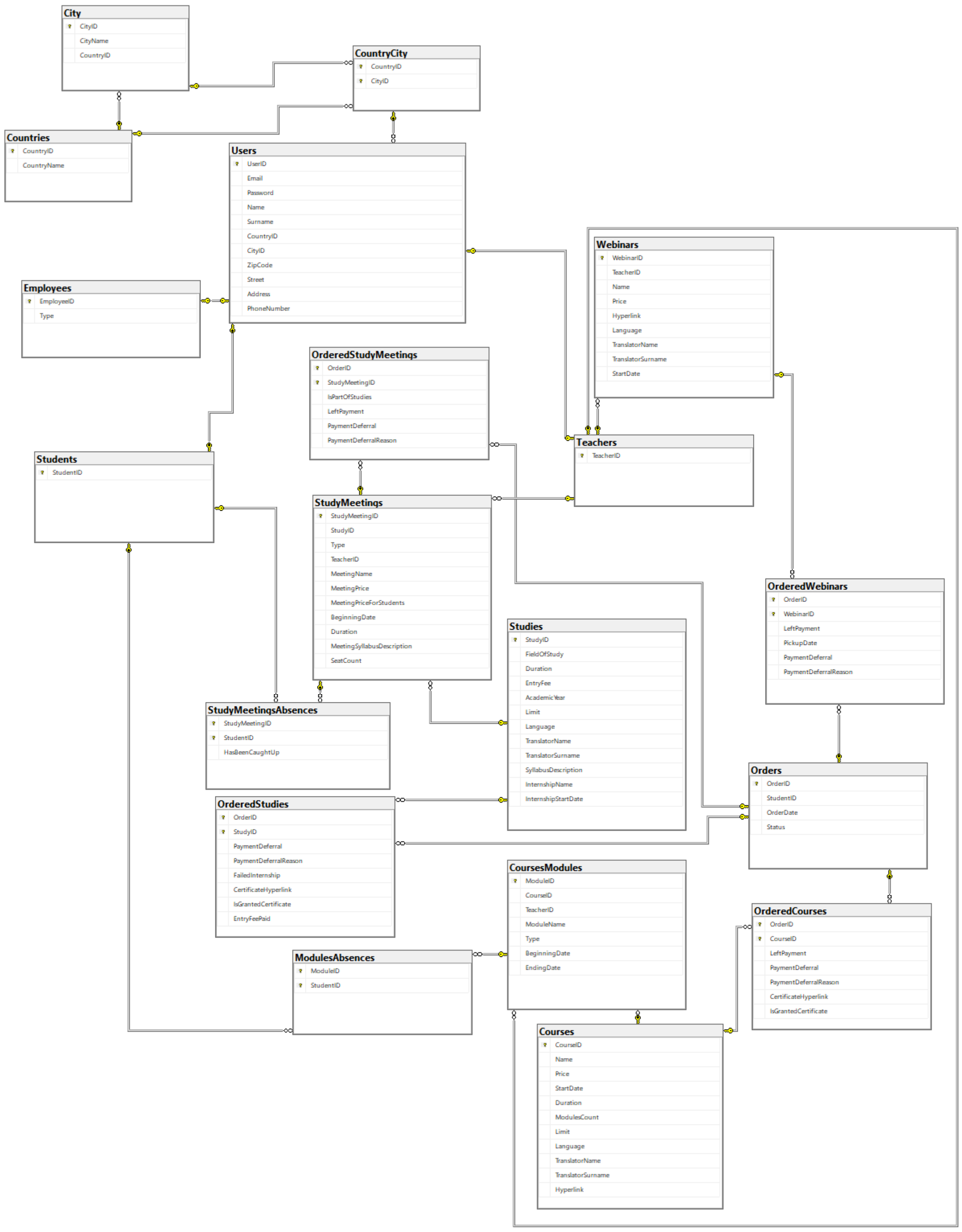
pbd_<14>_raport4 | Piotr Albiński, Adam Konior, Mateusz Maciaszczyk

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,
    [Language] [nvarchar](50) 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 [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,
    [EndingDate] [datetime] NOT NULL,
    [SeatCount] [int] 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 [Dates] CHECK  (([BeginningDate]<[EndingDate]))
GO

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

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

ALTER TABLE [dbo].[CoursesModules] CHECK CONSTRAINT [SeatCount]
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 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,
    [LeftPayment] [money] NOT NULL,
    [PaymentDeferral] [bit] NOT NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [CertificateHyperlink] [nvarchar](100) NULL,
    [IsGrantedCertificate] [bit] NOT 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_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_LeftPayment] CHECK  (([LeftPayment]>=(0)))
GO

ALTER TABLE [dbo].[OrderedCourses] CHECK CONSTRAINT [OC_LeftPayment]
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] NOT NULL,
    [PaymentDeferralReason] [nvarchar](max) NULL,
    [FailedInternship] [bit] NOT NULL,
    [CertificateHyperlink] [nvarchar](100) NULL,
    [IsGrantedCertificate] [bit] NOT NULL,
    [EntryFeePaid] [bit] NOT 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_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,
    [LeftPayment] [money] NOT NULL,
    [PaymentDeferral] [bit] NOT NULL,
    [PaymentDeferralReason] [nvarchar](max) 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]  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,
    [LeftPayment] [money] NOT NULL,
    [PickupDate] [datetime] NOT NULL,
    [PaymentDeferral] [bit] NOT NULL,
    [PaymentDeferralReason] [nvarchar](max) 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]  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_LeftPayment] CHECK  (([LeftPayment]>=(0)))
GO

ALTER TABLE [dbo].[OrderedWebinars] CHECK CONSTRAINT [OW_LeftPayment]
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] NOT NULL,
    [Status] [nvarchar](50) NOT 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]
GO

ALTER TABLE [dbo].[Orders]  WITH CHECK ADD  CONSTRAINT [O_Status] CHECK  (([Status]='Delivered' OR [Status]='Pending' OR [Status]='InBasket'))
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,
    [Language] [nvarchar](50) NOT NULL,
    [TranslatorName] [nvarchar](50) NULL,
    [TranslatorSurname] [nchar](10) 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

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

ALTER TABLE [dbo].[Studies] CHECK CONSTRAINT [S_Translator]
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,
    [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,
 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_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

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [SM_Type] CHECK  (([Type]='Zdalne' OR [Type]='Hybrydowe' OR [Type]='Stacjonarne'))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [SM_Type]
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,
    [Language] [nvarchar](50) NOT NULL,
    [TranslatorName] [nvarchar](50) NULL,
    [TranslatorSurname] [nvarchar](50) NULL,
    [StartDate] [datetime] 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_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_Language] CHECK  (([Language]<>''))
GO

ALTER TABLE [dbo].[Webinars] CHECK CONSTRAINT [W_Language]
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

Raport Dłużników Courses

```
SELECT dbo.Students.StudentID, dbo.OrderedCourses.LeftPayment, GETDATE() AS CurrentDate, dbo.Courses.StartDate
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
WHERE (dbo.OrderedCourses.LeftPayment > 0)
AND (dbo.OrderedCourses.PaymentDeferral = 0)
AND (DATEDIFF(day, GETDATE(), dbo.Courses.StartDate) <= 3)
AND (dbo.Orders.Status = 'Delivered')
```

Raport Dłużników Studies

```
SELECT dbo.Students.StudentID
FROM   dbo.OrderedStudies INNER JOIN
            dbo.Studies ON dbo.OrderedStudies.StudyID = dbo.Studies.StudyID INNER JOIN
            dbo.StudyMeetings ON dbo.Studies.StudyID = dbo.StudyMeetings.StudyID INNER JOIN
            dbo.OrderedStudyMeetings ON dbo.StudyMeetings.StudyMeetingID = dbo.OrderedStudyMeetings.StudyMeetingID INNER JOIN
            dbo.Orders ON dbo.OrderedStudies.OrderID = dbo.Orders.OrderID
       AND dbo.OrderedStudyMeetings.OrderID = dbo.Orders.OrderID INNER JOIN
            dbo.Students ON dbo.Orders.StudentID = dbo.Students.StudentID
WHERE (dbo.OrderedStudies.PaymentDeferral = 0)
    AND (dbo.OrderedStudies.EntryFeePaid = 0)
    AND (dbo.Orders.Status = 'Delivered') OR
    (dbo.Orders.Status = 'Delivered')
    AND (dbo.OrderedStudyMeetings.IsPartOfStudies = 1)
    AND (dbo.OrderedStudyMeetings.LeftPayment > 0)
    AND (dbo.OrderedStudyMeetings.PaymentDeferral = 0)
    AND (DATEDIFF(day, GETDATE(), dbo.StudyMeetings.BeginningDate) <= 3)
```

Raport Dłużników StudyMeetingsBezStudium

```
SELECT dbo.Students.StudentID
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
WHERE (dbo.OrderedStudyMeetings.IsPartOfStudies = 0)
AND (dbo.OrderedStudyMeetings.LeftPayment > 0)
AND (dbo.OrderedStudyMeetings.PaymentDeferral = 0)
AND (DATEDIFF(day, GETDATE(), dbo.StudyMeetings.BeginningDate) <= 3)
AND (dbo.Orders.Status = 'Delivered')
```

Raport DłużnikówWebinars

```
SELECT dbo.OrderedWebinars.LeftPayment, GETDATE() AS CurrentDate, dbo.Webinars.StartDate,
 dbo.OrderedWebinars.PaymentDeferral, dbo.Students.StudentID
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.OrderedWebinars.LeftPayment > 0)
AND (dbo.Webinars.StartDate < GETDATE())
AND (dbo.OrderedWebinars.PaymentDeferral = 0)
AND (dbo.Orders.Status = 'Delivered')
```

Raport DotyczącyLiczbyOsóbNaCourses

```
SELECT dbo.CoursesModules.Type, COUNT(dbo.OrderedCourses.CourseID) AS [Liczba osob], dbo.CoursesModules.ModuleName
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.CoursesModules.BeginningDate < GETDATE())
GROUP BY dbo.Students.StudentID, dbo.OrderedCourses.CourseID,
 dbo.CoursesModules.Type, dbo.CoursesModules.ModuleName
```

Raport DotyczącyLiczbyOsóbNaMeetings

```
SELECT dbo.StudyMeetings.MeetingName, dbo.StudyMeetings.Type, COUNT(dbo.OrderedStudyMeetings.StudyMeetingID) AS [Liczba osob]
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.StudyMeetings.BeginningDate < GETDATE())
GROUP BY dbo.OrderedStudyMeetings.StudyMeetingID, dbo.StudyMeetings.Type, dbo.StudyMeetings.MeetingName
```

Raport DotyczącyLiczbyOsóbNaWebinars

```
SELECT dbo.Webinars.Name, COUNT(dbo.OrderedWebinars.WebinarID) AS [Liczba osob]
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.Webinars.StartDate < GETDATE())
GROUP BY dbo.Webinars.Name, dbo.OrderedWebinars.WebinarID
```

Raport FinansowyKursy

```
SELECT dbo.Courses.Name, COALESCE (COUNT(dbo.OrderedCourses.OrderID) * dbo.Courses.Price, 0) AS Przychód
FROM   dbo.Courses LEFT OUTER JOIN
             dbo.OrderedCourses ON dbo.Courses.CourseID = dbo.OrderedCourses.CourseID
GROUP BY dbo.Courses.CourseID, dbo.Courses.Name, dbo.Courses.Price
```

Raport FinansowyStudia

```
SELECT dbo.Studies.FieldOfStudy, COUNT(dbo.OrderedStudies.OrderID) * dbo.Studies.Price AS Przychód
FROM   dbo.OrderedStudies RIGHT OUTER JOIN
             dbo.Studies ON dbo.OrderedStudies.StudyID = dbo.Studies.StudyID
GROUP BY dbo.Studies.StudyID, dbo.Studies.FieldOfStudy, dbo.Studies.Price
```

Raport FinansowyWebinary

```
SELECT dbo.Webinars.Name, COUNT(dbo.OrderedWebinars.OrderID) * dbo.Webinars.Price AS Przychód
FROM   dbo.OrderedWebinars RIGHT OUTER JOIN
             dbo.Webinars ON dbo.OrderedWebinars.WebinarID = dbo.Webinars.WebinarID
GROUP BY dbo.Webinars.WebinarID, dbo.Webinars.Name, dbo.Webinars.Price
```