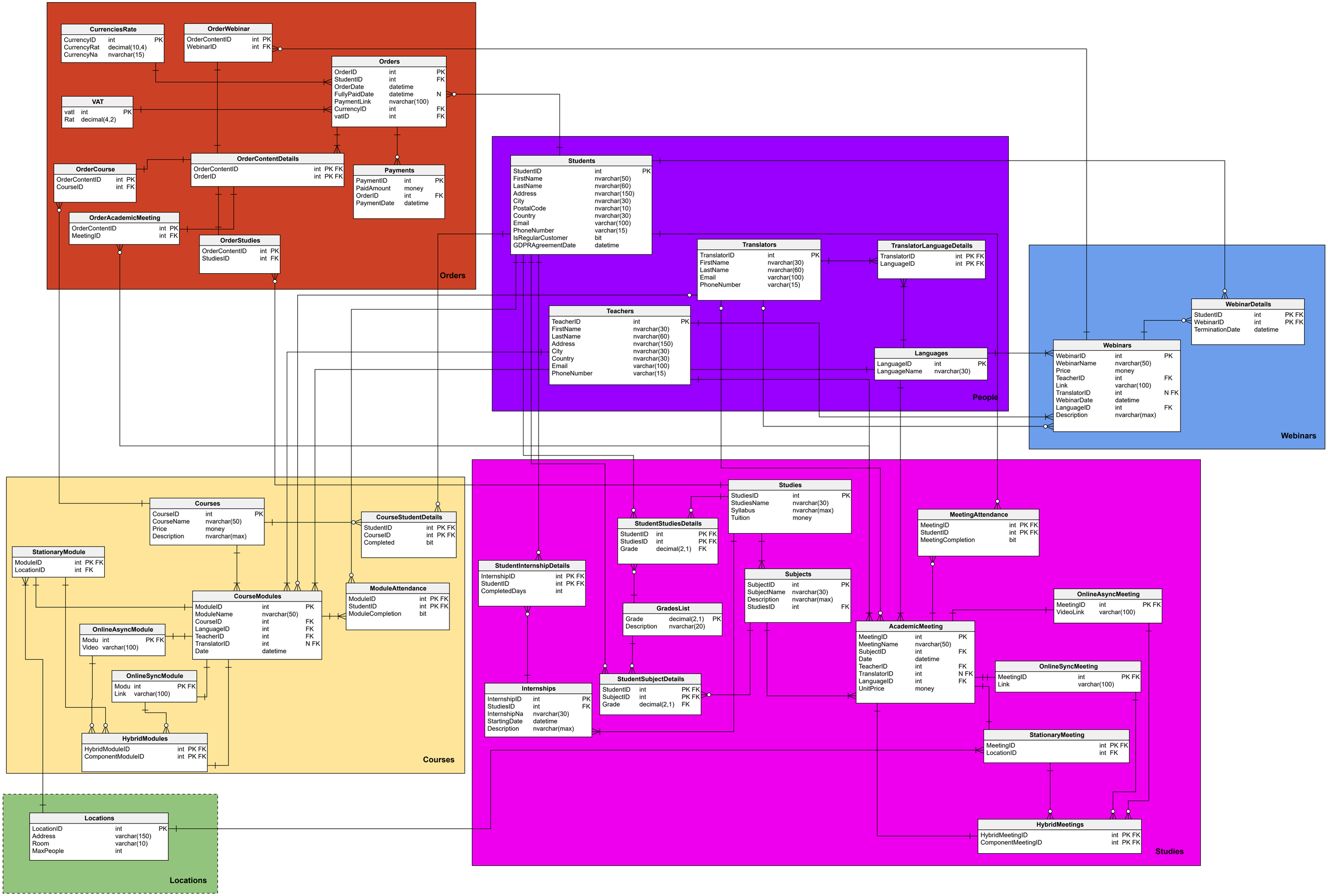
AGH Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Podstawy Baz Danych 2024/2025
Projekt

Kacper Feliks
Aleksander Jóźwik
Karol Półchłopek


CurrenciesRate
CurrencyID int PK
CurrencyRat decimal(10,4)
CurrencyNa nvarchar(15)
OrderWebinar
OrderContentID int PK
WebinarID int FK
Orders
OrderID int PK
StudentID int FK
OrderDate datetime
FullyPaidDate datetime N
PaymentLink nvarchar(100)
CurrencyID int FK
vatID int FK
VAT
vatI int PK
Rat decimal(4,2)
OrderContentDetails
OrderContentID int PK FK
OrderID int PK FK
Payments
PaymentID int PK
PaidAmount money
OrderID int FK
PaymentDate datetime
OrderCourse
OrderContentID int PK
CourseID int FK
OrderAcademicMeeting
OrderContentID int PK
MeetingID int FK
OrderStudies
OrderContentID int PK
StudiesID int FK
Orders
Students
StudentID int PK
FirstName nvarchar(50)
LastName nvarchar(60)
Address nvarchar(150)
City nvarchar(30)
PostalCode nvarchar(10)
Country nvarchar(30)
Email varchar(100)
PhoneNumber varchar(15)
IsRegularCustomer bit
GDPRAgreementDate datetime
Translators
TranslatorID int PK
FirstName nvarchar(30)
LastName nvarchar(60)
Email varchar(100)
PhoneNumber varchar(15)
TranslatorLanguageDetails
TranslatorID int PK FK
LanguageID int PK FK
Teachers
TeacherID int PK
FirstName nvarchar(30)
LastName nvarchar(60)
Address nvarchar(150)
City nvarchar(30)
Country nvarchar(30)
Email varchar(100)
PhoneNumber varchar(15)
Languages
LanguageID int PK
LanguageName nvarchar(30)
People
WebinarDetails
StudentID int PK FK
WebinarID int PK FK
TerminationDate datetime
Webinars
WebinarID int PK
WebinarName nvarchar(50)
Price money
TeacherID int FK
Link varchar(100)
TranslatorID int N FK
WebinarDate datetime
LanguageID int FK
Description nvarchar(max)
Webinars
Courses
CourseID int PK
CourseName nvarchar(50)
Price money
Description nvarchar(max)
CourseStudentDetails
StudentID int PK FK
CourseID int PK FK
Completed bit
StationaryModule
ModuleID int PK FK
LocationID int FK
CourseModules
ModuleID int PK
ModuleName nvarchar(50)
CourseID int FK
LanguageID int FK
TeacherID int FK
TranslatorID int N FK
Date datetime
ModuleAttendance
ModuleID int PK FK
StudentID int PK FK
ModuleCompletion bit
OnlineAsyncModule
Modu int PK FK
Video varchar(100)
OnlineSyncModule
Modu int PK FK
Link varchar(100)
HybridModules
HybridModuleID int PK FK
ComponentModuleID int PK FK
Courses
Studies
StudiesID int PK
StudiesName nvarchar(30)
Syllabus nvarchar(max)
Tuition money
StudentStudiesDetails
StudentID int PK FK
StudiesID int PK FK
Grade decimal(2,1) FK
StudentInternshipDetails
InternshipID int PK FK
StudentID int PK FK
CompletedDays int
GradesList
Grade decimal(2,1) PK
Description nvarchar(20)
Subjects
SubjectID int PK
SubjectName nvarchar(30)
Description nvarchar(max)
StudiesID int FK
MeetingAttendance
MeetingID int PK FK
StudentID int PK FK
MeetingCompletion bit
OnlineAsyncMeeting
MeetingID int PK FK
VideoLink varchar(100)
AcademicMeeting
MeetingID int PK
MeetingName nvarchar(50)
SubjectID int FK
Date datetime
TeacherID int FK
TranslatorID int N FK
LanguageID int FK
UnitPrice money
OnlineSyncMeeting
MeetingID int PK FK
Link varchar(100)
Internships
InternshipID int PK
StudiesID int FK
InternshipNa nvarchar(30)
StartingDate datetime
Description nvarchar(max)
StudentSubjectDetails
StudentID int PK FK
SubjectID int PK FK
Grade decimal(2,1) FK
StationaryMeeting
MeetingID int PK FK
LocationID int FK
HybridMeetings
HybridMeetingID int PK FK
ComponentMeetingID int PK FK
Studies
Locations
LocationID int PK
Address varchar(150)
Room varchar(10)
MaxPeople int
Locations

Wkład

Kacper Feliks: schemat bazy danych, widoki, indeksy, role i uprawnienia, dokumentacja.
Aleksander Jóźwik: schemat bazy danych, tabele, warunki integralności, generowanie danych, procedury, dokumentacja.
Karol Półchłopek: schemat bazy danych, tabele, warunki integralności, funkcje, wyzwalacze, dokumentacja.

Kacper Feliks: T = 32%
Aleksander Jóźwik: T = 35%
Karol Półchłopek: T = 33%

Opisy Tabel

People:

1. Students

- Kolumny:
    - StudentID: Unikalne ID studenta, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - FirstName: Imię studenta, typ nvarchar(max), nie może być NULL.
    - LastName: Nazwisko studenta, typ nvarchar(60), nie może być NULL.
    - Address: Adres studenta, typ nvarchar(150), nie może być NULL.
    - City: Miasto zamieszkania, typ nvarchar(30), nie może być NULL.
    - PostalCode: Kod pocztowy, typ varchar(10), nie może być NULL.
    - Country: Kraj zamieszkania, typ nvarchar(30), nie może być NULL.
    - Email: Unikalny adres e-mail, typ varchar(100), zgodny z formatem %_@_%._%, nie może być NULL.
    - PhoneNumber: Unikalny numer telefonu, typ varchar(15), w formacie +123456789, nie może być NULL.
    - IsRegularCustomer: Informacja o regularnym kliencie, typ bit, domyślnie 0, nie może być NULL.
    - GDPRAgreementDate: Data wyrażenia zgody na RODO, typ datetime, domyślnie aktualna data, nie może być NULL.

---

2. Translators

- **Kolumny:**
    - TranslatorID: Unikalne ID tłumacza, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - FirstName: Imię tłumacza, typ nvarchar(30), nie może być NULL.
    - LastName: Nazwisko tłumacza, typ nvarchar(60), nie może być NULL.
    - Email: Unikalny adres e-mail, typ varchar(100), zgodny z formatem %_@_%._%, nie może być NULL.
    - PhoneNumber: Unikalny numer telefonu, typ varchar(15), w formacie +123456789, nie może być NULL.

---

3. TranslatorLanguageDetails

- **Kolumny:**
    - TranslatorID: ID tłumacza, typ int, nie może być NULL.
    - LanguageID: ID języka, typ int, nie może być NULL.
- **Klucz główny: TranslatorID, LanguageID.**

---

4. Languages

- **Kolumny:**
    - LanguageID: Unikalne ID języka, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - LanguageName: Unikalna nazwa języka, typ nvarchar(30), nie może być NULL.

---

5. Teachers

- **Kolumny:**
    - TeacherID: Unikalne ID nauczyciela, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - FirstName: Imię nauczyciela, typ nvarchar(30), nie może być NULL.
    - LastName: Nazwisko nauczyciela, typ nvarchar(60), nie może być NULL.
    - Address: Adres nauczyciela, typ nvarchar(150), nie może być NULL.
    - City: Miasto zamieszkania, typ nvarchar(30), nie może być NULL.
    - Country: Kraj zamieszkania, typ nvarchar(30), nie może być NULL.
    - Email: Unikalny adres e-mail, typ varchar(100), zgodny z formatem %_@_%._%, nie może być NULL.
    - PhoneNumber: Unikalny numer telefonu, typ varchar(15), w formacie +123456789, nie może być NULL.

Webinars

1. WebinarDetails

- **Kolumny:**
    - StudentID: ID studenta, typ int, nie może być NULL.
    - WebinarID: ID webinaru, typ int, nie może być NULL.
    - TerminationDate: Data zakończenia uczestnictwa w webinarze, typ datetime, nie może być NULL.
- **Klucz główny: StudentID, WebinarID.**

---

2. Webinars

- **Kolumny:**
    - WebinarID: Unikalne ID webinaru, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - WebinarName: Nazwa webinaru, typ nvarchar(50), nie może być NULL.

- Price: Cena webinaru, typ money, większa lub równa 0, nie może być NULL.
- TeacherID: ID nauczyciela prowadzącego webinar, typ int, nie może być NULL.
- Link: Link do webinaru, typ varchar(100), nie może być NULL.
- TranslatorID: Opcjonalne ID tłumacza, typ int, może być NULL.
- WebinarDate: Data webinaru, typ datetime, nie może być NULL.
- LanguageID: ID języka webinaru, typ int, nie może być NULL.
- Description: Opis webinaru, typ nvarchar(max), nie może być NULL.

Courses:

1. Courses

- **Kolumny:**
  - CourseID: Unikalne ID kursu, typ int, klucz główny, generowany automatycznie (IDENTITY).
  - CourseName: Unikalna nazwa kursu, typ nvarchar(50), nie może być NULL.
  - Price: Cena kursu, typ money, większa niż 0, nie może być NULL.
  - Description: Opis kursu, typ varchar(max), nie może być NULL.

---

2. CourseStudentDetails

- **Kolumny:**
  - **StudentID: ID studenta, typ int, nie może być NULL.**
  - **CourseID: ID kursu, typ int, nie może być NULL.**
  - **Completed: Informacja o ukończeniu kursu, typ bit, domyślnie 0, nie może być NULL.**
- **Klucz główny: StudentID, CourseID.**

---

3. CourseModules

- **Kolumny:**
  - ModuleID: Unikalne ID modułu, typ int, klucz główny, generowany automatycznie (IDENTITY).
  - ModuleName: Nazwa modułu, typ nvarchar(50), nie może być NULL.
  - CourseID: ID kursu, typ int, nie może być NULL.
  - LanguageID: ID języka modułu, typ int, nie może być NULL.
  - TeacherID: ID prowadzącego nauczyciela, typ int, nie może być NULL.
  - TranslatorID: Opcjonalne ID tłumacza, typ int, może być NULL.
  - Date: Data modułu, typ datetime, nie może być NULL.

---

4. ModuleAttendance

- **Kolumny:**
  - ModuleID: ID modułu, typ int, nie może być NULL.
  - StudentID: ID studenta, typ int, nie może być NULL.
  - ModuleCompletion: Informacja o ukończeniu modułu, typ bit, domyślnie 0, nie może być NULL.
- **Klucz główny: ModuleID, StudentID.**

---

5. HybridModules

- **Kolumny:**
  - HybridModuleID: ID modułu hybrydowego, typ int, nie może być NULL.
  - ComponentModuleID: ID modułu składowego, typ int, nie może być NULL.
- **Klucz główny: HybridModuleID, ComponentModuleID.**

---

6. StationaryModule

- **Kolumny:**
  - ModuleID: ID modułu, typ int, nie może być NULL.
  - LocationID: ID lokalizacji modułu, typ int, nie może być NULL.
- **Klucz główny: ModuleID.**

---

7. OnlineAsyncModule

- **Kolumny:**
  - ModuleID: ID modułu, typ int, nie może być NULL.
  - VideoLink: Link do nagrania wideo, typ varchar(100), nie może być NULL.
- **Klucz główny: ModuleID.**

---

8. OnlineSyncModule

- **Kolumny:**
  - ModuleID: ID modułu, typ int, nie może być NULL.
  - Link: Link do synchronizacji online, typ varchar(100), nie może być NULL.
- **Klucz główny: ModuleID.**

Studies:

1. Studies

- **Kolumny:**
    - StudiesID: Unikalne ID studiów, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - StudiesName: Unikalna nazwa studiów, typ nvarchar(30), nie może być NULL.
    - Syllabus: Program nauczania, typ nvarchar(max), nie może być NULL.
    - Tuition: Czesne, typ money, większe niż 0, nie może być NULL.

---

2. Subjects

- **Kolumny:**
    - SubjectID: Unikalne ID przedmiotu, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - SubjectName: Nazwa przedmiotu, typ nvarchar(30), nie może być NULL.
    - Description: Opis przedmiotu, typ nvarchar(max), nie może być NULL.
    - StudiesID: ID studiów powiązanych z przedmiotem, typ int, nie może być NULL.

---

3. AcademicMeeting

- **Kolumny:**
    - MeetingID: Unikalne ID spotkania, typ int, klucz główny, generowany automatycznie (IDENTITY).
    - MeetingName: Nazwa spotkania, typ nvarchar(50), nie może być NULL.
    - SubjectID: ID powiązanego przedmiotu, typ int, nie może być NULL.
    - Date: Data spotkania, typ datetime, nie może być NULL.
    - TeacherID: ID prowadzącego nauczyciela, typ int, nie może być NULL.
    - TranslatorID: Opcjonalne ID tłumacza, typ int, może być NULL.
    - LanguageID: ID języka, typ int, nie może być NULL.
    - UnitPrice: Cena za udział, typ money, większa niż 0, nie może być NULL.

---

4. MeetingAttendance

- **Kolumny:**
    - MeetingID: ID spotkania, typ int, nie może być NULL.
    - StudentID: ID studenta, typ int, nie może być NULL.
    - MeetingCompletion: Informacja o ukończeniu spotkania, typ bit, domyślnie 0, nie może być NULL.
- **Klucz główny: MeetingID, StudentID.**

---

5. OnlineAsyncMeeting

- **Kolumny:**
    - MeetingID: ID spotkania, typ int, nie może być NULL.
    - VideoLink: Link do nagrania wideo, typ varchar(100), nie może być NULL.
- **Klucz główny: MeetingID.**

---

6. OnlineSyncMeeting

- **Kolumny:**
    - MeetingID: ID spotkania, typ int, nie może być NULL.
    - Link: Link do synchronizacji online, typ varchar(100), nie może być NULL.
- **Klucz główny: MeetingID.**

---

7. StationaryMeeting

- **Kolumny:**
    - MeetingID: ID spotkania, typ int, nie może być NULL.
    - LocationID: ID lokalizacji spotkania, typ int, nie może być NULL.
- **Klucz główny: MeetingID.**

---

8. HybridMeetings

- **Kolumny:**
    - HybridMeetingID: ID spotkania hybrydowego, typ int, nie może być NULL.
    - ComponentMeetingID: ID spotkania składowego, typ int, nie może być NULL.
- **Klucz główny: HybridMeetingID, ComponentMeetingID.**

---

9. Internships

- **Kolumny:**
    - InternshipID: Unikalne ID praktyki, typ int, klucz główny, generowane automatycznie (IDENTITY).
    - StudiesID: ID studiów, typ int, nie może być NULL.
    - InternshipName: Nazwa praktyki, typ nvarchar(30), nie może być NULL.
    - StartingDate: Data rozpoczęcia, typ datetime, nie może być NULL.
    - Description: Opis praktyki, typ nvarchar(max), nie może być NULL.

---

10. StudentInternshipDetails

- **Kolumny:**
  - InternshipID: ID praktyki, typ int, nie może być NULL.
  - StudentID: ID studenta, typ int, nie może być NULL.
  - CompletedDays: Liczba ukończonych dni praktyki, typ int, domyślnie 0, większa lub równa 0, nie może być NULL.
- **Klucz główny: InternshipID, StudentID.**

---

11. StudentStudiesDetails

- **Kolumny:**
  - **StudentID: ID studenta, typ int, nie może być NULL.**
  - **StudiesID: ID studiów, typ int, może być NULL.**
  - **Grade: Ocena uzyskana na studiach, typ decimal(2,1), nie może być NULL.**
- **Klucz główny: StudentID, StudiesID.**

---

12. StudentSubjectDetails

- **Kolumny:**
  - **StudentID: ID studenta, typ int, nie może być NULL.**
  - **SubjectID: ID przedmiotu, typ int, nie może być NULL.**
  - **Grade: Uzyskana ocena, typ decimal(2,1), może być NULL.**
- **Klucz główny: StudentID, SubjectID.**

---

13. GradesList

- **Kolumny:**
  - Grade: Ocena, typ decimal(2,1), unikalna, z zakresu 2.0–5.0, nie może być NULL.
  - Description: Opis oceny, typ nvarchar(20), nie może być NULL.

Orders:

1. Orders

- **Kolumny:**
  - OrderID: Unikalne ID zamówienia, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - StudentID: ID studenta składającego zamówienie, typ int, nie może być NULL.

- OrderDate: Data złożenia zamówienia, typ datetime, nie może być NULL.
- FullyPaidDate: Data pełnej płatności, typ datetime, może być NULL.
- PaymentLink: Link do płatności, typ nvarchar(100), nie może być NULL.
- CurrencyID: ID waluty zamówienia, typ int, nie może być NULL.
- vatID: ID stawki VAT, typ int, nie może być NULL.

---

2. OrderContentDetails

- **Kolumny:**
  - OrderContentID: Unikalne ID treści zamówienia, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - OrderID: ID zamówienia, typ int, nie może być NULL.
- **Klucz główny: OrderContentID.**

---

3. OrderWebinar

- **Kolumny:**
  - OrderContentID: ID treści zamówienia, typ int, nie może być NULL.
  - WebinarID: ID webinaru, typ int, nie może być NULL.
- **Klucz główny: OrderContentID.**

---

4. OrderStudies

- **Kolumny:**
  - OrderContentID: ID treści zamówienia, typ int, nie może być NULL.
  - StudiesID: ID studiów, typ int, nie może być NULL.
- **Klucz główny: OrderContentID.**

---

5. OrderCourse

- **Kolumny:**
  - OrderContentID: ID treści zamówienia, typ int, nie może być NULL.
  - CourseID: ID kursu, typ int, nie może być NULL.
- **Klucz główny: OrderContentID.**

---

6. OrderAcademicMeeting

- **Kolumny:**
  - OrderContentID: ID zamówienia, typ int, nie może być NULL.
  - MeetingID: ID spotkania, typ int, nie może być NULL.
- **Klucz główny: OrderContentID.**

---

7. CurrenciesRate

- **Kolumny:**
  - CurrencyID: Unikalne ID waluty, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - CurrencyRate: Kurs waluty, typ decimal(10,4), większy niż 0, nie może być NULL.
  - CurrencyName: Unikalna nazwa waluty, typ nvarchar(15), nie może być NULL.

---

8. Payments

- **Kolumny:**
  - PaymentID: Unikalne ID płatności, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - PaidAmount: Kwota wpłacona, typ money, większa niż 0, nie może być NULL.
  - OrderID: ID zamówienia powiązanego z płatnością, typ int, nie może być NULL.
  - PaymentDate: Data płatności, typ datetime, nie może być NULL.

---

9. VAT

- **Kolumny:**
  - vatID: Unikalne ID stawki VAT, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - Rate: Stawka VAT w procentach, typ decimal(4,2), z zakresu 0–99.99, nie może być NULL.

Locations:

1. Locations

- **Kolumny:**
  - LocationID: Unikalne ID lokalizacji, typ int, klucz główny, generowane automatycznie (IDENTITY).
  - Address: Adres lokalizacji, typ varchar(150), nie może być NULL.
  - Room: Nazwa/identyfikator pomieszczenia, typ varchar(10), nie może być NULL.
  - MaxPeople: Maksymalna liczba osób, typ int, większa niż 0, nie może być NULL.

Funkcje

UpdateStudentGrade

Procedura UpdateStudentGrade służy do aktualizacji oceny studenta w bazie danych. Weryfikuje ona, czy student oraz kurs istnieją, a także czy student jest zapisany na dany kurs. Jeśli warunki są spełnione, procedura aktualizuje ocenę studenta w tabeli StudentGrades.

```
EXEC UpdateStudentGrade
      @StudentID = 1,
      @CourseID = 101,
      @Grade = 'A';
```

UpdateStudentAddress

Procedura UpdateStudentAddress służy do aktualizacji adresu studenta w bazie danych. Weryfikuje ona, czy student istnieje, a następnie aktualizuje jego adres na podstawie podanych parametrów.

```
EXEC UpdateStudentAddress
      @StudentID = 1,
      @Street = 'Main St',
      @City = 'Warsaw',
      @PostalCode = '00-001';
```

CanEnrollInStudies

Funkcja CanEnrollInStudies służy do sprawdzenia, czy student może zapisać się na określone studia. Weryfikuje ona, czy student jest już zapisany na te studia oraz czy opłata za zapis została uiszczona.

```
SELECT * FROM CanEnrollInStudies(@StudentID = 1, @StudiesID = 101);
```

CanEnrollWebinar
Funkcja CanEnrollWebinar służy do sprawdzenia, czy student może zapisać się na określony webinar. Weryfikuje ona, czy webinar istnieje, czy student jest już zapisany na ten webinar oraz czy opłata za webinar została uiszczona.

```
SELECT dbo.CanEnrollWebinar(@StudentID = 1, @WebinarID = 101);
```

CheckCourseCapacity
Funkcja CheckCourseCapacity służy do sprawdzenia dostępności miejsc dla kursu. Weryfikuje ona, ile miejsc jest dostępnych w danym kursie, biorąc pod uwagę jego typ (online lub stacjonarny/hybrydowy) oraz aktualną liczbę zapisanych studentów.

```
SELECT * FROM CheckCourseCapacity(@CourseID = 101);
```

CheckStudiesCapacity
Funkcja CheckStudiesCapacity służy do sprawdzenia dostępności miejsc dla studiów. Weryfikuje ona, ile miejsc jest dostępnych w danych studiach, biorąc pod uwagę ich typ (online lub stacjonarny/hybrydowy) oraz aktualną liczbę zapisanych studentów.

```
SELECT * FROM CheckStudiesCapacity(@StudiesID = 101);
```

CheckTranslatorAvailability
Funkcja CheckTranslatorAvailability służy do sprawdzenia dostępności tłumaczy dla różnych języków używanych w spotkaniach akademickich, modułach kursów oraz webinarach. Weryfikuje ona, czy dla danego języka jest dostępny tłumacz, ile jest dostępnych tłumaczy oraz zwraca listę tłumaczy.

```
SELECT * FROM CheckTranslatorAvailability();
```

CheckTranslatorAvailabilityForLanguage
Funkcja CheckTranslatorAvailabilityForLanguage służy do sprawdzenia dostępności tłumaczy dla określonego języka używanego w spotkaniach akademickich, modułach kursów oraz webinarach. Weryfikuje ona, czy dla danego języka jest dostępny tłumacz, ile jest dostępnych tłumaczy oraz zwraca listę tłumaczy.

```
SELECT * FROM CheckTranslatorAvailabilityForLanguage(@LanguageID = 1);
```

IsCourseCompleted
Funkcja IsCourseCompleted służy do sprawdzenia, czy student ukończył dany kurs. Weryfikuje ona status ukończenia wszystkich modułów kursu przez studenta, z wyłączeniem modułów typu OnlineAsync, oraz oblicza procent ukończenia kursu.

```
SELECT * FROM IsCourseCompleted(@StudentID = 1, @CourseID = 101);
```

IsCourseModuleCompleted
Funkcja IsCourseModuleCompleted służy do sprawdzenia, czy student ukończył dany moduł kursu. Weryfikuje ona status ukończenia modułu przez studenta oraz sprawdza, czy student jest przypisany do tego modułu.

```
SELECT dbo.IsCourseModuleCompleted(@StudentID = 1, @ModuleID = 101);
```

IsInternshipPassed
Funkcja IsInternshipPassed służy do sprawdzenia, czy student ukończył staż. Weryfikuje ona, czy student przepracował co najmniej 14 dni stażu.

```
SELECT dbo.IsInternshipPassed(@StudentID = 1);
```

IsMeetingCompleted
Funkcja IsMeetingCompleted służy do sprawdzenia, czy student ukończył dane spotkanie akademickie. Weryfikuje ona, czy student ma dostęp do spotkania poprzez zamówienie oraz czy ukończył spotkanie.

```
SELECT dbo.IsMeetingCompleted(@StudentID = 1, @MeetingID = 101);
```

IsOrderPaid
Funkcja IsOrderPaid służy do sprawdzenia, czy zamówienie zostało w pełni opłacone. Weryfikuje ona całkowitą cenę zamówienia oraz sumę wpłaconych kwot, a następnie porównuje te wartości.

```
SELECT dbo.IsOrderPaid(@OrderID = 1);
```

StudentSchedule
Funkcja StudentSchedule służy do pobierania harmonogramu studenta, w tym spotkań akademickich, modułów kursów oraz webinarów, w których uczestniczy.

```
SELECT * FROM dbo.StudentSchedule(@StudentID = 1);
```

TeacherSchedule
Funkcja TeacherSchedule służy do pobierania harmonogramu nauczyciela, w tym spotkań akademickich, modułów kursów oraz webinarów, w których uczestniczy.

```
SELECT * FROM dbo.TeacherSchedule(@TeacherID = 1);
```

TranslatorSchedule
Funkcja TranslatorSchedule służy do pobierania harmonogramu tłumacza, w tym spotkań akademickich, modułów kursów oraz webinarów, w których uczestniczy.

```
SELECT * FROM dbo.TranslatorSchedule(@TranslatorID = 1);
```

Procedury

AddCourse:

Procedura AddCourse służy do dodawania nowego kursu do bazy danych. Weryfikuje ona, czy podane identyfikatory nauczyciela, języka oraz opcjonalnie tłumacza istnieją w odpowiednich tabelach. Jeśli wszystkie warunki są spełnione, wstawia nowy kurs do tabeli Courses.

```
EXEC AddCourse
      @CourseName = N'Podstawy programowania',
      @CourseDescription = N'Kurs wprowadzający do programowania w
Pythonie.',
      @CoursePrice = 199.99,
      @LanguageID = 1,
      @TeacherID = 10,
      @TranslatorID = 5;
```

AddCourseModule:

Procedura AddCourseModule służy do dodawania nowego modułu do istniejącego kursu w bazie danych. Weryfikuje ona, czy podane identyfikatory kursu, nauczyciela, języka oraz opcjonalnie tłumacza istnieją w odpowiednich tabelach. W zależności od typu modułu, wstawia odpowiednie dane do tabeli CourseModules oraz dodatkowych tabel specyficznych dla typu modułu.

```
EXEC AddCourseModule
      @CourseID = 1,
      @ModuleName = N'Wprowadzenie do Pythona',
      @Date = '2024-01-15 10:00:00',
      @TeacherID = 10,
      @LanguageID = 1,
      @ModuleType = 'OnlineSync',
      @Link = 'https://example.com/online-sync',
      @TranslatorID = 5;
```

AddInternship

Procedura AddInternship służy do dodawania nowego stażu do bazy danych. Weryfikuje ona, czy podane StudiesID istnieje w tabeli Studies. Jeśli warunek jest spełniony, wstawia nowy staż do tabeli Internships z podanymi danymi (InternshipName, StartingDate, Description).

```
EXEC AddInternship
      @StudiesID = 2,
      @InternshipName = N'Praktyki letnie',
      @StartingDate = '2024-06-01 09:00:00',
      @Description = N'Praktyki letnie w firmie IT.';
```

AddLanguage
Procedura AddLanguage służy do dodawania nowego języka do bazy danych. Weryfikuje ona, czy język o podanej nazwie już istnieje w tabeli Languages. Jeśli język o podanej nazwie już istnieje, procedura zwraca błąd. W przeciwnym razie, wstawia nowy język do tabeli Languages.

```
EXEC AddLanguage
      @LanguageName = 'Spanish';
```

AddLocation
Procedura AddLocation służy do dodawania nowej lokalizacji do bazy danych. Wstawia ona nową lokalizację do tabeli Locations z podanymi danymi (Address, Room, MaxPeople). W przypadku wystąpienia błędu, procedura zwraca komunikat o błędzie.

```
EXEC AddLocation
      @Address = N'ul. Przykładowa 123, Warszawa',
      @Room = 'A101',
      @MaxPeople = 30;
```

AddOrder
Procedura AddOrder służy do dodawania nowego zamówienia do bazy danych. Weryfikuje ona, czy podany StudentID istnieje w tabeli Students. Jeśli warunek jest spełniony, wstawia nowe zamówienie do tabeli Orders z podanymi danymi (StudentID, OrderDate, PaymentLink, CurrencyID, vatID). Procedura zwraca komunikat o pomyślnym dodaniu zamówienia.

```
EXEC AddOrder
      @StudentID = 1,
      @CurrencyID = 2,
      @vatID = 3,
      @PaymentLink = N'https://example.com/payment';
```

AddOrderContent

Procedura AddOrderContent służy do dodawania szczegółów zamówienia do bazy danych. Weryfikuje ona, czy zamówienie o podanym OrderID istnieje, a następnie sprawdza dostępność miejsc na kursie, studiach, webinarze lub spotkaniu akademickim. Jeśli wszystkie warunki są spełnione, procedura dodaje szczegóły zamówienia do odpowiednich tabel.

```
EXEC AddOrderContent
      @OrderID = 1,
      @ContentType = 'Course',
      @ContentID = 101;
```

AddPayment
Procedura AddPayment służy do dodawania płatności do zamówienia w bazie danych. Weryfikuje ona, czy zamówienie istnieje oraz czy nie zostało już w całości opłacone. Jeśli warunki są spełnione, dodaje płatność do tabeli Payments. Jeśli po dodaniu płatności zamówienie zostanie w całości opłacone, procedura aktualizuje datę pełnej płatności w tabeli Orders oraz dodaje studenta do odpowiednich komponentów zamówienia (kursów, webinarów, studiów, spotkań akademickich).

```
EXEC AddPayment
      @OrderID = 1,
      @PaidAmount = 100.00;
```

AddStudent
Procedura AddStudent służy do dodawania nowego studenta do bazy danych. Wstawia ona nowego studenta do tabeli Students z podanymi danymi (FirstName, LastName, Address, City, PostalCode, Country, Email, PhoneNumber, IsRegularCustomer). Automatycznie ustawia datę zgody na przetwarzanie danych osobowych (GDPRAgreementDate) na bieżącą datę i czas. W przypadku wystąpienia błędu, procedura zwraca komunikat o błędzie.

```
EXEC AddStudent
      @FirstName = N'Jan',
      @LastName = N'Kowalski',
      @Address = N'ul. Przykładowa 123',
      @City = N'Warszawa',
      @PostalCode = '00-001',
      @Country = N'Polska',
      @Email = 'jan.kowalski@example.com',
      @PhoneNumber = '+123456789',
      @IsRegularCustomer = 1;
```

AddStudentToCourse

Procedura AddStudentToCourse służy do dodawania studenta do kursu w bazie danych. Weryfikuje ona, czy student i kurs istnieją oraz czy student może zapisać się na kurs. Jeśli wszystkie warunki są spełnione, procedura dodaje studenta do kursu w tabeli CourseStudentDetails.

```
EXEC AddStudentToCourse
      @StudentID = 1,
      @CourseID = 101;
```

AddStudentToStudies
Procedura AddStudentToStudies służy do dodawania studenta do studiów w bazie danych. Weryfikuje ona, czy student i studia istnieją oraz czy student może zapisać się na studia. Jeśli wszystkie warunki są spełnione, procedura dodaje studenta do studiów oraz do wszystkich przedmiotów powiązanych z danym kierunkiem studiów.

```
EXEC AddStudentToStudies
```

```
      @StudentID = 1,
      @StudiesID = 202;
```

AddStudentToWebinar

Procedura AddStudentToWebinar służy do dodawania studenta do webinaru w bazie danych. Weryfikuje ona, czy student i webinar istnieją oraz czy student może zapisać się na webinar. Jeśli wszystkie warunki są spełnione, procedura dodaje studenta do webinaru w tabeli WebinarDetails.

```
EXEC AddStudentToWebinar
      @StudentID = 1,
      @WebinarID = 303;
```

AddStudy

Procedura AddStudy służy do dodawania nowych studiów do bazy danych. Weryfikuje ona, czy istnieje język o podanym LanguageID oraz, jeśli podano TranslatorID, czy tłumacz jest dostępny dla danego języka. Jeśli warunki są spełnione, procedura wstawia nowe studia do tabeli Studies.

```
EXEC AddStudy
      @StudiesName = N'Informatyka',
      @Syllabus = N'Programowanie, Bazy danych, Sieci komputerowe',
      @Tuition = 5000.00,
      @LanguageID = 1,
      @TranslatorID = 2;
```

AddStudyMeeting

Procedura AddStudyMeeting służy do dodawania nowego spotkania akademickiego do bazy danych. Weryfikuje ona, czy przedmiot, nauczyciel oraz tłumacz (jeśli podano) istnieją oraz czy tłumacz jest dostępny dla danego języka. Procedura obsługuje trzy typy spotkań: stacjonarne, online synchroniczne oraz online asynchroniczne, i wstawia odpowiednie dane do odpowiednich tabel.

```
EXEC AddStudyMeeting
      @SubjectID = 1,
      @TeacherID = 2,
      @MeetingName = N'Wprowadzenie do programowania',
      @UnitPrice = 150.00,
      @Date = '2023-12-01 10:00:00',
      @TranslatorID = 3,
      @LanguageID = 1,
      @MeetingType = N'Stationary',
      @LocationID = 5;
```

AddSubject
Procedura AddSubject służy do dodawania nowego przedmiotu do bazy danych. Weryfikuje ona, czy istnieją studia, nauczyciel oraz język o podanych identyfikatorach. Jeśli podano TranslatorID, procedura sprawdza również dostępność tłumacza dla danego języka. Jeśli wszystkie warunki są spełnione, procedura wstawia nowy przedmiot do tabeli Subjects.

```
EXEC AddSubject
      @StudiesID = 1,
      @SubjectName = N'Algorytmy i Struktury Danych',
      @Description = N'Kurs dotyczący podstawowych algorytmów i struktur
danych',
      @LanguageID = 1,
      @TeacherID = 2,
      @TranslatorID = 3;
```

AddTeacher
Procedura AddTeacher służy do dodawania nowego nauczyciela do bazy danych. Weryfikuje ona, czy wszystkie wymagane dane są dostarczone, a następnie wstawia nowego nauczyciela do tabeli Teachers.

```
EXEC AddTeacher
      @FirstName = N'Jan',
      @LastName = N'Kowalski',
      @Address = N'ul. Przykładowa 1',
      @City = N'Warszawa',
      @Country = N'Polska',
      @Email = 'jan.kowalski@example.com',
      @PhoneNumber = '+123456789';
```

AddTranslator
Procedura AddTranslator służy do dodawania nowego tłumacza do bazy danych. Weryfikuje ona, czy wszystkie wymagane dane są dostarczone, a następnie wstawia nowego tłumacza do tabeli Translators.

```
EXEC AddTranslator
      @FirstName = N'Anna',
      @LastName = N'Nowak',
      @Email = 'anna.nowak@example.com',
      @PhoneNumber = '+987654321';
```

AddTranslatorLanguage
Procedura AddTranslatorLanguage służy do dodawania nowego języka do tłumacza w bazie danych. Weryfikuje ona, czy język o podanej nazwie istnieje oraz czy tłumacz już zna ten język. Jeśli warunki są spełnione, procedura wstawia nowe powiązanie tłumacza z językiem do tabeli TranslatorLanguageDetails.

```
EXEC AddTranslatorLanguage
      @TranslatorID = 1,
      @LanguageName = English;
```

AddWebinar

Procedura AddWebinar służy do dodawania nowego webinaru do bazy danych. Weryfikuje ona, czy nauczyciel, tłumacz (jeśli podano), język oraz link są poprawne i istnieją w bazie danych. Jeśli wszystkie warunki są spełnione, procedura wstawia nowy webinar do tabeli Webinars.

Procedura AddWebinar służy do dodawania nowego webinaru do bazy danych. Weryfikuje ona, czy nauczyciel, tłumacz (jeśli podano), język oraz link są poprawne i istnieją w bazie danych. Jeśli wszystkie warunki są spełnione, procedura wstawia nowy webinar do tabeli Webinars.

```
EXEC AddWebinar
      @WebinarName = N'Podstawy SQL',
      @Price = 200.00,
      @TeacherID = 1,
      @Link = 'https://example.com/webinar',
      @TranslatorID = 2,
      @WebinarDate = '2023-12-15 14:00:00',
      @LanguageID = 1,
      @Description = N'Webinar dotyczący podstaw SQL';
```

SetCourseCompletion

Procedura SetCourseCompletion służy do ustawiania statusu ukończenia kursu przez studenta w bazie danych. Weryfikuje ona, czy student i kurs istnieją oraz czy student jest zapisany na dany kurs. Jeśli warunki są spełnione, procedura aktualizuje status ukończenia kursu w tabeli CourseStudentDetails.

```
EXEC SetCourseCompletion
      @StudentID = 1,
      @CourseID = 101,
      @Completed = 1;
```

SetStudentMeetingAttendance

Procedura SetStudentMeetingAttendance służy do ustawiania obecności studenta na spotkaniu akademickim w bazie danych. Weryfikuje ona, czy student, spotkanie oraz powiązane studia istnieją, a także czy spotkanie jest typu stacjonarnego, online synchronicznego lub hybrydowego. Jeśli warunki są spełnione, procedura dodaje lub aktualizuje obecność studenta na spotkaniu w tabeli MeetingAttendance.

```
EXEC SetStudentMeetingAttendance
      @StudentID = 1,
      @MeetingID = 101,
      @Attendance = 1;
```

SetStudentModuleAttendance

Procedura SetStudentModuleAttendance służy do ustawiania obecności studenta na module kursu w bazie danych. Weryfikuje ona, czy student, moduł oraz powiązany kurs istnieją, a także czy moduł jest typu stacjonarnego, online synchronicznego lub hybrydowego. Jeśli warunki są spełnione, procedura dodaje lub aktualizuje obecność studenta na module w tabeli ModuleAttendance.

```
EXEC SetStudentModuleAttendance
      @StudentID = 1,
      @ModuleID = 101,
      @Attendance = 1;
```

SetStudentCourseGrade

Procedura SetStudentCourseGrade służy do ustawiania oceny studenta za kurs w bazie danych. Weryfikuje ona, czy student i kurs istnieją oraz czy student jest zapisany na dany kurs. Dodatkowo sprawdza sumaryczną obecność studenta na wszystkich spotkaniach związanych z kursem i, jeśli obecność jest mniejsza niż 80%, nadpisuje ocenę na 2.0. Jeśli warunki są spełnione, procedura aktualizuje ocenę studenta w tabeli StudentCourseDetails.

```
EXEC SetStudentCourseGrade
      @StudentID = 1,
      @CourseID = 101,
      @Grade = 4.5;
```

UpdateHybridMeetingAttendance

Procedura UpdateHybridMeetingAttendance służy do aktualizacji obecności studenta na spotkaniu hybrydowym w bazie danych. Weryfikuje ona, czy student, spotkanie hybrydowe oraz powiązane studia istnieją, a także czy student był obecny na wszystkich wymaganych spotkaniach komponentowych. Jeśli warunki są spełnione, procedura dodaje lub aktualizuje obecność studenta na spotkaniu hybrydowym w tabeli MeetingAttendance.

```
EXEC UpdateHybridMeetingAttendance
      @StudentID = 1,
      @HybridMeetingID = 101;
```

UpdateHybridModuleAttendance

Procedura UpdateHybridModuleAttendance służy do aktualizacji obecności studenta na module hybrydowym w bazie danych. Weryfikuje ona, czy student, moduł hybrydowy oraz powiązany kurs istnieją, a także czy student był obecny na wszystkich wymaganych modułach komponentowych. Jeśli warunki są spełnione, procedura dodaje lub aktualizuje obecność studenta na module hybrydowym w tabeli ModuleAttendance.

```
EXEC UpdateHybridModuleAttendance
      @StudentID = 1,
      @HybridModuleID = 101;
```

UpdateInternshipCompletedDays

Procedura UpdateInternshipCompletedDays służy do aktualizacji liczby dni ukończonych przez studenta na stażu w bazie danych. Weryfikuje ona, czy student i staż istnieją oraz czy student jest zapisany na studia związane ze stażem. Jeśli student nie jest przypisany do stażu, procedura przypisuje go i ustawia początkową liczbę ukończonych dni na 0. Następnie aktualizuje liczbę ukończonych dni, dodając wartość z parametru @AdditionalDays.

```
EXEC UpdateInternshipCompletedDays
      @StudentID = 1,
      @InternshipID = 101,
      @AdditionalDays = 5;
```

UpdateOrInsertCurrencyRate

Procedura UpdateOrInsertCurrencyRate służy do aktualizacji lub dodania kursu waluty w bazie danych. Weryfikuje ona, czy waluta już istnieje w tabeli CurrenciesRate. Jeśli waluta istnieje, procedura aktualizuje jej kurs. W przeciwnym razie, dodaje nową walutę z podanym kursem.

```
EXEC UpdateOrInsertCurrencyRate
      @CurrencyName = 'USD',
      @CurrencyRate = 4.1234;
```

UpdateOrInsertVAT

Procedura UpdateOrInsertVAT służy do aktualizacji lub dodania stawki VAT w bazie danych. Weryfikuje ona, czy vatID jest podane. Jeśli vatID jest NULL, procedura dodaje nową stawkę VAT. W przeciwnym razie, aktualizuje istniejącą stawkę VAT.

```
EXEC UpdateOrInsertVAT
      @vatID = NULL,
      @Rate = 23.00;
```

Wyzwalacze

---

trg_AddStudentToWebinar

Trigger trg_AddStudentToWebinar jest uruchamiany po dodaniu nowego rekordu do tabeli OrderWebinar. Jego celem jest automatyczne dodanie studenta do webinaru, jeśli spełnione są określone warunki: student nie jest już zapisany na ten webinar, a zamówienie jest w pełni opłacone. Jeśli warunki są spełnione, student jest dodawany do tabeli WebinarDetails z datą zakończenia ustawioną na 30 dni od daty bieżącej.

```
CREATE TRIGGER trg_AddStudentToWebinar
ON OrderWebinar
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	-- Check if student is already enrolled
	IF EXISTS (
	SELECT 1
	FROM WebinarDetails wd
	JOIN inserted i ON wd.WebinarID = i.WebinarID
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE wd.StudentID = o.StudentID
	)
	BEGIN
	RAISERROR ('Student jest już zapisany na ten webinar.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Check if the order is fully paid by checking FullyPaidDate
	IF EXISTS (
	SELECT 1
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE o.FullyPaidDate IS NULL
	)
	BEGIN
	RAISERROR ('Zamówienie musi być w pełni opłacone przed zapisem na
webinar.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- If all checks pass, insert data into WebinarDetails
	INSERT INTO WebinarDetails (StudentID, WebinarID, TerminationDate)
	SELECT
	o.StudentID,
	i.WebinarID,
	DATEADD(DAY, 30, GETDATE())
	FROM inserted i
```

```
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	JOIN Webinars w ON i.WebinarID = w.WebinarID;
END;
```

trg_AddStudentToStudies

Trigger trg_AddStudentToStudies jest uruchamiany po dodaniu nowego rekordu do tabeli OrderStudies. Jego celem jest automatyczne dodanie studenta do studiów, jeśli zamówienie jest w pełni opłacone i student nie jest już zapisany na te studia. Jeśli warunki są spełnione, student jest dodawany do tabeli StudentStudiesDetails z początkową oceną 2.0, a także do wszystkich przedmiotów, spotkań akademickich i praktyk związanych z tymi studiami.

```
CREATE TRIGGER trg_AddStudentToStudies
ON OrderStudies
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	-- Check if the order is fully paid
	IF EXISTS (
	SELECT 1
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE o.FullyPaidDate IS NULL
	)
	BEGIN
	RAISERROR ('Zamówienie musi być w pełni opłacone przed zapisem na
studia.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Check if student is already enrolled in these studies
	IF EXISTS (
	SELECT 1
	FROM StudentStudiesDetails ssd
	JOIN inserted i ON ssd.StudiesID = i.StudiesID
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE ssd.StudentID = o.StudentID
```

```
	)
	BEGIN
	RAISERROR ('Student jest już zapisany na te studia.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Begin transaction
	BEGIN TRANSACTION;

	BEGIN TRY
	-- Get StudentID and StudiesID for further operations
	DECLARE @StudentID int, @StudiesID int;
	SELECT
		@StudentID = o.StudentID,
		@StudiesID = i.StudiesID
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID;

	-- Add student to StudentStudiesDetails with initial grade 2.0
	INSERT INTO StudentStudiesDetails (StudentID, StudiesID, Grade)
	VALUES (@StudentID, @StudiesID, 2.0);

	-- Add student to all subjects associated with the studies
	INSERT INTO StudentSubjectDetails (StudentID, SubjectID, Grade)
	SELECT
		@StudentID,
		s.SubjectID,
		NULL -- Initial grade is NULL
	FROM Subjects s
	WHERE s.StudiesID = @StudiesID;

	-- Add student to all academic meetings for these subjects
	INSERT INTO MeetingAttendance (StudentID, MeetingID,
MeetingCompletion)
	SELECT
		@StudentID,
		am.MeetingID,
		0 -- Not completed initially
	FROM AcademicMeeting am
	JOIN Subjects s ON am.SubjectID = s.SubjectID
	WHERE s.StudiesID = @StudiesID
	AND am.Date >= GETDATE(); -- Only future meetings
```

```
	-- Add student to internships associated with the studies
	INSERT INTO StudentInternshipDetails (StudentID, InternshipID,
CompletedDays)
	SELECT
		@StudentID,
		i.InternshipID,
		0 -- Initial completed days
	FROM Internships i
	WHERE i.StudiesID = @StudiesID
	AND i.StartingDate >= GETDATE(); -- Only future internships

	COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
	ROLLBACK TRANSACTION;
	DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
	RAISERROR ('Błąd podczas dodawania studenta do studiów: %s', 16, 1,
@ErrorMessage);
	RETURN;
	END CATCH;
END;
```

trg_AddStudentToCourse

Trigger trg_AddStudentToCourse jest uruchamiany po dodaniu nowego rekordu do tabeli OrderCourse. Jego celem jest automatyczne dodanie studenta do kursu, jeśli zamówienie jest w pełni opłacone i student nie jest już zapisany na ten kurs. Jeśli warunki są spełnione, student jest dodawany do tabeli CourseStudentDetails oraz do wszystkich modułów kursu w tabeli ModuleAttendance.

```
CREATE TRIGGER trg_AddStudentToCourse
ON OrderCourse
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	-- Check if the order is fully paid
	IF EXISTS (
	SELECT 1
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE o.FullyPaidDate IS NULL
	)
	BEGIN
```

```
	RAISERROR ('Zamówienie musi być w pełni opłacone przed zapisem na
kurs.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Check if student is already enrolled in the course
	IF EXISTS (
	SELECT 1
	FROM CourseStudentDetails csd
	JOIN inserted i ON csd.CourseID = i.CourseID
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE csd.StudentID = o.StudentID
	)
	BEGIN
	RAISERROR ('Student jest już zapisany na ten kurs.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Begin transaction for multiple inserts
	BEGIN TRANSACTION;

	BEGIN TRY
	-- Add student to CourseStudentDetails
	INSERT INTO CourseStudentDetails (StudentID, CourseID, Completed)
	SELECT
		o.StudentID,
		i.CourseID,
		0 -- Not completed initially
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID;

	-- Add student to all course modules in ModuleAttendance
	INSERT INTO ModuleAttendance (StudentID, ModuleID,
ModuleCompletion)
	SELECT
		o.StudentID,
		cm.ModuleID,
		0 -- Not completed initially
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
```

```
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	JOIN CourseModules cm ON i.CourseID = cm.CourseID;

	COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
	ROLLBACK TRANSACTION;
	DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
	RAISERROR ('Błąd podczas dodawania studenta do kursu: %s', 16, 1,
@ErrorMessage);
	RETURN;
	END CATCH;
END;
```

trg_AddStudentToAcademicMeeting

Trigger trg_AddStudentToAcademicMeeting jest uruchamiany po dodaniu nowego rekordu do tabeli OrderAcademicMeeting. Jego celem jest automatyczne dodanie studenta do spotkania akademickiego, jeśli zamówienie jest w pełni opłacone i student nie jest już zapisany na to spotkanie. Jeśli warunki są spełnione, student jest dodawany do tabeli MeetingAttendance.

```
CREATE TRIGGER trg_AddStudentToAcademicMeeting
ON OrderAcademicMeeting
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	-- Check if the order is fully paid
	IF EXISTS (
	SELECT 1
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE o.FullyPaidDate IS NULL
	)
	BEGIN
	RAISERROR ('Zamówienie musi być w pełni opłacone przed zapisem na
spotkanie.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Check if student is already enrolled in the meeting
```

```
	IF EXISTS (
	SELECT 1
	FROM MeetingAttendance ma
	JOIN inserted i ON ma.MeetingID = i.MeetingID
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID
	WHERE ma.StudentID = o.StudentID
	)
	BEGIN
	RAISERROR ('Student jest już zapisany na to spotkanie.', 16, 1);
	ROLLBACK TRANSACTION;
	RETURN;
	END

	-- Begin transaction
	BEGIN TRANSACTION;

	BEGIN TRY
	-- Add student to MeetingAttendance
	INSERT INTO MeetingAttendance (MeetingID, StudentID,
MeetingCompletion)
	SELECT
		i.MeetingID,
		o.StudentID,
		0 -- Not completed initially
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderContentID =
ocd.OrderContentID
	JOIN Orders o ON ocd.OrderID = o.OrderID;

	COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
	ROLLBACK TRANSACTION;
	DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
	RAISERROR ('Błąd podczas dodawania studenta do spotkania: %s', 16,
1, @ErrorMessage);
	RETURN;
	END CATCH;
END;
```

TR_PreventStudentDeletion

Trigger TR_PreventStudentDeletion jest uruchamiany zamiast usunięcia rekordu z tabeli Students. Jego celem jest zapobieganie usunięciu studenta, który ma aktywne zamówienia lub kursy. Jeśli student ma aktywne zamówienia lub kursy, trigger zgłasza błąd i zapobiega usunięciu rekordu.

```
CREATE TRIGGER TR_PreventStudentDeletion
ON Students
INSTEAD OF DELETE
AS
BEGIN
      IF EXISTS (
      SELECT 1
      FROM deleted d
      INNER JOIN Orders o ON o.StudentID = d.StudentID
      WHERE o.FullyPaidDate IS NULL
      )
      BEGIN
      THROW 50001, 'Cannot delete student with active orders.', 1;
      RETURN;
      END

      IF EXISTS (
      SELECT 1
      FROM deleted d
      INNER JOIN CourseStudentDetails csd ON csd.StudentID = d.StudentID
      WHERE csd.Completed = 0
      )
      BEGIN
      THROW 50002, 'Cannot delete student with active courses.', 1;
      RETURN;
      END

      DELETE FROM Students
      WHERE StudentID IN (SELECT StudentID FROM deleted);
END;
```

trg_ProcessFreeWebinarOrder

Trigger trg_ProcessFreeWebinarOrder jest uruchamiany po dodaniu nowego rekordu do tabeli Orders. Jego celem jest automatyczne przetworzenie zamówienia na darmowy webinar. Jeśli zamówienie dotyczy darmowego webinaru, trigger dodaje studenta do tabeli WebinarDetails z datą zakończenia ustawioną na 30 dni od daty bieżącej i ustawia pole FullyPaidDate na bieżącą datę.

```sql
CREATE TRIGGER trg_ProcessFreeWebinarOrder
ON Orders
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	-- Najpierw sprawdzamy czy to zamówienie na darmowy webinar
	IF NOT EXISTS (
	SELECT 1
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderID = ocd.OrderID
	JOIN OrderWebinar ow ON ocd.OrderContentID = ow.OrderContentID
	JOIN Webinars w ON ow.WebinarID = w.WebinarID
	WHERE w.Price = 0
	)
	BEGIN
	RETURN;
	END

	BEGIN TRANSACTION;

	BEGIN TRY
	-- Pobierz StudentID i WebinarID z zamówienia
	DECLARE @StudentID INT, @WebinarID INT;

	SELECT
		@StudentID = i.StudentID,
		@WebinarID = w.WebinarID
	FROM inserted i
	JOIN OrderContentDetails ocd ON i.OrderID = ocd.OrderID
	JOIN OrderWebinar ow ON ocd.OrderContentID = ow.OrderContentID
	JOIN Webinars w ON ow.WebinarID = w.WebinarID
	WHERE w.Price = 0;

	-- Sprawdź czy można zapisać studenta na webinar
	IF dbo.canEnrollWebinar(@StudentID, @WebinarID) = 0
	BEGIN
		ROLLBACK TRANSACTION;
		RETURN;
	END

	-- Dodaj WebinarDetails z terminem ważności 30 dni od daty zapisu
	INSERT INTO WebinarDetails (StudentID, WebinarID, TerminationDate)
	VALUES (
		@StudentID,
```

```
            @WebinarID,
            DATEADD(DAY, 30, GETDATE())
      );

      -- Ustaw FullyPaidDate dla zamówienia
      UPDATE Orders
      SET FullyPaidDate = GETDATE()
      FROM Orders o
      JOIN inserted i ON o.OrderID = i.OrderID;

      COMMIT TRANSACTION;
      END TRY
      BEGIN CATCH
      ROLLBACK TRANSACTION;
      DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
      RAISERROR ('Błąd podczas przetwarzania zamówienia na darmowy
webinar: %s', 16, 1, @ErrorMessage);
      RETURN;
      END CATCH;
END;
```

Widoki

FinancialReports:

Widok przedstawia łączne przychody dla webinariów, kursów, studiów oraz spotkań akademickich, grupując dane według nazw wydarzeń.

```
CREATE VIEW FinancialReports AS
SELECT
    'Webinar' AS EventType,
    W.WebinarName AS EventName,
    SUM(P.PaidAmount) AS TotalRevenue
FROM Webinars W
JOIN OrderWebinar OW ON W.WebinarID = OW.WebinarID
JOIN OrderContentDetails OCD ON OW.OrderContentID = OCD.OrderContentID
JOIN Payments P ON OCD.OrderID = P.OrderID
GROUP BY W.WebinarName
UNION ALL
SELECT
    'Course' AS EventType,
    C.CourseName AS EventName,
    SUM(P.PaidAmount) AS TotalRevenue
FROM Courses C
C

JOIN OrderCourse OC ON C.CourseID = OC.CourseID
JOIN OrderContentDetails OCD ON OC.OrderContentID = OCD.OrderContentID
JOIN Payments P ON OCD.OrderID = P.OrderID
GROUP BY C.CourseName
UNION ALL
SELECT
    'Studies' AS EventType,
    S.StudiesName AS EventName,
    SUM(P.PaidAmount) AS TotalRevenue
FROM Studies S
JOIN OrderStudies OS ON S.StudiesID = OS.StudiesID
JOIN OrderContentDetails OCD ON OS.OrderContentID = OCD.OrderContentID
JOIN Payments P ON OCD.OrderID = P.OrderID
GROUP BY S.StudiesName
UNION ALL
SELECT
    'Academic Meeting' AS EventType,
    AM.MeetingName AS EventName,
    SUM(P.PaidAmount) AS TotalRevenue
FROM AcademicMeeting AM
JOIN OrderAcademicMeeting OAM ON AM.MeetingID = OAM.MeetingID
JOIN OrderContentDetails OCD ON OAM.OrderContentID = OCD.OrderContentID
JOIN Payments P ON OCD.OrderID = P.OrderID
GROUP BY AM.MeetingName;
```

Debtors:

Widok zawiera listę studentów, którzy mają zamówienia bez daty pełnej płatności.

```
CREATE VIEW Debtors AS
SELECT DISTINCT
    S.StudentID,
    S.FirstName,
    S.LastName,
    O.OrderID,
    O.OrderDate
FROM Students S
JOIN Orders O ON S.StudentID = O.StudentID
WHERE O.FullyPaidDate IS NULL;
```

FutureEventsEnrollment:

Widok prezentuje informacje o zapisach na przyszłe wydarzenia, takie jak webinaria i kursy, wraz z liczbą zapisanych uczestników.

```
CREATE VIEW FutureEventsEnrollment AS
SELECT
    'Webinar' AS EventType,
    W.WebinarName AS EventName,
    W.WebinarDate AS EventDate,
    'Online' AS EventMode,
    COUNT(DISTINCT WD.StudentID) AS EnrolledCount
FROM Webinars W
JOIN WebinarDetails WD ON W.WebinarID = WD.WebinarID
WHERE W.WebinarDate > GETDATE()
GROUP BY W.WebinarName, W.WebinarDate
UNION ALL
SELECT
    'Course' AS EventType,
    C.CourseName AS EventName,
    M.Date AS EventDate,
    CASE WHEN SM.LocationID IS NOT NULL THEN 'Stationary' ELSE 'Online'
END AS EventMode,
    COUNT(DISTINCT MAD.StudentID) AS EnrolledCount
FROM Courses C
JOIN CourseModules M ON C.CourseID = M.CourseID
LEFT JOIN StationaryModule SM ON M.ModuleID = SM.ModuleID
LEFT JOIN ModuleAttendance MAD ON M.ModuleID = MAD.ModuleID
WHERE M.Date > GETDATE()
GROUP BY C.CourseName, M.Date, SM.LocationID;
```

PastEventsAttendance:

Widok przedstawia dane o frekwencji i uczestnictwie w zakończonych webinariach oraz kursach.

```
CREATE VIEW PastEventsAttendance AS
SELECT
    'Webinar' AS EventType,
    W.WebinarName AS EventName,
    W.WebinarDate AS EventDate,
    COUNT(DISTINCT WD.StudentID) AS TotalParticipants,
    SUM(CASE WHEN WD.TerminationDate IS NOT NULL THEN 1 ELSE 0 END) AS
AttendedCount
FROM Webinars W
JOIN WebinarDetails WD ON W.WebinarID = WD.WebinarID
WHERE W.WebinarDate <= GETDATE()
GROUP BY W.WebinarName, W.WebinarDate
UNION ALL
SELECT
    'Course' AS EventType,
    C.CourseName AS EventName,
    M.Date AS EventDate,
    COUNT(DISTINCT MAD.StudentID) AS TotalParticipants,
    SUM(CASE WHEN MAD.ModuleCompletion = 1 THEN 1 ELSE 0 END) AS
AttendedCount
FROM Courses C
JOIN CourseModules M ON C.CourseID = M.CourseID
LEFT JOIN ModuleAttendance MAD ON M.ModuleID = MAD.ModuleID
WHERE M.Date <= GETDATE()
GROUP BY C.CourseName, M.Date;
```

AttendanceList:

Widok zawiera szczegóły dotyczące obecności studentów na modułach kursów, wskazując status obecności.

```
CREATE VIEW AttendanceList AS
SELECT
    M.ModuleID,
    M.Date AS EventDate,
    S.FirstName,
    S.LastName,
    CASE WHEN MA.ModuleCompletion = 1 THEN 'Present' ELSE 'Absent' END
AS AttendanceStatus
FROM CourseModules M
JOIN ModuleAttendance MA ON M.ModuleID = MA.ModuleID
JOIN Students S ON MA.StudentID = S.StudentID;
```

OverlappingEnrollments:
Widok identyfikuje studentów zapisanych na nakładające się terminy modułów kursów.

```
CREATE VIEW OverlappingEnrollments AS
SELECT
    S.StudentID,
    S.FirstName,
    S.LastName,
    COUNT(DISTINCT M1.ModuleID) AS OverlappingCount
FROM Students S
JOIN ModuleAttendance MA1 ON S.StudentID = MA1.StudentID
JOIN CourseModules M1 ON MA1.ModuleID = M1.ModuleID
JOIN ModuleAttendance MA2 ON S.StudentID = MA2.StudentID
JOIN CourseModules M2 ON MA2.ModuleID = M2.ModuleID
WHERE M1.Date = M2.Date AND M1.ModuleID <> M2.ModuleID
GROUP BY S.StudentID, S.FirstName, S.LastName
HAVING COUNT(DISTINCT M1.ModuleID) > 1;
```

View_Course_Diplomas:
Widok prezentuje listę studentów, którzy ukończyli kursy, wraz z informacją o rodzaju otrzymanego dyplomu.

```
CREATE VIEW View_Course_Diplomas AS
SELECT
    s.StudentID,
    s.FirstName,
    s.LastName,
    s.Address AS CorrespondenceAddress,
    c.CourseName AS DiplomaType
FROM
    Students s
JOIN
    CourseStudentDetails csd ON s.StudentID = csd.StudentID
JOIN
    Courses c ON csd.CourseID = c.CourseID
JOIN
    ModuleAttendance cm ON cm.StudentID = s.StudentID
WHERE
    cm.ModuleCompletion = 1;
```

View_Study_Diplomas:
Widok zawiera listę studentów, którzy ukończyli studia, z uwzględnieniem ich końcowej oceny.

```sql
CREATE VIEW View_Study_Diplomas AS
SELECT
    s.StudentID,
    s.FirstName,
    s.LastName,
    s.Address AS CorrespondenceAddress,
    st.StudiesName AS DiplomaType,
    ss.Grade AS FinalGrade
FROM
    Students s
JOIN
    StudentStudiesDetails ss ON s.StudentID = ss.StudentID
JOIN
    Studies st ON ss.StudiesID = st.StudiesID
WHERE
    ss.Grade IS NOT NULL;
```

ModuleAttendanceView:

Widok przedstawia dane dotyczące obecności studentów na modułach kursów, uwzględniając typ modułu (stacjonarny lub online).

```sql
CREATE VIEW ModuleAttendanceView AS
SELECT
    ma.ModuleID,
    m.ModuleName,
    ma.StudentID,
    s.FirstName AS StudentFirstName,
    s.LastName AS StudentLastName,
    ma.ModuleCompletion,
    CASE
        WHEN sm.LocationID IS NOT NULL THEN 'Stationary'
        WHEN osm.Link IS NOT NULL THEN 'OnlineSync'
        ELSE 'Unknown'
    END AS ModuleType
FROM ModuleAttendance ma
JOIN Students s ON ma.StudentID = s.StudentID
JOIN CourseModules m ON ma.ModuleID = m.ModuleID
LEFT JOIN StationaryModule sm ON ma.ModuleID = sm.ModuleID
LEFT JOIN OnlineSyncModule osm ON ma.ModuleID = osm.ModuleID
WHERE sm.LocationID IS NOT NULL OR osm.Link IS NOT NULL;
```

MeetingAttendanceView:

Widok prezentuje szczegóły obecności studentów na spotkaniach akademickich, uwzględniając ich formę (stacjonarna lub online).

```sql
CREATE VIEW MeetingAttendanceView AS
SELECT
    ma.MeetingID,
    am.MeetingName,
    ma.StudentID,
    s.FirstName AS StudentFirstName,
    s.LastName AS StudentLastName,
    ma.MeetingCompletion,
    CASE
        WHEN sm.LocationID IS NOT NULL THEN 'Stationary'
        WHEN osm.Link IS NOT NULL THEN 'OnlineSync'
        ELSE 'Unknown'
    END AS MeetingType
FROM MeetingAttendance ma
JOIN Students s ON ma.StudentID = s.StudentID
JOIN AcademicMeeting am ON ma.MeetingID = am.MeetingID
LEFT JOIN StationaryMeeting sm ON ma.MeetingID = sm.MeetingID
LEFT JOIN OnlineSyncMeeting osm ON ma.MeetingID = osm.MeetingID
WHERE sm.LocationID IS NOT NULL OR osm.Link IS NOT NULL;
```

Indeksy

AcademicMeeting

Indeksy te przyspieszają wyszukiwanie spotkań akademickich według przedmiotu, nauczyciela, tłumacza oraz języka.

```sql
-- Indeksy dla tabeli AcademicMeeting
CREATE INDEX idx_AcademicMeeting_SubjectID ON AcademicMeeting
(SubjectID);
CREATE INDEX idx_AcademicMeeting_TeacherID ON AcademicMeeting
(TeacherID);
CREATE INDEX idx_AcademicMeeting_TranslatorID ON AcademicMeeting
(TranslatorID);
CREATE INDEX idx_AcademicMeeting_LanguageID ON AcademicMeeting
(LanguageID);
```

CourseModules

Indeksy te przyspieszają wyszukiwanie modułów kursów według kursu, języka, nauczyciela oraz tłumacza.

```sql
-- Indeksy dla tabeli CourseModules
CREATE INDEX idx_CourseModules_CourseID ON CourseModules (CourseID);
```

```
CREATE INDEX idx_CourseModules_LanguageID ON CourseModules (LanguageID);
CREATE INDEX idx_CourseModules_TeacherID ON CourseModules (TeacherID);
CREATE INDEX idx_CourseModules_TranslatorID ON CourseModules
(TranslatorID);
```

CourseStudentDetails

Indeksy te przyspieszają wyszukiwanie szczegółów kursów według kursu oraz studenta.

```
-- Indeksy dla tabeli CourseStudentDetails
CREATE INDEX idx_CourseStudentDetails_CourseID ON CourseStudentDetails
(CourseID);
CREATE INDEX idx_CourseStudentDetails_StudentID ON CourseStudentDetails
(StudentID);
```

Orders

Indeksy te przyspieszają wyszukiwanie zamówień według studenta, waluty oraz VAT.

```
-- Indeksy dla tabeli Orders
CREATE INDEX idx_Orders_StudentID ON Orders (StudentID);
CREATE INDEX idx_Orders_CurrencyID ON Orders (CurrencyID);
CREATE INDEX idx_Orders_vatID ON Orders (vatID);
CREATE INDEX idx_Orders_OrderDate ON Orders (OrderDate);
CREATE INDEX idx_Orders_FullyPaidDate ON Orders (FullyPaidDate);
```

Payments

Indeksy te przyspieszają wyszukiwanie płatności według zamówienia.

```
-- Indeksy dla tabeli Payments
CREATE INDEX idx_Payments_OrderID ON Payments (OrderID);
CREATE INDEX idx_Payments_PaymentDate ON Payments (PaymentDate);
```

MeetingAttendance

Indeksy te przyspieszają wyszukiwanie obecności na spotkaniach według spotkania oraz studenta.

```
-- Indeksy dla tabeli MeetingAttendance
CREATE INDEX idx_MeetingAttendance_MeetingID ON MeetingAttendance
(MeetingID);
CREATE INDEX idx_MeetingAttendance_StudentID ON MeetingAttendance
(StudentID);
```

ModuleAttendance
Indeksy te przyspieszają wyszukiwanie obecności na modułach według modułu oraz studenta.

```
-- Indeksy dla tabeli ModuleAttendance
CREATE INDEX idx_ModuleAttendance_ModuleID ON ModuleAttendance
(ModuleID);
CREATE INDEX idx_ModuleAttendance_StudentID ON ModuleAttendance
(StudentID);
```

StudentInternshipDetails
Indeksy te przyspieszają wyszukiwanie szczegółów praktyk według praktyki oraz studenta.

```
-- Indeksy dla tabeli StudentInternshipDetails
CREATE INDEX idx_StudentInternshipDetails_InternshipID ON
StudentInternshipDetails (InternshipID);
CREATE INDEX idx_StudentInternshipDetails_StudentID ON
StudentInternshipDetails (StudentID);
```

StudentStudiesDetails
Indeksy te przyspieszają wyszukiwanie szczegółów studiów według studenta oraz studiów.

```
-- Indeksy dla tabeli StudentStudiesDetails
CREATE INDEX idx_StudentStudiesDetails_StudentID ON
StudentStudiesDetails (StudentID);
CREATE INDEX idx_StudentStudiesDetails_StudiesID ON
StudentStudiesDetails (StudiesID);
```

StudentSubjectDetails
Indeksy te przyspieszają wyszukiwanie szczegółów przedmiotów według studenta oraz przedmiotu.

```
-- Indeksy dla tabeli StudentSubjectDetails
CREATE INDEX idx_StudentSubjectDetails_StudentID ON
StudentSubjectDetails (StudentID);
CREATE INDEX idx_StudentSubjectDetails_SubjectID ON
StudentSubjectDetails (SubjectID);
```

WebinarDetails
Indeksy te przyspieszają wyszukiwanie szczegółów webinarów według studenta oraz webinaru.

```
-- Indeksy dla tabeli WebinarDetails
CREATE INDEX idx_WebinarDetails_StudentID ON WebinarDetails (StudentID);
CREATE INDEX idx_WebinarDetails_WebinarID ON WebinarDetails (WebinarID);
```

Students
Indeksy te przyspieszają wyszukiwanie studentów według emaila, numeru telefonu oraz nazwiska.

```
-- Indeksy dla tabeli Students
CREATE INDEX idx_Students_Email ON Students (Email);
CREATE INDEX idx_Students_PhoneNumber ON Students (PhoneNumber);
CREATE INDEX idx_Students_LastName ON Students (LastName);
```

Teachers
Indeksy te przyspieszają wyszukiwanie nauczycieli według emaila oraz numeru telefonu.

```
-- Indeksy dla tabeli Teachers
CREATE INDEX idx_Teachers_Email ON Teachers (Email);
CREATE INDEX idx_Teachers_PhoneNumber ON Teachers (PhoneNumber);
```

Translators
Indeksy te przyspieszają wyszukiwanie tłumaczy według emaila oraz numeru telefonu.

```
-- Indeksy dla tabeli Translators
CREATE INDEX idx_Translators_Email ON Translators (Email);
CREATE INDEX idx_Translators_PhoneNumber ON Translators (PhoneNumber);
```

Subjects
Indeks ten przyspiesza wyszukiwanie przedmiotów według studiów.

```
-- Indeksy dla tabeli Subjects
CREATE INDEX idx_Subjects_StudiesID ON Subjects (StudiesID);
```

Internships
Indeksy te przyspieszają wyszukiwanie praktyk według studiów oraz daty rozpoczęcia.

```
-- Indeksy dla tabeli Internships
CREATE INDEX idx_Internships_StudiesID ON Internships (StudiesID);
CREATE INDEX idx_Internships_StartingDate ON Internships (StartingDate);
```

OrderContentDetails
Indeks ten przyspiesza wyszukiwanie szczegółów zawartości zamówienia według zamówienia.

```
-- Indeksy dla tabeli OrderContentDetails
CREATE INDEX idx_OrderContentDetails_OrderID ON OrderContentDetails
(OrderID);
```

OrderWebinar
Indeks ten przyspiesza wyszukiwanie zamówień webinarów według webinaru.

```
-- Indeksy dla tabeli OrderWebinar
CREATE INDEX idx_OrderWebinar_WebinarID ON OrderWebinar (WebinarID);
```

OrderCourse
Indeks ten przyspiesza wyszukiwanie zamówień kursów według kursu.

```
-- Indeksy dla tabeli OrderCourse
CREATE INDEX idx_OrderCourse_CourseID ON OrderCourse (CourseID);
```

OrderAcademicMeeting
Indeks ten przyspiesza wyszukiwanie zamówień spotkań akademickich według spotkania.

```
-- Indeksy dla tabeli OrderAcademicMeeting
CREATE INDEX idx_OrderAcademicMeeting_MeetingID ON OrderAcademicMeeting
(MeetingID);
```

OrderStudies
Indeks ten przyspiesza wyszukiwanie zamówień studiów według studiów.

```
-- Indeksy dla tabeli OrderStudies
CREATE INDEX idx_OrderStudies_StudiesID ON OrderStudies (StudiesID);
```

Webinars
Indeksy te przyspieszają wyszukiwanie webinarów według nauczyciela, tłumacza, języka oraz daty webinaru.

```
-- Indeksy dla tabeli Webinars
CREATE INDEX idx_Webinars_TeacherID ON Webinars (TeacherID);
CREATE INDEX idx_Webinars_TranslatorID ON Webinars (TranslatorID);
CREATE INDEX idx_Webinars_LanguageID ON Webinars (LanguageID);
CREATE INDEX idx_Webinars_WebinarDate ON Webinars (WebinarDate);
```

Locations
Indeks ten przyspiesza wyszukiwanie lokalizacji według adresu.

```
-- Indeksy dla tabeli Locations
CREATE INDEX idx_Locations_Address ON Locations (Address);
```

CurrenciesRate
Indeks ten przyspiesza wyszukiwanie kursów walut według nazwy waluty.

```
-- Indeksy dla tabeli CurrenciesRate
CREATE INDEX idx_CurrenciesRate_CurrencyName ON CurrenciesRate
(CurrencyName);
```

VAT
Indeks ten przyspiesza wyszukiwanie stawek VAT według stawki.

```
-- Indeksy dla tabeli VAT
CREATE INDEX idx_VAT_Rate ON VAT (Rate);
```

GradesList
Indeks ten przyspiesza wyszukiwanie ocen według oceny.

```
-- Indeksy dla tabeli GradesList
CREATE INDEX idx_GradesList_Grade ON GradesList (Grade);
```

Courses
Indeks ten przyspiesza wyszukiwanie kursów według nazwy kursu.

```
-- Indeksy dla tabeli Courses
CREATE INDEX idx_Courses_CourseName ON Courses (CourseName);
```

Role i uprawnienia

Admin
Rola admin ma pełne uprawnienia do wszystkich elementów w schemacie dbo.u_jozwik.

```
CREATE ROLE admin;
GRANT ALL PRIVILEGES ON dbo.u_jozwik TO admin;
```

Dyrektor
Rola dyrektor ma uprawnienia do przeglądania raportów finansowych, listy dłużników, zapisów na przyszłe wydarzenia oraz widoków związanych z obecnością i dyplomami.

```
CREATE ROLE dyrektor;
GRANT SELECT ON Debtors TO dyrektor;
GRANT SELECT ON FinancialReports TO dyrektor;
GRANT SELECT ON FutureEventsEnrollment TO dyrektor;
GRANT SELECT ON MeetingAttendanceView TO dyrektor;
GRANT SELECT ON ModuleAttendanceView TO dyrektor;
GRANT SELECT ON OverlappingEnrollments TO dyrektor;
GRANT SELECT ON PastEventsAttendance TO dyrektor;
GRANT SELECT ON View_Course_Diplomas TO dyrektor;
GRANT SELECT ON View_Study_Diplomas TO dyrektor;
GRANT SELECT ON StudentsSchedule TO dyrektor;
GRANT SELECT ON TranslatorsSchedule TO dyrektor;
GRANT SELECT ON TeachersSchedule TO dyrektor;
```

Księgowy
Rola księgowy ma uprawnienia do przeglądania raportów finansowych, listy dłużników, zapisów na przyszłe wydarzenia oraz zarządzania zamówieniami i płatnościami.

```
CREATE ROLE ksiegowy;
GRANT SELECT ON Debtors TO ksiegowy;
GRANT SELECT ON FinancialReports TO ksiegowy;
GRANT SELECT ON FutureEventsEnrollment TO ksiegowy;
GRANT SELECT ON MeetingAttendanceView TO ksiegowy;
GRANT SELECT ON ModuleAttendanceView TO ksiegowy;
GRANT SELECT ON OverlappingEnrollments TO ksiegowy;
GRANT SELECT ON PastEventsAttendance TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON Orders TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON Payments TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON OrderContentDetails TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON OrderCourse TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON OrderStudies TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON OrderWebinar TO ksiegowy;
GRANT SELECT, INSERT, UPDATE, DELETE ON OrderAcademicMeeting TO
ksiegowy;
```

Nauczyciel
Rola nauczyciel ma uprawnienia do przeglądania i aktualizowania obecności na spotkaniach i modułach, zarządzania webinarami oraz przeglądania harmonogramu nauczycieli.

```
CREATE ROLE nauczyciel;
GRANT SELECT ON MeetingAttendance TO nauczyciel;
GRANT SELECT ON ModuleAttendance TO nauczyciel;
GRANT UPDATE (MeetingCompletion) ON MeetingAttendance TO nauczyciel;
GRANT UPDATE (ModuleCompletion) ON ModuleAttendance TO nauczyciel;
GRANT EXECUTE ON AddWebinar TO nauczyciel;
GRANT SELECT, UPDATE ON Webinars TO nauczyciel;
GRANT SELECT ON TeachersSchedule TO nauczyciel;
GRANT EXECUTE ON TeacherSchedule TO nauczyciel;
```

Koordynator kursu
Rola koordynator_kursu ma uprawnienia do zarządzania kursami, modułami, obecnością oraz webinarami.

```
CREATE ROLE koordynator_kursu;
GRANT SELECT ON ModuleAttendanceView TO koordynator_kursu;
GRANT EXECUTE ON AddCourse TO koordynator_kursu;
GRANT EXECUTE ON AddCourseModule TO koordynator_kursu;
GRANT EXECUTE ON CheckCourseCapacity TO koordynator_kursu;
GRANT SELECT ON MeetingAttendance TO koordynator_kursu;
GRANT SELECT ON ModuleAttendance TO koordynator_kursu;
GRANT UPDATE (MeetingCompletion) ON MeetingAttendance TO
koordynator_kursu;
GRANT UPDATE (ModuleCompletion) ON ModuleAttendance TO
koordynator_kursu;
GRANT EXECUTE ON AddWebinar TO koordynator_kursu;
GRANT SELECT, UPDATE ON Webinars TO koordynator_kursu;
GRANT SELECT ON TeachersSchedule TO koordynator_kursu;
GRANT EXECUTE ON TeacherSchedule TO koordynator_kursu;
```

Koordynator przedmiotu

Rola koordynator_przedmiotu ma uprawnienia do zarządzania przedmiotami, spotkaniami akademickimi, kursami, modułami oraz webinarami.

```
CREATE ROLE koordynator_przedmiotu;
GRANT SELECT ON MeetingAttendanceView TO koordynator_przedmiotu;
GRANT EXECUTE ON AddSubject TO koordynator_przedmiotu;
GRANT EXECUTE ON AddStudyMeeting TO koordynator_przedmiotu;
GRANT SELECT, INSERT, UPDATE, DELETE ON Subjects TO
koordynator_przedmiotu;
GRANT SELECT, INSERT, UPDATE, DELETE ON AcademicMeeting TO
koordynator_przedmiotu;
GRANT INSERT ON StationaryMeeting TO koordynator_przedmiotu;
GRANT INSERT ON OnlineAsyncMeeting TO koordynator_przedmiotu;
GRANT INSERT ON OnlineSyncMeeting TO koordynator_przedmiotu;
GRANT SELECT ON ModuleAttendanceView TO koordynator_przedmiotu;
GRANT EXECUTE ON AddCourse TO koordynator_przedmiotu;
GRANT EXECUTE ON AddCourseModule TO koordynator_przedmiotu;
GRANT EXECUTE ON CheckCourseCapacity TO koordynator_przedmiotu;
GRANT SELECT ON MeetingAttendance TO koordynator_przedmiotu;
GRANT SELECT ON ModuleAttendance TO koordynator_przedmiotu;
GRANT UPDATE (MeetingCompletion) ON MeetingAttendance TO
koordynator_przedmiotu;
GRANT UPDATE (ModuleCompletion) ON ModuleAttendance TO
koordynator_przedmiotu;
GRANT EXECUTE ON AddWebinar TO koordynator_przedmiotu;
GRANT SELECT, UPDATE ON Webinars TO koordynator_przedmiotu;
GRANT SELECT ON TeachersSchedule TO koordynator_przedmiotu;
GRANT EXECUTE ON TeacherSchedule TO koordynator_przedmiotu;
```

Koordynator studiów

Rola koordynator_studiow ma uprawnienia do zarządzania studiami, przedmiotami, spotkaniami akademickimi, kursami, modułami oraz webinarami.

```
CREATE ROLE koordynator_studiow;
GRANT SELECT ON MeetingAttendanceView TO koordynator_studiow;
GRANT EXECUTE ON AddSubject TO koordynator_studiow;
GRANT EXECUTE ON AddStudyMeeting TO koordynator_studiow;
GRANT SELECT, INSERT, UPDATE, DELETE ON Subjects TO koordynator_studiow;
GRANT SELECT, INSERT, UPDATE, DELETE ON AcademicMeeting TO
koordynator_studiow;
GRANT INSERT ON StationaryMeeting TO koordynator_studiow;
GRANT INSERT ON OnlineAsyncMeeting TO koordynator_studiow;
GRANT INSERT ON OnlineSyncMeeting TO koordynator_studiow;
GRANT EXECUTE ON AddStudy TO koordynator_studiow;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Studies TO koordynator_studiow;
GRANT SELECT, INSERT, UPDATE, DELETE ON Internships TO
koordynator_studiow;
GRANT SELECT ON ModuleAttendanceView TO koordynator_studiow;
GRANT EXECUTE ON AddCourse TO koordynator_studiow;
GRANT EXECUTE ON AddCourseModule TO koordynator_studiow;
GRANT EXECUTE ON CheckCourseCapacity TO koordynator_studiow;
GRANT SELECT ON MeetingAttendance TO koordynator_studiow;
GRANT SELECT ON ModuleAttendance TO koordynator_studiow;
GRANT UPDATE (MeetingCompletion) ON MeetingAttendance TO
koordynator_studiow;
GRANT UPDATE (ModuleCompletion) ON ModuleAttendance TO
koordynator_studiow;
GRANT EXECUTE ON AddWebinar TO koordynator_studiow;
GRANT SELECT, UPDATE ON Webinars TO koordynator_studiow;
GRANT SELECT ON TeachersSchedule TO koordynator_studiow;
GRANT EXECUTE ON TeacherSchedule TO koordynator_studiow;
```

Opiekun praktyk

Rola opiekun_praktyk ma uprawnienia do zarządzania praktykami oraz szczegółami praktyk studenckich.

```
CREATE ROLE opiekun_praktyk;
GRANT EXECUTE ON AddInternship TO opiekun_praktyk;
GRANT SELECT, INSERT, UPDATE ON Internships TO opiekun_praktyk;
GRANT SELECT, INSERT, UPDATE ON StudentInternshipDetails TO
opiekun_praktyk;
```

Tłumacz

Rola tlumacz ma uprawnienia do przeglądania webinarów, modułów kursów, spotkań akademickich oraz harmonogramu tłumaczy.

```
CREATE ROLE tlumacz;
GRANT SELECT ON Webinars TO tlumacz;
GRANT SELECT ON CourseModules TO tlumacz;
GRANT SELECT ON AcademicMeeting TO tlumacz;
GRANT SELECT ON TranslatorsSchedule TO tlumacz;
GRANT EXECUTE ON TranslatorSchedule TO tlumacz;
```

Kadrowy
Rola kadrowy ma uprawnienia do dodawania studentów, tłumaczy oraz nauczycieli.

```
CREATE ROLE kadrowy;
GRANT EXECUTE ON AddStudent TO kadrowy;
GRANT EXECUTE ON AddTranslator TO kadrowy;
GRANT EXECUTE ON AddTeacher TO kadrowy;
```

System
Rola system ma uprawnienia do dodawania zamówień oraz szczegółów zamówień.

```
CREATE ROLE system;
GRANT EXECUTE ON AddOrder TO system;
GRANT EXECUTE ON AddOrderContent TO system;
```

Student
Rola student ma uprawnienia do przeglądania zapisów na przyszłe wydarzenia, sprawdzania statusu kursów, modułów, spotkań oraz zamówień, a także przeglądania harmonogramu studentów.

```
CREATE ROLE student;
GRANT SELECT ON FutureEventsEnrollment TO student;
GRANT EXECUTE ON CanEnrollInCourse TO student;
GRANT EXECUTE ON CanEnrollInStudies TO student;
GRANT EXECUTE ON CanEnrollWebinar TO student;
GRANT EXECUTE ON IsCourseCompleted TO student;
GRANT EXECUTE ON IsCourseModuleCompleted TO student;
GRANT EXECUTE ON IsMeetingCompleted TO student;
GRANT EXECUTE ON IsOrderPaid TO student;
GRANT SELECT ON StudentsSchedule TO student;
GRANT EXECUTE ON StudentSchedule TO student;
```

Wygenerowane dane

W celu wygenerowania danych testowych dla bazy danych firmy szkoleniowej, użyto języka Python oraz kilku popularnych bibliotek.

Użyte narzędzia i biblioteki

- Język programowania: Python
- Biblioteki:
    - Faker: Biblioteka do generowania fałszywych danych, takich jak imiona, nazwiska, adresy, numery telefonów itp.
    - Pandas: Biblioteka do manipulacji i analizy danych, używana do tworzenia i zapisywania danych w formacie CSV.
    - Random: Wbudowana biblioteka Pythona do generowania losowych liczb i wyborów.
    - CSV: Wbudowana biblioteka Pythona do pracy z plikami CSV.
    - TQDM: Biblioteka do wyświetlania paska postępu podczas iteracji.

Wszystkie dane zostały wygenerowane przy użyciu języka Python oraz bibliotek Faker, Pandas, Random, CSV i TQDM. Dane te zapisano w plikach CSV, co pozwoliło na łatwe ich zaimportowane do bazy danych w celu testowania i analizy.