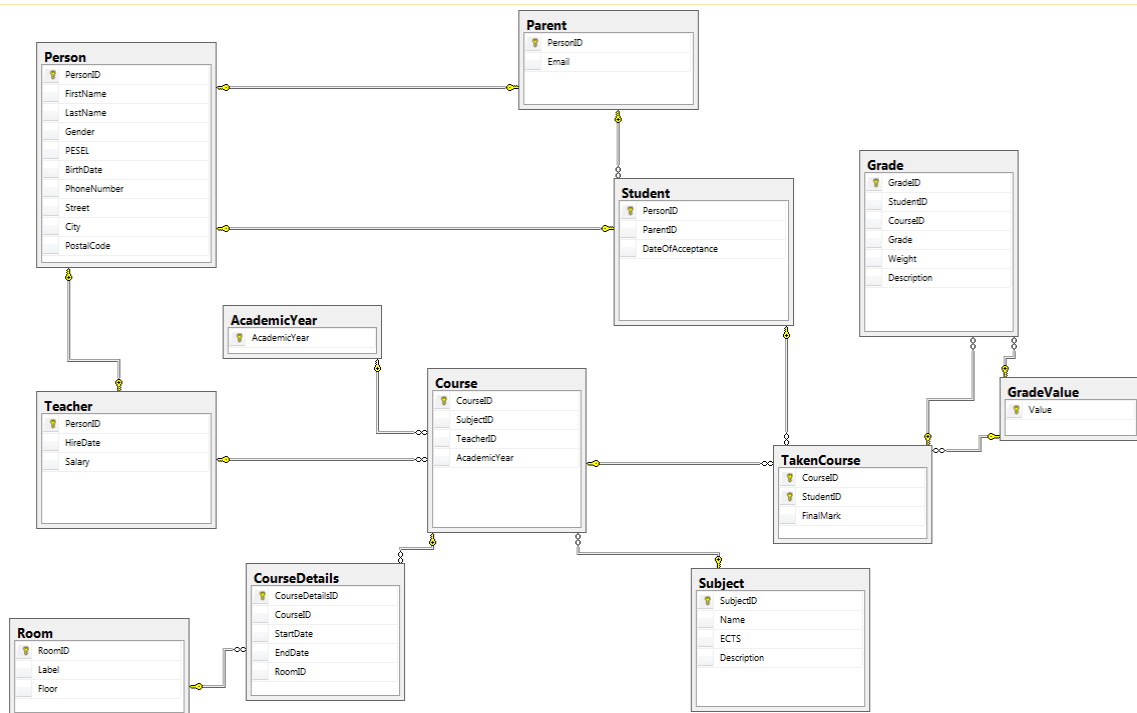


Diagram ER



Schemat bazy danych:

Tabela Person

	Column Name	Data Type	Allow Nulls
PK	PersonID	int	<input type="checkbox"/>
	FirstName	nvarchar(64)	<input type="checkbox"/>
	LastName	nvarchar(64)	<input type="checkbox"/>
	Gender	nchar(1)	<input checked="" type="checkbox"/>
	PESEL	nchar(11)	<input type="checkbox"/>
	BirthDate	date	<input checked="" type="checkbox"/>
	PhoneNumber	nvarchar(16)	<input checked="" type="checkbox"/>
	Street	nvarchar(32)	<input checked="" type="checkbox"/>
	City	nvarchar(32)	<input checked="" type="checkbox"/>
	PostalCode	nchar(6)	<input checked="" type="checkbox"/>

Tabela Parent

	Column Name	Data Type	Allow Nulls
PK	PersonID	int	<input type="checkbox"/>
	Email	nchar(64)	<input checked="" type="checkbox"/>

Tabela Teacher

	Column Name	Data Type	Allow Nulls
▶	PersonID	int	<input type="checkbox"/>
	HireDate	date	<input type="checkbox"/>
	Salary	money	<input type="checkbox"/>

Tabela Student

	Column Name	Data Type	Allow Nulls
▶	PersonID	int	<input type="checkbox"/>
	ParentID	int	<input type="checkbox"/>
	DateOfAcceptance	date	<input type="checkbox"/>

Tabela TakenCourse

	Column Name	Data Type	Allow Nulls
▶	CourseID	int	<input type="checkbox"/>
▶	StudentID	int	<input type="checkbox"/>
	FinalMark	float	<input checked="" type="checkbox"/>

Tabela Grade

	Column Name	Data Type	Allow Nulls
▶	GradeID	int	<input type="checkbox"/>
	StudentID	int	<input type="checkbox"/>
	CourseID	int	<input type="checkbox"/>
	Grade	float	<input type="checkbox"/>
	Description	nvarchar(64)	<input checked="" type="checkbox"/>

Tabela GradeValue

	Column Name	Data Type	Allow Nulls
▶	Value	float	<input type="checkbox"/>

Tabela Course

	Column Name	Data Type	Allow Nulls
▶	CourseID	int	<input type="checkbox"/>
	SubjectID	int	<input type="checkbox"/>
	TeacherID	int	<input type="checkbox"/>
	AcademicYear	nchar(9)	<input type="checkbox"/>

Tabela AcademicYear

	Column Name	Data Type	Allow Nulls
▶	AcademicYear	nchar(9)	<input type="checkbox"/>

Tabela Subject

	Column Name	Data Type	Allow Nulls
🔑	SubjectID	int	<input type="checkbox"/>
	Name	nvarchar(64)	<input type="checkbox"/>
	ECTS	tinyint	<input type="checkbox"/>
	Description	nvarchar(64)	<input checked="" type="checkbox"/>

Tabela CourseDetails

	Column Name	Data Type	Allow Nulls
🔑	CourseDetailsID	int	<input type="checkbox"/>
	CourseID	int	<input type="checkbox"/>
	StartDate	datetime	<input type="checkbox"/>
	EndDate	datetime	<input type="checkbox"/>
	RoomID	int	<input type="checkbox"/>

Tabela Room

🔑	RoomID	int	<input type="checkbox"/>
	Label	nvarchar(8)	<input type="checkbox"/>
	Floor	tinyint	<input checked="" type="checkbox"/>

Cel projektu

Stworzenie bazy danych na wzór szkoły/uczelni. W bazie danych przechowujemy informacje o nauczycielach, uczniu i jego opiece, przedmiotach które zaliczył/aktualnie zalicza uczeń.

Główne założenia/ograniczenia przyjęte przy projektowaniu.

- Osoba może być jednocześnie byłym studentem, aktualnym nauczycielem jak i rodzicem innego ucznia.
- Kurs to po prostu dany przedmiot, podział przedmiotów na kursy jest po to aby był podział na lata akademickie, stąd np. kurs z przedmiotu Programowanie w roku 2021/2022 może być prowadzony przez inną osobę oraz mieć lekcje w innych dniach niż kurs z przedmiotu Programowanie w roku 2022/2023
- Nie ma podziału na klasy, każdy uczeń wybiera na jakie przedmioty chce się zapisać.
- Lekcje z danego kursu nie konieczne odbywają się w ten sam dzień i o tej samej porze, zajęcia mogą np. odbywać się raz na 2 tygodnie, albo raz w jednej sali raz w drugiej itd.
- Student może mieć wiele ocen z danego kursu, ale ma jedną ocenę końcową, jeśli nie ma oceny końcowej to oznacza, że kurs jeszcze się nie zakończył

Opis widoków, procedur, wyzwalaczy, funkcji

Widoki

```
-- Widok nr 1 - Informacje o każdym nauczycielu, i jakie przedmioty prowadzi
CREATE VIEW TeachersInfo
AS
SELECT P.PersonID, P.FirstName, P.LastName, T.Salary, T.HireDate,
S.Name, C.AcademicYear
FROM Teacher T JOIN Person P ON T.PersonID = P.PersonID
JOIN Course C ON T.PersonID = C.TeacherID
JOIN [Subject] S ON C.SubjectID = S.SubjectID;
```

```
-- Widok nr 2 - Policz ile każdy kurs ma spotkań oraz łączną długość spotkań
CREATE VIEW Lessons
AS
SELECT C.AcademicYear, S.Name, COUNT(*) as NumberOfMeetings,
SUM(DATEDIFF(minute, CD.StartDate, CD.EndDate))/60 as [Length]
FROM Course C JOIN CourseDetails CD ON C.CourseID = CD.CourseID
JOIN Subject S ON C.SubjectID = S.SubjectID
GROUP BY C.AcademicYear, S.Name;
```

```
-- Widok nr 3 - Dla każdego rodzica wyświetl jego dzieci
CREATE VIEW ParentChildren
AS
SELECT Pinfo.FirstName [Parent FirstName], Pinfo.LastName [Parent
LastName], Sinfo.FirstName [Child FirstName], Sinfo.LastName [Child
LastName]
FROM Parent P JOIN Person Pinfo ON P.PersonID = Pinfo.PersonID
JOIN Student S ON P.PersonID = S.ParentID
JOIN Person Sinfo ON Sinfo.PersonID = S.PersonID;
```

```

-- Widok nr 4 - Dla każdej osoby kim jest (nauczycielem, studentem itp)
CREATE VIEW IdentifyPerson
AS
    SELECT P.FirstName, P.LastName,
    CASE
        WHEN (SELECT S.PersonID FROM Student S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [StudentCheck],
    CASE
        WHEN (SELECT S.PersonID FROM Teacher S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [TeacherCheck],
    CASE
        WHEN (SELECT S.PersonID FROM Parent S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [ParentCheck]
    FROM Person P;

```

```

-- Widok nr 5 - Dla każdego ucznia oblicz średnią ocen w danym kursie, na
którym zapisany był uczeń
CREATE VIEW AverageGrades
AS
    SELECT S.PersonID, P.FirstName, P.LastName, G.CourseID,
    CAST(AVG(G.Grade) as decimal(10,2)) as [Average]
    FROM Student S
    JOIN Person P ON S.PersonID = P.PersonID
    JOIN TakenCourse TC ON S.PersonID = TC.StudentID
    JOIN Grade G ON TC.StudentID = G.StudentID AND TC.CourseID =
G.CourseID
    GROUP BY S.PersonID, P.FirstName, P.LastName, G.CourseID;

```

Procedury

```
-- Procedura nr 1 - Dla konkretnego ucznia wypisz jego plan lekcji w
przedziale pomiędzy zadanymi datami
CREATE PROCEDURE StudentSchedule @StudentID int, @StartDate datetime,
@EndDate datetime
AS
SET @EndDate = DATEADD(day, 1, @EndDate);--zwiększam o jeden dzien, i
teraz tu będzie jeden dzien pozniej, ale godzina 00:00:00
SELECT * FROM Student S
JOIN TakenCourse TC ON S.PersonID = TC.StudentID
JOIN Course C ON C.CourseID = TC.CourseID
JOIN CourseDetails CD ON CD.CourseID = C.CourseID
WHERE TC.FinalMark IS NULL --jeszcze nie ma oceny końcowej więc to
oznacza, że kurs się jeszcze nie zakończył czyli powinien być w planie
lekcji
AND S.PersonID = @StudentID
AND CD.StartDate >= @StartDate AND CD.StartDate < @EndDate;
GO

--EXEC StudentSchedule @StudentID = 7, @StartDate='2020-04-04', @EndDate
= '2022-11-18';
```

```
-- Procedura nr 2 - Dla konkretnego ucznia zwraca jakie przedmioty już
ma zakończone
CREATE PROCEDURE StudentCompleted @StudentID int
AS
SELECT S.PersonID, P.FirstName, P.LastName, C.AcademicYear, SB.Name,
TC.FinalMark FROM Student S
JOIN TakenCourse TC ON S.PersonID = TC.StudentID
JOIN Course C ON C.CourseID = TC.CourseID
JOIN [Subject] SB ON C.SubjectID = SB.SubjectID
JOIN Person P ON S.PersonID = P.PersonID
WHERE TC.FinalMark IS NOT NULL
AND S.PersonID = @StudentID
GO

--EXEC StudentCompleted @StudentID = 7;
```

```
--Funkcja nr 1 (potrzebna do kolejnej procedury) - Zwraca daty z
zadanego przedziału, z odstępami o interwał
CREATE FUNCTION GenerateDatesInteval (@StartDate date, @EndDate date,
@Interval tinyint)
RETURNS TABLE
AS
RETURN
(
    WITH MY_CTE
    AS
    (SELECT @StartDate StartDate
    UNION ALL
    SELECT DATEADD(day, @Interval, StartDate)
    FROM MY_CTE
    WHERE DATEADD(day, @Interval, StartDate) <= @EndDate
    )
    SELECT * FROM MY_CTE
)
--SELECT * FROM dbo.GenerateDatesInteval('2023-08-09','2023-08-16',7);
```

```
-- Procedura nr 3 - Dodaje jakiś przedmiot np. w każdy wtorek o 12:00;
zakładam że podane daty są poprawne, czyli że startdate<=enddate oraz że
w przedziale tym istnieje przynajmniej jeden dzień tygodnia podany w
dayoftheweek
CREATE PROCEDURE AddCourseLesson @CourseID int, @DayOfTheWeek
nvarchar(16), @LessonStart time, @LessonEnd time, @StartDate date,
@EndDate date,
@RoomID int
AS
--Znalezienie pierwszego 'dobrego dnia'
WHILE(DATENAME(DW,@StartDate) != @DayOfTheWeek)
BEGIN
    SET @StartDate = DATEADD(day, 1, @StartDate)
END
INSERT INTO CourseDetails(CourseID, StartDate, EndDate, RoomID) SELECT
@CourseID, CAST(F.StartDate as datetime) + CAST(@LessonStart as
datetime),
CAST(F.StartDate as datetime) + CAST(@LessonEnd as datetime), @RoomID
FROM dbo.GenerateDatesInteval(@StartDate, @EndDate, 7) F;
GO

--EXEC AddCourseLesson @CourseID = 5, @DayOfTheWeek = 'Monday',
@LessonStart = '12:00:00', @LessonEnd = '13:00:00',
--@StartDate='2023-11-04', @EndDate = '2023-11-27', @RoomID = 1
```

```

-- Procedura nr 4 - dla podanego id studenta wypisuje informacje o nim,
jego rodzicu i jego rodzeństwu
CREATE PROCEDURE StudentFamily @StudentID int
AS
    DECLARE @ParentID int;
    SET @ParentID = (SELECT S.ParentID FROM Student S WHERE S.PersonID
= @StudentID);

    SELECT P.*, 'Student' as Relation FROM Student S
    JOIN Person P ON P.PersonID = S.PersonID
    WHERE S.ParentID = @ParentID
    UNION
    SELECT P.*, 'Parent' as Relation FROM Person P
    WHERE P.PersonID = @ParentID
    ORDER BY Relation;
GO

--EXEC StudentFamily 6;

```

```

-- Procedura nr 5 - dla podanego id studenta i id kursu wypisuje oceny
studenta, oraz średnią ocen w osobnym polu
CREATE PROCEDURE CourseGrades @StudentID int, @CourseID int
AS
    SELECT G.*, (SELECT AG.Average FROM AverageGrades AG WHERE
AG.PersonID = @StudentID AND AG.CourseID = @CourseID) AS [Average]
    FROM Grade G
    WHERE G.StudentID = @StudentID AND G.CourseID = @CourseID
GO

--EXEC CourseGrades @StudentID = 6, @CourseID = 1;

```


Wyzwalacze

```
-- Wyzwalacz nr 1 - Sprawdza czy student nie zaliczył już danego
przedmiotu w innym roku akademickim
CREATE TRIGGER SubjectAlreadyCompleted
ON TakenCourse
AFTER INSERT
AS
BEGIN
    DECLARE @counter int;
    SET @counter = (--zliczam ile razy student zapisał się na
przedmiot, jeśli jakikolwiek student zapisał się więcej niż raz to
rollback
    SELECT MAX([counter])
        FROM (
            SELECT TC.StudentID, C.SubjectID , COUNT(*) [counter]
            FROM TakenCourse TC
            JOIN Course C ON TC.CourseID = C.CourseID
            WHERE TC.StudentID IN (SELECT StudentID FROM inserted
WHERE StudentID IS NOT NULL)--zeby sprawdzac tylko dla studentow ktorzych
dotyczy zmiana
            AND TC.FinalMark IS NULL OR TC.FinalMark != 2 --jeśli
juz zaliczał, ale dostał ocene 2 to może poprawić więc jest ok
            GROUP BY TC.StudentID, C.SubjectID
        )Subquery
    )

    IF @counter > 1
        ROLLBACK
END;
```

```
-- Wyzwalacz nr 2 - Sprawdza czy rodzic podał adres email jeśli nie podał
nr telefonu
CREATE TRIGGER ParentAnyInfo
ON Parent
AFTER INSERT
AS
    IF EXISTS(
        SELECT * FROM Person PE JOIN inserted PA ON PE.PersonID =
PA.PersonID
        WHERE PE.PhoneNumber IS NULL AND PA.Email IS NULL
    )
        ROLLBACK
;
```

```

-- Wyzwalacz nr 3 - Sprawdza czy sala nie jest już zajęta przez inne
zajęcia
CREATE TRIGGER RoomTaken
ON CourseDetails
AFTER INSERT
AS
BEGIN
    IF EXISTS(
        SELECT * FROM inserted I
        CROSS APPLY(
            SELECT * FROM CourseDetails CD
            WHERE CD.CourseDetailsID != I.CourseDetailsID--bo
oczywiście kolizja z samym sobą zawsze będzie
            AND I.RoomID = CD.RoomID
            AND ( (I.StartDate > CD.StartDate AND I.StartDate <
CD.EndDate) OR (I.EndDate > CD.StartDate AND I.EndDate < CD.EndDate))
        )Subquery
    )
    BEGIN
        RAISERROR('Room already taken!.', 11, 2)
        ROLLBACK
    END
END;

```

```

-- Wyzwalacz nr 4 - Sprawdza czy zajęcia które dodajemy dla danego kursu
rzeczywiście są w odpowiednim roku akademickim
CREATE TRIGGER AcademicYearLessons
ON CourseDetails
AFTER INSERT
AS
BEGIN
    IF EXISTS(
        SELECT * FROM inserted I
        JOIN Course C ON I.CourseID = C.CourseID
        WHERE CAST(StartDate as date) NOT BETWEEN
        CAST(LEFT(C.AcademicYear, 4) + '-09-01' as date) AND
CAST(RIGHT(C.AcademicYear, 4) + '-06-30' as date) OR
        CAST(EndDate as date) NOT BETWEEN
        CAST(LEFT(C.AcademicYear, 4) + '-09-01' as date) AND
CAST(RIGHT(C.AcademicYear, 4) + '-06-30' as date)
    )BEGIN
        RAISERROR('Wrong Academic Year !', 10, 1)
        ROLLBACK
    END
END;

```

```
--Wyzwalacz nr 5 - Sprawdzenie czy jeśli student dostaje ocene końcową z
kursu z roku akademickiego który już się skończył to nie jest to null
CREATE TRIGGER CorrectFinalMark
ON TakenCourse
AFTER INSERT
AS
BEGIN
    IF EXISTS(
        SELECT * FROM inserted I
        JOIN Course C ON I.CourseID = C.CourseID
        WHERE FinalMark IS NULL
        AND C.AcademicYear != (SELECT TOP 1 AC.AcademicYear FROM
AcademicYear AC ORDER BY LEFT(AC.AcademicYear, 4) DESC)
    )
        ROLLBACK;
END;
```

Skrypty tworzące wszystkie obiekty bazy danych:

a) Skrypt tworzący bazę danych

```
USE [master]
GO
/***** Object:  Database [School]      Script Date: 12/25/2023 4:10:52 PM
*****/
CREATE DATABASE [School]
    CONTAINMENT = NONE
    ON PRIMARY
    ( NAME = N'School', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\School.mdf' , SIZE = 4096KB ,
MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
    LOG ON
    ( NAME = N'School_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\School_log.ldf' , SIZE = 1024KB ,
MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [School] SET COMPATIBILITY_LEVEL = 110
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [School].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [School] SET ANSI_NULL_DEFAULT OFF
```

```
GO
ALTER DATABASE [School] SET ANSI_NULLS OFF
GO
ALTER DATABASE [School] SET ANSI_PADDING OFF
GO
ALTER DATABASE [School] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [School] SET ARITHABORT OFF
GO
ALTER DATABASE [School] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [School] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [School] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [School] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [School] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [School] SET CURSOR_DEFAULT GLOBAL
GO
ALTER DATABASE [School] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [School] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [School] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [School] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [School] SET  DISABLE_BROKER
GO
ALTER DATABASE [School] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [School] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [School] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [School] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [School] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [School] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [School] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [School] SET RECOVERY FULL
```

```

GO
ALTER DATABASE [School] SET MULTI_USER
GO
ALTER DATABASE [School] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [School] SET DB_CHAINING OFF
GO
ALTER DATABASE [School] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [School] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
EXEC sys.sp_db_vardecimal_storage_format N'School', N'ON'
GO
USE [School]
GO
/***** Object:  StoredProcedure [dbo].[AddCourseLesson]      Script Date:
12/25/2023 4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[AddCourseLesson] @CourseID int, @DayOfTheWeek
nvarchar(16), @LessonStart time, @LessonEnd time, @StartDate date,
@EndDate date,
@RoomID int
AS
--Znalezienie pierwszego 'dobrego dnia'
WHILE(DATENAME(DW,@StartDate) != @DayOfTheWeek)
BEGIN
    SET @StartDate = DATEADD(day, 1, @StartDate)
END
INSERT INTO CourseDetails(CourseID, StartDate, EndDate, RoomID) SELECT
@CourseID, CAST(F.StartDate as datetime) + CAST(@LessonStart as
datetime),
CAST(F.StartDate as datetime) + CAST(@LessonEnd as datetime), @RoomID
FROM dbo.GenerateDatesInteval(@StartDate, @EndDate, 7) F;

GO
/***** Object:  StoredProcedure [dbo].[CourseGrades]      Script Date:
12/25/2023 4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[CourseGrades] @StudentID int, @CourseID int
AS

```

```

        SELECT G.*, (SELECT AG.Average FROM AverageGrades AG WHERE
AG.PersonID = @StudentID AND AG.CourseID = @CourseID) AS [Average]
        FROM Grade G
        WHERE G.StudentID = @StudentID AND G.CourseID = @CourseID

GO
/***** Object:  StoredProcedure [dbo].[StudentCompleted]      Script
Date: 12/25/2023 4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[StudentCompleted] @StudentID int
AS
SELECT S.PersonID, P.FirstName, P.LastName, C.AcademicYear, SB.Name,
TC.FinalMark FROM Student S
JOIN TakenCourse TC ON S.PersonID = TC.StudentID
JOIN Course C ON C.CourseID = TC.CourseID
JOIN [Subject] SB ON C.SubjectID = SB.SubjectID
JOIN Person P ON S.PersonID = P.PersonID
WHERE TC.FinalMark IS NOT NULL
AND S.PersonID = @StudentID

GO
/***** Object:  StoredProcedure [dbo].[StudentFamily]      Script Date:
12/25/2023 4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[StudentFamily] @StudentID int
AS
    DECLARE @ParentID int;
    SET @ParentID = (SELECT S.ParentID FROM Student S WHERE S.PersonID
= @StudentID);

        SELECT P.*, 'Student' as Relation FROM Student S
        JOIN Person P ON P.PersonID = S.PersonID
        WHERE S.ParentID = @ParentID
    UNION
        SELECT P.*, 'Parent' as Relation FROM Person P
        WHERE P.PersonID = @ParentID
        ORDER BY Relation;

GO
/***** Object:  StoredProcedure [dbo].[StudentSchedule]      Script Date:

```

```

12/25/2023 4:10:52 PM ***/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[StudentSchedule] @StudentID int, @StartDate
datetime, @EndDate datetime
AS
SET @EndDate = DATEADD(day, 1, @EndDate);--zwiększam o jeden dzien, i
teraz tu będzie jeden dzien pozniej, ale godzina 00:00:00
SELECT * FROM Student S
JOIN TakenCourse TC ON S.PersonID = TC.StudentID
JOIN Course C ON C.CourseID = TC.CourseID
JOIN CourseDetails CD ON CD.CourseID = C.CourseID
WHERE TC.FinalMark IS NULL --jeszcze nie ma oceny końcowej więc to
oznacza, że kurs się jeszcze nie zakończył czyli powinien być w planie
lekcji
AND S.PersonID = @StudentID
AND CD.StartDate >= @StartDate AND CD.StartDate < @EndDate;

GO
/***** Object: Table [dbo].[AcademicYear]      Script Date: 12/25/2023
4:10:52 PM ***/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AcademicYear](
    [AcademicYear] [nchar](9) NOT NULL,
    CONSTRAINT [PK_AcademicYear] PRIMARY KEY CLUSTERED
(
    [AcademicYear] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[Course]      Script Date: 12/25/2023 4:10:52
PM ***/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Course](
    [CourseID] [int] IDENTITY(1,1) NOT NULL,
    [SubjectID] [int] NOT NULL,

```

```

        [TeacherID] [int] NOT NULL,
        [AcademicYear] [nchar](9) NOT NULL,
    CONSTRAINT [PK_Lesson_1] PRIMARY KEY CLUSTERED
(
    [CourseID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

/***** Object:  Table [dbo].[CourseDetails]      Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CourseDetails](
    [CourseDetailsID] [int] IDENTITY(1,1) NOT NULL,
    [CourseID] [int] NOT NULL,
    [StartDate] [datetime] NOT NULL,
    [EndDate] [datetime] NOT NULL,
    [RoomID] [int] NOT NULL,
    CONSTRAINT [PK_CourseDetails] PRIMARY KEY CLUSTERED
(
    [CourseDetailsID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

/***** Object:  Table [dbo].[Grade]      Script Date: 12/25/2023 4:10:52
PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Grade](
    [GradeID] [int] IDENTITY(1,1) NOT NULL,
    [StudentID] [int] NOT NULL,
    [CourseID] [int] NOT NULL,
    [Grade] [float] NOT NULL,
    [Description] [nvarchar](64) NULL,
    CONSTRAINT [PK_Grade] PRIMARY KEY CLUSTERED
(
    [GradeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =

```



```

OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[GradeValue]    Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[GradeValue](
    [Value] [float] NOT NULL,
    CONSTRAINT [PK_GradeValue_1] PRIMARY KEY CLUSTERED
(
    [Value] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[Parent]    Script Date: 12/25/2023 4:10:52
PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Parent](
    [PersonID] [int] NOT NULL,
    [Email] [nchar](64) NULL,
    CONSTRAINT [PK_Parent] PRIMARY KEY CLUSTERED
(
    [PersonID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[Person]    Script Date: 12/25/2023 4:10:52
PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Person](
    [PersonID] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [nvarchar](64) NOT NULL,

```

```

        [LastName] [nvarchar](64) NOT NULL,
        [Gender] [nchar](1) NULL,
        [PESEL] [nchar](11) NOT NULL,
        [BirthDate] [date] NULL,
        [PhoneNumber] [nvarchar](16) NULL,
        [Street] [nvarchar](32) NULL,
        [City] [nvarchar](32) NULL,
        [PostalCode] [nchar](6) NULL,
    CONSTRAINT [PK_Person] PRIMARY KEY CLUSTERED
    (
        [PersonID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
    CONSTRAINT [UNIQUE_PESSEL] UNIQUE NONCLUSTERED
    (
        [PESEL] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

```

GO

```

/***** Object:  Table [dbo].[Room]      Script Date: 12/25/2023 4:10:52
PM *****/

```

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

```

CREATE TABLE [dbo].[Room](
    [RoomID] [int] IDENTITY(1,1) NOT NULL,
    [Label] [nvarchar](8) NOT NULL,
    [Floor] [tinyint] NULL,
    CONSTRAINT [PK_Room] PRIMARY KEY CLUSTERED
    (
        [RoomID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

```

GO

```

/***** Object:  Table [dbo].[Student]   Script Date: 12/25/2023
4:10:52 PM *****/

```

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

```

CREATE TABLE [dbo].[Student](

```

```

        [PersonID] [int] NOT NULL,
        [ParentID] [int] NOT NULL,
        [DateOfAcceptance] [date] NOT NULL,
    CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
(
    [PersonID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[Subject]      Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Subject](
    [SubjectID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](64) NOT NULL,
    [ECTS] [tinyint] NOT NULL,
    [Description] [nvarchar](64) NULL,
    CONSTRAINT [PK_Subject] PRIMARY KEY CLUSTERED
(
    [SubjectID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object: Table [dbo].[TakenCourse]   Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TakenCourse](
    [CourseID] [int] NOT NULL,
    [StudentID] [int] NOT NULL,
    [FinalMark] [float] NULL,
    CONSTRAINT [PK_CurrentSubject] PRIMARY KEY CLUSTERED
(
    [CourseID] ASC,
    [StudentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

```

```

) ON [PRIMARY]

GO
/***** Object:  Table [dbo].[Teacher]      Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Teacher](
    [PersonID] [int] NOT NULL,
    [HireDate] [date] NOT NULL,
    [Salary] [money] NOT NULL,
    CONSTRAINT [PK_Teacher] PRIMARY KEY CLUSTERED
(
    [PersonID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/***** Object:  UserDefinedFunction [dbo].[GenerateDatesInteval]
Script Date: 12/25/2023 4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GenerateDatesInteval] (@StartDate date, @EndDate
date, @Interval tinyint)
RETURNS TABLE
AS
RETURN
(
    WITH MY_CTE
    AS
    (SELECT @StartDate StartDate
    UNION ALL
    SELECT DATEADD(day, @Interval, StartDate)
    FROM MY_CTE
    WHERE DATEADD(day, @Interval, StartDate) <= @EndDate
    )
    SELECT * FROM MY_CTE
)

GO
/***** Object:  View [dbo].[AverageGrades]      Script Date: 12/25/2023

```

```

4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[AverageGrades]
AS
    SELECT S.PersonID, P.FirstName, P.LastName, G.CourseID,
    CAST(AVG(G.Grade) as decimal(10,2)) as [Average]
    FROM Student S
    JOIN Person P ON S.PersonID = P.PersonID
    JOIN TakenCourse TC ON S.PersonID = TC.StudentID
    JOIN Grade G ON TC.StudentID = G.StudentID AND TC.CourseID =
G.CourseID
    GROUP BY S.PersonID, P.FirstName, P.LastName, G.CourseID;

GO
/***** Object: View [dbo].[IdentifyPerson] Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[IdentifyPerson]
AS
    SELECT P.FirstName, P.LastName,
    CASE
        WHEN (SELECT S.PersonID FROM Student S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [StudentCheck],
    CASE
        WHEN (SELECT S.PersonID FROM Teacher S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [TeacherCheck],
    CASE
        WHEN (SELECT S.PersonID FROM Parent S WHERE S.PersonID =
P.PersonID) IS NOT NULL THEN 'YES'
        ELSE 'NO'
    END AS [ParentCheck]
    FROM Person P;

GO
/***** Object: View [dbo].[Lessons] Script Date: 12/25/2023 4:10:52
PM *****/

```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[Lessons]
AS
    SELECT C.AcademicYear, S.Name, COUNT(*) as NumberOfMeetings,
    SUM(DATEDIFF(minute, CD.StartDate, CD.EndDate))/60 as [Length]
    FROM Course C JOIN CourseDetails CD ON C.CourseID = CD.CourseID
    JOIN Subject S ON C.SubjectID = S.SubjectID
    GROUP BY C.AcademicYear, S.Name;

GO
/***** Object: View [dbo].[ParentChildren]    Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[ParentChildren]
AS
    SELECT Pinfo.FirstName [Parent FirstName], Pinfo.LastName [Parent
    LastName], Sinfo.FirstName [Child FirstName], Sinfo.LastName [Child
    LastName]
    FROM Parent P JOIN Person Pinfo ON P.PersonID = Pinfo.PersonID
    JOIN Student S ON P.PersonID = S.ParentID
    JOIN Person Sinfo ON Sinfo.PersonID = S.PersonID;

GO
/***** Object: View [dbo].[TeachersInfo]    Script Date: 12/25/2023
4:10:52 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[TeachersInfo]
AS
    SELECT P.PersonID, P.FirstName, P.LastName, T.Salary, T.HireDate,
    S.Name, C.AcademicYear
    FROM Teacher T JOIN Person P ON T.PersonID = P.PersonID
    JOIN Course C ON T.PersonID = C.TeacherID
    JOIN [Subject] S ON C.SubjectID = S.SubjectID;

GO
ALTER TABLE [dbo].[Course] WITH CHECK ADD CONSTRAINT
[FK_Course_AcademicYear] FOREIGN KEY([AcademicYear])

```

```

REFERENCES [dbo].[AcademicYear] ([AcademicYear])
GO
ALTER TABLE [dbo].[Course] CHECK CONSTRAINT [FK_Course_AcademicYear]
GO
ALTER TABLE [dbo].[Course] WITH CHECK ADD CONSTRAINT
[FK_Course_Subject] FOREIGN KEY([SubjectID])
REFERENCES [dbo].[Subject] ([SubjectID])
GO
ALTER TABLE [dbo].[Course] CHECK CONSTRAINT [FK_Course_Subject]
GO
ALTER TABLE [dbo].[Course] WITH CHECK ADD CONSTRAINT
[FK_Lesson_Teacher] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Teacher] ([PersonID])
GO
ALTER TABLE [dbo].[Course] CHECK CONSTRAINT [FK_Lesson_Teacher]
GO
ALTER TABLE [dbo].[CourseDetails] WITH CHECK ADD CONSTRAINT
[FK_CourseDetails_Course] FOREIGN KEY([CourseID])
REFERENCES [dbo].[Course] ([CourseID])
GO
ALTER TABLE [dbo].[CourseDetails] CHECK CONSTRAINT
[FK_CourseDetails_Course]
GO
ALTER TABLE [dbo].[CourseDetails] WITH CHECK ADD CONSTRAINT
[FK_CourseDetails_Room] FOREIGN KEY([RoomID])
REFERENCES [dbo].[Room] ([RoomID])
GO
ALTER TABLE [dbo].[CourseDetails] CHECK CONSTRAINT
[FK_CourseDetails_Room]
GO
ALTER TABLE [dbo].[Grade] WITH CHECK ADD CONSTRAINT
[FK_Grade_GradeValue] FOREIGN KEY([Grade])
REFERENCES [dbo].[GradeValue] ([Value])
GO
ALTER TABLE [dbo].[Grade] CHECK CONSTRAINT [FK_Grade_GradeValue]
GO
ALTER TABLE [dbo].[Grade] WITH CHECK ADD CONSTRAINT
[FK_Grade_TakenCourse] FOREIGN KEY([CourseID], [StudentID])
REFERENCES [dbo].[TakenCourse] ([CourseID], [StudentID])
GO
ALTER TABLE [dbo].[Grade] CHECK CONSTRAINT [FK_Grade_TakenCourse]
GO
ALTER TABLE [dbo].[Parent] WITH NOCHECK ADD CONSTRAINT
[FK_Parent_Person] FOREIGN KEY([PersonID])
REFERENCES [dbo].[Person] ([PersonID])
NOT FOR REPLICATION

```

```
GO
ALTER TABLE [dbo].[Parent] CHECK CONSTRAINT [FK_Parent_Person]
GO
ALTER TABLE [dbo].[Student] WITH CHECK ADD CONSTRAINT
[FK_Student_Parent] FOREIGN KEY([ParentID])
REFERENCES [dbo].[Parent] ([PersonID])
GO
ALTER TABLE [dbo].[Student] CHECK CONSTRAINT [FK_Student_Parent]
GO
ALTER TABLE [dbo].[Student] WITH CHECK ADD CONSTRAINT
[FK_Student_Person] FOREIGN KEY([PersonID])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Student] CHECK CONSTRAINT [FK_Student_Person]
GO
ALTER TABLE [dbo].[TakenCourse] WITH CHECK ADD CONSTRAINT
[FK_TakenCourse_Course] FOREIGN KEY([CourseID])
REFERENCES [dbo].[Course] ([CourseID])
GO
ALTER TABLE [dbo].[TakenCourse] CHECK CONSTRAINT [FK_TakenCourse_Course]
GO
ALTER TABLE [dbo].[TakenCourse] WITH CHECK ADD CONSTRAINT
[FK_TakenCourse_GradeValue] FOREIGN KEY([FinalMark])
REFERENCES [dbo].[GradeValue] ([Value])
GO
ALTER TABLE [dbo].[TakenCourse] CHECK CONSTRAINT
[FK_TakenCourse_GradeValue]
GO
ALTER TABLE [dbo].[TakenCourse] WITH CHECK ADD CONSTRAINT
[FK_TakenCourse_Student] FOREIGN KEY([StudentID])
REFERENCES [dbo].[Student] ([PersonID])
GO
ALTER TABLE [dbo].[TakenCourse] CHECK CONSTRAINT
[FK_TakenCourse_Student]
GO
ALTER TABLE [dbo].[Teacher] WITH CHECK ADD CONSTRAINT
[FK_Teacher_Person] FOREIGN KEY([PersonID])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Teacher] CHECK CONSTRAINT [FK_Teacher_Person]
GO
USE [master]
GO
ALTER DATABASE [School] SET READ_WRITE
GO
```


b) Przykładowe dane wejściowe

```
--Usuwanie przykładowych danych
```

```
DELETE FROM CourseDetails;  
DBCC CHECKIDENT ('CourseDetails', RESEED, 0);
```

```
DELETE FROM Room;  
DBCC CHECKIDENT ('Room', RESEED, 0);
```

```
DELETE FROM [Grade];  
DBCC CHECKIDENT ('Grade', RESEED, 0);
```

```
DELETE FROM TakenCourse;
```

```
DELETE FROM Course;  
DBCC CHECKIDENT ('Course', RESEED, 0);
```

```
DELETE FROM [Subject];  
DBCC CHECKIDENT ('Subject', RESEED, 0);
```

```
DELETE FROM GradeValue;  
DELETE FROM AcademicYear;
```

```
DELETE FROM Student;  
DELETE FROM Parent;  
DELETE FROM Teacher;
```

```
DELETE FROM Person;  
DBCC CHECKIDENT ('Person', RESEED, 0);
```

```
--Wstawianie przykładowych danych
```

```
INSERT INTO Person(FirstName, LastName, PESEL) VALUES
```

```
--nauczyciele
```

```
('Jan', 'Kowalski', '01345678901'),  
('Magda', 'Kwiatek', '02345678901'),
```

```
--rodzice
```

```
('Karolina', 'Małysz', '03345678901'),  
('Patrycja', 'Rondo', '04345678901'),
```

```
--nauczyciel i rodzic jednocześnie
```

```
('Piotrek', 'Grundwaldzki', '05345678901'),
```

```
--studenci
```

```
('Tomasz', 'Stoch', '06345678901'),  
('Natalia', 'Jan', '07345678901'),
```

```
('Anna', 'Donn', '08345678901'),  
( 'Mateusz', 'Święty', '09345678901');
```

```
INSERT INTO Parent(PersonID, Email) VALUES  
(3, '3@gmail.com'),  
(4, '4@gmail.com'),  
(5, '5@gmail.com');
```

```
INSERT INTO Student(PersonID, ParentID, DateOfAcceptance) VALUES  
(6, 3, '2021-08-31'),  
(7, 3, '2021-08-31'),  
(8, 4, '2022-08-31'),  
(9, 5, '2022-08-31');
```

```
INSERT INTO Teacher VALUES  
(1, '2021-03-23', 1100),  
(2, '2018-03-23', 600),  
(5, '2020-03-23', 1300);
```

```
INSERT INTO GradeValue VALUES  
(2.0),  
(3.0),  
(3.5),  
(4.0),  
(4.5),  
(5.0);
```

```
INSERT INTO AcademicYear VALUES  
( '2021/2022'),  
( '2022/2023');
```

```
INSERT INTO Room(Label, [Floor]) VALUES  
( '0053', 0),  
( '0056', 0),  
( '0059', 0),  
( '0089', 1);
```

```
INSERT INTO [Subject](Name, ECTS) VALUES  
( 'Logika i teoria mnogości', 8),  
( 'Algebra 1', 6),  
( 'Programowanie 1', 6);
```

```
INSERT INTO Course(SubjectID, TeacherID, AcademicYear) VALUES  
(1, 1, '2021/2022'),
```

```
(2, 1, '2021/2022'),  
(3, 2, '2021/2022'),  
(1, 5, '2022/2023'),  
(2, 1, '2022/2023');
```

```
INSERT INTO CourseDetails(CourseID, StartDate, EndDate, RoomID) VALUES
```

```
--Lekcje dla kursu nr 1
```

```
(1, '2021-10-15 13:00', '2021-10-15 13:45', 1),  
(1, '2021-10-22 13:00', '2021-10-22 13:45', 1),  
(1, '2021-10-29 13:15', '2021-10-29 14:00', 2),  
(1, '2021-11-05 13:00', '2021-11-05 13:45', 1),  
(1, '2021-11-12 13:00', '2021-11-12 13:45', 1),  
(1, '2021-11-19 13:00', '2021-11-19 13:45', 1),
```

```
--kurs nr 2
```

```
(2, '2021-10-15 10:00', '2021-10-15 13:00', 1),  
(2, '2021-10-22 10:00', '2021-10-22 13:00', 1),  
(2, '2021-10-29 10:00', '2021-10-29 13:00', 1),  
(2, '2021-11-05 10:00', '2021-11-05 13:00', 1),  
(2, '2021-11-12 10:00', '2021-11-12 13:00', 1),  
(2, '2021-11-19 10:00', '2021-11-19 13:00', 1),
```

```
--kurs nr 3
```

```
(3, '2021-10-15 10:00', '2021-10-15 13:00', 2),  
(3, '2021-10-22 10:00', '2021-10-22 13:00', 2),  
(3, '2021-10-29 10:00', '2021-10-29 13:00', 2),  
(3, '2021-11-05 10:00', '2021-11-05 13:00', 2),  
(3, '2021-11-12 10:00', '2021-11-12 13:00', 2),  
(3, '2021-11-19 10:00', '2021-11-19 13:00', 2),
```

```
--kurs nr 4
```

```
(4, '2022-10-15 10:00', '2022-10-15 13:00', 3),  
(4, '2022-10-22 10:00', '2022-10-22 13:00', 4),  
(4, '2022-11-12 10:00', '2022-11-12 13:00', 4),  
(4, '2022-11-19 10:00', '2022-11-19 13:00', 4),
```

```
--kurs nr 5
```

```
(5, '2022-10-15 10:00', '2022-10-15 13:00', 4),  
(5, '2022-10-22 10:00', '2022-10-22 13:00', 3),  
(5, '2022-10-29 10:00', '2022-10-29 13:00', 3),  
(5, '2022-11-12 10:00', '2022-11-12 13:00', 3),  
(5, '2022-11-19 10:00', '2022-11-19 13:00', 3);
```

```
INSERT INTO TakenCourse(CourseID, StudentID, FinalMark) VALUES
```

```
(1, 6, 5),  
(2, 6, 3.5),  
(3, 6, 3.5),  
(4, 7, NULL),  
(2, 7, 4.5),
```

```
(3, 7, 3.5),  
(2, 8, 2.0),  
(1, 9, 4.5),  
(5, 9, NULL),  
(3, 9, 3.5);
```

```
INSERT INTO Grade(StudentID, CourseID, Grade, [Description]) VALUES  
(6, 1, 5, 'Sprawdzian nr 1'),  
(6, 2, 5, 'Sprawdzian nr 1'),  
(6, 3, 5, 'Sprawdzian nr 1'),  
(6, 3, 5, 'Kartkówka');
```

```
INSERT INTO Grade(StudentID, CourseID, Grade) VALUES  
(6, 1, 5),  
(6, 1, 3.5),  
(6, 1, 3.0),  
(6, 2, 4.0),  
(6, 2, 4.5),  
(6, 2, 3.0),  
(6, 3, 5),  
(6, 3, 5),  
(6, 3, 4.5),  
(6, 3, 4.0),  
(6, 3, 3.0),  
(6, 3, 5),  
  
(7, 4, 5),  
(7, 2, 4.5),  
(7, 3, 3.5),  
(7, 4, 3.5),  
(7, 2, 4.5),  
  
(8, 2, 3.0),  
(8, 2, 3.0),  
(8, 2, 4.0),  
  
(9, 1, 5),  
(9, 1, 4.5),  
(9, 3, 4.0),  
(9, 3, 5),  
(9, 3, 4.5);
```