**Лабораторная работа № 1**

**Задание**

1. Установить Microsoft SQL Server 2005 Express Edition и Microsoft SQL Server Management Studio Express. **Объяснить основные различия между разными версиями продуктов семейства Microsoft SQL Server 2005. Дать рекомендации по выбору того или иного продукта в зависимости от ситуации.**
2. Настроить смешанную (Windows + SQL Server) аутентификацию. **Объяснить разницу между этими способами аутентификации. Дать рекомендации по необходимости применения этих способов аутентификации в зависимости от различных ситуаций.**
3. Каждому студенту необходимо создать базу данных, которая будет называться Student\_Name, где вместо Name будет написана фамилия латинскими буквами.
4. Каждый студент должен создать пользователя, имя (login) которого совпадает с фамилией студента, написанной латинскими буквами. Этот пользователь должен иметь привилегию (роль) для доступа к одноименной БД либо db\_owner, либо связку db\_datareader, db\_datawriter, db\_ddladmin и одноименную базу по умолчанию. **Объяснить различия между перечисленными привилегиями.** **Объяснить, что такое база данных по умолчанию, где ее можно выбрать (через Management Studio) и зачем она (база данных по умолчанию) нужна.**
5. **Проверить возможность соединения с Microsoft SQL Server 2005 Express Edition, используя созданного пользователя.**

**1.**

Оскільки при виконанні лабораторної використовувались продукти Microsoft SQL Server 2017, тому і порівняння продуктів даної версії будемо проводити. Аналіз продуктів грунтується на інформації із офіційної документації:

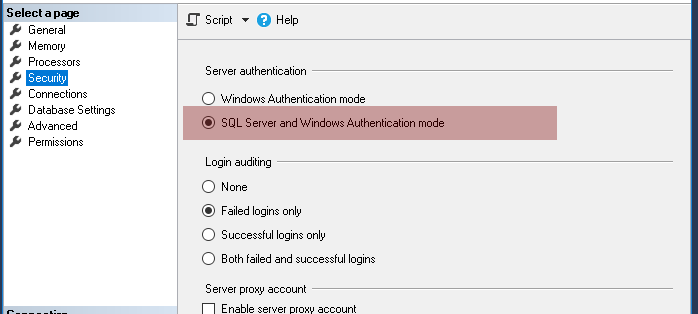
<https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2017?view=sql-server-ver15>

Існують наступні версії SQL Server: Enterprise, Standard, Web, Developer, Exprees editions. Першу дві версії найкраще підійдуть для розгортання систем для робочих продуктів, Web версія - для розміщення веб-ресурсів, Developer - для розробки та тестування та Express - для навчальних цілей та ознайомлення з продуктом. На сайті із документацію, блок Scale limits, зазначені різноманітні обмеження/розширення на функціонал в залежності від версії. Зрозуміло, що Express має найменше функціоналу, проте є безкоштовним.

Так наприклад, із таблиці Management tools, ми можемо дізнатись, що Express версія включає такі засоби як SQL Management Objects (SMO), SQL Configuration Manager, SQL CMD (Command Prompt tool)

2.

Скрін налаштування змішаної аутентифікації



**Відмінності між Windows та SQL аутентифікацією:**

Windows аутентифікація

- використовує логін та пароль користувача Windows, які перевіряються Windows, а не SQL Server

- вважається більш безпечним способом, ніж SQL аутентифіація

- використовує Kerberos security протокол

SQL аутентифікація

- логін та пароль зберігаються в базі SQL Server

- перевірка коректності даних відбувається на стороні SQL Server

- не підтримує Kerberos security протокол

Недоліки SQL аутентифікації:

- для користувача Windows виникає необхідність 2 рази вводити логін та пароль (для входу у Windows та для користування БД)

- Windows надає ширший спектр варіантів для password policies

Переваги SQL аутентифікації:

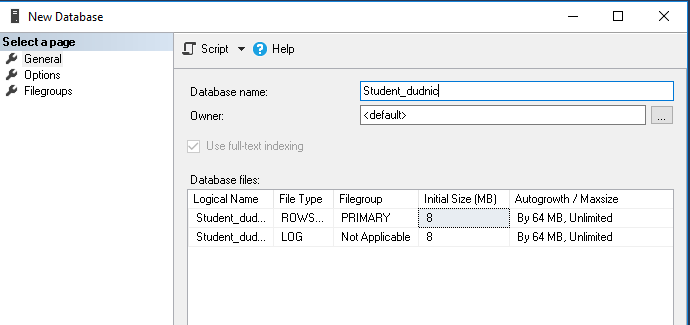
- підтримка більш старих додатків

- можливість роботи SQL Server із змішаними операційними системами, де не всі користувачі є користувачами Windows

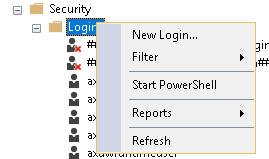
- можливість SQL Server працювати із Web

3.

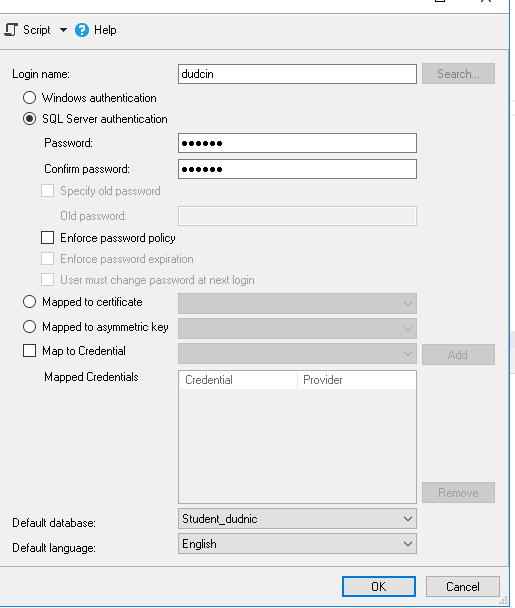
Скрін створення БД



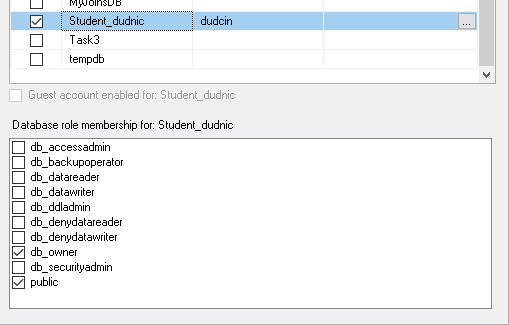
4. Скріни створення користувача



Налаштування пароля та дефолтною БД



Налаштування прав до створеної БД



**Згенерований скрипт для створення користувача**

USE [master]

GO

CREATE LOGIN [dudcin] WITH PASSWORD=N'dudcin', DEFAULT\_DATABASE=[Student\_dudnic], DEFAULT\_LANGUAGE=[us\_english], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF

GO

USE [Student\_dudnic]

GO

CREATE USER [dudcin] FOR LOGIN [dudcin]

GO

USE [Student\_dudnic]

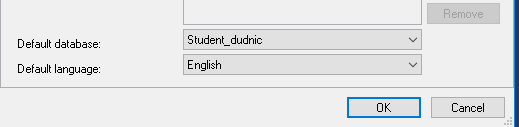
GO

ALTER ROLE [db\_owner] ADD MEMBER [dudcin]

GO

* + db\_datareader – дозволяє виконувати команду SELECT;
  + db\_datawriter – дозволяє виконувати команди INSERT, UPDATE, DELETE;
  + db\_ddladmin – дозволяє виконувати команди DDL (Data definition Language) : CREATE, ALTER, DROP, TRUNCATE;
  + db\_owner – об’єднує вище зазначені ролі.

БД по замовчуванню можна налаштувати для користувача перейшовши Explorer > Security > Logins, далі вибравши необхідного користувача вибрати Properties у вкладці General є поле Default database (яку ми встановили на нашу БД):



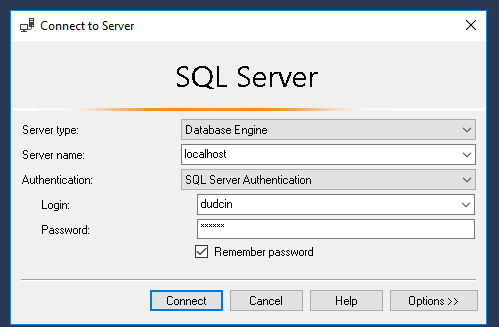
Дефолтна БД використовується при запуску скриптів, тому якщо ми використовуємо різні БД ми повинні перед скриптами додавати код:

USE [<database name>]

GO

Якщо ми цього не зробимо, то буде використовуватись дефолтна БД, тому відповідно ми її встановлюємо на БД, якою найчастіше користуємось. По замовчуванню значення дефолтної БД стоїть master.

5. Перевірка створеного користувача



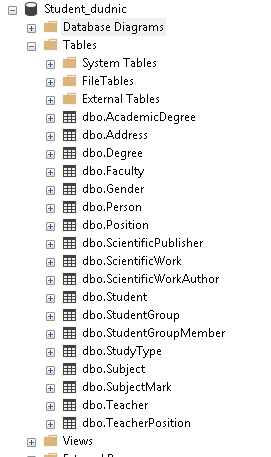
**Лабораторная работа № 2**

**Задание**

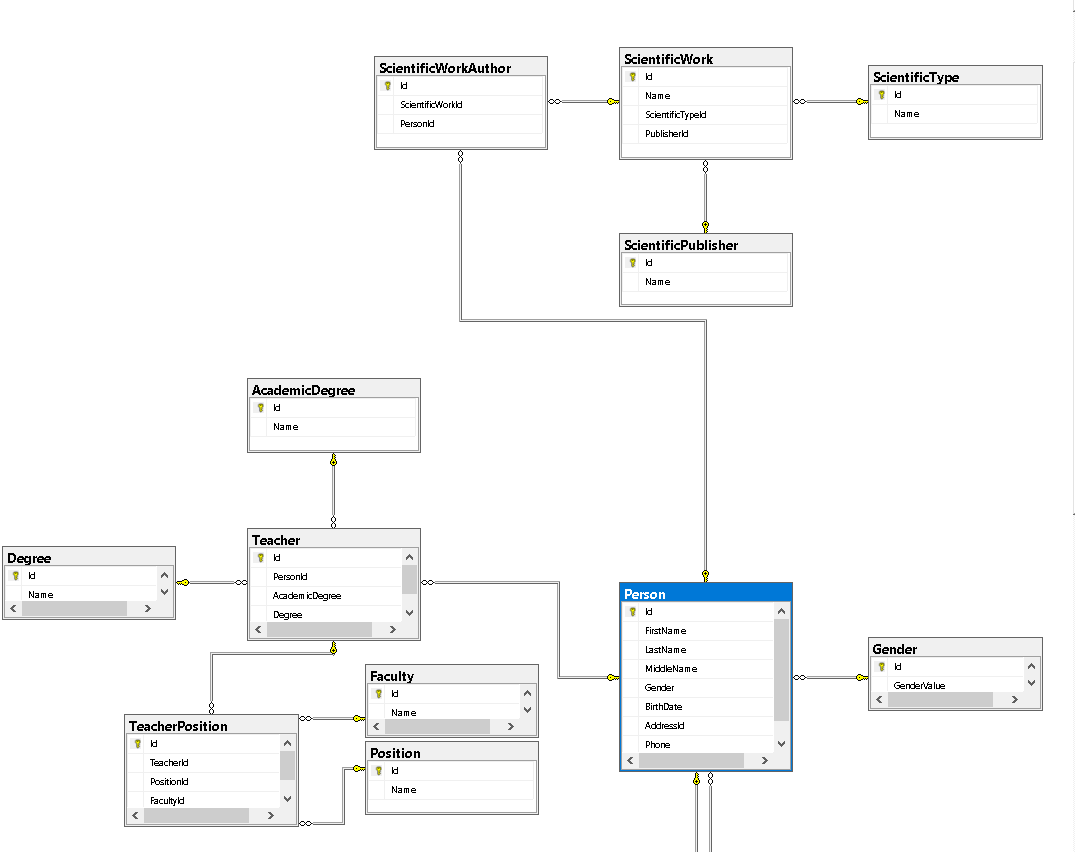
Создать логическую структуру БД, которая, будет состоять из нескольких таблиц, содержащих следующую информацию о студентах:

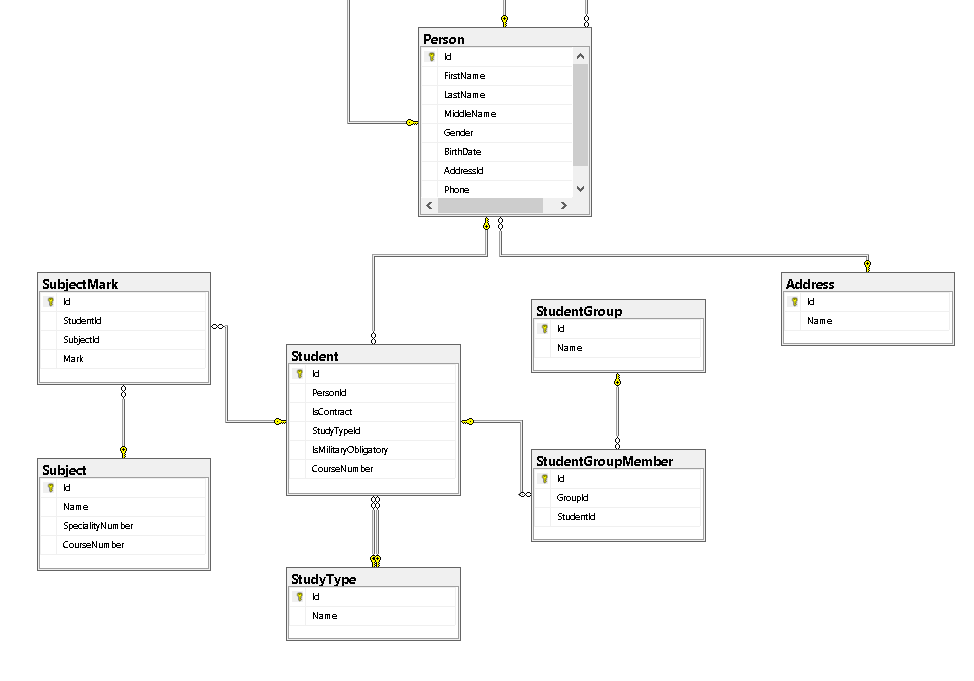
1. Фамилию.
2. Имя.
3. Отчество.
4. Пол.
5. Дату рождения.
6. Адрес (полный).
7. Телефон.
8. Учебную группу (может быть несколько).
9. Вид оплаты за обучение (бюджетное или контрактное).
10. Вид обучения (очное, заочное, дневное, вечернее, экстерн).
11. Для военнообязанных – информация о годности / негодности к службе в армии.
12. Список изученных предметов с оценками по этим предметам.
13. Список научных статей (у каждой научной статьи может быть несколько авторов).
14. Список тезисов, опубликованных в сборниках тезисов научных конференций (у тезисов докладов на конференции может быть несколько авторов).
15. Информация о преподавателях, которые являются соавторами статей и тезисов студентов. Эта информация должна включать в себя: фамилию, имя, отчество, степень, звание, должность.

Спершу генеруємо БД за допомогою скрипта GenerateDB.sql. Отримуємо такі таблиці:



За допомогою Database Diagrams > New Database Diagram отримуємо діаграму БД:





До роботи додається файл із діаграмою StudenDBDiagram та sql-файл для генерації БД, яку також можна побачити в лістингу в кінці файлу.

**Лабораторная работа № 3**

**Задание**

1. Используя команду ALTER TABLE или изменив приведенный пример создать БД, логическая модель которой создана в лабораторной работе № 2. **Для создания таблиц должен быть использовании универсальный скрипт, который создает таблицы В НЕЗАВИСИМОСТИ от их существования или несуществования (если таблица существует, то она удаляется и создается заново, проверка существования осуществляется автоматически скриптом). При этом следует отметить, что порядок создания/удаления таблиц будет разным в случае, если присутствует ограничение FOREIGN KEY. Также можно использовать универсальный скрипт удаления таблицы, предварительно удаляющий все ограничения FOREIGN KEY, ссылающиеся на нее.**

**До коду було додано наступні скрипти:**

-- ------------------------------------------Drop all foreign keys from the database

declare @sql nvarchar(max) = (

SELECT

'alter table ' + quotename(schema\_name(schema\_id)) + '.' +

quotename(object\_name(parent\_object\_id)) +

' drop constraint '+quotename(name) + ';'

FROM sys.foreign\_keys

for xml path('')

);

exec sp\_executesql @sql;

-- -------------------------------------------Drop all tables from the database

DECLARE @sql\_table NVARCHAR(max)=''

SELECT @sql\_table += ' Drop table ' + QUOTENAME(TABLE\_SCHEMA) + '.'+ QUOTENAME(TABLE\_NAME) + '; '

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE'

exec sp\_executesql @sql\_table

Перший скрипт видаляє всі foreign key, другий видаляє всі таблиці

1. Заполнить полученные таблицы данными, причем внесенные данные должны включать в себя информацию о:

* 25 студентах (5 иногородних / иностранных);
* 20 предметах;
* 10 оценках для каждого студента по имеющимся предметам;
* 10 статьях (5  совместные с преподавателями);
* 10 тезисах (5  совместные с другими студентами);
* 5 преподавателях.

**Скрипт додається файлом та наявний у лістингу у кінці файла (GenerateDBData.sql)**

1. Построить диаграмму, отображающую связи и первичные ключи полученных таблиц.

Діаграма наведена у звіті 2-гої лабораторної

1. Разработать SQL запросы, которые выполняют следующие действия:

**Для зручності перевірки в результати також додавали стовпці, які безпосередньо не вимагались у завданні**

* + Добавить ограничение для вводимых значений в поле форма обучения из таблицы groups с использованием команды ALTER TABLE.

-- Task 1

ALTER TABLE dbo.Student

ADD FOREIGN KEY (StudyTypeId) REFERENCES StudyType(Id);

GO

* + Вывести на экран Ф.И.О. (фамилию – полностью, имя и отчество – сокращенно) всех студентов специальности КМ, не живущих в общежитии.

За адресу гуртожитка береться одна із наявних адрес.

-- Task 2

DECLARE @AddressId INT, @GroupId INT

SET @AddressId = (SELECT Id from dbo.Address WHERE Name = '36832 Evergreen Hill') -- Address of a hostel

SET @GroupId = (SELECT Id FROM StudentGroup WHERE Name = 'KM') -- Group Name

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person, Address.Name AS Address FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN Address ON Person.AddressId = Address.Id

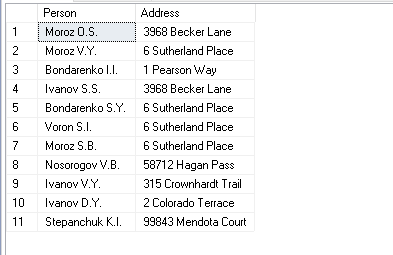
WHERE Address.Id <> @AddressId AND EXISTS(

SELECT \* FROM StudentGroupMember WHERE GroupId = @GroupId AND StudentId = Student.Id

)

GO

**Результат:**



* + Вывести на экран Ф.И.О. (фамилию – полностью, имя и отчество – сокращенно) и возраст (полных лет) всех военнообязанных студентов (мужчины, 18 <= возраст < 27, годные к срочной военной службе). Возраст определяется с точностью до дня.

-- Task 3

DECLARE @GenderId INT

SET @GenderId = (SELECT Id from dbo.Gender WHERE GenderValue = 'male') -- Male

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

Gender,

BirthDate,

FLOOR(DATEDIFF(DAY, BirthDate, SYSDATETIME()) / 365.25) AS Age,

IsMilitaryObligatory = 1

FROM Person

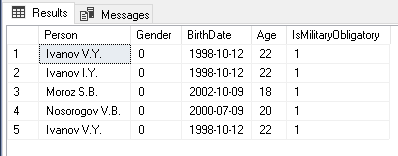
INNER JOIN Student ON Student.PersonId = Person.Id

WHERE Gender = @GenderId

AND (FLOOR(DATEDIFF(DAY, BirthDate, SYSDATETIME()) / 365.25) BETWEEN 18 AND 27)

AND IsMilitaryObligatory = 1

GO



* + Вывести на экран Ф.И.О. (фамилию – полностью, имя и отчество – сокращенно) и № курса всех студентов, которые учатся на специальностях КС и КМ и живут в общежитии.

-- Task 4

DECLARE @AddressId INT, @GroupIdKM INT, @GroupIdKC INT

SET @AddressId = (SELECT Id from dbo.Address WHERE Name = '36832 Evergreen Hill') -- Address of a hostel

SET @GroupIdKM = (SELECT Id FROM StudentGroup WHERE Name = 'KM') -- Group Names

SET @GroupIdKC = (SELECT Id FROM StudentGroup WHERE Name='KC') -- Group Names

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person, Address.Name AS Address FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

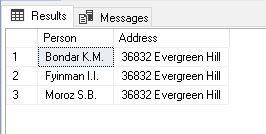
INNER JOIN Address ON Person.AddressId = Address.Id

WHERE Address.Id = @AddressId AND EXISTS(

SELECT \* FROM StudentGroupMember WHERE (GroupId = @GroupIdKM OR GroupId = @GroupIdKC) AND StudentId = Student.Id

)

GO



* + Вывести на экран Ф.И.О. (фамилию – полностью, имя и отчество – сокращенно) и номер курса всех студентов, имеющих печатные работы (статьи и тезисы), а также названия этих печатных работ и их тип (статья или тезисы).

-- Task 5

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

CourseNumber,

ScientificWork.Name AS Name,

ScientificType.Name AS Type

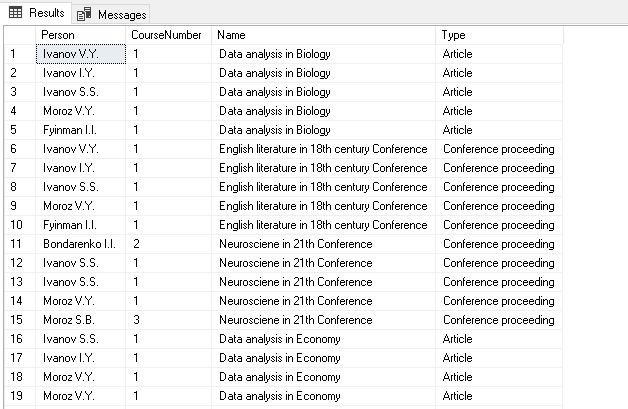
FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN ScientificWorkAuthor ON ScientificWorkAuthor.PersonId = Person.Id

INNER JOIN ScientificWork ON ScientificWork.Id = ScientificWorkAuthor.ScientificWorkId

INNER JOIN ScientificType ON ScientificType.Id = ScientificWork.ScientificTypeId



* + Вывести перечень предметов и оценки по ним для указанного студента (студент выбирается по номеру зачетки).

-- Task 6

DECLARE @StudentId INT

SET @StudentId = 2

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

Subject.Name AS Subject,

Mark

FROM Person

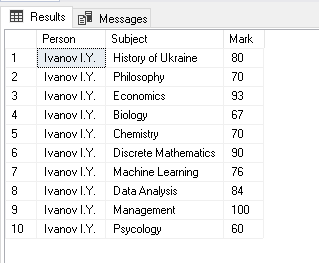
INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN SubjectMark ON SubjectMark.StudentId = Student.Id

INNER JOIN Subject ON SubjectMark.SubjectId = Subject.Id

WHERE Student.Id = @StudentId

GO



* + Используя common table expressions получить следующие данные: название предмета, код специальности, номер курса для специальностей КИ и КС.

-- Task 7

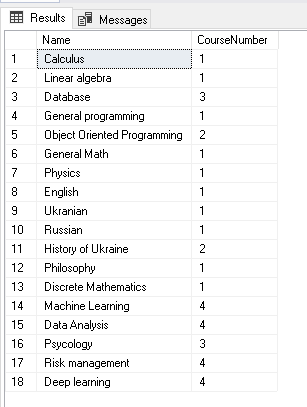
WITH SubjectsView AS(

SELECT \* FROM Subject

WHERE SpecialityNumber = 122 OR SpecialityNumber = 124

)

SELECT Name, CourseNumber FROM SubjectsView



Лістинг

**GenerateDB.sql**

USE [Student\_dudnic]

GO

-- ------------------------------------------Drop all foreign keys from the database

declare @sql nvarchar(max) = (

SELECT

'alter table ' + quotename(schema\_name(schema\_id)) + '.' +

quotename(object\_name(parent\_object\_id)) +

' drop constraint '+quotename(name) + ';'

FROM sys.foreign\_keys

for xml path('')

);

exec sp\_executesql @sql;

-- -------------------------------------------Drop all tables from the database

DECLARE @sql\_table NVARCHAR(max)=''

SELECT @sql\_table += ' Drop table ' + QUOTENAME(TABLE\_SCHEMA) + '.'+ QUOTENAME(TABLE\_NAME) + '; '

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE'

exec sp\_executesql @sql\_table

-- -------------------------------------------Drop a pecific table from the database

IF OBJECT\_ID('dbo.Gender', 'u') IS NOT NULL

DROP TABLE [dbo].[Gender]

CREATE TABLE [dbo].[Gender]

(

[Id] BIT NOT NULL PRIMARY KEY,

[GenderValue] NCHAR(6) NOT NULL

);

GO

IF OBJECT\_ID('dbo.Address', 'u') IS NOT NULL

DROP TABLE [dbo].[Address]

CREATE TABLE [dbo].[Address]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(40) NULL

)

IF OBJECT\_ID('dbo.Person', 'u') IS NOT NULL

DROP TABLE [dbo].[Person]

CREATE TABLE [dbo].[Person]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[FirstName] NCHAR(20) NULL,

[LastName] NCHAR(20) NULL,

[MiddleName] NCHAR(20) NULL,

[Gender] BIT NOT NULL,

[BirthDate] DATE NULL,

[AddressId] INT NULL,

[Phone] CHAR(15) NULL,

CONSTRAINT FK\_Gender FOREIGN KEY ([Gender])

REFERENCES Gender (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_Address FOREIGN KEY ([AddressId])

REFERENCES Address (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

IF OBJECT\_ID('dbo.Student', 'u') IS NOT NULL

DROP TABLE [dbo].[Student]

IF OBJECT\_ID('dbo.StudyType', 'u') IS NOT NULL

DROP TABLE [dbo].[StudyType]

CREATE TABLE [dbo].[StudyType]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(20)

)

GO

CREATE TABLE [dbo].[Student]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[PersonId] int NOT NULL REFERENCES Person,

[IsContract] BIT NULL,

[StudyTypeId] INT NULL REFERENCES StudyType,

[IsMilitaryObligatory] BIT NULL,

[CourseNumber] INT

)

GO

IF OBJECT\_ID('dbo.StudentGroup', 'u') IS NOT NULL

DROP TABLE [dbo].[StudentGroup]

CREATE TABLE [dbo].[StudentGroup]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(10) NULL

)

GO

CREATE TABLE [dbo].[StudentGroupMember]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[GroupId] INT NOT NULL,

[StudentId] INT NULL,

CONSTRAINT FK\_StudentId FOREIGN KEY ([StudentId])

REFERENCES Student (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_GroupId FOREIGN KEY ([GroupId])

REFERENCES StudentGroup (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

IF OBJECT\_ID('dbo.Subject', 'u') IS NOT NULL

DROP TABLE [dbo].[Subject]

CREATE TABLE [dbo].[Subject]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL,

[SpecialityNumber] INT,

[CourseNumber] INT

)

GO

IF OBJECT\_ID('dbo.SubjectMark', 'u') IS NOT NULL

DROP TABLE [dbo].[SubjectMark]

CREATE TABLE [dbo].[SubjectMark]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[StudentId] INT NULL,

[SubjectId] INT NULL,

[Mark] INT NULL,

CONSTRAINT FK\_StudentId\_SubjectMark FOREIGN KEY ([StudentId])

REFERENCES Student (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_SubjectId FOREIGN KEY ([SubjectId])

REFERENCES Subject (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

IF OBJECT\_ID('dbo.AcademicDegree', 'u') IS NOT NULL

DROP TABLE [dbo].[AcademicDegree]

CREATE TABLE [dbo].[AcademicDegree]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

IF OBJECT\_ID('dbo.Degree', 'u') IS NOT NULL

DROP TABLE [dbo].[Degree]

CREATE TABLE [dbo].[Degree]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

IF OBJECT\_ID('dbo.Teacher', 'u') IS NOT NULL

DROP TABLE [dbo].[Teacher]

CREATE TABLE [dbo].[Teacher]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[PersonId] INT NOT NULL,

[AcademicDegree] INT NULL,

[Degree] INT NULL,

CONSTRAINT FK\_RefRecId\_Teacher FOREIGN KEY ([PersonId])

REFERENCES Person (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_AcademicDegree\_Teacher FOREIGN KEY ([AcademicDegree])

REFERENCES AcademicDegree (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_DegreeTeacher FOREIGN KEY ([Degree])

REFERENCES Degree (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

IF OBJECT\_ID('dbo.Position', 'u') IS NOT NULL

DROP TABLE [dbo].[Position]

CREATE TABLE [dbo].[Position]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

IF OBJECT\_ID('dbo.Faculty', 'u') IS NOT NULL

DROP TABLE [dbo].[Faculty]

CREATE TABLE [dbo].[Faculty]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

IF OBJECT\_ID('dbo.TeacherPosition', 'u') IS NOT NULL

DROP TABLE [dbo].[TeacherPosition]

CREATE TABLE [dbo].[TeacherPosition]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[TeacherId] INT NOT NULL,

[PositionId] INT NULL,

[FacultyId] INT NULL,

CONSTRAINT FK\_TeacherId FOREIGN KEY ([TeacherId])

REFERENCES Teacher (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_PositionId FOREIGN KEY ([PositionId])

REFERENCES Position (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_FacultyId FOREIGN KEY ([FacultyId])

REFERENCES Faculty (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

IF OBJECT\_ID('dbo.ScientificPublisher', 'u') IS NOT NULL

DROP TABLE [dbo].[ScientificPublisher]

CREATE TABLE [dbo].[ScientificPublisher]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

IF OBJECT\_ID('dbo.ScientificWork', 'u') IS NOT NULL

DROP TABLE [dbo].[ScientificWork]

CREATE TABLE [dbo].[ScientificType]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(30) NULL

)

GO

CREATE TABLE [dbo].[ScientificWork]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] NCHAR(50) NOT NULL,

[ScientificTypeId] INT NOT NULL REFERENCES ScientificType,

[PublisherId] INT NULL,

CONSTRAINT FK\_PublisherId FOREIGN KEY ([PublisherId])

REFERENCES ScientificPublisher (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

IF OBJECT\_ID('dbo.ScientificWorkAuthor', 'u') IS NOT NULL

DROP TABLE [dbo].[ScientificWorkAuthor]

CREATE TABLE [dbo].[ScientificWorkAuthor]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[ScientificWorkId] INT NOT NULL,

[PersonId] INT NOT NULL,

CONSTRAINT FK\_ScientificWorkId FOREIGN KEY ([ScientificWorkId])

REFERENCES ScientificWork (Id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_PersonId\_ScientificWorkAuthor FOREIGN KEY ([PersonId])

REFERENCES Person (Id)

ON DELETE CASCADE

ON UPDATE CASCADE

)

GO

**GenerateDBData.sql**

USE [Student\_dudnic]

GO

INSERT INTO [dbo].[Gender]

([Id]

,[GenderValue])

VALUES

(0, 'male'),

(1, 'female')

GO

INSERT INTO [dbo].[Subject]

VALUES

('Calculus',122,1),

('Linear algebra',122,1),

('Database',122,3),

('General programming',122,1),

('Object Oriented Programming',122,2),

('General Math',122,1),

('Physics',122,1),

('English',122,1),

('Ukranian',122,1),

('Russian',122,1),

('History of Ukraine',122,2),

('Philosophy',122,1),

('Economics',120,1),

('Biology',110,2),

('Chemistry',110,3),

('Discrete Mathematics',122,1),

('Machine Learning',122,4),

('Data Analysis',124,4),

('Management',123,1),

('Psycology',122,3),

('Risk management',122,4),

('Deep learning',122,4)

GO

insert into dbo.Address (name) values ('58712 Hagan Pass');

insert into dbo.Address (name) values ('36 Dayton Crossing');

insert into dbo.Address (name) values ('3968 Becker Lane');

insert into dbo.Address (name) values ('1 Pearson Way');

insert into dbo.Address (name) values ('6 Sutherland Place');

insert into dbo.Address (name) values ('315 Crownhardt Trail');

insert into dbo.Address (name) values ('2 Colorado Terrace');

insert into dbo.Address (name) values ('99843 Mendota Court');

insert into dbo.Address (name) values ('36832 Evergreen Hill');

insert into dbo.Address (name) values ('33 Northland Terrace');

INSERT INTO [dbo].[Person]

([FirstName]

,[LastName]

,[MiddleName]

,[Gender]

,[BirthDate]

,[AddressId]

,[Phone])

VALUES

('Vasyliy', 'Ivanov', 'Yuriyovuch', 0, '10/12/1998', 1, '+38099123343'),

('Ignat', 'Ivanov', 'Yuriyovuch', 0, '10/12/1998', 2, '+38099123343'),

('Kateryna', 'Stepanchuk', 'Ivanivna', 1, '01/08/1997', 3, '+38066234934'),

('Kateryna', 'Bondar', 'Mykhailovna', 1, '01/06/2000', 2, '+38066224233'),

('Iryna', 'Fyinman', 'Ivanivna', 1, '01/08/1997', 3, '+3804334934'),

('Olgha', 'Moroz', 'Sergiyivna', 1, '05/02/2001', 3, '+380423434234'),

('Vasyliy', 'Moroz', 'Yuriyovuch', 0, '10/12/1998', 5, '+38099123343'),

('Ivan', 'Bondarenko', 'Ivanovuch', 0, '10/11/1995', 4, '+38099123243'),

('Sergiyi', 'Ivanov', 'Stepanovuch', 0, '10/12/1998', 3, '+380955094323'),

('Sergiyi', 'Bondarenko', 'Yuriyovuch', 0, '10/09/1990', 5, '+3809934243'),

('Sergiyi', 'Voron', 'Ivanovuch', 0, '09/23/1987', 5, '+380993234'),

('Sergiyi', 'Moroz', 'Bognanovuch', 0, '10/09/2002', 5, '+3809232243'),

('Volodymyr', 'Nosorogov', 'Bognanovuch', 0, '07/09/2000', 1, '+3802344243'),

('Vasyliy', 'Ivanov', 'Yuriyovuch', 0, '10/12/1998', 6, '+38099123343'),

('Dmytriyi', 'Ivanov', 'Yuriyovuch', 0, '11/12/1998', 7, '+38099123343'),

('Kateryna', 'Stepanchuk', 'Ivanivna', 1, '01/09/1997', 8, '+38066234934'),

('Kateryna', 'Bondar', 'Mykhailovna', 1, '01/06/2000', 9, '+38066224233'),

('Iryna', 'Fyinman', 'Ivanivna', 1, '01/08/1997', 9, '+3804334934'),

('Olgha', 'Moroz', 'Sergiyivna', 1, '05/10/2001', 9, '+380423434234'),

('Vasyliy', 'Moroz', 'Yuriyovuch', 0, '10/12/1998', 1, '+38099123343'),

('Ivan', 'Bondarenko', 'Ivanovuch', 0, '10/11/1995', 2, '+38099123243'),

('Sergiyi', 'Ivanov', 'Stepanovuch', 0, '10/12/1998', 9, '+380955094323'),

('Sergiyi', 'Bondarenko', 'Yuriyovuch', 0, '10/09/1990', 9, '+3809934243'),

('Sergiyi', 'Voron', 'Ivanovuch', 0, '09/23/1987', 9, '+380993234'),

('Sergiyi', 'Moroz', 'Bognanovuch', 0, '10/09/2002', 9, '+3809232243'),

('Volodymyr', 'Nosorogov', 'Bognanovuch', 0, '07/09/2000', 9, '+3802344243'),

('Kateryna', 'Stepanchuk', 'Ivanivna', 1, '01/09/1997', 9, '+38066234934'),

('Kateryna', 'Bondar', 'Mykhailovna', 1, '01/06/2000', 7, '+38066224233'),

('Iryna', 'Fyinman', 'Ivanivna', 1, '01/08/1997', 1, '+3804334934'),

('Olgha', 'Moroz', 'Sergiyivna', 1, '05/10/2001', 1, '+380423434234'),

('Vasyliy', 'Moroz', 'Yuriyovuch', 0, '10/12/1998', 1, '+38099123343')

GO

INSERT INTO [dbo].[AcademicDegree]

([Name])

VALUES

('None'),

('PhD')

GO

INSERT INTO [dbo].[Degree]

([Name])

VALUES

('Junior scientist'),

('Senior scientist'),

('Docent')

GO

INSERT INTO [dbo].[Position]

([Name])

VALUES

('Aspirant'),

('Professor'),

('Docent')

GO

INSERT INTO [dbo].[Faculty]

([Name])

VALUES

('Faculty 1'),

('Faculty 2'),

('Faculty 3')

GO

INSERT INTO [dbo].[ScientificPublisher]

([Name])

VALUES

('SCOPUS Journal: Science'),

('University Journal'),

('Ukraine Journal')

GO

INSERT INTO dbo.StudyType

([Name])

VALUES

('Offline'), ('Distance')

GO

INSERT INTO [dbo].[Student]

([PersonId]

,[IsContract]

,[StudyTypeId]

,[IsMilitaryObligatory]

,[CourseNumber])

VALUES

(1,1,1,1,1),

(2,1,1,1,1),

(3,1,1,0,1),

(4,1,1,0,1),

(5,1,1,0,1),

(6,0,1,0,1),

(7,0,1,0,1),

(8,0,1,0,1),

(9,0,1,0,1),

(10,0,1,0,1),

(11,0,2,1,1),

(12,0,2,1,1),

(13,0,2,1,2),

(14,0,2,1,2),

(15,0,2,0,2),

(16,0,2,0,2),

(17,0,2,0,2),

(18,0,2,0,2),

(19,0,2,0,2),

(20,0,2,0,2),

(21,1,2,0,2),

(22,1,2,0,1),

(23,0,1,0,3),

(24,0,1,0,3),

(25,0,1,0,3)

GO

INSERT INTO [dbo].[SubjectMark]

([StudentId]

,[SubjectId]

,[Mark])

VALUES

(1, 1, 100),

(1, 2, 90),

(1, 3, 95),

(1, 4, 80),

(1, 5, 70),

(1, 6, 90),

(1, 7, 76),

(1, 8, 84),

(1, 9, 100),

(1, 10, 60),

(2, 11, 80),

(2, 12, 70),

(2, 13, 93),

(2, 14, 67),

(2, 15, 70),

(2, 16, 90),

(2, 17, 76),

(2, 18, 84),

(2, 19, 100),

(2, 20, 60),

(3, 11, 83),

(3, 10, 70),

(3, 13, 95),

(3, 14, 67),

(3, 15, 75),

(3, 16, 90),

(3, 17, 79),

(3, 18, 84),

(3, 19, 78),

(3, 1, 60),

(4, 1, 84),

(4, 2, 70),

(4, 3, 95),

(4, 5, 98),

(4, 15, 75),

(4, 6, 90),

(4, 7, 95),

(4, 8, 98),

(4, 9, 96),

(4, 10, 100),

(5, 1, 80),

(5, 2, 76),

(5, 3, 90),

(5, 5, 88),

(5, 15, 75),

(5, 6, 90),

(5, 7, 95),

(5, 8, 88),

(5, 9, 96),

(5, 10, 100),

(6, 6, 100),

(6, 2, 90),

(6, 3, 95),

(6, 4, 40),

(6, 5, 50),

(6, 6, 90),

(6, 7, 70),

(6, 8, 84),

(6, 9, 80),

(6, 11, 65),

(7, 11, 60),

(7, 17, 60),

(7, 13, 63),

(7, 14, 67),

(7, 15, 70),

(7, 16, 80),

(7, 17, 56),

(7, 18, 84),

(7, 19, 100),

(7, 20, 60),

(8, 11, 88),

(8, 10, 70),

(8, 9, 93),

(8, 14, 65),

(8, 15, 75),

(8, 16, 80),

(8, 17, 79),

(8, 18, 84),

(8, 19, 78),

(8, 1, 60),

(9, 1, 89),

(9, 2, 70),

(9, 3, 95),

(9, 5, 98),

(9, 15, 75),

(9, 6, 67),

(9, 7, 90),

(9, 8, 98),

(9, 9, 96),

(9, 10, 90),

(10, 1, 80),

(10, 2, 76),

(10, 3, 90),

(10, 10, 88),

(10, 11, 71),

(10, 6, 90),

(10, 7, 91),

(10, 8, 88),

(10, 9, 96),

(10, 10, 100),

(11, 11, 100),

(11, 2, 90),

(11, 3, 95),

(11, 4, 80),

(11, 5, 70),

(11, 6, 90),

(11, 7, 76),

(11, 8, 84),

(11, 9, 100),

(11, 12, 60),

(12, 11, 80),

(12, 12, 60),

(12, 13, 93),

(12, 14, 67),

(12, 15, 70),

(12, 16, 80),

(12, 17, 76),

(12, 18, 84),

(12, 19, 100),

(12, 20, 60),

(13, 11, 83),

(13, 10, 70),

(13, 13, 95),

(13, 14, 67),

(13, 15, 75),

(13, 16, 90),

(13, 17, 79),

(13, 18, 84),

(13, 19, 78),

(13, 1, 60),

(14, 1, 84),

(14, 2, 70),

(14, 3, 95),

(14, 5, 98),

(14, 15, 75),

(14, 6, 90),

(14, 7, 95),

(14, 8, 98),

(14, 9, 96),

(14, 10, 100),

(15, 1, 80),

(15, 2, 76),

(15, 3, 90),

(15, 5, 88),

(15, 15, 75),

(15, 6, 90),

(15, 7, 95),

(15, 8, 88),

(15, 9, 96),

(15, 10, 100),

(16, 6, 100),

(16, 2, 90),

(16, 3, 95),

(16, 4, 40),

(16, 5, 50),

(16, 6, 90),

(16, 7, 70),

(16, 8, 84),

(16, 9, 80),

(16, 11, 65),

(17, 11, 60),

(17, 17, 60),

(17, 13, 63),

(17, 14, 67),

(17, 15, 70),

(17, 16, 80),

(17, 17, 56),

(17, 18, 84),

(17, 19, 100),

(17, 20, 60),

(18, 11, 88),

(18, 10, 70),

(18, 9, 93),

(18, 14, 65),

(18, 15, 75),

(18, 16, 80),

(18, 17, 79),

(18, 18, 84),

(18, 19, 78),

(18, 1, 60),

(19, 1, 89),

(19, 2, 70),

(19, 3, 95),

(19, 5, 98),

(19, 15, 75),

(19, 6, 67),

(19, 7, 90),

(19, 8, 98),

(19, 9, 96),

(19, 10, 90),

(20, 1, 100),

(20, 2, 90),

(20, 3, 95),

(20, 4, 80),

(20, 5, 70),

(20, 6, 90),

(20, 7, 76),

(20, 8, 84),

(20, 9, 100),

(20, 10, 60),

(21, 1, 100),

(21, 2, 90),

(21, 3, 95),

(21, 4, 80),

(21, 5, 70),

(21, 6, 90),

(21, 7, 76),

(21, 8, 84),

(21, 9, 100),

(21, 10, 60),

(22, 11, 80),

(22, 12, 70),

(22, 13, 93),

(22, 14, 67),

(22, 15, 70),

(22, 16, 90),

(22, 17, 76),

(22, 18, 84),

(22, 19, 100),

(22, 20, 60),

(23, 11, 83),

(23, 10, 70),

(23, 13, 95),

(23, 14, 67),

(23, 15, 75),

(23, 16, 90),

(23, 17, 79),

(23, 18, 84),

(23, 19, 78),

(23, 1, 60),

(24, 1, 84),

(24, 2, 70),

(24, 3, 95),

(24, 5, 98),

(24, 15, 75),

(24, 6, 90),

(24, 7, 95),

(24, 8, 98),

(24, 9, 96),

(24, 10, 100),

(25, 1, 80),

(25, 2, 76),

(25, 3, 90),

(25, 5, 88),

(25, 15, 75),

(25, 6, 90),

(25, 7, 95),

(25, 8, 88),

(25, 9, 96),

(25, 10, 100)

GO

INSERT INTO [dbo].[StudentGroup]

([Name])

VALUES

('KM'), ('KU'),('KC')

GO

INSERT INTO [dbo].[StudentGroupMember]

([GroupId]

,[StudentId])

VALUES

(3, 25), (3, 1), (3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (3, 11), (3, 12),

(1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11), (1, 12), (1, 13), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18),

(2, 20), (2, 19), (2, 21), (2,22), (2, 23), (2,24), (2, 25)

GO

INSERT INTO [dbo].[Teacher]

([PersonId]

,[AcademicDegree]

,[Degree])

VALUES

(21, 1, 1), (22, 1, 2), (23, 1, 3), (24, 2, 1), (25, 2, 2)

GO

INSERT INTO [dbo].[TeacherPosition]

([TeacherId]

,[PositionId]

,[FacultyId])

VALUES

(1, 1, 1), (1, 2, 2), (2, 2, 3), (2, 1, 2), (3, 1, 1), (4, 2, 3), (5, 2, 2), (5, 1, 3)

GO

INSERT INTO [dbo].[ScientificType]

VALUES

('Article'), ('Conference proceeding')

GO

INSERT INTO [dbo].[ScientificWork]

([Name]

,[ScientificTypeId]

,[PublisherId])

VALUES

('Data analysis in Biology', 1, 1), ('Data analysis in Chemistry', 1, 1), ('Data analysis in Economy', 1, 2),

('Comparison of cutting edge approaches', 1, 1), ('SQL Database approaches', 1, 1), ('Data mining in data science', 1,3),

('English literature in 17th century', 1, 3), ('English literature in 19th century', 1, 1), ('English literature in 18th century', 1,3),

('Neurosciene in 21th', 1,3),

('Data analysis in Biology Conference', 2, 1), ('Data analysis in Chemistry Conference', 2, 1), ('Data analysis in Economy Conference', 2, 2),

('Comparison of cutting edge approaches Conference', 2, 1), ('SQL Database approaches Conference', 2, 1), ('Data mining in data science Conference', 2,3),

('English literature in 17th century Conference', 2, 3), ('English literature in 19th century Conference', 2, 1), ('English literature in 18th century Conference', 2,3),

('Neurosciene in 21th Conference', 2,3)

GO

INSERT INTO [dbo].[ScientificWorkAuthor]

([ScientificWorkId]

,[PersonId])

VALUES

(1, 1), (1, 2), (1, 9), (1, 7), (1, 5), (19, 1), (19, 2), (19, 9), (19, 7), (19, 5),(20, 21), (20, 22), (20, 9), (20, 7), (20, 25),

(3, 9), (3, 2), (3, 7), (3, 7), (3, 5), (4, 9), (4, 2), (4, 7), (4, 7), (4, 5),(5, 9), (5, 2), (5, 7), (5, 7), (5, 25),

(9, 9), (9, 2), (9, 7), (9, 8), (9, 9), (7, 9), (7, 2), (7, 7), (7, 8), (7, 9),(8, 9), (8, 2), (8, 7), (8, 8), (8, 9),

(9, 9), (9, 2), (9, 7), (9, 8), (9, 9), (10, 9), (10, 2), (10, 7), (10, 8), (10, 9),(11, 9), (11, 22), (11, 7), (11, 8), (11, 9),

(12, 1), (12, 2), (12, 9), (12, 7), (12, 5), (19, 21), (19, 22), (19, 9), (19, 7), (19, 25),(20, 1), (20, 2), (20, 9), (20, 7), (20, 5),

(19, 9), (19, 7), (15, 8), (15, 9), (15, 25), (19, 1), (19, 2), (19, 9), (19, 7), (19, 5),(20, 1), (20, 2), (20, 9), (20, 7), (20, 5),

(18, 1), (18, 2), (18, 9), (18, 20), (18, 5), (19, 1), (19, 22), (19, 9), (19, 27), (19, 25),(20, 21), (20, 2), (20, 9), (20, 7), (20, 25)

GO

**DBSelect.sql**

USE [Student\_dudnic]

GO

-- Task 1

ALTER TABLE dbo.Student

ADD FOREIGN KEY (StudyTypeId) REFERENCES StudyType(Id);

GO

-- Task 2

DECLARE @AddressId INT, @GroupId INT

SET @AddressId = (SELECT Id from dbo.Address WHERE Name = '36832 Evergreen Hill') -- Address of a hostel

SET @GroupId = (SELECT Id FROM StudentGroup WHERE Name = 'KM') -- Group Name

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person, Address.Name AS Address FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN Address ON Person.AddressId = Address.Id

WHERE Address.Id <> @AddressId AND EXISTS(

SELECT \* FROM StudentGroupMember WHERE GroupId = @GroupId AND StudentId = Student.Id

)

GO

-- Task 3

DECLARE @GenderId INT

SET @GenderId = (SELECT Id from dbo.Gender WHERE GenderValue = 'male') -- Male

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

Gender,

BirthDate,

FLOOR(DATEDIFF(DAY, BirthDate, SYSDATETIME()) / 365.25) AS Age,

IsMilitaryObligatory = 1

FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

WHERE Gender = @GenderId

AND (FLOOR(DATEDIFF(DAY, BirthDate, SYSDATETIME()) / 365.25) BETWEEN 18 AND 27)

AND IsMilitaryObligatory = 1

GO

-- Task 4

DECLARE @AddressId INT, @GroupIdKM INT, @GroupIdKC INT

SET @AddressId = (SELECT Id from dbo.Address WHERE Name = '36832 Evergreen Hill') -- Address of a hostel

SET @GroupIdKM = (SELECT Id FROM StudentGroup WHERE Name = 'KM') -- Group Names

SET @GroupIdKC = (SELECT Id FROM StudentGroup WHERE Name='KC') -- Group Names

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person, Address.Name AS Address FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN Address ON Person.AddressId = Address.Id

WHERE Address.Id = @AddressId AND EXISTS(

SELECT \* FROM StudentGroupMember WHERE (GroupId = @GroupIdKM OR GroupId = @GroupIdKC) AND StudentId = Student.Id

)

GO

-- Task 5

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

CourseNumber,

ScientificWork.Name AS Name,

ScientificType.Name AS Type

FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN ScientificWorkAuthor ON ScientificWorkAuthor.PersonId = Person.Id

INNER JOIN ScientificWork ON ScientificWork.Id = ScientificWorkAuthor.ScientificWorkId

INNER JOIN ScientificType ON ScientificType.Id = ScientificWork.ScientificTypeId

-- Task 6

DECLARE @StudentId INT

SET @StudentId = 2

SELECT CONCAT(RTRIM(LastName), ' ', LEFT(FirstName, 1), '.', LEFT(MiddleName, 1), '.') AS Person,

Subject.Name AS Subject,

Mark

FROM Person

INNER JOIN Student ON Student.PersonId = Person.Id

INNER JOIN SubjectMark ON SubjectMark.StudentId = Student.Id

INNER JOIN Subject ON SubjectMark.SubjectId = Subject.Id

WHERE Student.Id = @StudentId

GO

-- Task 7

WITH SubjectsView AS(

SELECT \* FROM Subject

WHERE SpecialityNumber = 122 OR SpecialityNumber = 124

)

SELECT Name, CourseNumber FROM SubjectsView