# Database Basics MSSQL Exam – 17 Feb 2019

Exam problems for the [“Database Basics” course @ SoftUni](https://softuni.bg/courses/databases-basics-ms-sql-server).

Submit your solutions in the SoftUni Judge system at <https://judge.softuni.bg/>

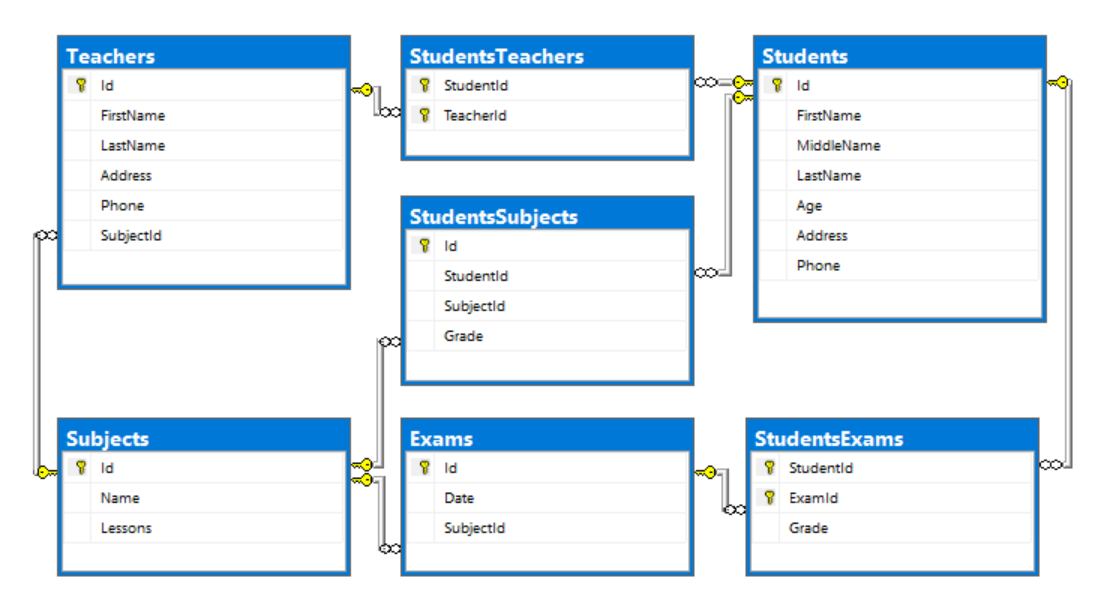
# School

Don’t be so stressed! Today you must build a very simple school system and execute some queries over it to check if it works correctly. From the very beginning **SoftUni** saw a huge potential in you and has assigned you a very exciting project. In **4 hours**, you must develop a complicated system for a small school.

# Your database must contain information about the students with their teachers and exams. Also, it must contain information about the subjects in the school

# Section 1. DDL (30 pts)

You are given an E/R Diagram of the School:



Crеate a database called **School**. You need to create **7 tables**:

* **Students** – contains information about the students.
* Subjects – contains information about the subjects.
* StudentsSubjects – contains information about every student’s subjects.
* Exams – contains information about the exams.
* StudentsExams – contains information about every student’s exams.
* Teachers – contains information about the teachers.
* StudentsTeachers – contains information about every student’s teachers.

### Students

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| FirstName | **String** up to **30** symbols, Unicode | **NULL** is **not** allowed |
| MiddleName | **String** up to **25** symbols, Unicode | None |
| LastName | **String** up to **30** symbols, Unicode | **NULL** is **not** allowed |
| Age | **Integer** from **5** to **100** | **Negative or zero numbers** are **not allowed** |
| Address | **String** up to **50** symbols, Unicode | None |
| Phone | **String** with **exactly** 10 symbols, Unicode | None |

### Subjects

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Name | **String** up to **20** symbols, Unicode | **NULL** is **not** allowed |
| Lessons | **Integer** must be more than 0 | **NULL** is **not** allowed |

### StudentsSubjects

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| StudentId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Students** |
| SubjectId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Subjects** |
| Grade | **Decimal** number with **two-digit** precision | **Grade** must be between 2 and 6, **NULL** is **not** allowed |

### Exams

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Date | **DateTime** | None |
| SubjectId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Subjects** |

### StudentsExams

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| StudentId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Students** |
| ExamId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Exams** |
| Grade | **Decimal** number with **two-digit** precision | **Grade** must be between 2 and 6, **NULL** is **not** allowed |

### Teachers

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| FirstName | **String** up to **20** symbols, Unicode | **NULL** is **not** allowed |
| LastName | **String** up to **20** symbols, Unicode | **NULL** is **not** allowed |
| Address | **String** up to **20** symbols, Unicode | **NULL** is **not** allowed |
| Phone | **String** with **exactly** **10** symbols | None |
| SubjectId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Subjects** |

### StudentsTeachers

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| StudentId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Students** |
| TeacherId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Teachers** |

## Database Design

Submit all of yours **create** **statements** to the **Judge** system.

CREATE TABLE Students(

[Id] INT PRIMARY KEY IDENTITY,

[FirstName] NVARCHAR(30) NOT NULL,

[MiddleName] NVARCHAR(25),

[LastName] NVARCHAR(30) NOT NULL,

[Age] INT CHECK ([Age] BETWEEN 5 AND 100) NOT NULL,

[Address] NVARCHAR(50),

[Phone] NCHAR(10)

)

CREATE TABLE Subjects(

[Id] INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(20) NOT NULL,

[Lessons] INT CHECK([Lessons] > 0) NOT NULL

)

CREATE TABLE StudentsSubjects(

[Id] INT PRIMARY KEY IDENTITY,

[StudentId] INT FOREIGN KEY REFERENCES Students(Id) NOT NULL,

[SubjectId] INT FOREIGN KEY REFERENCES Subjects(Id) NOT NULL,

[Grade] DECIMAL (18,2) CHECK ([Grade] BETWEEN 2 AND 6) NOT NULL

)

CREATE TABLE Exams(

[Id] INT PRIMARY KEY IDENTITY,

[Date] DATETIME,

[SubjectId] INT FOREIGN KEY REFERENCES Subjects(Id) NOT NULL

)

CREATE TABLE StudentsExams(

[StudentId] INT FOREIGN KEY REFERENCES Students(Id) NOT NULL,

[ExamId] INT FOREIGN KEY REFERENCES Exams(Id) NOT NULL,

[Grade] DECIMAL (18,2) CHECK ([Grade] BETWEEN 2 AND 6) NOT NULL

CONSTRAINT PK\_StudentIdExamID

PRIMARY KEY (StudentId, ExamId)

)

CREATE TABLE Teachers (

[Id] INT PRIMARY KEY IDENTITY,

[FirstName] NVARCHAR(20) NOT NULL,

[LastName] NVARCHAR(20) NOT NULL,

[Address] NVARCHAR(20) NOT NULL,

[Phone] CHAR(10),

[SubjectId] INT FOREIGN KEY REFERENCES Subjects(Id) NOT NULL

)

CREATE TABLE StudentsTeachers(

[StudentId] INT FOREIGN KEY REFERENCES Students(Id) NOT NULL,

[TeacherId] INT FOREIGN KEY REFERENCES Teachers(Id) NOT NULL,

CONSTRAINT PK\_StudentIdTeacherID

PRIMARY KEY (StudentId, TeacherId)

)

# Section 2. DML (10 pts)

**Before you start, you must import “**DataSet-School.sql**”. If you have created the structure correctly, the data should be successfully inserted without any errors.**

In this section, you have to do some data manipulations:

## Insert

**Insert** some sample data into the database. Write a query to add the following records into the corresponding tables. **All Ids should be auto-generated**.

INSERT INTO Teachers

([FirstName], [LastName], [Address], [Phone], [SubjectId])

VALUES

('Ruthanne', 'Bamb','84948 Mesta Junction', 3105500146, 6),

('Gerrard', 'Lowin','370 Talisman Plaza', 3324874824, 2),

('Merrile', 'Lambdin','81 Dahle Plaza', 4373065154, 5),

('Bert', 'Ivie','2 Gateway Circle', 4409584510, 4)

INSERT INTO Subjects

([Name], [Lessons])

VALUES

('Geometry', 12),

('Health', 10),

('Drama', 7),

('Sports', 9)

**Teachers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FirstName** | **LastName** | **Address** | **Phone** | **SubjectId** |
| Ruthanne | Bamb | 84948 Mesta Junction | 3105500146 | 6 |
| Gerrard | Lowin | 370 Talisman Plaza | 3324874824 | 2 |
| Merrile | Lambdin | 81 Dahle Plaza | 4373065154 | 5 |
| Bert | Ivie | 2 Gateway Circle | 4409584510 | 4 |

**Subjects**

|  |  |
| --- | --- |
| **Name** | **Lessons** |
| Geometry | 12 |
| Health | 10 |
| Drama | 7 |
| Sports | 9 |

## Update

Make all grades 6.00, where the subject id is 1 or 2, if the **grade** is above or equal to 5.50

UPDATE StudentsSubjects

SET Grade = 6.00

WHERE SubjectId IN (1, 2) AND Grade >= 5.50

## Delete

Delete all teachers, whose phone number contains ‘72’.

DELETE

FROM StudentsTeachers

WHERE TeacherId IN

(SELECT Id

FROM Teachers

WHERE Phone LIKE '%72%')

DELETE

FROM Teachers

WHERE Phone LIKE '%72%'

# Section 3. Querying (40 pts)

**You need to start with a fresh dataset, so recreate your DB and import the sample data again (**DataSet-School.sql**).**

## Teen Students

Select all **students** who are teenagers (their age is above or equal to 12). Order them by **first name (alphabetically)**, then by **last name (alphabetically)**. Select their first name, last name and their age.

SELECT FirstName, LastName, Age

FROM Students

WHERE Age >= 12

ORDER BY FirstName, LastName

### Example

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **Age** |
| Agace | Sneddon | 12 |
| Andres | Colliard | 12 |
| Brose | Yeats | 13 |
| Casper | Tite | 12 |
| … | … | … |

## Students Teachers

Select all students and the count of teachers each one has.

SELECT s.FirstName, s.LastName, COUNT(st.TeacherId) AS [TeachersCount]

FROM Students as s

JOIN StudentsTeachers as st

ON s.Id = st.StudentId

GROUP BY s.FirstName, s.LastName

### Example

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **TeachersCount** |
| Sandy | Abbison | 10 |
| Baxter | Abrahart | 13 |
| Demott | Addison | 13 |
| Deane | Adess | 10 |
| … | … | ... |

## Students to Go

Find all students, who have not attended an exam. Select their full name (first name + last name).

Order the results by full name (**ascending**)**.**

SELECT CONCAT(s.FirstName, ' ', s.LastName) AS [Full Name]

FROM Students AS s

LEFT JOIN StudentsExams AS se

ON s.Id = se.StudentId

WHERE se.ExamId IS NULL

ORDER BY [Full Name] ASC

### Example

|  |
| --- |
| **Full Name** |
| Bernardine Purrier |
| … |

## Top Students

Find top 10 students, who have highest average grades from the exams.

Format the grade, two symbols after the decimal point.

Order them by grade (**descending**), then by first name (**ascending**), then by last name (**ascending**)

SELECT TOP(10) s.FirstName, s.LastName, CONVERT(DECIMAL(3,2), AVG(se.Grade)) AS [Grade]

FROM Students AS s

JOIN StudentsExams AS se

ON s.Id = se.StudentId

GROUP BY s.FirstName, s.LastName

ORDER BY AVG(se.Grade) DESC, s.FirstName, s.LastName ASC

### Example

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Grade** |
| Lurlene | Orgee | 6.00 |
| Ivy | Bilovsky | 5.70 |
| Chariot | Giacobbo | 5.50 |
| … | … |  |

## Not So In The Studying

Find all students **who don’t have any subjects**. Select **their full name**. The full name is combination of first name, middle name and last name. Order the result by **full name**

**NOTE**: If the middle name is null you have to concatenate the first name and last name separated with single space.

SELECT s.FirstName + ' ' + ISNULL(s.MiddleName + ' ', '') + s.LastName AS [Full Name]

FROM Students AS s

LEFT JOIN StudentsSubjects as ss

ON s.Id = ss.StudentId

WHERE ss.SubjectId IS NULL

ORDER BY [Full Name]

### Example

|  |
| --- |
| **Full Name** |
| Allen Storre Piniur |
| Andria Geleman Andrioletti |
| Ashley Morecombe Summerell |
| Bobby Leggitt Domnin |
| … |

## Average Grade per Subject

Find the **average** **grade** for each subject. Select the subject name and the average grade.

Sort them by **subject id (ascending)**.

SELECT s.Name, AVG(ss.Grade) AS [AverageGrade]

FROM Subjects AS s

JOIN StudentsSubjects AS ss

ON s.Id = ss.SubjectId

GROUP BY ss.SubjectId, s.Name

ORDER BY ss.SubjectId ASC

### Example

|  |  |
| --- | --- |
| **Name** | **AverageGrade** |
| Biology | 4.059055 |
| History | 3.880370 |
| English | 4.060546 |
| Math | 3.957876 |
| Music | 3.923984 |
| Art | 4.070898 |
| … | … |

# Section 4. Programmability (20 pts)

## Exam Grades

Create a **user defined function**, named **udf\_ExamGradesToUpdate(@studentId, @grade)**, that receives a **student id and grade**.

The function should return the count of grades, for the student with the given id, which are above the received grade and under the received grade with **0.50** added (**example:** you are given grade **3.50** and you have to find all grades for the provided student which are between **3.50** and **4.00** inclusive):

If the condition is true, you must return following message in the format:

* “**You have to update {count} grades for the student {student first name}**”

If the provided student id is not in the database the function should return “The student with provided id does not exist in the school!”

If the provided grade is above **6.00** the function should return “Grade cannot be above 6.00!**”**

**Note: Do not update any records in the database!**

CREATE FUNCTION udf\_ExamGradesToUpdate(@studentId INT, @grade DECIMAL (3, 2))

RETURNS VARCHAR(MAX)

AS

BEGIN

DECLARE @studentIdExistsOrNot INT = (SELECT s.Id

FROM Students AS s

WHERE s.Id = @studentId)

IF(@grade > 6.00)

BEGIN

RETURN 'Grade cannot be above 6.00!'

END

IF(@studentIdExistsOrNot IS NULL)

BEGIN

RETURN 'The student with provided id does not exist in the school!'

END

DECLARE @count INT = (SELECT COUNT(\*)

FROM StudentsExams AS se

JOIN Students AS s

ON se.StudentId = s.Id

WHERE s.Id = @studentId AND se.Grade BETWEEN @grade AND @grade+0.50)

DECLARE @name NVARCHAR(60) = (SELECT s.FirstName

FROM Students AS s

WHERE @studentId = s.Id)

RETURN CONCAT('You have to update ', @count, ' grades for the student ', @name)

END

### Example:

|  |
| --- |
| **Query** |
| **SELECT dbo.udf\_ExamGradesToUpdate(12, 6.20)** |
| **Output** |
| Grade cannot be above 6.00! |

|  |
| --- |
| **Query** |
| **SELECT dbo.udf\_ExamGradesToUpdate(12, 5.50)** |
| **Output** |
| **You have to update 2 grades for the student Agace** |

|  |
| --- |
| **Query** |
| **SELECT dbo.udf\_ExamGradesToUpdate(121, 5.50)** |
| **Output** |
| The student with provided id does not exist in the school! |

## Exclude from school

Create a **user defined stored procedure**, named **usp\_ExcludeFromSchool(@StudentId)**, that receives a **student id** and attempts to **delete the current student**. A student will only be deleted if all of these conditions **pass**:

* If the **student** doesn’t exist, then it **cannot be deleted.** **Raise an error** with the message “This school has no student with the provided id!”

If all the above conditions pass, **delete the student and ALL OF HIS REFERENCES**!

CREATE PROCEDURE usp\_ExcludeFromSchool(@StudentId INT)

AS

BEGIN

DECLARE @studentIdExistsOrNot INT = (SELECT s.Id

FROM Students AS s

WHERE s.Id = @StudentId)

IF(@studentIdExistsOrNot IS NULL)

BEGIN

RAISERROR('This school has no student with the provided id!', 18, 2)

END

DELETE

FROM StudentsExams

WHERE StudentId = @StudentId

DELETE

FROM StudentsSubjects

WHERE StudentId = @StudentId

DELETE

FROM StudentsTeachers

WHERE StudentId = @StudentId

DELETE

FROM Students

WHERE Id = @StudentId

END

### Example usage:

|  |  |
| --- | --- |
| **Query** | **Output** |
| **EXEC usp\_ExcludeFromSchool 1**  **SELECT COUNT(\*) FROM Students** | **119** |
| **EXEC usp\_ExcludeFromSchool 301** | This school has no student with the provided id! |