**Лабораторная работа № 8. Основы программирования на T-SQL**

Оглавление

[Задание 1 1](#_Toc163574410)

[Задание 2 2](#_Toc163574411)

[Задание 3 2](#_Toc163574412)

[Задание 4 3](#_Toc163574413)

[Задание 5 4](#_Toc163574414)

[Задание 6 5](#_Toc163574415)

[Задание 7 6](#_Toc163574416)

[Задание 8 6](#_Toc163574417)

Задание 1

***Условие*:**  
Разработать скрипт, в котором:

− объявить переменные типа char, varchar, datetime, time, int, smallint, tinint, numeric(12, 5);

− первые две переменные проинициализировать в операторе объявления;

− присвоить произвольные значения переменным с помощью операторов SET и SELECT;

− значения одних переменных вывести с помощью оператора SELECT,значения других переменных распечатать с помощью оператора PRINT.

Проанализировать результаты.

***Решение:***

-- Объявление

DECLARE @charVar CHAR(10) = 'Test',

@varcharVar VARCHAR(30) = 'Test String',

@datetimeVar DATETIME,

@timeVar TIME,

@intVar INT,

@smallintVar SMALLINT,

@tinyintVar TINYINT,

@numericVar NUMERIC(12, 5)

-- Присвоение произвольных значений

SET @datetimeVar = GETDATE()

SET @timeVar = GETDATE()

SELECT @intVar = 123

SET @smallintVar = 12

SET @tinyintVar = 1

SET @numericVar = 12345.12345

-- Вывод значений (SELECT)

SELECT @charVar AS CharVar,

@varcharVar AS VarcharVar,

@datetimeVar AS DatetimeVar,

@timeVar AS TimeVar

-- Вывод значений (PRINT)

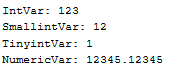
PRINT 'IntVar: ' + CAST(@intVar AS VARCHAR)

PRINT 'SmallintVar: ' + CAST(@smallintVar AS VARCHAR)

PRINT 'TinyintVar: ' + CAST(@tinyintVar AS VARCHAR)

PRINT 'NumericVar: ' + CAST(@numericVar AS VARCHAR)

***Результат:***

Задание 2

***Условие*:**  
Разработать скрипт, в котором определяется общая вместимость аудиторий.

Если общая вместимость превышает 200, то вывести количество аудиторий, среднюю вместимость аудиторий, количество аудиторий, вместимость которых меньше средней, и процент таких аудиторий.

Если общая вместимость аудиторий меньше 200, то вывести сообщение о размере общей вместимости.

***Решение:***

DECLARE @totalCapacity INT, @avgCapacity FLOAT, @countLessAvg INT, @totalAuditoriums INT

SELECT @totalCapacity = SUM(AUDITORIUM\_CAPACITY) FROM AUDITORIUM

IF @totalCapacity > 200

BEGIN

SELECT @avgCapacity = AVG(AUDITORIUM\_CAPACITY) FROM AUDITORIUM

SELECT @countLessAvg = COUNT(\*) FROM AUDITORIUM WHERE AUDITORIUM\_CAPACITY < @avgCapacity

SELECT @totalAuditoriums = COUNT(\*) FROM AUDITORIUM

SELECT @totalAuditoriums AS TotalAuditoriums, @avgCapacity AS AvgCapacity, @countLessAvg AS CountLessAvg, (@countLessAvg \* 100.0 / @totalAuditoriums) AS PercentLessAvg

END

ELSE

BEGIN

PRINT 'Общая вместимость аудиторий: ' + CAST(@totalCapacity AS VARCHAR)

END

***Результат:***



Задание 3

***Условие*:**  
Разработать скрипт, который выводит на печать глобальные переменные:

− @@ROWCOUNT (число обработанных строк);

− @@VERSION (версия SQL Server);

− @@SPID (возвращает системный идентификатор процесса, назначенный сервером текущему подключению);

− @@ERROR (код последней ошибки);

− @@SERVERNAME (имя сервера);

− @@TRANCOUNT (возвращает уровень вложенности транзакции);

− @@FETCH\_STATUS (проверка результата считывания строк результирующего набора);

− @@NESTLEVEL (уровень вложенности текущей процедуры).

Проанализировать результат.

***Решение:***

SELECT

@@ROWCOUNT AS 'Число обработанных строк',

@@VERSION AS 'Версия SQL Server',

@@SPID AS 'Системный идентификатор процесса',

@@ERROR AS 'Код последней ошибки',

@@SERVERNAME AS 'Имя сервера',

@@TRANCOUNT AS 'Уровень вложенности транзакции',

@@FETCH\_STATUS AS 'Проверка результата считывания строк результирующего набора',

@@NESTLEVEL AS 'Уровень вложенности текущей процедуры'

***Результат:***



Задание 4

***Условие*:**  
Разработать T-SQL-скрипты, выполняющие:

− вычисление значений переменной **z**



для различных значений исходных данных;

− поиск студентов, у которых день рождения в следующем месяце, и определение их возраста;

− поиск дня недели, в который студенты некоторой группы сдавали экзамен по БД.

***Решение:***

--a

DECLARE @t FLOAT, @x FLOAT, @z FLOAT

SET @t = 0.5

SET @x = 1.0

IF @t > @x

SET @z = SQUARE(SIN(@t))

ELSE IF @t < @x

SET @z = 4 \* (@t + @x)

ELSE

SET @z = 1 - EXP(@x - 2)

SELECT @z AS z

--b

SELECT NAME, DATEDIFF(YEAR, BDAY, GETDATE()) AS Age

FROM STUDENT

WHERE MONTH(BDAY) = MONTH(DATEADD(MONTH, 1, GETDATE()))

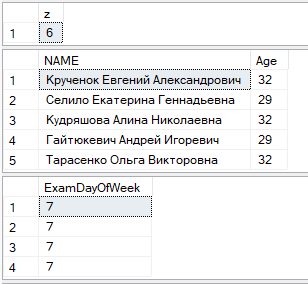
--с

SELECT DATEPART(WEEKDAY, PDATE) AS ExamDayOfWeek

FROM PROGRESS

WHERE SUBJECT = 'СУБД' AND IDSTUDENT IN (SELECT IDSTUDENT FROM STUDENT WHERE IDGROUP = 5)

***Результат:***



Задание 5

***Условие*:**  
Разработать сценарий, в котором с помощью CASE анализируются оценки, полученные студентами некоторого факультета при сдаче экзаменов.

***Решение:***

DECLARE @facultyName VARCHAR(50)

SET @facultyName = 'ИТ'

SELECT

STUDENT.NAME,

PROGRESS.NOTE,

CASE

WHEN PROGRESS.NOTE >= 9 THEN 'Отлично'

WHEN PROGRESS.NOTE >= 7 THEN 'Хорошо'

WHEN PROGRESS.NOTE >= 5 THEN 'Удовлетворительно'

ELSE 'Неудовлетворительно'

END AS Grade

FROM

STUDENT

JOIN

PROGRESS ON STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

JOIN

GROUPS ON STUDENT.IDGROUP = GROUPS.IDGROUP

JOIN

PROFESSION ON GROUPS.PROFESSION = PROFESSION.PROFESSION

JOIN

FACULTY ON PROFESSION.FACULTY = FACULTY.FACULTY

WHERE

FACULTY.FACULTY = @facultyName

***Результат:***



Задание 6

***Условие*:**  
Создать временную локальную таблицу из трех столбцов и 10 строк, заполнить ее и вывести содержимое. Использовать оператор WHILE.

***Решение:***

CREATE TABLE #TempTable

(

Column1 INT,

Column2 INT,

Column3 INT

)

DECLARE @i INT = 1

WHILE @i <= 10

BEGIN

INSERT INTO #TempTable (Column1, Column2, Column3)

VALUES (@i, @i \* 10, @i \* 100)

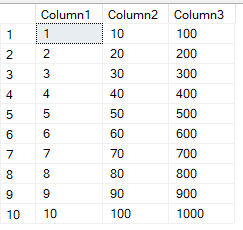
SET @i = @i + 1

END

SELECT \* FROM #TempTable

DROP TABLE #TempTable

***Результат:***



Задание 7

***Условие*:**  
Разработать скрипт, демонстрирующий использование оператора RETURN.

***Решение:***

DECLARE @num INT

SET @num = 5

IF @num < 0

BEGIN

PRINT 'Введенное число отрицательное.'

RETURN

END

PRINT 'Введенное число положительное.'

***Результат:***



Задание 8

***Условие*:**  
Разработать сценарий с ошибками, в котором используются для обработки ошибок блоки TRY и CATCH.

Применить функции ERROR\_NUMBER (код последней ошибки), ERROR\_MESSAGE (сообщение об ошибке), ERROR\_LINE (номер строки с ошибкой), ERROR\_PROCEDURE (имя процедуры или NULL), ERROR\_SEVERITY (уровень серьезности ошибки), ERROR\_STATE (метка ошибки). Проанализировать результат.

***Решение:***

BEGIN TRY

DECLARE @x INT = 0

SELECT 1 / @x

END TRY

BEGIN CATCH

SELECT

ERROR\_NUMBER() AS 'Код ошибки',

ERROR\_MESSAGE() AS 'Сообщение об ошибке',

ERROR\_LINE() AS 'Номер строки с ошибкой',

ISNULL(ERROR\_PROCEDURE(), 'NULL') AS 'Имя процедуры',

ERROR\_SEVERITY() AS 'Уровень серьезности ошибки',

ERROR\_STATE() AS 'Метка ошибки'

END CATCH

***Результат:***

