**1 СЕМИНАР**

**Условные конструкции**  
**Задание 1. Женщин или мужчин больше в 3-ем отделе?**  
DECLARE @m\_count INT, @w\_count INT  
SELECT @m\_count = COUNT (gender) FROM Sotr WHERE gender = 'm' AND dep=3  
SELECT @w\_count = COUNT (gender) FROM Sotr WHERE gender = 'ж' AND dep=3  
IF @w\_count > @m\_count

PRINT 'Женщин больше'

ELSE IF @m\_count = @m\_count

PRINT 'Количество равно'

ELSE

PRINT 'Мужчин больше'

**Задание 2. Большая или меньшая часть сотрудников получает зарплату выше средней?**  
DECLARE @avg\_salary DECIMAL, @rich INT, @broke INT  
SELECT @avg\_salary = AVG(salary) FROM Sotr  
SELECT @rich  COUNT(\*) FROM Sotr WHERE salary > @avg\_salary  
SELECT @broke = COUNT(\*) FROM Sotr WHERE salary < @avg\_salary  
IF @rich > @broke

PRINT 'Богатых больше'

ELSE IF @rich < @broke

PRINT 'Бедных больше'

ELSE

PRINT 'Одинаковое количество'

**Задание 3. Вывести содержимое таблицы StudGroup (NameGroup, Kurs, Kafedra), заменив аббревиатуру названия кафедры ее полным названием).**  
SELECT NameGroup, Kurs,

Case Kafedra

WHEN 'ПМ' THEN 'Прикладная математика'

WHEN 'ИС' THEN 'Информационные системы'

WHEN 'ИГ' THEN 'Инженерная графика'

ELSE Kafedra

END AS Kafedra FROM Sotr

**Задание 4. Перевести оценку @Mark, представленную по 50-бальной шкале, в словесное описание: неуд, удовлетворительно, хорошо, отлично.**  
DECLARE @mark INT, @text\_mark VARCHAR (20)  
IF @mark < 25

SET @text\_mark = 'неуд'

ELSE IF @mark < 35

SET @text\_mark = 'удовлетворительно'

ELSE IF @mark < 45

SET @text\_mark = 'хорошо'

ELSE

SET @text\_mark = 'отлично'

Case  
  WHEN @mark < 25 THEN 'неуд'  
  WHEN @mark <35 THEN 'удовлетворительно'  
  WHEN @mark <45 THEN 'хорошо'  
     ELSE 'отлично'  
END  
PRINT @text\_mark

**2 СЕМИНАР**

**Задание 5. Для каждого сотрудника определить частоту его участия в проектах: 0 - никогда, 1-2 - очень редко, 3-4 - иногда, более 5 раз - часто. Обратите внимание, что если сотрудник не участвовал в проектах, то его следует добавить к результирующим данным с помощью оператора объединения UNION.**  
//SELECT COUNT(\*) FROM Proj GROUP BY Sotr\_id  
  
SELECT Sotr\_id CASE WHEN COUNT (\*) BETWEEN 1 AND 2 THEN 'Очень редко'  
WHEN COUNT (\*) BETWEEN 3 AND 4 THEN ''Иногда'  
ELSE 'Часто'  
END AS 'Частота'  
FROM Project GROUP BY Sotr\_id  
UNION  
SELECT Sotr\_id 'Никогда' FROM Sotr WHERE Sotr\_id NOT IN (SELECT Sotr\_id FROM Project) / WHERE EXISTS (SELECT (\*) FROM Project WHERE Project.Sotr\_id = Sotr.Sotr\_id)  
  
**Задание 6. В предыдущем задании дополнительно вывести фамилию сотрудника из другой таблицы.**  
SELECT Sotr.Sotr\_id, Sotr.Fio CASE WHEN COUNT (\*) BETWEEN 1 AND 2 THEN 'Очень редко'  
WHEN COUNT (\*) BETWEEN 3 AND 4 THEN ''Иногда'  
ELSE 'Часто'  
END AS 'Частота'  
FROM Sotr JOIN Project ON Sotr.Sotr\_id = Project.Sotr\_id GROUP BY Sotr.Sotr\_id, Sotr.Fio  
UNION  
SELECT Sotr\_id, Sotr.Fio 'Никогда' FROM Sotr WHERE Sotr\_id NOT IN (SELECT Sotr\_id FROM Project) / WHERE NOT EXISTS (SELECT (\*) FROM Project WHERE Project.Sotr\_id = Sotr.Sotr\_id)  
  
**Задание 7. Как в большинстве сдали студенты группы ИДБ-22-04 экзамен по Архитектуре ЭВМ (на отлично, хорошо или удовлетворительно)?**  
DECLARE @otl INT, @hor INT, @udovl INT  
SET @otl = COUNT (\*) FROM Exam, Student WHERE Exam.id\_stident = Student.id\_student AND Mark=5 AND NameGroup = 'ИДБ-22-04' AND Subject = 'Архитектура ЭВМ'  
SELECT @hor = COUNT (\*) FROM Exam, Student WHERE Exam.id\_stident = Student.id\_student AND Mark=4 AND NameGroup = 'ИДБ-22-04' AND Subject = 'Архитектура ЭВМ'  
SELECT @udovl = COUNT (\*) FROM Exam, Student WHERE Exam.id\_stident = Student.id\_student AND Mark = 3 AND NameGroup = 'ИДБ-22-04' AND Subject = 'Архитектура ЭВМ'  
IF @otl > @hor AND @otl > @udovl

PRINT 'Отлично'

ELSE IF @hor > @otl AND @hor > @udovl

PRINT 'Хорошо'

ELSE IF @udovl > @otl AND @udovl > @hor

PRINT 'Удовлетворительно'

ELSE @otl = @hor AND @otl = @udovl AND @hor = @udovl

PRINT 'Одинаково'

**Задание 9. На Новый год решили премировать всех сотрудников и попросили вычислить сумму бонусов по следующей схеме: -сотрудникам 1-го отдела выдать по 15% от ЗП; -сотрудникам 3-го отдела - по 10% от ЗП; -всем остальным - по 5% от ЗП.**

**Выведите таблицу премиальных с указанием ФИО сотрудника, его номера, номера отдела, процент премиальных, значение премиальных (в рублях).**  
SELECT Sotr.Sotr\_id, Sotr.Fio, Sotr.Dep  
CASE

WHEN Sotr.Dep = 1 THEN '15%'

WHEN Sotr.Dep = 3 THEN '10%'

ELSE '5%',

END AS [Процент премиальных],

CASE

WHEN Sotr.Dep = 1 THEN  Sotr.Salary \* 0,15

WHEN Sotr.Dep = 3 THEN Sotr.Salary \* 0,1

ELSE Sotr.Salary \* 0,05

END AS [Сумма премиальных],

FROM Sotr

**3 СЕМИНАР. КУРСОРЫ**

**Задание 1. Используя курсоры, получить из таблицы Client(Id, Name, Fam, …) упорядоченный список клиентов, объединив фамилию и имя в одну строку.**

DECLARE @name VARCHAR(20), @fam VARCHAR(20)  
DECLARE client\_cursos CURSOR FOR  
SELECT name, fam FROM Client ORDER BY fam, name  
OPEN client\_cursor  
FETCH NEXT FROM client\_cursor INTO @name, @fam  
WHILE @@FETCH\_STATUS = 0  
BEGIN  
    PRINT @fam + ' ' + @name  
    FETCH NEXT FROM client\_cursor INTO @name, @fam  
END  
CLOSE client\_cursor  
DEALLOCATE client\_cursor  
  
**Задание 1 (с нумерацией)**  
DECLARE @name VARCHAR(20), @fam VARCHAR(20), @it INT  
DECLARE client\_cursos CURSOR FOR  
SELECT name, fam FROM Client ORDER BY fam, name  
SET @it = 1  
OPEN client\_cursor  
FETCH NEXT FROM client\_cursor INTO @name, @fam  
WHILE @@FETCH\_STATUS = 0  
BEGIN  
    PRINT CAST (@it AS VARCHAR(20)) + ' ' +  @fam + ' ' + @name  
    FETCH NEXT FROM client\_cursor INTO @name, @fam  
    SET @it = @it + 1  
END  
CLOSE client\_cursor  
DEALLOCATE client\_cursor  
  
**Задание 2. Используя таблицу Exam(Id\_student, Subject, Date, Mark), определить для каждого студента результаты сдачи сессии: на отлично, с четверками, с тройками.**  
DECLARE studentMark CURSOR FOR  
SELECT Id\_Student, min(Mark) FROM Exam GROUP BY Id\_Student, Mark  
DECLARE @stud\_id INT, @stud\_mark INT  
DECLARE @tmp TABLE (stud\_num INT, mark VARCHAR(20))  
OPEN studentMark  
FETCH NEXT FROM studentMark INTO @stud\_id, @stud\_mark  
WHILE @@FETCH\_STATUS = 0  
BEGIN  
    IF @stud\_mark INSERT INTO @tmp VALUES (@stud\_id, 'с тройками')  
    IF @stud\_mark = 4 INSERT INTO @tmp VALUES (@stud\_id, 'с четверками')  
    IF @stud\_mark = 5 INSERT INTO @tmp VALUES (@stud\_id, 'с пятерками')  
    FETCH NEXT FROM studentMark INTO @stud\_id, @stud\_mark  
END  
SELECT \* FROM @tmp  
CLOSE studentMark  
DEALLOCATE studentMark  
  
**Задание 3. Рассчитать и изменить заработную плату сотрудников фирмы по следующей формуле: Зарплата = Оклад + Доплата за стаж + Премия (если она есть). Данные хранятся в таблице TCalc(Id\_sotr, Oklad, Stage, Bonus, …, Salary). Доплата за стаж вычисляется как:**

**Стаж от 10 до 15 лет − доплата 10% от оклада**

**Стаж от 15 до 20 лет − доплата 20% от оклада**

**Стаж свыше 20 лет − доплата 25% от оклада**  
DECLARE @newS DECIMAL(8,2)  
DECLARE new\_cursor CURSOR LOCAL FORWARD\_ONLY DYNAMIC OPTIMISTIC FOR  
SELECT Oklad, Stage, Bonus FROM TCalc  
OPEN new\_cursor  
DECLARE @ok INT, @st INT, @bon INT  
FETCH NEXT FROM new\_cursor INTO @ok, @st, @bon  
WHILE @@FETCH\_STATUS = 0  
BEGIN  
    SET @newS = CASE  
    WHEN @st BETWEEN 10 AND 15 THEN 1,1 \* @ok  
    WHEN @st BETWEEN 15 AND 20 THEN 1,2 \* @ok  
    WHEN @st BETWEEN 20 AND 25 THEN 1,25 \* @ok  
    ELSE @ok  
END  
IF @bon IS NOT NULL SET @newS = @newS + @bon  
UPDATE TCalc SET salary = @newS WHERE CURRENT OFF new\_cursor  
FETCH NEXT FROM new\_cursor INTO @ok, @st, @bon  
END  
CLOSE new\_cursor  
DEALLOCATE new\_cursor

**3 СЕМИНАР. ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ, ХРАНИМЫЕ ПРОЦЕДУРЫ**

**Задание 1**  
**Компания хочет выплатить всем сотрудникам первого отдела премию в размере 20% от зарплаты. Определить общую сумму премиальных выплат. Оформить запрос в виде хранимой процедуры с выходным параметром.**  
  
CREATE PROCEDURE Bonus  
@TOTAL\_BONUS DECIMAL (8,2) OUTPUT  
AS  
SELECT @TOTAL\_BONUS = SUM(Salary\*0,2) FROM Sotr WHERE Dep=1  
DECLARE @Payment DECIMAL (8,2)  
EXEC Bonus @Payment OUTPUT  
PRINT @Payment  
  
**Задание 2**  
**Переписать предыдущую процедуру, указав номер отдела и процент премиальных, как входные параметры процедуры.**  
  
CREATE PROCEDURE Bonus @Dep INT, @Proc INT,  
@TOTAL\_BONUS DECIMAL (8,2) OUTPUT  
AS  
SELECT @TOTAL\_BONUS = SUM(Salary\*@Proc/100) FROM Sotr WHERE Dep=1  
DECLARE @P DECIMAL (8,2)  
EXEC Bonus 1, 20, @P OUTPUT  
PRINT @P  
  
**Задание 3**  
**На основе предыдущей процедуры, создать новую, в которой отдельно определен процент премиальных для сотрудников и процент премиальных для руководства (должность – начальник).**  
  
CREATE PROCEDURE Bonus @Dep INT, @P1 INT, @P2 INT,  
@Total DECIMAL (8,2) OUTPUT  
AS  
BEGIN  
SELECT @Total = SUM(SALARY \* @P1 / 100) FROM Sotr WHERE Dep=@Dep AND job LIKE '%начальник%'

SELECT @Total = @Total (+ SUM(Salary\*@P2 / 100) FROM Sotr WHERE Dep=@Dep AND job NOT LIKE '%начальник%'

DECLARE @result DECIMAL (8,2)

EXEC Bonus 1, 35, 5, @res OUTPUT

PRINT @result

**Задание 4**

**Вывести упорядоченный по фамилии список сотрудников (номера и фамилии), которые могут выполнить указанное задание. Реализовать с помощью функции, возвращающей табличное значение.**

CREATE FUNCTION GetEmpTasks (@task VARCHAR(50))

RETURNS TABLE

AS BEGIN

RETURN

SELECT Sotr\_id, FIO FROM Sotr WHERE (SELECT Task FROM Project WHERE Project.Sotr\_id = Sotr.Sotr\_id) ORDER BY FIO

END

SELECT \* FROM GetEmpTask ('отчет')

ORDER BY FIO

**Задание 5**

**Написать функцию, возвращающую номер отдела, в котором работает наименьшее количество сотрудников.**

CREATE FUNCTION Find()  
RETURNS INT  
AS  
BEGIN  
DECLARE @tmp TABLE (numb INT, sotrCount INT)  
INSERT INTO @tmp SELECT Dep, COUNT(\*) FROM Sotr GROUP BY Dep  
DECLARE @min INT  
SET @min = (SELECT MIN(SotrCount) FROM @tmp)  
DECLARE @res INT  
SET @res = (SELECT numb FROM @tmp WHERE SotrCount = @min)  
RETURN @res  
END  
PRINT Find()  
  
**Задание 6**  
**Выполнить перевод сотрудника Филина С.С. в отдел номер 3 с установлением заработной платы в размере 45 000 рублей, предварительно проверив его наличие в базе данных. С помощью процедуры или функции можно это сделать? Почему? В основной программе вызвать данную процедуру (или функцию) и прокомментировать результаты.**  
CERATE PROCEDURE transfer  
@fio VARCHAR(50), @salary DECIMAL (8,2), @dep INT, @b VARCHAR (6) OUTPUT  
AS  
BEGIN  
SET @n = 'успех'  
IF EXISTS (SELECT \* FROM Sotr WHERE FIO = @fio)  
UPDATE Sotr  
SET Salary = @salary, Dep = @dep WHERE FIO = @fio  
ELSE  
SET @b = 'провал'  
END  
DECLARE @status VARCHAR(6)  
EXEC transfer 'Филин С.С.', 45000, 3, @status OUTPUT  
PRINT @status

**5 СЕМИНАР. ТРИГГЕРЫ**

**1 задание**  
**Разработать триггер, который при добавлении к таблице Sotr нового сотрудника автоматически сохраняет частичные данные о нем в таблице цчета приема-увольнения сотрудников Journal (id записи (счетчик), Sotr\_id, FIO, Birthday, DateBeg (дата приема), DateEnd (увольнения)).**  
CREATE TRIGGER Tr\_1  
ON Sotr AFTER INSERT  
AS  
BEGIN

INSERT INTO Journal (Sotr\_id, FIO, Birthday, DateBeg)

SELECT Sotr\_id, FIO, Birthday, GETDATE() FROM Inserted

END  
  
**2 задание**  
**Разработать триггер, который при удалении сотрудника из таблицы Sotr автоматически записывает в таблицу Journal дату увольнения данного сотрудника.**CREATE TRIGGER Tr\_2  
ON Sotr AFTER DELETE  
AS  
BEGIN

UPDATE Journal

SET DataEnd = GETDATE() WHERE Sotr\_id IN (SELECT Sotr\_id FROM Deleted)

END  
  
**3 задание**  
**Разработать триггер, который при добавлении к таблице Sotr нового сотрудника автоматически вычисляет общее число сотрудников в том отделе, в который был зачислен этот сотрудник. Данное значение фиксируется в таблице DepSotr (номер отдела NDep, число сотрудников Size), если строка с таким отделом существует, или создается новая запись с соответствующими значениями.**CREATE TRIGGER Tr\_3  
ON Sotr AFTER INSERT  
AS  
BEGIN

DECLARE @NDep INT

SELECT @NDep = Dep FROM Inserted

IF EXISTS (SELECT \* FROM DepSotr WHERE Dep = @Ndep)

UPDATE DepSotr

SET Size = Size + 1

WHERE Dep = @NDep

ELSE INSERT INTO DepSotr VALUES (@NDep, SELECT COUNT (\*) FROM DepSotr WHERE Dep = @NDep)

END

**4 задание**  
**Разработать триггер, который при удалении из таблицы Sotr сотрудника автоматически уменьшает общее число сотрудников в отделе, которое фиксируется в таблице DepSotr. Если отдел становится пустым, то соответствующая отделу строка удаляется из таблицы DepSotr.**  
CREATE TRIGGER Tr\_4  
ON Sotr AFTER DELETE  
AS  
BEGIN

UPDATE DepSotr

SET Size = Size - 1

WHERE NDep IN (SELECT Dep FROM Deleted)

IF EXISTS (SELECT \* FROM DepSotr WHERE Size = 0)

DELETE FROM DepSotr WHERE Size = 0

END  
  
**5 задание**  
**Создать триггер, который запрещает обновление поля Salary, если его новое значение превышает старое более чем на 10 000, и проверить, что обновляется только одна строка.**CREATE TRIGGER  Tr\_5  
ON Sotr AFTER UPDATE  
AS BEGIN

IF (SELECT (\*) FROM Inserted) > 1

BEGIN

PRINT ('Нужен только один сотрудник')

RILLBACK TRANSACTION

RETURN

END

DECLARE @N INT, @O INT

SELECT @N = Salary FROM Inserted

SELECT @O = Salary FROM Deleted

IF (@N > @O + 10000)

BEGIN

PRINT ('Слишком большая зарплата')

ROLLBACK TRANSACTION

END

END  
  
**6 задание**  
**В базе данных есть 2 таблицы: Мероприятие (Код, Название, Начало, Конец) и Билет (Номер, Код\_мероприятия, Дата\_посещения). Создать триггер, который при добавлении или изменении данных в таблице Билет проверяет правильность введенного номера мероприятия, а также то, что значение столбца Дата\_посещения не выходит за пределы проведения данного мероприятия.**  
CREATE TRIGGER Tr\_6  
ON Билет AFTER UPDATE, INSERT  
AS BEGIN

DECLARE @id INT

IF ((SELECT Код\_мероприятия FROM Inserted) NOT IN (SELECT Код FROM Меропритие))

BEGIN

ROLLBACK TRANSACTION

RETURN

END

SET @id = (SELECT Код\_мероприятия FROM Inserted)

IF ((SELECT Дата\_посещения FROM Inserted) BETWEEN (SELECT Начало FROM Мероприятие WHERE Код\_мероприятия = @id) AND (SELECT Конец FROM Мероприятие WHERE Код = @id)

**ЛР1. ПРЕДСТАВЛЕНИЯ**

**1. С помощью представления вывести результаты экзаменов по указанной дисциплине (фамилии студентов, номера групп, оценки).**

CREATE VIEW Exam\_result (ФИО, [Номер группы], Оценка) AS

SELECT TOP (100) PERCENT dbo.Student.FIO, dbo.Student.NameGroup, dbo.Exam.Mark

FROM dbo.Exam INNER JOIN dbo.Student ON dbo.Exam.Id\_Student = dbo.Student.Id\_Student

WHERE (dbo.Exam.Subject = 'ЭВМ')

ORDER BY NameGroup

**2. Используя созданное представление, определить количество пятерок в каждой группе.**

CREATE VIEW Exam\_count5 AS

SELECT TOP (100) PERCENT [Номер группы], COUNT(\*) AS [Кол-во пятерок]

FROM dbo.Exam\_result

WHERE (Оценка > 44)

GROUP BY [Номер группы]

**ЛР2. КУРСОРЫ**

**1. Используя курсор, определите студентов указанной группы, имеющих высокий средний балл по результатам экзаменов.**

**Порядок выполнения:**

**1) Для этого задания создайте курсор с данными о студентах и их**

**средних баллах.**

**2) Введите n − нужное количество студентов с высоким показателем.**

**3) Создайте в курсоре цикл для выборки фамилий и результатов n**

**студентов с средним баллом.**

**4) Сохраните их во временной таблице TOP\_STUDENTS.**

**5) Проверьте случай n = 0. Выдайте соответствующее сообщение.**

**6) После каждого теста просмотрите данные из таблицы TOP\_**

**STUDENTS.**

USE Must; --создание курсора с данными о студентах из определенной группы и их

средним баллом

DECLARE student\_cursor CURSOR LOCAL FORWARD\_ONLY STATIC

FOR SELECT FIO, AVG(Exam.Mark) AS [Средний балл]

FROM Student

JOIN Exam ON Student.Id\_Student = Exam.Id\_Student

WHERE Student.NameGroup = 'ИДБ-22-06'

GROUP BY FIO

ORDER BY [Средний балл] DESC

OPEN student\_cursor

--создание временной таблицы top\_students с данными о ФИО и средним баллом

DECLARE @top\_students TABLE (ФИО VARCHAR(100), [Средний

балл] INT);

DECLARE @avg\_mark INT, @fio VARCHAR(30), @n INT;

SET @n =4; --количество студентов

--проверка, если количество студентов равно 0

IF @n = 0

BEGIN

PRINT 'Количество студентов не может быть равно 0.';

CLOSE student\_cursor;

DEALLOCATE student\_cursor;

RETURN;

END --начало выборки данных из курсора

FETCH NEXT FROM student\_cursor INTO @fio, @avg\_mark

DECLARE @i INT;

SET @i = 0; --счетчик для отслеживания количества добавленных

студентов

WHILE @@FETCH\_STATUS = 0 AND @i < @n –итерации, чтобы в

таблицу внеслись данные только о заявленном количестве студентов

BEGIN

--вставка данных во временную таблицу

INSERT INTO @top\_students (ФИО, [Средний балл])

VALUES (@fio, @avg\_mark);

SET @i = @i + 1; --увеличение счетчика

FETCH NEXT FROM student\_cursor INTO @fio, @avg\_mark; -

получение следующего студента

END

CLOSE student\_cursor; --закрытие курсора

DEALLOCATE student\_cursor;

SELECT \* FROM @top\_students; --вывод данных из временной таблицы

**ЛР3. ХРАНИМЫЕ ПРОЦЕДУРЫ И ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ**

**1. С помощью хранимой процедуры выполнить перевод лектора Воронова В.В. на кафедру ИС, предварительно проверив его наличие в базе данных и отсутствие на новой кафедре полных однофамильцев. В основной программе вызвать данную процедуру и прокомментировать результаты (выдать сообщение об успешности выполнения перевода или о возникшей ошибке).**

CREATE PROCEDURE TransferL

@fio VARCHAR(30), @kafedra VARCHAR(10), @flag VARCHAR(50)

OUTPUT //входные параметры: ФИО лектора, кафедра и флаг, содержащий

сообщение о результате операции

AS

BEGIN

DECLARE @Lect\_id INT //переменная, для хранения Id лектора

//запрос для поиска лектора по его ФИО

SELECT @Lect\_id = Id\_Lect

FROM Lecturer

WHERE FIO = @fio;

//проверка, если лектор не найден выводится сообщение

IF @Lect\_id IS NULL

BEGIN

SET @flag = 'Лектор с заданным фио не найден';

RETURN;

END

//проверка, есть ли на новой кафедре однофамилец

IF EXISTS (

SELECT \*

FROM Lecturer

WHERE FIO = @fio AND Kafedra = @kafedra

)

BEGIN

SET @flag = 'На новой кафедре есть однофамилец'; //вывод сообщения о том, что на кафедре есть однофамилец

RETURN;

END

//обновление кафедры лектора на новую

UPDATE Lecturer

SET Kafedra = @kafedra

WHERE Id\_Lect = @lect\_id;

SET @flag = 'Лектор успешно переведен на кафедру'; //вывод

сообщения об успешном переводе лектора

END;

//вызов процедуры

DECLARE @flag VARCHAR(50) //объявление переменной, для хранения результата выполнения процедуры

SELECT \* FROM Lecturer //вывод записей до выполнения процедуры

EXEC TransferL 'Быстрикова В.А.', 'ПМ', @flag OUTPUT; //выполнение процедуры с заданными параметрами: ФИО, кафедра и сообщение о результате выполнения

PRINT @flag //вывод сообщения из переменной после выполнения процедуры

SELECT \* FROM Lecturer // вывод записей после выполнения процедуры

**2. С помощью функции определить, кто среди студентов 3-го курса сдал указанный экзамен на отлично. В основной программе вывести их фамилии и группы для дисциплины Математика.**

CREATE FUNCTION ReturnMark (@Kurs INT, @Subj VARCHAR(10))

RETURNS @stud TABLE (Fio VARCHAR(100), N\_group

VARCHAR(20)) //создание таблицы с параметрами: ФИО и номер группы

AS

BEGIN

INSERT INTO @stud (Fio, N\_group) //занесение данных в табличную переменную @stud

SELECT Student.FIO, Student.NameGroup

FROM Student

JOIN Exam ON Student.Id\_Student = Exam.Id\_Student // объединение с таблицей Exam по полю Id\_Student

JOIN StudGroup ON Student.NameGroup = StudGroup.NameGroup // Объединение с таблицей StudGroup по полю NameGroup

WHERE StudGroup.Kurs = @Kurs

AND Exam.Subject = @Subj

AND Exam.Mark BETWEEN 45 AND 54; //выбираются только

отличники

RETURN;

END;

//вызов функции

SELECT \* FROM ReturnMark (3, 'Математика')

**3. Создать функцию, возвращающую общее количество студентов отличников 3-го курса, используя при этом функцию из п.2.5 (в). В основной программе вызвать данную функцию для дисциплины Управление данными.**

// Создание функции CountStud, которая принимает предмет (@Subj) и

возвращает количество студентов

CREATE FUNCTION CountStud (@Subj VARCHAR(50))

RETURNS INT

AS

BEGIN

DECLARE @Count INT; // Объявление переменной @Count для

хранения количества студентов

SELECT @Count = COUNT(\*)

FROM ReturnMark(3, @Subj); //используем функцию для подсчета кол-ва студентов 3-го курса и сдали экзамен по предмету @subj с оценкой 45-54

RETURN @Count; // Возвращаем результат - количество студентов

END;

//вызов функции

DECLARE @Cnt INT;// Объявление переменной @Cnt для хранения количества студентов

SELECT @Cnt = dbo.CountStud('УД'); // Вызов функции CountStud с параметром 'УД' и результат записываем в переменную @Cnt

PRINT 'Кол-во студентов отличников = ' + CAST(@Cnt AS VARCHAR) // Вывод сообщения с количеством студентов

**ЛР5. ТРИГГЕРЫ**

**1. Создать триггер, который при добавлении или изменении данных втаблице «Student», проверяет правильность введенной группы (ее наличиев таблице StudGroup) и отсутствие в группе полных однофамильцев.**

**Предварительно проверить, что вставляется или обновляется только одна запись и удалите связь между таблицами Student и StudGroup (внешний ключ).**

CREATE TRIGGER CheckNameAndGroup

ON Student

INSTEAD OF INSERT, UPDATE

AS

BEGIN

IF (SELECT COUNT(\*) FROM inserted) > 1

BEGIN

PRINT 'Нельзя добавить или обновить больше одной записи'

RETURN

ROLLBACK TRAN

END

DECLARE @group VARCHAR(20), @fio VARCHAR(50), @cnt INT;

SELECT @group = NameGroup, @fio = FIO FROM inserted;

IF NOT EXISTS (SELECT \* FROM StudGroup WHERE NameGroup = @group)

BEGIN

PRINT 'Данной группы не существует'

RETURN

ROLLBACK TRAN

END

SET @cnt = (SELECT COUNT(\*) FROM Student WHERE NameGroup = @group AND FIO = @fio);

IF @cnt > 1

BEGIN

PRINT 'В данной группе есть полный однофамилец'

RETURN

ROLLBACK TRAN

END

END