For example, consider a table Faculties and Tutors (here are operators for creation and data adding to represent the structure of tables):

CREATE TABLE faculties

(

title CHAR(10) NOT NULL UNIQUE,

deanSurname VARCHAR(20) NOT NULL,

phoneNumber VARCHAR(8) NOT NULL,

baseDate DATETIME,

chairNumber INTEGER,

studentNumber INTEGER,

PRIMARY KEY (title)

);

INSERT INTO faculties

VALUES ('FIT','Gadjiyev', '63-45-56', '7/6/2001', 3, 500, 1, 0.56)

INSERT INTO faculties

VALUES ('FEOGI','Smith', '63-99-00', '8/21/2001', 4, 2000, 0, 0.23)

INSERT INTO faculties

VALUES ('FEF','MisterX', '63-12-50', '9/1/2001', 6, 1500, 1, 0.3)

INSERT INTO faculties

VALUES ('ISE','Brien', '63-34-75', '9/30/2003', 1, 300, 1, 0.1)

CREATE TABLE tutor

(

id INTEGER UNIQUE NOT NULL,

name VARCHAR(20),

title VARCHAR(20),

PRIMARY KEY (id)

)

INSERT INTO tutor

VALUES (112, 'Lutz', 'FIT')

INSERT INTO tutor

VALUES (145, 'Kurmanov', 'FIT')

INSERT INTO tutor

VALUES (566, 'Kamalidenova', 'FEF')

**Task #1.1**

Create custom stored procedures to create dynamic queries. The input data must be variable. Apply the possibilities of expression LIKE% \_ etc. Removal, modification st.procedure

**The result**: Create a procedure output of the list of all faculty members whose names begin with the specified user token.

The command to create the procedure:

CREATE PROCEDURE show @faculty VARCHAR(20)

AS

BEGIN

SELECT \* FROM tutor

WHERE title LIKE @faculty+'%'

END

The procedure execution:

EXEC show 'FI'

id name title

----------- -------------------- --------------------

112 Lutz FIT

145 Kurmanov FIT

(2 row(s) affected)

Modify the procedure as follows: list the faculty members whose names contain the substring specified by the user

The command to create the procedure:

ALTER PROCEDURE show @faculty VARCHAR(20)

AS

BEGIN

SELECT \* FROM tutor

WHERE title LIKE '%'+@faculty+'%'

END

Procedure verification:

EXEC show 'E'

id name title

----------- -------------------- --------------------

566 Kamalidenova FEF

(1 row(s) affected)

Procedure removal command:

DROP PROC show

Checking:

EXEC show 'E'

Msg 2812, Level 16, State 62, Line 1

Could not find stored procedure 'show'.

**Task #1.2**

Create a stored procedure with default values

**The result**: modify the “*show*” procedure, defining the substring default 'FIT'.

Procedure modification command:

ALTER PROCEDURE show @faculty VARCHAR(20) = 'FIT'

AS

BEGIN

SELECT \* FROM tutor

WHERE title LIKE '%'+@faculty+'%'

END

Procedure execution command:

EXEC show

id name title

----------- -------------------- --------------------

112 Lutz FIT

145 Kurmanov FIT

(2 row(s) affected)

**Task #1.3**

Create a stored procedure with a notice on the successful implementation of the process. (Using “*if*” condition).

**The result**: Modify the procedure: Add a warning that there is no search terms and the lack of data that satisfy the query.

Procedure modification command:

ALTER PROCEDURE show @faculty VARCHAR(20) = NULL

AS

BEGIN

IF @faculty IS NULL

BEGIN

PRINT 'There is not enough parameters to procedure show'

END

ELSE

BEGIN

IF EXISTS(SELECT \* FROM tutor WHERE title LIKE '%'+@faculty+'%')

BEGIN

PRINT 'Tutors list'

SELECT \* FROM tutor WHERE title LIKE '%'+@faculty+'%'

END

ELSE

BEGIN

PRINT 'There is no data found for your query'

END

END

END

Procedure verification:

EXEC show

There is not enough parameters to procedure show

EXEC show 'Z'

There is no data found for your query

EXEC show 'I'

Tutors list

id name title

----------- -------------------- --------------------

112 Lutz FIT

145 Kurmanov FIT

(2 row(s) affected)

**Task #1.4**

Create a stored procedure that calls another procedure. The need must be justified.

**The result**: Create a new procedure that displays a list of teachers given faculties, and list of phone numbers of these phones. For this case, you must use the above procedure call in the body of the new procedure

The command to create the procedure:

CREATE PROC outputList @faculty VARCHAR(20) = NULL

AS

BEGIN

EXEC show @faculty

IF EXISTS(SELECT \* FROM faculties)

BEGIN

PRINT 'Phone numbers of faculties'

SELECT title, phonenumber FROM faculties

END

ELSE

BEGIN

PRINT 'No faculties found'

END

END

Procedure verification:

EXEC outputList 'E'

Tutors list

id name title

----------- -------------------- --------------------

566 Kamalidenova FEF

(1 row(s) affected)

Phone numbers of faculties

title phonenumber

---------- -----------

FEF 63-12-50

FEOGI 63-99-00

FIT 63-45-56

ISE 63-34-75

(4 row(s) affected)

**Task #1.5**

Multivalued data output as a result of several stored procedures call at the same time

**The result**: Modify the created above procedure. One of them returns the name of the teacher and the department in which he works, and the other the faculty information as a report.

Procedure modification command:

ALTER PROCEDURE show

@Name VARCHAR(20) = NULL,

@title VARCHAR(20) OUTPUT,

@tutorName VARCHAR(20) OUTPUT

AS

BEGIN

IF @name IS NULL

BEGIN

PRINT 'There is not enough parameters to procedure show'

END

ELSE

BEGIN

IF EXISTS(SELECT \* FROM tutor WHERE name LIKE '%'+@name+'%')

BEGIN

PRINT 'Tutors list'

SELECT @title = title, @tutorName = name

FROM tutor WHERE name LIKE '%'+@name+'%'

END

ELSE

BEGIN

PRINT 'There is no data found for your query'

END

END

END

ALTER PROC outputList @faculty VARCHAR(20) = NULL

AS

BEGIN

DECLARE @title VARCHAR(20)

DECLARE @name VARCHAR(20)

EXEC show @faculty, @title OUTPUT, @name OUTPUT

PRINT 'Tutor '+@name

PRINT 'Faculty Information'

SELECT title, phonenumber FROM faculties

WHERE title = @title

END

Procedure verification:

EXEC outputList 'I'

Tutors list

Tutor Kamalidenova

Faculty Information

title phonenumber

---------- -----------

FEF 63-12-50

(1 row(s) affected)