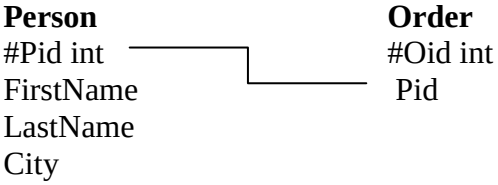
Laboratory 3

Consider 2 tables:



1. Add new column
   ALTER TABLE Person
   ADD Dob date

2. Modify the type of a column
   ALTER TABLE Person
   ALTER COLUMN Dob int NOT NULL

3. Remove a column
   ALTER TABLE Person
   DROP COLUMN Dob

4. Create new table
   CREATE TABLE Person(
   Pidint NOT NULL PRIMARY KEY,
   FirstName varchar(50) NOT NULL,
   LastName varchar(50),
   City varchar(50)
   );

5. Add new column with default constraint
   ALTER TABLE Person
   ADD Dob int DEFAULT 2000;

6. Modify column with default constraint
   ALTER TABLE Person
   ADD DEFAULT 18 FOR Age;

   ALTER TABLE Person
   ADD CONSTRAINT df_18 DEFAULT 18 FOR Age

7. Remove default constraint from a column
   ALTER TABLE Person
   DROP CONSTRAINT df_18;

8. Delete a table
   - Delete all the structure of the table and the records
     DROP TABLE Person

   - Delete only the records (with condition)
     DELETE FROM Person
     [WHERE Dob>2000]

9. Create a foreign key constraint on a new table
   CREATE TABLE Order (
   Oid int NOT NULL PRIMARY KEY,
   Pid int CONSTRAINT fk_Order_Person FOREIGN KEY(Pid) REFERENCES Person(Pid)
   );

10. Create a foreign key as a new add column in a table
    ALTER TABLE Order
    ADD CONSTRAINT fk_Order_Person FOREIGN KEY(Pid) REFERENCES Person(Pid)

11. Remove a foreign key
    ALTER TABLE Order
    DROP CONSTRAINT fk_Order_Person;

5 procedures do	**5 procedures undo (reverse)**

do_proc_1 – modify the type of the column	undo_proc_1 – modify the type of the column (back)
do_proc_2 – add a default constraint	undo_proc_2 – remove a default constraint
do_proc_3 – create a new table	undo_proc_3 – remove a table
do_proc_4 – add a column	undo_proc_4 – remove a column
do_proc_5 – create a foreign key constraint	undo_proc_5 – remove a foreign key constraint

PAY ATTENTION to the name of the procedures – because with their name you work in the main procedure.

Suppose we take the table Version where we keep the version of the database (version 0 – the first one – the one it is now)

main 4 – will take the database from version 0 to version 4 (crossing version 1, 2, 3)
version 1 – will be given by executing do_proc_1
version 2 – will be given by executing do_proc_2
version 3 – will be given by executing do_proc_3

main 2 – will take the database from version 4 (the one you have) to version 2 (crossing version 3)
version 3 – will be given by executing undo_proc_3

Please do not use these names for your stored procedures.

STORED PROCEDURES can be found in the Database (your database) -> Programmability -> Stored Procedures -> (right click) Stored Procedures. Please use the template that will appear.

Examples of a stored procedure name without parameter:

CREATE PROCEDURE do_proc_1 AS BEGIN -- the code SELECT * FROM Produs END /* EXECUTE (to create the stored procedure and find it in the list of stored procedures */	Run the procedure (in a new query): EXECUTE do_proc_1 / EXEC do_proc_1 / do_proc_1
CREATE PROCEDURE undo_proc_1 …	

Examples of a stored procedure with parameters:

CREATE PROCEDURE main @vers int, @t varchar(50) AS BEGIN IF @vers>5 BEGIN SELECT * FROM Produs END	Run the procedure (in a new query): EXEC main 6, 'Alba' / EXEC main 1, 'Cluj' / EXEC main 7, 'Cluj'

```
  IF @t='Cluj'
  BEGIN
    PRINT ' DONE'
  END
END
```

Each stored procedure will have a different name and after EXECUTE it will appear in the list of the stored procedures (at Refresh). This means that the procedure was created and can be use (in main procedure or whenever you want).

To run the procedure: open a New Query and write EXECUTE procedure_name [parameters].
EXECUTE main 3 / EXEC main 4 / EXEC main @vers=3

Instructions:

1. WHILE condition
   BEGIN
   ….
   END
2. IF condition
   BEGIN
   …
   END
   [ELSE
   BEGIN
   …
   END]
3. PRINT 'Your message.';
4. DECLARE @a INT
   DECLARE @text VARCHAR(50) --each variable must be declared
5. SET @a=@a +1 -- to modify the value of a variable
   SET @text ='do _proc_' + CONVERT(VARCHAR(5), @a) -- a possible way to concatenate a text with a number – here we can have do_proc_1, do_proc_2, do_proc_3, … depending on the value of the variable @a
6. EXEC @text -- to execute a stored procedure in another stored procedure with the help of a parameter
7. SELECT TOP 1 @a=Vid FROM Version -- save in a variable a value from a table, where Vid is a field of INT type (as @a INT) from the table Version

You must have:

- A table that keeps the current version of your database (you have to modify the version in the table after every run of the stored procedure) (the modification of the version can be made, for example, with an update (UPDATE Version SET Vid=@a))
- 10 stored procedures for each of the operations indicated in the laboratory text (5 for the direct operations and 5 for the reverse operations). (If you create a table, you must delete the same table that you have created before.) Use only specific fields, tables, .. from your database (not generic, as a generalization). Give suggestive messages ('The column … has been added to table …') in the stored procedures. Please use the order of the operations that is given in the text of the problem.
- A main stored procedure in which all the 10 stored procedures are used. Also verify the particular cases. This main must have a parameter that has to be checked (not null, not an incorrect number, not a text, ...)

PLEASE USE THE LABORATORY 2 DOCUMENTATION.