Tutorial 5 SQL DDL

Keep a record of all the SQL produced during tutorial in a separate text file.

# Creation of tables

1. Create the DEPARTMENT table based on the following description. Confirm that the table is created (e.g. try to: select \* from department; or check if table appears in SQL Management Studio in Tables list).

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **Is nullable** |
| Id | integer | No |
| Title | String, max length 100 | Yes |
| Location | String, max length 50 | Yes |

1. Create table departmet2 that will have Id and Title NOT NULL.
2. Create table department3 that will have Id and Title NOT NULL also allow only unique values in Tile column (add UNIQUE constraint to Title column).
3. Create table department4 and make combination of Title and Location primary key.
4. Rename table department4 to department5. [Hint: use sp\_rename see example in appendix].
5. Rename column Location to Loc in table department5. [Hint: use sp\_rename see example in appendix].
6. Delete tables department2, department3 and department5.
7. Create the EMPLOYEE table based on the following description.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Data type** | **Is nullable** | **Constraint** |
| Id | integer | No | PK |
| FirstName | String, max length 25 | No |  |
| LastName | String, max length 25 | No |  |
| DOB | Date | Yes |  |
| Address | String, max length 200 | Yes |  |
| Email | String, max length 100 | Yes | UNIQUE |
| Phone | String max length 20 | Yes |  |
| Salary | Floating point number, precision 10, scale 2 | Yes |  |
| DateEmployed | Date and time | No | Default value -current date |
| DepartmentId | integer | Yes |  |

For all constraints provide names that should follow the template:

***<2 letters constraint name abbreviation>\_<table name>\_<column name(s)>***

e.g.: pk\_employee\_id, uq\_employee\_firstname\_lastname

*(see more examples in appendix)*

1. First create table EMPLOYEE using column level constraints syntax (see example in appendix)
2. Delete table employee
3. Next create table EMPLOYEE but now use table level constraints syntax (see example in appendix)
4. Change table DEPARTMENT so that it will have Id as a primary key, Title should allow only unique values.
5. Modify the EMPLOYEE table to allow for longer employee last and first names (60 characters).
6. Modify EMPLOYEE table and disallow NULL values in DOB column.
7. Drop the LASTNAME column from the EMPLOYEE table and add it back.
8. Drop the primary key and unique constraint for Email from EMPLOYEE table and add them back.
9. Add a foreign key constraint on the EMPLOYEE table that will ensure that the employee is not assigned to a nonexistent department.
10. Make DepartmentId column in EMPLOYEE table NOT NULL.
11. Add CHECK constraint to EMPLOYEE table to ensure that salary of employee is greater than 0 and does not exceed 5000.
12. Make default value for salary equal to 1000.
13. Modify EMPLOYEE table and make combination of FirstName and LastName unique.
14. Make sure that email will always have “@” character in it.

# Insertion and update of data in tables

1. Insert 5 employees and 3 departments.
2. Update salaries of all employees and increase them by 5%.
3. Update all emails and add trailing and leading spaces to them.
4. Update all emails and make sure that all leading and trailing spaces are removed. [Hint: use *ltrim()* and *rtrim()*]. Check the results.
5. Delete one of the departments that contains employees. Note errors that you will receive. To resolve errors try these solutions:
   1. drop foreign key constraint on EMPLOYEE table and create it back but now specify ON DELETE SET NULL (see examples in appendix)
   2. drop foreign key constraint on EMPLOYEE table and create it back but now specify ON DELETE CASCADE (see examples in appendix)
   3. update employee table and set DepartmentId = NULL for those employees that work in department that you are going to delete and delete department.
   4. Delete all employees in department that you are going to delete and delete department.
6. Update Id of one of the departments that have employees and set it to new value. To resolve error try these solutions:
   1. drop foreign key constraint on EMPLOYEE table and create it back but now specify ON UPDATE SET NULL (see examples in appendix)
   2. drop foreign key constraint on EMPLOYEE table and create it back but now specify ON UPDATE CASCADE (see examples in appendix)
   3. Produce corresponding update statements to move all employees in department with updated Id to new department Id. [Hint: first set departmentId of all employees in that department to Null, then update department Id and after that update departmentId of those employees to new value]
7. Create a table EMP\_COPY with the same structure as table EMP populated with all data from table EMP. [Hint: use SELECT .. INTO syntax, see example in appendix]
8. Create table EMP\_COPY2 with the same structure as table EMP but EMP\_COPY2 should be empty.
9. Copy data for all clerks from EMP table to EMP\_COPY2 table.

# Views

1. Create view that will display EMPNO, ENAME, HIREDATE. Try to update employee data through this view. Notice and explain results.
2. Modify created above view and add DEPTNO column. Try to update employee through view again. Notice and explain results.
3. Create view that will display EMPNO, ENAME, DNAME, LOC from EMP table. Try to insert and update employee using this view. Notice and explain results.
4. Create view that will display employee name, EMPNO and total earnings (sal+comm) and this employee supervisor’s name as well as salary grade. Try to insert and update employee using this view. Notice and explain results.
5. Create views based on sql statements that you produced in Tutorial 5 for tasks 11-15. Comment on views updatability.

# Appendix SQL Server examples

## Table creation

--Column level constraints specification

CREATE TABLE Customer (

[Id] int IDENTITY(1,1) NOT NULL constraint pk\_customer\_id PRIMARY KEY,

[FirstName] nvarchar(40) NOT NULL,

[LastName] nvarchar(20) NOT NULL,

[Company] nvarchar(80) NULL,

[Address] nvarchar(70) NULL,

[Phone] nvarchar(24) NULL constraint ck\_customer\_phone CHECK(Phone like '(\_\_\_)-\_\_-\_\_\_ \_\_ \_\_'),

[Email] nvarchar(60) NOT NULL constraint uq\_customer\_email UNIQUE,

[SupportRepId] [int] NULL,

[DateRegistered] datetime NOT NULL constraint df\_customer\_dateregistered default getdate(),

[DOB] datetime constraint ck\_customer\_dob CHECK( datediff(day, DOB, getdate()) > 18)

)

--Table level constraints specification (preferred)

CREATE TABLE Customer (

[Id] int IDENTITY(1,1) NOT NULL,

[FirstName] nvarchar(40) NOT NULL,

[LastName] nvarchar(20) NOT NULL,

[Company] nvarchar(80) NULL,

[Address] nvarchar(70) NULL,

[Phone] nvarchar(24) NULL ,

[Email] nvarchar(60) NOT NULL ,

[SupportRepId] [int] NULL,

[DateRegistered] datetime NOT NULL constraint df\_customer\_dateregistered DEFAULT getdate(),

[DOB] datetime ,

--constraints

constraint pk\_customer\_id PRIMARY KEY(Id),

constraint ck\_customer\_phone CHECK(Phone like '(\_\_\_)-\_\_-\_\_\_ \_\_ \_\_'),

constraint uq\_customer\_email UNIQUE(Email),

constraint ck\_customer\_dob CHECK(datediff(day, DOB, getdate()) > 18)

);

--Foreign keys creation

CREATE TABLE CustomerType (

Id integer primary key,

Name nvarchar(100)

);

--Foreign key constraint: different behaviors on Delete and Update (default behavior: no action i.e. disallow operation)

--1. Standard behaviour (on delete no action on update no action could be omitted from constraint definition)

alter table customer add constraint fk\_customer\_typeid foreign key(typeid) references CustomerType(id) on delete no action on update no action

--2. On delete and update of parent table set corresponding FK value to null

alter table customer add constraint fk\_customer\_typeid foreign key(typeid) references CustomerType(id) on delete set null on update set null

--3. On delete of row in parent table delete all records in child table. On update of parent change FK value to match updated value in parent table

alter table customer add constraint fk\_customer\_typeid foreign key(typeid) references CustomerType(id) on delete cascade on update cascade

## Altering table

alter table tt2 add a int

alter table tt2 alter column a varchar(20)

alter table tt2 alter column a varchar(20) NULL

alter table tt2 alter column a varchar(20) NOT NULL

alter table tt2 add constraint df\_tt2\_a DEFAULT 'test' for a

alter table tt2 drop constraint df\_tt2\_a

alter table tt2 drop column a

sp\_rename 'Customer.FirstName', 'First\_Name', 'COLUMN'; --rename column

sp\_rename 'Customer', 'Customer123'; --rename table

## Copying Rows from Another Table (Note: Table Manager should exists)

INSERT INTO manager(id, name, salary, hiredate)

SELECT empno, ename, sal, hiredate

FROM emp

WHERE job = 'MANAGER';

## Creating table and copying rows

SELECT \* INTO EMP\_COPY FROM EMP;

## Copy only table structure (without constraints, PK, FK etc)

SELECT \* INTO EMP\_COPY\_EMPTY FROM EMP WHERE 1 <> 1

Views

CREATE VIEW EMP\_VIEW AS

SELECT \* FROM EMP

--modify existing view

ALTER VIEW EMP\_VIEW AS

SELECT EMPNO, ENAME, SAL, JOB, DEPTNO

FROM EMP

WHERE JOB LIKE 'CLERK'

WITH CHECK OPTION

/\* WITH CHECK OPTION - make sure that it is not possible to insert/update data that will not satisfy WHERE condition

\*/

--insert rows through view

SET IDENTITY\_INSERT EMP ON –-allow to insert PK values for identity column

INSERT INTO EMP\_VIEW(EMPNO, ENAME, SAL, JOB, DEPTNO) VALUES (1, 'ALEX', 1000.0, 'CLERK', 10)

SET IDENTITY\_INSERT EMP OFF –-disallow to insert PK values for identity column

–-select data from view

SELECT \* FROM EMP\_VIEW

--delete view

DROP VIEW EMP\_VIEW