**Managing Databases and Tables**

create database PayRollManagement\_db

create table Employee (empid int primary key ,empname varchar(max) null,

salary decimal not null,designation varchar(50),dateofbirth date not null,

deptid int)

select \* from Employee

insert into Employee values(101,'JOHN',50000,'Manager','1945-10-09',10)

insert into Employee values(102,'Smith',65000,'Secretary','1981-10-09',10)

insert into Employee values(103,'kevin',750000,'Developer','1945-10-09',11)

insert into Employee values(104,'Sarhlevo',35000,'Hr','1985-01-23',12)

To get the details about sqlserver installed

select @@VERSION as'serverDetails'

# SQL Server - Get The Computer Name On Which The SQL Server Instance Is Running

**What's the "SERVERPROPERTY"?**

"SERVERPROPERTY" is a System-defined Function used to return the SQL Server instance information.

**"SERVERPROPERTY" Syntax**

SERVERPROPERTY ('propertyname')

**MACHINENAME**

Use MachineName property to get the computer name on which the SQL Server instance is running.

**Select SERVERPROPERTY('MachineName') as 'MachineName'.**

Query below lists databases on SQL Server instance.

To list all databases

select [name] as database\_name,

database\_id,

create\_date

from sys.databases

order by name

(or)

EXEC sp\_databases;

🡪Query below lists all tables from all databases on SQL Server instance

declare @sql nvarchar(max);

select @sql =

(select ' UNION ALL

SELECT ' + + quotename(name,'''') + ' as database\_name,

s.name COLLATE DATABASE\_DEFAULT

AS schema\_name,

t.name COLLATE DATABASE\_DEFAULT as table\_name

FROM '+ quotename(name) + '.sys.tables t

JOIN '+ quotename(name) + '.sys.schemas s

on s.schema\_id = t.schema\_id'

from sys.databases

where state=0

order by [name] for xml path(''), type).value('.', 'nvarchar(max)');

set @sql = stuff(@sql, 1, 12, '') + ' order by database\_name,

schema\_name,

table\_name';

execute (@sql);

**Columns**

* **database\_name** - name of the database within schema resides
* **schema\_name** - name of the schema
* **table\_name** - name of the table

**Rows**

* **One row** represents one table in database
* **Scope of rows:** all tables from all schemas and all databases on SQL Server instance
* **Ordered by** database name, schema name, table name

**RENAME A Database**

**EXEC sp\_renamedb 'Test', 'Test2'**

**HOST\_ID() function returns a ten character workstation identification number. This number is an application process ID running in the client computer.**

**HOST\_NAME() function returns the client computer name. The return value of this function is nvarchar(128).**

**Example:**

**Select HOST\_ID()**

**Rename the Table**

sp\_rename 'old\_table\_name', 'new\_table\_name'

**Data Definition Language:** This is a 1st sub Language in SQL which is

used to define the database objects such as table, view etc.

This language contains five commands

1. Create

2. Alter

3. SP\_Rename

4. Truncate

5. Drop

Create database databasename

Create table tablename(col1 datatype constraints identity (seed,increment).colum2 dataype constraints,colm3)

Constraints

Unique

Notnull

Primarykey = unique+NotNull

Foreignkey

Check

create database PointofSales

* how to name the tables?(coding standard)
* Prefix table names with the owner name3

## SQL keywords lowercase

First, choose a case for your SQL keywords. Historically, these tended to be uppercase, but I have seen a gradual movement to lowercase, and I actually prefer this. Having keywords in lowercase is quicker to type, and easier to read.

## Camel, Pascal, Snake, or Kebab?

No, it’s not an exotic lunch menu! And yes, kebab-case is a thing!

Decide on your case for naming tables, columns, variables, etc. Your options are:

**camelCase** (all words following the first word have title case, no separator)

**snake\_case** (all lowercase, \_ separator)

**PascalCase** (all words in title case, no separator)

**kebab-case** (all lowercase, -separator)

**How to name tables?**

* ***Hint****: Use lower letters when naming database objects. For separating words in the database object name, use underscore*

When naming tables, you have two options – to use the singular for the table name or to use a plural. My suggestion would be to always go with names in the singular.

If you’re naming entities that represent real-world facts, you should use nouns. These are tables like **employee**, **customer**, **city**, and **country**. If possible, use a single word that exactly describes what is in the table. On the example of our 4 tables, it’s more than clear what data can be found in these tables.

* ***Hint****: Use singular for table names (user, role), and not plural (users, roles). The plural could lead to some weird table names later (instead of user\_has\_role, you would have users\_have\_roles, etc.)*

If there is a need to use more than 1 word to describe what is in the table – do it so. In our database, one such example would be the **call\_outcome** table. We can’t use only “call”, because we already have the table **call** in the database. On the other hand, using the word outcome wouldn’t clearly describe what is in the table, so using the **call\_outcome** as the table name seems like a good choice.

create table Customers(id int primary key identity,

custname varchar(50) ,pone int not null,age int check(age>19),

dob date)

select \* from Customers

**select \* from Customers**

**INSERT INTO Customers values('Jojn',884816733,32,'2022-08-03')**

**--Multiple rows insert with single insert statment**

**insert into Customers values('Jackey',804816733,32,'2022-08-03'),**

**('raja',884816733,32,'2022-08-09')**

**ALTER:**

This command is used to modify the structure of a table using this

command, we can perform four different operations

Using this command we can increase (or) decrease the size of the data

type & also we can change the data type of a column from old data type to new

data type

We can add a new column to the existing table

We can change the column name from old column name to new

column name

We can remove the column from the existing table

--Query below lists databases on SQL Server instance.

select [name] as database\_name,

database\_id,

create\_date

from sys.databases

order by name

--(or)

EXEC sp\_databases;

select @@VERSION as'serverDetails'

Select SERVERPROPERTY('MachineName') as 'MachineName'

Select HOST\_ID()

select HOST\_NAME()

EXEC sp\_databases

--query to get all the rows

select \* from Employee

--query to get any one employe

select \* from Employee where empid=104

--query to gt details with alias names

select empname, designation as 'job' from Employee

where empid=104

--Handling null valeues

insert into Employee(empid,salary,dateofbirth) values(106,75000,'2020-11-03')

--to get details of Employee whose name is null

select \* from Employee where empname Is NULL

--Update

Update Employee set empname='Rani',designation='ANALYST',

deptid=10 WHERE empid=106

--ALTER

--to add new column

alter table Employee add age int

--to change the column structure

--to change the column structure to make employeeid as identity column

drop table Employee

create table Employee (empid int primary key identity(1,2) ,empname varchar(max) null,

salary decimal not null,designation varchar(50),dateofbirth date not null,

deptid int)

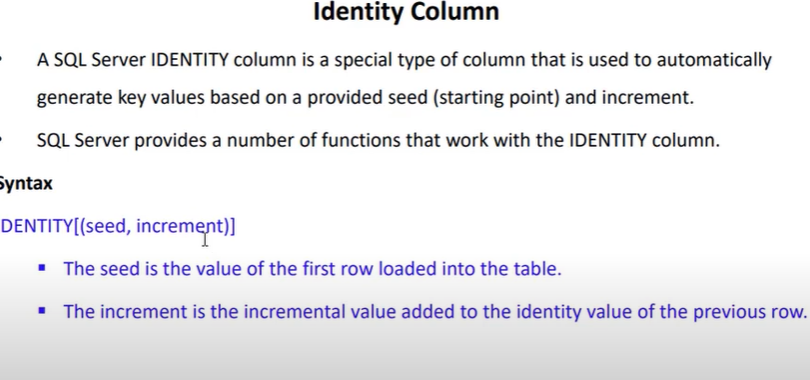
select \* from Employee

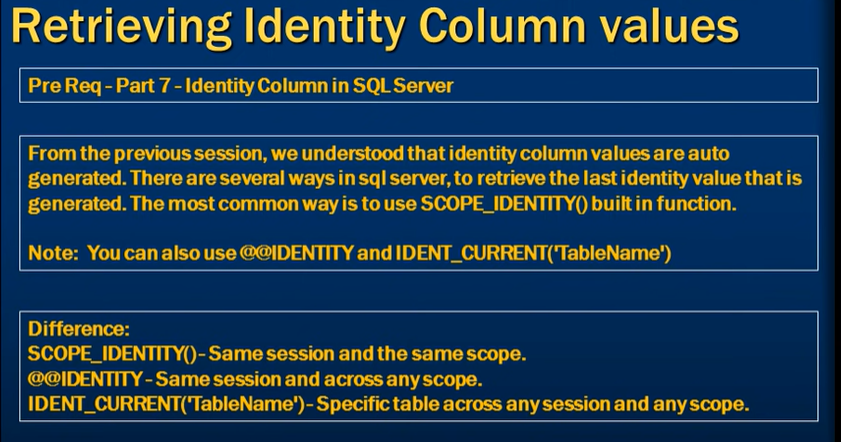
insert into Employee values('JOHN',50000,'Manager','1945-10-09',10)

insert into Employee values('Smith',65000,'Secretary','1981-10-09',10)

insert into Employee values('kevin',750000,'Developer','1945-10-09',11)

insert into Employee values('Sarhlevo',35000,'Hr','1985-01-23',12)



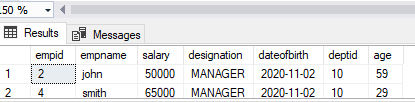


**🡪 Syntax: Identity (seed, increment)**

**if you delete one record what happened to identity?**

delete Employee where empid=3

**insert into Employee Values('smith',65000,'MANAGER','2020-11-2',10,29)**



**Now delte again**

**Syntax:SET Identity\_Insert tablename ON**

delete Employee where empid=3

**SET Identity\_Insert Employee ON**

**insert into Employee(empid,empname,salary,designation,dateofbirth,deptid,age) Values(3,'smith1',6500,'MANAGER','2020-11-2',10,29)**

how to reset identity colum?

how to write custom identity

column

sbi244565768

* cosntraints

constraints are rule

1.check constraint check(condi)

2.unique constraint unique

3.defualt constraint

4.primary key=unique +notnull

5.foregin key

there are two types of constraints

table level constraints

row level cosntrains

* constraints can be created while creating

constraints

or

* after creating the table using alter

statement

When we impose the foreign key constraint and establish relation between the table,the followiong three rules will come into picture.

Rule1:- Cannot insert a value into the foreign key column provided that value is not existing under the refernce key column of the parent table.

**Rule2:-** Cannot update the reference key value of a parent table provided that value has corresponding child record in the child table with out addressing what to do with the child record.

**Rule3:-** Cannot delete a record from the parent table provided that records reference key value has child record in the child table with out addressing what to do with the child record. If we want to delete or update a record in the parent table when they have corresponding child records in the child table we are provide with a set of rules to perform delete and update operations knows as cascade rules.

create table student1(studid int primary key identity(2,1)

check(studid>0),studname varchar(50) Unique not null,ponenumber unique,

age int check(age>18),course varchar(50) DEFAULT 'csharp')

insert into student1 values('john',19,default)

select \* from student1

adding constraint to already existing table

syntax alter table tablename add constraint constraintname constraint(columnname)

--add a column by altering existing table as customername

--add constraing to the newly added column with default

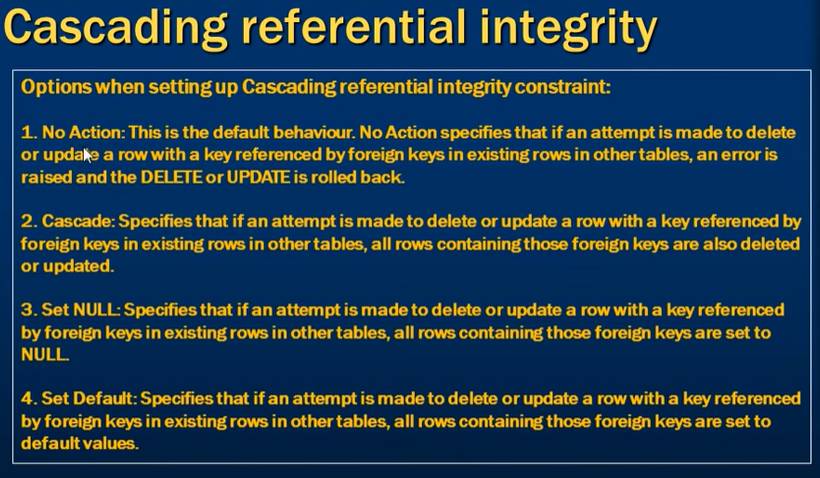
create table orders(orderid int primary key identity(3,1),

orderdate datetime check(orderdate<=getdate()),ordername varchar(50))

solution

alter table orders add constraint ordunique unique(ordername)

# 



# Delete And Update Cascade in SQL Server

[Vaishali Goilkar](https://vaishaligoilkar3322.medium.com/?source=post_page-----f262038550da-----------------------------------)

we learn about delete and update cascade in SQL Server.

* First I will create a two table: EMP table and EMP\_Address table. And I will apply the primary key on both of the tables.

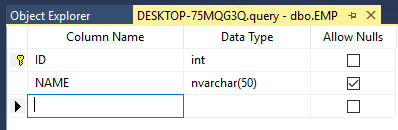
**TABLE 1:- EMP**

CREATE TABLE EMP(

ID INT PRIMARY KEY,

NAME NVarchar(50))

https://miro.medium.com/max/60/1*d5go31SMvMJbr9rI-F9Cgw.png?q=20



EMP TABLE

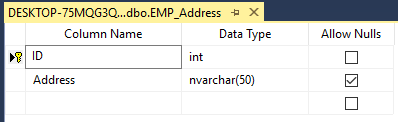
**TABLE 2:- EMP\_Address**

CREATE TABLE EMP\_Address(

ID INT PRIMARY KEY,

Address NVarchar(50))

https://miro.medium.com/max/60/1*21IgmprmkJ_VxE-Ve2dKQQ.png?q=20



EMP\_Address TABLE

* After that, I will create a foreign key relationship with two tables. And Delete and Update cascade apply to that table.
* **Query:**

ALTER TABLE [dbo].[EMP\_Address]

ADD CONSTRAINT FK\_EMP\_Address

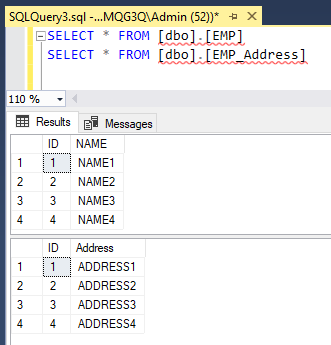
FOREIGN KEY([ID])

REFERENCES [dbo].[EMP]([ID])

ON DELETE CASCADE

ON UPDATE CASCADE

https://miro.medium.com/max/58/1*He5oAaeW_nOfMSjRLYDhHg.png?q=20

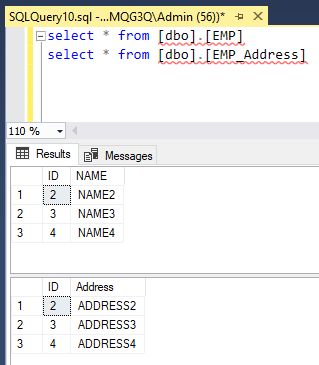


INSERTED RECORD

***DELETE CASCADE:***

* In the delete cascade, If we delete the record from the source table also it will delete the record from another table.
* **Query:**DELETE FROM [dbo].[EMP] WHERE [Id]=1
* Run the query and check the records by using the select query. In both tables, the first record is deleted.

https://miro.medium.com/max/52/1*Ui2kqduOIeNz7DZD2UdjxA.png?q=20

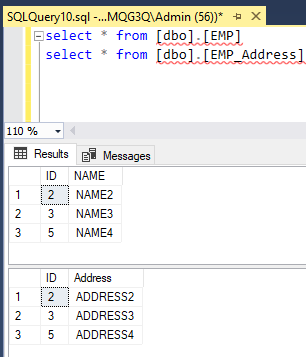


DELETE CASCADE

***UPDATE CASCADE:***

* In the update cascade, If we update the record from the table also it will update the record from another table.
* **Query:**UPDATE [dbo].[EMP] SET Id = 5 where [ID] = 4
* Check the tables again to see the record by using the select query.

https://miro.medium.com/max/52/1*ELQLTIHNEoyrYlecB7BabA.png?q=20



UPDATE CASCADE

* After executing the table Id 4 is updated into 5 in both tables.

**ALTER-DROP:**

**Syntax:** ALTER TABLE <TABLE NAME> DROP COLUMN <COLUMN

NAME>;

**Ex:** ALTER TABLE EMP DROP COLUMN SAL;

**d. SP\_RENAME:**

**Syntax:** SP\_RENAME „TABLENAME.OLDCOLUMN‟,‟NEW COLUMN

NAME‟,‟COLUMN,;

**Ex**

EXECUTE SP\_RENAME 'Employee.designation','job','COLUMN'

Execute sp\_rename 'product.price','cost','Column';

**\**

**3. SP\_RENAME:**

This command is used to change the table name from old table name to new

table name

**Syntax:** SP\_Rename „old table name‟,‟ New table name‟

**Ex:** SP\_Rename ‘EMP’,EMP1’

--change the datatype of a column

alter table Employee alter column Salary money not null

Difference between drop and Truncate

select \* from Customer

Truncate table Customer

Data will be deleted and structure of table will remain same

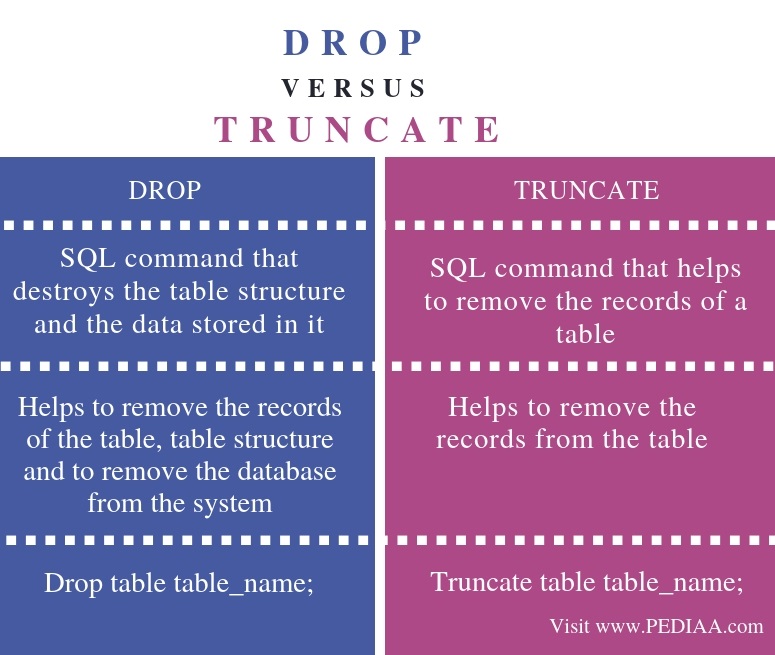
Drop table Customer

select \* from Customer

table with data is deleted

et’s see the difference between DELETE and TRUNCATE command:-

| **S.NO** | **DELETE** | **TRUNCATE** |
| --- | --- | --- |
| 1. | The DELETE command is used to delete specified rows(one or more). | While this command is used to delete all the rows from a table. |
| 2. | It is a DML(Data Manipulation Language) command. | While it is a DDL(Data Definition Language) command. |
| 3. | There may be WHERE clause in DELETE command in order to filter the records. | While there may not be WHERE clause in TRUNCATE command. |
| 4. | In the DELETE command, a tuple is locked before removing it | While in this command, data page is locked before removing the table data. |
| 5. | We can rollback the data even after using DELETE command. | While in this command, we can’t rollback. |
| 6. | DELETE command is slower than TRUNCATE command. | While TRUNCATE command is faster than DELETE command |



**difference between Truncate & Delete:**

* **Truncate is a DML command where delete is a DDL command.**
* **Both command does delete rows from table. Delete command can roll back where truncate command can not roll back**
* **Delete command can be used with where condition and delete specific rows but truncate will delete all rows from table**
* **Truncate command reset identity column but delete command doesn't**

Constraints

Not Null

Check

Unique

Priamary key

Foreign key

Default

--Constraints

create table Bank(Custid int Primary Key Identity(1,2),

Custname varchar(50) Unique,Balance decimal Check(Balance>500) Not null)

select \* from Bank

insert into Bank values('john',4000)

insert into Bank values('john',400)

--add column with default constraint

alter table Bank add bankname varchar(50) default 'sbi'

insert into Bank(Custname,Balance) values('selva',5000)

BANK Name is not given it takes sbi as default value

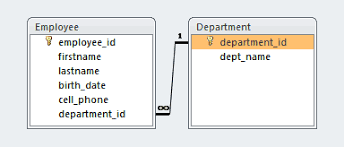
When we have relation between two tables is one to Many

One side PrimarY Key and many side we have to repeat the primary with same column and its data

Example

One department can have many Employees

deptId is Primary key and deptid have to be there in the Employee tale as foreign key



In on to many relation first we have to create a parent table it means dept without

Dept we can not have employees

Example:

create table dept(deptid int primary key identity(1,2),deptname varchar(50),location varchar(50))

insert into dept values('hr','chennai')

insert into dept values('MARKETING','Delhi')

insert into dept values('Development','Hyderabad')

insert into dept values('Training','mumbai')

insert into dept values('development','Bangaluru')

select \* from dept

🡪Table LEVEL

To create Employee Table(CHILD TABLE):-

create table Employee(EID int,ENAME varchar(50),SALARY money,Deptno int

foreign key references Department(Deptno))

example-2 COLUMN LEVEL

create table book (bookid int primary key identity,bookname varchar(50),

price decimal check(price>200),authid int,constraint fk\_auth\_book

foreign key(authid) references author(authid) on delete cascade)

create table author(authid int primary key identity,

authname varchar(50) not null)

🡪To Add Foreign Key to an Existing Table

alter table Employee add constraint fk\_dept\_emp foreign key(deptid) references dept(deptid)

deptid in dept table values must match with

deptid values in emptable otherwise we get foreign key violation value

so first update value in existing emp table and then runt he alter statement

select \* from Employee

update Employee set deptid=1 where empid=3

update Employee set deptid=3 where empid=4

update Employee set deptid=5 where empid=3

alter table Employee add constraint fk\_dept\_emp foreign key(deptid) references dept(deptid)

-🡪we can not insert into child table directly without having related record in

Parent table

That is we can not insert an employee directy into emp table without inserting dept record in dept table

When we impose the foreign key constraint and establish relation between the

table,the followiong three rules will come into picture.

**Rule1:-** Cannot insert a value into the foreign key column provided that value is

not existing under the refernce key column of the parent table.

**Rule2:-** Cannot update the reference key value of a parent table provided that

value has corresponding child record in the child table with out addressing what to

do with the child record.

**Rule3:-** Cannot delete a record from the parent table provided that records

reference key value has child record in the child table with out addressing what to

delete dept where deptid=1

Msg 547, Level 16, State 0, Line 65

The DELETE statement conflicted with the REFERENCE constraint "fk\_dept\_emp". The conflict occurred in database "PayRollManagement\_db", table "dbo.Employee", column 'deptid'.

The statement has been terminated.

do with the child record.

If we want to delete or update a record in the parent table when they have

corresponding child records in the child table we are provide with a set of rules to

perform delete and update operations knows as cascade rules.

**On**

We can create cascade rules while creating tables

(Or)

🡪If we want to add cascading rules to foreign key table first drop the foreign key constraint and run the alter statament

alter table Employee drop constraint fk\_dept\_emp

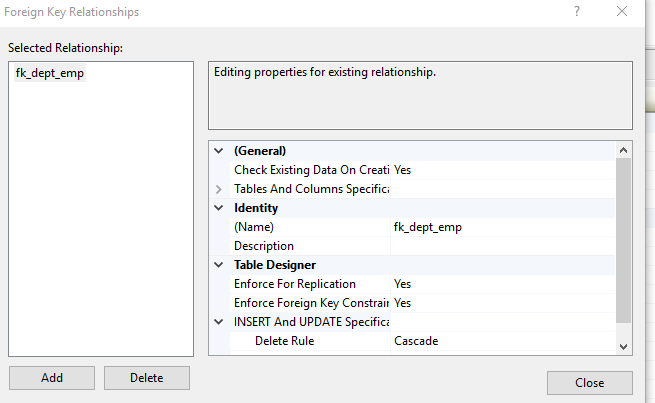
alter table Employee add constraint fk\_dept\_emp foreign key(deptid) references dept(deptid) on delete cascade on delete cascade

we can mention both the cascading rules

🡪to add cascade rules throught gui right click on foreignkey



to add with ssms



To see that on delete cascade is working or not

select \* from Employee

delete dept where deptid=3

select \* from Employee

example

To add multiple rows to a table at once, you use the following form of the INSERT statement:

**INSERT** **INTO** table\_name (column\_list)

**VALUES**

(value\_list\_1),

(value\_list\_2),

...

(value\_list\_n);

In this syntax, instead of using a single list of values, you use multiple comma-separated lists of values for insertion.

The number of rows that you can insert at a time is 1,000 rows using this form of the INSERT statement. If you want to insert more rows than that, you should consider using multiple INSERT statements, BULK INSERT or a derived table.

Note that this INSERT multiple rows syntax is only supported in SQL Server 2008 or later.

To insert multiple rows returned from a SELECT statement, you use the [INSERT INTO SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert-into-select/) statement.

## SQL Server INSERT multiple rows – examples

We will use the sales.promotions table created in the [previous tutorial](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) for the demonstration.

If you have not yet created the sales.promotions table, you can use the following [CREATE TABLE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) statement:

**CREATE** **TABLE** sales.promotions (

promotion\_id INT PRIMARY **KEY** **IDENTITY** (1, 1),

promotion\_name VARCHAR (255) **NOT** NULL,

discount NUMERIC (3, 2) **DEFAULT** 0,

start\_date DATE **NOT** NULL,

expired\_date DATE **NOT** NULL

);

### 1) Inserting multiple rows example

The following statement inserts multiple rows to the sales.promotions table:

**INSERT** **INTO** sales.promotions (

promotion\_name,

discount,

start\_date,

expired\_date

)

**VALUES**

(

'2019 Summer Promotion',

0.15,

'20190601',

'20190901'

),

(

'2019 Fall Promotion',

0.20,

'20191001',

'20191101'

),

(

'2019 Winter Promotion',

0.25,

'20191201',

'20200101'

);

SQL server issued the following message indicating that three rows have been inserted successfully.

(3 rows affected)

Let’s verify the insert by executing the following query:

**SELECT**

\*

**FROM**

sales.promotions;

Here is the output:



### 2) Inserting multiple rows and returning the inserted id list example

This example inserts three rows into the sales.promotions table and returns the promotion identity list:

**INSERT** **INTO**

sales.promotions (

promotion\_name, discount, start\_date, expired\_date

)

**OUTPUT** inserted.promotion\_id

**VALUES**

('2020 Summer Promotion',0.25,'20200601','20200901'),

('2020 Fall Promotion',0.10,'20201001','20201101'),

('2020 Winter Promotion', 0.25,'20201201','20210101');



In this example, we added the OUTPUT clause with the column that we want to return using the inserted.column\_name syntax. If you want to return values from multiple columns, you can use the following syntax:

OUTPUT inserted.column1, inserted.column2..

## introduction to SQL Server INSERT INTO SELECT statement

To [insert](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) data from other tables into a table, you use the following SQL Server INSERT INTO SELECT statement:

**INSERT** [ TOP ( expression ) [ **PERCENT** ] ]

**INTO** target\_table (column\_list)

**query**

In this syntax, the statement inserts rows returned by the query into the target\_table.

The query is any valid [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement that retrieves data from other tables. It must return the values that are corresponding to the columns specified in the column\_list.

The TOP clause part is optional. It allows you to specify the number of rows returned by the query to be inserted into the target table. If you use the PERCENT option, the statement will insert the percent of rows instead. Note that it is a best practice to always use the TOP clause with the [ORDER BY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-order-by/) clause.

## SQL Server INSERT INTO SELECT examples

Let’s [create a table](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) named addresses for the demonstration:

**CREATE** **TABLE** sales.addresses (

address\_id INT **IDENTITY** PRIMARY **KEY**,

street VARCHAR (255) **NOT** NULL,

city VARCHAR (50),

state VARCHAR (25),

zip\_code VARCHAR (5)

);

### 1) Insert all rows from another table example

The following statement [inserts](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) all addresses from the customers table into the addresses table:

**INSERT** **INTO** sales.addresses (street, city, state, zip\_code)

**SELECT**

street,

city,

state,

zip\_code

**FROM**

sales.customers

**ORDER** **BY**

first\_name,

last\_name;

To verify the insert, you use the following query:

**SELECT**

\*

**FROM**

sales.addresses;

Here is the result:

DATE FUNCTIONS

select DAY( GETDATE())

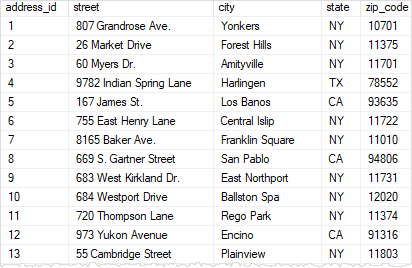
select \* from Employee

select DAY(dateofbirth) from Employee where empid=

select GETUTCDATE() from Employee where empid=2

Select DATENAME (DW, getdate ())

Select DATEDIFF (MM,'2012-12-15', getdate())

Select DATEPART(MM, getdate())

### 2) Insert some rows from another table example

Sometimes, you just need to insert some rows from another table into a table. In this case, you limit the number of rows returned from the query by using conditions in the [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause.

The following statement adds the addresses of the stores located in Santa Cruz and Baldwin to the addresses table:

**INSERT** **INTO**

sales.addresses (street, city, state, zip\_code)

**SELECT**

street,

city,

state,

zip\_code

**FROM**

sales.stores

**WHERE**

city **IN** ('Santa Cruz', 'Baldwin')

SQL Server returned the following message indicating that two rows have been inserted successfully.

(2 rows affected)

### 3) Insert the top N of rows

First, you use the following statement to delete all rows from the addresses table:

**TRUNCATE** **TABLE** sales.addresses;

Second, to insert the top 10 customers sorted by their first names and last names, you use the INSERT TOP INTO SELECT statement as follows:

**INSERT** TOP (10)

**INTO** sales.addresses (street, city, state, zip\_code)

**SELECT**

street,

city,

state,

zip\_code

**FROM**

sales.customers

**ORDER** **BY**

first\_name,

last\_name;

SQL Server returned the following message showing that ten rows have been inserted successfully.

(10 rows affected)

### 4) Insert the top percent of rows

Instead of using an absolute number of rows, you can insert a percent number of rows into a table.

First, [truncate](https://www.sqlservertutorial.net/sql-server-basics/sql-server-truncate-table/) all rows from the addresses table:

**TRUNCATE** **TABLE** sales.addresses;

Second, insert top two percent of rows from the customers table sorted by first names and last names into the addresses table:

**INSERT** TOP (10) **PERCENT**

**INTO** sales.addresses (street, city, state, zip\_code)

**SELECT**

street,

city,

state,

zip\_code

**FROM**

sales.customers

**ORDER** **BY**

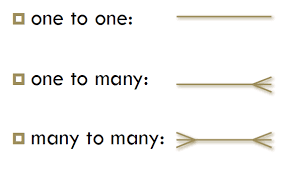
first\_name,

last\_name;

SQL Server issued the following message indicating that 145 rows have been inserted successfully.

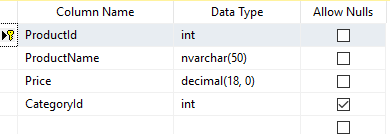
(145 rows affected)

In this tutorial, you have learned how to use the SQL Server INSERT INTO SELECT statement to insert rows from other tables into a table.

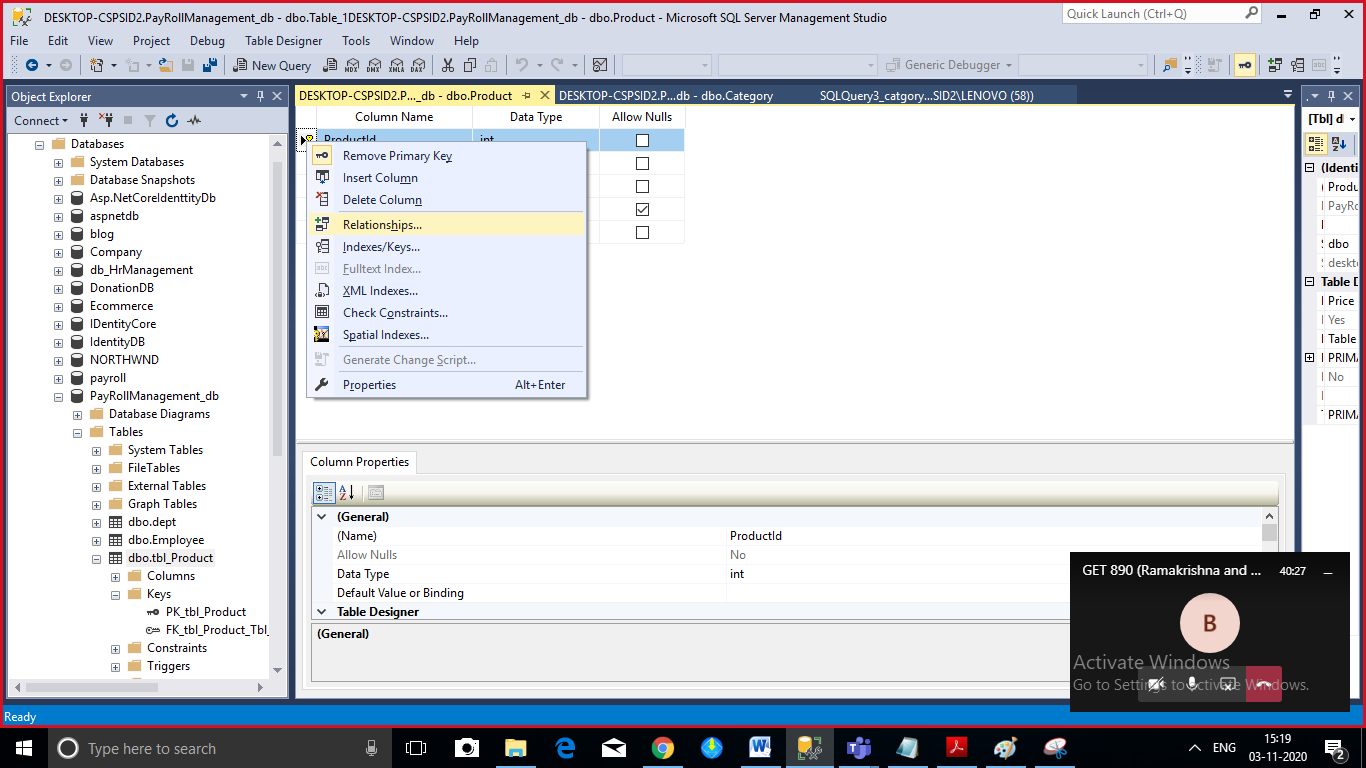


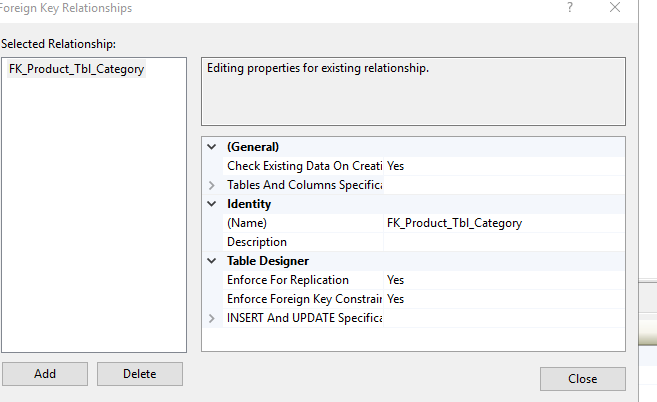
drop table tbl\_Category

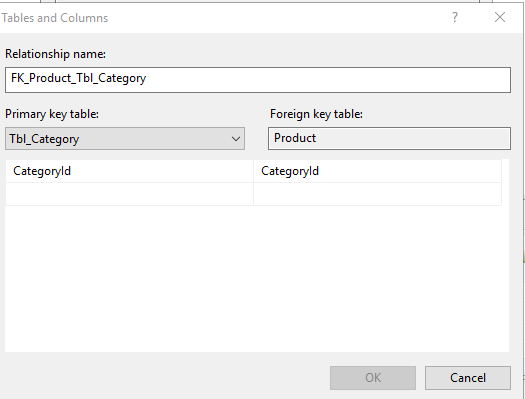
Could not drop object 'tbl\_Category' because it is referenced by a FOREIGN KEY constraint.



Right click on productid column and choose relationships

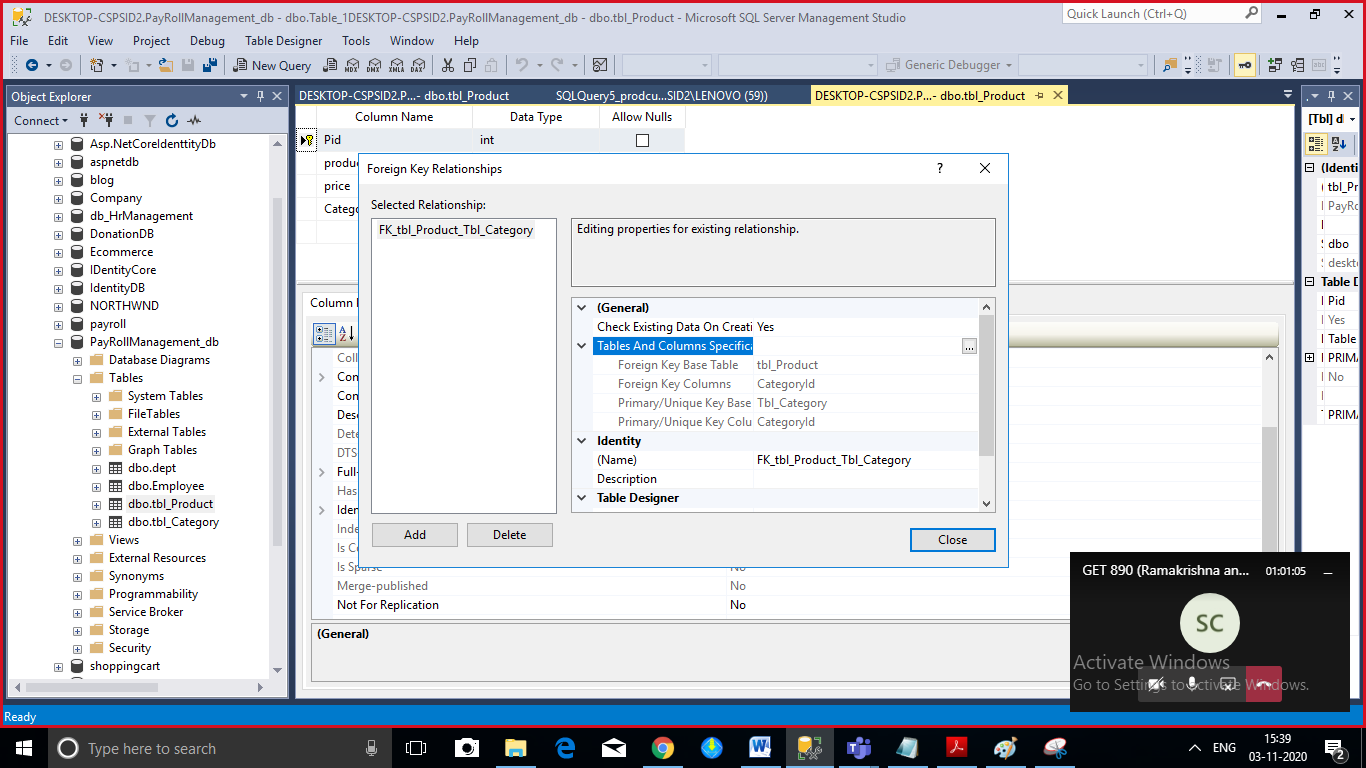


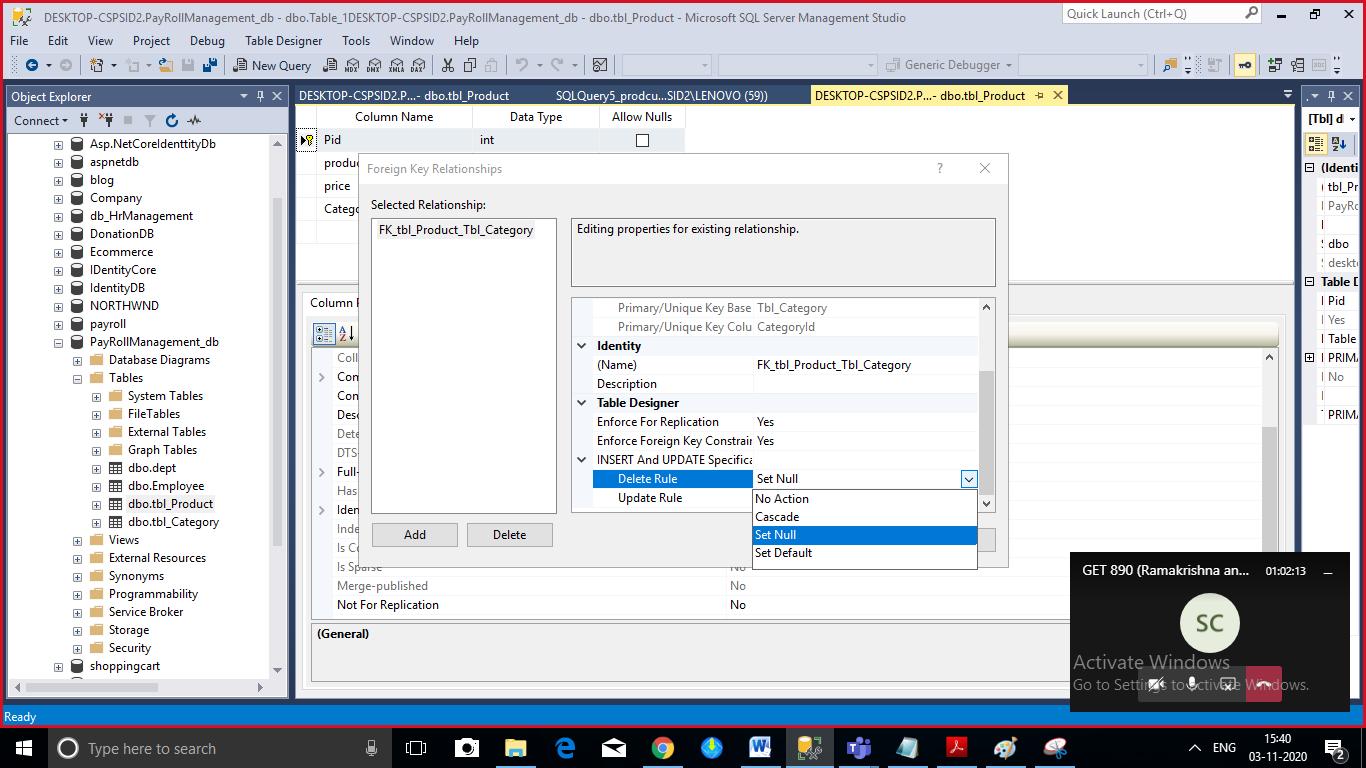




To give cascading rules after creating data using ssms

In child table





select \* from tbl\_Product

select \* from tbl\_Category

delete tbl\_Category where CategoryId=1

Many to many

when we have relation as many-many then break that relation into many to one and one to many and keep two foreign keys in intermediate table

customer and Proucts are related with Orders

one customer can order many products

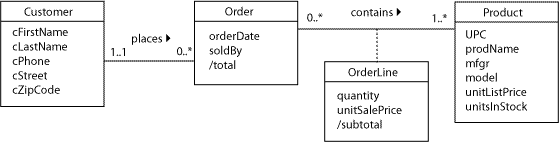
one product can be ordered by many Customers

table one will Customer with customerid as primary key

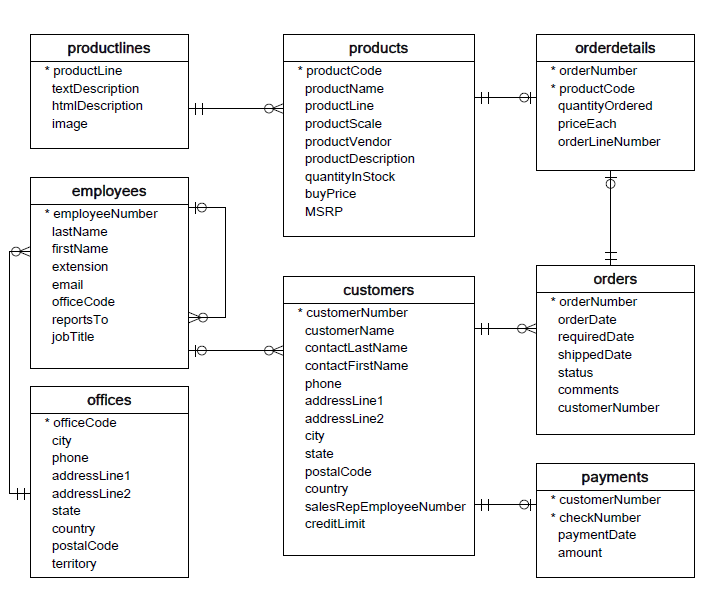
table two Product table will have product id as primarty key

pid and cid becomes foreignkeys in rder s table

OrderId will be primary key in Order table



Exercise create the following database with primary and foreign keys and cascadingrules and identity columns



Create a table product and dispay the total cost of product by multiplying product \* quantity

Write a query to reset the identity

What is sequence in sqlserver explain ith example

Write is difference sequence and identity

