**Defining stored procedures in sqlserver**

**A stored procedure is precompiled collection of transact sql statements stored with a name and processed as a unit.**

Sql server provides some precompiled stored procedures for managing sqlserver and displaying information about the databases and users. These sp are called system stored procedures.

Sp in sqlserver are similar to procedures in other languages as they can :

1. Accept input parameters and return values to the calling procedure or statements
2. Contain programming statements that perform operations in the database or call another stored procedure.
3. Return a status value to a calling procedure to indicate the success or failure

A sp can be as simple as a single select statement or a as complex as a series of sql statements using control of flow statements

**Benefits of sp**

1 – speed

2- faster access to data

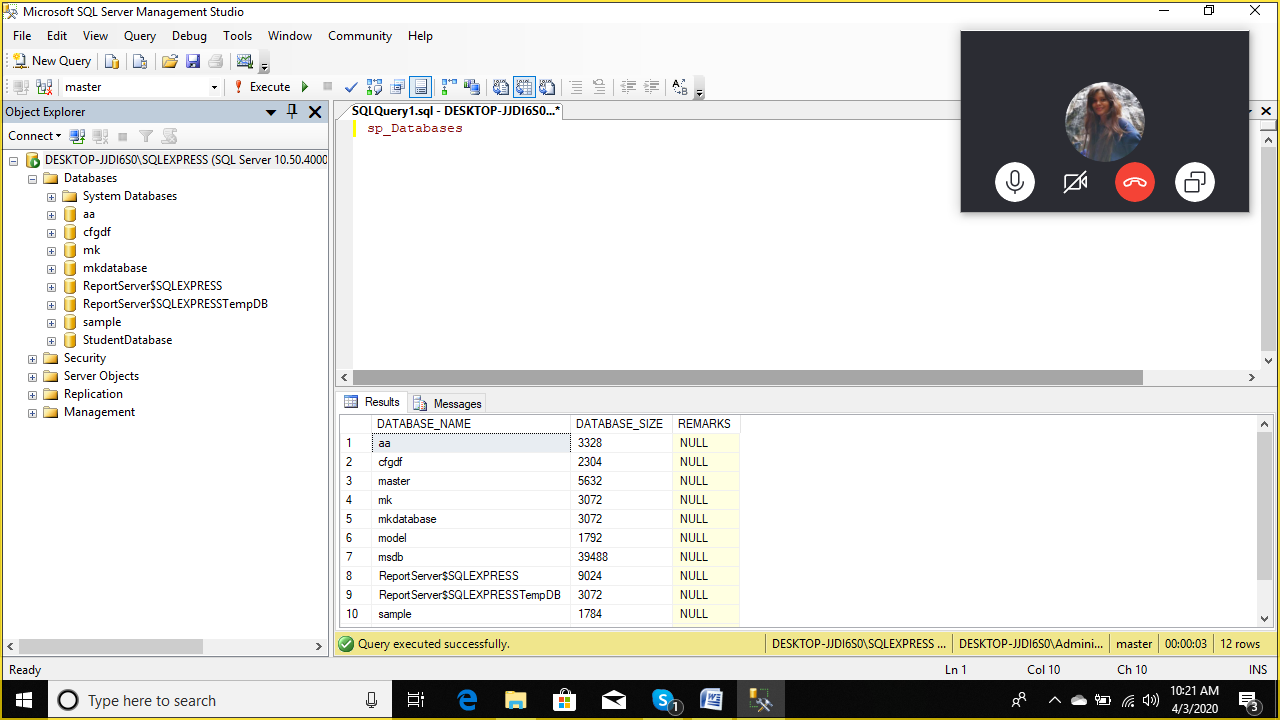
3 - less compilation time

**Types of stored procedures**

1. System stored procedure ( can be executed only)
2. User defined stored procedure(can be created and executed)

Some of the system stored procedures are

sp\_Databases - list all the databases available on sql server



Sp\_server\_info – lists the server information such as version etc

Sp\_Tables – list all the objects that can be used in select statement.

**User defined stored procedures**

**Ie you can create your own stored procedure**

**So to create sp**

**Create procedure or create proc statements are used**

Ie create proc <procedure\_name>

We are creating a stored procedure named proc1

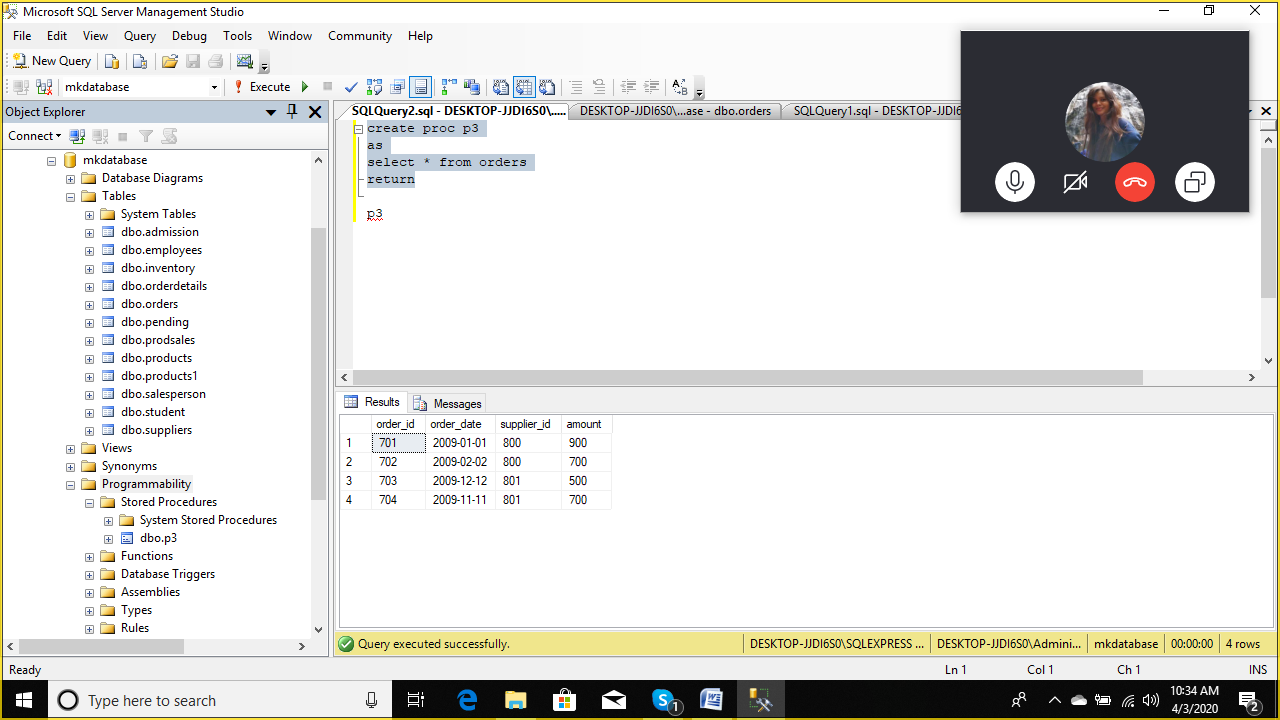
To get all the records from orders table

create proc p3

as

select \* from orders

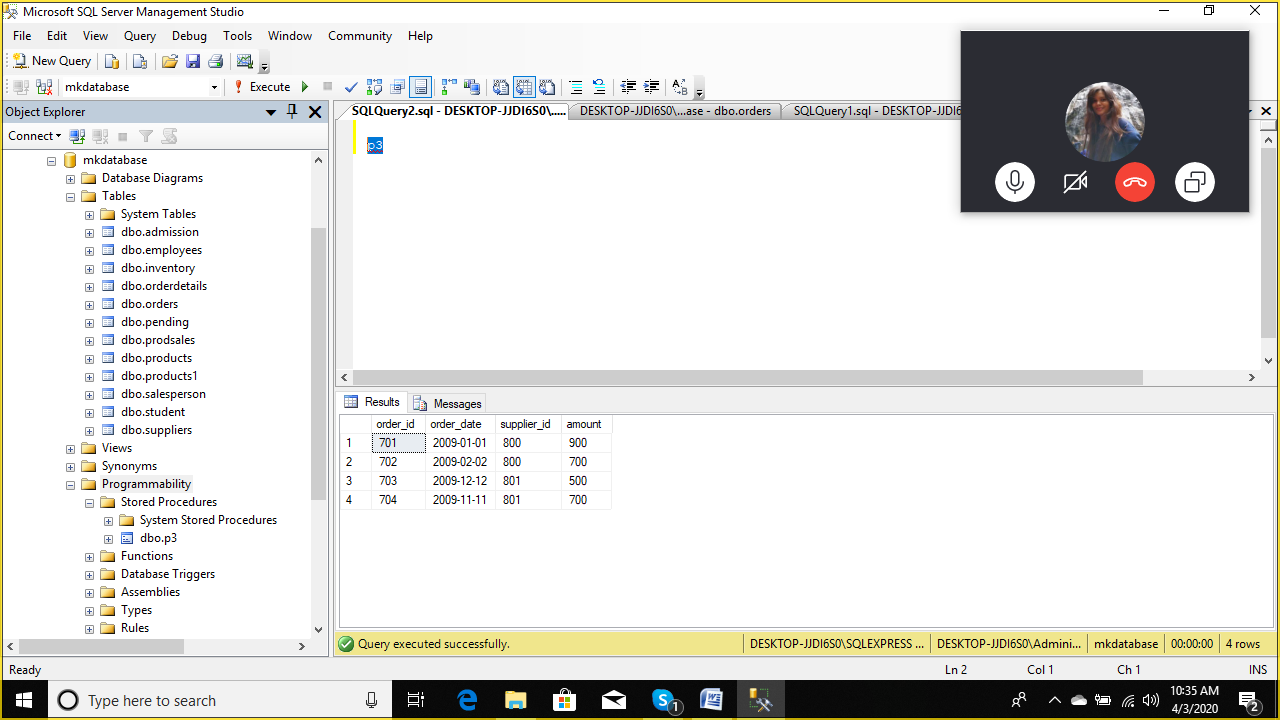
return



To execute a stored procedure

Type in query window

P3



create proc p4

as

select \* from orders

where order\_id=703

return

p4

create proc proc\_supplier

as

select ord.order\_id,ord.order\_date,ord.supplier\_id,ord.amount,sup.supplier\_name

from orders ord

inner join

suppliers sup

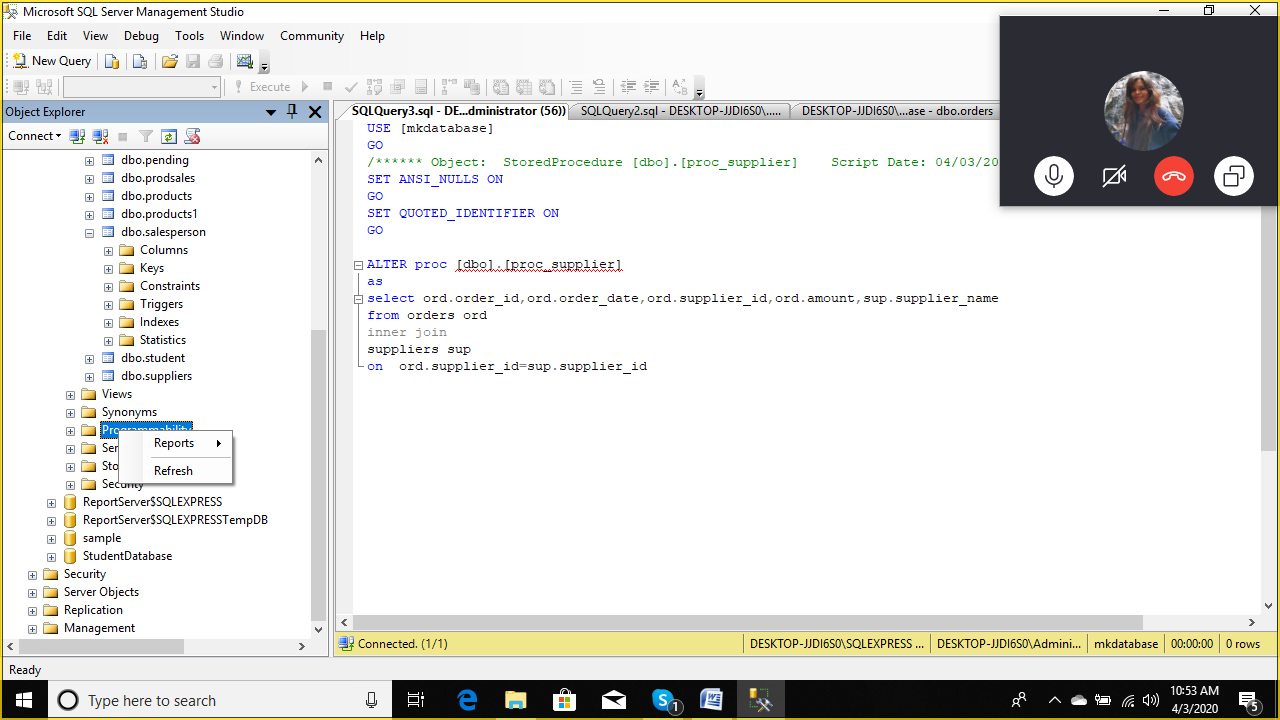
on ord.supplier\_id=sup.supplier\_id

--calling stored procedure

proc\_supplier

--- how to modify existing stored procedure

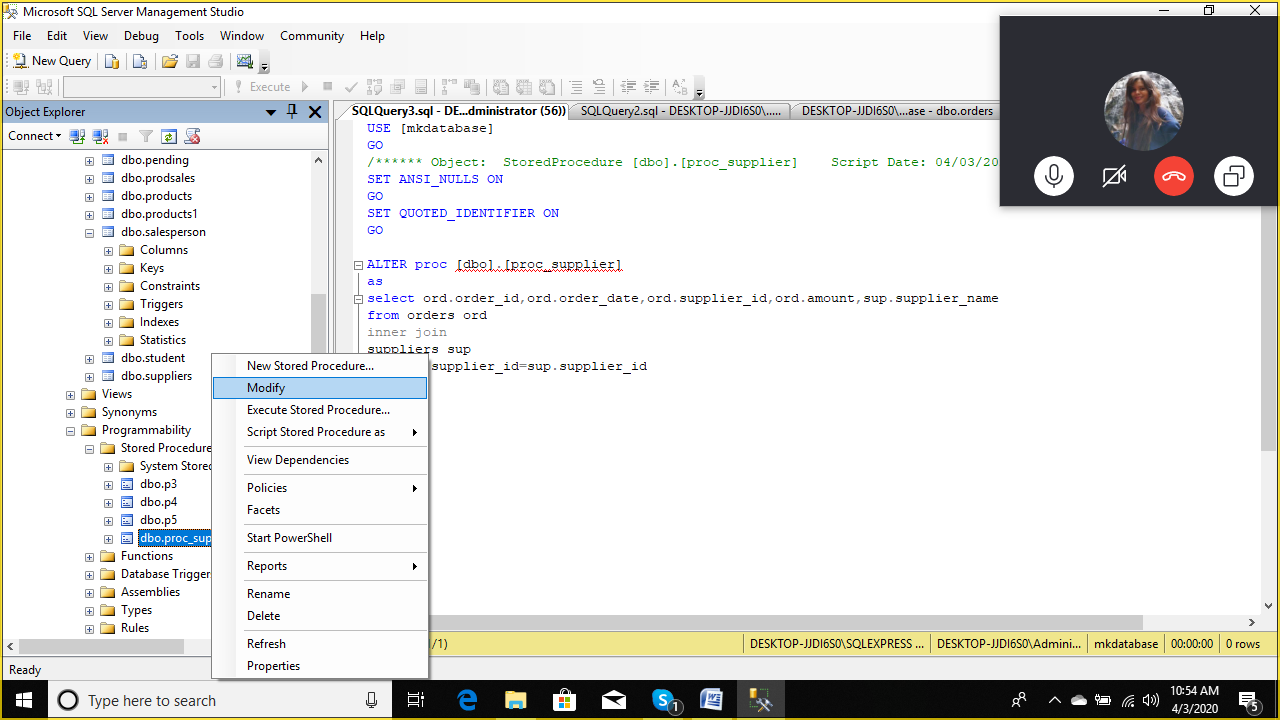
Go to programmability



Right click on it and select refresh

Click on + sign of progrmability

Click on + sign stored procedure



And select modify

USE [mkdatabase]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_supplier] Script Date: 04/03/2020 10:56:00 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER proc [dbo].[proc\_supplier]

as

select ord.order\_id,ord.order\_date,ord.supplier\_id,ord.amount,sup.supplier\_id,sup.supplier\_name

from orders ord

inner join

suppliers sup

on ord.supplier\_id=sup.supplier\_id

execute proc\_supplier

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

**Using parameters in stored procedures**

A sp communicates with the calling program through its parameters.

When a statement executes a stored proc it can pass values to the sp.

**Parameters are used to exchange data between stored procedures and the routine that called the stored procedure.**

**When you use parameters you can choose to allow the input or output values from the sp.**

**Syntax**

**Create proc proc\_name @parameter\_name data\_type**

**Example**

create proc newsum

@num1 int,

@num2 int

as

declare @mysum int

select 'Numbers ar : ' ,@num1, @num2

set @mysum = @num1 + @num2

select 'sum is : ' ,@mysum

now click on execute button

now to run the above stored proc we have to pass 2 no.

--to execute above sp

execute newsum 10,20

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

create proc procsquare

@num1 int

as

declare @sq int

set @sq=@num1 \* @num1

select 'square of number is : ' ,@sq

execute procsquare 5

execute procsquare 9

introduction to variable in sql server

a variable is an object that can hold a data value.

You can pass data to sql statements using local variables

The name of local variable must be prefixed with @sign

Syntax

Declare

{

@localvariable name datatype

}

For example

declare @ordno int

set @ordno=704

select \* from orders

where order\_id = @ordno

stored procedure with parameters

create proc procorders

@ordno int

as

select \* from orders

where order\_id=@ordno

--executing stored procedure

exec procorders 704

-- sp to view student record with 2 parameters rn and sn

create proc procstudent

@rn int,

@sn varchar(20)

as

select \* from student

where rno=@rn

and sname =@sn

--execute sp

execute procstudent 1,'amit'

**triggers**

**triggers are specical stored procedures created by user and provoked by sqlserver when data modification statements are issued.**

**Triggers are special objects created on the table and are part of database.**

**A trigger is invoked automically whenever the data in the table is modified. They are invoked in response to insert, update and delete sql statements.**

**Creating triggers**

**The code contained in the trigger has access to two virtual tables which are local to the trigger and are refered as inserted and deleted.**

**Deleted and inserted tables contain images of data prior to and after the updation. Data that is not affected by the update is not**

**Included in the inserted and deleted tables**

**Note : since the inserted and deleted tables are created dynamically in ram(random access memory) and do not use any permanent storage, these tables are called logical tables**

**Role played by the inserted and deleted tables while executing insert, update and delete triggers**

|  |  |  |
| --- | --- | --- |
| **trigger** | **Inserted table** | **Deleted table** |
| **Update** | **Copy of records after the command is completed.** | **Copy of the records before the command is executed.** |
| **Delete** | **Not used** | **Contains the data to be removed** |
| **Insert** | **Contains the data to be added** | **Not used** |

**Update table set sname = ‘ravi ‘ where rno=1**

**The following points should be considered before creating triggers**

1. **The create trigger statement must be the first statement while defining a trigger.**

**There are 3 types of triggers**

**1-insert trigger**

1. **Delete trigger**
2. **Update trigger**

**Syntax for creating a trigger**

**Create trigger trigger\_name**

**On table**

**For { [delete]. [insert] , [update] }**

**As**

**Sql statements**

**For example**

**Let us create an update trigger**

--creating update trigger trigger\_update\_student on the table student of mkdatabase database

-- trigger will be executed when an update is carried out on the student table

create trigger trigger\_update\_student on student

for update as

begin

print '\*\*\*\*\* student table \*\*\*\*\*\*'

select \* from student where rno=1

print '\*\*\*\* inserted table \*\*\*\*'

select \* from inserted

print '\*\*\*\*\* deleted table contains \*\*\*\*'

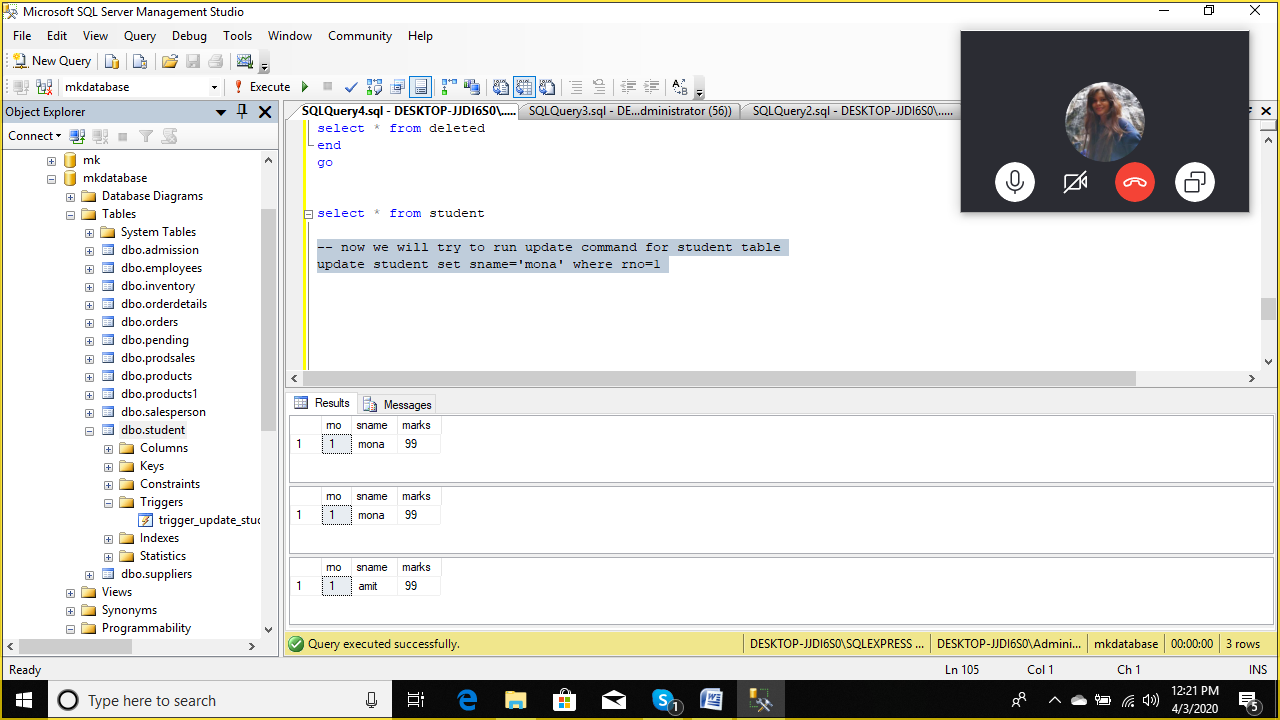
select \* from deleted

end

go

-- now we will try to run update command for student table

update student set sname='mona' where rno=1



**Now we see insert trigger**

**Insert triggers ensure the validity of the data being inserted.**

**When an insert transaction is detected the insert trigger is executed.**

**The inserted data is held in the logical inserted table.**

**--so we will be creating insert trigger**

**To insert record in student table if marks is less or equal to**

**100 other wise it will rollback**

create trigger markscheck

on student

for insert

as

if (select marks from inserted) <= 100

begin

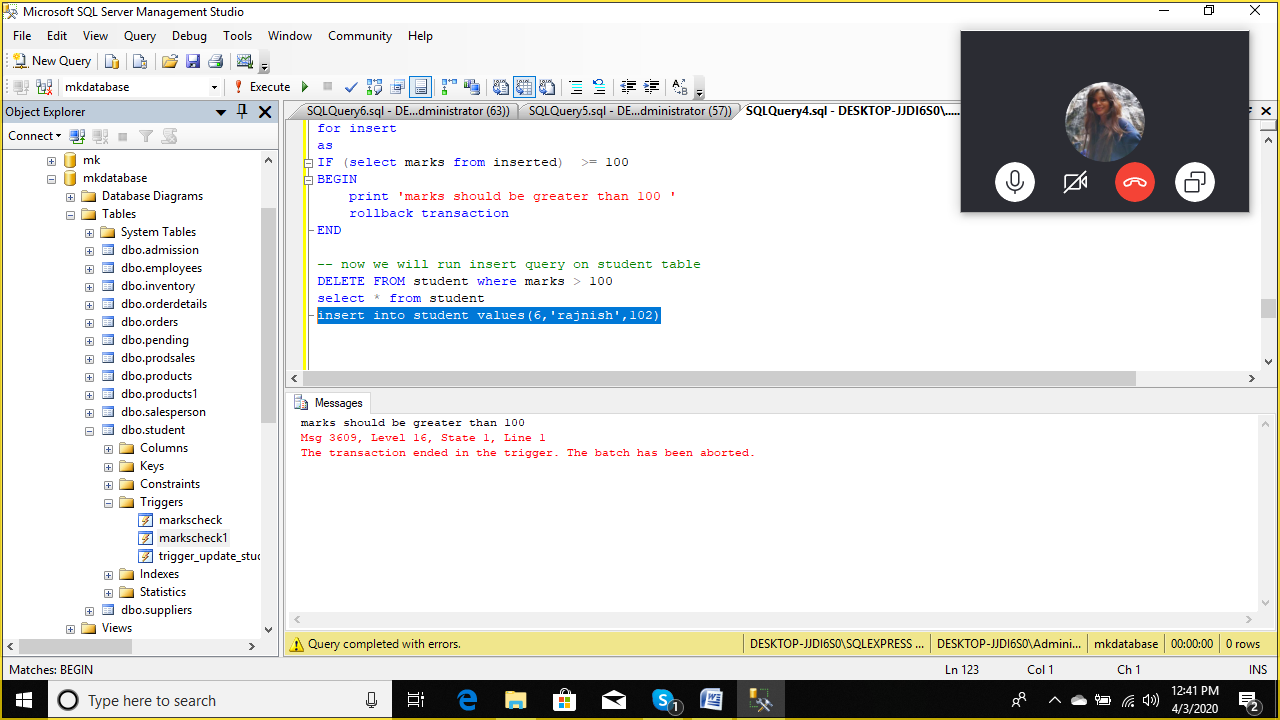
print 'marks should be greater than 100 '

rollback transaction

end

now when I run the insert query it will give me error

insert into student values(6,'rajnish',102)

****

insert into student values(6,'rajnish',99)

select \* from student

delete trigger

create