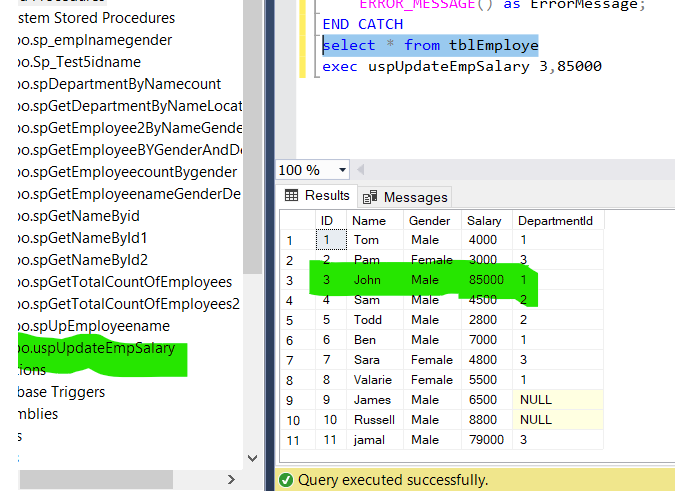
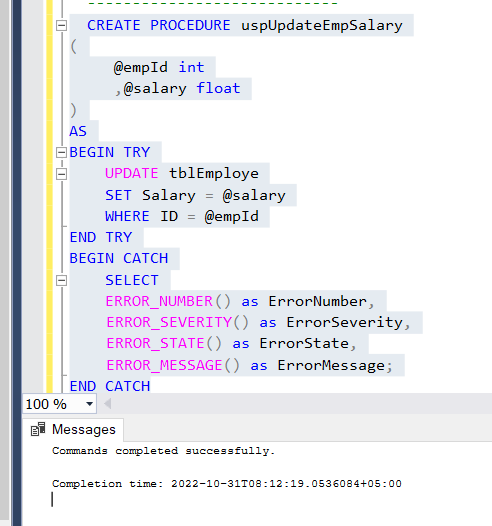
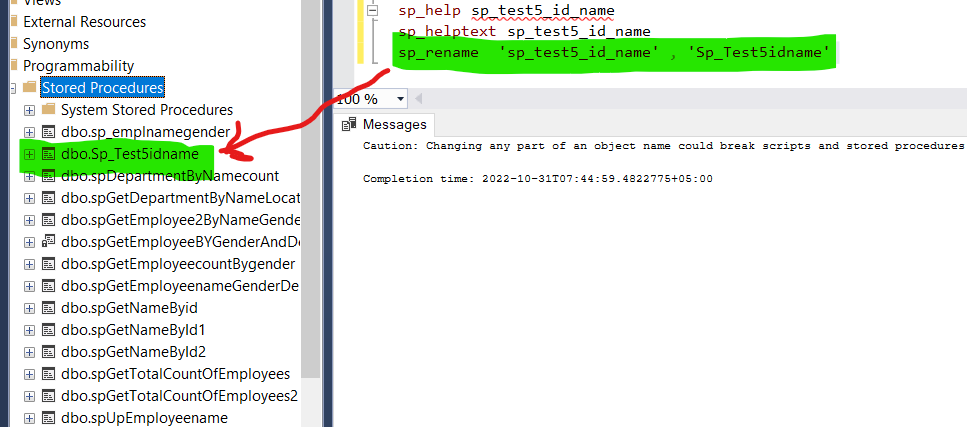
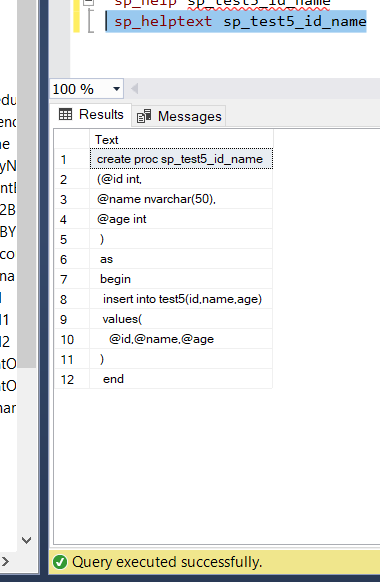
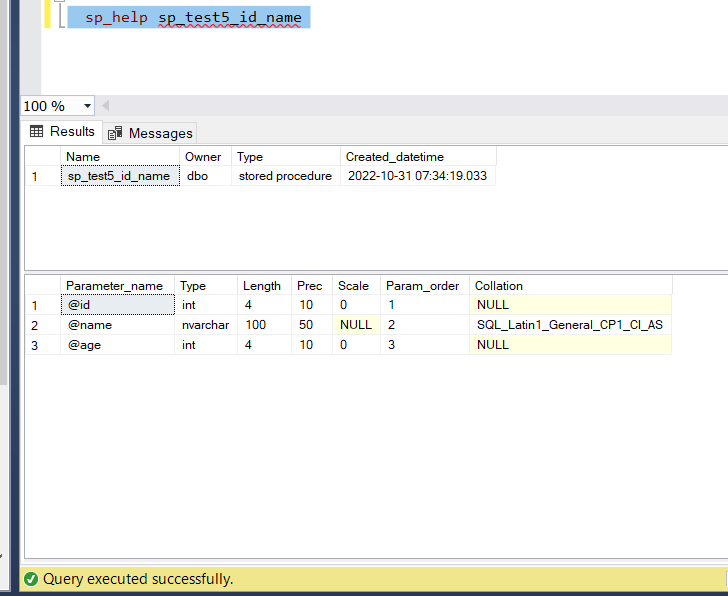
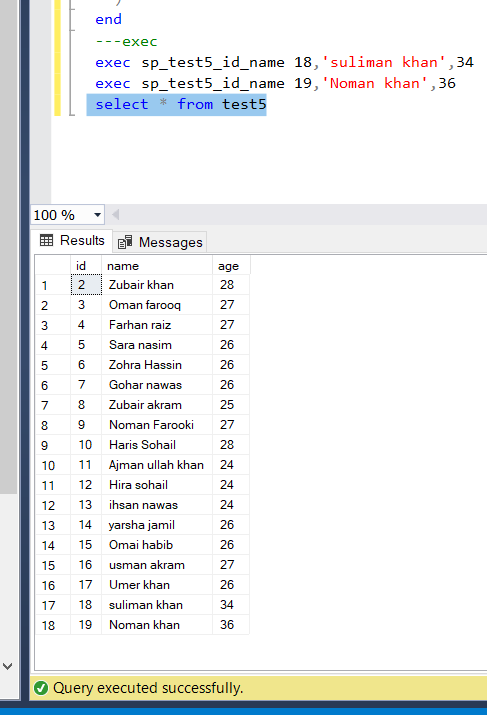
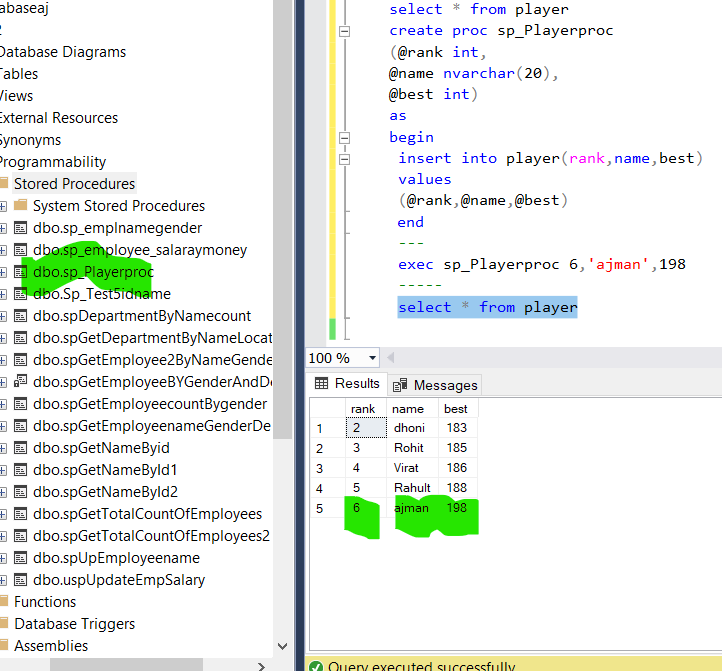
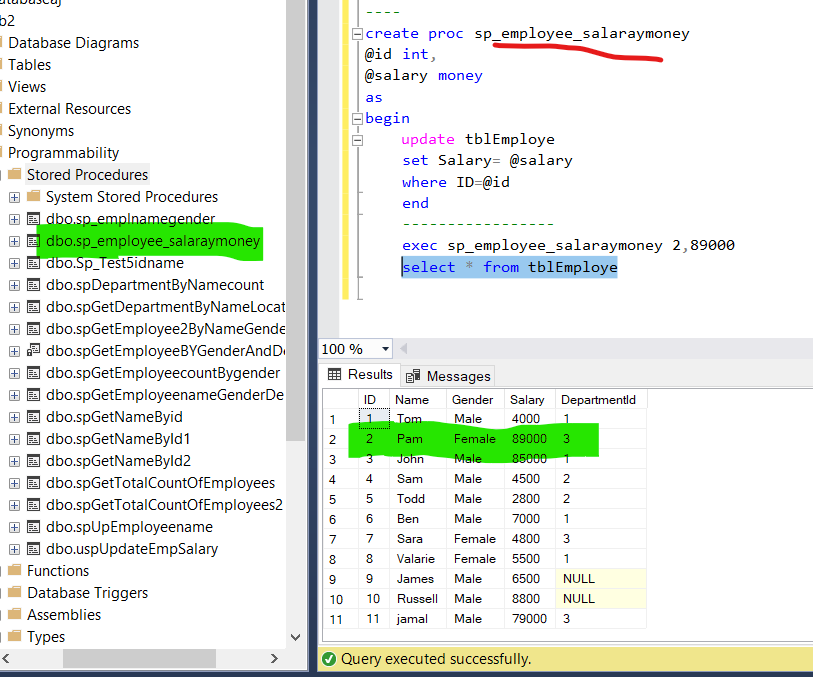
SQL Server Advanced



## Advantages of Stored procedures

* Stored procedures are reusable. Multiple users in multiple applications can use the same Stored Procedure (SP)
* As SPs reside in the database, it reduces network traffic. Applications have to make a procedure call to the database and it communicates back to the user.
* Database objects are encapsulated within a stored procedure, and this acts as a security mechanism by restricting access to the database objects.
* Reduced development cost, easily modified, and increased readability.
* Improves performance. When a stored procedure is executed for the first time, the database processor creates an execution plan which is re-used every time this SP is executed.



# SQL Server Indexes: Clustered Indexes

An Index in SQL Server is a data structure associated with tables and views that helps in faster retrieval of rows.

Data in a table is stored in rows in an unordered structure called Heap. If you have to fetch data from a table, the query optimizer has to scan the entire table to retrieve the required row(s). If a table has a large number of rows, then SQL Server will take a long time to retrieve the required rows. So, to speed up data retrieval, SQL Server has a special data structure called indexes.

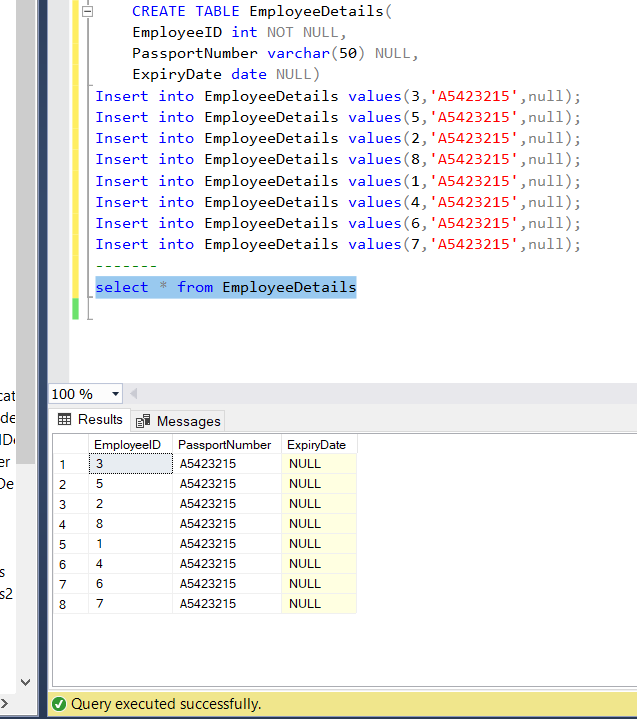
An index is mostly created on one or more columns which are commonly used in the [SELECT clause](https://www.tutorialsteacher.com/sqlserver/select-query) or [WHERE clause](https://www.tutorialsteacher.com/sqlserver/where-clause).

## Clustered Indexes

The clustered index defines the order in which the table data will be sorted and stored. As mentioned before, a table without indexes will be stored in an unordered structure. When you define a clustered index on a column, it will sort data based on that column values and store it. Thus, it helps in faster retrieval of the data.

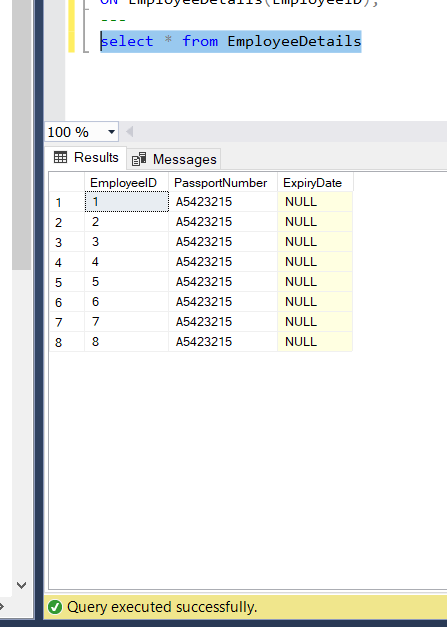
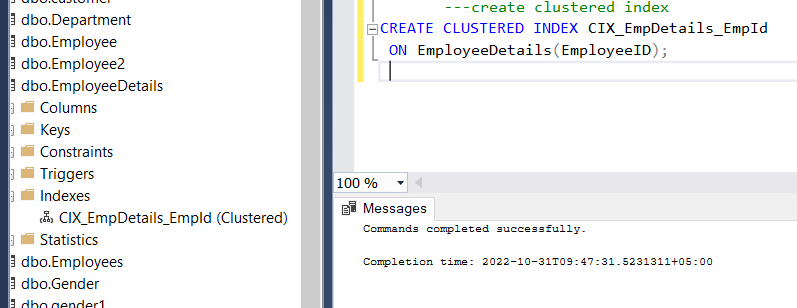
There can be only one clustered index on a table because the data rows can be stored in only one order.

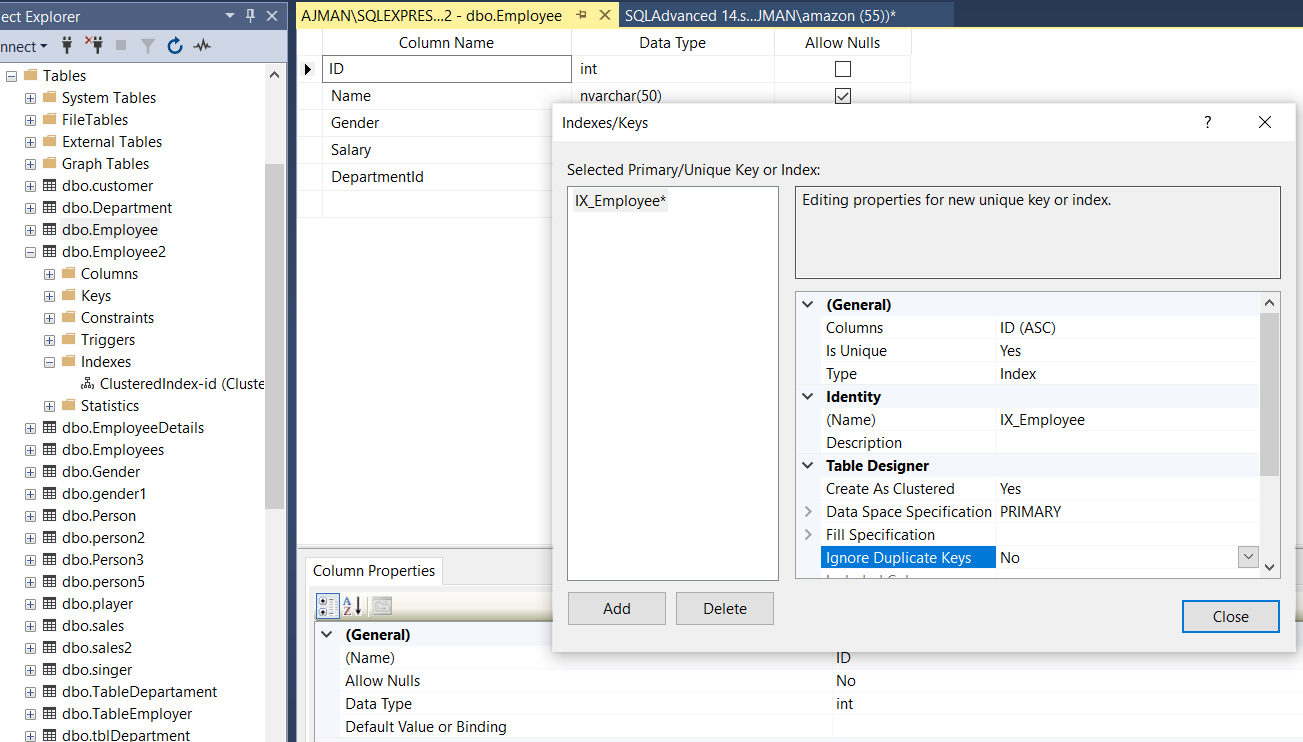
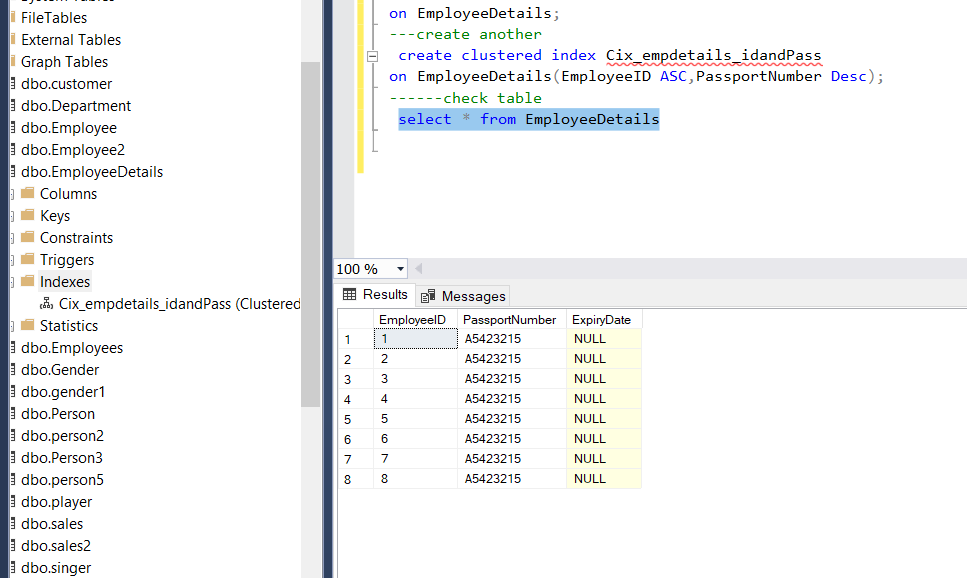
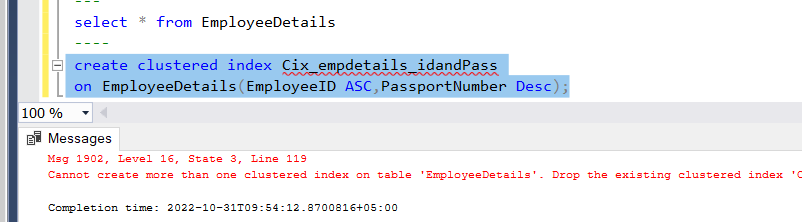
When you create a Primary Key constraint on a table, a unique clustered index is automatically created on the table.



As you can see, the data is stored in the order in which they are inserted. Now, if you use the WHERE clause to filter out data, the query optimizer will scan the entire table rows to arrive at the required data. This is because the data is not in any order.

Create a clustered index on the EmployeeID column to sort and store the the data by EmployeeID. The data in the table will be sorted based on EmployeeID and executing the above query will return rows faster. This can be verified using SQL Server's execution plan



In a clustered index, the data is organized using a special structure called B-tree or a balanced tree structure. In B-tree, the top node is called the root node and the bottom nodes are called the leaf nodes. All index levels between the root and the leaf nodes are called intermediate levels. The leaf nodes contain the data pages. The root and intermediate levels contain index pages holding index rows and each index row contains a pointer either pointing to a data row in leaf node or to another intermediate level page. The pages in each level of the index are linked in a doubly linked list.

SQL Server provides two types of indexes, clustered and non-clustered indexes. Here you will learn non-clustered indexes.

The non-clustered index does not sort the data rows physically. It creates a separate key-value structure from the table data where the key contains the column values (on which a non-clustered index is declared) and each value contains a pointer to the data row that contains the actual value. It is similar to a textbook having an index at the back of the book with page numbers pointing to the actual information.

**Coding:**

select \* from Employee2

-----

create proc sp\_emplnamegender

as

begin

select ID,

Name,Gender from

Employee2

end

-- exex

exec sp\_emplnamegender

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

select \* from test5

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

create proc sp\_test5\_id\_name

(@id int,

@name nvarchar(50),

@age int

)

as

begin

insert into test5(id,name,age)

values(

@id,@name,@age

)

end

---exec

exec sp\_test5\_id\_name 18,'suliman khan',34

exec sp\_test5\_id\_name 19,'Noman khan',36

select \* from test5

sp\_help sp\_test5\_id\_name

sp\_helptext sp\_test5\_id\_name

sp\_rename 'sp\_test5\_id\_name' , 'Sp\_Test5idname'

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

CREATE PROCEDURE uspUpdateEmpSalary

(

@empId int

,@salary float

)

AS

BEGIN TRY

UPDATE tblEmploye

SET Salary = @salary

WHERE ID = @empId

END TRY

BEGIN CATCH

SELECT

ERROR\_NUMBER() as ErrorNumber,

ERROR\_SEVERITY() as ErrorSeverity,

ERROR\_STATE() as ErrorState,

ERROR\_MESSAGE() as ErrorMessage;

END CATCH

select \* from tblEmploye

exec uspUpdateEmpSalary 3,85000

----

create proc sp\_employee\_salaraymoney

@id int,

@salary money

as

begin

update tblEmploye

set Salary= @salary

where ID=@id

end

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

exec sp\_employee\_salaraymoney 2,99000

select \* from tblEmploye

----lets take another examples

select \* from player

create proc sp\_Playerproc

(@rank int,

@name nvarchar(20),

@best int)

as

begin

insert into player(rank,name,best)

values

(@rank,@name,@best)

end

---

exec sp\_Playerproc 6,'ajman',198

-----

select \* from player

select \* from tblDepartment

create proc sp\_Departmenthead

@empid int,

@Dephead nvarchar(20) output

as

begin

select DepartmentHead=@Dephead

from tblDepartment

where ID=@empid

end

---execution

declare @Dephead varchar(20)

exec sp\_Departmenthead 5,@Dephead output

print @Dephead

--------------create cluster indexes

CREATE TABLE EmployeeDetails(

EmployeeID int NOT NULL,

PassportNumber varchar(50) NULL,

ExpiryDate date NULL)

Insert into EmployeeDetails values(3,'A5423215',null);

Insert into EmployeeDetails values(5,'A5423215',null);

Insert into EmployeeDetails values(2,'A5423215',null);

Insert into EmployeeDetails values(8,'A5423215',null);

Insert into EmployeeDetails values(1,'A5423215',null);

Insert into EmployeeDetails values(4,'A5423215',null);

Insert into EmployeeDetails values(6,'A5423215',null);

Insert into EmployeeDetails values(7,'A5423215',null);

-------

select \* from EmployeeDetails

---create clustered index

CREATE CLUSTERED INDEX CIX\_EmpDetails\_EmpId

ON EmployeeDetails(EmployeeID);

---

select \* from EmployeeDetails

----

create clustered index Cix\_empdetails\_idandPass

on EmployeeDetails(EmployeeID ASC,PassportNumber Desc);

----Drop index

Drop index CIX\_EmpDetails\_EmpId

on EmployeeDetails;

---create another

create clustered index Cix\_empdetails\_idandPass

on EmployeeDetails(EmployeeID ASC,PassportNumber Desc);

------check table

select \* from EmployeeDetails

-- drop cluter index

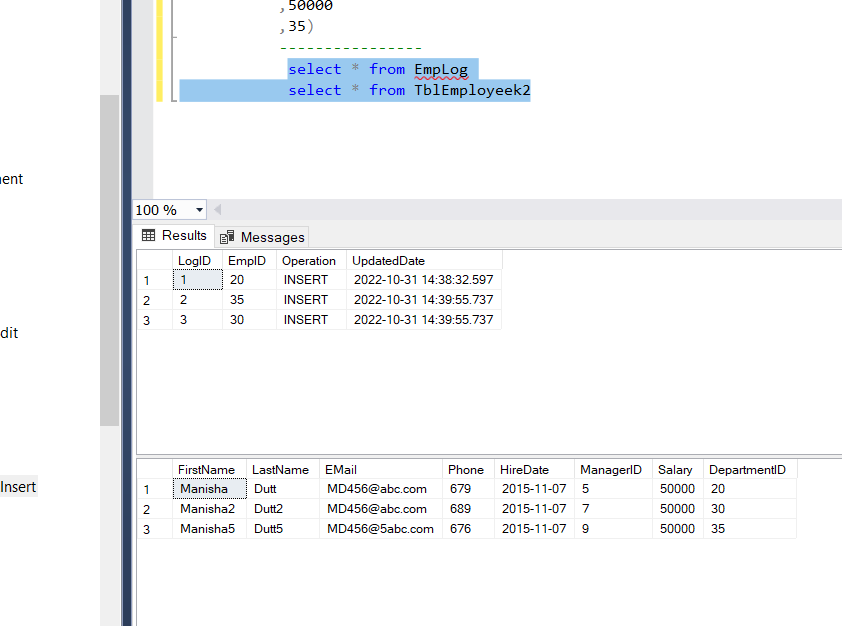
select \* from tblEmployee

--- create non-clustored index

select \* from Person

create nonclustered index ncix\_Email

on Person(Email);

 --- crete trigger

CREATE TABLE EmpLog (

LogID int IDENTITY(1,1) NOT NULL,

EmpID int NOT NULL,

Operation nvarchar(10) NOT NULL,

UpdatedDate Datetime NOT NULL

)

CREATE TABLE TblEmployeek (

LogID int IDENTITY(1,1) NOT NULL,

EmpID int NOT NULL,

Operation nvarchar(10) NOT NULL,

UpdatedDate Datetime NOT NULL

)

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

CREATE TRIGGER trgEmployeeKInsert

ON TblEmployeek2

FOR INSERT

AS

INSERT INTO EmpLog(EmpID, Operation, UpdatedDate)

SELECT DepartmentID ,'INSERT',GETDATE() FROM INSERTED; --virtual table INSERTED

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

create table TblEmployeek2(FirstName nvarchar(20),

LastName nvarchar(20)

,EMail nvarchar(20)

,Phone int

,HireDate date

,ManagerID int

,Salary int

,

DepartmentID int )

INSERT INTO TblEmployeek2

VALUES('Manisha2'

,'Dutt2'

,'MD456@abc.com'

,689

,'11/07/2015'

,7

,50000

,30),

('Manisha5'

,'Dutt5'

,'MD456@5abc.com'

,676

,'11/07/2015'

,9

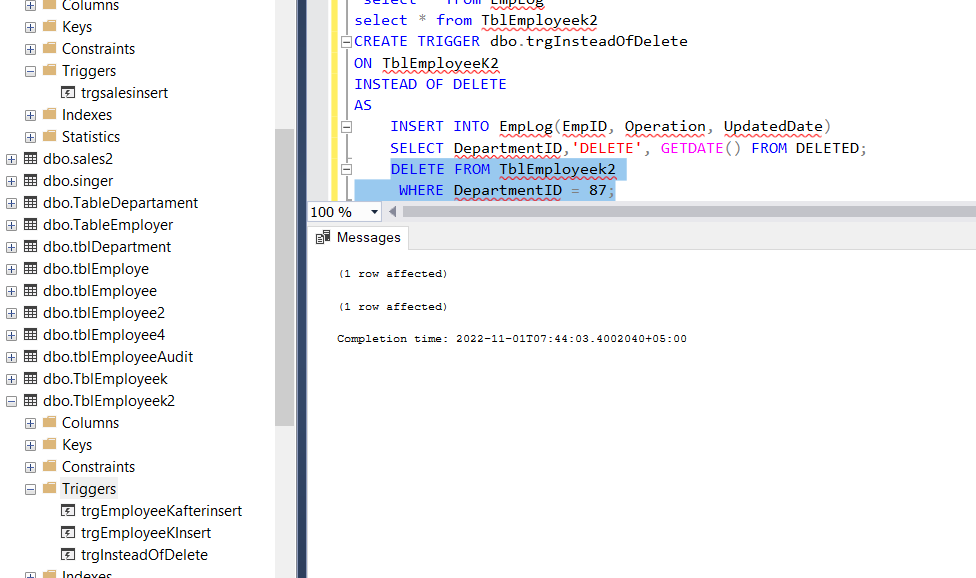
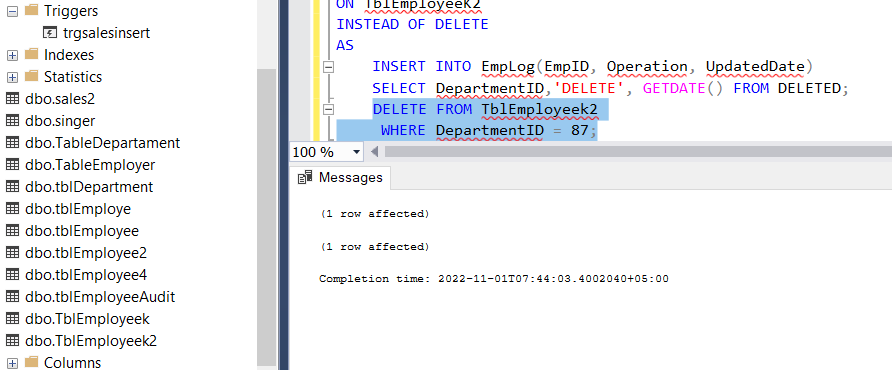
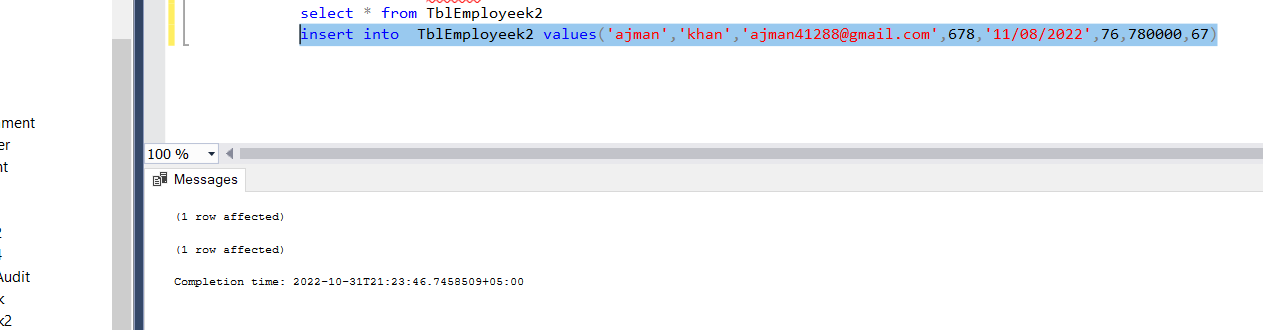
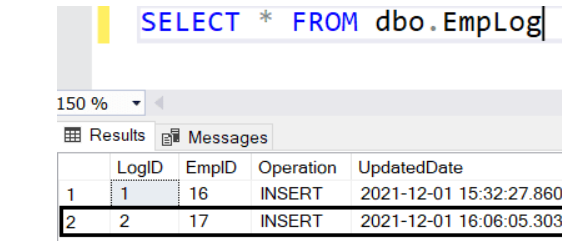
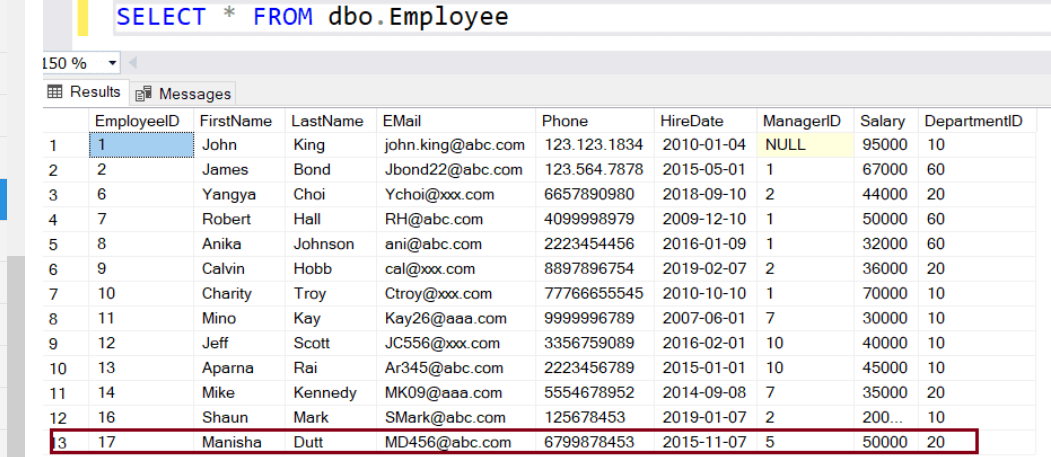
,50000

,35)

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

select \* from EmpLog

select \* from TblEmployeek2



sp\_settriggerorder [ @triggername = ] 'triggername',[ @order = ] 'value', [ @stmttype = ] 'statement\_type', [ @namespace = { 'DATABASE' | 'SERVER' | NULL } ]

sp\_settriggerorder @triggername= 'dbo.trgEmployeeUpdate',

@order='First',

@stmttype = 'UPDATE';

