

CREATE TABLE Customer

(

CustomerID int IDENTITY(10,5)primary key,

[Name] VARCHAR(50),

City VARCHAR (50),

Department VARCHAR (10)

)

INSERT INTO Customer(Name,City,Department, MobileNo) VALUES('Jack','Pune','HR',2131242234)

INSERT INTO Customer(Name,City,Department,MobileNo) VALUES('Stive','Mumbai','Analyst',2434234)

INSERT INTO Customer(Name,City,Department,MobileNo) VALUES('Welltor','DC','Buss',231323)

INSERT INTO Customer(Name,City,Department,MobileNo) VALUES('CEF','EWE','DWD',231323)

ALTER TABLE Customer

ADD MobileNo Int

CREATE TABLE Orders

(

OrderID int IDENTITY(10,3) primary key,

Product VARCHAR(50),

[Type] VARCHAR (50),

)

INSERT INTO Orders VALUES ('Tea','TATA',10)

INSERT INTO Orders(Product,Type,CustomerID) VALUES('Tea','TATA',10)

INSERT INTO Orders(Product,Type,CustomerID) VALUES('Coffee','OUUU',15)

INSERT INTO Orders(Product,Type,CustomerID) VALUES('MILK','AMUL',20)

INSERT INTO Orders(Product,Type,CustomerID) VALUES('MILK','AMUL',25)

INSERT INTO Orders(Product,Type,CustomerID) VALUES('MILK','AMUL',37)

ALTER TABLE Orders

ADD CustomerID Int

SELECT \* FROM Customer

SELECT \* FROM Orders

DROP TABLE Customer

DROP TABLE Orders

ALTER TABLE Orders

ADD CONSTRAINT Orders\_CustomerID\_FK

FOREIGN KEY (CustomerID)

REFERENCES Customer(CustomerID)

ALTER TABLE Orders

ADD CONSTRAINT NOTNULL

ALTER TABLE Orders

DROP CONSTRAINT Orders\_CustomerID\_FK

////////////////////////////////////////////////////////////////////////////////////////////////

CREATE TABLE Orders

(

OrderID int IDENTITY(10,3) primary key,

Product VARCHAR(50),

CustomerID int,

[Type] VARCHAR (50),

FOREIGN KEY CustomerID REFERENCES Customer(CustomerID)

)

////////////////////////////////////////// **SELECT QUERY**///////////////////////////////////////////////////  
  
SELECT \* FROM Employee

SELECT Department,Designation from Employee

SELECT Department, EmpName FROM Employee where Department = 'Accounts'

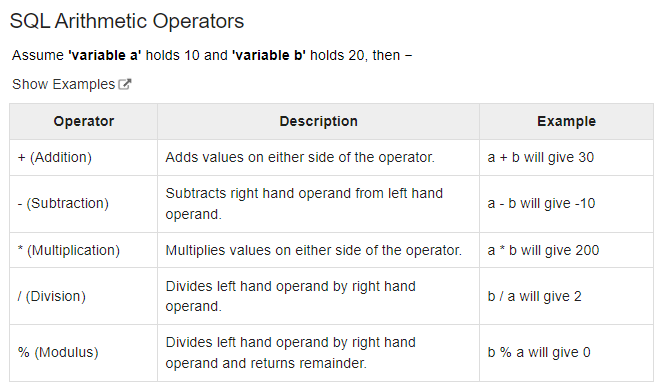
SELECT Department, EmpName FROM Employee where EmpName = 'Matts'

SELECT DEPARTMENT, EmpName from Employee where EmpID <> 3-- NOT EQUAL TO

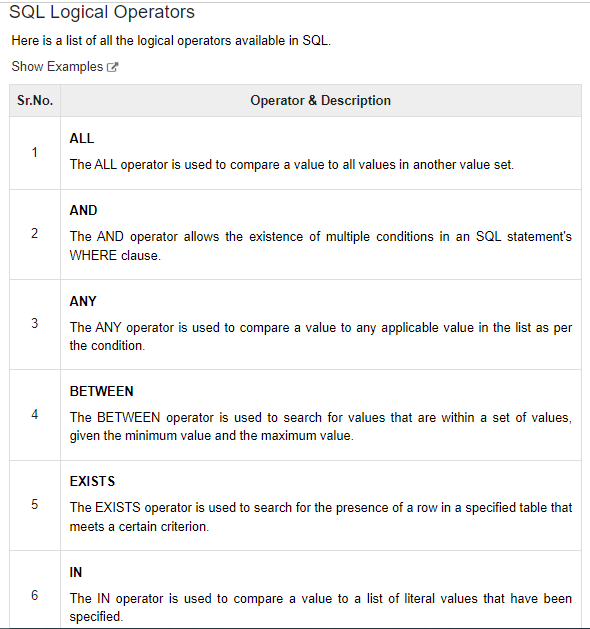
SELECT DEPARTMENT, EmpName from Employee where EmpID != 4-- NOT EQUAL TO

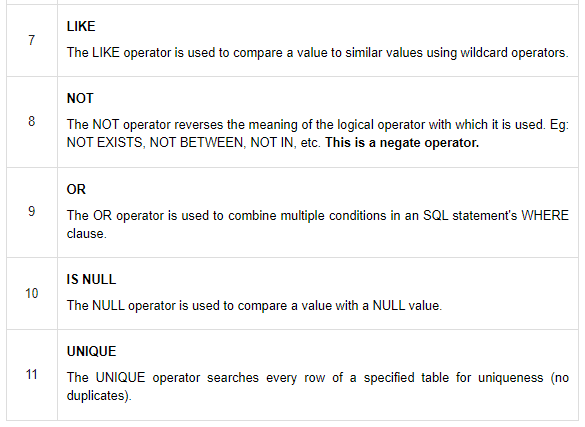
SELECT \* FROM Employee WHERE department='it' and salary >50000

select empname,designation from employee where department ='accounts' and Joiningdate>2021-11-20  
  
SELECT department, salary from employee where Designation ='lab assistant' or Designation ='accountant'

select department,salary from employee where Designation in('lab assistant','accountant')  
  
  
what is SQL operators?  






  
  
select department,empname,Salary from employee where salary between 100000 and 500000  
  
select department,empnamefrom employee where salary between 100000 and 500000

select department,empname,Salary from employee where salary between 100000 and 500000

select department, salary,empname from Employee where empname like '%n%'

select department, salary,empname from Employee where empname like 'n%' –start from n

select department, salary,empname from Employee where empname like '%n' –end from n

select department, salary,empname from Employee where empname like '%N Y%' --start and end letters

select \* from employee where Department like '\_\_B%' –no matter which is starting letters

select \* from employee where Department like '\_ccounts% '

select \* from Employee where EmpName like '[pmdg]%' –start contains pmdg

select \* from Employee where EmpName like '%[DLO]' –end contains DLO

select \* from Employee where EmpName like '%[OUD]%'

select \* from Employee where empname like '[a-h]%' –start contains a to h

select \* from Employee where empname like '%[a-d]' –end contains a to d

select \* from Employee where empname like '%[o-r]%'

select \* from Employee order by EmpName

select \* from employee order by EmpName desc

select \* from employee order by EmpName asc

select empname,Department,salary from Employee order by Salary,EmpName

select empname,Department,salary from Employee order by EmpName,Salary

select top 5 empname,salary,department from Employee order by salary

select top 5 empname,salary,department from Employee order by salary desc

select top 5 \* from Employee order by salary

**distinct –Not take duplicate value**

select department from Employee

select distinct department from Employee

select distinct department,empname from Employee

select distinct Engineername,Department,salary from Engineer

select distinct Engineername,Department,salary from Engineer order by salary,Engineername

IDENTITY  
  
CREATE TABLE EMPLOYEE

(  
ID int identity (1,1)--🡪> 1-start number 1 increment number  
name varchar(50) not null

)  
  
delete from employee where ID =2  
  
**SET IDENTITY \_INSERT EMPLOYEE ON  
SET IDENTITY \_INSERT EMPLOYEE OFF**  
  
DELETE DROP TRUNKET  
  
**DELETE FROM Employee** 🡪all data delete from table but table structure not deleted and **identity value not reset.**  
**TRUNCATE TABLE Employee** ->> all data deleted from table but table structure not deleted and **identity value reset.  
  
DROP TABLE Employee 🡪>**The DROP TABLE statement is used to drop an existing table in a database.

* delete Engineer1
* TRUNCATE table Engineer1
* drop table Engineer1

delete from Engineer where ID=8  
  
  
  
  
1.SQL DEFAULT CONSTRAINT

( chapterNo int identity (3,5) primary key ,

chaptername varchar (50) not null,

pageno int,

datecompleted datetime,

teacher varchar(10) DEFAULT 'Guru'

)

select \* from book

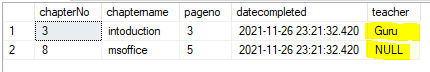
drop table book

insert into book(chaptername,pageno,datecompleted)

values('intoduction',3,getdate())

insert into book(chaptername,pageno,datecompleted,teacher)

values('msoffice',5,getdate(),null)



Add Default constraint

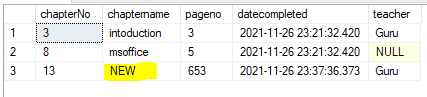
ALTER TABLE book

add constraint DF\_book\_DOJ

default 'NEW' for chaptername

insert into book(pageno,datecompleted)

values(653,getdate())



Delete constraint

alter table book

drop constraint DF\_book\_DOJ

2. SQL UNIQUE CONSTRAINT

create table Engineer

( ID int identity (1,1) primary key ,

EngineerName varchar (50) not null,

Department char (20),

DateOfJoining datetime,

salary int,

city varchar(10) UNIQUE,

shifts nvarchar (10)

)

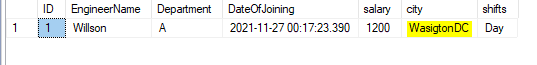
select \* from Engineer

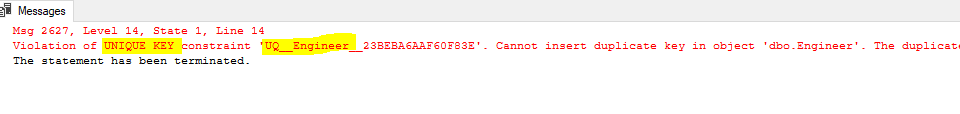
drop table Engineer

insert into Engineer(EngineerName,Department,DateOfJoining,salary,city,shifts)

values('Willson','DataEngineer',getdate(),1200,'WasigtonDC','Day')

('Jack','PBEngineer',getdate(),4700,'WasigtonDC','Night')



 ALTER TABLE Engineer

add constraint UK\_Engineer\_salary UNIQUE(salary)  


insert into Engineer(EngineerName,Department,DateOfJoining,salary,city,shifts)

values ('Willson','A',getdate(),1200,'WasigtonDC','Day'),

('Jack','PBEngineer',getdate(),1200,'London','Night')



ALTER TABLE Engineer

DROP constraint UK\_Engineer\_salary

**STORE PROCEDURE**

CREATE TABLE Tourist

(

CustomerID int IDENTITY(10,5)primary key,

[Name] VARCHAR(50),

City VARCHAR (50),

Department VARCHAR (10),

MobileNo Int

)

INSERT INTO Tourist(Name,City,Department, MobileNo) VALUES('Jack','Pune','HR',2131242234)

INSERT INTO Tourist(Name,City,Department,MobileNo) VALUES('Stive','Mumbai','Analyst',2434234)

INSERT INTO Tourist(Name,City,Department,MobileNo) VALUES('Welltor','DC','Buss',231323)

INSERT INTO Tourist(Name,City,Department,MobileNo) VALUES('CEF','EWE','DWD',231323)

INSERT INTO Tourist(Name,City,Department, MobileNo) VALUES('Jack','Pune','HR',2131242234)

INSERT INTO Tourist(Name,City,Department, MobileNo) VALUES('Jack','Wasington','HR',2131242234)

delete Tourist

TRUNCATE table Tourist

drop table Tourist

--Withoud parameter

CREATE PROCEDURE SPNameMobilenew

AS

BEGIN

SELECT [Name],City,mobileno from Tourist where city <> 'ewe' and CustomerId > 10

END

execute SPNameMobilenew

select \* from Tourist

--with parameter

CREATE PROCEDURE SPDepartment

@department varchar(30)

AS

BEGIN

SELECT [Name],city FROM tourist where department = @department

END

execute SPDepartment 'HR'

--ALTER STORE PROCEDURE

ALTER PROCEDURE SPDepartment

@department varchar(30),

@city varchar (10)

AS

BEGIN

SELECT [Name],city FROM tourist where department = @department AND city= @city

END

--EXECUTE STORE PROCEDURE

execute SPDepartment 'HR','Wasington'

execute SPDepartment @city='Wasington',@department='HR'

--RENAME STORE PROCEDURE

sp\_rename 'SPDepartment','SPDepartmentRenamed'

--VIEWING DATA FROM STORE PROCEDURE

sp\_helptext 'SPDepartment'

--STORE PROCEDURE ENCRYPTION

ALTER PROCEDURE SPDepartment

@department varchar(30),

@city varchar (10)

WITH ENCRYPTION

AS

BEGIN

SELECT [Name],city FROM tourist where department = @department AND city= @city

END

--DROP PROCEDURE

DROP PROCEDURE SPDepartment

--STORE PROCEDURE WITH OUTPUT VALUE

CREATE PROCEDURE EmployeData

@City varchar(10),

@AdditionOfCustomerId INT output

AS

BEGIN

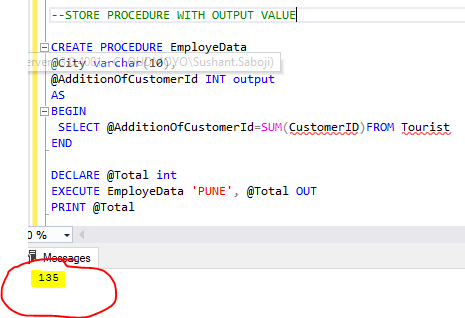
SELECT @AdditionOfCustomerId=SUM(CustomerID)FROM Tourist

END

DECLARE @Total int

EXECUTE EmployeData 'PUNE', @Total OUT

PRINT @Total



--STORE PROCEDURE WITH RETURN VALUE

CREATE PROCEDURE EmploeDataReturnVal

@city varchar(30)

AS

BEGIN

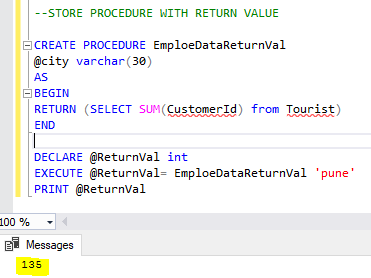
RETURN (SELECT SUM(CustomerId) from Tourist)

END

DECLARE @ReturnVal int

EXECUTE @ReturnVal= EmploeDataReturnVal 'pune'

PRINT @ReturnVal



-- STORE PROCEDURE WITH OPTIANL PARAMETERS,

CREATE PROCEDURE EmpleOptionalParameter

@Name Nvarchar(40)=NULL,

@City Nvarchar(40) =null,

@department Nvarchar(30) =null,

@Mobileno int = null

AS

BEGIN

SELECT \* FROM Tourist

where

(Name like @Name+'%' or @Name is null) AND

(City=@City or @City is null) AND

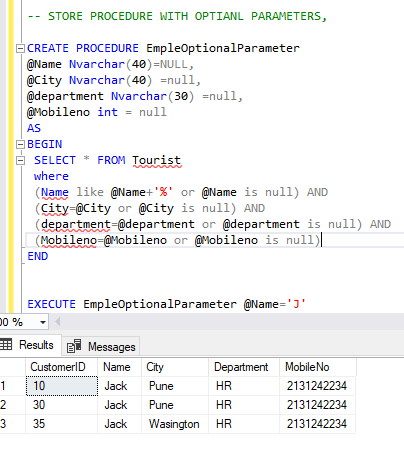
(department=@department or @department is null) AND

(Mobileno=@Mobileno or @Mobileno is null)

END

EXECUTE EmpleOptionalParameter @Name='J'

EXECUTE EmpleOptionalParameter @Name='St'



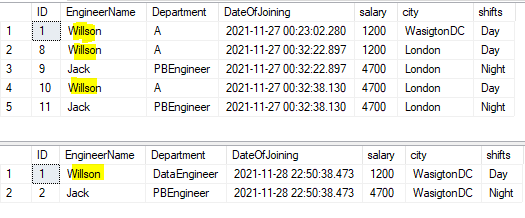
**OUTPUT VALUE🡪** Use all type of data Type. Get multiple value data.

**Return Value 🡪** Use only for **“Int”** data Type. Get only singe value data.

**UNION:**

select \* from Engineer

select \* from Engineer1

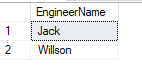


--UNION

select EngineerName from Engineer

UNION

select EngineerName from Engineer1

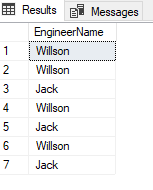


--UNION ALL

select EngineerName from Engineer

UNION ALL

select EngineerName from Engineer1



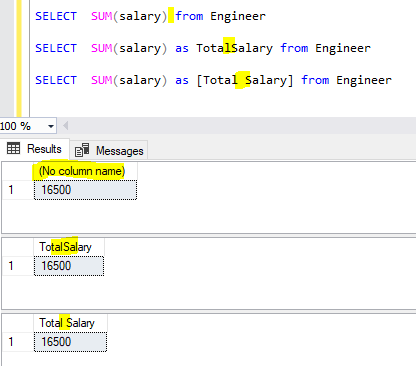
**Aliases Name**

select \* from Engineer

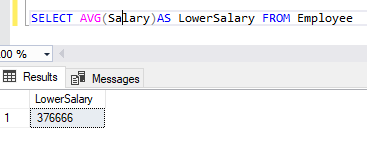
SELECT SUM(salary) from Engineer

SELECT SUM(salary) as TotalSalary from Engineer

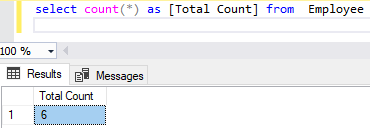
SELECT SUM(salary) as [Total Salary] from Engineer



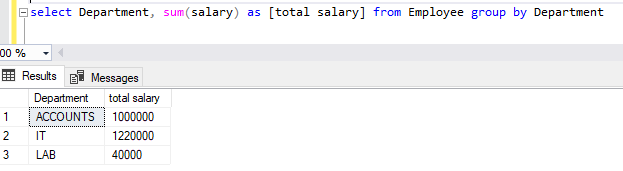
SELECT AVG(Salary)AS LowerSalary FROM Employee

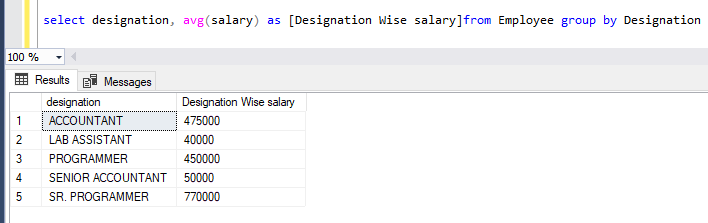


select count(\*) as [Total Count] from Employee

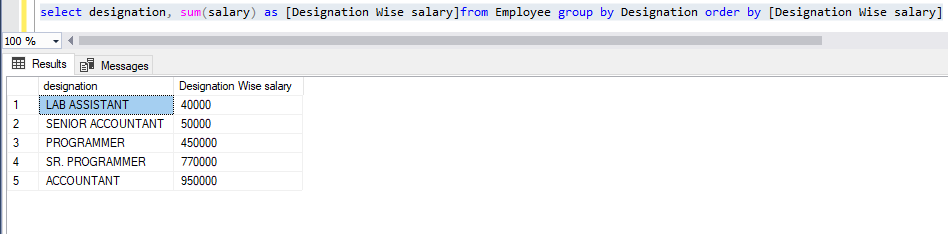
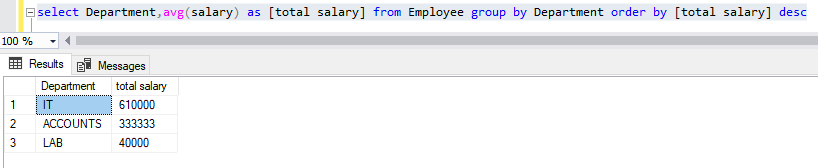


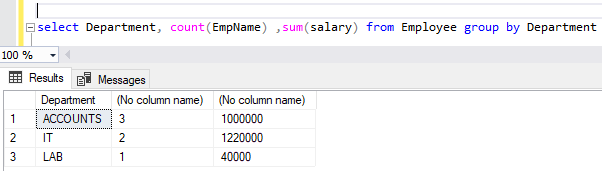
**GROUP BY:**

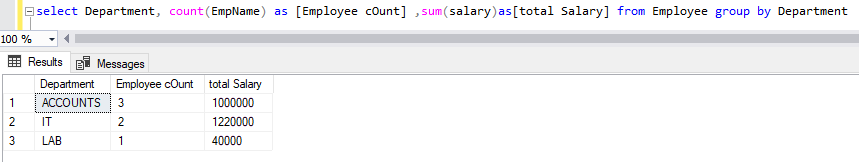
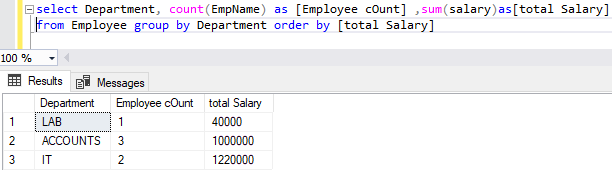
select Department, sum(salary) as [total salary] from Employee group by Department  


select designation, avg(salary) as [Designation Wise salary]from Employee group by Designation  


select designation, sum(salary) as [Designation Wise salary]from Employee group by Designation order by [Designation Wise salary]

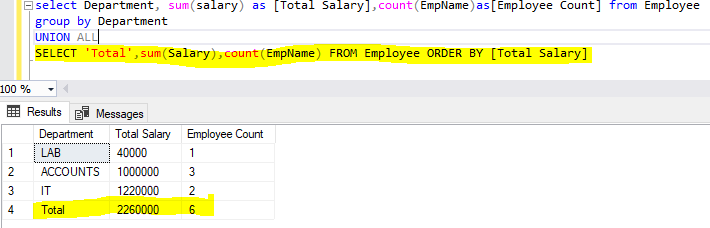
  
select Department,avg(salary) as [total salary] from Employee group by Department order by [total salary] desc  


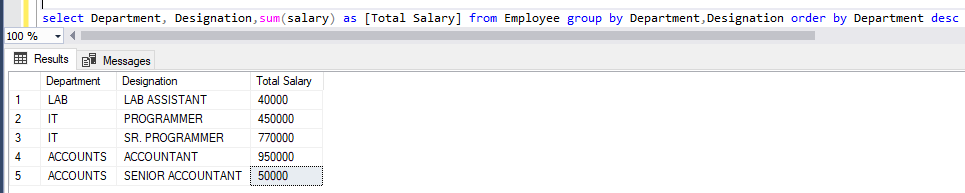
select Department, count(EmpName) ,sum(salary) from Employee group by Department  


select Department, count(EmpName) as [Employee count] ,sum(salary)as[total Salary] from Employee group by Department  
  
select Department, count(EmpName) as [Employee cOunt] ,sum(salary)as[total Salary] from Employee group by Department order by [total Salary]  
  
select Department, sum(salary) as [Total Salary],count(EmpName)as[Employee Count] from Employee

group by Department

UNION ALL

SELECT 'Total',sum(Salary),count(EmpName) FROM Employee ORDER BY [Total Salary]  
  


select Department, Designation,sum(salary) as [Total Salary] from Employee group by Department,Designation order by Department desc  


-- 3rd Highst Salary

SELECT TOP 1 salary

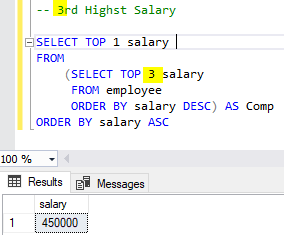
FROM

(SELECT TOP 3 salary

FROM employee

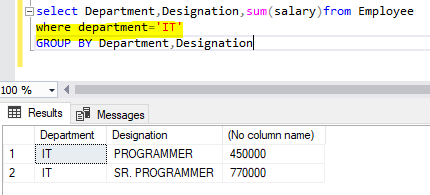
ORDER BY salary DESC) AS Comp

ORDER BY salary ASC

  
 **Having:**select Department,Designation,sum(salary)from Employee

where department='IT'

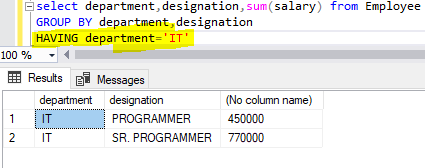
GROUP BY Department,Designation



select department,designation,sum(salary) from Employee

GROUP BY department,designation

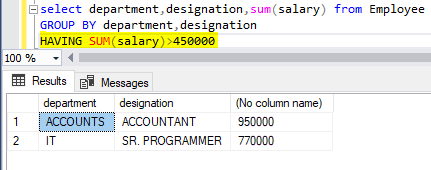
HAVING department='IT'



select department,designation,sum(salary) from Employee

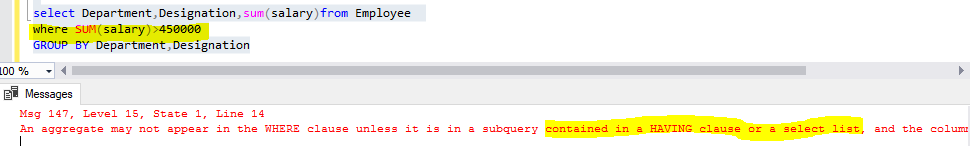
GROUP BY department,designation

HAVING SUM(salary)>450000



select Department,Designation,sum(salary)from Employee

where SUM(salary)>450000

GROUP BY Department,Designation  


**CHECK CONSTRAINT**

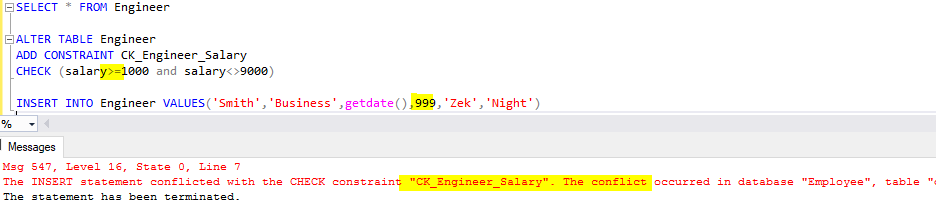
--USING CONSTRAINT NAME

ALTER TABLE Engineer

ADD CONSTRAINT CK\_Engineer\_Salary

CHECK (salary>=1000 and salary<>9000)

INSERT INTO Engineer VALUES('Smith','Business',getdate(),999,'Zek','Night')



CREATE TABLE MEMBER

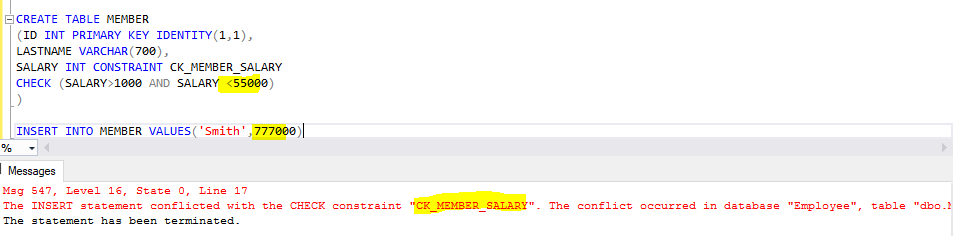
(ID INT PRIMARY KEY IDENTITY(1,1),

LASTNAME VARCHAR(700),

SALARY INT CONSTRAINT CK\_MEMBER\_SALARY

CHECK (SALARY>1000 AND SALARY <55000)

)

INSERT INTO MEMBER VALUES('Smith',777000)  
  


--WITHOUT USING CONSTRAINT NAME

CREATE TABLE MEMBER2

(ID INT PRIMARY KEY IDENTITY(1,1),

LASTNAME VARCHAR(700),

SALARY INT CHECK (SALARY>1000 AND SALARY <55000)

)

**FOREIGN KEY:**insert into SCHOOL values

('school of US','WashingtonDC'),

('High pleter school','Tokiyo'),

('Sant Jevir school','Lomdon'),

('Jkesar Academy','Filadelfiya'),

('UAE disterlocs','Dubai'),

('Jakcstons school','Swzerland')

insert into STUDENT values

('Smith',4),

('Jack',12),

('Wilson',10),

('Dacktes',2),

('Jonson',7),

('Riki',5),

('Adward',9),

('Samgron',9,5),

('Jatsols',11,6)

select \* from SCHOOL

select \* from STUDENT

ALTER TABLE STUDENT

ADD CONSTRAINT FK\_STUDENT\_SchoolID

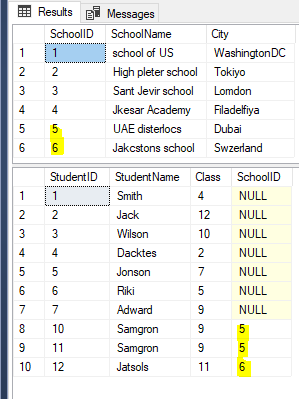
FOREIGN KEY (SchoolID)REFERENCES SCHOOL(SchoolID)

ALTER TABLE STUDENT

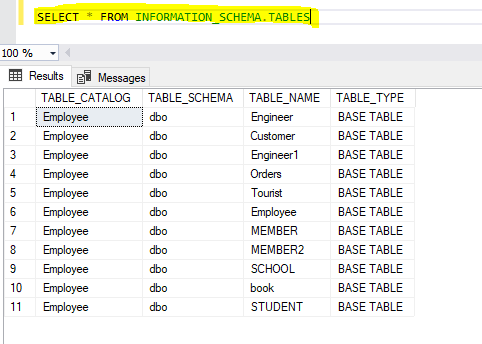
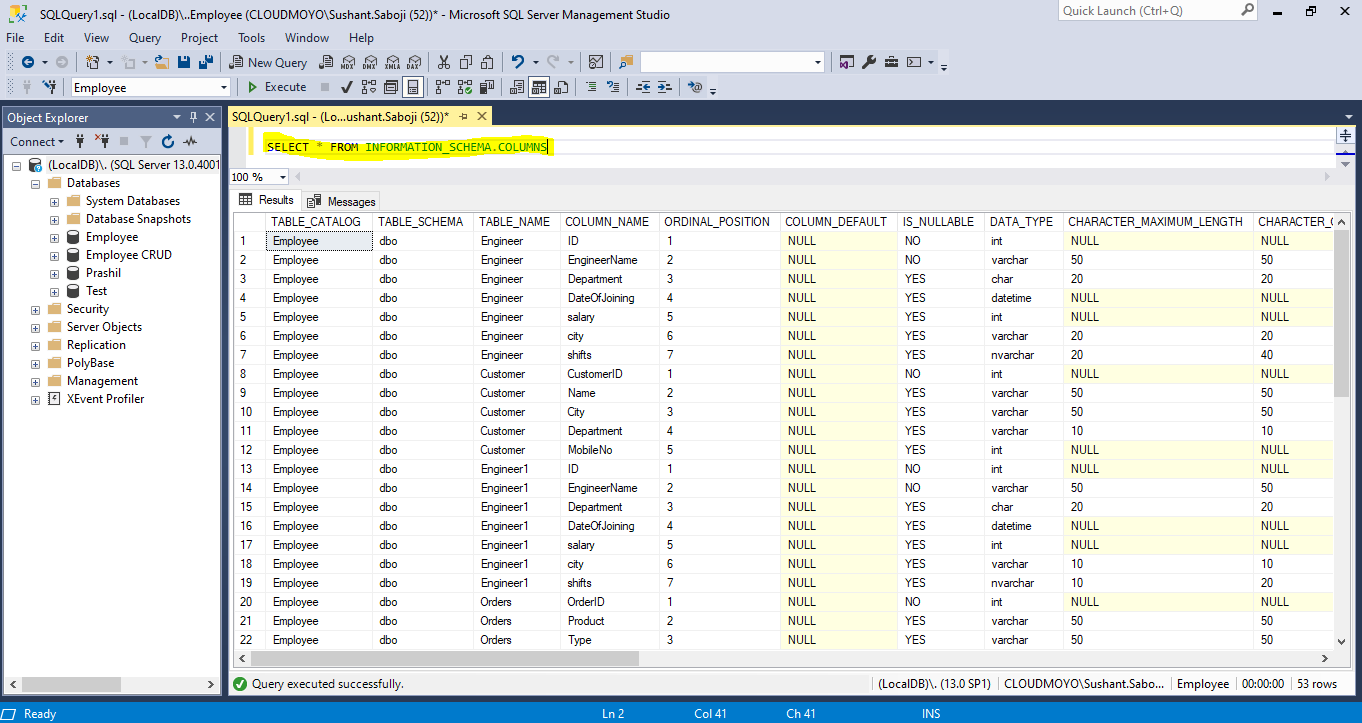
DROP CONSTRAINT FK\_STUDENT\_SchoolID

ALTER TABLE STUDENT

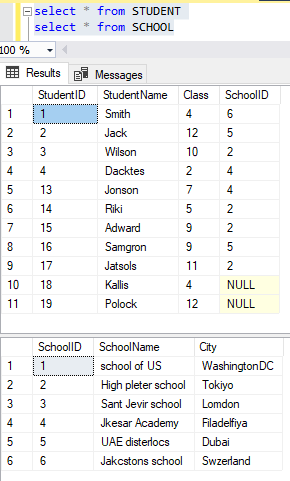
DROP COLUMN SchoolDetails



**ALL TABLE INFO:**

SELECT \* FROM INFORMATION\_SCHEMA.TABLES  
  
  
SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS  
  


**Joins:**  
select \* from STUDENT

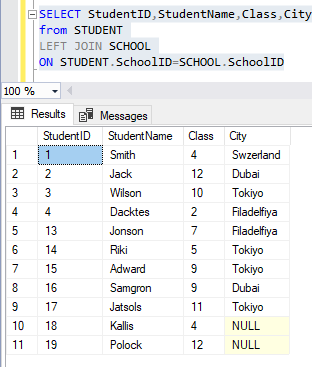
select \* from SCHOOL  


SELECT StudentID,StudentName,Class,City

from STUDENT

LEFT JOIN SCHOOL

ON STUDENT.SchoolID=SCHOOL.SchoolID

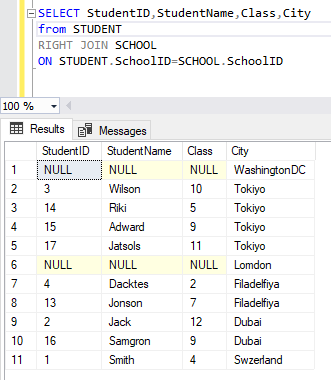


SELECT StudentID,StudentName,Class,City

from STUDENT

RIGHT JOIN SCHOOL

ON STUDENT.SchoolID=SCHOOL.SchoolID

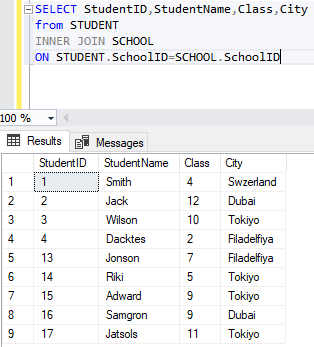


SELECT StudentID,StudentName,Class,City

from STUDENT

INNER JOIN SCHOOL

ON STUDENT.SchoolID=SCHOOL.SchoolID

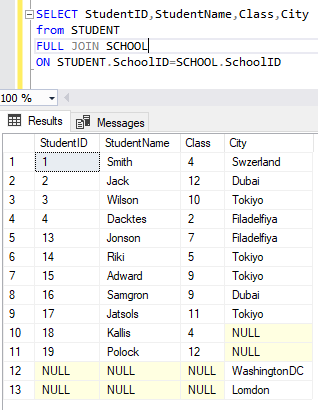


SELECT StudentID,StudentName,Class,City

from STUDENT

FULL JOIN SCHOOL

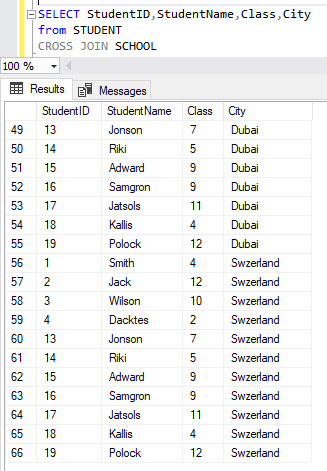
ON STUDENT.SchoolID=SCHOOL.SchoolID



SELECT StudentID,StudentName,Class,City

from STUDENT

CROSS JOIN SCHOOL



SELF JOIN:

select \* from Engineer

select A.ID as EngineerID, A.EngineerName as Names,

B.ID as bossId, B.EngineerName as BossName

from Engineer A, Engineer B

where A.city=B.city

Join on 3 table:

select \* from information\_schema.tables

select \* from SCHOOL

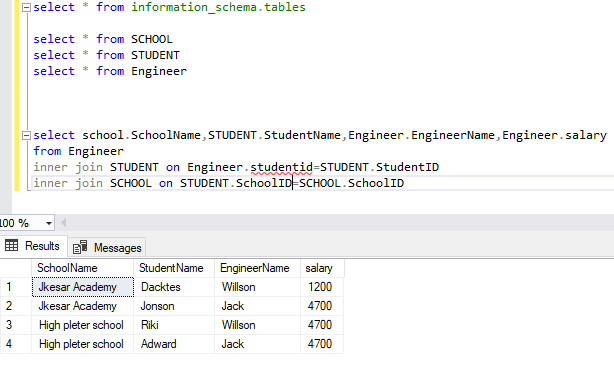
select \* from STUDENT

select \* from Engineer

select school.SchoolName,STUDENT.StudentName,Engineer.EngineerName,Engineer.salary

from Engineer

inner join STUDENT on Engineer.studentid=STUDENT.StudentID

inner join SCHOOL on STUDENT.SchoolID=SCHOOL.SchoolID 

select school.SchoolName,STUDENT.StudentName,Engineer.EngineerName,Engineer.salary

from Engineer

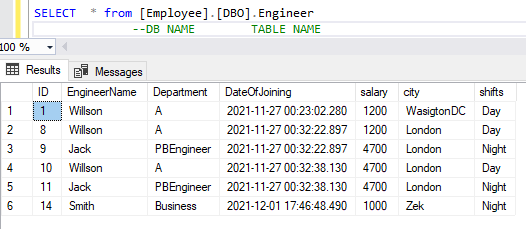
inner join STUDENT on Engineer.studentid=STUDENT.StudentID

inner join SCHOOL on STUDENT.SchoolID=SCHOOL.SchoolID

**TRICKS:**

SELECT \* from [Employee].[DBO].Engineer

--DB NAME TABLE NAME



**SHORT CUT:  
Alt+F1 🡪** Table details.

PROCEDURE 🡪 PROC  
execute🡪 EXEC

SELECT \* from [Employee].[DBO].Engineer

--DB NAME TABLE NAME

Questions:

1. which is summarisation function?

* Avg(): It returns the average of the values in the expression. ...
* Count: It returns the number of values in an expression. ...
* Min(): It returns the lowest value in the expression. ...
* Max(): It returns the highest value in the expression.

1. what is difference between ‘nvarchar’, ‘varchar’ and ‘char’?

**char** and **nchar** are **fixed-length** which will **reserve storage space** for number of characters you specify even if you don't use up all that space.

* **varchar** and **nvarchar** are **variable-length** which will only use up spaces for the characters you store. It **will not reserve storage like char or nchar**.

1. what is difference between DELTE Table and TRUNCATE Table?
2. What Is Indexing?  
   Indexes are **used to retrieve data from the database more quickly than otherwise**. The users cannot see the indexes, they are just used to speed up searches/queries. Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update).  
     
   eg:  
   CREATE INDEX idx\_pname  
   ON Persons (LastName, FirstName);
3. How to check all tables from Database?

* Answer: SELECT \* FROM INFORMATION\_SCHEMA.TABLES 🡪 for Tables
* SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS 🡪 For columans

1. what is trigger in sql?

A database trigger is **a stored program which is automatically fired or executed when some events occur**.

7.what is cursor in sql?

A SQL Server cursor is **a set of T-SQL logic to loop over a predetermined number of rows at a time**. The purpose for the cursor may be to update one row at a time or perform an administrative process such as SQL Server database backups in a sequential manner

DECLARE @name VARCHAR(50) -- database name

DECLARE @path VARCHAR(256) -- path for backup files

DECLARE @fileName VARCHAR(256) -- filename for backup

DECLARE @fileDate VARCHAR(20) -- used for file name

SET @path = 'C:\Backup\'

SELECT @fileDate = CONVERT(VARCHAR(20),GETDATE(),112)

DECLARE db\_cursor CURSOR FOR

SELECT name

FROM MASTER.dbo.sysdatabases

WHERE name NOT IN ('master','model','msdb','tempdb')

OPEN db\_cursor

FETCH NEXT FROM db\_cursor INTO @name

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @fileName = @path + @name + '\_' + @fileDate + '.BAK'

BACKUP DATABASE @name TO DISK = @fileName

FETCH NEXT FROM db\_cursor INTO @name

END

CLOSE db\_cursor

DEALLOCATE db\_cursor

8.what is function in sql?

SQL Server Functions are **useful objects** in SQL Server databases. A SQL Server function is a code snippet that can be executed on a SQL Server. ... Functions can be used anywhere in SQL, like AVG, COUNT, SUM, MIN, DATE and so on with select statements. Functions compile every time. Functions must return a value or result.

9.Difference between function and store procedure in sql?

1. The function must return a value but in**Stored Procedure** it is optional. Even a procedure can return zero or n values.
2. Functions can have only input parameters for it whereas Procedures can have input or output parameters.
3. Functions can be called from Procedure whereas Procedures cannot be called from a Function.
4. The procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
5. Procedures cannot be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
6. Stored Procedures cannot be used in the [SQL](https://en.wikipedia.org/wiki/SQL) statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
7. Functions that return tables can be treated as another row set. This can be used in JOINs with other tables.
8. Inline Function can be though of as views that take parameters and can be used in JOINs and other Row set operations.
9. An exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
10. We can use Transactions in Procedure whereas we can't use Transactions in Function.

10. what is replication in sql server?

Bidirectional transactional replication is a specific transactional replication topology that **allows two servers to exchange changes with each other**: each server publishes data and then subscribes to a publication with the same data from the other server.

11. when 2 users booking same seat at a time for traveling. What happen and How to manage system on sql side?