Create a SP which accepts deptno and display all emp details who belong to that deptno.

CREATE PROCEDURE GetAllEmpDetails(@depno INT)

as

Begin

SELECT empno,ename,job,mgr,hiredate,sal,comm,deptno from emp where deptno=@depno

End

exec GetAllEmpDetails 20

Create a SP which accepts empno and display his/her annual salary.

CREATE PROCEDURE GetEmpAnnualSal(@depno INT)

as

Begin

SELECT sal\*12 from emp where deptno=@depno order by sal

End

create a SP which accepts deptno and display how many employees working in that deptno

CREATE PROCEDURE GetEmpCount(@depno INT)

as

Begin

SELECT Count(ename) as 'Number Of Enployees' from emp where deptno=@depno

End

exec GetEmpCount 20

create a SP which accepts deptno and display what is the min and max sal taken in that deptno

CREATE PROCEDURE MinMaxSal(@depno INT)

as

Begin

SELECT MIN(sal) as 'Minimum Salary', Max(sal) as 'Maximum Salary' from emp Where deptno=@depno group by deptno

End

exec MinMaxSal 20

create function which accepts empno and return his/her annual sal

CREATE Function GetAnnualsal(@empno int)

returns INT

Begin

Declare @Annual INT

SELECT @Annual=sal from emp WHERE empno=@empno

return @Annual\*12

END;

Select dbo.GetAnnualsal(7369)

create a function which accepts deptno and returns no of employee available in that deptno

Create Function GetNumOfEmp(@deptno int)

returns INT

Begin

Declare @CountEmp INT

SELECT @CountEmp=Count(ename) from emp WHERE deptno=@deptno

return @CountEmp

END;

select dbo.GetNumOfEmp(20)

Create a function which accepts mgrid and display all emps who report to that person.

Create function GetEmpByMgrId(@mgrid int)

returns TABLE

As

RETURN

( Select ename from emp where mgr=@mgrid

)

Select \* from GetEmpByMgrId(7839)

Create a function which accepts num as a salary and display all emps who get sal more than given sal

Create function MorethnSal(@sal int)

returns TABLE

As

RETURN

( select ename, sal from emp where sal>@sal

)

Select \* from MorethnSal(1600)

**NUMERIC FUNCTIONS**

SELECT CEILING($123.45), CEILING($-123.45), CEILING($0.0);

GO

SELECT FLOOR(123.45), FLOOR(-123.45), FLOOR($123.45);

GO

SELECT RAND(5)

GO

SELECT ROUND(5.26445,2)

GO

SELECT SQRT(85)

GO

**STRING FUNCTIONS**

SELECT

ASCII('S') as 'S',

ASCII('W') as 'W',

ASCII('A') as 'A',

ASCII('r') as 'r'

GO

SELECT

CHAR(83) as '83',

CHAR(87) as '87',

CHAR(65) as '65'

SELECT CONCAT('HELLO',' ','WORLD') as 'Greeting'

SELECT CONCAT('MY',' ','Name',' ','Is',' ', 'Swaroop') as 'Name'

SELECT LEFT(ename,3)

from emp

SELECT RIGHT(ename,3)

from emp

SELECT LEN(ename) as 'length'

from emp

SELECT LOWER(ename) as 'lower name'

from emp

SELECT UPPER('swaroop') as 'Name'

SELECT LTRIM(' My name is Swaroop');

SELECT RTRIM(' My name is Swaroop ');

SELECT REPLACE('APPLEPPOPPPE','PP','BB');

SELECT REPLICATE('S',4)

SELECT SUBSTRING(ename,1,3) as 'SUBSTRING'

from emp

**DATE TIME FUNCTIONS**

SELECT GETDATE() as 'DATE'

SELECT DATENAME(year,hiredate) as 'YEAR'

from emp

WHY GO IS USED IN TSQL?

GO is not a SQL keyword.

It's a batch separator used by client tools (like SSMS) to break the entire script up into batches

Go means, whatever SQL statements are written before it and after any earlier GO, will go to SQL server for processing.

What are OUT Parameters?

A SQL Server stored procedure that you can call is one that returns one or more OUT parameters, which are parameters that the stored procedure uses to return data back to the calling application. The Microsoft JDBC Driver for SQL Server provides the [SQL Server Callable Statement](https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/sqlservercallablestatement-class?view=sql-server-ver15) class, which you can use to call this kind of stored procedure and process the data that it returns.