SQL SERVER

- SQL stands for Structured Query Language
- SEQUEL SERVER(structured English Query language)
- Database: is a collection of information
- It is just to show the data or represent the data
- Data: raw facts or any user input---bjghkewhgkhewkgkjsadbgjbjgbksejh
- information: processed data or organised data or meaningful data
- information will be stored in database in the form of tables.
- Table: rows and columns
- In sql server table is also called as entity
- row is also called as record or tuple
- column is also called as field or attribute

Rules to create table:

---

1. to create table at least one column is required

- id name address salary ---columns
- table can have max columns(1024)

2 .In sql server table name or column name should not start with digits.

3. Table name or column name should not be two parts

ex: emp deratils---wrong

ex: emp_details--- right

in general 3 types of databases are there

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

1.local databases----ms-access,dbase,foxpro

2.remote databases---sql server,oracle

3.online or internet databases---OLAP(online analytical processing)
OLTP(online transaction processing)

to manage any type of database we need a software
i.e DBMS(Database management system)

to manage local databases we need DBMS
to manage Remote databases we need RDBMS(relational database management system)
to manage online databases we need ORDBMS(object relational DBMS)

what is sql serever:
---
sql server is a relational database management system which is used to manage and access the data from database.

why sql server: is used to create or design databases whic are use to store information.

to work with sql server we use software sql server-2014

when we install sql server 2014 s/w we will get
SSMS(sql server management studio)

ssms is not database server it is just client tool or editor tool
ssms is editor which is used to write commands or queries

Types of services:

1.database engine
2.analysis service
3.reporting service
4.integration service
5.sql server compact edition or mobile edition

database engine: is used to manage and access the data from database
analysis service: this service can be used for data warehousing concepts

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

reporting service: can be used to generate reports from database
integration service: it can be used to convert from one relational database tables to another.
sql server compact edition: it can be used for mobile application development

SQL commandset:

1.DDL(Data definition language)----Create,Alter,Drop

2.DML(Data Manipulation language)----insert,delete,update

3.DCL(Data control language)-----Grant ,Revoke

4.TCL(Transaction control language)----Rollback,commit,savepoint

5.DQL(Data Query language)----select

Create:
---
is used to create table,database,function,procedures ,constraints etc

alter:
---
is used to modify the table structure like add column,drop column or change column datatype or its size

update:
---
is used to modify the table data or content or record

drop:
---
is used to drop the database or table or any thing permanently

delete:
---
is used to delete the data from the table

insert:

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

is used to insert records into table or to store the data.

select:
---
is used to read the data from table

in sql server by default 4 types of system databases

1.master
2.model
3.msdb
4.tempdb

master is a main database which is used to manage user and administartive tasks

model is just template for other databases

msdb is a microsoft database which is used to store job history
like user execution statements history.

tempdb y is a temporary database which is used to store temporary tables, files,procedures..

---create database databasename
create database chaitanya

--to move from one database to another
--use targetdatabasename
use chaitanya

---when we create database, internally two files gets create
--1.primary file---.MDF(master data file)
--2.secondary file or transaction log file---.LDF(log data file)

---physical location of these files

--C:\Program Files (x86)\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA

---to drop database
---drop database databasename

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
drop database chaitanya

--error:
--Cannot drop database "chaitanya" because it is currently in use.

--move to other database then try to drop

drop database chaitanya

---when we drop database internally primary and secondary files also
---drop

---create table tablename(column1 datatype,column2 datatype(size))

create table emp(id int,name varchar(20))

---to get data from table

select*from emp

-- * indicates all columns

select id from emp

---to store the data
insert into emp values(1,'sai')
insert into emp values(2,'ram')

select*from emp

---to modify the record

update emp set name='manoj' where id=1


--to delete one record

delete from emp where id=1

---to delete all records
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
delete from emp

select*from emp

---to add column

alter table emp
add address varchar(20)

select*from emp

---to drop column

alter table emp
drop column address


---to drop table
---drop table tablename
drop table emp1
```

rollback and commit,savepoint:

rollback is used to cancel the transaction
commit is used to make the transaction as permanent
savepoint is used to save the transaction

```
select*From emp

begin transaction
delete from emp where id=1

---to get back deleted record
rollback

---after doing transaction if u want to make the transaction as
--permanent the we use commit

commit

select*from emp
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
begin transaction
save transaction t1
delete from emp where id=2

save transaction t2
delete from emp where id=3

save transaction t3
delete from emp where id=4

select*from emp

rollback transaction t3
rollback transaction t2
rollback transaction t1

--to get complete table information
---sp_help tablename
sp_help emp

--to get all table list from current database

select*from sysobjects where xtype='u'

---to rename the table
--sp_rename 'oldtablename','newtablename'
sp_rename 'emp','emp1'
--select*from emp --Invalid object name 'emp'.
select*from emp1

----to rename the column
--sp_rename 'tablename.oldcolumnname','newcolumnname'
sp_rename 'emp1.name','empname'

select*from emp1

---to change column datatype size

alter table emp1
alter column empname varchar(10)

---when we try to change column datatype or its size through design
```

--we will get error i.e saving chnages not permitted
---to avoid this...goto tools menu---click on options---expand designers ---select table and database designer---deactivate the checkbox prevent saving changes---ok

--now we can change aolumn datatype and its size through design also.

Logical Operators:
---

and----both conditions should to satisfy then only result will be true
else result is false

or-- any one of the condition is satisfy then result will be true
else false

not--- reverse i.e if condition is true then result is false
if condition is false then result is true

between--- it is used to get records based on between condition

```
select*from empdetails

select*from empdetails where
designation='manager' and age between 30 and 45

select*from empdetails where
designation='manager' or age between 30 and 45

select*from empdetails where address!='hyd'

select*from empdetails where address<>'hyd'

select*from empdetails where designation<>'manager'

select*from empdetails where salary between 50000 and 85000
```

in--- is used to pass multiple values in where condition

```
select*from empdetails where id=1 or id=2 or id=3
```

select*from empdetails where id in(1,2,3,4,5)

is --- is used to compare null values and to get records

select*From empdetails where salary is null

---distinct: is used to get unique values from table

select distinct designation from empdetails
select distinct address from empdetails

---order by: is used to display the records in order
--by default order by will display records in ascending order
--but to display descending order then we use desc keyword

select*from empdetails order by salary
select*from empdetails order by salary desc

select*from empdetails order by name desc

select*from empdetails order by 5

select*from empdetails order by 3

---group by:is used to group the records then get result

---ex: to get employee count based on their designation

select designation,count(*) from empdetails group by designation

select designation,count =count(*) from empdetails group by designation

select designation,count(*) as total from empdetails group by designation

---top: is used to get top most records from the tables based on condition

select top 2* from empdetails

select top 2*from empdetails order by salary desc

---like operators: these are used to get the records based on provided
---format

---to work with like operators we use two things

-- % --- represent group of characters

-- _ --- represent single character

---to get whose name starts with letter 'r'

```
select*from empdetails where name like 'r%'

select*from empdetails where name not like 'r%'

select*from empdetails where name  like '_a%'

select*from empdetails where name  like '__n%'

select*From empdetails where name like '%n'
```

----set operators are used to get result of two select statements

---union
---unionall
---intersection
---except

```
create table tbldrinks(id int,drink varchar(10))

create table tbldrinks1(id int,drink varchar(10))

insert into tbldrinks values(1,'pepsi')
insert into tbldrinks values(2,'slice')
insert into tbldrinks values(3,'thumsup')


insert into tbldrinks1 values(1,'pepsi')
insert into tbldrinks1 values(2,'maaza')
```

insert into tbldrinks1 values(3,'sprite')

---union: unique values from both tables

```
select*from tbldrinks
union
select*from tbldrinks1
```

---unionall: all records from both tables

```
select*from tbldrinks
union all
select*from tbldrinks1
```

--intersect: common values present in both tables

```
select*from tbldrinks
intersect
select*from tbldrinks1
```

--except: return values from first table which are not present in
----second table

```
select*From tbldrinks
except
select*from tbldrinks1
```

---identity: is used to generate automatic column values
---used to generate sequence of values

---syntax: identity(seedvalue,incrementvalue)
---ex: identity(1,1)---default

--when we have identity column in table for that column we need not to
--supply values ,internally identity will generate values

---note:identity we can apply only integer datatype columns

---how to check whether the table is having identity or not

sp_help emp1

```
---to create table with identity column

create table emp(id int identity(1,1),name varchar(20))

sp_help emp

insert into emp values('sai')
insert into emp values('ram')
insert into emp values('mohan')

select*from emp

--try to delete one record

delete from emp where id=1

insert into emp values('shashi')

select*from emp

---if u want allot id=1 to someone

insert into emp values(1,'tarun')

--Error:An explicit value for the identity column in table 'emp'
--can only be specified when a
---column list is used and IDENTITY_INSERT is ON.

set identity_insert emp on

insert into emp(id,name) values(1,'tarun')

select*from emp

set identity_insert emp off

---try delete all records from table

delete from emp

select*from emp
```

```
---now table is empty
---try to insert few more records then observ id numbers
insert into emp values('sai')
insert into emp values('ram')
insert into emp values('mohan')

select*From emp

---if you want accept id values from begining i.e from 1
---to reset identity

delete from emp

dbcc checkident(emp,reseed,0)

---dbcc-database consistence check

insert into emp values('sai')
insert into emp values('ram')
insert into emp values('mohan')

select*from emp


--difference between delete and truncate

---using delete command we can delete one record with where condition
---we can delete all records from table
---delete command can support where condition

---using truncate we can only delete all records from table
---we cant delete one by one record
---truncate command can not support where condition

--when we have a table with identity column values
---when we apply delete command ,all records will be delete
--but identity will not reset
---when we apply truncate command ,all records will be delete
-- identity will be reset

select*from emp
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

delete from emp

---try to insery few records

insert into emp values('sai')
insert into emp values('ram')
insert into emp values('mohan')

---check id numbers from where it will start

select*From emp

---id numbers will start from 4 so that identity not reset

---try to apply truncate

---truncate table tablename

truncate table emp

---now all records are deleted

----try to insert few records then abserv id numbers

insert into emp values('sai')
insert into emp values('ram')
insert into emp values('mohan')

select*From emp

---id numbers will starts from 1 that means identity was reset.

---constraints: constraint means condition
---are used to avoid data duplication and repetation
---are used to make the data consistency

---types of constraints:
--1.unique
--2.not null
--3.primary key

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

--4.check
--5.default
--6.foreign key

---we can apply constraints in two differnt ways

--option1: applying constraint without name

--syntax:
---create table tablename(column1 datatype constraintname,
---column2 datatype(size))

--option2: applying constraint with name

--syntax:
---create table tablename(column1 datatype constraint constraintname
constrainttype,
---column2 datatype(size))

create table emp2(id int unique,name varchar(20))

sp_help emp2

---to get constraint details only
sp_helpconstraint emp2

create table emp3(id int constraint unq_emp3_id unique,
name varchar(20))

sp_helpconstraint emp3

---unique: will accept unique vaues only
---will not accept duplicate values
---will accept only one null value ,because one null value is treated as
--unique

select*from emp2

insert into emp2 values(1,'sai')---insert

insert into emp2 values(1,'sai')---not insert

insert into emp2 values(2,'sai')---insert

insert into emp2 values(null,'sai')---insert

insert into emp2 values(null,'sai')---not insert

select*from emp2

---to create constraint for existing table

sp_helpconstraint emp

alter table emp
add unique(id)

--to drop the constraint

alter table emp
drop constraint UQ__emp__3213E83E0A820218

sp_helpconstraint emp

1.unique
2.not null
3.primarykey
4.check
5.default
6.foreignkey

not null: will not accept any null values,but it will accept duplicate values

create table emp4(id int not null,name varchar(20))

insert into emp4 values(1,'sai') --insert
insert into emp4 values(1,'sai') --insert
insert into emp4 values(null,'sai') --not insert
insert into emp4 values(2,'sai') --insert

select*from emp4

primarykey:

---

it will accept unique values
it will not accept duplicate values
it will not accept null values

in sql server a table should have only one primary key
more than one primary in a table not allowed

```
create table emp5(id int primary key,name varchar(20))

sp_helpconstraint emp5

insert into emp5 values(1,'sai')---insert

insert into emp5 values(1,'sai')---not insert

insert into emp5 values(2,'sai')---insert

insert into emp5 values(null,'sai')---not insert
```

Note: when the table alreday having duplicate data,when we apply constraint like primary key or unique explicitly that cant apply.

check:
---

it is a special constraint which is used to apply some conditions on any column based on requirement

```
create table emp7
(
id int primary key,
name varchar(20),
age int check(age between 20 and 80)
)

sp_helpconstraint emp7

insert into emp7 values(1,'sai',89)--not insert

insert into emp7 values(1,'sai',19)--not insert
```

insert into emp7 values(1,'sai',35)-- insert

select*from emp7

Default:
---

syntax: default(value)
ex: default(1)

when we have a table with default constraint on perticular column
if u not supply that column value then it will take default value.

though the table is having default constraint when we supply column value, supplied value only consider.

create table emp8(id int default(1),name varchar(20))

insert into emp8 values(1,'sai')

insert into emp8 values(2,'sai')

insert into emp8 values(3,'sai')

insert into emp8 values('mohan')
--Error--Column name or number of supplied values does not match table definition.
insert into emp8(name) values('mohan')
insert into emp8(name) values('durga')

select*from emp8

foreign key:
---

*is used to maintain referential integrity between tables
*is used to maintain relationship with tables
*to work with forgeign key, one table should have primary key
another table should have foreign key
*the table which is having primary key is said to be parent table

*the table which is having foreignkey is said to be child table
*both tables should have atleast one common column name

```
---parent table
create table tblemp
(
id int primary key,
ename varchar(20)
)

--child table
create table tbldept
(
id int foreign key references tblemp(id) on delete cascade
on update cascade,
deptname varchar(20)
)

select*from tblemp
select*from tbldept

---try to insert one record in child table

insert into  tbldept values(1,'IT')

---record cant be insert because if the parent table dont have id=1
--then we cant insert id=1 in child table

--try to insert id=1 in parent table

insert into tblemp values(1,'sai')
insert into tblemp values(2,'mohan')


insert into tbldept values(1,'IT')
insert into tbldept values(2,'HR')

select*from tblemp
select*from tbldept
```

--on delete cascade on update cascade means when we update or delete record
--in parenet table automatically record gets updated or deleted in child table

update tblemp set id=111 where id=1

delete from tblemp where id=2

select*from tblemp
select*from tbldept

sub query: a query which is having one more query within it is known as sub query

or

a select statement which is having one more select statement within it is known as sub query

always sub query will execute first then after main query will be execute.

Ex: main query(sub query)

select*from empdetails

select max(salary) from empdetails

--to get emp record who is having max salary
select name from empdetails where
salary=(select max(salary) from empdetails)

select * from empdetails where
salary=(select max(salary) from empdetails)

--to get emp record who is having second highest salary

select max(salary) from empdetails where
salary<(select max(salary) from empdetails)

---to get emp records whose salary more than avg salary of all employees

select avg(salary) from empdetails

select*from empdetails
where salary>(select avg(salary) from empdetails)

Nested query:
---

a query which is having more than one query within it is called nested query

or

a subquery which is having more than one query within it is called nested query

always nested query execute first then after subquery then after main query

Ex: main query(sub query(netsed query))

select*from empdetails

--to get 2nd highest salary emp record
select name from empdetails where
salary=(select max(salary) from empdetails where
salary<(select max(salary) from empdetails))

select * from empdetails where
salary=(select max(salary) from empdetails where
salary<(select max(salary) from empdetails))

---to get 3rd highest salary

select max(salary) from empdetails
where salary<(select max(salary) from empdetails
where salary<(select max(salary) from empdetails))

---to get nth highest salary

select*from empdetails where
salary=(select min(salary) from empdetails
where salary in(select distinct top 5 salary from empdetails

order by salary desc))

---temporary tables:

---these tables are similar to permanent tables
---permanent tables gets create and stored permanently in database
---until we delete or drop them

---temporary tables gets create and stored temporarly in database
---these tables will be delete automatically when they are no longer
---use or when we close query window

---temporary tables will be stored in tempdb

---2 types of temp tables

--1.local temp tables
--2.global temp tables

--1.local temp tables:these table names prefix with single # symbol
--these tables can only be access within same query window
--multiple users can create local temp tables with same name
---local temp table names end with random number i.e unique for every
---table

--2.global temp tables:these table names prefix with double ## symbol
--these tables can be access in any other query window
---multiple users can not create global temp tables with same name
---global temp table names not end with random number

```
create table #tbltemp(id int,name varchar(10))

insert into #tbltemp values(1,'sai')

select*From #tbltemp

create table ##tbltemp(id int,name varchar(10))
```

```
insert into ##tbltemp values(1,'sai')

select*from ##tbltemp
```

Functions in sql:
---
function is a statement or group statements which is used to perform specific task.

1.single row functions
2.multi row functions

single row functions: these functions will process at a time single row and return result as single value

multi row functions:these functions will process at a time multiple rows and return result as single value

single row functions:

1.mathematical functions
2.string functions

multi row functions:

1.aggregate functions

```
--mathematical functions

--abs()--return absolute value
select abs(-34)
--square():return squre value
select SQUARE(4)
--sqrt():return sqrt of given value
select sqrt(16)
--power(base,exponent):
select POWER(3,2)
---floor():return  integer value,remove fractional part
```

```
select FLOOR(34.56)
---ceiling():return next integer value,remove fractional part
select ceiling (34.56)

---string functions:
-----------------

--len():return length of the string
select len('mohan')
--lower():convert from upper to lower
select lower('MOHAN')
--upper():convert from lower to upper
select upper('mohan')
---ltrim(): remove white spaces from left side of string
select ltrim('   durga')
---rtrim(): remove white spaces from right side of string
select RTRIM('durga   ')
select RTRIM('durga'+    +'soft')
--ASCII():
select ASCII('A')
select ascii('a')
--reverse():return reverse of the string
select reverse('SAI')
--space(value): used to provide required spaces between strings
select 'durga'+space(100)+'soft'

---aggregate functions:

--max(),min(),sum(),avg(),count()

select*from empdetails

select max(salary) from empdetails
select min(salary) from empdetails
select sum(salary) from empdetails
select avg(salary) from empdetails
select count(*) from empdetails

determinstc and non determnistic functions:
-------------------------------------------

deterministic functions:
```

---
these functions will give every time same result how many times we execute

ex: all aggregate functions are determinsitic functions
ex: sqrt(),square(),power()

non-deterministic functions:
---
these functions will give every time different result how many
times we execute

select getdate()
select CURRENT_TIMESTAMP

by default rand() is non detereministic function
but when we supply seed value in rand() then it will become deterministic

select rand()
select rand(1)

getdate():
---

this function is used to display current system date and time
in the following format

year month day hour min sec microsec

select getdate()

--day(date): it extract day from given date
select day(getdate())
select day('2019-02-23')

--month(date): it extract month from given date
select MONTH(getdate())

--year(date): it extract year from given date
select year(getdate())

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

--date_add(): is used to add no of days or months or years to a given date

---syntax: dateadd(datepart,number,date)

select DATEADD(day,3,getdate())

select DATEADD(month,3,getdate())

select dateadd(year,6,getdate())

--datediff(): is used to find difference between two dates in terms of
--days,months,years

---syntax: datediff(datepart,start_date,end_date)

select DATEDIFF(day,'2013-05-15','2019-11-29')
select DATEDIFF(month,'2013-05-15','2019-11-29')
select DATEDIFF(year,'2013-05-15','2019-11-29')

EOMONTH(end of the month):

- Introduced in SQL Server 2012
- Returns the last day of the month of the specified date

Syntax : EOMONTH (start_date [, month_to_add])

start_date : The date for which to return the last day of the month
month_to_add : Optional. Number of months to add to the start_date.
EOMONTH adds the specified number of months to start_date,
and then returns the last day of the month for the resulting date.

select EOMONTH(getdate())
select EOMONTH('2019-03-12')
select EOMONTH('2016-02-12')
select EOMONTH('2017-02-12')
select EOMONTH(getdate(),3)
select EOMONTH('2019-05-23',5)

DATEFROMPARTS function

- Introduced in SQL Server 2012
- Returns a date value for the specified year, month, and day
- The data type of all the 3 parameters (year, month, and day) is integer

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

- If invalid argument values are specified, the function returns an error
- If any of the arguments are NULL, the function returns null

Syntax : DATEFROMPARTS (year, month, day)

```
select DATEFROMPARTS(2019,2,12)

select DATEFROMPARTS(2019,13,23)--invalid month(13)

select DATEFROMPARTS(2019,9,31)---invalid day(31)

select DATEFROMPARTS(2019,2,null)
```

coalesce function:
---

this function is used to return first non null value from rows

```
create table tblcoalesce
(id int,fname varchar(10),mname varchar(10),lname varchar(10))

insert into tblcoalesce values(1,'sai',null,null)
insert into tblcoalesce values(2,null,'ram',null)
insert into tblcoalesce values(3,null,null,'krish')
insert into tblcoalesce values(4,'sai','ram','krish')

select*From tblcoalesce

select id,coalesce(fname,mname,lname) as name
from tblcoalesce
```

Cast and convert functions:

To convert one data type to another
CAST and CONVERT functions can be used.

Syntax of CAST and CONVERT functions from MSDN:

```
CAST ( expression AS data_type [ ( length ) ] )

CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
create table tblcastconvert
(
id int,
name varchar(10),
dateofbirth date
)

insert into tblcastconvert values(1,'sai','2002-9-23')
insert into tblcastconvert values(2,'mohan','1984-01-03')
insert into tblcastconvert values(3,'ram','2012-9-28')

select*from tblcastconvert

--CAST ( expression AS data_type [ ( length ) ] )
Select Id, Name, DateOfBirth,
 CAST(DateofBirth as nvarchar(10)) as ConvertedDOB
from tblcastconvert


---CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
Select Id, Name, DateOfBirth,
 Convert(nvarchar(15), DateOfBirth,106) as ConvertedDOB
from tblcastconvert
```

style dateformat
101 mm/dd/yyyy
102 yy.mm.dd
103 dd/mm/yyyy
104 dd.mm.yy
105 dd-mm-yy
106 dd mmm yyyy(23 aug 2019)

Rank and Dense_Rank functions

- Introduced in SQL Server 2005
- Returns a rank starting at 1 based on the ordering of rows imposed by the ORDER BY clause
- ORDER BY clause is required
- PARTITION BY clause is optional
- When the data is partitioned, rank is reset to 1 when the partition changes

Difference between Rank and Dense_Rank functions
Rank function skips ranking(s) if there is a tie where as Dense_Rank will not.

```
Create Table Employees
(
   Id int primary key,
   Name nvarchar(50),
   Gender nvarchar(10),
   Salary int
)


Insert Into Employees Values (1, 'Mark', 'Male', 8000)
Insert Into Employees Values (2, 'John', 'Male', 8000)
Insert Into Employees Values (3, 'Pam', 'Female', 5000)
Insert Into Employees Values (4, 'Sara', 'Female', 4000)
Insert Into Employees Values (5, 'Todd', 'Male', 3500)
Insert Into Employees Values (6, 'Mary', 'Female', 6000)
Insert Into Employees Values (7, 'Ben', 'Male', 6500)
Insert Into Employees Values (8, 'Jodi', 'Female', 4500)
Insert Into Employees Values (9, 'Tom', 'Male', 7000)
Insert Into Employees Values (10, 'Ron', 'Male', 6800)

select*from Employees

---without partition
SELECT Name,Gender, Salary,
RANK() OVER (ORDER BY Salary DESC) AS [Rank],
DENSE_RANK() OVER (ORDER BY Salary DESC) AS DenseRank
FROM Employees

---with partition
SELECT Name,Gender, Salary,
RANK() OVER (PARTITION BY Gender ORDER BY Salary DESC) AS
[Rank],
DENSE_RANK() OVER (PARTITION BY Gender ORDER BY Salary DESC)
AS DenseRank
FROM Employees
```

Running Total:

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

we can add column values row by row and
find running total for every record after.

```
Create Table Employees1
(
    Id int primary key,
    Name nvarchar(50),
    Gender nvarchar(10),
    Salary int
)


Insert Into Employees1 Values (1, 'Mark', 'Male', 5000)
Insert Into Employees1 Values (2, 'John', 'Male', 4500)
Insert Into Employees1 Values (3, 'Pam', 'Female', 5500)
Insert Into Employees1 Values (4, 'Sara', 'Female', 4000)
Insert Into Employees1 Values (5, 'Todd', 'Male', 3500)
Insert Into Employees1 Values (6, 'Mary', 'Female', 5000)
Insert Into Employees1 Values (7, 'Ben', 'Male', 6500)
Insert Into Employees1 Values (8, 'Jodi', 'Female', 7000)
Insert Into Employees1 Values (9, 'Tom', 'Male', 5500)
Insert Into Employees1 Values (10, 'Ron', 'Male', 5000)

select*from Employees1

select sum(salary) from Employees1

---SQL Query to compute running total without partitions
SELECT Name, Gender, Salary,
      SUM(Salary) OVER (ORDER BY ID) AS RunningTotal
FROM Employees1

---SQL Query to compute running total with partitions
SELECT Name, Gender, Salary,
      SUM(Salary) OVER (PARTITION BY Gender ORDER BY ID) AS
RunningTotal
FROM Employees1

--What happens if I use order by on Salary column
--If you have duplicate values in the Salary column,
---all the duplicate values will be added to the running total at once.
```

-- In the example below notice that we have 5000 repeated 3 times.
--- So 15000 (i.e 5000 + 5000 + 5000) is added to the running total at once.

```
SELECT Name, Gender, Salary,
      SUM(Salary) OVER (ORDER BY Salary) AS RunningTotal
FROM Employees1
```

IIF function:
---

iif function is the replacement of case expression

iif is the short hand way of writing case expession
•Introduced in SQL Server 2012
•Returns one of two the values, depending on whether the Boolean expression evaluates to true or false
•IIF is a shorthand way for writing a CASE expression
Syntax : IIF (boolean_expression, true_value, false_value)

```
Create table Employees2
(
    Id int primary key identity,
    Name nvarchar(10),
    GenderId int
)


Insert into Employees2 values ('Mark', 1)
Insert into Employees2 values ('John', 1)
Insert into Employees2 values ('Amy', 2)
Insert into Employees2 values ('Ben', 1)
Insert into Employees2 values ('Sara', 2)
Insert into Employees2 values ('David', 1)

select*from Employees2

---Using CASE statement
SELECT Name, GenderId,
      CASE WHEN GenderId = 1
            THEN 'Male'
```

```
                ELSE 'Female'
              END AS Gender
FROM Employees2

---Using IIF function
SELECT Name, GenderId,
 IIF(GenderId = 1, 'Male', 'Female') AS Gender
FROM Employees2
```

Advanced sql server topics:

Joins:
---

are used to retrieve the data from two or more tables
and display in one table format.

types of joins:

1.inner join
2.outer join---left join or left outer join
right join or right outer join
full join or full outer join

3.cross join
4.self join

general formula for joins:

```
select      columnlist
from        lefttablename
join_type   righttablename
on          join_condition


create table tblemployee
(
id int,
name varchar(20),
gender varchar(10),
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
salary int,
deptid int
)

insert into tblemployee values(1,'sai','male',34000,1)
insert into tblemployee values(2,'sana','female',45000,2)
insert into tblemployee values(3,'mohan','male',38000,1)
insert into tblemployee values(4,'shashi','male',24000,2)
insert into tblemployee values(5,'shanvu','female',64000,3)
insert into tblemployee values(6,'manoj','male',39000,3)
insert into tblemployee values(7,'kiran','male',38000,1)
insert into tblemployee values(8,'raja','male',64000,2)
insert into tblemployee values(9,'sri','female',38000,null)
insert into tblemployee values(10,'durga','male',46000,null)


select*from tblemployee

create table tbldepartment
(
id int,
deptname varchar(10),
depthead varchar(10)
)

insert into tbldepartment values(1,'IT','shashi')
insert into tbldepartment values(2,'HR','shanvi')
insert into tbldepartment values(3,'payroll','mohan')
insert into tbldepartment values(4,'other','sai')

select*from tbldepartment

--inner join:
-----------
--it will return matching rows from both tables
--and eleminate non matching rows

select      name,gender,salary,deptname
from        tblemployee
inner join  tbldepartment
on          tblemployee.deptid=tbldepartment.id
```

---left outer join:

--it will return matching rows from both tables
---and non matching rows from left table

```
select          name,gender,salary,deptname
from            tblemployee
left outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
```

---right outer join:

--it will return matching rows from both tables
---and non matching rows from right table

```
select          name,gender,salary,deptname
from            tblemployee
right outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
```

---full outer join:

--it will return matching rows from both tables
---and non matching rows from both tables

```
select          name,gender,salary,deptname
from            tblemployee
full outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
```

---advanced joins or itelligent joins:

---to get non matching rows from left table only
---to get non matching rows from right table only
---to get non matching rows from both tables only

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

--then we use advanced joins

---to get non matching rows from left table only

```
select          name,gender,salary,deptname
from            tblemployee
left outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
where           tblemployee.deptid is null
```

---to get non matching rows from right table only

```
select          name,gender,salary,deptname
from            tblemployee
right outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
where           tblemployee.deptid is null
```

---to get non matching rows from both tables only

```
select          name,gender,salary,deptname
from            tblemployee
full outer join  tbldepartment
on              tblemployee.deptid=tbldepartment.id
where           tblemployee.deptid is null or
                tbldepartment.id is null
```

---cross join

--it return cartesian product of two tables
---suppose first table is having 5 records ,second table is having
---4 records then cartesian product of tables is 20 records

---cross join should not have on clause

```
select name gender,salary,deptname
from   tblemployee
cross join tbldepartment
```

---self join: a table is going to join with itself is called self join

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
select*from tblself


create table tblself
(
EmployeeId int,
name varchar(10),
ManagerId int
)

insert into tblself values(1,'sai',3)
insert into tblself values(2,'shashi',1)
insert into tblself values(3,'raj',null)
insert into tblself values(4,'ram',1)
insert into tblself values(5,'krish',1)

select*from tblself

select E.name as Employee,
      M.name as Manager
from     tblself E
inner join tblself M
on        E.ManagerId=M.EmployeeId


--how to use joins for three tables
---------------------------------


create table table1
(
id int,
name varchar(10)
)

insert into table1 values(1,'sai')
insert into table1 values(2,'sana')
insert into table1 values(4,'ram')
insert into table1 values(4,'raj')

create table table2
```

```
(
id int,
name varchar(10)
)

insert into table2 values(1,'sai')
insert into table2 values(2,'sana')
insert into table2 values(3,'mohan')
insert into table2 values(5,'raj')

create table table3
(
id int,
name varchar(10)
)

insert into table3 values(1,'sai')
insert into table3 values(2,'sana')
insert into table3 values(3,'mohan')
insert into table3 values(4,'raj')
insert into table3 values(5,'durga')

select*From table1
select*from table2
select*from table3


Select     table1.ID ,table1.Name
from       Table1
inner join Table2
on         Table1 .ID =Table2 .ID
inner join Table3
on         table2.ID=Table3 .ID

--Where Condition (Inner Join with Three Tables)

Select     table1.ID ,table1.Name
from       Table1
inner join Table2
on         Table1 .ID =Table2 .ID
inner join Table3
on         table2.ID=Table3 .ID
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
where     table1.Name=Table3.Name


---Different ways to replace NULL in sql server


select*from tblself

---Replacing NULL value using ISNULL() function:
--- We are passing 2 parameters to IsNULL() function.
---If M.Name returns NULL, then 'No Manager' string is used as the
replacement value.

SELECT E.Name as Employee,
ISNULL(M.Name,'No Manager') as Manager
FROM tblself E
LEFT JOIN tblself M
ON E.ManagerID = M.EmployeeID


---Replacing NULL value using CASE Statement:

SELECT E.Name as Employee,
  CASE WHEN M.Name IS NULL THEN 'dont have manager'
   ELSE M.Name END as Manager
FROM  tblself E
LEFT JOIN tblself M
ON   E.ManagerID = M.EmployeeID

---Replacing NULL value using COALESCE() function:
--- COALESCE() function, returns the first NON NULL value.

SELECT E.Name as Employee,
 COALESCE(M.Name, 'manger has not assign') as Manager
FROM tblself E
LEFT JOIN tblself M
ON E.ManagerID = M.EmployeeID

Views in sql:
------------

1.view is a saved sqlquery
```

2.view is for reusability
3.it saves lot of time
4.view is a window or virtual table
5.a table which is used to create a view is called base table
6.view will provide security i.e row level and column level
7.views can be updatable
8.view can be alter and drop

types of views

1.simple view
2.complex view
3.encrypted view
4.schembinding view

simple view:
---
a view which is create for single table or single select statement

syntax:

```
create view viewname
as
select statement
```

```
create view vwsimple
as
select*from tblemployee where gender='male'

---to get view data
select*from vwsimple

--to get view text
sp_helptext vwsimple

select*from empdetails

---create view to provide rowlevel security
create view vwgetrows
as
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
select*from empdetails where id between 1 and 5

select*from vwgetrows

sp_helptext vwgetrows

----create view to provide column level security

create view vwgetcolumns
as
select name,address,designation,age from empdetails

select*from vwgetcolumns

create view v1
as
select*from tblemployee

select*from tblemployee
select*from v1

--view can be updatable means when we insert,delete,update records
--in base table that will be effect on view table
--same like when we insert,delete,update records
--in view table that will be effect on base table

insert into v1 values(11,'nani','male',7000,3)

select*from tblemployee
select*from v1

delete from tblemployee where id=11

create view v2
as
select*from tblemployee where name like 's%'

select*from v2

sp_helptext v2

---alter view
```

```
alter view v2
as
select* from tblemployee where name not like 's%'

select*From v2

sp_helptext v2

---to get all views from current database

select*From sysobjects where xtype='v'

---to drop view

drop view v1
```

complex view:
---

a view which is create for two or more tables with order by or group by
or joins etc ..is called complex view.

```
---create complex view to get name,gender,salary,deptname
---where deptname=IT

select*from tblemployee
select*from tbldepartment

create view vwcomplex
as
select      name ,gender,salary,deptname
from        tblemployee
join        tbldepartment
on          tblemployee.deptid=tbldepartment.id
where       tbldepartment.deptname='IT'

select*From vwcomplex

sp_helptext vwcomplex

---create view to count no of employees based on deptname
```

```
CREATE view vwnoofEmployees
as
select       deptname, count(*) as total
from         tblEmployee
inner join   tblDepartment
on          tblEmployee.deptid=tblDepartment.id
group by    deptname

select*from vwnoofEmployees
```

encrypted view:
---

a view which is associated with an option called with encryption
is called encrypted view

in encrypted view we cant see the view text,it will be hide.

```
create view vwcomplex1 with encryption
as
select      name ,gender,salary,deptname
from        tblemployee
join        tbldepartment
on          tblemployee.deptid=tbldepartment.id
where       tbldepartment.deptname='IT'

sp_helptext vwcomplex1

--The text for object 'vwcomplex1' is encrypted.

select*From vwcomplex1
```

Note: when a table is having view ,that table is called base table
when we drop base table that will be drop but view on that table
will not drop.
whenever we call that view ,it wont work because view should get the data
from its base table.

```
create view v9
as
select*from employees

select*from v9

drop table Employees

select*from v9
```

schemabinding view:
---

a view which is associated with an option called with schembinding
is called schembinding view
the main advantage of schemabinding view is base table cant be drop
in schemabinding view * not allowed
in schemabinding view tablename should preceeded with schema i.e dbo

```
create view vwschema with schemabinding
as
select name,gender,salary from dbo.tblemployee

select*from vwschema

drop table tblemployee
```

--error:Cannot DROP TABLE 'tblemployee' because it is being referenced by object 'vwschema'.

indexes in sql:
---

indexes are used to get or access the data from table quickly

if there is no indexes in table , sql server engine will arrange table scan ,after table scan we will get records.

always table scan will give bad performence

types of indexes:

1.clustered index
2.non clustered index

in sql serevr indexes will work automatically

clustered index:

1.it will create automatically when we create table with primary key
2.a table should have only one clustered index
3.clustered index will arrange the records in physical order

non-clustered index:

1.it will create automatically when we create table with unique key
2.a table may have more than one non clustered index
3.non clustered index will not arrange the records in physical order

syntax to create index:

```
create index indexname
on tablename(columnname)


create index ix_tblemployee_id
on tblemployee(id)

---to get index details

sp_help tblemployee

----to get only index details
sp_helpindex tblemployee

---to drop index

drop index myindex on tbldepartment

sp_helpindex tbldepartment

create table tblindex(id int primary key,name varchar(10))
```

```
sp_helpindex tblindex

insert into tblindex values(2,'sai')
insert into tblindex values(1,'ram')
insert into tblindex values(4,'sana')
insert into tblindex values(3,'mohan')

select*from tblindex


create table tblindex1(id int unique,name varchar(10))

sp_helpindex tblindex1

insert into tblindex1 values(2,'sai')
insert into tblindex1 values(1,'ram')
insert into tblindex1 values(4,'sana')
insert into tblindex1 values(3,'mohan')

select*from tblindex1

----the indexes which are automatically created through primary key
---or unique key cant be drop

---if u want to drop that, through design is possible

sp_helpindex tblindex

drop index PK__tblindex__3213E83F56A7D3B5
on tblindex

--An explicit DROP INDEX is not allowed

drop index UQ__tblindex__3213E83E04F7845F
on tblindex1

---creating perticular indexes

create clustered index myindex
on empdetails(id)
```

```
create nonclustered index myindex1
on empdetails(id)

sp_helpindex empdetails
```

storedprodecure in sql:

1.sp is a database object which contain set of precompiled queries.
2.in sp once the query compiled successfully again and again we need not to compile ,directly we can call for execution.
3.sp are replacement of direct queries which are passed from front end
4. we use direct command or queries from front end to back end
at back end query need to compile every time and execute.
if the query not compiled successfully ,the query as it is sends to front end ,again as programmer we need rectify that query and sends to back end
5.how many times we are sending queruies from front end to back end
every time query need to be compile and execute.
6. this will make burdon on to the serever.

drawbacks of direct queries:

1.unnecessary compilations will takes place on serever
2.server burdon is more
3.queries will not provide security
4.queries will not provide reusability
5.there may be chance to get sql server injection attacks

to avoid these drawbacks we use sp

advantages of sp:

1.unnecessary compilations will not takes place on serever
2.server burdon is less
3.sp will provide security and reusability
4.will avoid sql server injection attacks

Types of sp:

1.systemdefined sp
2.temporary sp
3.userdefined sp

systemdefined sp: these sp are predfined

all system defined sp are prefixed with sp_

ex: sp_help
sp_helpindex
sp_helpconstraint
sp_rename
sp_helptext
sp_depends

temporary sp: these sp are temporary purpose
all these sp are prfixed with #

userdefined sp: these are created by programmer which is used in the application
according to requirement

microsoft is not recommend to use sp_ for userdefined sp
because there will be conflict b/w userdefined and systemdefined sp

syntax:

```
create procedure procedurename
as
begin
statements to be execute..
end
```

```
create procedure prc1
as
begin
delete from empdetails where id=1
end
```

---to execute procedure

```
exec prc1
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
--or
execute prc1

select*from empdetails

---in the above procedure if user wants to delete more records
--like id=2 or id=3, every time we need to modify the sp
---but it is not good practice in reality
---to avoid this we sp with parameter

create procedure prc2
@id int
as
begin
delete from empdetails where id=@id
end

exec prc2

--error:Procedure or function 'prc2' expects parameter '@id', which was not
supplied.

exec prc2 2
select*From empdetails

---craete sp to get emp records where gender is male

create procedure prc3
@gender varchar(50)
as
begin
select*From tblemployee where gender=@gender
end

exec prc3 'male'
exec prc3 'female'

select*from empdetails

---create sp for insert

create procedure prcinsert
```

```
@id int,
@name varchar(50),
@address varchar(50),
@designation varchar(50),
@age int,
@salary int
as
begin
insert into empdetails
values(@id,@name,@address,@designation,@age,@salary)
end

exec prcinsert 1,'mohan','hyd','trainer',35,90000

select*from empdetails

---create sp for  update

create procedure prcupdate
@id int,
@name varchar(50),
@address varchar(50),
@designation varchar(50),
@age int,
@salary int
as
begin
update empdetails set
name=@name,address=@address,designation=@designation,
age=@age,salary=@salary where id=@id
end

execute prcupdate 1,'shanvi','ameerpet','teacher',25,35000

select*from empdetails

---to modify the sp we use alter

create procedure prc9
as
begin
select*from empdetails where name like 's%'
```

```
end

exec prc9

---to alter prcedure

alter procedure prc9
as
begin
select*from empdetails where name not like 's%'
end

exec prc9

---to get all sp from current database

select*From sysobjects where xtype='p'

---drop sp

drop procedure prc1

---to get procedure text

sp_helptext prcinsert


--sp_depends--- it is used to get dependencies of table

sp_depends empdetails

sp_depends tbldepartment
```

Triggers in sql:

---

trigger is a special type of stored procedure
trigger is an event which will be fired automatically when we execute commands
trigger is used to monitor user activities and store the information.
triggers are used to apply some conditions on table
triggers are used to provide restriction on tables and databases which

should not access.

types of triggers:

1.DML Triggers
2.DDL Triggers

DML Triggers will be fired in responce to DML commands like insert,delete update

DDL Triggers will be fired in responce to DDL command like alter,drop,create

sql server support after triggers only i.e after insert,after update,after delete

where as oracle suport both after and before triggers

Triggers will use internally magical tables
magical tables are used to strore trigger information

magical tables are 2 types

1.inserted magical table
2.deleted magical table

inserted magical table: will store information when insert trigger will be fired

deleted magical table: will store information when delete trigger will be fired

syntax to create trigger:

```
create trigger triggername
on tablename
for insert,delete,update
as
begin
statements to be execute
end
```

```
---create trigger for insert

create trigger trinsert
on tblemployee
for insert
as
begin
select*from inserted
end

select*from tblemployee

insert into tblemployee values(11,'ram','male',6000,2)

create trigger tr1
on tblemployee
for insert
as
begin
delete from empdetails where id=2
end

select*from empdetails

insert into tblemployee values(12,'rana','male',8000,1)

---create trigger for delete

create trigger trdelete
on tblemployee
for delete
as
begin
select*from deleted
end

delete from tblemployee where id=11

select*from tblemployee

---create trigger for update
```

```
create trigger trupdate
on tblemployee
for update
as
begin
select*from inserted
select*from deleted
end

update tblemployee set name='mohan',gender='male',salary=5550,deptid=9
where id=12

select*from tblemployee

---create trigger and capture the data and store in other table

create table tblcapture
(
id int identity,
capturedata varchar(100)
)

select*from tblcapture


create trigger trcapture
on tblEmployee
for insert
as
begin
declare @id int
select @id=id from inserted
insert into tblcapture values
(
'New Employee with Id='+CAST(@id as nvarchar(20))+
'Is Added at'+CAST(getdate() as varchar(100))
)
end

insert into tblemployee values(13,'tarun','male',5000,2)

select*From tblcapture
```

```
create trigger trcapture1
on tblEmployee
for delete
as
begin
declare @id int
select @id=id from deleted
insert into tblcapture values
(
'Employee with Id='+CAST(@id as nvarchar(20))+
'Is Removed at'+CAST(getdate() as varchar(100))
)
end

delete from tblemployee where id =10

select*From tblcapture



---to get all triggers from current database
select*from sysobjects where xtype='tr'

---to get trigger text

sp_helptext trcapture

--to drop trigger

drop trigger tr1

select*from sysobjects where xtype='tr'

----create trgger for do not accept insert,delete ,update

create trigger tr8
on tbldepartment
for insert,update,delete
as
begin
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
print 'access is denied'
rollback
end

select*from tbldepartment

delete from tbldepartment where id=1
insert into tbldepartment values(5,'sales',5)

drop trigger tr8

delete from tbldepartment where id=1

select*from tbldepartment


----ddl trigger

--create a trigger for do not accept alter and drop tables on database

create trigger tr7
on database
for alter_table,drop_table
as
begin
print 'alter and drop is not possible'
rollback
end

alter table emp
alter column name varchar(50)

drop table emp
---if u want drop or alter later then we should drop triiger

---to drop trigger on database

drop trigger tr7 on database

---now alter and drop is possible
alter table emp
alter column name varchar(50)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

drop table emp

schema:
---
schema is nothing but grouping the tables
if we create schema for group of tables, if u give permision on schema
then use can access only on that schema tables.

Grant: is used to give permission to access
Revoke: is used to cancel permision to access

default schema in sql server is dbo(database object)

syntax to create schema:

---create schema schemaname
create schema myschema

----create table with schema

create table myschema.tblsample(id int,name varchar(10))

insert into myschema.tblsample values(1,'sai')

select*from myschema.tblsample

---login: is used to connect to the sql serever

---create login loginname
---with password='password'

create login shanvi
with password='shanvi'

---create user username
--for login loginname
--with default_schema=schemaname

create user chaitanya
for login shanvi

with default_schema=myschema

---now try to connect sql server with server authentication by providing username as shanvi and password as shanvi

after connecting move to chaitanya database

then try access like:

select*From myschema.tblsample

then we will get error:
The SELECT permission was denied on the object 'tblsample', database 'chaitanya', schema 'myschema'.

to avoid this--goto windows authentication query window

then execute grant command like:

--to provide permision

grant select,insert,update,delete on schema::myschema to chaitanya

now goto server authentication window and exeute like:

select*From myschema.tblsample

now we can abl to access

---to cancel permision --goto windows authentication query window

then execute revoke command like:

revoke select,insert,update,delete on schema::myschema to chaitanya

Trnasaction in sql server:

What is a Transaction?
A transaction is a group of commands that change the data stored in a database. A transaction, is treated as a single unit.
A transaction ensures that, either all of the commands succeed,

or none of them. If one of the commands in the transaction fails, all of the commands fail, and any data that was modified in the database is rolled back. In this way, transactions maintain the integrity of data in a database.

Transaction processing follows these steps:
1. Begin a transaction.
2. Process database commands.
3. Check for errors.
   If errors occurred,
      rollback the transaction,
   else,
      commit the transaction

```
create table tbltransaction
(
id int,name varchar(20),address varchar(10)
)

create table tbltransaction1
(
id int,name varchar(20),address varchar(10)
)

insert into tbltransaction values(1,'sai','hyd')

insert into tbltransaction1 values(1,'sai','hyd')

select*from tbltransaction
select*From tbltransaction1


Create Procedure spupdateaddress
as
Begin
 Begin Try
  Begin Transaction
   update tbltransaction set address='hyderabad' where id=1
   update tbltransaction1 set address='hyderabad' where id=1
  Commit Transaction
 End Try
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
 Begin Catch
  Rollback Transaction
 End Catch
End

exec spupdateaddress

select*From tbltransaction
select*From tbltransaction1

--if any one of the command fails entire the transaction fails
--let see

alter Procedure spupdateaddress
as
Begin
 Begin Try
  Begin Transaction
   update tbltransaction set address='hyderabadhyderabad' where id=1
   update tbltransaction1 set address='hyderabad' where id=1
  Commit Transaction
 End Try
 Begin Catch
  Rollback Transaction
 End Catch
End

exec spupdateaddress

select*from tbltransaction
select*from tbltransaction1
```

Backup and Restore:

---

backup is the process of creating backup file from database
restore is the process of creating database using backup file

syntax:

```
backup database databasename
to disk='G:\databasename.bak'
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Ex:

backup database chaitanya
to disk='G:\chaitanya.bak'

syntax:

restore database databasename
from disk='G:\databasename.bak'

restore database chaitanya
from disk='G:\chaitanya.bak'

DTS(Data Transformation service):
---

we can trnsafer the data from one location to another
ex: from one database to another
ex: from sql serever to excel
ex: from excel to sql server

import: we can get the data from other locations to our sql database

export: we can trnsafer the data from our sql database to another location

sql server agent:
---

it is a service which is used to create jobs and schedules and other alerts

to work with sql server agent ---first we need to enable sql server agent service.

to enable this service --goto run---type services.msc---right click on sql server agent ---then click on start ---then service will start.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com