INTRODUCTION

DATABASE

A Specific physical location in the memory of a system where we store data at **DBMS**

DBMS Products: Database,FoxBase,Foxpro,etc..

RDBMS :Referential integrity + DBMS

RDBMS Products

Most commonly used RDBMS products are

PRODUCT NAME	MANUFACTURER NAME
Sql Server	Microsoft Corp
Oracle	Oracke Corp
DB2	IBM
SyBase	SyBase Incorp

Difference Between DBMS and RDBMS

DBMS	RDBMS
Data is stored in flat file format	Data is stored in the form of relations and is structure encrypted form
The data doesnot have security	Since the data is encrypted it can be processed by the application pgm only hence there is high security for the data
Huge volume of data cannot be stored	We can store huge volume of data
Data stored in different files cannot	Since the data is stored in theform of tables,we can logically establish relationship between more than one table and extract complete information

INTRODUCTION TO DATABASES

A DB is a collection of tables where we store the data about a business,and/or some other objects such as Views,Stored Procedures,Triggers and User Defined Functions,etc..

Files to store a DB

SQl server maps a Database using a set of Operating System files.

Primary Data File

This file contains the startup information of the database
Every database has ONE data file
The extension of this data file is.mdf

Transaction Log File

This file stores the log information used to recover the Database
Every database has ONE log file
The extension for transaction log file is.ldf

Secondary Data File

These files are used to store data that doesnot fit into the primary data file The extension of the secondary data file is .ndf

TYPES OF DATABASES

SQL Server provides two types of databases they are,
System Databases
UserDefined Databases

System Databases

Along with the installation of SQL Server,the following databases will be created automatically and thease are called System Databases.
master-Stores all Databases information model-acts as a model to the new databases msdb-SQL Server agent will use this tempdb-temporary works will be stored in this
pubs-training purpose only northwind-
training purpose only

User Defined Databases

The database that are created by users are called User Defined Databases **EX**:-employee,payroll,inventory,production,student etc..

Creating a Database

SYNTAX:-

```
CREATE DATABASE<DB NAME>
```

System Stored Procedures

This stored procedures lists all the properties of a given database

SYNTAX:-

```
SP_HELPDB<DB NAME>
```

Deleting a Database

SYNTAX:-

```
      DROP DATABASE<DB NAME>
```

DATA TYPES

A Datatype is an attribute that specifies what type of data can be stored in a column or a variable.

Categories of Data Types

SQL Server provides a set of predifined datatypes,they are

1.Character
2.Integer
3.Floating point
4.Date and time

Character

A Character data consists of any combination of letters,symbols and numbers

Char	=Fixed Length <8kb
Varchar	=Variable Length <=8kb
Text	=Variable Length >8kb

Integer

Integer data consists of -ve and +ve whole numbers

Tinyint	0-255
Smallint	-32768to+32767
Int	4bytes
Bigint	8bytes

Floating Point

This datatype allows fractional values to be stored as values in a column of a table Numeric(<precision>,<scale>)

Example:-

```
      Numeric(7,2)-99999.99
      Numeric(6,2)-9999.99
```

Date and Time

Data and Time data consists date or time combinations base Datatype

Small Datatype	jan-1-1753 to dec-31-9999
DateTime	jan-1-1990 to june-6-2079

CREATING A TABLE

SYNTAX:-

```
CREATE TABLE<TABLE NAME>
(Colname1 datatype,
 ColName2 datatype,
 ColName3 datatype,
..................
 ColNameN datatype)
```

To View The Structure and Properties of the Table

sp_help

The above stored procedure displays information about all the objects in the currently active databases

sp_help <table name>

The above stored procedure displays all the properties of the given table

Inserting Data into the Tables

SYNTAX:-

```
INSERT INTO <TABLE NAME>(<COLUMN LIST>)
VALUES(<LIST OF VALUES>)
```

Update Data into the Table

SYNTAX:-

```
UPDATE <TABLE NAME>
SET <COL1>=<NEW VALUE>(,<COL2>=<NEW VALUE>,...)
WHERE <CONDITION>
```

Delete Data from the Table

SYNTAX:-

```
DELETE FROM <TABLE NAME>
WHERE <CONDITION>
```

Truncate Table

SYNTAX:-

```
TRUNCATE TABLE <TABLE NAME>
```

EXAMPLES:-

```
create table dept
(deptno numeric(2,0),
dname varchar(14),  loc
varchar(14))
```

```
insert into dept values(10,'accounting','newyork')
insert into dept values(20,'research','dallas')
insert into dept values(30,'sales','chicago')
insert into dept values(40,'operations','boston')
select * from dept

create table emp
(empno numeric(4),
ename varchar(10),
job varchar(9), mgr
numeric(4), hiredate
datetime, sal
numeric(7,2), comm
numeric(7,2), deptno
numeric(2,0))

insert into emp
values(7369,'SMITH','CLERK',7902,'17-DEC-81',800,NULL,20)
insert into emp
values(7499,'ALLEN','SALESMAN',7698,'20-FEB-81',1600,300,30)
insert into emp
values(7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,500,30)      insert
into emp
values(7566,'JONES','MANAGER',7839,'02-APR-81',2975,NULL,20)
insert into emp
values(7654,'MARTIN','SALESMAN',7698,'28-SEP-81',1250,1400,30)
insert into emp
values(7698,'BLAKE','MANAGER',7839,'01-MAY-81',2850,NULL,30)
insert into emp
values(7782,'CLARK','MANAGER',7839,'09-JUN-81',2450,NULL,10)
insert into emp
values(7788,'SCOTT','ANALYST',7566,'19-APR-87',3000,NULL,20)
insert into emp
values(7839,'KING','PRESEDENT',NULL,'17-NOV-81',5000,NULL,10)
insert into emp
values(7844,'TURNER','SALESMAN',7698,'08-SEP-81',1500,0,30)
insert into emp
values(7876,'ADAMS','CLERK',7788,'23-MAY-87',1100,NULL,20)
insert into emp
values(7900,'JAMES','CLERK',7698,'03-DEC-81',950,NULL,30)
insert into emp
values(7902,'FORD','ANALYST',7566,'03-DEC-81',3000,NULL,20)
insert into emp
values(7934,'MILLER','CLERK',7782,'23-JAN-82',1300,NULL,10)

SELECT * from emp
```

ALTER TABLE

Once a table is created we can modify the structure of the table using ALTER statement.
ALTER statement is used by

1. To add a new column to an existing table
2. To alter the datatype/size of an existing column
3. To delete the unwanted column
4. To add a constraint
5. To drop a constraint

Adding a new column

This command is used to add a new column to an existing table
We can also add constraint to the newly adding column

SYNTAX:-

```
ALTER TABLE<TABLENAME>
ADD<COLNAME><DATATYPE>[<CONSTRAINT>]
```

Example:select * from
emp

```
alter table emp add
bonus numeric(15)
```

Altering the datatype / size of the existing column

SYNTAX:-

```
ALTER TABLE<TABLENAME>
ALTER COLUMN<COLNAME><NEWDATATYPE>
```

Example:-

Datatype size modifing :-

```
alter table emp alter column
ename varchar(50)
```

Datatype changing :-

```
alter table emp alter
column job char(25)
```

To delete an unwanted column from a table

SYNTAX:-

```
ALTER TABLE<TABLENAME>
DROP COLUMN<COLNAME>
```

Example:alter table
emp drop column
bonus

To add a constraint to the existing column

SYNTAX:-

```
ALTER TABLE<TABLENAME>
ADD CONSTRAINT<CONSTNAME><CONSTTYPE>(COLNAME)
```

Example:alter table

```
emp
add constraint q1 unique(empno)
sp_help emp
```

To drop a constraint on the existing column

SYNTAX:-

```
ALTER TABLE<TABLENAME>
DROP CONSTRAINT<CONSTNAME>
```

Example:alter table

```
emp drop
constraint q1
sp_help emp
```

Renaming a Database

SYNTAX:-

```
ALTER DATABASE payrolldb
MODIFY NAME=paydb
```

STORED PROCEDURE

```
sp_rename db
sp_rename 'OLD DB','NEW DB'
```

Renaming the column name of a table

In order to change the column name of the table is

```
SP_RENAME 'TABLENAME.COLNAME','NEW COLNAME'
```

Example:sp_rename

```
'emp.comm','bonus'
sp_rename
'emp.bonus','comm'
```

Renaming the name of a table

```
SP_RENAME 'OLD TABLE NAME','NEW TABLE NAME'
```

Example:sp_rename

```
'emp','employee'
sp_rename 'employee','emp'
```

OPERATORS IN SQL SERVER

An operator is a symbol specifying an action that is performed on one or more expressions

Arithmetic operators

Arthmetic operators perform mathamatical operations on two expressions of any of the datatypes of the numeric data type category

1.add(+)
2.subtract(-)
3.multiply(*)
4.divide(/)
5.modulo(%)

Comparision Operators

Comparision operators test whether or not two expressions are the same
>,>=,<,<=,!=,<>,!> and !<

Logical Operators

Logical operators test for the truth of some condition.
AND,OR,NOT,LIKE,IN,BETWEEN,ANY,ALL,SOME and EXISTS

String Concatenation Operators

The string concatanation operators allows string concatenation with the addition sign(+).
All other string manipulation is handled through string functions such as SUBSTRING

FUNCTIONS IN SQL SERVER

A function is a sequence of code/set of T-SQL commands that performs a specific task.
SQL server provides two types of functions

-Build in functions
-User defined functions

Build in functions

These are pre-defined functions within sql server eg:-
sum(),avg(),min(),max(),etc.

User defined functions

Sql server allows the user to create some functions called user defined functions

SYSTEM DEFINED FUNCTIONS

1.String functions
2.Number functions
3.Date functions
4.Aggregate functions

STRING FUNCTIONS

1.ASCII()-This functions returns the ASCII value of the leftmost character in a given string.
2.CHAR()-This functions returns the character equilent of a given ASCII value.
3.UPPER()-This function converts the characters in a given string into upper case. 4.LOWER()-This function converts the upper case characters in a given string into lower case. 5.LTRIM()-This function removes the leading blank spaces in a given string.
6.RTRIM()-This function removes the trailing blank spaces in a given string.
7.SUBSTRING()-This function is used to obtain a part of a given string.
8.LEN()-This function is use to find the length of a given string.

```
select ascii(0) select ascii('a') select ascii('A') select

ascii('f') select char(23) select char(56) select char(67)

select     char(65)     select     lower('APPLE')     select

upper('apple') select ltrim('                    apple')a

select rtrim('apple                    ')a select ltrim('

apple           ')a select ltrim(rtrim('           apple

'))a       select       substring('india',2,1)       select

substring('january',3,5)                              select

ename,substring(ename,1,7)       from       emp       select

ename,substring(ename,len(ename),1)    from    emp    select

len('krishnanji555@gmail.com')                        select

len('aramachandrareddy55555@gmail.com')
```

NUMBER FUNCTIONS

1.ABS-Absolute value of a given number
      ABS(<VALUE/FN/COLNAME>/<EXPECT FUNCTION>)
2.SQRT-Squareroot of a given number
      SQRT(<VALUE/COLNAME/EXP>)
3.SQUARE-Square of a given number
      SQUARE(<VALUE/COLNAME/EXP>)
4.POWER-X raised of the power of Y
      POWER(<BASENAME>,<INDEX>)
5.CEILING-Ceiling value of a given number

```
      CEILING(<VALUE/COLNAME/EXP>)
6.FLOOR-Floor value of a given number
      FLOOR(<VALUE/COLNAME/EXP>)
7.ROUND-Round any no of the special no of decimal number
      ROUND(<VALUE/COLNAME/NO OF DECIMAL PLACES>)

select abs(-10) select sqrt(25) select sqrt(-25)

select square(2.3) select square(-5.2) select

power(5,4) select power(3,2) select

floor(1.9999999999999) select

ceiling(9.000000000000000000000000000000001)

select round(5.12456777,5) select round(4.12,-1)

select round(345.1234,-1) select round(99.11123,-1)
```

DATE TIME FUNCTIONS

```
1.GETDATE-Will return current system date and time and time value
2.DATEPART-Will return a portion of date from the date
3.DATENAME-Will return name format of the given date
4.DATEADD-Add a specific number of date parts to the given date
5.DATEDIFF-Returns date part difference between any two given date
6.DAY-Return the day number from the given date
7.MONTH-Returns the month number from the given date
8.YEAR-Returns the year number from the given date

DATEPART SYNTAX:datepart(<dateportion>,<datevalue>/<colname>)
      Possible values for dateportion are as follows
            DD-DAY NUMBER
            MM-MONTH NUMBER
            YY-YEAR NUMBER
            HH-HOUR NUMBER
            MI-MINUTES NUMBER
            SS-SECONDS NUMBER
            DW-WEEK DAY NUMBER(SUNDAY AS 1 AND SATURDAY AS 7)

select getdate() select datepart(mm,hiredate) as

month from emp select datename(mm,hiredate) as

month from emp select dateadd(mm,5,hiredate) as

month from emp select

datediff(dd,hiredate,getdate()) from emp select

datediff(mm,hiredate,getdate()) from emp select
```

datediff(yy,hiredate,getdate()) from emp select

datediff(mi,hiredate,getdate()) from emp select

datediff(ss,hiredate,getdate()) from emp select

datediff(ss,'02-sep-2009',getdate()) from emp

select datediff(hh,'02-sep-09',getdate()) from emp

select day(getdate()) select month(getdate())

select year(getdate()

AGGREGATE FUNCTIONS

1.COUNT(*)-This function returns the values of the no of rows available in the table.
      SELECT COUNT(<COLNAME>) FROM <TABLENAME> 2.SUM()-This function returns the sum of values in a given column.
      SELECT SUM(<COLNAME>) FROM <TABLENAME> 3.AVG()-This function returns the avg value in a given column.
      SELECT AVG(<COLNAME>) FROM <TABLENAME> 4.MIN()-This function returns the min values in a given column.
      SELECT MIN(<COLNAME>) FROM <TABLENAME> 5.MAX()-This function returns the max values in a given column.
      SELECT MAX(<COLNAME>) FROM

<TABLENAME> select count(*) from emp select

sum(sal) from emp

select avg(sal) from emp select min(sal) from

emp select max(sal) from emp select max(comm)

from emp select min(comm) from emp where comm

is not null select * from emp select sum(empno)

from emp select count(comm) from emp

CONSTRAINTS

DATA INTEGRITY CONSTRAINTS

Data integrity ensures the correctness of the data stored in DB
A constraint is one that is applied on the data available in a DB

CONSTRAINT DEFINATION

A constraint can be defined on a column at the time of creating a table or it can be added to an already existing column in a table

CATEGORIES OF DATA INTIGRITY

Data integrity is brodly classified into the following categories
1.Entity Integrity
2.Domain Integrity
3.Referential Integrity

Entity Integrity

Entity integrity ensures that each row can be uniquly identified
1.PRIMARY KEY constraint
2.UNIQUE constraint

Domain Integrity

Domain integrity enforces data integrity by restricting the type of data and range of values
in a column
1.NOT NULL constraint
2.DEFAULT constraint
3.CHECK constraint

Referential Integrity

referential integrity maintains the integrity of the data by ensuring that the changes made in the parent table are reflected in all the dependent(child) tables
1.FOREIGN KEY constraint

NOT NULL

•NOT NULL constraint ensures that NULL values are not entered into the column(s) of a table. •However we can enter the duplicate values into those columns that have been declared using NOT NULL constraint. •A NULL value is not same as ZERO or BLANKSPACE.
•NULL means NOT DEFINED/NO ENTRY has been made.
•A NOT NULL constraint cannot implemented of a table that already contains NULL values as data in it.

EXAMPLES:-

```
CREATE TABLE IPL
(tno int NOT NULL,  tname varchar(25),  caption varchar(15),  coach
varchar(15)) INSERT INTO IPL VALUES(1,'KOLKATA KNIGHT
RIDERS','GANGULY','JOHNRIGHT')

INSERT INTO IPL VALUES(2,'DECCAN CHARGES','LAXMAN','AKRAM')
INSERT INTO IPL VALUES(3,'RAYOL CHALANGERS BANGALOR','DRAVID','KUMBLE')

INSERT INTO IPL VALUES(4,'PUNE WARRIERS','YOURAJ SINGH','KIRSTEN')

INSERT INTO IPL VALUES(5,'MUMBAI INDIANS','SACHIN','AJARUDDIN')

SELECT * FROM IPL

INSERT INTO IPL VALUES(1,'RAJASTHAN ROYALS','HARBAJAN SINGH','WARN')
```

ERROR:-

```
INSERT INTO IPL VALUES(NULL,'DELHI DARE DEVILIS','SEHWAG','MIYANDAD')
```

ERROR:-

```
INSERT INTO IPL(TNAME,CAPTION,COACH)
VALUES('KOCHI TUSKERS','GANGULY','KAPIL DEV')
```

SYNTAX:-

```
ALTER TABLE <TABLE NAME>
ALTER COLUMN<COL NAME><DATATYPE>NOT NULL

ALTER TABLE IPL
ALTER COLUMN TNAME VARCHAR(25) NOT NULL
```

ERROR:-

```
INSERT INTO IPL(TNO,CAPTION,COACH)
VALUES(1,'DHONI','CHAPEL')

INSERT INTO IPL
VALUES(1,'CHENNAI','DHONI','CHAPEL')

DROP TABLE IPL
```

UNIQUE

The condition for adding a UNIQUE constraint to an already existing column in a table is that the column must not contain any duplicate values in it.

EXAMPLES:-

```
CREATE TABLE IPL
(tno int,  tname varchar(25),  caption varchar(15),  coach varchar(15))
INSERT INTO IPL VALUES(1,'KOLKATA KNIGHT RIDERS','GANGULY','JOHNRIGHT')

INSERT INTO IPL VALUES(2,'DECCAN CHARGES','LAXMAN','AKRAM')

INSERT INTO IPL VALUES(3,'RAYOL CHALANGERS BANGALOR','DRAVID','KUMBLE')

INSERT INTO IPL VALUES(4,'PUNE WARRIERS','YOURAJ SINGH','KIRSTEN')

INSERT INTO IPL VALUES(5,'MUMBAI INDIANS','SACHIN','AJARUDDIN')

SELECT * FROM IPL

INSERT INTO IPL VALUES(1,'RAJASTHAN ROYALS','HARBAJAN SINGH','WARN')
```

ADD CONSTRAINT UNIQUE INTO COL LEVEL

SYNTAX:-

```
      ALTER TABLE <TABLE NAME>
      ADD CONSTRAINT <COL NAME><CONSTYPE>(COLNAME)
```

ERROR:-

```
      ALTER TABLE IPL
      ADD CONSTRAINT UQ1 UNIQUE(TNO)

      SELECT * FROM IPL

      DELETE FROM IPL
      WHERE TNAME='RAJASTHAN ROYALS'

      ALTER TABLE IPL
      ADD CONSTRAINT UQ1 UNIQUE(TNO)
```

ERROR:-

```
      INSERT INTO IPL
      VALUES(2,'PANJAB TIGERS','GABBAR SINGH','MANMOHAN SINGH')
```

DELETE UNIQUE CONSTRAINT

SYNTAX:-

```
      ALTER TABLE <TABLE NAME>
      DROP CONSTRAINT <CONST NAME>

      ALTER TABLE IPL
      DROP CONSTRAINT UQ1

      DROP TABLE IPL
```

PRIMARY KEY

If a primary key is implemented on a column of a table then
The values cannot be duplicated in the column
Null values cannot be entered A table
must have only ONE primary key.

NOTE:-

If we want to add the primary key constraint to a column of an already existing table,then the values in the column should not contain duplicate values and also must not contain any null values

EXAMPLES:-

```
      CREATE TABLE IPL
      (tno int,  tname varchar(25),  caption varchar(15),  coach varchar(15))
      INSERT INTO IPL VALUES(1,'KOLKATA KNIGHT RIDERS','GANGULY','JOHNRIGHT')

      INSERT INTO IPL VALUES(2,'DECCAN CHARGES','LAXMAN','AKRAM')

      INSERT INTO IPL VALUES(3,'RAYOL CHALANGERS BANGALOR','DRAVID','KUMBLE')

      INSERT INTO IPL VALUES(4,'PUNE WARRIERS','YOURAJ SINGH','KIRSTEN')
```

```
INSERT INTO IPL VALUES(5,'MUMBAI INDIANS','SACHIN','AJARUDDIN')

INSERT INTO IPL VALUES(NULL,'CHENNAI','DHONI','KRISHNA')

SELECT * FROM IPL
```

ADD CONSTRAINT INTO COLUMN LEVEL

ERROR:-

```
ALTER TABLE IPL
ADD CONSTRAINT PK1 PRIMARY KEY(TNO)
```

ERROR:-

```
ALTER TABLE IPL
ALTER COLUMN TNO INT NOT NULL

SELECT * FROM IPL
```

In the above table contains null values at col(tno).We can add primary key at this column first we choose that column contain not null values

```
UPDATE IPL
SET TNO=6
WHERE TNAME='CHENNAI'

ALTER TABLE IPL
ALTER COLUMN TNO INT NOT NULL

ALTER TABLE IPL
ADD CONSTRAINT PK1 PRIMARY KEY(TNO)

SELECT * FROM IPL
```

ERROR:-

```
INSERT INTO IPL VALUES(5,'DDD','SEHWAG','CHAPEL')
```

DROPPING CONSTRAINT

```
ALTER TABLE IPL
DROP CONSTRAINT PK1

DROP TABLE IPL
```

DEFAULT

Duplicate specifies what values are stored in a column if we do not specify a value for the column when inserting a row/record

EXAMPLE 1:-

```
CREATE TABLE emp
(empno char(4) PRIMARY KEY,
ename varchar(10),  sal
numeric(10,2),  gen char(1)
DEFAULT'M')
```

```
INSERT INTO EMP(EMPNO,ENAME,SAL)
VALUES(1111,'KRISHNA',7000)

SELECT * FROM EMP

INSERT INTO EMP VALUES(2222,'RAMA',9000,'M')
```

ERROR:-

```
INSERT INTO EMP VALUES(3333,'ANJI',5000,'MALE')
```

EXAMPLE 2:-

```
CREATE TABLE PROD
(pno int,  pname
varchar(10),
 mfg datetime DEFAULT getdate() not null,  exp
datetime DEFAULT dateadd(yy,2,getdate()))

INSERT INTO PROD(PNO,PNAME)
VALUES(1,'BRITANIA')
INSERT INTO PROD(PNO,PNAME)
VALUES(2,'GOODDAY')

SELECT * FROM PROD

DROP TABLE PROD
```

CHECK

check constraints enforces intigrity by limiting values that are accepted by a column
A check constraint can be implemented with
IN keyword,or
BETWEEN keyword

EXAMPLE:-

```
CREATE TABLE emp
(empno char(4),
ename varchar(10),
 sal numeric(10,2)CHECK(sal BETWEEN 1000 AND 5000))

INSERT INTO EMP VALUES(1111,'KRISHNA',3000)

INSERT INTO EMP VALUES(2222,'RAMA',5000)

INSERT INTO EMP VALUES(3333,'RAJA',6000)
```

DROPPING CHECK CONSTRAINT

```
ALTER TABLE EMP
DROP CONSTRAINT CK__emp__sal__1C1D2798

INSERT INTO EMP VALUES(3333,'RAJA',6000)
```

ADDING CHECK CONSTRAINT

```
ALTER TABLE EMP ADD CONSTRAINT C1 CHECK(SAL BETWEEN 1000 AND 6000)

INSERT INTO EMP VALUES(4444,'ANJI',1000)
```

IDENTITY

Identity constraint is useful to generate sequential values that uniquly identifies each row within the table

SYNTAX:-

```
IDENTITY(seed,increment)
```

We cannot update the data stored in an identity column

EXAMPLE:-

```
CREATE TABLE DEPT
(deptno int IDENTITY(10,10),
dname varchar(10)NOT NULL)
```

FOREIGN KEY

A Foreign Key is a column whose values match with the Primary Key of the other table

EXAMPLE:-

```
CREATE TABLE dept
(deptno int PRIMARY KEY,
dname varchar(15),  loc
varchar(15))

CREATE TABLE emp
(empno varchar(4)PRIMARY KEY,
ename varchar(15),  sal
numeric(10,2),
 deptno int FOREIGN KEY REFERENCES dept(deptno))

INSERT INTO DEPT VALUES(10,'PRIME MINISTER','DELHI')

INSERT INTO DEPT VALUES(20,'CHIEF MINISTER','HYDERBAD')

INSERT INTO DEPT VALUES(30,'COLLECTER','ONGOLE')

INSERT INTO DEPT VALUES(40,'MRO','NANDYAL')

SELECT * FROM DEPT

INSERT INTO EMP VALUES(1111,'KRISHNA',50000,10)

INSERT INTO EMP VALUES(2222,'YSR',45000,20)

SELECT * FROM EMP
```

ERROR:-

```
INSERT INTO EMP VALUES(3333,'CHANDRA BABU',20000,50)
INSERT INTO DEPT VALUES(50,'VRO','VILLEGE')

INSERT INTO EMP VALUES(3333,'CHANDRA BABU',20000,50)
```

ERROR:-

```
INSERT INTO EMP VALUES(4444,'CHIRANGIVI',25000,60)

INSERT INTO DEPT VALUES(60,'MPTC','MANDLAM')

INSERT INTO EMP VALUES(4444,'CHIRANGIVI',25000,60)

DELETE FROM DEPT
WHERE DEPTNO=30
```

ERROR:-

```
DELETE FROM DEPT
WHERE DEPTNO=50
```

ADDING ON DELETE CASCADE

```
ALTER TABLE EMP
ADD CONSTRAINT FK1 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO) ON DELETE
CASCADE

DELETE FROM DEPT
WHERE DEPTNO=50

SELECT * FROM EMP

DELETE FROM DEPT
WHERE DEPTNO=60

DELETE FROM DEPT
WHERE DEPTNO=10
```

ERROR:-

```
UPDATE DEPT
SET DEPTNO=1
WHERE DEPTNO=20
```

ADDING ON UPDATE CASCADE

```
ALTER TABLE EMP
ADD CONSTRAINT FK2 FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO) ON UPDATE
CASCADE

UPDATE DEPT
SET DEPTNO=1
WHERE DEPTNO=20

SELECT * FROM EMP

ALTER TABLE EMP
DROP CONSTRAINT FK1,FK2
DROP TABLE EMP,DEPT
```

EXAMPLE 2:-

```
CREATE TABLE dept
(deptno int PRIMARY KEY,
dname varchar(15),  loc
varchar(15))

CREATE TABLE emp
(empno varchar(4),
ename varchar(15),
sal numeric(10,2),
deptno int,
 Constraint p1 PRIMARY KEY(EMPNO),
 Constraint F1 FOREIGN KEY(deptno) References dept(deptno),
 Constraint f2 FOREIGN KEY(deptno) References dept(deptno) ON UPDATE
 CASCADE,
 Constraint f3 FOREIGN KEY(deptno) References dept(deptno) ON DELETE
 CASCADE)
INSERT INTO DEPT VALUES(10,'PRIME MINISTER','DELHI')

INSERT INTO DEPT VALUES(20,'CHIEF MINISTER','HYDERBAD')

INSERT INTO DEPT VALUES(30,'COLLECTER','ONGOLE')

INSERT INTO DEPT VALUES(40,'MRO','NANDYAL')

SELECT * FROM DEPT

INSERT INTO EMP VALUES(1111,'KRISHNA',50000,10)

INSERT INTO EMP VALUES(2222,'YSR',45000,20)

SELECT * FROM EMP
SELECT * FROM DEPT

UPDATE DEPT
SET DEPTNO=1
WHERE DEPTNO=10

UPDATE DEPT
SET DEPTNO=2
WHERE DEPTNO=20

INSERT INTO EMP VALUES(3333,'CHANDRA BABU',20000,30)

INSERT INTO EMP VALUES(4444,'CHIRANGIVI',20000,40)

DELETE FROM DEPT
WHERE DEPTNO=30

DELETE FROM DEPT
WHERE DEPTNO=40
```

JOINS

SYNTAX:-

```
select<tab1><col1>,<tab2><col1>,<tab1><col2>,........
from <tab1><jointype><tab2> on<condition>

select<A all-a-half all-a-half pounds a year if all-a-half a-half-life
half-life a-half-life her half-F half-F the half-F the
tab1><col1>,<Atab2><col1>,<Atab1><col2>,..... from
<tab1>as<Atab1><jointype><tab2>as<Atab2> on<join-
condition>
```

JOINT TYPES

1.EQUI /INNER JOIN:-

Matching records from the tables involved in join

2. OUTER JOIN:-

Matching as well as non-matching records from the tables involved if I have in join

A .Left outer join

Displays all the records from the table to the left of join and only matching records from the table to the right of join.

B .Right outer join

Displays all the records from the table to the right of join and only
Matching records from the table to the left of join.

C Full outer join

Displays all the matching as well as non-matching records from the tables that are involved in the join.

3. CROSS JOIN:-

When ever we want to join every record in one table with every record in one table with every other record in another table then we use cross join. The result of cross join is cross product of the number of records in both the tables that are involved in join.

4. SELF JOIN:-

Joining a table itself is called self join.self join is same as equijoin/inner join.

Therefore result set of self-join will contain only matching records from tables involved in the join.

NOTE:-

On both sides of a join-type if we write the same table name then it is

called as self join. Use database select * from emp select * from dept

```
insert into emp(empno,ename) values(1001,'krishna')

select empno,ename,job,hiredate,emp.deptno,dname,loc
from emp join dept on
emp.deptno=dept.deptno

select empno,ename,job,hiredate,dept.deptno,dname,loc
from emp left outer join dept
on emp.deptno=dept.deptno

select empno,ename,job,hiredate,dept.deptno,dname,loc
from emp right outer join dept
on emp.deptno=dept.deptno

select empno,ename,job hiredate,emp.deptno,dname,loc
from emp cross join dept

select e.empno,e.ename,e.mgr as managernumber,m.ename as managername
from emp e join emp m
on e.mgr=m.empno
```

SUB -QUERIES

DEFINATION :-

A SELECT statement if written inside another SELECT statement is called sub-query
The inner SELECT statement that is contained inside another SELECT statement is called as the INNER Query
The SELECT statement is containing another SELECT statement is called the OUTER Query

TYPES OF SUB- QUERIES

We have two types of Sub-Quers,They are
Nested Sub-Quers
Co-related Sub-Quers

SYNTAX:SELECT

```
.......
...............
OPERATOR(SELECT........)
```

If ani nner quert returns only scalar value then use any of the relational operators(>,>=,etc,....),and if the inner query returns multiple values then use logical operators(IN,ANY,etc,....)

Nested Sub - Queries

1.Inner query is executed
2.Result of the inner query is passed to the outer query
3.Outer query is executed

Co- Related Sub - Query

In a Sub-Query if inner query execution is dependent upon the outer Query then it is called Co-Related Sub-Query
1.A row from the outer query is passed to the inner query
2.Inner query is executed and the result is passed to the outer query
3.The outer query is

```
processed use mkrishna select *

from emp
```

Details of all employees who are working in the same department where 'JONES' is working

```
select deptno from
emp where ename
='jones'

select * from emp
where deptno=20

select * from emp where
deptno=(select deptno
from emp                where ename
='jones')
```

Detaile of all those employees who are working with the same job as that of 'CLARK'

```
select job from
emp where
ename='clark'

select * from emp
where job='manager'

select * from emp where
job=(select job
from emp              where
ename='clark')
```

Displays all these employees who are earning same salary as that of 'SCOTT'

```
select sal from
emp where
ename='scott'

select * from emp
where sal=3000

select * from emp where
sal=(select sal
from emp              where
ename='scott')
```

Display of all the employees who are working in same department where 'scott' is working,exclusing 'SCOTT'

```
select deptno from
emp where
ename='scott'

select * from emp
where deptno=20
select * from emp
where deptno=(select
deptno
from emp
where ename='scott')

select * from emp
where ename!='SCOTT' AND deptno=(select
deptno                from emp
where ename='scott')
```

Display the details of all those employees who are working in a department that is situated at 'CHICAGO'

```
select deptno from
dept where
loc='chicago'

select * from emp
where deptno=30

select * from emp where
deptno=(select deptno,loc
from dept                where
loc='chicago') select * from dept
```

Display the details of all those employes who are worked in a department whose name is 'ACCOUNTING'

```
select deptno
from dept
where dname='accounting'

select * from emp
where deptno=10

select * from emp where
deptno=(select deptno
from dept
              where dname='accounting')
```

Displaying the details of the employee who is having highest salary

```
select max(sal)
from emp

select *
from emp
where sal=(select max(sal)
from emp)
```

Display all the details of the employees who is having least salary

```
select * from emp where
sal=(select min(sal)
from emp)

select all deptno
from emp

select deptno
from emp

select distinct deptno
from emp

select distinct deptno
from emp where deptno
is not null

select all job
from emp

select job
from emp

select distinct job
from emp

select distinct job
from emp where job
```

```
is not null select *
from emp

select top 1*
from emp

select top 2*
from emp

select top 15*
from emp

select top 1*
from emp order by
empno asc

select top 2 *
from emp order by
empno desc

select min(empno) from emp where
empno in (select top 2 empno
from emp                order by
empno desc) select * from emp
where empno=(select min(empno)
from emp
             where empno in (select top 2 empno
from emp                             order by
empno desc))
```

Display all the details of the emp who is having 5-th least salary

```
select sal from
emp order by
sal asc

select distinct top 5 sal
from emp
order by sal

select *
from emp
where sal=(select max(sal)
from emp
           where sal in (select distinct top 5 sal
                        from emp
order by sal))
```

Display all the details of 4-th highest paid employee

```
select sal
from emp
```

```
select sal from
emp order by sal
desc

select top 4 sal
from emp order
by sal desc

select min(sal)
from emp
where sal in (select top 4 sal
from emp                order by
sal desc)

select *
from emp
where sal=(select min(sal)
from emp
           where sal in (select top 4 sal
from emp                         order by
sal desc))
```

Display all the details of those employees whose salary is greater then their corresponding department's average salary

```
select deptno
from emp group
by deptno

select sal from
emp where
deptno =10

select avg(sal)
from emp where
deptno=10 select
* from emp

select avg(sal)
from emp             where
deptno=20
```

This is morethan avg sal employees from deptno 20

```
select * from emp where
sal in (select sal
from emp
```

```
              where  deptno=20  and  sal>(select  avg(sal)
from emp                                            where
deptno=20))
```
This is morethan avg sal employees from deptno 10

```
select * from emp where
sal in (select sal
from emp
              where  deptno=10  and  sal>(select  avg(sal)
from emp                                            where
deptno=10))
```
This is morethan avg sal employees from deptno 30

```
select * from emp where
sal in (select sal
from emp
                where deptno=30 and sal>(select avg(sal)
   from emp                                       where
                                            deptno=30))
```
This is morethan avg sal employees from deptno 10,20,30

```
select * from emp where
sal in (select sal
from emp
                       where deptno in(10,20,30) and sal>(select
  avg(sal)                                              from
                                             emp  where deptno in
                                                 (10,20,30)))
                                                 order by deptno asc

select deptno , avg(sal)
from emp group
by deptno

select deptno,avg(sal)
from emp where deptno
is not null group by
deptno

select empno,ename,sal,deptno from
emp e1 where sal >(select avg(sal)
from emp e2             where
e2.deptno=e1.deptno)

select * from etab
select * from dtab

create table etab
(eno int, ename
varchar(20), sal
numeric(5), dno
int)
```

```
insert into etab(eno,ename,dno)
values(1,'aaa',10)

insert into etab(eno,ename,dno)
values(2,'bbb',20)

insert into etab(eno,ename,dno)
values(3,'ccc',10)

insert into etab(eno,ename,dno)
values(4,'ddd',20)

insert into

etab(eno,ename,dno)

values(5,'eee',30) select *

from etab

create table dtab
(deptno int,
dnme
varchar(20), sal
numeric(5))
insert into dtab
values(10,'sales
',5000) insert
into dtab
values(20,'marke
t',6000) insert
into dtab
values(30,'manag
er',7000)
select * from etab
select * from dtab

update etab set
sal=(select sal
from dtab           where
deptno=dno)

select * from etab
select * from dtab
```

VIEWS

DEFINATION :-

A view is a VIRTUAL table whose contents are defined by a SELECT statement

```
1.A View is a virtual table.
2.View doesnot contain any records
```

3.View contain select statement.
4.All the operations that we perform on a table can be performed on a view. 5.View are used to hide data from end user.
6.View are mainly used for security purpose only.

Uses and Benifits of VIEWS

1.Vies are useful for providing security to the data stotred in the table by limiting set of columns/rows
2.Vies can be used to aggregate the information
3.Vies can be used for displaying information by combining it from multiple tables,and projecting the data as if it was retrieved from a single table

Creating A View

```
Create View<View Name>[with encryption] as
<Select Statement>
[with check option]
```

Displaying data through a view

```
Select * (or) col(s)
From <view name>
```

Insert

```
Insert into <Viewname>[(ColName(list))]
Values(<List of Values>)
```

Update

```
Update <Viewname>
Set <Col 1>-<new val>[,<col 2>-<new val>,......]
[Where <Condition>]
```

Delete

```
Delete from <ViewName>
[Where <Condition>]
use mkrishna

 select * from

 emp

 select empno,ename,sal,deptno
 from emp

 create view v1 as select

 empno,ename,sal,deptno from

 emp select * from v1 drop

 view v1

 select empno,ename,sal,deptno
```

```
from emp where
deptno=10

create view ev10 as select
empno,ename,sal,deptno from
emp where deptno=10 select *
from ev10

create view ev20 as select
empno,ename,sal,deptno
from emp where deptno=20 select * from ev20

insert into ev10 values(1001,'rama',9000,10)

select * from emp select * from ev10 insert

into ev10 values(1010,'ranga',8000,20)

select * from ev10
select * from ev20
update ev10 set
deptno=20 where
ename='rama'

select * from ev10 select * from ev20 insert

into ev20 values(1002,'raja',1500,30)

select * from ev10

select * from ev20

select * from emp

update ev20 set
deptno=10 where
ename='rama'

update ev10 set
deptno=20 where
ename='rama'

select * from ev10
select * from ev20

update ev10 set
deptno=10 where
ename='ranga'

select empno,ename,sal,deptno
from emp where
deptno=30

create view ev30 as select

empno,ename,sal,deptno from emp where deptno=30

with check option insert into ev30
```

```
values(1020,'krishna',2000,20) insert into ev30
values(1020,'krishna',2000,30) select * from
ev30

select * from ev10
select * from ev20

update ev10 set
deptno=30 where
ename='rama'

select * from ev10
select * from ev20
select * from ev30
update ev30
set deptno=10
where ename='rama'

update emp set
deptno=10 where
ename='rama'

select * from ev10
select * from ev20
select * from ev30
drop view ev10

create view ev10 as select
empno,ename,sal,deptno from
emp where deptno=10 with
check option sp_helptext ev10

alter view ev20 as select
empno,ename,sal,deptno from
emp where deptno=20 with
check option sp_helptext ev20

alter view ev10 with encryption as
select empno,ename,sal,deptno from
emp where deptno=10 with check
option sp_helptext ev10

alter view ev20 with encryption as
select empno,ename,sal,deptno from
emp where deptno=20 with check
option sp_helptext ev20

alter view ev30 with encryption as
select empno,ename,sal,deptno from
emp where deptno=30 with check
option sp_helptext ev30
```

INDEXES

INTRODUCTION:-

•Indexes in a database are similar to that of indexes in books.In a book an index allows us to locate the information quickly without reading the entire book. •In a DB the index allows us to locate the information in a table without scanning the entire table. •SQL Server automatically creates indexes for certain types of constraints(eg.PRIMARY KEY and UNIQUE).

ADVANTAGES :-

Data retrieval is fast
Speed up the joins between tables
Enforces the uniqueness of the data
Improve the speed of execution of quiries

DISADVANTAGES :-

It takes disk space to store the indexes
Data modification takes time as indexes must be updated

Guidelines For Creating Indexes

A Column that is frequently used in a SELECT list and in a WHERE clause
A Column that will be used with GROUP BY or an ORDER BY clause to sort the data
A Column used in a join such as a FOREIGN KEY column
A Column is used as a PRIMARY KEY

Types of INDEXES

SQL Server provides two types of INDEXES,they are
1.Clusterd Indexes
There can be only ONE clustred index on a table/view
2.Non-Clustered Index A maximum of **249** non-clustred indexes can be created on a table/view

T-SQL

DECLARATION STATEMENTS

```
DECLARE @<VARNAME1>[AS]<DATATYPE>[,@<VARNAME2>[AS]<DATATYPE>,.....]
```

INITIALIZE THE CONTENTS OF THE VARIABLE

```
SET----->ONE VARIABLE
SELECT----->ONE/MORE THAN ONE VARIABLE
```

DISPLAYING THE CONTENTS OF THE VARIABLE

```
PRINT['MESSAGE STRING'+]@<VARNAME>
```

EXAMPLES:-

```
declare @x as int
set @x=10

declare @a int,@b int
set @a=100 set @b=200

declare @a int,@b int
select @a=10,@b=20

declare @a as int
set @a=100 print
@a

declare @p int,@q int
select @p=10,@q=20
print @p print @q
```

ERROR:
```
declare @p int,@q
int select
@p=10,@q=20 print
@p,@q
```

```
declare @p int,@q int
select @p=10,@q=20
print @p+@q

declare @p int,@q int
select @p=10,@q=20
print cast(@p as char)+'    '+cast(@q as char)

declare @p int,@q int
select @p=10,@q=20
print cast(@p as char(2))+'          '+cast(@q as char(2))

declare @eno int,@ena varchar(20)
select @eno=100,@ena='ram' print
@eno print @ena

declare @eno int,@ena varchar(20)
select @eno=100,@ena='ram'
print 'employee number is ::'+cast(@eno as char(4))
print 'employee name is ::'+@ena

declare @n1 as int,@n2 as int
declare @sum int,@diff int
```

```
select @n1=10,@n2=30 set
@sum=@n1+@n2
set @diff=@n1-@n2
print @sum print
@diff

declare @n1 as int,@n2 as int
declare @sum int,@diff int
select @n1=10,@n2=30 set
@sum=@n1+@n2 set @diff=@n1-
@n2
print 'sum of two numbers 10 and 30 is ::'+cast(@sum as char(3)) print
'difference of two numbers 10 and 30 is ::'+cast(@diff as char(3))

declare @n1 as int,@n2 as int
declare @sum int,@diff int
select @n1=10,@n2=30 set
@sum=@n1+@n2 set @diff=@n1-
@n2
print cast(@n1 as char(2))+'+'+cast(@n2 as char(2))+'='+cast(@sum as
char(3))
print cast(@n1 as char(2))+'-'+cast(@n2 as char(2))+'='+cast(@diff as

char(3)) select getdate()

declare @dno as int,@mno as int,@yno as int,@dna as char(10)
select @dno=datepart(dd,getdate()) select
@mno=datepart(mm,getdate()) select
@yno=datepart(yy,getdate()) select
@dna=datename(dw,getdate()) print 'today date'
print 'day number ::'+cast(@dno as char(5)) print
'month number ::'+cast(@mno as char(5)) print 'year
number ::'+cast(@yno as char(5)) print 'today weak
name is ::'+cast(@dna as char(10))

declare @na varchar(20),@country as varchar(10)
select @na='anji',@country='india'
print 'my name is ::'+@na print 'i
live in ::'+@country
```

WRITE A T-SQL PGM TO DISPLAY THE TOTAL NUMBER OF DAYS COMPLETSES

FROM JAN 1,2011 TILL TODAY select getdate()

```
declare @nod as int
select @nod=datediff(dd,'1/1/2011',getdate())
print 'total no of days completed in this year till todays
date::'+cast(@nod as char(10))

select ename,sal,job,deptno
from emp where mgr=7839

declare @ena varchar(20),@sal numeric(10),@job varchar(10),@deptno
numeric(10)
```

```
select @ena=ename,@sal=sal,@job=job,@deptno=deptno
from emp where empno=7839
print 'employee name is ::'+cast(@ena as char(10))
print 'employee sal is ::'+cast(@sal as char(10)) print
'employee job is ::'+cast(@job as char(10)) print
'employee deptno is ::'+cast(@deptno as char(10))
select * from emp
```

FUNCTIONS

CREATING A FUNCTION

SYNTAX:-

```
CREATE FUNCTION[<USER NAME>]<FUN NAME>(<PARAMETERS>)
RETURNS<DATATYPE> AS
{
      BEGIN
      <T-SQL Statement>
      <Return Statement>
      END
}
```

CALLING A SCALAR VALUED FUNCTIONS

SYNTAX:-

```
SELECT[<USER NAME>]<FUNCTION NAME>(<Values To The List of Parameters>)
```

CALLING A TABLE VALUED FUNCTIONS

SYNTAX:-

```
      SELECT * FROM [<USER NAME>]<FUNCTION NAME>(<Values To The List of
Parameters>)
```

EXAMPLE 1:
```
create function getname(@eno int)  returns varchar(100) as
begin  declare @ename as varchar(100) if (select count(*) from emp
where empno=@eno)>0  begin   select @ename=ename from emp where
empno=@eno set @ename='The ename of the given empno is ::'+@ename
      end
else  begin set @ename='The ename of the given empno is not into the
      emp table.'
      end
            return @ename
end


select dbo.getname(12345)
```

EXAMPLE 2:create function

```
getsal(@eno   int)        returns
char(100) as
begin  declare @esal as char(100) if(select count(*) from emp
where empno=@eno)>0  begin   select @esal=sal from emp where
empno=@eno set @esal='the sal of the given emono is ::'+@esal
if @esal is null   set @esal=0
      end
else  begin   set @esal='The Given sal is not into the emp table'
      end
    return @esal end

select

dbo.getsal(100000)

select dbo.getsal(1234)

select dbo.getsal(7788)

select dbo.getsal(7839)
```

EXAMPLE 3:create function

```
getcomm(@eno   int)        returns
char(100) as
begin declare @ecom as char(100) if(select count(8) from
emp where empno=@eno)>0 begin   select @ecom=comm from emp
where empno=@eno set @ecom='The comm of given empno is
::'+@ecom if @ecom is null   set @ecom=0
      end
else  begin set @ecom='The comm of given empno is not into the emp
              table.' end
      return @ecom
end select


dbo.getcomm(1000) select


dbo.getcomm(7788) select


dbo.getcomm(7654) select


dbo.getcomm(1234)
```

EXAMPLE 4:create function

```
netsal(@eno   int)        returns
char(100) as
begin   declare @nsal as char(100) if (select count(*) from
emp where empno=@eno)>0   begin    declare @esal as
char(100),@ecom as char(100) select @esal=dbo.getsal(@eno)
select @ecom=dbo.getcomm(@eno)
            select @nsal='The net salary of given empno is ::'+@nsal
```

```
                set @nsal=@esal+@ecom
          end
    else   begin set @nsal='The net salary of ginven empno is not into
          emp table.' end    return @nsal
    end


    select dbo.netsal(7788)

    select dbo.netsal(10000)

    select sal,comm,(sal+comm)
    from emp where

    empno=7788 select

    * from emp
```

EXAMPLE 5:create function

```
    getemps(@dno int)
    returns table
    return(select empno,ename,sal,comm,deptno
    from emp where @dno=deptno)

    select * from
    dbo.getemps(10)

    select * from
    dbo.getemps(20)

    select * from
    dbo.getemps(30)

    select * from
    dbo.getemps(40)
```

How to display highestwise salary from the emp table

```
    select top 2 sal
    from emp order by
    sal desc

    select distinct top 2 sal
    from emp order by
    sal desc

    select min(sal)
    from emp
    where sal in (select distinct top 2 sal
          from emp       order by sal desc)

    select * from
    emp
    where sal=(select min(sal)        from emp    where
          sal in (select distinct top 2 sal
          from emp          order by sal desc))
```

```
create function highsal(@val int)
 returns table return(select *        from emp    where sal=(select
   min(sal)          from emp       where sal in (select distinct
   top(@val) sal                from emp               order by sal
                                                              desc)))

select * from dbo.highsal(1)
select * from dbo.highsal(3)
select * from dbo.highsal(5)
select * from dbo.highsal(7)
select * from dbo.highsal(9)
```

How to display leastwise salary from the emp table

```
select top 5 sal
from emp

select distinct top 5 sal
from emp order
by sal asc

select max(sal)
from emp
where sal in (select distinct top 5 sal
      from emp       order by sal asc)

select *
from emp
where sal=(select max(sal)        from emp    where
      sal in (select distinct top 5 sal
      from emp          order by sal asc))

create function lowsal(@val int)
returns table
return(select *
from emp
         where sal=(select max(sal)            from emp   where sal in
            (select distinct top (@val) sal              from emp
            order by sal asc)))

select * from dbo.lowsal(2)
select * from dbo.lowsal(4)
select * from dbo.lowsal(6)
select * from dbo.lowsal(8)
select * from dbo.lowsal(10)
```

Write the function to display the addition,multification,division and substrction

```
declare @a int,@b int,@m numeric(30)
select @a=0,@b=9 while @a<10 begin
set @a=@a+1 select @m=@a-@b
print cast(@b as char(30))+'-'+cast(@a as char(5))+'='+cast(@m as
char(9)) select @m=@a/@b
```

```
print cast(@b as char(30))+'/'+cast(@a as char(5))+'='+cast(@m as
char(9)) select
@m=@a*@b
print cast(@b as char(30))+'*'+cast(@a as char(5))+'='+cast(@m as
char(9)) select @m=@a+@b
print cast(@b as char(30))+'+'+cast(@a as char(5))+'='+cast(@m as
char(9))
end
```

Write the function to display the details of given empno

```
create function date(@eno int)
returns char(100) as
begin
declare @doj as char(100)
if(select count(*) from emp where empno=@eno)>0
begin
select @doj=hiredate from emp where empno=@eno
set @doj='The hiredate of given empno is ::'+@doj
if @doj is null
set @doj=0
end else
begin
set @doj='The given empno is not into the emp table.'
end return @doj end

select dbo.date(7788)
select dbo.date(1001)
```

Write the function to display the details of given job

```
create function krishna(@eno int)
returns varchar(100) as begin
declare @job as varchar(100)
if(select count(*) from emp where empno=@eno)>0
begin
select @job=job from emp where empno=@eno
set @job='The job of given empno is ::'+@job
if @job is null set @job=0 end else begin
set @job='The given empno is not into the emp table.'
end return @job end select dbo.krishna(7839)
```

Write The function to displaying the details of same deptno at a time

```
create function dpt(@dno int)
returns char(100) as
begin
declare @dept as char(100)
if(select count(*) from dept where deptno=@dno)>0
begin
select @dept=dname from dept where deptno=@dno set
@dept='The hiredate of given deptno is ::'+@dept
```

```
if @dept is null
set @dept=0 end
else begin
set @dept='The given deptno is not into the emp table.'
end return @dept
end select
dbo.dpt(10) select
dbo.dpt(20) select
dbo.dpt(30) select
dbo.dpt(40)
```

STORED PROCEDURES

DEFINATION:-

A store procedure is a set of T-SQL statements stored as a unit of code and is a server side component that resides on the server

Advantages of Stored Procedures

- Improve the Query Performance
- Reduces the new window traffic
- Reusability,i.e,once a stored procedure is created it can be utilized by other users too

Types of Stored Procedures

1. System Stored Procedure
2. User Defined Stored Procedure

System Stope Procedure

SQL Server provides a set of system stored procedures
Some of The system stored procedures are
sp_help,sp_helpdb,sp_rename,sp_renamedb,sp_helptext,etc.....

User Defined Stored Procedures

User defined store procedures are once that are created by users

Creating a Stored Procedures

SYNTAX:-

```
CREATE PROCEDURE<procedures>[(list of parameters)] AS
BEGIN
<T-SQL statement(s)>
--------------------
END
```

ALTERING A PROCEDURE

SYNTAX:-

```
      ALTER PROCEDURE<procedure name>[(parameter-list)] AS
      BEGIN
            T-SQL Statements
      END
```

DELEATING A STORE PROCEDURE

SYNTAX:-

```
DROP PROCEDURE<procedure name>
```

Store Procedure With Output Parameters

A user defined store procedure is also capable of returning values as a result to the user and thease type of procedures are called as the store procedure with output parameters

EXAMPLES:use

```
mkrishna select

* from emp

create procedure p1 as
begin select *
from emp end
execute p1

create procedure p2(@dpt int)
as begin select * from emp
where @dpt=deptno end

exec p2 10
exec p2 20
exec p2 30
exec p2 40

create procedure p3(@dpt char(100)) as
begin
```

```
if(@dpt in
(10,20,30)) select *
from emp where
@dpt=deptno
else
print 'no emp works in the given

dept' end exec p3 50

create procedure p4(@dno1 int,@dno2 int) as
begin
if(select count(*) from emp where deptno in
(@dno1,@dno2))>0 select * from emp
where deptno in (@dno1,@dno2)
else
print 'no employees work in the given dept'
end

exec p4 10,20 exec p4 20,30 exec p4

50,40 exec p4 40,30 select * from emp

where job='manager' select * from emp

where job='salesman' select * from emp

where job='clerk' select * from emp

where job='presedent' select * from

emp where job='analyst'

create procedure p5(@job varchar(10)) as
begin select *

from emp where

@job=job end exec

p5 'analyst' exec

p5 'manager' exec

p5 'salesman' exec

p5 'presedent'

create procedure p6(@job varchar(10)) as
begin
if(select count(*) from emp where
@job=job)>0 select * from emp where @job=job
else
print 'The given job is not

exists' end exec p6 'jindabad'

exec p6 'caption' exec p6

'manager' exec p6 'presedent'

exec p6 'salesman'

create procedure p7(@dno char(10),@job char(100)) as
```

```
begin
if(select count (*) from emp where @job=job and @dno=deptno)>0
select *
from emp
where deptno=@dno and @job=job
else
print 'The given job is not exists'
end

exec p7 10,'manager'
exec p7 40,'manager'

create procedure p8(@dno char(10),@job char(100)) as
begin
if(select count (*) from emp where @job=job and
@dno=deptno)>0 select * from emp
where deptno=@dno and @job=job
else
print 'no emp works into the combination of given deptno and

job' end exec p8 10,'salesman' exec p8 20,'presedent'

select * from emp where sal
between 1000 and 2000

create procedure p9(@sal1 char(100),@sal2 char(100)) as
begin
if(select count (*) from emp where @sal1=sal and @sal2=sal)>0
select *
from emp
where sal between @sal1 and @sal2
else
print 'The given sal is not

exists' end exec p9 1000,2000 exec

p9 20000,10000

create procedure p10(@sal1 char(100),@sal2 char(100)) as
begin
if(@sal1<=@sal2
) select * from
emp
where sal between @sal1 and
@sal2 else select * from emp
where sal between @sal2 and @sal1
end exec p10

2000,1000 exec

p10 1000,2000

exec p10
```

```
5000,1000 exec

p10 5000,4000

create table logtab
(uname varchar(20),
pword varchar(20))

insert into logtab
values('krishnanji555','anji')

insert into logtab
values('krishnanji555','9790011111')

insert into logtab
values('aramachandrareddy555','9790011111')

insert into logtab
values('kotaswamyreddy9','kothapalli')

insert into logtab

values('krishnanji555','9790011111'

) select * from logtab truncate

table logtab

create procedure p11(@un varchar(20),@pw varchar(20)) as
begin insert into logtab

values(@un,@pw) end exec

p11 'krishna','anji'

select * from logtab exec

p11 'rama','hanuma' select

* from logtab exec p11

'raja','sekar' select *

from logtab exec p11

'laxman','bharata' select

* from logtab exec p11

'krishna','anji' select *

from logtab truncate table

logtab

create procedure p12(@un varchar(20),@pw varchar(20)) as
begin
if(select count (*) from logtab where uname=@un)>0 print
'user name is already exists try another new user name'
```

```
else insert into
logtab
values(@un,@pw)
end


exec p12 'krishna','anji'

select * from logtab exec

p12 'krishna','anji'

select * from logtab exec

p12 'rama','anji' select *

from logtab exec p12

'bema','pandava' select *

from logtab exec p12

'arjuna','pandava' select

* from logtab truncate

table logtab

create procedure p13 (@un varchar(20),@pw varchar(20))
as begin if(@un<>@pw) begin
if(select count(*) from logtab where uname=@un)>0 print
'user account already exists try with another name'
else begin insert
into logtab
values(@un,@pw)
print 'account created
successfully' end end else
print 'Should not give username and password as

same' end exec p13 'krishna','krishna' select * from

logtab

exec p13 'rama','krishna'

select * from logtab exec

p13 'rama','rama' select *

from logtab exec p13

'hanuma','krishna' select

* from logtab exec p13

'a','a'
```

TRIGGER

DEFINATION

A Trigger is a special kind of SP which will be executed automatically based on the user events

Generally a user will perform INSERT,UPDATE and DELETE operations.

TRIGGER EVENTS

An INSERT event will be generated whenever the user performs an insertion operation on a table/view,and this event executes automatically the code within the trigger that

An UPDATE event will be generated whenever the user performs an update operation on a table/view,and this event executes automatically the code within the trigger that

An DELETE event will be generated whenever the user performs an deletion operation on a table/view,and this event executes automatically the code within the trigger that

Use of Triggers

•Trigger are useful to implement high-level complex business logic in a DB

•Triggers are useful for automating data uodates

•Triggers are useful to display user friendly messages

Types of Triggers

SQL Server supports TWO types of Triggers,they are

1.FOR/AFTER Trigger

2.INSTEAD OF Trigger

FOR/ AFTER Trigger

Specifing AFTER is same as specifing FOR AFTER Trigger can be specified only on tables

IF we define a FOR/AFTER Trigger on a table,the operations such as INSERT,UPDATE AND DELETE will not be performed on the table first and then the code within the trigger will be executed

INSTEAD OF Trigger

INSTEAD OF Trigger can be defined on a tables as well as on views

IF we define an INSTEAD OF Trigger on a table,the operations such as INSERT,UPDATE AND DELETE will not be performed on the table,first the code within the trigger will be executed

Logical / Pseudo Tables

When an insert,Update,or Delete trigger files this event creates one or more logical tables in SQL servers memory. There are two logical tables,they are INSERTED and DELETED

INSERTED Tables

- An Insert or Update trigger create an INSERTED logical table.
- The INSERTED table contains the record that is newly added or modified

DELEATED Table

- The Update and Delete trigger creates DELEATED logical table.
- The DELEATED table contains the original record before modification i.e,the deleated record.

CREATING A TRIGGER

SYNTAX:-

```
CREATE TRIGGER<trigger name>ON<table name>
FOR/AFTER/INSTEAD OF INSERT/UPDATE/DELETE AS
BEGIN
       T-SQL
```

statements end

EXAMPLES:use mkrishna

```
select * from emp drop

table emp

drop table cust drop
table        oldcust
```

CREATING A TABLE

```
create table cust
(ano numeric(20),
aname varchar(20),
gmail varchar(30),
pno numeric(15), bal
numeric(20))
```

INSERT INTO RECORDS

```
insert into cust
values(30760626948,'krishna
reddy','krishnanji555@gmail.com',9676104345,1500000000)

insert into cust
values(10997283502,'raja
reddy','mraja555@facebook.com',9676172172,2000000000)

insert into cust
```

```
values(5000643224,'mkr','mkr555@yahoo.com',9790086482,1234567890)

insert into cust
values(10992929287,'ysr','ysr555@reddif.com',9876543210,9876543210)

select * from cust

insert into cust
values(98709870980,'yjr','yjr555@sakshi.com',9000055555,9876543234)

select * from cust
```

DELETE FROM RECORDS

```
delete from cust

where aname='ysr'

select * from cust
```

CREATING ANOTHER TABLE

```
create table oldcust
(aname varchar(20),
gmail varchar(30),
pno numeric(15))
```

APPLYING TRIGGERS

```
create trigger a1 on
cust after delete as
begin
  insert into oldcust
select aname,gmail,pno
from deleted   select *
from inserted   select
* from deleted end

select * from cust
select * from oldcust
```

DELETE RECORD

```
delete from cust where
aname='raja reddy'

select  *  from  cust

select * from oldcust
```

DELETE RECORD

```
delete from cust
where aname='mkr'
```

```
select  *  from  cust
select * from oldcust
```

DELETE RECORD

```
delete from cust where
aname='krishna reddy'

select  *  from  cust
select * from oldcust
```

DELETE RECORD

```
delete from oldcust
where aname='mkr'

select  *  from  cust
select * from oldcust
```

APPLYING TRIGGER

```
create trigger a2 on oldcust
instead of delete as
begin
print 'the delete operation is not allowed into this session'
end

select  *  from  cust
select * from oldcust
sp_help  cust
```

DELETE RECORD

```
delete from oldcust
where aname='raja
reddy'

select  *  from  cust
select * from oldcust
```

ALTERING TRIGGER

```
alter trigger a2 on oldcust
instead of delete as
begin
select * from inserted
select * from deleted
end

select  *  from  cust
select * from oldcust
```

DELETE RECORD

```
delete from oldcust
where aname='raja
reddy'

select * from cust
select * from oldcust

drop table oldcust
drop   table   cust
```

CREATE TABLE

```
create  table  cust
(ano   numeric(20),
aname varchar(20),
gmail varchar(30),
pno   numeric(15),
bal   numeric(20))
select * from cust
```

ALTER TABLE

```
alter table cust
add stat char(1) default 'o' check(stat in ('c','o'))

select * from cust
```

INSERT RECORDS

```
insert into cust(ano,aname,gmail,pno,bal)
values(30760626948,'krishna
reddy','krishnanji555@gmail.com',9676104345,1500000000)

insert into cust(ano,aname,gmail,pno,bal)
values(10997283502,'raja
reddy','mraja555@facebook.com',9676172172,2000000000)

insert into cust(ano,aname,gmail,pno,bal)
values(5000643224,'mkr','mkr555@yahoo.com',9790086482,1234567890)

insert into cust(ano,aname,gmail,pno,bal)
values(10992929287,'ysr','ysr555@reddif.com',9876543210,9876543210)

select * from cust

insert into cust(ano,aname,gmail,pno,bal)
values(98709870980,'yjr','yjr555@sakshi.com',9000055555,9876543234)
select * from cust

select ano,aname,gmail,pno,bal
from cust

create view v1 as select
ano,aname,gmail,pno,bal
from  cust  WHERE
STAT ='O' select
```

```
*    from    v1
```

DELETE RECORD

```
delete  from  v1  where

aname='raja       reddy'

select * from v1 select

* from cust drop trigger

t1
```

APPLY TRIGGER

```
create trigger t1 on v1
instead of delete as
begin   update cust   set bal=0
,stat='c'
  where aname in (select aname from deleted)
end
```

DELETE RECORD

```
delete from v1
where aname='ysr'

select * from cust
select * from v1

delete from v1 where

aname='krishna reddy'

delete from v1
```

TRANSACTIONS

DEFINATION

ATransaction is a sequence of operations performed as a single unit.A unit of work is said to be successful transaction if it satisfies the following properties

1.AUTOMICITY
2.CONSISTENCY
3.ISOLATION
4.DURABILITY

AUTOMICITY

It means that in a transaction either ALL the operations in a transaction should be performed or NONE

CONSISTENCY

This ensures that the operations performed on a DB if related must reflect the same data everywhare

ISOLATION

This indicates that each operations performed on a DB is performed in an isolated manner that doesnot effect any other transaction.

DURABILITY

The changes made to the database should be applied on the database permanently and must be long lasting

Types of Transaction

1. Explicit
2. Implicit

Explicit Transaction

- Thease transactions allow users to have control on data operations
- Every Explicit transaction requires a begining point and an ending point
- The begining point can be specified using the statement "BEGIN TRANSACTION"
- The ending point of this type of transaction can be specified by any of the following statements
"COMMIT TRANSACTION"
"ROLLBACK TRANSACTION"
    - COMMIT is the statement issued to make the changes applied on the DB permanent
    - ROLLBACK is the statement used to restore the DB to the previous state

Implicit Transaction

- This is the default mode of transaction.
- Every statement issued will be committed in this mode

```
use mkrishna

select * from

cust drop table

cust


create table cust
(acno int, cname
varchar(20), phno
numeric(20),
email
varchar(30),
status char(10) check (status in ('s','c')),
bal numeric(15),
pass char(10),
trans char(5))
select * from cust

insert into cust
select
1,'krishna',9676104345,'krishnanji555@gmail.com','s',10000,9790011111,5
insert into cust
select 2,'dgp',9676088909,'arcr55555@gmail.com','c',1500,9791111111,5
insert into cust
select 3,'raja',9676088991,'aluva555@gmail.com','s',12000,9790000000,5
insert into cust
select 4,'anji',9676088937,'anjaneya55555@gmail.com','c',7000,'krishna',5
insert into cust
select 5,'rama',9866695495,'ramarangareddy555@gmail.com','s',10000,'anji',5

select * from cust




create procedure SBI(@acno int,@pass char(10),@wdrl numeric(5)) as
begin
      declare @stat as char(1),@bal as char(55),@ata char(55),@tran
char(100),@a char(55),@cname char(55)
      select @cname=(select cname from cust where @acno=acno and
@pass=pass)       select @stat=(select status from cust where @acno=acno
and @pass=pass)       select @bal=(select bal from cust where @acno=acno
and @pass=pass)       select @tran=(select trans                    from
cust
                    where @acno=acno and @pass=pass)
if(select count(*)from cust where @acno=acno)>0
begin
      if (select count(*) from cust where
@pass=pass)>0                begin            if
@stat='s'                    begin                  if
@bal>=500                         begin
                         select @ata=@bal-
500                      if @ata>=@wdrl
begin
                              select @ata=@bal-@wdrl
                              update cust
set bal=@ata
                              where @acno=acno and @pass=pass
print 'hello: '+@cname
                              print 'Your transaction is successfull.'
print 'Your remaining balance is :'+@ata
print getdate()
-----------------------------------------------------------------------------
if @tran<=5                                begin
```

```
select @a=@tran-1                                if @a>=0
begin                                          select @a=@tran-1
update cust                                          set trans=@a
                                         where @acno=acno and @pass=pass
print 'The no of transactions remaining
is:'+@a
                                         end
else
                                         print 'Your transactions are completed.You
cannot perform the transactions till today.Please visit
tomorrow.'                                     end
else
                                    print 'transactions over'
-----------------------------------------------------------------------------------
end                     else                             begin
                         print 'You have no suffecient balance for this
transaction.'
                         print 'Your savings account balance is:'+@bal
print 'Your maximum transaction for this account
is:'+@ata
                         print getdate()

end                     end
else
                    print 'You have no minimum balance.'
               end
else
begin             if
@bal>=5000
begin
                    select @ata=@bal-5000
                if @ata>=@wdrl
begin
                         select @ata=@bal-
@wdrl                          update cust
set bal=@ata
                         where @acno=acno and @pass=pass
print 'hello: '+@cname
                         print 'Your transaction is successfull.'
print 'Your remaining balance is :'+@ata
print getdate()
-----------------------------------------------------------------------------------
--------------------                       if
@tran<=5                                   begin
select @a=@tran-1                               if
@a>=0                                        begin
select @a=@tran-1
update cust
set trans=@a
                                         where @acno=acno and @pass=pass
print 'The no of transactions remaining is:'+@a
                                         end
else
                                         print 'Your transactions are completed.You
cannot perform the transactions till today.Please visit tomorrow.'
```

```
                              end
else
                              print 'transactions over'
-----------------------------------------------------------------------------
-------------------
end                 else
begin
                     print 'You have no suffecient balance for this
transaction.'
                     print 'Your current account balance is :'+@bal
print 'Your maximum transaction for this account is:'+@ata
                     print
getdate()                     end
end           else
               print 'You have no minimum
balance'           end           end      else
          print 'You are entered incorrect password,Please try
again.'      end else
     print 'The given account no is invalid, Please try correct account
number.'
end


exec SBI 1,9790011111,500

exec SBI 2,9791111111,500

exec SBI 3,9790000000,500

exec SBI 4,krishna,1000

exec SBI 5,anji,5000
```