--CLASS-1

--SQL- Structured query Language.

--PL/SQL - Programming Language SQL.

--Q.What is data?

--Collection of meanigful information.

--OR

--Collection of record information.

--Q.What is Database?

--It is collection of data in file format.

--For ex: Excel,word,text file, notepad etc.

--Disadvantage

--It stores very less amount of data.

--Data accesibility is slower.

--No relationship between two files.

--Q.What is RDBMS(Relatinal Data Base Managment DB)?

--It is collection of table related information.

--It stores huge amount of data and to extract this data we have simple language called as SQL.

--There is relation between two or more tables.

--Q.What is table?

--It is collection of rows and columns.

--Diffrent flavours of RDBMS

--1.SQL Server - Microsoft

--2.SQL Developer - Oracle

--3.Tera Data --Teradata

--4.DB2 - IBM

--5.Mongo DB

--6.Postgrey SQL

--7.MySQL etc.

--There are Two types of databases

--1.System Defined DB

--2.User defined DB.

--There are four types of system defined DB.

--1.Master -- By default DB of SQL

--2.Model

--3.Msdb

--4.temp DB

--SQL is not case sensitive language.

--EX: meaning of SCODEEN is as same as scodeen.

--Colors

--Blue - System defined keywords

table,create,select etc.

--Pink - system defined functions

min,max,sum,avg etc

--Q.How to create database?

create database Testing21

create database Revison

--Q.How to excecute SQL statements.

--1.By using Execute tab from Top

--2.By pressing F5 key from keyboard.

--Q.How to navigate user created DB or Testing21 DB?

  Use Testing21

  use Revison

--Comments

--Two types of comments

--1.Single line comment (--)

--2.Multiple line comment (/* multple statement */)

--1.

--My name is SQL

--2.

```
/*i am studying SQL
which is important for Database
by using sql we can become ETL,BI and Bigdata analyst.*/

--Data Types
--Type of data/value of an object can hold is known as data type



--Diifrent type of Data types in SQL
--1.Numeric Data Type
--2.Approximate Numeric Data type
--3.String or Charecter Data type
--4.Date and Time Data Type.

--CLASS-2
--1.Numeric data type

--1.BIT
--it stores value 0 or 1.

--2.TINYINT
--It will store the value ranging from 0 to 255. -    128 64 32 16 8 4 2 1 --
255
declare @a tinyint                                          --       0  0  1    1
1 1 1 0
```

```
set @a = 256

print(@a)


--3.SMALLINT

--it will store value ranging -32768 to 32767

declare @b smallint                                          --       0  0  1
        1 1 1 1 0

set @b = 32767

print(@b)

--4.Decimal

--an exact fixed point number

declare @c decimal                                       --       0  0  1     1
1 1 1 0

set @c = 326543584353453893.47594389

print(@c)


--5. INT

--it stores an integer value i.e. ranging from -2147483648 to 2147483647

declare @d int

set @d = 2147483647

print(@d)


--Exception :Arithmetic overflow error converting expression to data
type int.

declare @e int
```

```
set @e = 2147483648
print(@e)

--2.Approximate numeric data type
--1.Float
--it will store floating point number range is -1.8E to 308 to 1.8E to 308
--for Ex: 8.2345, 0.9876 etc
declare @f float
set @f = 21474836486584859353253465342897 64
print(@f)

--2.Real
--it will also store an floating point numbers -3.40E to 38 to 3.40E to 38
declare @g real
set @g = 2147483648658485935325346534289764
print(@g)

--3.String or charecter data types
--1.Char -A-Z to a-z ,0-9,special chareectrs
--Static memory allocation and it size 8000 charecters.
--For EX: char(5) - it will block or cease 5 blocks of memory - M, F remaing
4 blocks is waistage.

--2.varchar -- A-Z,a-z and special charecters and numbers i.e 0-9
--Dynamic memory allocation and it size 8000 charecters.
```

--For EX: varchar(5) - it will not block or cease 5 blocks of memory insted it will use as per column - M, F remaing 4 blocks it will release.

```
--char
declare @val char(8000)
set @val ='AMIT@@@'
print @val
print datalength(@val)
print len(@val)


--Varchar
declare @val1 varchar(8000)
set @val1 ='AMIT@@@'
print @val1
print datalength(@val1)
print len(@val1)
 --3.nchar
 --Static memory allocation and it size 4000 charecters.
 --(fixed,4000(2 bytes)


--4.nvarchar
--Dynamic memory allocation and it size 4000 charecters.
--Dynamic(variable lenght),4000(2 bytes)


--char
declare @val3 nchar(4000)
```

set @val3 ='AMIT'

print datalength(@val3)

print len(@val3)

--Varchar

declare @val4 nvarchar(4000)

set @val4 ='Amit' -- if we want to store A then we need 2 blocks of memnory

print @val4

print datalength(@val4)

print len(@val4)

--CLASS-3

--4.Date time data type

--1.date --

--you can define or insert the date in multiple formats like YYYY/MM/DD, DD/MM/YYYY ,MM/DD/YYYY etc.

date

select GETDATE()

--2 Time

-- it will allow you to insert the time, and format is HH:MM:SS

time

--3.Datetime

--it will allow you to insert the date as well time format is YYYY-MM-DD HH:MM:SS:MS

--ex:2021-08-17 08:16:13.327

```
datetime
 --4.Timestamp
 --It will combine date and time in the form  of numbers.
 --ex:2021-08-17 08:16:13.327

 timestamp

 use Testing21

 --Q.How to create a table?
 create table Employee (
 EID int,
 EMP_NAME Varchar(20),
 EMP_LOC varchar(20),
 EMP_SAL int,
 EMP_DOJ Datetime)

 --Q.How to select Table ?
 select * from Employee

 --star (*) - all the columns from table.

 select EID,EMP_NAME from Employee

 --By using select statement we can select all the columns by using *
```

--We can select specific columns by specifying particular column names.

--Q.How to insert the data or values into a table?

--By using two methods we can insert data into a table

--METHOD - I

--It will allow you to insert values into a table as per the sequence created while creating table.

--If you miss the sequence or order then it will through an exception.

insert into Employee values (1,'Praveen','PUNE',2500,GETDATE())

insert into Employee values (2,'Amit','Mumbai',2000,'2020-10-11 08:52:14.420')

insert into Employee values (3,'Sumit','',4000,'2020-10-11 08:52:14.420')

insert into Employee values (4,'Rohan','Jaipur','','2017-10-09 10:25:14.420')

insert into Employee values (7,'Ronit','Udaipur',3000,'')

--below statement through an exception : Column name or number of supplied values does not match table definition.

insert into Employee values (3,'Mohit','Mumbai','2020-10-11 08:52:14.420')

select * from Employee

--the below statement through an exception

--exception:Implicit conversion from data type datetime to int is not allowed. Use the CONVERT function to run this query.

insert into Employee values ('Heena','Raipur',3500,GETDATE(),4)

```
select * from Employee
```

--while inserting a values into a table if you not inserted a vlue and simply you have provided ''

--then by default it will take the below values

--1.If column is defined with varchar data type then it will insert balnk/empty space

--2.If column is defined with integer then by default it will take ZERO (0)

--3.If column is defined with datetime the by default it will take value'1900-01-01 00:00:00.000'

--METHOD -II

--It will allow you to insert the values as per your choice but condition is that you have follow your own defined order or sequence of columns

```
insert into Employee (EMP_NAME,EID) values ('Mohit',5)
insert into Employee (EID,EMP_NAME,EMP_DOJ) values (6,'Heena','2018-08-14 09:30:14.420')
```

--Q.How to select a specific column from table?

```
select EMP_NAME from Employee

select * from Employee
```

--SQL Claues

--Caluses are used to filter tha data by providing condition.

--Claues are used for filtering purpose.

--Whenever we want to use clauses then we can use along with SQL opeartor.

--There are various types of clauses inside SQL

--1.WHERE

--2.ORDER BY

--3.GROUP BY

--4.HAVING

select * from Employee where EID =1

--CLASS-4

--Operators

--Their are various types of operator

--1.Comparision

--2.Logical

--3.Arithmatic

--4.IN and NOT IN

--5.Between and Not Between

--6.LIKE

--1.Comparison

--It is used to comapre the condition provided into where clause

```
-- = -- equal to
-- > -- grater than
-- < -- less than
-- >= -- gretar than equal to
-- <= -- less than equal to
-- <> or != -- Not equal to


select * from Employee

select * from Employee where EID = 4
select * from Employee where EID > 4
select * from Employee where EID < 4
select * from Employee where EID >= 4
select * from Employee where EID <= 4
select * from Employee where EID <> 4
select * from Employee where EID != 4

select * from Employee where EMP_NAME ='sumit'

--2.Logical operator
```

--It used to comapre the two inputs logically based upon the operation specified into where clause.

```
--1.AND
--2.OR     --3.NOT
```

--1.AND

--It is just multiplication

--AND operation

--A	B	O/P
--1	1	1
--1	0	0
--0	1	0
--0	0	0

--A	B	O/P
--TRUE	TRUE	TRUE
--TRUE	False	False
--False	True	False
--False	False	False

```
select * from Employee where eid =1 and EMP_SAL = 2500
 select * from Employee where eid =3 AND EMP_SAL = 2500
```

--2.OR

--It is used to comapre two inputs logically , it will act as addition operation.

--OR operation

```sql
--A         B         O/P
--1         1         1
--1         0         1
--0         1         1
--0         0         0

--A         B         O/P
--TRUE           TRUE      TRUE
--TRUE           False     TRUE
--False     True TRUE
--False     False     False

select * from Employee where eid =3 OR EMP_SAL = 2500
select * from Employee where eid =7 OR EMP_SAL = 2500
select * from Employee where eid =3 OR EMP_SAL = 3000

--3.NOT
--It will perform negation
--It will perform opposite operation.

--NOT operation
--INPUT          O/P
--1              0
--TRUE                False
```

select * from Employee where eid not in (3)

--Clause

--1.WHERE

--Where clause is used with compariosion , logical and arithmatic operator.

Create table emp (eid int ,ename varchar(20),eloc varchar(20),sal int)

insert into emp values (1,'Meena','HYD',4000)

insert into emp values (2,'tina','Pune',3000)

insert into emp values (3,'Mona','Mumbai',5000)

insert into emp values (4,'priya','Jaipur',7000)

insert into emp values (5,'Meera','Patna',8000)

insert into emp values (6,'radha','Delhi',9000)

insert into emp values (7,'shina','Goa',2000)

select * from emp where eid = 3 and sal > 3000

--2.Order by

--It is used to display the content of acolumn either in Ascending or descending order.

--If you have not defined anything after ORDER BY clause then By default it will be Ascending

--It will use notation for Ascending as ASC and for Descending as DESC.

--syntax: ORDER BY col1,col2 ...coln

select * from emp order by sal

select * from emp order by sal ASC

select * from emp order by sal DESC

--3.Arithmatic operator
--it is used to perform aritmatic operation.
--We have multiple operator in SQL i.e. +,-, * ,/ and %

select *,salary= sal + 1000 from emp
select *,salary= sal - 1000 from emp

select *,Package= sal * 12 from emp where eid =3
select eid,ename from emp
select (ename +'   '+ eloc) as FULLNAME from emp

--Alias in SQL
--By using Alis in SQL we can specify user defined names to a column.

--Synatx : AS UserDefinedColName

```
--CLASS-5
select * from emp


--Q.How to calculate per day salary?
select *,Perday= sal/30  from emp


--Q.How to display even records from table ?
select * from emp where eid % 2 =0


--Q.How to display odd records from table ?
select * from emp where eid % 2 =1


--IN and NOT IN
--This opeartor or clause allow you to navigate or point out values specified into the list.
--NOT IN operator will perform the reverse or opposite as IN operation.


select * from emp where eid in (1,2,3,5)


select * from emp where eid not in (1,2,3,5)


select * from emp where eloc in ('HYD','PUNE','GOA')


select * from emp where eloc not in ('HYD','PUNE','GOA')
```

--BETWEEN and NOT BETWEEN

--This operator or clause allow you to display the values or records between the range you have specified.

--this operator will work with Logical operator.

select * from emp where eid between 2 and 4

select * from emp where eid not between 11 and 14

select * from emp order by eloc

select * from emp where eloc between 'd' and 'ha' --goa

select * from emp where sal <> 3000 and eid not in (1,4)

--LIKE

--LIKE operator will allow you to search pattern from given string or number.

--LIke mostly used with WHERE clause

--LIKE most oftern used with charecter and we can also used with integer.

--LIKE operator used with Following wildcards for searching pattern

--1.% - It represents one or multiple charecters.

--2.'_' - represents one or single charecter/ Sustitute for exacctly one charecter.

```
--3.[Charlist]% - Any single charecter from that list from start
--      %[Charlist] - Any single charecter from that list from end
--      %[Charlist]% - Any single charecter from that list start from end.
--4.[^Charlist] or [!Charlist] -- any single chatrecter not in charlist.
```

--'A%' - start with A charecter and it will display all the values or names which start with A.

--'%A' - ENds with A charecter and it will display all the values or names which Ends with A.

--'%A%' - Anywhere inside the record/Column if A charecter is present

```
select * from emp where ename like 'm%'

select * from emp where ename like '%a'

select * from emp where ename like '%e%'

select * from emp where ename like '_i%'

select * from emp where ename like '__n%'

select * from emp where ename like 't%a'

select * from emp where ename like 'n%n'
```

```
--it will display Name which start with M,R and S
select * from emp where ename like '[MRS]%'

--it will display Name which is not start with M,R and S
select * from emp where ename like '[^MRS]%'

--select * from emp where ename like '[!MRS]%'

--It will display the range of names who start with a,b,c,d,e,f,g.
select * from emp where ename like '[a-g]%'

insert into emp values (8,'Amit','Panji',4000)
insert into emp values (9,'Bali_Patil','Panji',3400)
insert into emp values (10,'SAM_Curran','UK',4500)
--Want to display which have '_' in between
select * from emp where ename like '%#_%'escape '#'

--or

select * from emp where ename like '%[_]%'

--or '
select * from emp where ename like '%@_%'escape '@'
```

--CLASS-6

--SQL Aggregate functions

--SQL aggregate functions return a single value, calculated from values in a column.

--below are some of useful aggregate functions are

--1.AVg()

--2.Count()

--3.MIN()

--4.MAX()

--5.SUM

--6.First()

--7.Last()

--1.AVG()

--This function is used to find the average value of the column values from table.

```
select * from emp
```

```
select AVG(sal) as AVGSAL from emp
```

--2.Count()

--The count() function returns the number of rows that macthes specified criteria.

```
--The count function requires 1 argument(s).
select count(*) as recordcount from emp

select count('SCODEEN') --O/P = 1

select count('1234') + count('5678') --O/P = 2

select count('1234') * count('5678') --O/P = 1

select count('1','2','4') --Exception: The count function requires 1 argument(s).

select count(1234512345678996)   --O/P =1

select count('Scodeen')  --O/P =1
--Q.Count the no of employess whose salary is greater than 5000.
select count(*) as empsal from emp where sal > 5000

--3.MIN()
--The MIN() function returns the smallest value of the selected column.

select * from emp order by sal

select min(sal) as minsal from emp --First minimum sal
--Second Min sal
```

select min(sal) as secondMinSAL from emp where sal >(select min(sal) from emp)

--OR

select min(sal) as secondMinSAL from emp where sal <>(select min(sal) from emp)

--OR

select min(sal) as secondMinSAL from emp where sal not in (select min(sal) from emp)

--Third Min sal

select min(sal) as secondMinSAL from emp

where sal >(select min(sal) from emp where sal >(select min(sal) from emp))

select min(ename) as minsal from emp

--second min sal

--4.MAX()

--The MAX() function returns the largest value of the selected column.

select * from emp order by sal desc

select max(sal) as maxsal from emp --First highest sal

select max(ename) as maxname from emp --tina , tia,tz

```
--Second Highest sal

select max(sal) as secondMinSAL from emp where sal <(select max(sal) from emp)

--OR

select max(sal) as secondMinSAL from emp where sal <>(select max(sal) from emp)

--OR

select max(sal) as secondMinSAL from emp where sal not in (select max(sal) from emp)


select max(sal) from emp


--Q.Find out the second highest sal?


--5.SUM()

--The sum function returns the total sum of numeric value.

--Sum function always accept nemeric values.


select * from emp


select sum(sal) as SUMSAL from emp where eloc ='Panji'


select sum(ename) as SUMSAL from emp
```

--6.First()

--The first function returns the first value of the selected column.

--'first' function is not a recognized built-in function name inside SQL server Managment studio(SSMS).

select * from emp

select first(sal) from emp -- Oracle - SQL developer

--7.LAST()

--The LAST function returns the LAST value of the selected column.

--'Last' function is not a recognized built-in function name inside SQL server Managment studio(SSMS).

select * from emp

select Last(sal) from emp -- Oracle - SQL developer -

--CLASS-7

--TOP

--It will allow to select Top records from table.

--It is useful when we are dealing with large amount of data

select * from emp order by sal desc

select top 5 * from emp

```
--Third highest sal from emp table ?
select min(sal) as thirdmaxsal from emp where sal in --(9000,8000,7000)
(select top 3  sal from emp order  by sal desc)

--Third min sal
select max(sal) as thirdminsal from emp where sal in
(select top 3  sal from emp order  by sal asc)

--Distinct
--It will allow you to find unique or distinct values from a cloumn.
--Syntax: distinct(colname)
select sal from emp
select distinct(sal) from emp

create table DISTINCT_TEST(DID int ,DNAME varchar(20),dept
varchar(20))

insert into DISTINCT_TEST values (1,'Seema','HR')
insert into DISTINCT_TEST values (2,'meera','Finance')
insert into DISTINCT_TEST values (3,'amit','Account')
insert into DISTINCT_TEST values (4,'sumit','HR')
insert into DISTINCT_TEST values (5,'rohit','Admin')
insert into DISTINCT_TEST values (6,'mohit','account')
insert into DISTINCT_TEST values (7,'jaya','admin')
insert into DISTINCT_TEST values (8,'Riya','Finance')
insert into DISTINCT_TEST values (9,'sohil','Testing')
insert into DISTINCT_TEST values (10,'amir','Dev')

select * from DISTINCT_TEST
select distinct(dept) from DISTINCT_TEST
--Q.How will you avoid duplicate records from column?
--Q.how will you find unique values from column?
```

--NULL Values

--If a column in table is optinal and we are inserting or updating an existing record sby skiping an optional column.

--This means that optinal column will be saved as NULL value.

--NULL values are treated diffrently from other values.

--NULL values used as a Placeholder for unknown or inapplicable values.

--for ex:

```
select * from DISTINCT_TEST

insert into DISTINCT_TEST (did,dname) values (11,'Sumit')
insert into DISTINCT_TEST (did,dname) values (12,'Vickey')

select * from DISTINCT_TEST where dept = NULl
```

--It is not possible to test NULL values by using comparision operator

--To test NULL value inside a SQL we have to use below functions

--1.IS NULL

--2.IS NOT NULL

select * from DISTINCT_TEST where dept is NULL

select * from DISTINCT_TEST where dept is not NULL

select count(distinct(dept)) from DISTINCT_TEST

select count(NULL) --Operand data type NULL is invalid for count operator.

select * from DISTINCT_TEST order by dept asc

select * from DISTINCT_TEST order by dept desc

--CLASS-8

--SQL Constarints

--Constraints are used to maintain the accuracy and integrity of the table.

--Constraints are use to limit the type of data that can go into the table.

--There are varoius types of constraints are available in SQL

--1.Primary key

--2.Foreign key

--3.NULL Key

--4.Unique Key

--5.Default key

--6.Check Key

--1.Primary Key(PK)

--NOT NULL + UNIQUE

--PK constraint uniquely identifies each record in database table.

--PK is used most oftenly with numeric values.

Create Table PRIME_TEST (PID int primary key,

FirstName Varchar(20),

City varchar(20))

insert into PRIME_TEST values (1,'Praveen','Bidar')

insert into PRIME_TEST values (2,'Praveen','Bidar')

insert into PRIME_TEST values (NULL,'amit','PUNE') --Cannot insert the value NULL into column 'PID', table 'Testing21.dbo.PRIME_TEST'; column does not allow nulls. INSERT fails.

insert into PRIME_TEST values (3,NULL,NULL)

select * from PRIME_TEST

--2.Foreign Key(FK)

--A foreign key in one table points to PRIMARY KEY in another table.

--It ensures that all the values in a columns are diffrenet.

--One NULL value can be applied at column level.

--A FK key is a collection of columns in one table that refers to the PK in another table.

--Parent(PK)---Child(FK)

Create table department(DID int primary key ,DEPT_NAME varchar(20))

insert into department values (1,'ECE')

insert into department values (2,'ME')

insert into department values (3,'Civil')

insert into department values (4,'Electrical')

insert into department values (5,NULL)

create table student(S_ID int primary key ,

Firstname varchar(20),

City varchar(20),

DID int foreign key references department(DID))

insert into student values(1,'Sumit','Mumbai',4)

insert into student values(2,'Rohit','Jaipur',5) --exception : The INSERT statement conflicted with the FOREIGN KEY constraint "FK__student__DID__3B75D760". The conflict occurred in database "Testing21", table "dbo.department", column 'DID'.

insert into student values(2,'Rohit','Jaipur',NULL)

insert into student values(3,'Shika','Mandi',NULL)

insert into student values(4,'Meera','Indore',5) select * from department

select * from student

--Q.Without defining PK on Parent table , define FK on Child table?

--3.NOT NULL

--The NOT NULL constraint enforces/restricts a column to NOT accept NULL values.

--The NOT NULL constraint enforces/restricts a column to always conatin a value.

Create Table NOTNULL_TEST (NID int primary key,

FirstName Varchar(20) NOT NULL,

City varchar(20))

insert into NOTNULL_TEST values(1,'shita','Pune')

insert into NOTNULL_TEST values(2,NULL,'Mumbai') --Exception:Cannot insert the value NULL into column 'FirstName', table 'Testing21.dbo.NOTNULL_TEST'; column does not allow nulls. INSERT fails.

insert into NOTNULL_TEST values(2,'shita','Mumbai')

insert into NOTNULL_TEST values(3,'','jaipur')

select * from NOTNULL_TEST

--4.Unique

--The UNIQUE constraint uniquely identifies each record in a database table.

--One NULL value can be inserted at column level.

Create Table UNIQUE_TEST (U_ID int primary key,

FirstName Varchar(20) unique,

City varchar(20))

insert into UNIQUE_TEST values(1,'shita','Pune')

insert into UNIQUE_TEST values(2,'shita','udaipur')

--Exception:Violation of UNIQUE KEY constraint 'UQ__UNIQUE_T__B31331C90814D506'. Cannot insert duplicate key in object 'dbo.UNIQUE_TEST'. The duplicate key value is (shita).

insert into UNIQUE_TEST values(2,'','Mumbai')

insert into UNIQUE_TEST values(3,NULL,'indore')

insert into UNIQUE_TEST values(4,' ','jaipur') --Exception : Violation of UNIQUE KEY constraint 'UQ__UNIQUE_T__B31331C90814D506'. Cannot insert duplicate key in object 'dbo.UNIQUE_TEST'. The duplicate key value is ().

insert into UNIQUE_TEST values(4,NULL,'indore') --Exception :Violation of UNIQUE KEY constraint 'UQ__UNIQUE_T__B31331C90814D506'. Cannot insert duplicate key in object 'dbo.UNIQUE_TEST'. The duplicate key value is (<NULL>).

select * from UNIQUE_TEST

```
Create Table UNIQUE_TEST1 (U_ID int primary key,
FirstName Varchar(20) unique NOT NULL,
City varchar(20))

insert into UNIQUE_TEST1 values(1,'Meena','indore')
insert into UNIQUE_TEST1 values(1,NULL,'indore') --Exception:Cannot insert the value NULL into column 'FirstName', table 'Testing21.dbo.UNIQUE_TEST1'; column does not allow nulls. INSERT fails.
insert into UNIQUE_TEST1 values(2,'Meena','indore')

--CLASS-9
--5.Check Key
--Check constraint is used to restrict or limit the value range that can be placed in a column.
--If you define CHECK constraint on a single column it allows only certain values for this column.

Create table check_test(CID int,
FirstName varchar(20) Unique,
Loc varchar(20) NOT NULL,
Age int check (age>=18))

insert into check_test values(1,'Praveen','Pune',20)
insert into check_test values(1,'Amit','Mumbai',18)
insert into check_test values(2,'Shiva','Jaipur',17) --exception : The INSERT statement conflicted with the CHECK constraint "CK__check_test__Age__45F365D3". The conflict occurred in database "Testing21", table "dbo.check_test", column 'Age'.

select * from check_test

--6.Defuault Constraint
--DEFAULT constraint is used to insert a default value into a column.
--The default value will be added to all new records, if no other value is specified.

create table default_test(D_ID int primary key,
DName varchar(20) unique,
Loc varchar(20) DEFAULT 'PUNE',
Pincode int Default '411061')

--I
insert into default_test values (1,'Amol','Mumbai','000001')
insert into default_test values (2,'Sumit',default,411027)
insert into default_test values (3,'Amit',default,default)
insert into default_test values (6,'Ronit',default,000000)
select * from default_test

--II
insert into default_test (D_ID,DName) values (4,'Meena')
insert into default_test (D_ID,DName) values (7,Rina)

create table default_test1(D_ID int primary key,
DName varchar(20) unique,
Loc varchar(20) DEFAULT 'PUNE',
Pincode  varchar(20) default  '000000')

insert into default_test1 values (1,'shiva','Latur',default)

select * from default_test1

--Auto-increment
--It is used to create the unique values inside the table on defined column.
--While cretaing a unique value we can provide the diffrence of uniqueness.
--Syntax : column_name IDENTITY(start,Diff of next value)
--ex: accountno identity (1111288780,5) --1111288780,1111288785,1111288790

create table Auto_test(AID int primary key identity (1,1),
Name varchar(20),
Pin int )

insert into Auto_test values ('amay','123456')
```

```
select * from Auto_test

create table Account_open(Account_Number int identity(1111288780,1),
account_Name varchar(20),
Branch varchar(20))

insert into Account_open values('Praveen','Delhi')
insert into Account_open values('Amit','Pune')
insert into Account_open values('Sumit','Patna')
insert into Account_open values('Mohit','Jaipur')
insert into Account_open values('Rohit','Mumbai')
--1111288781-85 90 95

select * from Account_open

--Q.Create a employee table by using all the constraints?

--CLASS-10
--3.Gorup by
--Group by Statement is used in conjunction with aggregate functions to group by the result set by one or more columns.
```

```
--Syntax
--select COL_NAME,aggregate_function(COL_NAME)
--From Table_Name
--Where Col_Name operator value
--GROUP BY COL_NAME

create table orders (O_ID int, Customer varchar(20),orderprice int,ordercategory varchar(20))

insert into orders values (1,'Kate',2000,'grocery')
insert into orders values (2,'Mark',3000,'Fruits')
insert into orders values (3,'Tim',4000,'grocery')
insert into orders values (4,'Paine',5000,'Cloths')
insert into orders values (5,'Steve',6000,'Sports')
insert into orders values (6,'Bill',7000,'grocery')
insert into orders values (7,'Andy',8000,'Electronics')
insert into orders values (8,'Grant',9000,'Sports')
insert into orders values (9,'Pat',2500,'Cloths')
insert into orders values (10,'Shon',3500,'Electronics')

select * from orders
select distinct(ordercategory) from orders

select ordercategory,sum(orderprice) from orders group by ordercategory
```

--Q.How to display the minimum price of each order category ?

select ordercategory,min(orderprice) as MInOrder from orders group by ordercategory

select ordercategory,sum(orderprice) from orders where ordercategory ='Electronics' group by ordercategory

select ordercategory, count(*) from orders group by ordercategory

--Q.How to display order category whose having more than two customer and its min price.

select ordercategory, min(orderprice), count(*) from orders group by ordercategory

select * from orders where sum(orderprice) > 2000

--HAVING Clause

--The Having Clause was added to SQL because WHERE Clause is not used with aggregate functions.

--Synatx

--select COL_NAME,aggregate_function(COL_NAME)

--From Table_Name

--Where Col_Name operator value

--GROUP BY COL_NAME

--HAVING aggregate_function(COLUMN_NAME) operator value

--Display the ordercategory whose customer base is more than 1

select ordercategory, min(orderprice), count(*) from orders group by ordercategory having count(*) > 1

--Want to display the ordercategory whose sum is grater than 10000.

select ordercategory,sum(orderprice) from orders

group by ordercategory having sum(orderprice)> 10000

select ordercategory,sum(orderprice) from orders where ordercategory = 'fruits'

group by ordercategory having sum(orderprice)< 10000

--SQL Statements

--1.DML(Data Manipulation Language) --S_UID

--These statements are used to play with table data which is stored inside the table.

--SELECT,UPDATE,INSERT and DELETE

--2.DDL(Data Defination Language)-- DR.CAT

--Thse statements are used to play with table related activities.

--CREATE,ALTER,TRUNCATE,DROP, RENAME

--3.DCL(Data Control Language)

--This statements are used to provide the access to a user , this role is assigned to DBMS admin. --GRANT,REVOKE

--4.TCL(Transction Control Language)

--This statements are used to perform control related activites during SQl statements execution.

--TRAN,COMMITTRAN, ROLLBACK.

--Q.How to display the duplicate records from table?

--CLASS-11

--UPDATE

--The UPDATE Statement is used to update existing records in a table.

--While Updating column if you have not specified condition then it will modify/update the complete column.

--Syntax

--UPDATE table_name SET Col1 = value1,col2 = value2,....

--WHERE Some_Column = Some_value.

--single column update

update UNIQUE_TEST SET firstName ='Rohan' where U_ID =3

update UNIQUE_TEST SET City ='PUNE'

--Multiple column updateupdate

UNIQUE_TEST set FirstName ='Mohan',City ='Mumbai' where U_ID =2

select * from UNIQUE_TEST

```
--DELETE
--DELETE Statement is used to delete the rows in a table.
--DELETE Statement delete the records from a table ROW-by-ROW.
--DELETE statement will not able to delete the structure of the table.

--Syntax
--DELETE FROM table_name
--WHERE some_col = Some_Vale

select * from emp

select * from Employee
--Delete specific row
delete from Employee where EID =7

--Delete the table data
delete from UNIQUE_TEST

select * from UNIQUE_TEST

--delete multiple rows from a specific range
delete from Employee where eid between 2 and 4

delete from Employee where eid in (6)
```

delete from Employee where EMP_NAME like 'S%'

select * from Employee

--DDL(DR.CAT)

--Always with DDL statements We need to use Table keyword with statement.

--Truncate

--It will you to delete the records from a table at once.

--Truncate statement will not delete the table structure.

--It will not allow you to delete the records from a table on ROW-by-ROW basis.

--Syntax:

--TRUNCATE TABLE table_name

select * from Employee

truncate table Employee

--DROP

--DROP Statement will allow you to delete the table structure along with table data.

--DROP is also used to drop the DATABASE.

```
--Syntax
--DROP Table table_name
--DROP DATABASE database_name

select * from DISTINCT_TEST

--Drop table
drop table DISTINCT_TEST

--Drop database
create database sampletest
drop database sampletest

--Q.What is the diffrence between DELETE ,DROP and Truncate?
--Q.What is the diffrence between DELETE and DROP?
--Q.What is the diffrence between Truncate and DROP?
--Q.What is the diffrence between Delete and DROP?

--Table Back up
--We can take table back up by using the below statement/syntax we can take the back up

--Syntax:
--SELECT * INTO Table_backup_Name FROM table_name
```

--Q.How to take the table backup

select * Into emp_261021 from emp

select * from emp

select * from PRIME_TEST_261021

select * from PRIME_TEST

select * into PRIME_TEST_261021 from PRIME_TEST

--DATABASE Backup

--While taking the back of database we need to mention the path where we want to take backup.

BACKUP DATABASE Testing21 TO DISK = 'E:\.bak'

--CLASS-12

--INSERT INTO SELECT

--This Statement is used to copy data from one table and inserts it into another table.

--But condition is that, it requires the data type in source and target tables should match.

create table orders1 (O_ID int, Customer varchar(20),orderprice int,ordercategory varchar(20))

```
select * from orders
select * from orders1

insert into orders1 select * from orders where ordercategory ='grocery'

select * from orders1

create table orders2 (O_ID int, Customer varchar(20))
select * from orders2

insert into orders2 select O_ID,Customer from orders where ordercategory ='Electronics'

select * from orders3

insert into orders3 select O_ID,Customer,orderprice,ordercategory from orders where ordercategory ='Cloths'

--ALTER
--By using ALTER Statements we can manipulate/Play with table columns/Attributes/Fields.
--By using ALTER function we can perform multiple operations.
--1.WE can add one or more columns at a time into table.
--2.WE can delete one or more columns at a time into table.
```

```
--3.WE can Increase and decrease the size of column into table.
--4.We can change the data type of a column.
--5.We can drop the constraints from column into a table.
--6.We can define the constraints to column.

--Syantax
--ALTER TABLE Table_Name ADD Column_Name dataType --To add
Column
--ALTER TABLE Table_Name DROP COLUMN Column_Name  --To Delete
Column
--ALTER TABLE Table_Name Add constraint pk_key_Name primary
key(Column_name) -- Add constraint

Create Table ALTER_TEST(AID int ,ANAME varchar(20))

--Add single column into table
ALTER Table ALTER_TEST ADD ALOC varchar(20)

select * from ALTER_TEST

insert into ALTER_TEST values (1,'Mohit','Ranchi',Default)
insert into ALTER_TEST values (2,'Parveen_Patil','Patna',Default)
insert into ALTER_TEST values (3,'Amit','Delhi',Default)
```

--Adding multiple columns into a table.

ALTER table alter_test add pin int,contactno varchar(20)

--Delete single column from table

ALTER table alter_test drop column contactno

--Delete Multiple columns from table

ALTER table alter_test drop column Aloc,pin

select * from ALTER_TEST

--IF you want to check the structure of table then we have to use below syntax --****VVIMP

SP_HELP ALTER_TEST

--Increase the size of column

ALTER table alter_test alter column ANAME varchar(50)

--Decrese the size of column

ALTER table alter_test alter column ANAME varchar(2)

--NOTE: if you are trying to decrease the size of column less than the lenghth of inserted data i to that column then it will through an exception

--Exception: String or binary data would be truncated in table 'Testing21.dbo.ALTER_TEST', column 'ANAME'. Truncated value: ''.

```
select len(aname) from ALTER_TEST

ALTER table alter_test alter column ANAME varchar(5)

ALTER table alter_test alter column AID bigint

--Chnage the datatype of column
SP_HELP ALTER_TEST

ALTER table alter_test alter column ANAME char(5)

truncate table alter_test --truncating a table
--If the table dont have any records into it then we can change its data type to any new datatyoe.
ALTER table alter_test alter column ANAME int

select * from ALTER_TEST

--Adding a constraint to the column
ALTER TABLE ALTER_TEST Add constraint PK primary key(AID)

--Note
--1.While defining PK ,If their is no data inserted into the table then we can define PK but column should define with NOT NULL constraint.
```

```
--2.If the data is inserted into the table then if it is unique data and NOT NULL constraint is defined then only we can define PK on column.

ALTER TABLE ALTER_TEST ALTER column AID int NOT NULL

alter table ALTER_TEST add constraint un unique(aloc)

sp_help  ALTER_TEST
--4byte = 8bit
--Drop the constraint from column
ALTER table alter_test Drop constraint PK

select * from alter_test
--Duplicate
Create table duplicate (DID int , DNAME varchar(20),Dloc varchar(20))

insert Into duplicate values(1,'Amit','Bidar')
insert Into duplicate values(2,'Sumit','Pune')
insert Into duplicate values(1,'Amit','Bidar')
insert Into duplicate values(2,'Sumit','Pune')
insert Into duplicate values(3,'Mohit','Mumbai')
insert Into duplicate values(4,'Rohit','Ranchi')
insert Into duplicate values(3,'Mohit','Mumbai')
insert Into duplicate values(4,'Rohit','Ranchi')
select * from duplicate
```

--Structure

sp_help duplicate

--Q.How to find duplicate records from table?

--There are multiple ways we can find duplicate records from table.

--1.By using PK in select list we can find out the duplicate records

--Syntax:

--Select <PK1>,<PK2>...<PKn> , count(*) as Duplicate from Tbale_Name group by <PK1>,<PK2>...<PKn> having count(*) > 1

--2.If PK is not defined then we can use all the columns from table into the select list and we can find duplicate records from table.

--syntax:

--select col1,col2,...coln , count(*) as duplicate from Table_name group by col1,col2,...coln having count(*) > 1

select DID,DNAME,Dloc , count(*) as duplicaterecords from duplicate group by DID,DNAME,Dloc having count(*) >1

select * from duplicate

select * from ALTER_TEST

sp_help ALTER_TEST

--If PK is defined the how to write query to identify duplicate records from table.

```
select aid,count(*) as Duplicate from ALTER_TEST group by aid having
count(*) > 1
```

--Information Schema

--By using information schema we will identify how many tables are their in a database.

```
select * from INFORMATION_SCHEMA.TABLES
```

```
select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME like
'e%'
```

--By using information schema we will identify how many Column are their in a table.

```
select count(*) from INFORMATION_SCHEMA.COLUMNS where
TABLE_NAME = 'Employee'
```

```
select * from Employee
```

--By using information schema we will identify Column names from table.

```
select COLUMN_NAME from INFORMATION_SCHEMA.COLUMNS where
TABLE_NAME = 'Employee'
```

```
select EID,EMP_NAME,EMP_LOC,EMP_SAL,EMP_DOJ,
count(*) from Employee group by EID,EMP_NAME,EMP_LOC,EMP_SAL,
EMP_DOJ
having count(*) > 1
```

--Excel Formula

--How to place place comma(,) after every column in excel sheet.

--=""&column(a1)&","

--CLASS-13

--JOIN

--Join is used to return a value from both the table which should have common column in both the tables.

--JOIN is the keyword is used in SQL statements to extract the data from two or more tables.

--JOIN = CROSS PRODUCT + CONDITION

--Types Of joins

--1.JOIN/Inner Join

--2.Outer Join

--		a.Left Join /Left Outer join

--		b.Right Join /Right Outer join

--		c.FULL Join /Full Outer join

--3.SELF join

--4.Equi-join

--5.Cross Join

--1.JOIN/Inner Join

```
--This join return the only matching records from Table
--Join = CROSS PRODUCT + CONDITION

--Syntax:
--select */Column_name(s) from Table_Name1 T1
--INNER JOIN /JOIN Table_Name2 T2
--ON T1.Column_name =T2.Column_name

create table A (AID int, ANAME varchar(10),ACITY varchar(20))

insert into A values (1,'Mohit','Indore')
insert into A values (2,'Sumit','Jaipur')
insert into A values (3,'amit','Delhi')
insert into A values (4,'preeti','Hyderabad')
insert into A values (5,'Rohit','Warangal')

create table B (BID int, BNAME varchar(10),BCITY varchar(20),AID int)

insert into B values (11,'Sheela','Mumbai', 3)
insert into B values (12,'Maya','Pune',4)
insert into B values (13,'Riya','Nasik',5)
insert into B values (14,'Shina','Sangli',6)
insert into B values (15,'Meena','Satara',7)
insert into B values (16,'Rohit','Warangal',8)
```

```
create table C (CID int, Age int,BID int)
insert into C values (21,17,11)
insert into C values (22,19,12)
insert into C values (23,21,13)
insert into C values (24,23,18)
insert into C values (25,25,19)
select * from A
select * from B
select * from C
--joinin Two Tables
select * from A JOIN B ON A.AID =B.AID

--Joining three tables
select * from A join B ON A.AID = B.AID Join C on B.BID = C.BID

select A.ANAME,B.BCITY,C.Age from A join B ON A.AID = B.AID Join C on
B.BID = C.BID

--2.Outer Join
--		a.Left Join /Left Outer join
--All the records from left table and matching records from right table it
will show and for non matching records it will display as NULL values

--Syntax:
--select */Column_name(s) from Table_Name1 T1
```

```
--LEFT OUTER JOIN /LEFT JOIN Table_Name2 T2
--ON T1.Column_name =T2.Column_name
select * from A
select * from B
select * from A left outer join B ON A.aid =b.aid
select * from A left join B ON A.aid =b.aid
```

--   b.Right Join /Right Outer join

--All the records from right table and matching records from left table it will display and for non matching records it will display as NULL values

```
--Syntax:
--select */Column_name(s) from Table_Name1 T1
--Right OUTER JOIN /Right JOIN Table_Name2 T2
--ON T1.Column_name =T2.Column_name
select * from A
select * from B
select * from A right outer join B ON A.aid =b.aid
select * from A right join B ON A.aid =b.aid
```

--   c.FULL Join /Full Outer join

--It will display the complete records from both the tables.

--If the records are not matching then for the respective table records it will display the NULL values.

```
--Syntax:
--select */Column_name(s) from Table_Name1 T1
--FULL OUTER JOIN /FULL JOIN Table_Name2 T2
--ON T1.Column_name =T2.Column_name

select * from A
select * from B
select * from A full outer join B ON A.aid =b.aid



--CLASS-14
--3.Self Join
--It combine a tableon itself or Join a table itself with equal and non-equal condition.

create table SELFJOIN_TEST (EID int ,ENAME varchar(10),MANAGERID int)

insert into SELFJOIN_TEST values (1,'Praveen',3)
insert into SELFJOIN_TEST values (2,'Amit',5)
insert into SELFJOIN_TEST values (3,'Kumar',4)
insert into SELFJOIN_TEST values (4,'Meena',2)
insert into SELFJOIN_TEST values (5,'Pushpa',1)
insert into SELFJOIN_TEST values (6,'Sumit',7)
select A1.EID,A1.ENAME from SELFJOIN_TEST A1
```

--SELF JOIN

select * from SELFJOIN_TEST S1,SELFJOIN_TEST S2 where S1.MANAGERID = S2.EID

select S1.EID,S1.ENAME,S2.ENAME from SELFJOIN_TEST S1,SELFJOIN_TEST S2 where S1.MANAGERID = S2.EID

--SELF JOIN can be used as

--1.Inner Join

select S1.EID,S1.ENAME,S2.ENAME from SELFJOIN_TEST S1,SELFJOIN_TEST S2 where S1.MANAGERID = S2.EID;

select S1.EID,S1.ENAME,S2.ENAME from SELFJOIN_TEST S1 inner join SELFJOIN_TEST S2 ON S1.MANAGERID = S2.EID;

--2.Left Join

select * from SELFJOIN_TEST S1 left join SELFJOIN_TEST S2 ON S1.MANAGERID = S2.EID

--S1.EID,S1.ENAME,S2.ENAME

--Q. Write SQL query for left join by using self join?

--3.Right Join

select * from SELFJOIN_TEST S1 right join SELFJOIN_TEST S2 ON S1.MANAGERID = S2.EID

--Q. Write SQL query for right join by using self join?

--4.Equi-join

--If you are not using JOIN keyword while joining the two tables then that will be considered as equi-join

```
select * from A

Select * from B


select * from A , B where A.AID = B.AID
```

--5.Cross Join

--It is cartesian product.

--For ex: If table A is having 5 records and table B is having 4 records then it will multiply and display the result.

--i.e 2o records it will display (5*4 = 20).

```
select * from A, B


select * from A cross join B
```

--CLASS-15

--SET operator

--1.UNION

--2.UNION ALL

--3.INTERSECT

--4.EXCEPT/MINUS

```
--1.UNION
--The Union operator is used to combine the result-set of two or more SELECT statements or Table.
--The UNION operator selects distinct values by default.

--Note:
--1.Each select statement or table within UNION must have the same number of columns.
--2.The columns must have similar data types.
--3.The columns in SELECT statement or table must be in the same order.

--Example :
--A =[1,2,3,4,5]
--B =[3,4,5,6,7]

--A union B =O/P =[1,2,3,4,5,6,7]

create table set1 (S_ID int ,SNAME varchar(20))

create table set2 (S_ID int ,SNAME varchar(20))

insert into set1 values(1,'A')
insert into set1 values(2,'B')
insert into set2 values(3,'C')
insert into set2 values(4,'D')
insert into set2 values(5,'E')
insert into set2 values(6,'F')
insert into set2 values(7,'G')
insert into set1 values(8,'H')
insert into set2 values(9,'Hamesha')

alter table set1 alter column SNAME varchar(2)

SELECT * FROM SET1
SELECT * FROM SET2

SELECT s_id,sname FROM SET1
UNION
SELECT s_id,sname FROM SET2
```

--2.Union All

--This operator is used to combine two or more tables using select statement when both the tables have same no. of columns.

--Combine the two or more tables with all the values. it means that it will allow duplicate values in it.

```
select * from set1
select * from set2

 select * from set1     Union  all     select * from set2
 --3.Intersection
--It will return only distinct (common records) values from two or more
tables.

select * from set1
select * from set2

 select * from set1
 intersect
 select * from set2

--4.Except/minus
--It will display the difference in records.
--For ex: A = [1,2,3] and B= [3,4,5]
--then A except B - O/P =[1,2]
--then B except A - O/P =[4,5]

 select * from set1
 except
 select * from set2
 select * from set1
 select * from set2
 Except
 select * from set1
 --Date and Time Function
--getdate
select getdate() as Todays_date-- Todays date

select getdate() -1 as Yesterday_date --Yesterday date

select getdate() +1 as Tomorrow_date --Tomorrow date

select getdate() +2

--There are three diffrent functions in SQL to modify or perform any date
related task
--1.DATEDIFF()
--2.DATEPART()
--3.DATEADD()

--1.Datediff() function
--The datediff function requires 3 argument(s).
--If we provide more than 3 arguments then it will through an exception

--syntax : DATEDIFF(interval,date1,date2)

--(YY,MM,DD,HH,Minutes and seconds)

select DATEDIFF(DAYOFYEAR,'1987/09/13','2021/09/13')
select DATEDIFF(DAY,'1987/09/13','2021/09/13')

select DATEDIFF(YYYY,getdate(),GETDATE()+365)
select DATEDIFF(DAY,getdate(),GETDATE()+2)

--interval
--Year,YYYY, YY = Year
--Quarter,QQ, Q = Quarter
--Month - MM, M = Month
--DAYOFYEAR - day of the year
--DAY,dy,y = day
--WEEK,WW,WK = weekday
--HOUR,HH = hour
--MINUTE,MI,N = Minute
--SECOND,SS,S = Second
--MILISECOND , MS = Millisecond

select datediff(MINUTE,'2015/01/01','2021/08/01')

--Q.HOW  to calculate your age ?

select DATEDIFF(YY,'1992/08/15',getdate()) as Present_Age


select DATEDIFF(YEAR,'MONTH',getdate()) as Present_Age
```

```
Create table Account_details (
ACCT_NUMBER int primary key identity(11112881,1),
ACCT_NAME varchar(20),
ACCT_OPEN_DATE varchar(20),
BRANCH Varchar(20))

insert into Account_details values ('Shubham','2015/12/09','MUMBAI')
insert into Account_details values ('Rihan','2016/01/12','Jaipur')
insert into Account_details values ('Sheetal','2017/08/11','GOA')
insert into Account_details values ('Priyanka','2017/01/01','Chennai')
insert into Account_details values ('Manik','2015/01/08','Agra')
insert into Account_details values ('Veena','2021/01/01','Patna')
insert into Account_details values ('Rohan','2019/07/01','Pune')
insert into Account_details values ('Laxmi',GETDATE(),'rohatak')
insert into Account_details values ('Jinal',GETDATE(),'Indore')

--Acoounts opened during the current year
select *,DATEDIFF(YY,ACCT_OPEN_DATE,GETDATE()) as accountage from Account_details where DATEDIFF(YY,ACCT_OPEN_DATE,GETDATE()) = 0

--Acoounts opened during the current year and current month
select *,DATEDIFF(MONTH,ACCT_OPEN_DATE,GETDATE()) as accountage from Account_details where DATEDIFF(MONTH,ACCT_OPEN_DATE,GETDATE()) = 0
```

```
--Number of acoounts opened during the current year

select COUNT(*) as currentyearaccount from Account_details where DATEDIFF(YY,ACCT_OPEN_DATE,GETDATE()) = 0


select * from Account_details

select GETDATE()


select ACCT_NUMBER,ACCT_NAME,ACCT_OPEN_DATE , DATEDIFF(MM,ACCT_OPEN_DATE,GETDATE()) as Ageofaccount from Account_details

where DATEDIFF(yy,ACCT_OPEN_DATE,GETDATE()) >1


--Q.What is the age of your bank account


select ACCT_NUMBER, ACCT_NAME, DATEDIFF(YY,ACCT_OPEN_DATE,getdate()) as ACCOUNT_AGE from Account_details


--Q.Calculate the no of accounts which is opened during the current year.


select ACCT_NUMBER, ACCT_NAME, DATEDIFF(YY,ACCT_OPEN_DATE,getdate()) as ACCOUNT_AGE,count(*) from Account_details

where DATEDIFF(YY,ACCT_OPEN_DATE,getdate()) =0
```

```
--CLASS-16
--2.DATEPART
--This will allow you to display the date part

--Syntax : DATEPART(interval,date/column_name)

select getdate()
select DATEPART(HH,GETDATE())
select DATEPART(YYYY,GETDATE())

select * from Account_details
select *,datepart(YY,ACCT_OPEN_DATE) as Year from Account_details
where datepart(YY,ACCT_OPEN_DATE) =2021

select *,datepart(MONTH,ACCT_OPEN_DATE) as Months from
Account_details

select * from Account_details where datepart(YY,ACCT_OPEN_DATE)
=2021

select * from Account_details where ACCT_OPEN_DATE in ('2021')  --
wrong assumption while comparing date related columns from source
to target.

--if we want to validate date related column which is in terms of
timestamp
```

--and it is very difficult to mention each and every time stamp related column with every date

--in order to avoid that we can use date part so it will consider with mentioned interval.

```
Create table Account_detail (
ACCT_NUMBER int primary key identity(11112881,1),
ACCT_NAME varchar(20),
ACCT_OPEN_DATE varchar(20),
BRANCH Varchar(20))
```

```
insert into Account_detail values ('Jinal','2021','Indore')
insert into Account_detail values ('minal','2020','jaipur')
insert into Account_detail values ('amit','2022','kohi')
insert into Account_detail values ('sumit','2021','delhi')
```

```
select count(*) from Account_details where DATEPART(YY,ACCT_OPEN_DATE) in ('2021','2015')
```

```
select datepart(yy,getdate()) as years, datepart(MM,getdate()) as months  --- yers and months
```

--3.DATEADD()

--it will allow you to add the dates.

--it will accept three arguments.

--syntax : DATEADD(interval,value,date/datecolumn)

```
select DATEADD(DD,30,GETDATE()) as after30days

select DATEADD(YY,10,GETDATE())
select GETDATE() +3650

--Null Value replacement
--We can replace NUll values from column by using three fuctions
--1.ISNULL()
--2.Coalesce()
--3.Case()

create table NULL_TEST(NID int, EMP_NAME varchar(20),Manager varchar(20))

insert into NULL_TEST values (1,'Piya','Sohan')
insert into NULL_TEST values (2,'kate',NULL)
insert into NULL_TEST values (3,'meera','aman')
insert into NULL_TEST values (4,'amit',NULL)
insert into NULL_TEST values (5,'sumit','Kiran')

select * from NULL_TEST

--1.ISNULL() --SQL server  --This function will help us to replace NULL value with any other user defined value.
```

--synatx: ISNULL(Col_name,'String')

--It will accept Two arguments

--NVL - SQL DEVELOPER - Oracle

--IFNULL - MySql

select NID,Emp_name ,ISNULL(Manager,'No Manger') as Manager from NULL_TEST

select NID,EMP_NAME, ISNULL(Manager,'No Manager') AS MangerDetails from NULL_TEST

--2.COALESCE

--It will find or try to locate first appearance of NON-NULL value from a row of table.

--If it is not possible to find or locate NON-NULL value then it always returns a NULL value.

--If there are any blank or empty spaces then it will display space in result because space is not considered as NULL value.

--syntax:COALESCE (col1,col2,....coln)

create table COALESCE_TEST (CID int,FN varchar(20),MN varchar(20),LN varchar(20), sal int)

insert into COALESCE_TEST values (1,'A',NULL,NULL,200)

insert into COALESCE_TEST values (2,NULL,'B',NULL,450) insert into COALESCE_TEST values (3,NULL,NULL,'C',200)

insert into COALESCE_TEST values (4,'',NULL,'D',200)

insert into COALESCE_TEST values (5,'E','F','G',200)

insert into COALESCE_TEST values (6,NULL,NULL,NULL,500)

insert into COALESCE_TEST values (NULL,NULL,NULL,'H',500)

select * from COALESCE_TEST

--FN+MN+LN = FULL NAME

select cid,coalesce(FN,MN,LN) as FullName,sal from COALESCE_TEST

select coalesce(cid,FN,MN,LN) as NonNullValue,sal from COALESCE_TEST

select * from COALESCE_TEST;

select coalesce(FN,MN,LN,cid) as NonNullValue,sal from COALESCE_TEST

select coalesce(cid,sal,FN,MN,LN) as NonNullValue from COALESCE_TEST

select coalesce(NULL,NULL,'SCODEEN')

select coalesce(NULL,NULL,'SCODEEN',5) --exception :Conversion failed when converting the varchar value 'SCODEEN' to data type int.

--3.Case statement

--Case statement identify the condition and returns a values.

```
--It will work like IF-ELSE statement
--If there is no ELSE statement then it will return NULL Value .

--Synatx
--case
--      WHEN condtion then result1
--      WHEN condtion then result2
--      ELSE result
--END

select * from NULL_TEST

select NID,EMP_NAME, Case
                        WHEN Manager IS NULL  THEN 'NO MANGER'
                        WHEN Manager ='Sohan'  THEN 'MOHAN'
                        WHEN Manager ='Kiran'  THEN 'AMAN'
                        Else Manager
                        END
from NULL_TEST
select * from NULL_TEST
update NULL_TEST set Manager ='Mohan' where Manager is NULL
--By using Update statement will write case statement
```

```
--CLASS-17

update NULL_TEST SET Manager =
                                Case
                                when Manager ='Aman' then 'PIYA'
                                when Manager ='Mohan' then 'Sumit'
                                else 'ROHIT'
                                END

update NULL_TEST SET Manager =
                                Case
                                when NID IN (2,4) then 'MOHAN'
                                when NID IN (1,3,5) Then 'PRIYA'
                                when Manager ='PRIYA' then 'SUMIT'
                                else 'ROHIT'
                                END

select * from NULL_TEST


--Exist and Not Exist
--EXIST is used to check whether the result of co-related nested query is empty or not.
--Exist(S)
--TRUE: S has atleast one row/record
--FALSE : S has no row/record.
```

```
--NOT EXIST(S)
--TRUE:S has no row/record.
--FALSE :S has atleast one row/record

Create Table customer(C_ID varchar(5) primary key ,CNAME varchar(20),Loc varchar(20))

insert into customer values('C1','AMIT','PUNE')
insert into customer values('C2','Sumit','Delhi')
insert into customer values('C3','varun','Mumbai')
insert into customer values('C4','snehal','Latur')
insert into customer values('C5','Raj','Sangli')
insert into customer values('C6','Mohit','Satara')

select * from customer

create table order1(OID int primary key, CID varchar(5),groceries varchar(20))

insert into order1 values(1,'C2','almonds')
insert into order1 values(2,'C3','deo')
insert into order1 values(3,'C1','milk')
insert into order1 values(4,'C2','soap')
insert into order1 values(5,'C4','dishes')
insert into order1 values(6,'C2','rice')
```

select * from customer

select * from order1

--OQ

--IQ

select * from customer C where exists (select * from order1 O where C.C_ID =O.CID)

select * from customer C where not exists (select * from order1 O where C.C_ID =O.CID)

--Sub query and Co-Relational Query

--Sub query(Nested subquery)

--Query within query i.e outer query(OQ) and inside inner query(IQ).

--OQ and IQ is independent.

--It follows bottom up approach

--Inside Subquery, IQ always execute only once.

select * from customer where C_ID in (select CID from order1) --('C2','C3','C1','C2','C4','C2')

select * from Emp

select MAX(sal) from emp where sal < (select MAX(sal) from emp)

select * from emp

--Co-relational query

--Query within query i.e outer query(OQ) and inside inner query(IQ).

--IQ is dependent on outer query.

--It follows top down up approach.

select * from customer C where exists (select * from order1 O where C.C_ID =O.CID)

--Q.What is the diffrence between Sub query and Co-relational query.

--SQL SERVER FUNCTIONS

--1.UPPER()

--UPPER() Function converts the value of field/Column to upper case.

--The upper case function requires 1 argument

--syntax :upper (text/column name)

select UPPER('scodeen global')

select *,UPPER(ename) as UPPERCASE from emp

--2.LOWER()

--Lower() Function converts the value of field/Column to lower case.

--The lower case function requires 1 argument

--syntax :lower (text/column name)

select LOWER('SCODEEN GLOBAL')

select *,LOWER(eName) as lowercase from emp

```
--CLASS-18
--3.Substring
--The substring function used to extract charecter from text field
--Synatx : substring(Column_Name,Start,end[lenth]) from table_Name
--Ex: substring ( 'varchar',int,int)

select SUBSTRING('SCODEEN',1,3)

select * from emp
select *,SUBSTRING(ename,2,LEN(ename)) as FisrtLetter from emp

--4.DATALENGTH() and LEN()
--This function returns the number of bytes used to reprsent the expression.
--Syntax : DATALENGTH(string),LEN(String/Column_name)

create table length_check (Lid int, Lname varchar(20)) --20

insert into length_check values(1,'Praveen') --7
insert into length_check values(2,'Amit')
insert into length_check values(3,'Meena')
insert into length_check values(4,'Sohan')
insert into length_check values(5,'Rajni')
insert into length_check values(16,'Mohini')
```

```
select * from length_check


select *,len(Lname) as lengths from length_check
select *,datalength(Lname) as datalengths from length_check


select *,LEN(Lid) as lengths from length_check
select *,datalength(Lid) as datalengths from length_check


--5.CONCAT() , CONCAT with + and CONCAT_WS()
--The CONCAT() function adds two or more strings together.
--Syntax: CONCAT(string1,string2....)


--The + operator allows you to add two or more strings together.
--syntax:string1 + string2 + string_n


--The CONCAT_WS() function adds two or more strings together with a separator.
--syntax : CONCAT_WS(separator, string1/Col_name, string2/Col_Name, ...., string_n/NCol_name)


select CONCAT('Scodeen',' ','Global') as concatination


select 'Scodeen' +' ' +'Global' + ' '+'PUNE' as ConcatWithPlus


select CONCAT_WS ('_','Scodeen','Global','PUNE') as conwithWS
select * from emp


select EID,concat(eName,' ',eloc) as Name_Loc  from emp


select EID,eName +' '+ eloc as Name_Loc  from emp


select EID,concat_ws('@',ename , eloc) as Full_name  from emp where
eid between 2 and 4


--6.LTRIM(), RTRIM() and TRIM()
--The LTRIM() function removes leading spaces from a string.
--The RTRIM() function removes trailing spaces from a string.
--TRIM() function removes leading as well as trailing spaces from string.


select datalength(LTRIM('            LTRIM                         '))
select LTRIM('            LTRIM            ')
select datalength('            LTRIM            ')
select datalength('LTRIM            ')
select len('         LTRIM          ')
select Rtrim('            SCODEEN            ') --'LTRIM            '
select ltrim('            LTRIM           ')
select datalength(TRIM('            LTRIM                         '))
```

--7.Reverse()

--The REVERSE() function reverses a string and returns the result.

--synatx : REVERSE(string)

select REVERSE('PUNE')

select REVERSE('MITHALI')

--8.Round

--The ROUND() function rounds a number to a specified number of decimal places.

--Syntax : ROUND(NUMERIC_EXPRESSION, length, [(function)])

--NUMERIC_EXPRESSION : it takes the number to be roundoff.

--Length : the number of digits that we want to round off.

-- if length is +ve then rounding is applied after decimal and if length is -ve the before decimal

--function : is used to indicate rounding or truncation operation.

-- 0 -indicates rounding and non-zero indicates truncation, by default it is 0.

select ROUND('value',1) -- Exception

select ROUND(74.85,1)

select ROUND(789.56,-1)

select ROUND(789.56,-1,1)

select ROUND(719.56,-2)

select ROUND(78.56,1)

select ROUND(78.56,2)

select ROUND(78.467,0)

select round(750.556,2) -- Round to 2 places after the decimal point

select round(750.556,2,1) --Truncate anything after 2 places after the decimal point.

select round(750.4456666,-2,1)

select round(750.4456666,3)

select round(750.4456666,3,1)

select round(750.4454666,3)

-- previous number <o.5>= Next number

--9.REPLACE()

--The REPLACE() function replaces all occurrences of a substring within a string, with a new string.

--Note: The search is not case-insensitive.

--Syntax - REPLACE(string/Col_NAME, old_string/Old_VALUE, new_string/New_VALUE)

-- A-a , B-b meaning is same in replace function.

select replace ('SCODEen','E','M')

```
select *,REPLACE (S_NAME,'R','U') from student
select *,REPLACE (S_ID,1,10) from student

--10.REPLICATE()
--The REPLICATE() function repeats a string a specified number of times.
--Syntax :REPLICATE(string, integer)
select replicate ('SODEEN ',4)

select *,replicate (S_NAME,4) as FourTime from student

--11.CONVERT()
--The CONVERT() function converts a value (of any type) into a specified datatype.
--Syntax :CONVERT(data_type[(length)], expression/Col_NAME, [(style)])

create table DOJ (id int, NAME varchar(20),DOJ datetime)

insert into DOJ values (1,'Mansa','2020-01-01 10:10:10')
insert into DOJ values (2,'Vasavi','2015-06-01 10:20:10')
insert into DOJ values (3,'Pravlika','2014-04-01 11:10:10')
insert into DOJ values (4,'Jyoti','2017-08-01 12:10:10')
insert into DOJ values (5,'Pushpa','2016-05-01 01:23:10')
insert into DOJ values (6,'Seema',GETDATE())
select * from DOJ
select GETDATE()
select convert(varchar,getdate(),2)
select convert(varchar,getdate(),102)

select *, convert(varchar(11),DOJ) as NewCreatedDOJ from DOJ


select *, convert(varchar(11),DOJ,1) as NewCreatedDOJ from DOJ


select *, convert(varchar(11),DOJ,112) as NewCreatedDOJ from DOJ


--Style
--Converting datetime to character:
--Without      With      Input/Output                    Standard
--century      century
--0            100       mon dd yyyy hh:miAM/PM          Default
--1            101       mm/dd/yyyy                      US
--2            102       yyyy.mm.dd                      ANSI
--3            103       dd/mm/yyyy                 British/French
--4            104       dd.mm.yyyy                    German
--5            105       dd-mm-yyyy                      Italian
--6            106       dd mon yyyy                       -
--7            107       Mon dd, yyyy                      -
--8            108       hh:mm:ss                          -
--9            109     mon dd yyyy hh:mi:ss:mmmAM (or PM) Default
+ millisec
```

--10 110 mm-dd-yyyy USA

--11 111 yyyy/mm/dd Japan

--12 112 yyyymmdd ISO

--13 113 dd mon yyyy hh:mi:ss:mmm Europe (24 hour clock)>

--14 114 hh:mi:ss:mmm 24 hour clock

--20 120 yyyy-mm-dd hh:mi:ss ODBC canonical (24 hour clock)

--21 121 yyyy-mm-dd hh:mi:ss.mmm ODBC canonical (24 hour clock)

-- 126 yyyy-mm-ddThh:mi:ss.mmm ISO8601

-- 127 yyyy-mm-ddThh:mi:ss.mmmZ ISO8601 (with time zone Z)

-- 130 dd mon yyyy hh:mi:ss:mmmAM Hijiri

-- 131 dd/mm/yy hh:mi:ss:mmmAM Hijiri

--12.CAST()

--The CAST() function converts a value (of any type) into a specified datatype.

--Syntax :CAST(expression AS datatype [(length)])

select convert(varchar,GETDATE(),101)

select cast(getdate() as varchar)

select * from DOJ

select *,CAST(DOJ as varchar) as DOJConverted from DOJ

select *,convert (varchar,DOJ,101) as DOJConverted from DOJ

--CLASS-19

--13.CHARINDEX()

--The charindex() function searches for a substring in a string and returns position.

--if the substring is not found, this function returns 0.

--syntax : CHARINDEX(Substring,string,[start])

--Q.How will you findout the place of charecter 'E' in 'SCODEEN'?

select CHARINDEX ('g','Scodeen@global.com')

select SUBSTRING ('Scodeen@global.com',9,LEN('Scodeen@global.com'))

select *,CHARINDEX('@',email)+1 as start1,LEN(email) as end1 from student

select *,SUBSTRING(email,CHARINDEX('@',email)+1,LEN(email)) as Domain from student

--Q.How to find domain or server from email column ?VVVIMP****

select S_ID,SUBSTRING([email],CHARINDEX('@',[email])+1,len([email])) as domain from student

select substring (email,charindex ('@',email)+1,LEN(email)) from student

select charindex ('@','Praveen123@gmail.com')

select CHARINDEX('@',[email]) from student --11+1 =12

select len(email) from student -- 20

--Q.How to find the number of occurance of 'e' chareter in string? L&T

SELECT DATALENGTH('lleelleellee') ---12

select datalength ('lleelleelleellll') -len(REPLACE('lleelleelleellll','e','')) --17-11 =6

--Occurance of 'a'

select (DATALENGTH('aaaakkkkyyyyaaaa'))-len(REPLACE('aaaakkkkyyyyaaaa','a',''))

--Over Clause

--Over by Clause is used to detrmine which row has been applied to the fuction.

--Over clause combined with a partition by clause and is used to divide data into partitions.

--syntax:

--function() over (partition by col1,col2 ...etc)

--Along with functions like Count(),AVG(),Max(),Min(),sum(),rank(),dense_rank() and rownumber() etc.

create table over_Test(EMPID int, FirstName varchar(20),Gender varchar(2),salary int)

```
insert into over_Test values(1,'Mohini','F',1000)
insert into over_Test values(2,'Rohit','M',2000)
insert into over_Test values(3,'Amit','M',4000)
insert into over_Test values(4,'Sonal','F',5000)
insert into over_Test values(5,'Minal','F',6000)
insert into over_Test values(6,'Amar','M',3600)
insert into over_Test values(7,'Shital','F',4500)
insert into over_Test values(8,'Sohil','M',6000)
insert into over_Test values(9,'praveen','M',9000)
insert into over_Test values(10,'Mithali','F',9000)
insert into over_Test values(11,'seema','F',9000)
insert into over_Test values(12,'meena','F',9000)
```

```
select * from over_Test
```

```
---Aggregated data
select gender, count(*) as Totalcount ,Avg(salary) as avg1 , max(salary)
as Max1 ,MIN(salary) as min1,SUM(salary) as sumsal from over_Test
group by Gender

--Non aggregated as well as aggregated data
select EMPID,firstname,o1.Gender,
gen.Totalcount,gen.Max1,gen.min1,gen.avg1
from over_Test o1
inner join
(select gender, count(*) as Totalcount ,Avg(salary) as avg1 , max(salary)
as Max1 ,MIN(salary) as min1 from over_Test
group by Gender) as gen
On o1.Gender = gen.Gender;

--We can avoid bunch code by using simplyOVER Clause
select EMPID,FirstName,salary,Gender

,COUNT(gender) over (Partition by gender ) as Totalcount
,AVG(salary) over () as completeavg1
,AVG(salary) over (Partition by gender) as avg1
,min(salary) over (Partition by gender ) as minsal
,max(salary) over (Partition by gender ) as maxsal
,sum(salary) over (Partition by gender ) as sumsal
,sum(salary) over (Partition by gender order by empid ) as Runningsal
from over_Test
```

```
--Aggreagted male and female details
select *,gender, count(*) as totalcount ,avg(salary) as avgsal
,max(salary) as maxsal, min(salary) as minsal from over_Test
group by Gender

select *, COUNT(Gender) over  (Partition by Gender)
,avg(salary) over  (Partition by Gender) as avgsal
,max(salary) over  (Partition by Gender) as maxsal,
min(salary) over  (Partition by Gender) as minsal
from over_Test

 select FirstName,gender, count(*) as totalcount ,avg(salary) as avgsal
,max(salary) as maxsal, min(salary) as minsal from over_Test
group by Gender,FirstName

--Aggregated data with name and employee details by using join but the
query is complex
select firstName,Salary,o.Gender, g1.totalcount,g1.avgsal
,g1.maxsal,g1.minsal,g1.TotalSal
from over_Test O
Inner Join
(select gender, count(*) as totalcount ,avg(salary) as avgsal ,max(salary)
as maxsal, min(salary) as minsal,sum(salary) as TotalSal from over_Test
group by Gender) as g1
ON g1.Gender =o.Gender;
```

--Instead of writing the above statement we can replace by using over clause

```
select * from over_Test


select *,
COUNT(gender) over (Partition by gender) as TotalCount
,Sum(salary) over (Partition by gender) as TotalSalary
,MAx(salary) over (Partition by gender) as TotalSalary
,Min(salary) over (Partition by gender) as TotalSalary
,Avg(salary) over (Partition by gender) as TotalSalary
,sum(salary) over (Partition by gender order by emp_ID) as TotalRunningSalary
from over_Test


select empid,FirstName,salary,Gender
,count(gender) over (Partition by gender) as totalcount
,avg(salary) over (Partition by gender) as avgsal
,max(salary) over (Partition by gender) as maxsal
,min(salary) over (Partition by gender) as minsal
,sum(salary) over (Partition by gender) as TotalSalary
,sum(salary) over (Partition by gender order by empid) as Totalrunningsalary
from over_Test
```

--Q. Need to calculate the Running salary from emp_over table over partition by Gender.

--CLASS -20

--RANK,DENSE_RANK AND ROW_NUMBER

--Rank() and DenseRank()

--It will return a rank starting at 1 based on ordering of rows and imposed by order by clause.

--Order by clause is required mandatory.

--PARTITION BY Clause is optional.

```
--500/ 496,488,488,488 ,485,481
---         1,   2,   2,   2,   3,   4  --- Dense_Rank
---         1,   2,   2,   2,   5,   6  --- Rank
```

--Rank Syntax: RANK() OVER (ORDER BY col1,col2,....coln ASC/DESC [PARTITION BY Col1,col2...coln])

--Dense_Rank Syntax: DENSE_RANK() OVER (ORDER BY col1,col2,....coln ASC/DESC [PARTITION BY Col1,col2...coln])

--example

--Marks =496,496,495,494,494,490

--rank = 1,1,3,4,4,6

--Dense_rank = 1,1,2,3,3,4

--Example:

--[sal] = [1000,1000,2000,3000,4000]

--Rank() -- [1,1,3,4,5]

--Dense_rank() --[1,1,2,3,4] -- school level mark inside the class

select * ,rank() over (order by salary) as rank1 from over_Test

select * ,dense_rank() over (order by salary) as denserank from over_Test

--Q.What is the diffrence between Rank() and Dense_Rank()

--Rank() -- Rank function skips ranking if there is same value or number.

--Dense_Rank() --It will not skips ranking if their is same value or number.

select * from over_Test order by salary desc

--2nd highest salary by using rank()

select min(salary) as second1 from over_Test where salary in

(select top 2 salary from over_Test order by salary desc)

select min(salary) as second1 from over_Test where salary in

(select top 3 salary from over_Test order by salary desc)

select * from over_Test order by salary desc

-- highest salary by using rank and dense rank select * ,rank() over (order by salary desc) as rank1 from over_Test where rank() over (order by salary) = 2

select * ,dense_rank() over (order by salary desc) as denserank from over_Test where dense_rank() over (order by salary)=2

--The above query will through an exception in

--i.e. Windowed functions can only appear in the SELECT or ORDER BY clauses.

--In order to avoid this kind of exception or Error in SQl we have to use CTE i.e COMMON TBALE EXPRESSION

--CLASS-21

--CTE (Common Table Expression)

--It is temporary result set.

--It will store the temporary results to make use of that in your main query.

--It can be referred within a SELECT,INSERT,UPDATE and DELETE statements that immediately follows the CTE.

--Only DML(S_UID) type of operation we can perform on CTE

--Syntax

--With CTE_NAME [(COL1, COL2,.....etc)]

--AS

--CTE_Query

```
with RankResult as
(select * ,rank() over(order by salary) as rank1 from over_test)
select * from RankResult where rank1 =7

with DenseRankResult AS
(select * ,dense_rank() over (order by salary desc) as denserank from over_Test )
select * from DenseRankResult where denserank = 2

select *,RANK() over (order by salary desc) as Rank1 from over_Test
select *,DENSE_RANK() over (order by salary desc) as DenseRank1 from over_Test

select *,RANK() over (order by salary desc) as Rank1 ,DENSE_RANK() over (order by salary desc) as DenseRank from over_Test

--1.Second highest salary by using max and top function
select min(salary) from over_Test where salary in
(select top 2 salary from over_Test order by salary desc);

--2.Third highest salary is same as second because it will show the records as per TOP clause
select min(salary) from over_Test where salary in
(select top 3 salary from over_Test order by salary desc);
```

--3.When the abpve query implemented by using RANK then again will face the same issue but result will be blank.

with SecondHighest as

(select *,RANK() over (order by salary desc) as Ranks from over_test)

select * from SecondHighest where Ranks =2;

--4.third it shows blank because it has already skipped rank

with thirdhighest as

(select *,RANK() over (order by salary desc) as Ranks from over_test)

select * from thirdhighest where Ranks =3;

--In order solve the issues faced in point 1,2,3 and 4 we can use dense rank function and we can show the correct order of salaries.

with SecondHighest as

(select *,DENSE_RANK() over (order by salary desc) as denseRanks from over_test)

select * from SecondHighest where denseRanks =2;

with thirdhighest as

(select *,DENSE_RANK() over (order by salary desc) as denseRanks from over_test)

select * from thirdhighest where denseRanks =3;

create table students (S_ID int,S_NAME varchar(20),LOC varchar (20),Dept varchar(20))

```
insert into students values(1,'praveen','mumbai','ETL')
insert into students values(2,'Rohit','Mumbai','Testing')
insert into students values(3,'Akash','Jaipur','HR')
insert into students values(4,'Sada','Warangal','HR')
insert into students values(5,'Rajesh','Hyderabad','Account')
insert into students values(6,'Umesh','Kolar','CCD')

select * from students
--Q.How to find Duplicate  from table?
--1. We can identify duplicate records by using PK(Primary Key)
--Select <PK1>,<PK2>,... ,count(*) as duplicate from Table_Name group by <PK1>,<PK2>,... having count(*) > 1

--2.All the column from table need to include in select and group  by list
select S_ID,S_NAME,LOC,Dept,count(*) as duplicate from students group by S_ID,S_NAME,LOC,Dept having count(*) > 1

--If we will try with CTE function then it will through an below exception
--Cannot update the view or function 'CTE' because it contains aggregates, or a DISTINCT or GROUP BY clause, or PIVOT or UNPIVOT operator.
with CTE as
(select S_ID,S_NAME,LOC,Dept,count(*) as duplicate from students group by S_ID,S_NAME,LOC,Dept having count(*) > 1)
delete from CTE where duplicate > 1
```

--ROW NUMBER

--It will return the sequential number of row starting at 1

--Order by clause is required.

--PARTITION BY clause is optional

--When the data is partitioned, row number reset to 1 when the partition changes.

--syantx

--ROW_NUMBER() OVER(ORDER BY Col1,col2)

select *,ROW_NUMBER() Over (partition by S_ID order by S_id) as Rownumber from students

select *, ROW_NUMBER() over (order by salary) as RowNo,

ROW_NUMBER() over (Partition by salary order by salary) as PartitionRowNo from over_Test

--Q.How to delete duplicate records from table?

with DuplicateDelete As

(select *,ROW_NUMBER() Over (partition by S_ID order by S_id) as Rownumber from students)

delete from DuplicateDelete where Rownumber >= 2

select * from students

--Class- 22

--Rename concepts

--In SQL we have in buit Store procedure to Rename the column from table or we can rename the table name.

--we dont have any Keyword called RENAME in SQL server.

--If we work with SQL developer or Teradata then we have RENAME keyword.

--While Changing the column name it will give us warning saying that Changing any part of an object name could break scripts and stored procedures.

--To Rename the columns in a table.

--Syntax: SP_RENAME 'TABLE_NAME.COLUMN_NAME(OLD)','COLUMN_NAME(New)'

```
SP_RENAME 'students.dept', 'Department'
```

```
select * from student
```

```
SP_RENAME 'student.email','EMAIL_ID'
```

----Caution: Changing any part of an object name could break scripts and stored procedures.

--To Rename the Tables in a database

--syntax:SP_RENAME 'DATABASE_NAME.DBO.TABLE_NAME(OLD)',TBALE_NAME(NEW)

```
SP_RENAME 'testing21.dbo.student','newstudent'
----Caution: Changing any part of an object name could break scripts and stored procedures.

--To back up the tables or rename the tables
select * from information_schema.tables where table_name like 'newStu%'
select * from information_schema.tables where table_name like 'Stu%'
select * into Student1 from student

--VIEW
--It is a virtual table.
--It is copy of original table.
--We can perform DDL and DML operation on view

--Syntax :
--CREATE VIEW vVIEW_NAME
--AS
--SQL Statements

--How to Create View ?
Create View vDetailsAB
AS
select A.AID,B.BID,A.ANAME,B.Bname from A JOIN B ON A.AID =B.AID
```

```
--How to select View?
select * from vDetailsAB

--How to Drop view?
drop view vDetailsAB
```

--Inside view we can have only one select statement, if we take two select statement then it will through an exception.

```
--Single select statement
create view vStudentDetails
As
select * from newstudent

--Multiple select statments
create view vTwoSelectStatments
AS
select * from newstudent
select  * from students
```

--NOTE: With use of multiple select statement it will through the below exception

```
--              Exception : Incorrect syntax near the keyword 'select'.
```

```
--Q.How to display the SQL script of view?
sp_helptext vStudentDetails
sp_helptext vDetailsAB


--Insert record in view
select * from vStudentDetails
select * from newstudent
insert into vStudentDetails values (7,'Lipakshi','Madurai','Devops','lipakshi143@microsoft.com')
insert into newstudent values (8,'abc','chennai','Devops','abc@microsoft.com')
--Update record in view
select * from vStudentDetails
select * from newstudent

update vStudentDetails set loc= 'Banglore' where S_ID =1


--Alter view
select * from vStudentDetails


Alter view vStudentDetails
As
select S_id,S_name,EMAIL_ID from newstudent
```

```
--CLASS-23
--Store Procedure(SP)
--It is a block of code to perform certain action whenever it is nedeed.
--It is working like function.

--We can crete two types of Views
--1.Without Parameter
--2.With parameter

--Syntax
--Create procedure / proc spSTORE_PROCEDURENAME
--[(@parametrs) <data type> <SIZE>]
--AS
--BEGIN

--SQL Statements

--END
--1.Without Parameter
Create procedure spTwoSelect
AS
Begin
select * from over_Test
select * from newstudent
END
```

```
--How to execute or select the data from store procedure.
--1
spTwoSelect
--2
EXEC spTwoSelect
--3
EXECUTE spTwoSelect



Alter procedure spTwoSelect
AS
Begin
select * from over_Test

END

--2.With Parameter
create proc sum1(@a int, @b int)
as
begin

declare @sum int

set @sum = @a + @b
```

```
print @sum

END

--==============================
--To execute the SP with parameter we need to supply the values

EXEC sum1 100,200

create proc Calculator(@a int, @b int)
as
begin

declare @addition int
declare @Substraction int
declare @Multiplication int
declare @Division int

set @addition = @a + @b
set @Substraction = @a - @b
set @Multiplication = @a * @b
set @Division = @a/@b
```

```
print @addition

print @Substraction

print @Multiplication

print @Division


END


Exec Calculator 200,20



--Diffrence between Views and Store procedure

--              View
        store Procedure

--1.Views does not accept parameters                  1.SP        accepts
Parameters

--2.Views can contain only single SELECT query          2.SP       contains
several SELECT statements with condition like if-else etc.

--3.By using views we cant perform modification        3.SP        perform
modification to one or more tables.

--      to multiple tables.

--4.Views act as virtual table                          4.SP     acts     as
function.
```

```
--Class-24

--Triggers

--Triggers fired automatically , Once you perform any DML(Insert,delete
and update) operation on table.

--Triggers are also known as special king of store procedure.


--Syntax:

--create trigger tr_trigger_Name

--on TABLE_NAME

--for insert/update/delete

--as

--begin


----trigger/SQL statements---


--END


create table emp1(EID int,EName varchar(20),salary int, gender
varchar(2))


insert into emp1 values (1,'A',300,'M')

insert into emp1 values (2,'V',300,'F')

insert into emp1 values (3,'C',300,'F')

insert into emp1 values (4,'D',300,'M')
```

```
Create table RecordInfo(ID int identity ,recordinfo varchar(300))

create trigger TEST_INSERTED_TABLE
on emp1
for insert
as
begin
            select * from inserted

END

Create trigger trTEST_DELETED_TABLE
on emp1
for Delete
as
begin
            select * from deleted

END

--Q How to drop trigger ?
--1.
drop trigger [trDeleted]
--2
```

--From Object explore navigate to Database from which you want to delete the trigger and go to paricular table expand click on triger after that right click on created trigger and navigate delete and delete it

```
select * from emp1


delete from emp1 where EID =5

delete from emp1 where EID =6


insert into emp1 values (5,'Y',280,'Z')

insert into emp1 values (6,'T',180,'U')

insert into emp1 values (7,'E',110,'F')


--Create a trigger fro inserted records
Create trigger trINSERTED
on emp1
for insert
as
begin
        Declare @EID int
        select @EID = EID from inserted


        insert into RecordInfo values (cast(@eid as varchar(5)) + ' ' +
'Is added at ' +' '+ cast(getdate() as varchar(20)))


END
```

```
--Deleted Trigger
create trigger trDeleted
on emp1
for delete
as
begin
            Declare @EID int
            select @EID = EID from deleted

            insert into RecordInfo values (cast(@eid as varchar(5)) + ' ' +
'Is deleted at ' +' '+ cast(getdate() as varchar(20)))

END

select * from emp1
select * from recordinfo

insert into emp1 values (6,'Mohit',180,'M')
Delete from emp1 where EID =5

insert into emp1 values (5,'Y',280,'Z')
insert into emp1 values (6,'T',180,'U')
insert into emp1 values (7,'E',110,'F')
```

```
--Note:
--Inserted Table is special kind of table used by trigger and it is available only into the the context of trigger.
--Select * from Inserted/Deleted
--1.it returns the copy of rows inserted into the table.
--2.Struture of the inserted table is identical or same , whatever the table on which we have created trigger.

select * from emp1
select * from RecordInfo --trigger
insert into emp1 values (6,'H',600,'F')

--Update Trigger
create trigger trupdateRecord
On  emp1
for update
as
begin

		select * from deleted
		select * from inserted

END

select * from emp1
select * from RecordInfo


update emp1 set EName= 'Tina',gender ='M' where EID =6


--Update trigger info into a table
Create trigger tr_updateEMP
On EMP1
for Update
as
begin


            Declare    @EID  int
            select @eID = eid from deleted


      insert into RecordInfo values (cast(@eid as varchar(5)) +' '+ ' is
updated  at ' +' '+    cast(getdate() as varchar(20)))


END


select * from emp1
select * from RecordInfo


update emp1 set Ename ='Manik',salary = 320,gender ='M' where eid =
1
```

```
--INDEX
--Indexes are used to retrive the data from the database more quikely or fast than anything.
--The user can not see the indexes but they are used to speed up the searches.

--synatx
--Create Index IX_Index_Name
--ON Table Name (Col1,col2....etc)

create table INDEX_TEST1(IID int primary key ,IX_NAME varchar(20))
drop table INDEX_TEST

select * from INDEX_TEST

select * from INDEX_TEST1
create nonclustered index IXnew2
on INDEX_TEST1 (IX_NAME)

sp_helpindex INDEX_TEST1

--How to find out the indexes on table?
sp_helpindex INDEX_TEST

insert into INDEX_TEST1 values (1,'unique')
insert into INDEX_TEST1 values (3,'non-unique')

create Index IX_NAMEINDEX
on INDEX_TEST (IX_NAME)

--How to Drop Index?
--Syntax : DROP INDEX TABLE_NAME.IX_INDEXNAME
drop index INDEX_TEST.IX_NAMEINDEX



--Q.How to create nonclustered index?
create nonclustered index IX_INDEXTEST_NAME
On INDEX_TEST (IX_NAME)

-- two types of indexes
--1.Clustered Index
--2.Non_Clusterd Index
```

--1.Clustered Index

--Whenever you apply clustered indexing in a table, it will perform sorting in that table only.

--You can create only one clustered index in a table like primary key.

--Clustered index is as same as dictionary where the data is arranged by alphabetical order.

--In clustered index, index contains pointer to block but not direct data.

--synatx

--Create Clustered Index IX_Index_Name

--ON Table Name (Col1,col2....etc)

--2.Non_Clusterd Index

--Non-Clustered Index is similar to the index of a book.

--The index of a book consists of a chapter name and page number,

--if you want to read any topic or chapter then you can directly go to that page by using index of that book.

--No need to go through each and every page of a book.

--The data is stored in one place, and index is stored in another place.

--Since, the data and non-clustered index is stored separately, then you can have multiple non-clustered index in a table.

--synatx

--Create NonClustered Index IX_Index_Name

--ON Table Name (Col1,col2....etc)

--Diifrence between clustered and Non-Clusterd index

--CLUSTERED INDEX	NON-CLUSTERED INDEX
--Clustered index is faster.	Non-clustered index is slower.
--Clustered index requires less memory for operations.	Non-Clustered index requires more memory for operations.
--In clustered index, index is the main data.	In Non-Clustered index, index is the copy of data.

--A table can have only one clustered index.   A table can have multiple non-clustered index.

--Clustered index store pointers to block not data.   Non-Clustered index store both value and a pointer to actual row that holds data.

```
CREATE TABLE newemp(

id int primary key identity,

ename varchar(20),

email varchar(50),

department varchar(20))


select * from newemp


---Inserting 1 lac of record into the table
SET NOCOUNT ON


Declare @counter int = 1


While(@counter <=100000)
Begin
	Declare @ename varchar(20) = 'Employee' + RTRIM (@counter)
	Declare @email varchar(50) = 'Batch' + RTRIM (@counter) + 
'@ScodeenGlobal.com'
	Declare @dept varchar(20) = 'ETL/BI/Bigdata' + RTRIM (@counter)
```

```
        Insert into newemp values (@ename,@email,@dept)

        set @counter = @counter + 1

        if (@counter%10000 = 0)
                print RTRIM (@counter) + 'Rows inserted '
END

--Q.Verifying the clustered and non-clusterd indexes
sp_helpindex newemp

select * from newemp where id = 9999;

select * from newemp where ename ='Employee9999';
select * from newemp where email ='Batch7@ScodeenGlobal.com';

Create NonClustered index IX_ENAME
ON newemp (ename)

--Q.How to display last 5 bottom records from  Table ?
select * from newemp where id > (select count(*) from newemp ) -5

select top 5 * from newemp order by id desc
```

--Cursors

--However , if there is ever a need to process the rows, on ROW-by-ROW basis, then cursors are good choice

--Cursors are nothing but more than pointer to row.

```
select * from newemp


--Q.Display ID and Name of each row from new emp table?
declare @id int
declare @ename varchar(30)

Declare Empcursor cursor for
select id,ename from newemp where id <= 1000

open Empcursor

fetch next from Empcursor into @id ,@ename

while (@@FETCH_STATUS =0)
begin

	print 'id = ' +' '+ cast(@id as varchar(10))  + 'ename= ' +@eName ---
id = 1 ename employee1
	Fetch next from Empcursor into @id ,@ename
END
```

```
Close Empcursor

Deallocate Empcursor

--=================================
select * from newemp
Create table EMPSAL (S_ID int ,id int,Sal int)

insert into EMPSAL values (11,5,4000)
insert into EMPSAL values (12,78,5000)
insert into EMPSAL values (13,89,8000)
insert into EMPSAL values (14,100,9000)
insert into EMPSAL values (15,201,7600)
insert into EMPSAL values (16,1001,4000)
insert into EMPSAL values (17,4002,5000)
insert into EMPSAL values (18,4567,8000)
insert into EMPSAL values (19,9876,9000)
insert into EMPSAL values (20,89977,7600)



select * from newemp
select * from EMPSAL

select ne.id ,ne.ename, es.sal from newemp ne join EMPSAL es on ne.id
= es.id
```

```
where ne.ename = 'Employee5' or ne.ename = 'employee78' or
ne.ename like 'employee%'

--Multiple rows update
declare @id int
declare @ename varchar(30)

Declare Empcursor cursor for
select id,ename from newemp where id <= 100000

open Empcursor

fetch next from Empcursor into @id ,@ename

while (@@FETCH_STATUS =0)
begin

        select @ename = ename from newemp where id = @id
        if (@ename = 'Employee5')
        begin
                update EMPSAL set Sal=5432 where id = @id
        End
        else if (@ename = 'Employee78')
        begin
                update EMPSAL set Sal= 7654 where id = @id
        End
        else if (@ename = 'Employe89')
        begin
                update EMPSAL set Sal= 9999 where id = @id
        End
        else if (@ename like 'Employe%')
        begin
                update EMPSAL set Sal= 8888 where id = @id
        End
        Fetch next from Empcursor into @id ,@ename
END

Close Empcursor
Deallocate Empcursor
select * from EMPSAL
```

--Pivot & UNPIVOT--VVIMP --How will rows into table

--Pivot is SQL operator that can be used to extract unique values from one column into multiple columns into the output.

--Syntax:

```
--SELECT <NON_PIVOTED COLUMNS> ,First Pivoted col1,second Pivoyed col2,....Last Pivoted coln
--FROM
--      Select <Query that produces the data> as alias for the source query
--              OR
--              TABLE_NAME
--PIVOT
--(
--              <aggregate function>(column being aggreagated)
--FOR
--     [<Columns that contains the value that will become column headers/Names>]
--IN( [First Col],[Second col].....[Last Col])
-- )
 --AS <alisas for the Pivot table >
```

```
create table ProductSales (Salesagent varchar(100),salescountry varchar(100) , salesamount int)
```

```
insert into ProductSales values ('john','UK',200)
insert into ProductSales values ('tom','Us',190)
insert into ProductSales values ('john','UK',450)
insert into ProductSales values ('David','India',120)
insert into ProductSales values ('Tom','India',240)
insert into ProductSales values ('David','Us',660)
insert into ProductSales values ('tom','Us',320)
insert into ProductSales values ('john','India',280)
insert into ProductSales values ('john','UK',540)
insert into ProductSales values ('David','Us',130)
insert into ProductSales values ('tom','Us',220)
insert into ProductSales values ('tom','UK',290)
insert into ProductSales values ('john','India',400)
insert into ProductSales values ('David','India',300)
insert into ProductSales values ('tom','UK',280)
insert into ProductSales values ('david','UK',400)
insert into ProductSales values ('john','US',700)

select * from  ProductSales


select Salesagent,SalesCountry, sum(salesamount) as TotalSales from ProductSales
group by Salesagent,salescountry
order by Salesagent,salescountry


select * from ProductSales
```

```
--salesagent India   Uk          US
--David          420  400 790
--tom            680  1190       700
--John           240  570 730

select Salesagent, INDIA,UK,US from ProductSales
pivot
(
        sum(salesamount)
        FOR salescountry
        IN (INDIA,UK,US)
) as pivottable

select * from ProductSales
create Table Sales (Salesagent varchar(20),India int,uk int,us int)

insert into Sales values ('David',420,400,790)
insert into Sales values ('jhon',680,1190,1800)
insert into Sales values ('tom',240,570,730)

select * from ProductSales;  --PIVOT

select * from Sales; --UNPIVOT

select Salesagent, country, amount from Sales
```

```
unpivot
(
        amount
        FOR country
        IN (INDIA,UK,US)
) as unpivottable;
```