--SQL - Structured Query language
--PL/SQl - Progarmming Language SQL

--diffrent versions RDBMS
--1.SQL server - Microsoft
--2.SQL developer - Oracle
--3.Tera data - Teradata
--4.DB2 - IBM
--5.MangoDB - Open Source
--6.MySQL - open source etc

--Q.What is data?
--Collection of meaniful information.
--OR
--Collection record information.

--Q. What is database(DBMS)?
--it is collection of data in file format.
--ex:Excel,word file,text file, notepad , notepad++ etc.

--it stores less amount of data
--no relationship between two files or tables

--Q.What is RDBMS(Relatinal data base management system)?
--it is collection of table related information.
--it stores huge amount of data and to extract the data we have a simple language i.e. SQL
--There is relation between two or more tables.

--Q.what is table ?
--it is collection of rows and columns.

--There are two types of databases
--1.System defined database
--2.User defined database

--SQL is not a case sensitive language
--The meaning 'A' is always same 'a'
--for ex: 'AMAR' it has same meaning of 'amar'

--Q.How to create the Database testing18?

Create database Testing20

--Q.how to execute SQL statements?
--1.By using Execute tab form top
--2.By pressing F5 key from key board

--Q.How to navigate to Testing18 database?
use testing20

--In SQL
--Blue color indicates system defined keywords
--for ex
create, use, insert etc
--Pink color indicates system defined functions
--for ex
sum,min,max etc

--Data types
--Type of data/value of an object can hold is known as data type.
--A].Numeric data type

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

--2.TINYINT
--It will store the value ranging from 0 to 255

--3.SMALLINT
--it will store value ranging -32768 to 32767

--4.Decimal
--an exact fixed point number

--5. INT
--it stores an integer value i.e. ranging from -2147483648 to 2147483647

--B].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
--2.Real
--it will also store an floating point numbers -3.40E to 38 to 3.40E to 38

--C].String or charecter data type
--1.char - 0-9,a-z,A-Z and Special symbol it will store data as 1 bit
--Static memory allocation and it is having size of 8000 chars
--for ex: char(20) -- AMAR - 4 char reaming 16 blocks of memory waisted because it has been fixed

--2.varchar - - 0-9,a-z,A-Z and Special symbol
--It is dynamic memory allocation and it will store data as 1 bit
--for ex: varchar(20) -- AMAR - 4 char reaming 16 blocks of memory it will releasse has been fixed

```
declare @val1 varchar(8000)='AMARPatil';
print @val1
print datalength(@val1)
print len(@val1)
```

--3.nchar
--It is static memory allocation and it can store 4000 charecters (1 char it will occupy 2bytes)

```
declare @value nchar(4000) = 'AMAR'
print @value
print datalength(@value)
```

```
print len(@value)

--4.nvarchar
--It is dynamic memory allocation and it can store 4000 charecters (1 char it will occupy 2bytes).

declare @value1 nvarchar(4000) = 'AMAR'
print @value1
print datalength(@value1)
print len(@value1)

--D].data and Time data type
--1.date
--It will allow you to insert the date in mulptiple formats
--For Ex: YYYY/MM/DD,DD/MM/YYYY,YYYY/DD/MM etc
select GETDATE()

declare @date1 date = getdate()
print @date1
--2.time
--it will allow you to insert the time in below format
--HH:MM:SS:MS

declare @time time    = getdate()
print @time

--3.Datetime
--It will allow you to insert date and time together.
--YYYY/MM/DD HH:MM AM/PM

declare @datetime datetime    = getdate()
print @datetime

--SQL statements
--There are four types of SQL statements
--1.DDL(Data Defination lamguage) -- Table level operation.
--2.DML(Data Manipulation Language) --Data stored inside the table
--3.DCL(Data Control Language)
--4.TCL(Transction Control Language)


--1.DDL(Data Defination lamguage)
--These statements are basically used to perform structure related operation w.r.to table.
--Create,Drop,Truncate,Alter,Rename (Dr.CAT)

--Create
--This Is DDL SQL statement and used to create database and Table.

--Q.How to create Database ?
Create database Testing20

--Q.How to select or Navigate to a particular database?
--By using USE keyword and database name
use Testing20
--Just go to top scroll bar and click on down arrow and select your database
```

```
--Q.How to create Table?
Create table FirstTable(
FID int,
FirstName varchar(20),
LastName varchar(20),
City varchar(20),
Age int);

--In SQL if you want to terminate SQL statement then use ; at the end of statement.

--2.DML(Data Manipulation Language)
--These statements are used to operate the data stored inside into the table.
--DML statements are used to play with table data.
--below are the diffrent statements
--SELECT,INSERT,UPDATE,DELETE (S_UID)

--SELECT
--Select statement is used to select the data which you have written
--This is DML statement and used to fetch the records from table.
select 88888

select 'Scodeen Global'

select scodeen    --Exception/Error -Invalid column name 'scodeen'.


--Q.How to fetch the complete data from a table?
select * from FirstTable;

--Q.how to select a particular column from table?
select firstname,age from FirstTable

--* - it will indicate we are selecting complete data from table

--Q.How will you insert the data into the table?
--INSERT
--Insert Statement is used to insert the data into the table.
--By two ways we can insert the data into the table
--METHOD-I
--We have to insert the data sequence wise how we have created the table
insert into FirstTable values(1,'Praveen','Patil','Pune',30);

insert into FirstTable values(1,'Praveen','Patil','Pune'); --Erorr -Column name or number of supplied
values does not match table definition.

select * from FirstTable

--METHOD-II
--We dont have restrictions to insert the data as per column sequence defined in table.
--While inserting the data we have to mention column names in insert statement.
insert into FirstTable (FID,FirstName) values (2,'Amit')

insert into FirstTable (FirstName,FID,Age,LastName) values ('Puneet',3,24,'sharma');

insert into FirstTable (Age,FID,FirstName,LastName) values (35,4,'Puskar','Verma')
insert into FirstTable (Age,FID,FirstName,LastName) values (27,5,'Meena','Patil')
```

```
select * from FirstTable


--Clauses/filters
--We have Clausses in Sql to perform filter related information
--1.Where
--2.order by
--3.group by
--4.having

select * from FirstTable

--1.Where
--Where clause is used with comaprision operator,arithmatic and logical operators.
--It used to filter the specific condition or we can select a particlar records from table.

--Operators
--1.Comaprision Operator
--2.Logical Operator
--3.Arithmatic operator
--4.IN and NOT IN
--5.Between and NOT BETWEEN
--6.LIKE

--1.Comaprision Operator
--it is used to compare the condition provided in filter clauses or where clause.
-- = - equal to
-- > - greater than
-- < - less than
-- >= - greater than equal to
-- <= - less than equal to
-- != or <> -not equal to


create table employee
(EID int,
FirstName varchar(20),
LastName varchar(20),
Loc varchar(20),
Dept varchar(20),
salary int)

insert into employee values (1,'Rohan','Mane','Sangali','HR',15000)
insert into employee values (2,'Sheetal','Chavan','Parbhani','Finance',25000)
insert into employee values (3,'Amit','Patil','Latur','HR',16000)
insert into employee values (4,'Riya','Verma','Pune','Account',20000)
insert into employee values (5,'Sita','Sharma','Patna','HR',15000)
insert into employee values (6,'Kirti','Gold','Solapur','Staffing',35000)
insert into employee values (7,'Sohan','Jadhav','Miraj','Account',45000)
insert into employee values (8,'Priyanka','Sharma','Nagpur','Finance',46000)
insert into employee values (9,'Virat','Patil','Jaipur','Staffing',34000)
insert into employee values (10,'Sohil','Khan','Mumbai','HR',33000)
insert into employee values (11,'Ronit','Patil','Miraj','Admin',NULL)
select * from employee where dept = 'HR'
select * from employee where EID > 5
```

```
select * from employee where EID < 4
select * from employee where EID >= 7
select * from employee where EID <= 4
select * from employee where EID <> 7    --not equal to
select * from employee where EID != 7    --not equal to

--Logical Operator
--these operators are used to compare two inputs logically and provide the result.
--1.AND
--It will be just like multiplication

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

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

select * from employee where eid = 1 and dept = 'Finance'
select * from employee where EID =5 and loc ='Patna'

--2.OR
--It will work like addition
--OR operation
--A        B        O/P
--0        0        0
--0        1        1
--1        0        1
--1        1        1

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

select * from employee where eid = 11 or dept = 'Finance'
select * from employee

select * from employee where EID =5 or salary > 40000 or loc ='Pune'



--IN and NOT IN operator
--This operator will allow you to navigate or point out the values whisch mentained or mentioned inside the in clause.
--Not In operator will perform vice-versa opeartion as compared to IN opeartor.

select * from employee where eid => 1,2 -- incoreect syntax because comparision opeartor will allow you to insert only one condition.
```

```
select * from employee where eid in (1,3,4,5)
select * from employee where eid not in (1,3,4,5)

select * from employee where loc in ('Pune','sangali','Miraj')
select * from employee where loc not in ('Pune','sangali','Miraj')


--Between and Not between
--This operator/ clause is used to display the values or records between the range you have specified.
--This operator works along with AND operator.

select * from employee where EID between 2 and 6

select * from employee where Loc between 'A' and 'N'      --A to N

--Not Between
select * from employee where EID not between 2 and 6

select * from employee where Loc not between 'A' and 'N'    -- excluding A to N cities name    <=N  ,ex:na,nb

--Arithmatic operator
--these operators used to perform mathematical operation like +,-,*,/ and %

select * from employee

select *,MonthlyIcrement =salary+1000    from employee

select *,lossofpay = 2*(salary/30) from employee

select *,AnnualPackage =salary*12 from employee

select *,AnnualPackage =salary*12,lossofpay = 2*(salary/30),MonthlyIcrement =salary+1000 from employee



--Aliases
--We can create a temporary or alias name for column.
-- Syntax : column/operation AS alias_name

select (FirstName + ' ' + LastName) as FullName from employee
select (FirstName + ' ' + LastName) FullName from employee

--LIKE
--LIKE operator is used to search for a specified pattern in a column.
--Mostly like operator is used in where clause.
--Like operator used wildcards for searching a pattern
--1. % - Represents zero,one or multiple charecters or numbers./ A substitue for Zero or more characters
--2. _ - Represents one or single charecters./A A substitue for exactly one character.
--3.[Charlist] - Any single charecter in charlist ex: [ABC]
--4.[^Charlist] -any charecter not in charlist
```

--ex: Seeta,meeta,geeta sena, sona siya

--'S%' - start with 'S' chareter and it will display all the names which starts with S.
--'%S' - End with 'S' charetcer and it will display all the names which END with S.
--'%S%' -Anywhere inside record/column if 'S' charetcer and it will display all the names which starts or ends or anywhere inside into a column.

select * from employee where FirstName like 's%' ---at the start of name s

select * from employee where FirstName like '%A' -- at the end of name a

select * from employee where FirstName like '%A%' --anywhere inside or start or end.

--Display the name whose third letter starts with r
select * from employee where FirstName like '__r%' --Kirti,Virat

--Display the name which starts with s and ends with A
select * from employee where FirstName like 's%a'

--Q.How will find the number of tables are present in database?
--By using Information_schema we can find the number of tables
select * from INFORMATION_SCHEMA.tables --Number of tables present in data base

select * from INFORMATION_SCHEMA.tables where TABLE_NAME like 'emp%' -- by using some pattern which you are aware about.

select * from employee where FirstName like '[ARV]%' -- it will display the names which start with A,R and V.

select * from employee where FirstName like '[^ARV]%' -- it will display the name which not start with A,R and V.

select * from employee where FirstName like '%[ARV]'

select * from employee where firstname like '[A-E]%' -- it will diplay all the names which is in range of A to E

select * from employee where firstname like '[A-O]%'

--Q.How will you display the names which ends with r and t?
--Q.How will you display the name whose second last letter is T?

--2.order by
--This clause is used to sort the result in ascending (ASC) or Descending(DESC) order.
--If the column contains NULL value in it and if we are performing order by operation then NULL value should be first in ASC and Last in DESC.

select * from employee order by salary -- this is by default ascending order

select * from employee order by salary ASC

select * from employee order by salary desc

select * from employee order by LastName desc
select * from employee order by LastName
insert into employee values(11,'Meena','','Hyderabad','HR','')

```
insert into employee (EID,FirstName,LOc) values(12,'Roshan','Yavatmal')

select * from employee order by FirstName asc


--NULL Values
--A column with a NULL value is column with NO value
--NULL valu is diffrent from 0(zero) and blank/empty sapce.

select * from employee where salary = NULL

--Q.How to test the NULL values from column?
--There are two ways to check the NULL values from column
--1.IS NULL
--2.IS NOT NULL

select * from employee where salary = NULL   -- Blank /not possible to check by using comaprision/logical/arithmatic opeartor
select * from employee where salary is NULL

select * from employee where salary is not NULL


--2.DML(Data Manipulation Language)
--UPDATE
--Update statement is used to update complete column data or specific record if condition is provided.
-- By using update statement you can only play with table data.

--synatx:
--UPDATE TABLE_NAME SET COLUMN_NAME ='VALUE' where COLUMN_NAME ='CONDITION'



create table UPDATE_DELETE (U_ID int, UNAME varchar(20) ,ULOC varchar(20))

insert into UPDATE_DELETE values (1,'Sagar','PUNE')
insert into UPDATE_DELETE values (2,'Amit','Sangli')
insert into UPDATE_DELETE values (3,'Sarika','Bijapur')
insert into UPDATE_DELETE values (4,'Rohan','Mumbai')
insert into UPDATE_DELETE values (5,'Amrita','Palampur')


select * from UPDATE_DELETE

update UPDATE_DELETE SET ULOC ='Pune' where U_ID >=2

update UPDATE_DELETE SET ULOC ='Jaipur' where U_ID =5

update UPDATE_DELETE SET UNAME ='Sohan' where U_ID =4

--DELETE
--Delete statement is used to delete the data from table row by row.
--By using DELETE statement it is not possible to delete the structure.
--We can delete the table data at one time or row by row by specifying an condition.
```

```
--syntax:
--DELETE TABLE_NAME where COULMN_NAME ='CONDITION'

select * from UPDATE_DELETE

delete UPDATE_DELETE -- it will delete the complete data from table.

delete UPDATE_DELETE where U_ID =5

delete UPDATE_DELETE where U_ID <=2

--1.Data Defination Language(DDL) - DR.CAT
--Along with DDL statements "TABLE" Keyword is mandotory.

--DROP
--DROP statement will delete the table structure as well as table data.
--Drop statement we can drop or delete the database.

--syntax:
--DROP TABLE TABLE_NAME
--DROP DATABASE DATABASE_NAME

DROP table UPDATE_DELETE -- it will delete table data as well as table structutre.

--Q.Diffrence between Delete and Drop?


select * from employee where firstname like '%[_]%'
select * from employee where FIRSTname like '%#_%'escape'#'
--'%[rt]'

--Truncate
--Truncate statement allow you to delete the records from a table at once.
--It wont delete the structure of the table
--In Truncate you can't delete the data Row-By-Row by specifying a condtion.

--syntax : truncate table table_name

create table Truncate1 (U_ID int, UNAME varchar(20) ,ULOC varchar(20))

insert into Truncate1 values (1,'Sagar','PUNE')
insert into Truncate1 values (2,'Amit','Sangli')
insert into Truncate1 values (3,'Sarika','Bijapur')
insert into Truncate1 values (4,'Rohan','Mumbai')
insert into Truncate1 values (5,'Amrita','Palampur')

select * from Truncate1

truncate table truncate1

--Q. What is the diffrence between Delete,Drop and Truncate?
--Q.    What is the diffrence between DML,and DDL statements?
--Q.How will you delete the data from a table at once?

--ALTER
```

```
--Alter statement is used to perform operation on table level attributes/Columns.
--By using Alter
--We can ADD one or More columns.
--We can delete one more columns.
--We can change the data type for a particular column.
--We can increase or decrese the size of particular column.

Create table ALTER_OPERATION(AID int not Null , ANAME varchar(20))
drop table ALTER_OPERATION

insert into ALTER_OPERATION values (1,'Amit')
insert into ALTER_OPERATION values (2,'sumit')
insert into ALTER_OPERATION values (3,'rohit')
insert into ALTER_OPERATION values (4,'anil')
insert into ALTER_OPERATION values (5,'anil123456')

select * from ALTER_OPERATION

--Adding a single column into a table

alter table ALTER_OPERATION ADD loc varchar(20)

---Adding multiple columns in a table
alter table alter_operation add Pincode int,city varchar(20)

--Dropping/Deleting single column from table
alter table alter_operation drop column city

--Dropping/Deleting multiple columns from table
alter table alter_operation drop column loc,pincode

--to increse or decrease the size of a column
alter table alter_operation alter column aname varchar(10)

alter table alter_operation alter column aname varchar(15)

--To define constraint on table
alter table alter_operation add constraint PK primary key (aid)

--by using alter statemnet we    can add contraiunt to the column
--While adding PK constraint to column in a table we need to follow bwlow rules
--1.To define PK on any column it should define with NOT NULL constraint if no records inserted.
--2.If the records are inserted into table and after that if you are trying to define the PK then it
should have NOT NULL and UNIQUE.
alter table alter_operation add constraint PK primary key (aid)

SP_HELP alter_operation

SP_HELP alter_operation

--Functions
--1.min()
--2.max()
--3.count()
--4.TOP
--5.sum()
```

```
--6.avg()
--7.Distinct()

--1.MIN()
--This function will return the minimum value from a selected column

select * from employee order by salary Desc
select min(salary) as minsal from employee --numbers === 0 to 9

select min(FirstName) from employee    -- text value === A to Z

--2.MAX()
--This function will return maximum value from selected column
select * from employee order by salary Desc

select max(salary) as MAX_SAL from employee where salary < --39000
(select max(salary) from employee ---45000
where salary <
(select max(salary) from employee)) --46000


--3.Count()
--This function is used to count the number of records from table or column.
--Count function always accepts one argument.
--It wont count NULL values from the table or column.

select count(*) as EmpCount from employee
select * from employee
select count(loc) from employee

--Q.In count function NULL value can be considered?
--NO, Null value is not considered in count function.
select count(8888) --- 1
select count('SCODEEN') --1

select count() ---- error - The count function requires 1 argument(s).

select count('A','B') ---
select count('SCODEEN') + count(8888) ---2
select count(SCODEEN)    --Invalid column name 'SCODEEN'.

insert into employee values(11,'Ashok',NULL,NULL,NULL,21000)

--4.TOP()
--This function is used to display the top records from table as per specified count.
--This function is very useful when we have large amount of data in table .

select Top 3 * from employee -- it will display the top 3 records from table.

select min(salary) thirdsal from employee where salary in
(select Top 3 salary from employee order by salary desc)


--6th highest salary
select min(salary) as secondMaxsal from employee where salary in
(select top 6 salary from employee order by salary desc)
```

--4th minimum salary
select max(salary) as fourthminsal from employee where salary in
(select top 4 salary from employee order by salary asc)

--Q.How to display the bottom 4 records from table ?

--5.Sum
--this function add all recorfds from a column.
--it will return the total sum value in numeric expression.
--It will ignore NULL values from column.
select * from employee
select sum(salary) as totalsary from employee

select sum(loc) as totalsary from employee --exception : Operand data type varchar is invalid for sum operator.

--6.avg()
--This function is used to find the avg of the column.
--It will ignore the NULL values.

select sum(salary) as totalsary from employee
select count(salary) from employee
select avg(salary) as AVGSAL from employee

--NOTE: In count,sum, and Avg function NULL values are ignored.

--7.Distinct
--This function is used to find the unique records from column.
select * from employee

select distinct(Dept) from employee

select count(distinct(Dept)) from employee

select count(*) from employee where Dept ='Account'

select distinct(salary) from employee

--DDL(Data defination language)
--Rename
alter table employee RENAME To Employee_details -- it will work in SQL devloper or MySql or PostgreySql

--Inside in SQL server we can rename the column names by using internal store procedure i.e. SP_RENAME.
--Syntax: SP_RENAME Tablename.OLDCOLUMN_NAME,NEWCOLUMN_NAME

select * from employee

sp_rename 'employee.dept','department'
--Caution: Changing any part of an object name could break scripts and stored procedures.

```
--Constraints
--Constraints are used to maintain the accuracy and integrity of the data.
--1.Primary Key
--2.Foreign Key
--3.NOT NULL key
--4.Unique Key
--5.Check Key
--6.Default key

--1.Primary Key --PK
--NOT NULL + UNIQUE
--It will always identifies unique record into column of the table.
--PK is used in general with numeric values .

Create table student(S_ID int primary key,
STUDENT_NAME varchar(20),
LOC varchar(20))

insert into student values (1,'praveen','pune')
insert into student values (2,'Rohan','mumbai')
insert into student values (3,'Rohan','mumbai')
insert into student values (NULL,'veen','pune')

select * from student

sp_help student

alter table alter_Operation Drop constraint PK_

sp_help alter_Operation


--Auto Increment
--It will automatically insert or increment the unique values into table once you define the auto increment.

--It will allow you to specify the range of values by which you want to create a unique values.
--Syntax : Column_name IDENTITY(start,diff)

create table BankAccount(Account int identity, --11128871,11128872,11128873
AccName varchar(20),
Branch varchar(20),
City varchar(20))

--or
create table BankAccount1(Account int primary key identity(11128870,1), --11128871,11128872,11128873
AccName varchar(20),
Branch varchar(20),
City varchar(20))


insert into BankAccount values ('Shon','KR PURAM','Banglore')
```

```
insert into BankAccount values ('Rohan','SP Road','Pune')
insert into BankAccount values ('Amit','Katraj','Pune')
insert into BankAccount values ('Mansi','Miyapur','HYD')
insert into BankAccount values ('Sagar','Shivaji Nagar','Sangli')

select * from BankAccount


--2.Foreign Key(FK)
--A FK is column or collection o columns in one table that referes to the primary key in another table.
--NULL value can be allowed in foreign key column.

create table department(DID int primary key identity, Dept varchar(20))

insert into department values('CIVIL')
insert into department values('Mech')
insert into department values('IT')
insert into department values('ECE')

select * from department

create table student (S_ID int primary key identity,S_NAME varchar(20),
DID int foreign key references department(DID) )

insert into student values ('Praveen',2)
insert into student values ('amit',2)
insert into student values ('Ronit',1)
insert into student values ('Meena',4)
insert into student values ('shanmuka',3)
insert into student values ('monika',Null)
insert into student values ('monika',7)

select * from student

--3.NOT NULL
--NOT NULL constraint restrict you to insert NULL values into a column.
--If you define NOT NULL constraint on column then you cant insert the NULL values in it.
--It will allow duplicates.

create table NOTNULL (NID int , FirstName varchar(20) NOT NULL, AGE int NOT NULL)

insert into NOTNULL values (1,'Amrita',27)
insert into NOTNULL values (2,'Amrita',27)
insert into NOTNULL values (3,NULL,27)

select * from NOTNULL

--4.Unique
--It ensures that all the values in a column should be unique or diffrent value.
--It will accept one NULL value into the column.

create table UNIQUE_TEST (U_ID int Unique , FirstName varchar(20) NOT NULL unique, AGE int NOT NULL)

insert into UNIQUE_TEST values (1,'Amrita',27)
```

```
insert into UNIQUE_TEST values (2,'Sangita',27)
insert into UNIQUE_TEST values (NULL,'Arpita',23)

insert into UNIQUE_TEST values (NULL,'mehir',23)

select * from UNIQUE_TEST



--5.Check key
--It ensures that all values in a column statisfies a specific condition.
--Check constarints is used to restrict the value of a column.
--It is just like condition checking before inserting the data into column.

Create table CHECK_KEY(
C_ID int primary key ,
C_Name varchar(10) NOT NULL UNIQUE,
C_AGE int check(C_AGE >18))

insert into CHECK_KEY values(1,'Sumit',19)
--The below statment through an exception while inserting the data
insert into CHECK_KEY values(2,'Ronit',17)
--Exception/Error
--The INSERT statement conflicted with the CHECK constraint "CK__CHECK_KEY__C_AGE__440B1D61".
--The conflict occurred in database "Testing18", table "dbo.CHECK_KEY", column 'C_AGE'.

--6.Default constarint
--Set a default value to column when value is not defined/inserted/specified.

Create table DEFAULT_VALUE(
D_ID int primary key,
D_name varchar(10) NOT NULL Unique,
D_City varchar(10),
D_AGE int check(D_age >=20),
D_LOC varchar(20) default 'Balaji Nagar')

select * from DEFAULT_VALUE
--METHOD-I
insert into DEFAULT_VALUE values(1,'Smita','Jaipur',20,'katraj')
insert into DEFAULT_VALUE values(2,'Amla','Chennai',28,default)
insert into DEFAULT_VALUE values(3,'Asin','Madurai',34,'')

--METHOD-II
insert into DEFAULT_VALUE (D_ID,D_name,D_City,D_AGE) values(4,'Surya','Banglore',43)


--Group by
--Group  by statements are used in conjection witrh aggregate functions to group result-set by one or more columns.
--MIN,MAX,Count,AVG,SUM,
--Syntax:
--select column_name , aggregate_Function(column_name) from Table_NAME
--where Column_name <Condition>
--group by Column_name
```

```
select * from information_schema.Tables

select * from employee order by salary desc

--Q.Find the sum of salary of each department?
select department,sum(salary) as DeptSal from employee
group by department

--Q.How to display Maximum salary from each department?
select * from employee
where salary IN ( select max(salary) from employee group by department)

--NOTE:
--An aggregate may not appear in the WHERE clause unless it is in a subquery contained in a HAVING clause or a select list,
--and the column being aggregated is an outer reference.

--Q.How will you display the second highest salary department wise?
--Q.Display the department name with highest salary?
--Q.In group where we can use 'WHERE' clause and Why?

--HAVING Clause
--Having clause is added in SQL because the WHERE clause not used with aggreagate function.
--Syntax:
--select column_name , aggregate_Function(column_name) from Table_NAME
--where Column_name <Condition>
--group by Column_name
--HAVING Aggregate_Function(Column_NAME) operator Value.

--Q. How to display the department wise total salary is greater    than 70000?
select department,SUM(salary) from employee
--where SUM(salary) > 70000
group by department
having SUM(salary) > 70000

--By Using Group by and Having Clause we can identify the duplicate records from table.
--To identify the duplicate from Table

--1.By using Primary Key?
--select <PK1>,<PK2>....,Count(*) from Table_NAME group by <PK1>,<PK2> having count(*) > 1
select EID,count(*)as Duplicate from employee group by EID having count(*) > 1

--2.Without primary Key
--Without PK
select EID,FirstName,Loc,department,salary,count(*) from employee
group by EID,FirstName,Loc,department,salary
having count(*) > 1

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

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

Create Table A (Aid int, Name varchar(20))
Create Table B (Bid int, Name varchar(20),Aid int)
Create Table C (Cid int, Name varchar(20),Bid int)
select * from A
select * from B

select * from A
INNER JOIN    B
ON A.Aid =B.Aid


select * from A
JOIN    B
ON A.Aid =B.Aid

insert Into A values(1,'Sam')
insert Into A values(2,'tom')
insert Into A values(3,'harry')
insert Into A values(4,'katich')
insert Into A values(5,'kate')


insert Into B values(11,'harry',3)
insert Into B values(12,'katich',4)
insert Into B values(13,'kate',5)
insert Into B values(14,'mate',6)
insert Into B values(15,'sat',7)

insert Into C values(21,'harry',13)
insert Into C values(22,'katich',14)
insert Into C values(23,'kate',15)
insert Into C values(24,'mate',16)
insert Into C values(25,'sat',17)


select A.Aid,A.Name,B.Bid,C.Cid from A join B    ON A.Aid = B.Aid join C On B.Bid = C.Bid

--Outer Join
--1.Left Outer Join/Left Join
```

```
--The LEFT JOIN returns all rows from the left side table, even if there are no matches in the right table.
--
--For Ex: Table A= [1,2,3,4,5] and Table B =[3,4,5,6,7]
--A left join B = [1,2,3,4,5]
--A		B
--1		NULL
--2		NULL
--3		3
--4		4
--5		5

--B left join A = [1,2,3,4,5]
--B		A
--3		3
--4		4
--5		5
--6		NULL
--7		NULL

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

select * from A
select * from B

select * from A left join B ON A.Aid = B.Aid

select * from B left join A ON A.Aid = B.Aid

--2.Left Outer Join/Left Join
--The RIGHT JOIN returns all rows from the right side table, even if there are no matches in the right table.
--It will display complete right table i.e B with all the matcing records from A.
--For Ex: Table A= [1,2,3,4,5] and Table B =[3,4,5,6,7]
--A Right join B = [1,2,3,4,5]
--A		B
--3		3
--4		4
--5		5
--Null		6
--NULL		7

--B right join A = [1,2,3,4,5]
--B		A
--NULL		1
--NULL		2
--3		3
--4		4
--5		5

--Syntax:
--select */Column_name(s) from Table_Name1
--Right JOIN    Table_Name2
```

```
--ON Table_Name1.Column_name =Table_Name2.Column_name

select * from A
select * from B

select * from A left join B ON A.Aid = B.Aid ;

select * from B left join A ON A.Aid = B.Aid ;


select * from A right join B ON A.Aid = B.Aid ;

select * from B right join A ON A.Aid = B.Aid ;

--3.FULL Outer Join/Left Join
--The Full JOIN returns all rows from the both side table.
--It will display complete table A and B.

--For Ex: Table A= [1,2,3,4,5] and Table B =[3,4,5,6,7]
--A FULL join B = [1,2,3,4,5,6,7]
--A         B
--1         NULL
--2         NULL
--3         3
--4         4
--5         5
--Null      6
--NULL      7

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

select * from A Full outer join B ON A.Aid =B.Aid

select A.Aid,B.Aid    from A Full outer join B ON A.Aid =B.Aid
select * from A
select * from B

--4.Equi-Join(inner Join)
--Equi_join is join but without using a join keyword we can join the two or more tables.
--While writing Equi-join will use where clause

select * from A ,B,C where A.Aid=B.aid and b.Bid =c.Bid

--5.Cross Join
--Cross Join is nothing but a cartesian product.

select * from A cross join B

select * from A
select * from B
--By using cross join we can create inner join by providing condition.
select * from a cross join b where a.Aid =b.Aid
```

```
--Maximum salary from each department
select department, max(salary) as maxsal from employee group by department;

select * from employee;

--2nd maximum salary from each department
WITH Test2nd AS
(
SELECT department,salary,
     DENSE_RANK() OVER (partition by department ORDER BY Salary Desc) AS Rnk
FROM Employee group by department,salary
)
SELECT department,salary
FROM Test2nd
WHERE Rnk=2;

--3.Self join
--Joining a table with itself is nothing but self join.
--self join can be classified as
--1.inner join
--2.outer join(left,right,full)
--3.cross join

create table SELF_TEST_EMP(EID int, ENAME varchar(20),ManagerID varchar(20))

insert into SELF_TEST_EMP values(1,'Shivam',2)
insert into SELF_TEST_EMP values(2,'krishna',4)
insert into SELF_TEST_EMP values(3,'meera',NULL)
insert into SELF_TEST_EMP values(4,'radha',2)
insert into SELF_TEST_EMP values(5,'bali',1)


select s1.EID,s1.ENAME,s2.ENAME from SELF_TEST_EMP S1,SELF_TEST_EMP S2 where S1.ManagerID = S2.EID

--inner join
select * from SELF_TEST_EMP S1 , SELF_TEST_EMP S2 where S1.ManagerID = S2.EID -- by using self join/equi-join we can create inner join

select * from SELF_TEST_EMP S1 join SELF_TEST_EMP S2 on S1.ManagerID = S2.EID --query of inner join
select * from SELF_TEST_EMP;
--left join
select s1.EID,s1.ENAME,s2.ENAME from SELF_TEST_EMP S1 left join SELF_TEST_EMP S2 on S1.ManagerID = S2.EID

--right join
select s1.EID,s1.ENAME,s2.ENAME from SELF_TEST_EMP S2 right join SELF_TEST_EMP S1 on S1.ManagerID = S2.EID
--full join
select    * from SELF_TEST_EMP S1 full join SELF_TEST_EMP S2 on S1.ManagerID = S2.EID


--SET operator
--1.UNION
```

--2.UNION ALL
--3.INTERSECT
--4.EXCEPT

--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 is 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 set1 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 set2 values(8,'H')
insert into set2 values(9,'Hamesha')

alter table set1 alter column SNAME varchar(2)


SELECT * FROM SET1
UNION
SELECT * 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
Union    all
select * from set2
```

--3.Intersection
--It will return only distinct (common records) values from two or more tables.

```
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
  select * from set2

  select * from set1
  Except
  select * from set2

  select Sname from set2
  Except
  select Sname 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

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

select DATEDIFF(YYYY,'1987/09/13','2021/09/13')

select DATEDIFF(HH,getdate(),GETDATE()+2)
--syntax : DATEDIFF(interval,date1,date2)

--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 date,
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')

select * from Account_details


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

--2.DATEPART
--This will allow you to display the date part
--Syntax : DATEPART(interval,date/column_name)

select getdate()
```

```
select DATEPART(HH,GETDATE())

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

select * from Account_details where ACCT_OPEN_DATE in ('2021')

--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.

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


--Null Value replcement
--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()

  --1.ISNULL()
  --This function will help us to replace NULL value with any other user defined value.
  --synatx: ISNULL(Colname,'String')
  --It will accept Two arguments

  --NVL - SQL DEVELOPER _ Oracle
  --IFNULL - MySql

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

insert into NULL_TEST values (1,'Piya','Sohan')


--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

delete from COALESCE_TEST where cid is null
select cid,coalesce(FN,MN,LN) as NonNullValue,sal from COALESCE_TEST

select coalesce(cid,FN,MN,LN) as NonNullValue,sal 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,NULL,'SCODEEN')

select coalesce(NULL,5,NULL,'SCODEEN',3)

select coalesce(NULL,NULL,'SCODEEN',3)
```
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 Manager is present'
                        ELSE Manager
                        END
from NULL_TEST
```

```
select * from NULL_TEST

update NULL_TEST SET Manager =
                        Case
                        when NID =2 then 'PIYA'
                        when NID =4 then 'AMIT'
                        else Manager
                        END
```

```
--Exist and Not Exist
--EXIST is used to check whether the result of co-related nested query is emptyor 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 orders(OID int primary key, CID varchar(5),groceries varchar(20))

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

select * from orders

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

select * from customer C where not exists (select * from orders 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 orders) --(C2,C3,C1,C2,C4,C2)

--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 orders O where C.C_ID =O.CID )

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


--Over Clause
--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','F',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',10000)
 select * 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 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 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.


--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.

--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.

--2nd highest salary by using rank()
select min(salary) from over_Test where salary in
(select top 4 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) as rank1 from over_Test where rank() over (order by salary) = 2

select * ,dense_rank() over (order by salary) 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

--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 type of operation we can perform on CTE

--Syntax
--With CTE_NAME (COL1, COL2,.....etc)
--AS
--CTE_Query


--second and Third highest salary by using max and top function
select min(salary) from over_Test where salary in
(select top 3 salary from over_Test order by salary desc)

--Third highest salary    by using Rank fuction
with HighestSal as
(select * ,rank() over (order by salary desc) as rank1 from over_Test )
select * from HighestSal where rank1 = 3

with HighestSal as
(select *,dense_rank() over (order by salary desc) as denserank1 from over_Test )
select * from HighestSal where denserank1 = 3;


with LowestSal as
(select *,dense_rank() over (order by salary) as rank1 from over_Test )
select * from LowestSal where rank1 = 1;
--second and Third highest salary    by using dense Rank fuction

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

select * from student where S_ID =1

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

--Q.How to find Duplicate    from table?
```

```
--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 (order by salary    ) as RowNo from over_Test

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

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


--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 * from INFORMATION_SCHEMA.tables

select *,UPPER(firstname) as UPPERCASE from employee

--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(LastName) as lowercase from employee

--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',3,2)

select *,SUBSTRING(FirstName,1,1) as FisrtLetter from employee

select *,SUBSTRING(FirstName,5,len(firstname)) as FisrtLetter ,len(firstname) as lenths from
employee


--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 char(20))

insert into length_check values(1,'Praveen')
```

```
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')

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, string2, ...., string_n)
select CONCAT('Scodeen',' ','Global') as concatination

select 'Scodeen' +' ' +'Global' + ' '+'PUNE'

select CONCAT_WS ('@','Scodeen','Global') as conwithWS

select * from employee

select EID,concat(firstName,' ',Lastname) as Full_name    from employee

select EID,firstName +' '+ Lastname as Full_name    from employee

select EID,concat_ws('@',firstName , Lastname) as Full_name    from employee


--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 len('                    LTRIM')
select len(LTRIM('                    LTRIM') )

select len('              LTRIM              ')
select Rtrim('                SCODEEN                ') --'LTRIM                '
select ltrim('                LTRIM                ')
select 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(74.4,0)
select ROUND(789.56,-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, old_string, new_string)

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

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


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

--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,DOJ) as NewCreatedDOJ from DOJ

select *, convert(varchar(11),DOJ,102) as NewCreatedDOJ from DOJ
select *, convert(varchar(11),DOJ,2) as NewCreatedDOJ from DOJ


--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())

select cast(getdate() as varchar)
select * from DOJ
select *,CAST(DOJ as varchar) as DOJConverted from DOJ
select *,convert (varchar,DOJ,102) as DOJConverted from DOJ

--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 ('@','Scodeen@global.com')
select * from student

  update student set email = 'Umesh@yahoo.com' where S_ID =4

--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 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' charetcer in string? L&T
SELECT DATALENGTH('lleelleellee') ---12
select len(REPLACE('lleelleellee','e','')) --6

select                              (datalength('aaaahhhvcidsfbsdcbhhhhhh')                              -
len(REPLACE('aaaahhhvcidsfbsdcbhhhhhh','a','')))
```