Functions : To break big program into smaller subprograms (functions)

**On function should perform how many tasks : ONE**

We don’t want to store all columns related to different entities in a single table

We break a big table into smaller tables

**One table shud contain data that belongs to only one entity**

Primary Key , Foreign Key

In Normalization , while dividing table to smaller tables, we use some rules

Functional Dependency > Tells us that which column is dependent on which column

Keys

Candidate Keys > The columns which has properties of becoming primary key

Primary Key > The key column which is unique

Alternate Key > Left over candidates key, which cannot become primary key

Id Ecode name address marks batch

Candidate Keys : Id , Ecode

Primary key : id

Alternate Key > Ecode

Set Operators

Operators which work on relations / tables /sets

UNION

UNION ALL

INTERSERCT

MINUS use newdb

drop table cricket

drop table hockey

Create table cricket(name varchar(20) , sportsname varchar(10), runs int )

Create table hockey(name varchar(20) ,sportsname varchar(10), score int )

insert into cricket values

('Lalit','crikcet',90),

('Jatin','crikcet', 90),

('Vijay','crikcet', 30),

('Gagan','crikcet',34),

('Pawan','crikcet',89),

('Sagar','crikcet',10),

('Deepak','crikcet',50)

insert into hockey values

('Karan','hockey', 90),

('Pradeep','hockey', 10),

('Vijay','hockey', 30),

('Gagan','hockey', 34),

('Om','hockey', 89),

('Sagar','hockey', 10),

('Deepak','hockey', 50)

select name , runs from cricket

select name , score from hockey

Give you all the players who plays either game (With UNION, it gives single record)

select name , runs from cricket

UNION ALL

select name , score from hockey

Give you all the players who plays either game (With UNION ALL , it gives duplicate records)

select name , runs from cricket

UNION ALL

select name , score from hockey

Gives you records who are playing both games (Common records)

select name , runs from cricket

INTERSECT

select name , score from hockey

Gives you records who only plays cricket (MINUS) ( Table1 - Table2)

select name , runs from cricket

EXCEPT

select name , score from hockey

Gives you records who only plays hockey (MINUS) ( Table1 - Table2)

select name , score from hockey

EXCEPT

select name , runs from cricket

SubQueries : Nested Query

Outer Query OPERATOR (INNER QUERY)

Output of Inner Query will become condition for Outer Query

Inner Query can give you output (records)

Depeneding upon no. of records , operator will change

If inner query returns single value, in that case operator can be

=

>

<

>=

<=

<>

If inner query returns single value, in that case operato

If inner query returns more than one value, in that case operator

<ANY

>ANY

>ALL

<ALL

IN

Select \* from emp where salary > ALL( 167,9089,89877)

select \* from emp

-- Give max salary

select max(salary) from emp

-- Give name of employee who gets max salary

select empname from emp

where salary = (Select max(salary) from emp)

-- Give name of employees who gets salary more han ahri & Deepak

select empname from emp

where salary < ALL (select salary from emp

where empname IN ('Hari','Deepak'))

select empname from emp

where salary IN (select salary from emp

where empname IN ('Hari','Deepak'))

-- Arrange Records

-- order by clause

select \* from emp order by salary desc

select \* from emp order by address , salary desc

select address from emp

select distinct(address) from emp

select count(address) from emp

select count(distinct(address)) from emp

Rank Function

CREATE TABLE StudentScore

(

Student\_ID INT PRIMARY KEY,

Student\_Name NVARCHAR (50),

Student\_Score INT

)

GO

INSERT INTO StudentScore VALUES (1,'Ali', 978)

INSERT INTO StudentScore VALUES (2,'Zaid', 770)

INSERT INTO StudentScore VALUES (3,'Mohd', 1140)

INSERT INTO StudentScore VALUES (4,'Jack', 770)

INSERT INTO StudentScore VALUES (5,'John', 1240)

INSERT INTO StudentScore VALUES (6,'Mike', 1140)

INSERT INTO StudentScore VALUES (7,'Goerge', 885)

select \* from StudentScore

select student\_name , student\_score,

row\_number() over (order by student\_score desc)

AS "Ranking" from studentScore

select student\_name , student\_score,

rank() over (order by student\_score desc)

AS "Ranking" from studentScore

select student\_name , student\_score,

dense\_rank() over (order by student\_score desc)

AS "Ranking" from studentScore