Class\_7

--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

--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.

--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 Top 3 salary from employee order by salary desc

--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

-------------------------------------------------------------------------------------------------

Class\_8

--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

--2.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

--3.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

--4.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)

------------------------------------------------------------------------------------------------

Class\_8

--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

--.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

--.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,'')

--------------------------------------------------------------------------------------

Class\_9

--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

--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

select EID,FirstName,Loc,department,salary,count(\*) from employee

group by EID,FirstName,Loc,department,salary

having count(\*) > 1

----------------------------------------------------------------------------------------

Class\_10

--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,'Ha')

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

------------------------------------------------------------------------------------

Class\_11

--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

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

---------------------------------------------------------------------------------------------------

Class\_12

--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.

--------------------------------------------------------------------

Class\_13

--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.

--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

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

-------------------------------------------------------------------------------------------------

Class\_14

------Equi\_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

select \* from A\_1 ,B\_1,C\_1 where A\_1.aid=B\_1.aid and B\_1.bid=C\_1.bid;

create table EMP\_new (id int, name varchar (10),Company varchar (10),Work varchar (10));

insert into EMP\_new values (1,'Amit','Info','pune')

insert into EMP\_new values (2,'Puja','Tcs','Mumbai')

insert into EMP\_new values (3,'Poonam','Tech','Pune')

insert into EMP\_new values (4,'Abhi','Logic','Nagpur')

insert into EMP\_new values (5,'Kirti','Lim','Nagar')

select \* from EMP\_new

create table Job (salary int,base varchar (10),id int)

insert into job values (10000,'Pune',1)

insert into job values (20000,'Mumbai',3)

insert into job values (30000,'Nagpur',4)

insert into job values (40000,'Pune',5)

insert into job values (35000,'Nagar',2)

select \* from Job;

select \* from EMP\_new,job where EMP\_new.id=job.id and EMP\_new.work=job.base;

Q. Find EMP name who worked in a department having location same as their address. (Equi Join)

-----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

select \* from A\_1 cross join B\_1 cross join C\_1

---Self join

--Joining a table with itself is nothing but self 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 \* from SELF\_TEST\_EMP

select \* from SELF\_TEST\_EMP as T1 ,SELF\_TEST\_EMP as T2 where T1.ManagerID = T2.EID

select T1.eid ,T1.Ename from SELF\_TEST\_EMP as T1 ,SELF\_TEST\_EMP as T2 where T1.ManagerID = T2.EID

Q. Display name and respected manager name. (Self join)

Use same table -SELF\_TEST\_EMP