Lesson 3

**Querying Data by using Joins and Subqueries**

**Querying Data Using Joins**

drop table employee;

drop table salary;

create table employee (empid int,empname char(30));

insert into employee values (001,'sha');

insert into employee values (002,'radha');

insert into employee values (003,'balaji');

select \* from employee;

create table salary ( empid int,empsal money );

insert into salary values (001,2000);

insert into salary values (003,5000);

insert into salary values (005,10000);

select \* from salary;

select \* from employee;

select \* from salary;

**Equi join**

Note : A join that uses an asterisk sign in select list. it displays redundant column data in the result set is terminated as equi join

select \* from employee e join salary s on e.empid=s.empid;

**Natural join**

Note : A join that restricts the redundant column data from the result set is known as a natural join.

select e.empid, e.empname, s.empsal from employee e join salary s on e.empid=s.empid;

select e.empid, e.empname,s.empid, s.empsal from employee e join salary s on e.empid=s.empid;

**Left outer join**

Note : Left outer join ensures the inclusion of all rows from the first table and the matching rows from the another table

select e.empid, e.empname, s.empsal from employee e left outer join salary s on e.empid=s.empid;

select s.empid, e.empname, s.empsal from salary s left outer join employee e on e.empid=s.empid;

**Right outer join**

Note :

Right outer join ensures the inclusion of all rows from the right side table and the mathing rows from the left side table

select e.empid, e.empname, s.empsal from employee e right outer join salary s on e.empid=s.empid;

select e.empid, e.empname,s.empid, s.empsal from employee e right outer join salary s on e.empid=s.empid;

**Cross Join**

Note : A join that includes more than one table using the keyword CROSS is called a cross join. ( Cartesian Product)

select e.empid, e.empname, s.empsal from employee e cross join salary s ;

select \* from employee e cross join salary s ;

**Inner Join**

Note : In Inner join , Data from multiple table can be displayed after comparing values present in a comman column. Only the rows with values satisfying the join condition in the comman column are displayed. that

select e.empid, e.empname, s.empsal from employee e inner join salary s on e.empid=s.empid;

selecte.empid,e.empname,s.empsalfromempeinnerjoinsalarys

one.empid=s.empidands.empsal=2000

**Self Join:**

Note : A join is said to be a self join when one row in a table correlates with other rows in the same table . since the same table is used twice for comparison, An alias name differentiates the two copies of the table. All join operators exept the outer join operators can be used in a self join.

select s1.empid,s2.empid,s1.empsal from salary s1 join salary s2 on s1.empsal=s2.empsal where s1.empsal=5000;

select s1.empid,s2.empid,s1.empsal from salary s1 join salary s2 on s1.empsal=s2.empsal where s1.empsal=5000.00;

select e1.empid,e2.empid,e1.empname from employee e1 join employee e2 on e1.empname=e2.empname where e1.empname='radha';

drop table employee;

drop table salary;

**Example for creating join more than two tables :**

selecte.empid,e.empname,d.dname,d.dsal,o.status

fromemployeeejoindeptd

one.empid=d.empidjoin

order2oond.empid=o.OrIdwhere

Customer\_Namein('aaa','gg')

**Using IN , NOT IN, EXISTS , NOT EXISTS Keywords**

**COMPARISION OPERATORS**

drop table emp;

drop table sal;

drop table emp;

create table emp (eid int, efirstname char(15), elastname char(15), ecity char(25));

insert into emp values (001, 'rama', 'krishnan', 'coimbatore');

insert into emp values (002, 'radha','krishnan', 'salem');

insert into emp values (003, 'balaji','s', 'madurai');

insert into emp values (004, 'moideen', 'babu', 'salem');

insert into emp values (005, 'alagiri','ajay', 'chennai');

insert into emp values (006, 'krishna','kumar', 'madurai');

select \* from emp;

drop table dept;

create table dept (did int, empid int, dname char(20), dsal float);

insert into dept values (1001,001, 'acc', 5000);

insert into dept values (1001,002, 'man', 10000);

insert into dept values (1002,003, 'acc', 5000);

insert into dept values (1002,004, 'man', 10000);

insert into dept values (1003,005, 'civ', 12000);

insert into dept values (1003,006, 'civ', 15000);

select \* from dept;

select \* from emp;

select \* from dept;

Note : Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subquery is used as an expression.

select \* from emp where eid = (select empid from dept where dsal<10000);

**in**

select \* from emp where eid in (select empid from dept where dsal<10000);

**not in**

select \* from emp where eid not in (select empid from dept where dsal<10000);

**in**

select \*from emp where eid in(select empid from dept where dsal>=10000);

**!= (not equal to)**

select \*from emp where eid in(select empid from dept where dsal!=10000);

**Any** (Note : it means the minimum value in the list)

The expersion | column \_name >all ( 10,20,30 )means ‘Least than 30 ’

select \* from emp where eid > any (select empid from dept where dsal<10000);

**All**

(Note : it means maximum value in the list)

The expersion | column \_name >all ( 10,20,30 )means ‘greater than 30 ’

select \* from emp where eid > all (select empid from dept where dsal<10000);

select \* from dept

**Any**

select \* from emp where eid > any (select empid from dept where dsal>12000);

**=Any** (Note : it means any of the value in the list)

The expersion | column \_name = any ( 10,20,30 )means ‘ equal to either 10, 20,or 30’

select \* from emp where eid = any (select empid from dept where dsal>12000);

**All**

select \* from emp where eid = all (select empid from dept where dsal>12000);

**exists**

Note : The EXISTS keyword is not preceded by any column name, constant, or other expression , and it contains an asterisk (\*) in the SELECT list of the inner query

**Eg :**

select \* from emp where exists (select \* from dept where dsal=16000);

select \* from emp where exists (select \* from dept where dsal=12000);

select \* from emp where exists (select \* from dept where dsal=5000);

**not exists**

select \* from emp where not exists (select \* from dept where dsal=3000);

select \* from emp where not exists (select \* from dept where dsal=5000);

**AGGREGATE FUNCTIONS**

select \* from emp where eid > (select avg(empid) from dept);

select \* from emp where eid > (select avg(empid) from dept where dsal>5000);

select \* from emp where eid > (select avg(empid) from dept where dsal>5000);

select \* from emp where eid < (select count(empid) from dept);

select \* from emp where eid < (select count(\*) from dept);

select \* from emp where eid < (select max(empid) from dept);

select \* from emp where eid < (select min(empid) from dept);

select \* from emp where eid < (select sum(empid) from dept);

**Union & Union all**

drop table titles;

drop table authors;

create table titles(titleid int,title char(20),type char(20),advance float);

insert titles values(1,'Java2','comp',100);

insert titles values(2,'Circuits','elec',100);

insert titles values(3,'sysadmn','network',300);

insert titles values(4,'relamgmt','bpo',400);

insert titles values(5,'Circuits','elec',50);

select \* from titles;

create table authors (auid int, au\_name char(20));

insert into authors values ( 001, 'bala' );

insert into authors values ( 002, 'yesw' );

select \* from authors;

**union**

select \* from authors;

select \* from titles;

select au\_name from authors union select title from titles;

**union all**

select au\_name from authors union all select title from titles

**NESTED and CORRELATED Subqueries**

**NESTED Subqueries**

A Subquery can contain one or more subqueries. Subqueries are used when the condition of a query is dependent on the result of another query. Which in turn is dependent on the result of another subquery.

Note : We can implement subqueries upto 32 levels.

The numbar of levels that can be used depends on the memory available on the database

server.

drop table titiles;

drop table titleauthor;

drop table authors;

create table titles(title\_id int,title varchar(20),type varchar(20), advance int);

insert titles values('001','Java2','comp','100');

insert titles values('002', 'Circuits','elec','200');

insert titles values('003', 'sysadmn','network','300');

insert titles values('004', 'relamgmt','bpo','400');

select \* from titles;

create table titleauthor(title\_id int,au\_id int);

insert titleauthor values('001','101');

select \* from titleauthor;

create table authors(au\_id int,au\_name varchar(20));

insert authors values('101','Balaguruswamy');

select \* from authors;

select \* from titles;

select \* from titleauthor;

select \* from authors;

select au\_name from authors where au\_id in (select au\_id from titleauthor where title\_id=(select title\_id from titles where title='Java2'));

select empid from employee

intersect

select empid from salary

 MINUS operator, which returns only unique rows returned by the first query but not by the second:

select empid from employee

minus

select empid from salary

select length('welcome') from dual

select to\_Char(Current\_date,'DD-MON-YYYY HH:MI:SS') from dual

select to\_Char(Current\_date,'DD-MONth-YYYY HH:MI:SS') from dual

select to\_Char(Current\_date,'DD-MM-YYYY HH:MI:SS') from dual

select current\_date from Dual