6) Implementation of

- Group by & Having clause
- order by clause
- Indexing

Group by :-

This query is used to group to all the records in a relation together for each and every value of a specific Key(s) and then display them for a selected set of fields the relation.

Syntax : select column(s) name from table
group by column_name;

Ex: select Emp_no , sum(salary) from Employee
group by Emp_no;

Group by - having :

This was added to SQL because the where keyword could not be used with aggregate functions. It must follow the group by clause in a query and must also precede the order by clause if used.

Ex: select Employee.LastName, count(order.orderID) AS Numberof orders from (order inner Join Employee on order.EmployeeID = Employee.EmpID) Group by LastName having count(order.orderID)>10;

order By :-

This query is used to ~~to~~ display a selected set of fields from a relation in an ordered manner base on some field.

Syntax: select <set of fields> from table order by column_name;

Ex: select emp_no, emp_name, Job_id from employee order by Job_id;

Indexing :-

It is an ordered set of pointers to the data in a table. It is based on data values in one of more columns of table.

Syntax: create Index <index_name) on table_name (attr1, attr2, ... attrn);

Ex: create Index id1 on employee (emp_no, dept_no);

* write the queries for the following

1) consider the following schema

Employees (emp_id, emp_name, Job_category, manger_id, dept_id, salary)

Departments (dept_id, dept_name)

2) Display total salary spent for each job category

3) Display lowest paid employee details under each manager.

4) Display number of employees working in each department and their dept name

5) Display the details of employees sorting the salary in increasing order.

6) show the record of employee earing salary greater than 45000 in each department.

schema diagram :

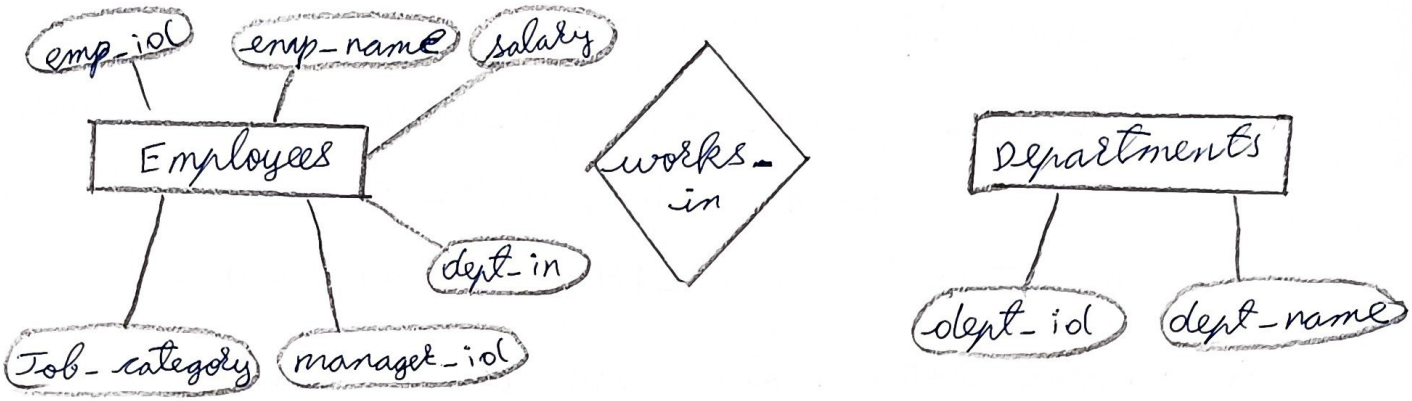
Employees

Emp_id	Emp_name	Job_category	manger_id	dept_id	salary

Departments

dept_id	dept_name

ER- diagram:



1) create table employees (emp_id int(5),
emp_name varchar(20), Job_category varchar(20),
manager_id (5), dept_id int (5), salary int (5));

create table departments (dept_id int (5);
dept_name varchar (20));

insert into employees values
(1, "John", "Manager", NULL, 1, 50000),
(2, "Suneet", "Analayst", 1, 1, 48000),
(3, "satvik", "Analayst", 4, 2, 42000),
(4, "Abhi", "Manager", NULL, 2, 40000),
(5, "Mike", "Manager", 6, 3, 60000);

insert into departments values
(1, "HR"),
(2, "Finance"),
(3, "IT");

select * from employees;

emp_id	Emp_name	Job_category	Manager_id
1	John	Manager	NULL
2	Suneet	Analyst	1
3	satvik	Analyst	4
4	Abhi	Manager	4
5	Mike	Manager	6

dept_id	salary
1	50000
1	48000
2	42000
2	40000
3	60000

select * from departments;

dept_id	dept_name
1	HR
2	Finance
3	IT

2) select Job_category, sum(salary) AS total_salary_spent from employees group by Job_category;

Job_category	Total_salary_spent
Manager	150000
Analyst	90000

3) select manager_id, min(salary) as lowest_salary from employees group by manager_id;

Manager_id	lowest_salary
NULL	50000
1	48000
4	40000
6	60000

4) select dept_name, count(emp_id) as num_employees from employees, departments group by dept_name;

dept_name	num_employees
HR	2
Finance	2
IT	1

5) select * from employees order by salary asc;

Emp_id	Emp_name	Job_category	Manager_id
4	Abhi	Manager	4
3	satvik	Analyst	4
2	sumeet	Analyst	1
1	John	Manager	NULL
5	Mike	Manager	6

dept_id	salary
2	40000
2	42000
1	48000
1	50000
3	60000

6) select * from employees where salary > 45000 order by dept_id

emp_id	Emp_name	Job_category	manager_id
1	John	Manager	NULL
2	sumeet	Analyst	1
5	Mike	Analyst	6

dept_id	salary
1	50000
1	48000
3	60000

Experiment No - 7

7) Study & implementation of

- sub queries
- views

sub queries :-

The query within another is known as a sub query. A statement containing sub query is called parent statement. The row returned by sub query are used by parent statement.

We can place subquery in SQL clauses:

- where clause
- Having clause
- From clause
- operators (in, Any, All, <, > etc).

1) Sub queries with select statement

syntax: select column_name
from table_name
where column_name expression operator
(select column_name from
table_name where ...);

Ex: select * from Employee
where ID in (select ID from Employee
where salary > 4500);

2) Subqueries with insert statement.

Syntax: insert into table_name1 (column1, column2, ...)
select * from table_name2
where value operator;

Ex: insert into Employee_BKP
select * from employee
where ID in (select ID from employee);

3) Subqueries with update statement.

Syntax:
update table set column_name = new_value
where value operator
(select column_name from table_name
where condition);

Ex: update Employee set salary = salary * 0.25
where age in (select age from customers_BKP
where Age >= 29);

4) Subqueries with delete statement

Syntax :

Delete from table_name where value operator
(select column_name from table_name
where condition);

Ex: Delete from employee where AGE in
(select age from employee_BKP where
Age >= 29);

View :-

In sql, a view is a virtual table based on the result set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in database.

Syntax : create view <view_name> AS select
<set of fields> from relation_name
where (condition);

Ex: create view employee AS select emp_no,
emp_name, Job from employee
where Job = "clerk";

updating a view:

A view can be updated by using the following syntax:

```
create view or Replace view view_name
AS select column_name(s)
from table_name where condition;
```

ex:
```
Replace view [Brazil customers]
AS select customer-name, contact, city
from customers
where country = "Brazil";
```

Dropping a view:

A view can be deleted with drop command.

Syntax: Drop view <view_name>;

ex: Drop view [Brazil customers];