Class: SE Comp **Experiment No: 7**

Ashish Jha Roll no. 27 Batch : B

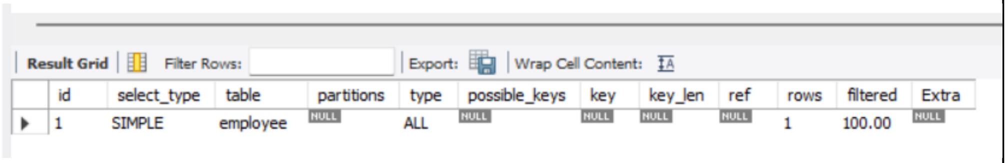
Topic:	Perform Sub Queries, Nested Queries and Joins.
Prerequisite :	Knowledge of concepts sub query, nested query, Joins and SQL syntax.
Mapping With COs:	CSL402.3, CSL402.4
Objective:	- To implement Subqueries, Nested Queries and Joins. - Write different types of problems that can solve by: - Sub queries - Nested queries - Combine data across tables according to their system. (Implement JOIN)
Outcome:	After completion of this lab, the students will understand and be able to do the following: - Describe the types of problems that subqueries can solve. - Sub queries are nested within a SELECT, INSERT, UPDATE, or DELETE statement. - A subquery can be used inside the WHERE or HAVING clauses of the outer SELECT, INSERT, UPDATE, or DELETE statements. - Build and execute sub query. - Define and execute various types of joins.
Instructions:	1. This experiment is a compulsory experiment. All the students are required to perform this experiment individually. 2. Implement Subqueries, Nested Queries and all the types of Joins for the assigned system.
Deliverables:	1. Implement Subqueries, Nested Queries and all the types of Joins for the assigned system. Implementation : use mysql -- creating table employee CREATE TABLE employee(emp_id INT NOT NULL,

```
first_name VARCHAR(20),

last_name VARCHAR(100),

emp_salary int,

dep_id int,

PRIMARY KEY (emp_id)

);


-- creating Table depart

CREATE TABLE department(

dep_id int,

dep_name VARCHAR(20),

manager_id int,

location_id int,

PRIMARY KEY (dep_id)

);


-- insering values in table employee

insert into employee values (100,'erik','john',20000,12);

insert into employee values (101,'steven','cohen',10000,10);

insert into employee values (102,'edwin','thomas',15000,11);

insert into employee values (103,'harry','potter',20000,12);



-- insering values in table department

insert into department values (10,"IT",200,1700);

insert into department values (11,"Marketing",201,1800);
insert into department values (13,"Resource",203,2400);

insert into department values (14,"Shipping",121,1500);



-- Iner join

select e.emp_id,e.first_name,e.last_name,d.dep_id,d.dep_name

from employee e

inner join department d

on e.dep_id = d.dep_id;


-- Left join

select e.emp_id,e.first_name,e.last_name,d.dep_id,d.dep_name

from employee e

left outer join department d

on e.dep_id = d.dep_id;


-- Right join

select e.emp_id,e.first_name,e.last_name,d.dep_id,d.dep_name

from employee e

right join department d

on e.dep_id = d.dep_id;


-- full outer join

select e.emp_id,e.first_name,e.last_name,d.dep_id,d.dep_name

from employee e

left join department d
```

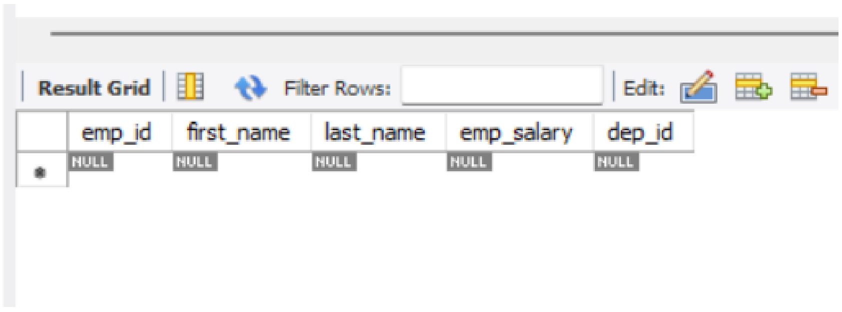
```
on e.dep_id = d.dep_id

union

select e.emp_id,e.first_name,e.last_name,d.dep_id,d.dep_name

from employee e

right join department d

on e.dep_id=d.dep_id;


-- Sql Subqueries


-- Single row Sub-queries


select * from employee

where emp_salary = ( select avg(emp_salary) from employee);


select * from employee

where emp_salary > ( select avg(emp_salary) from employee);


select * from employee

where emp_salary < ( select avg(emp_salary) from employee);


select * from employee;

-- multi row sub-queries


select * from employee

where emp_salary in ( select max(emp_salary) from employee

                    group by dep_id );
```

```
select * from employee

where emp_salary not in  ( select max(emp_salary) from employee

                    group by dep_id );
```

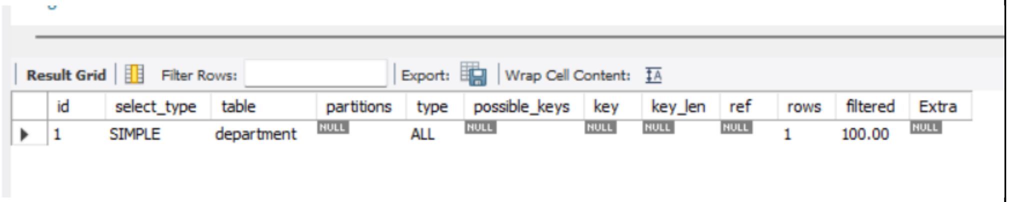
Outputs :



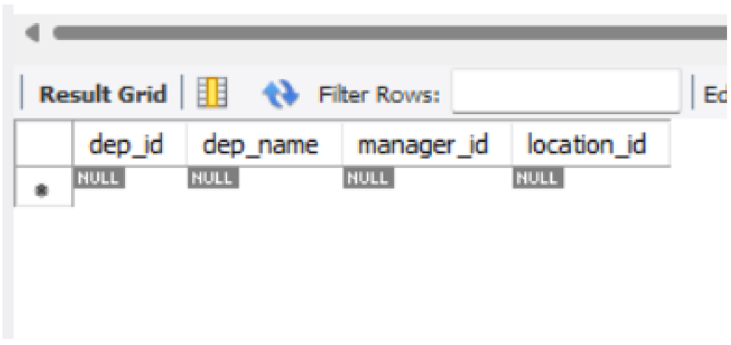
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employee	NULL	ALL	NULL	NULL	NULL	NULL	1	100.00	NULL



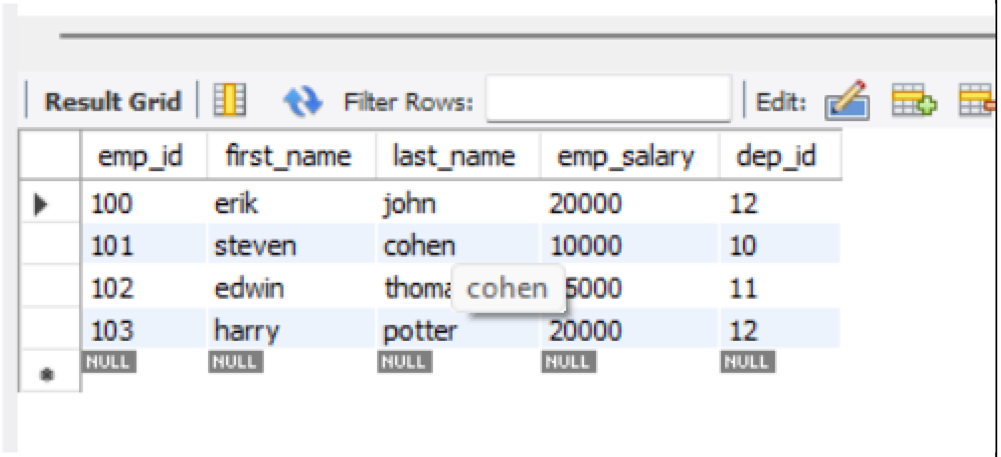
emp_id	first_name	last_name	emp_salary	dep_id
NULL	NULL	NULL	NULL	NULL



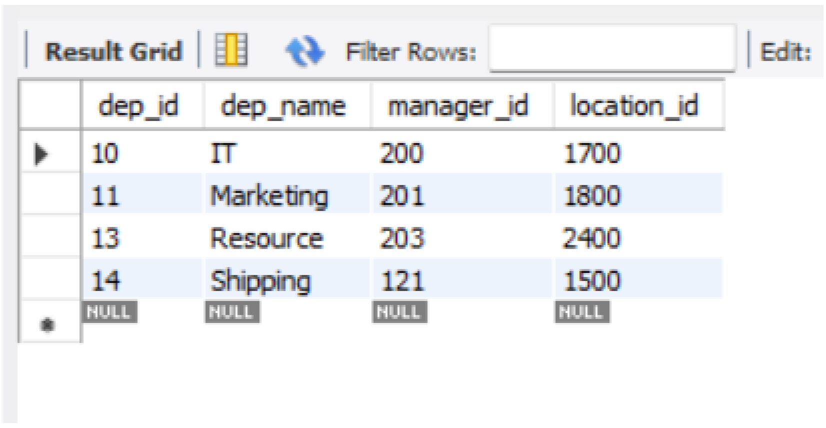
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	department	NULL	ALL	NULL	NULL	NULL	NULL	1	100.00	NULL



dep_id	dep_name	manager_id	location_id
NULL	NULL	NULL	NULL

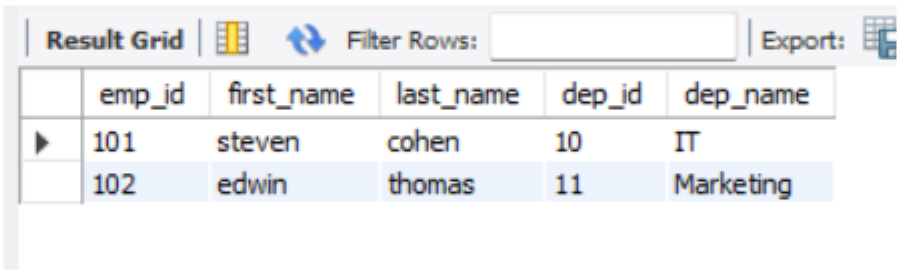


	emp_id	first_name	last_name	emp_salary	dep_id
▶	100	erik	john	20000	12
	101	steven	cohen	10000	10
	102	edwin	thoma cohen	5000	11
	103	harry	potter	20000	12
*	NULL	NULL	NULL	NULL	NULL



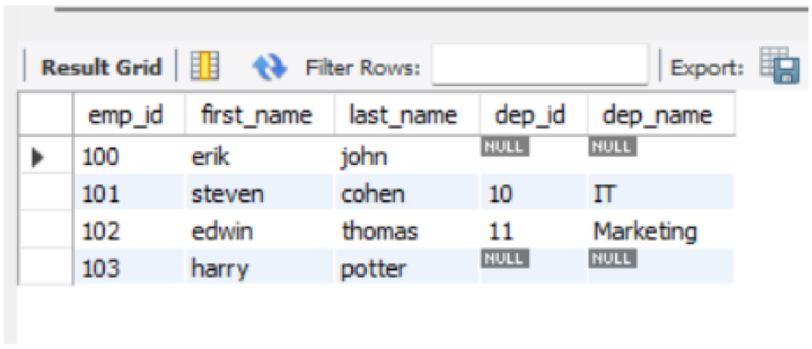
	dep_id	dep_name	manager_id	location_id
▶	10	IT	200	1700
	11	Marketing	201	1800
	13	Resource	203	2400
	14	Shipping	121	1500
*	NULL	NULL	NULL	NULL

- Inner join



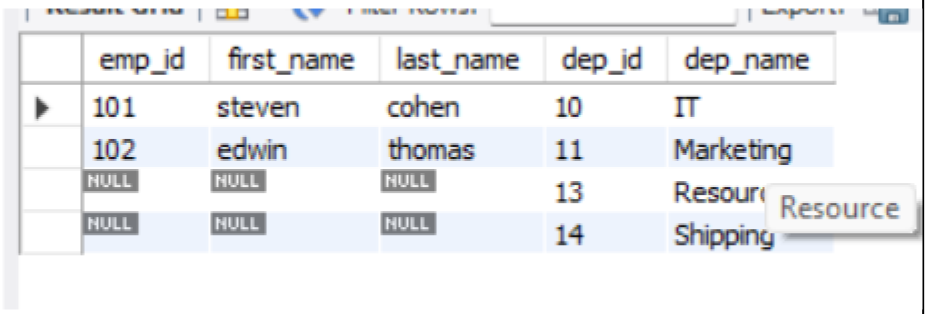
	emp_id	first_name	last_name	dep_id	dep_name
▶	101	steven	cohen	10	IT
	102	edwin	thomas	11	Marketing

- Left join



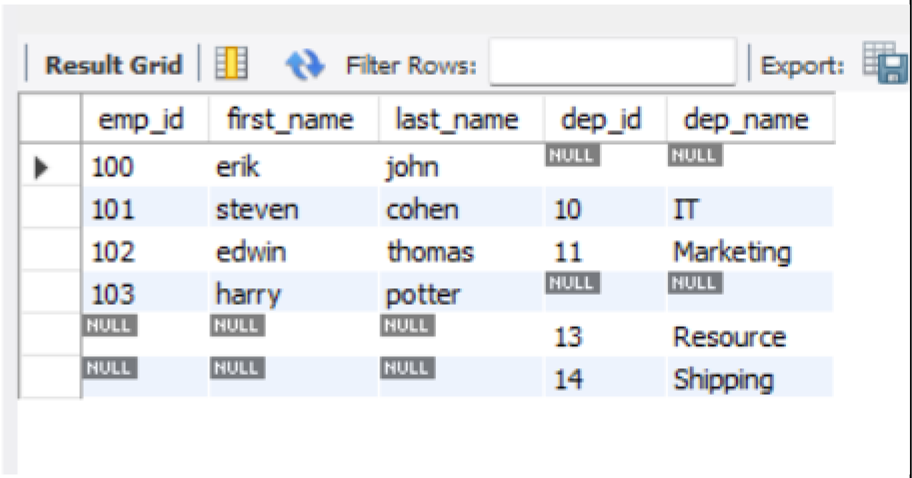
	emp_id	first_name	last_name	dep_id	dep_name
▶	100	erik	john	NULL	NULL
	101	steven	cohen	10	IT
	102	edwin	thomas	11	Marketing
	103	harry	potter	NULL	NULL

- Right join



emp_id	first_name	last_name	dep_id	dep_name
101	steven	cohen	10	IT
102	edwin	thomas	11	Marketing
NULL	NULL	NULL	13	Resource
NULL	NULL	NULL	14	Shipping

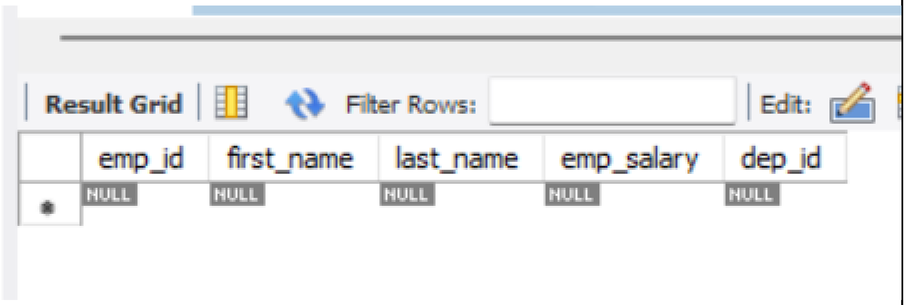
- Full outer join



Result Grid | Filter Rows: | Export:

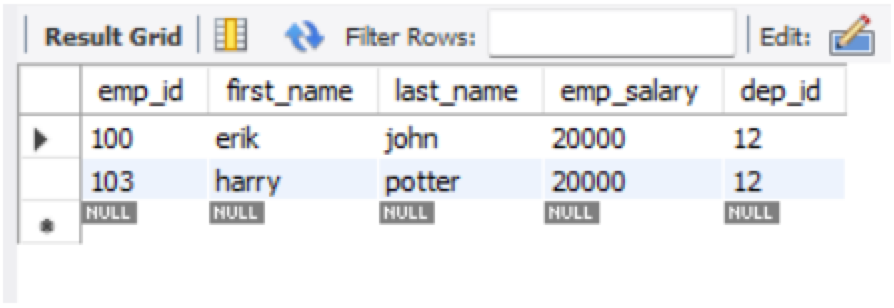
emp_id	first_name	last_name	dep_id	dep_name
100	erik	john	NULL	NULL
101	steven	cohen	10	IT
102	edwin	thomas	11	Marketing
103	harry	potter	NULL	NULL
NULL	NULL	NULL	13	Resource
NULL	NULL	NULL	14	Shipping

- Sql Sub Queries :



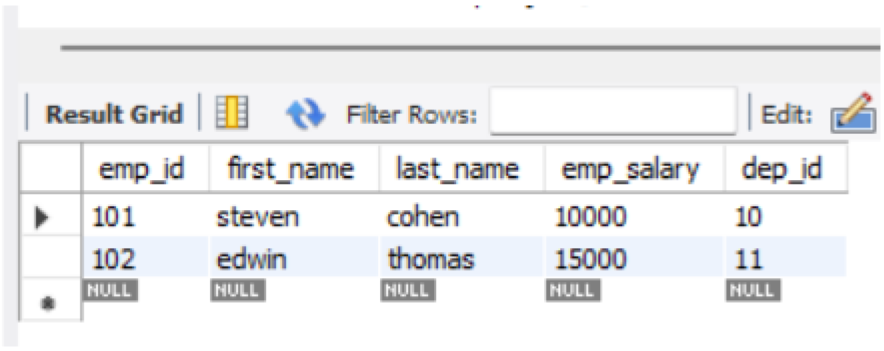
Result Grid | Filter Rows: | Edit:

emp_id	first_name	last_name	emp_salary	dep_id
NULL	NULL	NULL	NULL	NULL

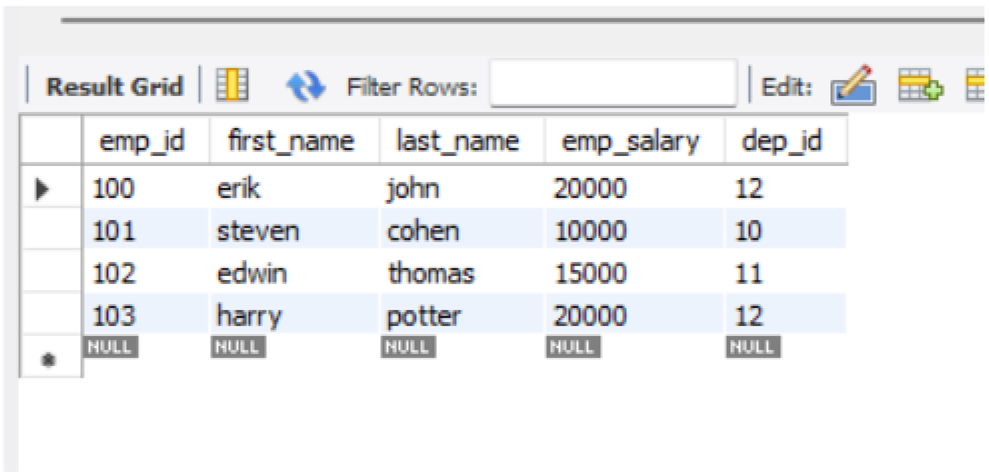


Result Grid | Filter Rows: | Edit:

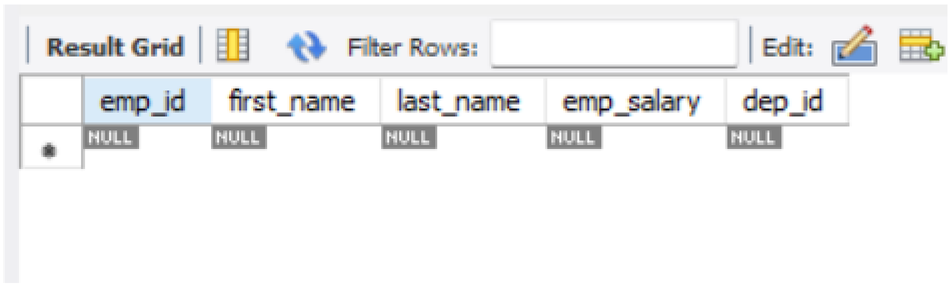
emp_id	first_name	last_name	emp_salary	dep_id
100	erik	john	20000	12
103	harry	potter	20000	12
NULL	NULL	NULL	NULL	NULL



	emp_id	first_name	last_name	emp_salary	dep_id
▶	101	steven	cohen	10000	10
	102	edwin	thomas	15000	11
*	NULL	NULL	NULL	NULL	NULL



	emp_id	first_name	last_name	emp_salary	dep_id
▶	100	erik	john	20000	12
	101	steven	cohen	10000	10
	102	edwin	thomas	15000	11
	103	harry	potter	20000	12
*	NULL	NULL	NULL	NULL	NULL



	emp_id	first_name	last_name	emp_salary	dep_id
*	NULL	NULL	NULL	NULL	NULL

Conclusion:

In this experiment, understand and be able to do the following: Describe the types of problems that subqueries can solve.
Sub queries are nested within a SELECT, INSERT, UPDATE, or DELETEstatement.
A subquery can be used inside the WHERE or HAVING clauses of the outer

	SELECT, INSERT, UPDATE, or DELETE statements. Build and execute sub query. Define and execute various types of joins.
References:	Lecture notes

Experiment No 7 Database Management System Lab 2021-22 Faculty: Sana Shaikh

Class: SE Comp

Don Bosco Institute of Technology

Department of Computer Engineering

Assessment Rubric for Experiment No. 7

Title of Experiment : Perform Sub Queries, Nested Queries and Joins. **Performance Date :**
Year and Semester : 2nd Year and IVth Semester **Submission Date : Name: Batch : Roll No. :**

	. Criteria	1 Marks	2 Marks	3 Marks 4 Marks	5 Marks
1	Execution	Executed 10-30% queries based on following: -Sub query - nested querying - Joins	Executed 31-50% queries based on following: -Sub query - nested querying - Joins	Executed 51-70% queries based on following: -Sub query - nested querying - Joins Executed 71-89% queries based on following: -Sub query - nested querying - Joins	Executed 90-100% queries based on following: -Sub query - nested querying - Joins

2 Documentation 20-39% of solutions are documented properly.	40-59% of solutions are documented properly.	60-79% of solutions are documented properly.	80-100% of the solution is documented properly.
3 Viva Students hardly answered.	Students have problems while answering.	Questions are answered fairly well.	Questions are answered completely and correctly.

4 Submission on
Time
Submitted after the
given
deadline
Submitted
before the
given deadline