

Subquery

Table:

Employee:

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer
104	Dorvin	Finance	6500
107	Preet	HR	7000
109	Sanjay	IT	6500
110	Vasudha	IT	7000
111	Melinda	IT	8000
112	Komal	IT	10000

Department:

dept_id integer	dept_name [PK] character varying (50)	location character varying (100)
2	HR	Bangalore
3	IT	Bangalore
4	Finance	Mumbai
5	Marketing	Bangalore
6	Sales	Mumbai

Sales:

store_id integer	store_name character varying (50)	product_name character varying (50)	quantity integer	price integer
1	Apple Store 1	iPhone 13 Pro	1	1000
1	Apple Store 1	MacBook pro 14	3	6000
1	Apple Store 1	AirPods Pro	2	500
2	Apple Store 2	iPhone 13 Pro	2	2000
3	Apple Store 3	iPhone 12 Pro	1	750
3	Apple Store 3	MacBook pro 14	1	2000

Non_Correlated Subquery

```
select * -- outer query / main query
from employee
where salary > (select avg(salary) from employee); -- subquery/ inner query
```

```
-- Scalar subquery
-- it always returns 1 row and 1 column.
select *
from employee e
join (select avg(salary) sal from employee) avg_sal
on e.salary > avg_sal.sal;
```

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer	sal numeric
119	Cory	HR	8000	5791.6666666666666667
121	Rosalin	IT	6000	5791.6666666666666667
122	Ibrahim	IT	8000	5791.6666666666666667

```
-- multiple row subquery
-- subquery which returns multiple column and multiple row
-- subquery which returns only 1 column and multiple rows.

/* QUESTION: Find the employees who earn the highest salary in each department. */
```

```
select dept_name, max(salary)
from employee
group by dept_name
```

```
select *
from employee
where (dept_name, salary) in (select dept_name, max(salary)
                             from employee
                             group by dept_name)
```

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer
104	Dorvin	Finance	6500
116	Satya	Finance	6500
119	Cory	HR	8000
120	Monica	Admin	5000
124	Dheeraj	IT	11000

```
-- single column, multiple row subquery
/* QUESTION: Find department who do not have any employees */

select *
from department
where dept_name not in (select distinct dept_name from employee);
```

dept_id integer	dept_name [PK] character varying (50)	location character varying (100)
5	Marketing	Bangalore
6	Sales	Mumbai

Correlated Subquery

```
-- Correlated subquery
-- A subquery which is related to the outer query.
```

Find the employees in each department who earn more than the average salary in that department

```
select *
from employee e1
where salary > (select avg(salary)
                from employee e2
                where e2.dept_name = e1.dept_name
                )
```

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer
101	Mohan	Admin	4000
102	Rajkumar	HR	3000
103	Akbar	IT	4000
104	Dorvin	Finance	6500
105	Rohit	HR	3000
106	Rajesh	Finance	5000

/* QUESTION: Find department who do not have any employees */

```
select *
from department d
where not exists (select 1 from employee e where e.dept_name = d.dept_name);
```

dept_id integer	dept_name [PK] character varying (50)	location character varying (100)
5	Marketing	Bangalore
6	Sales	Mumbai

What is about?

Select *

From employee e

Where exists (select * from department d where e.dept_name=d. dept_name)

Output?

Select *

From department d

Where exists (select * employee e from where e.dept_name=d. dept_name)

Output?

More Complicated Subquery

-- Subquery inside a Subquery

/* QUESTION: Find stores who's sales where better than the average sales accross all stores */

- 1) find the total sales for each store.
- 2) find avg sales for all the stores.
- 3) compare 1 & 2

Step1:

```
select store_name, sum(price) as total_sales
from sales
group by store_name
```

store_name	total_sales
character varying (50)	bigint
Apple Store 3	9700
Apple Store 2	2000
Apple Store 1	7500
Apple Store 4	5000

Step 2:

```
select avg(total_sales)
from (select store_name, sum(price) as total_sales
      from sales
      group by store_name) x
```

avg
numeric
6050.00000000000000000000

Step 3:

```
select *  
from (select store_name, sum(price) as total_sales  
      from sales  
      group by store_name) sales  
join (select avg(total_sales) as sales  
      from (select store_name, sum(price) as total_sales  
            from sales  
            group by store_name) x) avg_sales  
on sales.total_sales > avg_sales.sales;
```

store_name character varying (50)	total_sales bigint	sales numeric
Apple Store 3	9700	6050.000000000000000000
Apple Store 1	7500	6050.000000000000000000

Different SQL Clause Where subquery is allowed

Four Clauses

- SELECT
- FROM
- WHERE
- HAVING

Having:

```
-- HAVING
/* QUESTION: Find the stores who have sold more units than the average units sold by all stores.

select store_name, sum(quantity)
from sales
group by store_name
having sum(quantity) > (select avg(quantity) from sales);
```

store_name character varying (50)	sum bigint
Apple Store 3	11
Apple Store 1	6
Apple Store 4	3

Select: Do yourself

SQL Commands Which allow subquery

- SQL Query
- INSERT
- UPDATE
- DELETE

SQL Query we have done

Insert:

```
-- INSERT
/* QUESTION: Insert data to employee history table. Make sure not insert duplicate records. */
select * from employee_history;
```

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer	location character varying (100)

```
insert into employee_history
select e.emp_id, e.emp_name, d.dept_name, e.salary, d.location
from employee e
join department d on d.dept_name = e.dept_name
where not exists (select 1
                  from employee_history eh
                  where eh.emp_id = e.emp_id);
```

emp_id [PK] integer	emp_name character varying (50)	dept_name character varying (50)	salary integer	location character varying (100)
101	Mohan	Admin	4000	Bangalore
102	Rajkumar	HR	3000	Bangalore
103	Akbar	IT	4000	Bangalore
104	Dorvin	Finance	6500	Mumbai
105	Rohit	HR	3000	Bangalore
106	Rajesh	Finance	5000	Mumbai
107	Preet	HR	7000	Bangalore

Update:

```
-- UPDATE
/* QUESTION: Give 10% increment to all employees in Bangalore location based on the maximum salary earned by an emp in each dept. Only consider employees in employee_history table. */

update employee e
set salary = (select max(salary) + (max(salary) * 0.1)
              from employee_history eh
              where eh.dept_name = e.dept_name)
where e.dept_name in (select dept_name
                      from department
                      where location='Bangalore')
and e.emp_id in (select emp_id from employee_history);
```

Delete:

```
-- DELETE
/* QUESTION: Delete all departments who do not have any employees. */

delete from department
where dept_name in (select dept_name
                    from department d
                    where not exists (select 1 from employee e where e.dept_name = d.dept_name)
                    )
```