Subqueries and Nested Queries - Task 5

RDBMS used : MySql

A **subquery** is a query inside another query.

- **SUBQUERY in WHERE clause**

  i. Using "=" : only 1 row can be returned from the subquery

```
select * from products where
prod_id = (select prod_id from orders where quantity=1);
```

prod_id	prod_name	category	price	color	size	gender
1003	boots	semi-formal	3000	black	5	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

  ii. Using "in" operator : multiple rows can be returned from the subquery

```
select * from products where prod_id in (select prod_id from orders where quantity>2);
```

prod_id	prod_name	category	price	color	size	gender
1001	sneakers	casual	2500	dark brown	6	1
1002	loafers	formal	3500	deep black	7	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

  iii. Using other operators

```
select * from products where price > (select avg(price) from products);
```

prod_id	prod_name	category	price	color	size	gender
1001	sneakers	casual	2500	dark brown	6	1
1002	loafers	formal	3500	deep black	7	1
1003	boots	semi-formal	3000	black	5	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select * from products where prod_id not in (select prod_id from orders);
```

prod_id	prod_name	category	price	color	size	gender
1004	flip-flops	casual	500	blue	5	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **SUBQUERY in SELECT statement**

```
select prod_id, prod_name,
(select order_status from orders where quantity=1) as CURRENT_STATUS
from products;
```

prod_id	prod_name	CURRENT_STATUS
1001	sneakers	shipped
1002	loafers	shipped
1003	boots	shipped
1004	flip-flops	shipped

➢ **SUBQUERY in FROM clause**

select prod_name from (select * from products) as PNAME;

prod_name
sneakers
loafers
boots
flip-flops

select p.prod_name, p.price from products p,
(select avg(price) as avg_p from products) as avg_table
where p.price>avg_table.avg_p;

prod_name	price
sneakers	2500
loafers	3500
boots	3000

➢ **NOTES**

- A subquery that returns only **one value** (one row, one column) can be used in a SELECT, WHERE, or HAVING clause.
- A subquery that returns **multiple rows** can be used with IN, ANY, or EXISTS.
- The **subquery (inside FROM)** creates a **temporary table (avg_table)** with a single value: the average price.
- Difference between non-correlated and correlated subquery:

NON-CORRELATED	CORRELATED
• Independent of the outer query	• Uses values from the outer query
• Runs only once	• Runs once for each row in the outer query
• Faster and efficient	• Slower and repetitive