SQL Task 5

# **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
٠	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
HULL	NULL	HULL	NULL	NULL	HULL	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	HULL	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 NULL	flip-flops	casual NULL	500 NULL	blue NULL	5 NULL	NULL

#### > SUBQUERY in SELECT statement

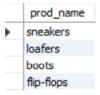
select prod\_id, prod\_name,

(select order\_status from orders where quantity=1) as CURRENT\_STATUS from products;

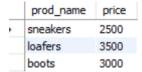
prod_id	prod_name	CURRENT_STATUS
1001	sneakers	shipped
1002	loafers	shipped
1003	boots	shipped
1004	flip-flops	shipped

## > SUBQUERY in FROM clause

select prod\_name from (select \* from products) as PNAME;



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;



#### > 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
<ul> <li>Independent of the outer query</li> </ul>	<ul> <li>Uses values from the outer</li> </ul>
	query
Runs only once	<ul> <li>Runs once for each row in the</li> </ul>
	outer query
<ul> <li>Faster and efficient</li> </ul>	<ul> <li>Slower and repetitive</li> </ul>