sub Query basic

A subquery is a query that is nested within another query. It is used to retrieve data that will be used as input for another query. A subquery can be used in different parts of a SQL statement, such as in the SELECT, WHERE, and FROM clauses.

A basic subquery can be created using the following syntax:

SELECT column1, column2

FROM table1

WHERE column1 = (SELECT column1 FROM table2 WHERE column2 = 'value');

**Subqueries can also be used in the SELECT and FROM clauses to perform calculations and join tables, respectively. They can be nested multiple times to create more complex queries.**

**Examples**

1. **Select 2022 orders from customers & orders table**

Suppose we have two tables: "customers" and "orders". The "customers" table contains information about each customer, including their customer ID, name, and email address. The "orders" table contains information about each order, including the order ID, the customer ID of the customer who placed the order, and the order date.

Suppose we want to retrieve the names and email addresses of all customers who have placed an order in the year 2022. We can use a subquery to accomplish this, like so:

--1 saqib Ali m4377535

--2 Noman A b43577535@gmail.com

--3 Billal Ali m4377535

--4 Sajid akram 4343@gmail.com

--5 saqib Ali m4377535

--6 Noman A b43577535@gmail.com

--7 Billal Ali m4377535

--8 Rakham akram 4343@gmail.com

--1 2022-05-20 1

--2 2023-01-20 2

--3 2022-11-19 1

--4 2020-04-20 3

SELECT f\_name, email

FROM customers

WHERE id IN ( SELECT customer\_id FROM orders WHERE YEAR(order\_date) = 2022 );

select \* from orders

In this query, the subquery is used to retrieve the customer IDs of all customers who placed an order in the year 2022.

**2. Adding Department Column using**

**SubQuery**

Suppose we have two tables: "employees" and "departments". The "employees" table contains information about each employee, including their employee ID, name, and department ID. The "departments" table contains information about each department, including the department ID and the department name.

Suppose we want to retrieve a list of all employees and their corresponding department names. We can use a subquery to accomplish this, like so:

create table department(

id int primary key,

name varchar(25)

)

insert into department values (1,'HR') ,(2,'Admin'),(3,'Gard'),(4,'Poem')

create table employee(

id int primary key,

name varchar(25),

departmentId int constraint cns\_departementId\_foreignKey foreign key references department(id)

)

insert into employee values (1,'haseeeb',1) ,(2,'Moeeen',2),(3,'Shaazad',1),(4,'Personer',2)

--\_\_\_\_\_\_\_\_\_\_ join \_\_\_\_\_\_\_\_\_\_\_

select

e.id,

e.name,

d.name

from employee e

left join department d on d.id = e.departmentId

--\_\_\_\_\_\_\_\_\_\_ join \_\_\_\_\_\_\_\_\_\_\_

--\_\_\_\_ 1st Way \_\_\_\_\_\_\_\_

SELECT

first\_name,

(SELECT department\_name FROM departments WHERE department\_id = employee.department\_id) AS department\_name

FROM employee;

select \* from employee

select \* from departments

--\_\_\_\_ 2nd Way \_\_\_

SELECT

first\_name,

department\_name

FROM (

select

first\_name,

(SELECT department\_name FROM departments WHERE department\_id = employee.department\_id) AS department\_name

from employee

) as emp;

**3. Retrieve list All out of Stock Product from inventory table product=0;**

Suppose we have a table called "inventory" that contains information about the quantity of each product in stock at a store, including the product ID and the quantity. Suppose we want to retrieve a list of all products that are out of stock (i.e., have a quantity of 0). We can use a subquery to accomplish this, like so:

sqlCopy code

select \* from product

--1 Mouse 250.00

--2 USB 250.00

--3 Keyboard 250.00

--4 Computer 250.00

create table inventory(

i\_id int primary key identity,

quentity int,

product\_id int foreign key references product(id)

)

insert into inventory values (5,1),(0,2),(4,3),(0,4);

select \* from inventory

--\_\_\_\_\_\_ 1st Way \_\_\_\_\_\_

SELECT id, pName

FROM product

WHERE id IN (SELECT product\_id FROM inventory WHERE quentity = 0);

--\_\_\_\_\_\_ 2nd Way \_\_\_\_\_\_

SELECT

id,

pName,

'OutOfStock' outofstock

FROM product

WHERE id IN (SELECT product\_id FROM inventory WHERE quentity = 0);

In this query, the subquery is used to retrieve the product IDs of all products that have a quantity of 0 in the "inventory" table. The main query then selects the product ID and product name fields from the "products" table for only those products whose product ID is found in the subquery results. The IN keyword is used to specify that we are looking for products whose ID is in the subquery results.

**4. Retrieve list Of Orders which has products Unit Price 100$.**

Suppose we have two tables: "orders" and "order\_details". The "orders" table contains information about each order, including the order ID, the customer ID of the customer who placed the order, and the order date. The "order\_details" table contains information about the products included in each order, including the order ID, the product ID, the quantity, and the unit price.

Suppose we want to retrieve a list of all orders that include a product with a unit price greater than $100. We can use a subquery to accomplish this, like so:

select \* from orderDetail

--1 400 5 1

--2 450 10 2

--3 40 5 2

--4 100 5 NULL

--5 100 9 NULL

select \* from orders

--1 2022-05-20 1

--2 2023-01-20 2

--3 2022-11-19 1

--4 2020-04-20 3

SELECT id, customer\_id, order\_date

FROM orders

WHERE id IN ( SELECT od\_id FROM orderdetail WHERE UnitPrice > 100 );

**5. Retrieve the total quantity sold for each product in the "sales" table.**

Suppose we have a table called "sales" that contains information about the sales made by a company, including the sales ID, the product ID, the sales date, and the quantity sold. Suppose we want to retrieve the total quantity sold for each product in the "sales" table. We can use a subquery to accomplish this, like so:

CREATE TABLE sales (

sales\_id INT PRIMARY KEY,

product\_id INT,

sales\_date DATE,

quantity\_sold INT

);

INSERT INTO sales (sales\_id, product\_id, sales\_date, quantity\_sold)

VALUES (1, 100, '2022-01-01', 10),

(2, 101, '2022-01-01', 5),

(3, 100, '2022-01-02', 8),

(4, 102, '2022-01-02', 12),

(5, 101, '2022-01-03', 7),

(6, 100, '2022-01-04', 15),

(7, 102, '2022-01-05', 20),

(8, 100, '2022-01-06', 5),

(9, 101, '2022-01-06', 10);

--\_\_\_\_\_\_\_\_\_\_\_\_\_ 1st Way \_\_\_\_\_\_\_\_\_\_\_\_\_\_

SELECT product\_id, SUM(quantity\_sold) AS total\_quantity\_sold

FROM sales

GROUP BY product\_id

--\_\_\_\_\_\_\_\_\_\_\_\_\_ 2ndWay \_\_\_\_\_\_\_\_\_\_\_\_\_\_

SELECT

product\_id,

(

SELECT SUM(quantity\_sold)

FROM sales

WHERE product\_id = p.product\_id

) AS total\_quantity\_sold

FROM sales p

GROUP BY product\_id;

**6.**

Suppose we have three tables: "customers", "orders", and "order\_details". The "customers" table contains information about each customer, including the customer ID and the customer name. The "orders" table contains information about each order, including the order ID, the customer ID of the customer who placed the order, and the order date. The "order\_details" table contains information about the products included in each order, including the order ID, the product ID, the quantity, and the unit price.

Suppose we want to retrieve a list of all customers who have made an order for a product with a unit price greater than $100. We can use a subquery to accomplish this, like so:

sqlCopy code

SELECT customer\_id, customer\_name FROM customers WHERE customer\_id IN ( SELECT customer\_id FROM orders WHERE order\_id IN ( SELECT order\_id FROM order\_details WHERE unit\_price > 100 ) );

**7.**

Suppose we have a table called **employees** that contains information about the employees of a company, including the employee ID, the department ID they work in, their salary, and their hire date. Suppose we want to retrieve the average salary for each department, as well as the total number of employees in each department who were hired before a certain date.

CREATE TABLE employeees (

employee\_id INT PRIMARY KEY,

department\_id INT,

salary INT,

hire\_date DATE

);

INSERT INTO employeees (employee\_id, department\_id, salary, hire\_date) VALUES

(1, 1, 50000, '2021-01-01'),

(2, 1, 60000, '2022-02-01'),

(3, 2, 55000, '2020-05-01'),

(4, 2, 65000, '2019-06-01'),

(5, 2, 70000, '2022-01-01'),

(6, 3, 45000, '2018-03-01'),

(7, 3, 55000, '2017-01-01'),

(8, 3, 60000, '2020-12-01'),

(9, 3, 65000, '2021-07-01');

--- id wise --------- hiring befor 2022 -- with Average Salary

SELECT

department\_id,

AVG(salary) AS avg\_salary,

(

SELECT COUNT(\*)

FROM employeees

WHERE department\_id = e.department\_id AND hire\_date < '2022-01-01'

) AS num\_employees\_hired\_before\_2022

FROM employeees e

GROUP BY department\_id;

--- id ,year wise --------- hiring befor 2022 -- with Average Salary

SELECT

department\_id,

AVG(salary) AS avg\_salary,

(

SELECT COUNT(\*)

FROM employeees

WHERE department\_id = e.department\_id AND hire\_date < '2022-01-01'

) AS num\_employees\_hired\_before\_2022,

Year(hire\_date) as year

FROM employeees e

GROUP BY department\_id,Year(hire\_date);