COURSE NAME: DBMS
COURSE CODE:23AD2102A

TOPIC:

COMPLEX QUERIES WITH SYNTAX(SUBQUERIES)

Session - 11

AIM OF THE SESSION

To familiarize students with the advance and complex Subqueries in PostgreSQL.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Discuss the subqueries.
2. Various guidelines and types of subqueries.

LEARNING OUTCOMES

At the end of this session, you should be able to understand the basic concepts of Subqueries and learn how to write complex subqueries with PostgreSQL commands.

SESSION INTRODUCTION

A **subquery** or **Inner query** or **Nested query** is a query within another PostgreSQL query and embedded within the WHERE clause.

- In PostgreSQL, subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL.
- A subquery can be treated as an inner query, which is a SQL query placed as a part of another query called as outer query.
- The inner query executes first before its parent query so that the results of an inner query can be passed to the outer query.

SUBQUERY SYNTAX

```
Select      select_list
From        table
Where       expr operator
                    ( Select    select_list
                      From      table );
```

- The subquery (inner query) executes once before the main query (outer query) executes.

- The main query (outer query) use the subquery result.

SUBQUERY EXAMPLE

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING@example.com	515.123.4567	1987-06-17	AD_PRES	24000.00	0.00	0	90
101	Neena	Kochhar	NKOCHHAR@example.com	515.123.4568	1987-06-18	AD_VP	17000.00	0.00	100	90
102	Lex	De Haan	LDEHAAN@example.com	515.123.4569	1987-06-19	AD_VP	17000.00	0.00	100	90
103	Alexander	Hunold	AHUNOLD@example.com	590.423.4567	1987-06-20	IT_PROG	9000.00	0.00	102	60
104	Bruce	Ernst	BERNST@example.com	590.423.4568	1987-06-21	IT_PROG	6000.00	0.00	103	60
105	David	Austin	DAUSTIN@example.com	590.423.4569	1987-06-22	IT_PROG	4800.00	0.00	103	60
106	Valli	Pataballa	VPATABAL@example.com	590.423.4560	1987-06-23	IT_PROG	4800.00	0.00	103	60
107	Diana	Lorentz	DLORENTZ@example.com	590.423.5567	1987-06-24	IT_PROG	4200.00	0.00	103	60
114	Den	Raphaely	DRAPHEAL@example.com	515.127.4561	1987-07-01	PU_MAN	11000.00	0.00	100	30
115	Alexander	Khoo	AKHOO@example.com	515.127.4562	1987-07-02	PU_CLERK	3100.00	0.00	114	30
116	Shelli	Baida	SBAIDA@example.com	515.127.4563	1987-07-03	PU_CLERK	2900.00	0.00	114	30
117	Sigal	Tobias	STOBIAS@example.com	515.127.4564	1987-07-04	PU_CLERK	2800.00	0.00	114	30
118	Guy	Himuro	GHIMURO@example.com	515.127.4565	1987-07-05	PU_CLERK	2600.00	0.00	114	30
119	Karen	Colmenares	KCOLMENA@example.com	515.127.4566	1987-07-06	PU_CLERK	2500.00	0.00	114	30
120	Matthew	Weiss	MWEISS@example.com	650.123.1234	1987-07-07	ST_MAN	8000.00	0.00	100	50
121	Adam	Fripp	AFRIPP@example.com	650.123.2234	1987-07-08	ST_MAN	8200.00	0.00	100	50

For example, List the name of the employees, paid more than 'Alexander' from employees.

SUBQUERY EXAMPLE

```
Select      first_name, last_name, salary
From        employees
Where       salary >
                    ( Select    salary
                    From        employees
                    Where       first_name='Alexander' );
```

9000

Sample Output:

first_name	last_name	salary
Steven	King	24000.00
Neena	Kochhar	17000.00
Lex	De Haan	17000.00
Nancy	Greenberg	12000.00
Den	Raphaely	11000.00
John	Russell	14000.00
Karen	Partners	13500.00
Alberto	Errazuriz	12000.00
Gerald	Cambrault	11000.00
Eleni	Zlotkey	10500.00
Peter	Tucker	10000.00
David	Bernstein	9500.00
Janette	King	10000.00

SUBQUERY GUIDELINES

There are some guidelines to consider when using subqueries:

- A subquery must be enclosed in parentheses.
- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.
- If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.
- If you're using a PostgreSQL Subquery with the SELECT clause, then you can have only one column along with it unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY clause cannot be used in a PostgreSQL Subquery, although the main query can use an ORDER BY.
- PostgreSQL Subqueries that return more than one row can only be used with multiple value operators, such as the IN, EXISTS, ALL, NOT IN, ANY, SOME, etc.

TYPES OF SUBQUERIES

Types of Subqueries are:

- Comparisons using Subqueries
- Subqueries with ALL, ANY, or, IN
- Subqueries with EXISTS
- Correlated Subqueries
- Subqueries with the INSERT Statement
- Subqueries with the UPDATE Statement
- Subqueries with the DELETE Statement

POSTGRESQL SUBQUERIES: USING COMPARISONS

A subquery can be used before or after any of the comparison operators. The subquery can return at most one value. The value can be the result of an arithmetic expression or a column function. SQL then compares the value that results from the subquery with the value on the other side of the comparison operator. You can use the following comparison operators:

Operator	Description
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal to
<>	Not equal to

For example, suppose you want to find the employee id, first_name, last_name, and salaries for employees whose average salary is higher than the average salary throughout the company.

```
Select    employee_id, first_name, last_name, salary
From      employees
Where     salary > 6461.682243
          ( Select   AVG ( salary )
            From     employees );
```

employee_id	first_name	last_name	salary
100	Steven	King	24000.00
101	Neena	Kochhar	17000.00
102	Lex	De Haan	17000.00
103	Alexander	Hunold	9000.00
108	Nancy	Greenberg	12000.00
109	Daniel	Faviet	9000.00
110	John	Chen	8200.00
111	Ismael	Sciarra	7700.00
112	Jose Manuel	Urman	7800.00
113	Luis	Popp	6900.00
.....	.....	.....	.....

IN operator is used to compare a column to a list of values. The right-hand side is a parenthesized subquery, which must return exactly one column. The left-hand expression is evaluated and compared to each row of the subquery result.

Example: The following query selects those employees who work in the location 1800. The subquery finds the department id in the 1800 location, and then the main query selects the employees who work in any of these departments.

```
SELECT first_name, last_name, department_id FROM employees WHERE
department_id
                    IN
(SELECT department_id FROM departments WHERE location_id=1800);
```

The **ALL** operator compares value to every value returned by the subquery. The right-hand side is a parenthesized subquery, which must return exactly one column. The left-hand expression is evaluated and compared to each row of the subquery result using the given operator. For example, the following query selects the department with the highest average salary. The subquery finds the average salary for each department, and then the main query selects the department with the highest average salary.

```
SELECT department_id, AVG(SALARY) FROM employees GROUP BY department_id HAVING AVG(SALARY) >= ALL (SELECT AVG(SALARY) FROM employees GROUP BY department_id);
```

```
 department_id |          avg
---------------+----------------------
            90 | 19333.333333333333
(1 row)
```

The **ANY** operator compares the value to each value returned by the subquery. Therefore ANY keyword (which must follow a comparison operator) returns TRUE if the comparison is TRUE for ANY of the values in the column that the subquery returns.

The following query selects any employee who works in the location 1700. The subquery finds the department id in the 1700 location, and then the main query selects the employees who work in any of these departments.

```
SELECT first_name, last_name, department_id FROM employees WHERE department_id = ANY (SELECT DEPARTMENT_ID FROM departments WHERE location_id=1700);
```

```
first_name   | last_name   | department_id
-------------+-------------+---------------
Steven       | King        |            90
Neena        | Kochhar     |            90
Lex          | De Haan     |            90
Nancy        | Greenberg   |           100
Daniel       | Faviet      |           100
John         | Chen        |           100
Ismael       | Sciarra     |           100
Jose Manuel  | Urman       |           100
Luis         | Popp        |           100
Den          | Raphaely    |            30
Alexander    | Khoo        |            30
Shelli       | Baida       |            30
Sigal        | Tobias      |            30
Guy          | Himuro      |            30
Karen        | Colmenares  |            30
Jennifer     | Whalen      |            10
Shelley      | Higgins     |           110
William      | Gietz       |           110
(18 rows)
```

POSTGRE SUBQUERIES WITH EXISTS

The argument of **EXISTS** is an arbitrary SELECT statement, or subquery. The subquery is evaluated to determine whether it returns any rows. If it returns at least one row, the result of EXISTS is true; if the subquery returns no rows, the result of EXISTS is false. For Example: The following query finds employees (employee_id, first_name, last_name, job_id, department_id) from employees table who have at least one person reporting to them.

```
SELECT employee_id, first_name, last_name, job_id, department_id FROM employees E WHERE EXISTS (SELECT * FROM employees WHERE manager_id = E.employee_id);
```

employee_id	first_name	last_name	job_id	department_id
100	Steven	King	AD_PRES	90
101	Neena	Kochhar	AD_VP	90
102	Lex	De Haan	AD_VP	90
103	Alexander	Hunold	IT_PROG	60
108	Nancy	Greenberg	FI_MGR	100
114	Den	Raphaely	PU_MAN	30
120	Matthew	Weiss	ST_MAN	50
121	Adam	Fripp	ST_MAN	50
122	Payam	Kaufling	ST_MAN	50
123	Shanta	Vollman	ST_MAN	50

.....

CORRELATED SUBQUERIES

A correlated subquery is a subquery that contains a reference to a table (in the parent query) that also appears in the outer query. PostgreSQL evaluates from inside to outside. For example, following correlated subquery find all employees who earn more than the average salary in their department.

```
SELECT last_name, salary, department_id  FROM employees outer WHERE salary > (SELECT AVG(salary)  FROM employees  WHERE department_id = outer.department_id);
```

```
last_name |   salary   | department_id
-----------+------------+---------------
King       | 24000.00 |            90
Hunold     |  9000.00 |            60
Ernst      |  6000.00 |            60
Greenberg  | 12000.00 |           100
Faviet     |  9000.00 |           100
Raphaely   | 11000.00 |            30
Weiss      |  8000.00 |            50
Fripp      |  8200.00 |            50
Kaufling   |  7900.00 |            50
Vollman    |  6500.00 |            50
Mourgos    |  5800.00 |            50
.....................
```

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. For example, to insert all records into ‘employee1' table from ‘employee' table, the following SQL statement can be used:

```
INSERT INTO employee1 SELECT * FROM employee;
```

We can insert records using subqueries with where clause also. For example, to insert those records into ‘employee1' table from ‘employee' table who gets more than 5000 salary, the following SQL statement can be used:

```
INSERT INTO employee1 SELECT * FROM employee WHERE salary>5000;
```

SUBQUERIES WITH THE UPDATE STATEMENT

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement. The following example updates SALARY by 0.50 times in the employee table for all the employees, who work in the department ‘Accounts’.

```
UPDATE  employee SET salary = salary * 0.5 where department_id
                  IN
(SELECT  department_id FROM departments WHERE department_name=‘Accounts’);
```

The subquery can also be used with the DELETE statement. Either single or multiple columns in a table can be deleted when using a subquery with the DELETE statement. The following example deletes all employees except those employees, who work in the department 'HR'.

```
DELETE FROM employee where department_id
                NOT IN
(SELECT department_id FROM departments WHERE department_name='HR');
```

IMPORTANT FACTS RELATED TO THE SESSION

- A **subquery** or **Inner query** or **Nested query** is a query within another PostgreSQL query and embedded within the WHERE clause.
- Subqueries are generally used with the SELECT statement but these queries can also be used with the INSERT Statement, UPDATE Statement, and the DELETE Statement.
- We can write Subqueries with ALL, ANY, or, IN operators.

SUMMARY

In this section, we discussed the complex sql queries, general guidelines for creating these complex subqueries.

We also discussed the different types of subqueries in detail with its syntax and examples.

SELF-ASSESSMENT QUESTIONS

1. Which of the following is true about sub-queries?

a) They execute after the main query executes.
b) They execute in parallel to the main query.
c) The user can execute the main query and then, if wanted, execute the sub-query.
d) They execute before the main query executes.

2. Which of the following clause is mandatorily used in a sub-query?

(a) SELECT
(b) WHERE
(c) ORDER BY
(d) GROUP BY

SELF-ASSESSMENT QUESTIONS

3. Which of the following multi-row operators can be used with a sub-query?

(a) IN
(b) ANY
(c) ALL
(d) ALL OF THE ABOVE

4. Which of the following is true about the result of a sub-query?

a) The result of a sub-query is generally ignored when executed.
b) The result of a sub-query doesn't give a result, it is just helpful in speeding up the main query execution.
c) The result of a sub-query is used by the main query.
d) The result of a sub-query is always NULL.

TERMINAL QUESTIONS

1. Describe various types of SQL complex subqueries.

2. List out the guidelines for creating the SQL subqueries.

3. Analyze the use of ALL,IN, or ANY operator while using subqueries in PostgreSQL.

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. Database System Concepts, Sixth Edition, Abraham Silberschatz, Yale University Henry, F. Korth Lehigh University, S. Sudarshan Indian Institute of Technology, Bombay.

2. An Introduction to Database Systems by Bipin C. Desai

3. Fundamentals of Database Systems, 7th Edition, RamezElmasri, University of Texas at Arlington, Shamkant B. Navathe, University of Texasat Arlington.

Sites and Web links:

1. https://www.geeksforgeeks.org/postgresql-create-table/

2. https://www.tutorialsteacher.com/postgresql

THANK YOU

Team – DBMS