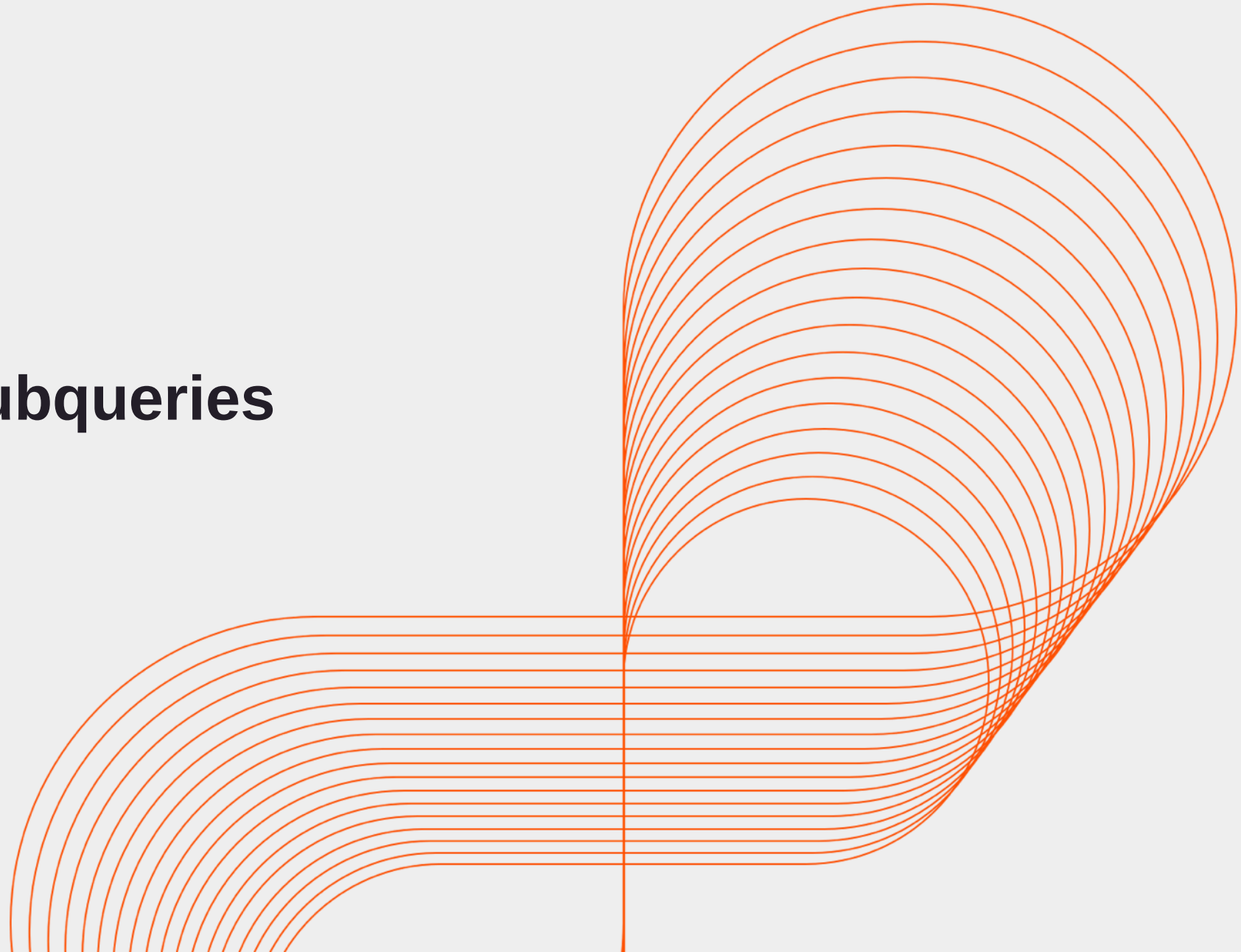
Persistent

NUGGET 7: Subqueries

Persistent University



Key Learning Points

1. **Nested Subqueries**
2. **Correlated Subqueries**
3. **DML using Subqueries**

Sample Data

Table Name : Employee

EmployeeId	FirseName	LastName	Email	PhoneNumber	HireDate	JobId	Salary	CommitionPct	ManagerId	DepartmentId
1	John	Demn	JohnD@yahoo.com	9898780979	1/10/2001	IT_PROF	70000	0.5	NULL	10
2	Ken	Dale	kendaleD@gmail.com	7877787655	4/1/2001	SALES_HEAD	50000	NULL	NULL	10
3	James	Walton	JW@yahoo.com	5787887888	1/1/2001	IT_REP	30000	0.2	1	20
4	robin	sngal	robin@gmail.com	4990988839	5/1/2001	SALES_REP	40000	0.3	2	20
5	ajay	ghosala	ghosala@hotmail.com	9809888898	6/10/2002	SALES_REP	30000	0.4	2	20
6	John	Reddies	John@gmail.com	6878900989	6/10/2003	M_per	50000	NULL	NULL	NULL

Table Name : Department

DEPARTMENTID	DEPARTMENTNAME	MANAGERID	LOCATIONID
10	Sales	1	1
20	IT	2	2
30	Marketing	(null)	1

Table Name : Location

LOCATIONID	CITY
1	Pune
2	Mumbai

Table Name : Job_History

EMPLOYEEID	START_DATE	END_DATE	JOBID	DEPARTMENTID
1	10-Jan-01	31-Dec-01	IT_PROF	10
1	1-Jan-02	31-Aug-05	IT_REP	10
2	1-Apr-01	31-Jan-02	SALES_REP	20
2	1-Feb-02	31-Jan-05	SALES_PERS	10

Subquery

- A Subquery is a SELECT statement that is embedded in a clause of another SQL statement. Subqueries can be very useful when user need to select rows from a table with condition that depends on data in a table itself.
- Syntax:

```
Select select_list
From table
Where expr Operator (Select select_list
                     From Table);
```

- There are 2 types of Subqueries; namely:
  - Nested Subquery
  - Correlated Subquery

Guidelines for using Subqueries

- A subquery must be enclosed in Parenthesis.
- Place subqueries on the right side of comparison operator.
- The order by clause in Subqueries is not needed unless you are performing Top N analysis.
- Subquery can be used in number of SQL clauses, including
  - Where clause
  - Having clause
  - From Clause
- Subqueries can also be used in
  - CREATE VIEW statement
  - CREATE TABLE statement
  - SET and WHERE clause of UPDATE statement
  - INTO clause of INSERT statement
  - WHERE clause of DELETE statement,

Nested Subquery

- Nested Subqueries are further divided into 2 types
  - Single Row Subquery: When inner query returns a single row/value at a time.
  - Multiple Row Subquery: When inner query returns multiple rows at a time.
- Execution flow:

  In case of Nested subquery
  - The inner query executed first and finds a value.
  - Then the outer query executes using the value from the inner query.
- Single row Subquery uses the single row comparison operators : =,<>,>,>=,<,<=
- Multiple row Subquery used multiple row comparison operators : IN, ANY, ALL

Why Subquery?

Find all such employees whose salary is greater than Robin’s salary.

- First query to find salary of robin.

```
Select firstname, employeeid, salary from employee
where firstname='robin’;
```

- Second query to find all employees whose salary is greater than that of robin.

```
Select first_name, employee_id, salary from employee
where salary> 40000;                    -- Robin's salary is 40000.
```

- Use of subquery to combine above 2 queries in single query.

```
Select firstname, employeeid, salary
from employee
where salary> (select salary from employee
where firstname='robin');
```

FIRSTNAME	EMPLOYEEID	SALARY
John	1	70000
Ken	2	50000
John	6	50000

Single Row Subquery

- Queries that **return only one row** from the inner SELECT statement.
- Uses single row comparison operators : = ,> ,< ,>= ,<= ,<>
- Examples:
  - Display all employees whose job id is same as that of employee id 4

```
Select employeeid, lastname, jobid from employee
 Where jobid=(select jobid from employee
                Where employeeid=4);
```

EMPLOYEEID	LASTNAME	JOBID
4	sngal	SALES_REP
5	ghosala	SALES_REP

  - Display all employees whose job id is same as that of employee id 4 and salary is greater than salary of employee id 3.

```
Select employeeid, lastname, jobid, salary from employee
where jobid=(select jobid from employee Where employeeid=4)
and salary  > (select salary from employee Where employeeid=3);
```

EMPLOYEEID	LASTNAME	JOBID	SALARY
4	sngal	SALES_REP	40000

Single Row Subquery

- Using **group functions** in subquery

```
Select lastname, salary, jobid
from employee
where salary=(select min(salary)
                    From employee);
```

LASTNAME	SALARY	JOBID
Walton	30000	IT_REP
ghosala	30000	SALES_REP

- Subquery in **having clause**

```
Select departmentid, max(salary)
from employee
where departmentid is not null
group by departmentid
having max(salary)< (select max(salary) FROM employee
                          Where departmentid=10);
```

DEPARTMENTID	MAX(SALARY)
20	50000

Multiple Row Subquery

- Queries that **return more than one row** from the inner SELECT statement.
- Uses multiple row comparison operators:
  - IN :- Equal to any member in the list
  - ANY :- Compare value to each value returned by the subquery.
  - ALL :- Compare value to every value returned by the subquery.

  Note: < ANY means less than Maximum. > ANY means more than Minimum.

  < ALL means less than Minimum. > ALL means more than Maximum.
- Examples:
  - Find the employees who earn same salary as minimum salary in each department.

```
Select lastname, salary, departmentid from employee
where salary in(
                Select min(salary) from employee
                 where departmentid is not null
                 group by departmentid);
```

LASTNAME	SALARY	DEPARTMENTID
Walton	30000	10
ghosala	30000	20

Multiple Row Subquery

- Display employees whose job id is not IT representative and whose salary is greater than that of any sales representative. -- Use of ANY Operator

```
Select lastname, jobid, salary from employee
where lower(jobid)<> ‘it_rep'
and salary  > ANY(     Select salary from employee
                       Where lower(jobid)='sales_rep');
```

LASTNAME	JOBID	SALARY
Demn	IT_PROF	70000
Dale	SALES_HEAD	50000
Reddies	M_per	50000
sngal	SALES_REP	40000

- Display employees whose job id is not IT representative and whose salary is greater than all sales representatives.

-- Use of ALL Operator

```
Select lastname, jobid, salary from employee
where lower(jobid)<> 'it_rep'
and salary > ALL(      Select salary from employee
                       where lower(jobid)='sales_rep');
```

LASTNAME	JOBID	SALARY
Dale	SALES_HEAD	50000
Reddies	M_per	50000
Demn	IT_PROF	70000

Multiple Column Subquery

- Queries that **return more than one column** from the inner SELECT statement. It can return one row or multiple rows.
- Syntax:

```
Select col1, col2…
From table
Where (col1, col2,…) IN (Select col2, col3,….
                         from table Where condition );
```

- Example:
  - Display details of employees who are managed by same manager and work in the same department as employee id 5.

```
Select employeeid, managerid, departmentid from employee
where (managerid, departmentid) in
        (select managerid, departmentid
        from employee
        Where employeeid=5)
and employeeid <> 5;
```

EMPLOYEEID	MANAGERID	DEPARTMENTID
4	2	20

Null values in Subquery

- Whenever null values are likely to be part of result set of subquery, do not use the NOT IN operator. NOT IN operator is equivalent to <> ALL . The reason is that all **conditions that compare NULL value, result in NULL.**
- Note that, If you are using IN operator, then null value as a part of result set of subquery is not a problem. IN is equivalent to =ANY
- Example:

To find Employees who are not managers:

```
Select employeeid, lastname from employee
where employeeid not in  (select managerid
                          from employee);
```

Output: 'No Rows returned'. As the Manager id column contains null values for some employees

To get the required above output, we can write a query like below:

```
Select employeeid, lastname from employee
where employeeid not in   (select managerid from employee
                           Where managerid is not null);
```

EMPLOYEEID	LASTNAME
6	Reddies
5	ghosala
4	sngal
3	Walton

Using Subquery in the From clause

- User can use Subquery in FROM clause of SELECT statement.
- A Subquery in FROM clause is called as **INLINE VIEW.**
- It defines data source for that particular SELECT statement, and only that SELECT statement.
- Example:

  Display employee last names, salaries, dept numbers and average salaries for all employees who earn more than average salary in their department

```
Select a.lastname, a.salary, a.departmentid, b.avgsal
From employee a, (select departmentid, avg(salary) avgsal
                  From employee
                  group by departmentid) b
Where a.departmentid=b.departmentid
and a.salary> b.avgsal;
```

LASTNAME	SALARY	DEPARTMENTID	AVGSAL
Demn	70000	10	50000
Dale	50000	20	40000

Note: This is a also a good example of combining detailed row data and aggregate data in the same output.

Correlated subquery

- Oracle server performs a correlated subquery when the subquery **references a column** from the table referred **in outer(parent) statement**.
- A correlated subquery is evaluated once for each row processed by parent statement.
- User can use IN, ANY and ALL operators in correlated subquery.
- The parent statement can be SELECT, UPDATE or DELETE statement.
- It defines data source for that particular SELECT statement, and only that SELECT statement.
- Correlated Subquery Execution:
  - Get a candidate row fetched by the outer query. (Outer query executes first)
  - Execute the inner query using the value of the candidate row. (Inner query executes for each row returned by outer query)
  - Use the values resulting from the inner query to qualify or disqualify the candidate row.
  - Repeat until no candidate row remains.

Correlated subquery

- Correlated SubQuery Syntax:

  Select col1, col2,…

  From **table 1 “outer”**

  Where col1 operator (Select col1

  From table2

  Where col2 = **outer.col2**)

- Example: Find all employees who earn more than the average salary in their department.

  Select lastname, salary, departmentid

  from employee **outer**

  where salary > (select avg(salary) from employee

  Where departmentid= **outer.departmentid**);

LASTNAME	SALARY	DEPARTMENTID
Demn	70000	10
Dale	50000	20

Correlated subquery

- Example: Display details of those employees who have switched their job at least twice.

```
Select e.employeeid, lastname, e.jobid

from employee e

Where 2 <= (select count(1) FROM job_history

        Where employeeid=e.employeeid);
```

EMPLOYEEID	LASTNAME	JOBID
1	Demn	IT_PROF
2	Dale	SALES_HEAD

Correlated subquery

- EXISTS operator is basically used with correlated subqueries to test whether a value returned by the outer query exists in the result set of values returned by inner query
- In case of EXISTS, if the subquery returns atleast one matching row
  - The search does not continue in inner query
  - The condition is flagged true
- In case of EXISTS, if the row returned by Subquery does not match
  - The search condition is flagged false.
  - The search continues in inner query
- Accordingly, NOT EXISTS operator tests whether a value retrieved by outer query is not a part of the result set of the values returned by inner query.

Correlated subquery

- EXAMPLE: Find employees who have at least one person reporting to them.

```
Select employeeid, lastname, jobid, departmentid
from employee outer
where EXISTS(select 1 from employee
        Where managerid= outer.employeeid);
```

EMPLOYEEID	LASTNAME	JOBID	DEPARTMENTID
1	Demn	IT_PROF	10
2	Dale	SALES_HEAD	20

- EXAMPLE: Find Departments that do not have any employees.

```
Select departmentid, departmentname
from department d
where NOT EXISTS(select 1 from  employee
        Where departmentid= d.departmentid);
```

DEPARTMENTID	DEPARTMENTNAME
30	Marketing

Subquery with INSERT Statement

- Subquery in INSERT statement is used to copy Rows from one table into another table.
  - Syntax: Insert into table[(column1 [,column2…])]

    Subquery;

    -- Here, Subquery is the Select query that returns rows to insert the data in the table.
  - Replace subquery in place of VALUES clause.
  - Match the number of columns in the INSERT clause and their data types with number of values and their data types in subquery.
  - Example: Insert into emp_history(employeeid, firstname, lastname, email, phonenumber)

    select employeeid, firstname, lastname, email, phonenumber from employee;

Subquery with INSERT Statement

- User can also use Subquery in INTO clause of INSERT statement which will be used to identify table for the INSERT statement.
  - Syntax: Insert into (select col1, col2 from tab1 where condition) values(val1, val2);
  - The select list of this Subquery must have the same number of columns as the column list of values clause.
  - WITH CHECK OPTION keyword in Subquery prohibits user from modifying rows that are not in subquery.
  - Specify WITH CHECK OPTION to indicate that, if subquery is used in place of table in an INSERT statement, then no changes that would produce rows that are not included in the subquery are permitted.
  - Example: INSERT into (select employeeid, lastname, email, hiredate,salary From employee

    Where departmentid=50 WITH CHECK OPTION)

    Values(8883, ‘smith’, ‘sh@hotmail.com’, ‘01-DEC-1997’, 20000);

Note: Above query will result in below error : ORA-01402: view WITH CHECK OPTION where-clause violation. In this example, the subquery in the INTO clause considers all the employees in department id 50.

Department id is not in the select list of the subquery and thus value is not provided for department id, so it will be considered as NULL while inserting. So, Inserting null department id will violate the WITH CHECK OPTION condition.

Subquery in UPDATE Statement

- Subquery can be used in the SET and WHERE clause of the UPDATE statement to update the rows in the table based on the values from same or another table.

  - Syntax:

```
Update table
Set column=(select column from table where condition)
[, column=(select column from table where condition)]
[Where condition];
```

  - Example:

```
Update employee
set departmentid=(select departmentid from employee where employeeid=1),
jobid=(select jobid from employee where employeeid=2)
Where employeeid=3;
```

Subquery in DELETE Statement

- DELETE statement is used to remove rows from the table. User can use subquery in the Where condition of a SELECT statement to remove the rows based on values from another table.

  - Syntax:

```
DELETE  from Table
Where Columnname=(subquery)
```

  - Example:

```
Delete from employee
Where departmentid = (select departmentid from department
                      where lower(departmentname) like '%sales%');
```

Session 7: Summary

With this we have come to an end of our Seventh session where we discussed about meaning and use of Subquery, types of subqueries, Subqueries with DML statements etc.

- At the end of Nugget 7, we see that you are now able to answer following questions:
  - What do mean by Subquery? Explain with syntax.
  - What are the different types of Subqueries?
  - Explain “Correlated Subquery in detail”.
  - Explain “Use of Subqueries in DML statements”.

Reference Material: Sites

http://www.akadia.com/services/sqlsrv_subqueries.html

http://www.tutorialspoint.com/sql/sql-sub-queries.htm

http://beginner-sql-tutorial.com/sql-subquery.htm

http://en.wikipedia.org/wiki/Correlated_subquery

http://www.zentut.com/sql-tutorial/understanding-correlated-subquery/

http://www.comp.nus.edu.sg/~ooibc/courses/sql/dml_query_subquery.htm

Persistent

Thank you!

Persistent University