CSE – 302
Database Management System Sessional

SUBQUERIES

SUBQUERIES

- You want a list of all Employees who have higher Salary than that of the Employee Name “E_H”.

- To solve this problem, you need two queries:
  - One to find the Salary of E_H
  - And second to find who earns more than that amount.

SUBQUERIES (CONTD.)

- You can solve the problem by combining the two queries, placing one query inside the other query.

- The inner query or the subquery returns a value that is used by the outer query or main query.
  - The subquery executes once before the main query.
  - The result of the subquery is used by the main query.

SUBQUERIES (CONTD.)

- You want a list of all of the Employees that have a higher Salary than that of the Employee Name “E_H”.

(SELECT SALARY FROM EMPLOYEE

WHERE EMPLOYEE_NAME = 'E_H');

SUBQUERIES (CONTD.)

- You want a list of all of the Employees that have a higher Salary than that of the Employee Name “E_H”.

```
SELECT EMPLOYEE_NAME FROM EMPLOYEE
WHERE SALARY >
(SELECT SALARY FROM EMPLOYEE
WHERE EMPLOYEE_NAME = 'E_H');
```

SUBQUERIES (CONTD.)

We can place the Subquery in a number of SQL Clauses:

- WHERE Clause
- HAVING Clause
- FROM Clause

GUIDELINES FOR USING A SUBQUERY

- Enclose subqueries in Parentheses
- Place subqueries on the right side of the comparison operator
- Don’t add an ORDER BY clause to a subquery
- Use **Single row operators** with **single row subqueries**
- Use multiple row operators with multiple row subqueries

TYPES OF SUBQUERIES

- **Single Row Subquery**: Queries that return only one row from the inner SELECT Statement.
- **Multiple Row Subquery:** Queries that return more than one row from the inner SELECT Statement.
- **Multiple Column Subquery:** Queries that return more than one column from the inner SELECT Statement.

SINGLE ROW SUBQUERIES

- Return only one Row

- Use single row comparison operators
  - = Equal to
  - > Greater Than
  - >= Greater Than or Equal
  - < Less Than
  - <= Less than or equal
  - <> Not Equal

SINGLE ROW SUBQUERIES (CONTD.)

- Display the Employees Name whose City is the same as that of Employee_id = ' E00000000004'.

SINGLE ROW SUBQUERIES (CONTD.)

- Display the Employees Name whose City is the same as that of Employee_id=’ E00000000004’.

(SELECT Employee_city From Employee
Where Employee_id = 'E00000000004');

SINGLE ROW SUBQUERIES (CONTD.)

- Display the Employees Name whose City is the same as that of Employee_id=’ E00000000004’.

```
SELECT Employee_name From Employee
  Where Employee_city =
(SELECT Employee_city From Employee
 Where Employee_id = 'E00000000004');
```

SINGLE ROW SUBQUERIES (CONTD.)

- The Outer and the Inner Query can get data from different tables.

Display the Employee's Name whose dob is the same as that of Cust_id=’C00000000010’.

SINGLE ROW SUBQUERIES (CONTD.)

Display the Employees Name whose dob is the same as that of Cust_id=’C0000000010’.

```
SELECT Employee_name
From Employee Where Employee_dob =
(SELECT Cust_dob From Customer
Where Cust_id=’C0000000010’);
```

SINGLE ROW SUBQUERIES (CONTD.)

- There can be more than one inner queries:

- Display the Employee's Name whose City is the same as that of Employee_id=’ E00000000004’ and salary is greater than average salary.

SINGLE ROW SUBQUERIES (CONTD.)

- Display the Employees Name whose City is the same as that of Employee_id=’ E00000000004’ and salary is greater than average salary.

```
SELECT Employee_name
From Employee
Where Employee_city= ( SELECT Employee_city
From Employee
Where Employee_id=’ E00000000004’)
AND  Salary > (SELECT AVG(Salary)
FROM   EMPLOYEE);
```

SINGLE ROW SUBQUERIES (CONTD.)

- Display the Employees Name whose City is the same as that of Employee_id=’ E00000000004’ and salary is greater than average salary.

```
SELECT Employee_name
From Employee
Where Employee_city= ( SELECT Employee_city
From Employee
Where Employee_id=’ E00000000004’)
AND  Salary > (SELECT AVG(Salary)
FROM    EMPLOYEE);
```

You can use GROUP FUNCTION in a subquery to return a single row

Practice:

- Show the name of the employee who has the lowest salary.

HAVING CLAUSE WITH SUBQUERIES

- You can use subqueries also in the HAVING clause

- To display all the employees' city that have a minimum salary greater than that of Employee city 'e_city_001'.

HAVING CLAUSE WITH SUBQUERIES

- To display all the employees' city that have a minimum salary greater than that of Employee city 'e_city_001'.

```
SELECT EMPLOYEE_CITY, MIN(SALARY)
FROM EMPLOYEE
GROUP BY EMPLOYEE_CITY
HAVING MIN(SALARY)>
(SELECT MIN(SALARY) FROM EMPLOYEE
WHERE EMPLOYEE_CITY='e_city_001');
```

```
SELECT Employee_name, EMPLOYEE_CITY
FROM EMPLOYEE
WHERE SALARY= (SELECT MIN(SALARY)
FROM EMPLOYEE
GROUP BY EMPLOYEE_CITY);
```

```
SELECT Employee_name, EMPLOYEE_CITY
FROM EMPLOYEE
WHERE SALARY= (SELECT MIN(SALARY)
FROM EMPLOYEE
GROUP BY EMPLOYEE_CITY);
```

What is Wrong with this statemant?

```
SELECT Employee_name, EMPLOYEE_CITY
FROM EMPLOYEE
WHERE SALARY= (SELECT MIN(SALARY)
FROM EMPLOYEE
GROUP BY EMPLOYEE_CITY);
```

The single row query returns more than one row. The solution is to use IN operator.

MULTIPLE ROW SUBQUERIES (CONTD.)

- Subqueries that return more than one row.
- You have to use multiple row comparison operators.

Operator	Meaning
IN	**Equal to any member in the list (The IN operator returns TRUE if the comparison value is contained in the list)**
ANY	**ANY operator returns TRUE if the comparison value matches any of the values in the list.**
ALL	**ALL operator returns TRUE only if the comparison value matches all the values in the list.**

USING IN OPERATOR IN MULTIPLE-ROW SUBQUERIES

```
SELECT Employee_name, Employee_city
FROM EMPLOYEE
WHERE SALARY IN (1400, 2850, 2000, 1000);
```

USING IN OPERATOR IN MULTIPLE-ROW SUBQUERIES

```
SELECT Employee_name, Employee_city
FROM EMPLOYEE
WHERE Salary IN (SELECT MIN(Salary)
FROM EMPLOYEE
GROUP BY Employee_city);
```

USING ANY OPERATOR IN MULTIPLE-ROW SUBQUERIES

```
SELECT Employee_name, Employee_id,
EMPLOYEE_CITY
FROM EMPLOYEE
WHERE SALARY < ANY (SELECT MAX(SALARY)
FROM EMPLOYEE
GROUP BY EMPLOYEE_CITY);
```

< **ANY** means less than the Maximum
>**ANY** means more than the minimum
=**ANY** is equivalent to IN

USING ANY OPERATOR IN MULTIPLE-ROW SUBQUERIES

Practice

- Find all the employee's Name, ID and Salary who have the salary more than the minimum salary grouped by Employee's city.

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES:

- Find all the employee's Name, ID and Salary who have the salaries more than the Average salaries of all Employees grouped by Employee's city.

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES:

- Find all the employee's Name, ID and Salary who have the salaries more than the Average salaries of all Employees grouped by Employee's city.

(SELECT AVG(SALARY) FROM EMPLOYEE GROUP BY Employee_city);

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES:

- Find all the employee's Name, ID and Salary who have the salaries more than the Average salaries of all Employees grouped by Employee's city.

```
SELECT Employee_name, Salary,
Employee_City
FROM EMPLOYEE
WHERE SALARY > ALL
(SELECT AVG(SALARY) FROM EMPLOYEE
GROUP BY Employee_city);
```

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES

- Create a query to display the Cust_ID and the Cust_Name for all the customers who have the balance more than the average balance.

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES

SELECT AVG(BALANCE) FROM ACCOUNT);

USING ALL OPERATOR IN MULTIPLE ROW SUBQUERIES

```
SELECT Cust_id, Cust_Name,Balance
FROM Customer NATURAL JOIN Depositor
Natural Join ACCOUNT
WHERE BALANCE > ALL
(SELECT AVG(BALANCE)
FROM ACCOUNT
GROUP BY Type);
```

Multiple Column Subquery

```
SELECT Employee_Name
FROM EMPLOYEE
WHERE (SALARY, Employee_city) IN
( SELECT SALARY, Employee_city
FROM EMPLOYEE
WHERE EMPLOYEE_NAME = 'E_H‘ );
```

Practice:

1. Create a query to display the Cust_ID and the Loan_ID for all the Loan Greater than average loan.

2. Display the Employee name and ID of all the employee who report the manager of Employee(E_B)’s manager.

3. Find all the Customer who has balance equal to minimum balance.

4. Display the Manager’s name and the Average Salary of all the employees managed by them.

EXIST AND NON EXIST COMMAND

EXIST AND NON EXIST COMMAND

- The Oracle EXISTS condition is used in combination with a subquery and is considered "to be met" if the subquery returns at least one row.

- **SYNTAX**

  WHERE EXISTS (subquery);

EXIST AND NON EXIST COMMAND

- If the subquery returns at least one record in its result set, the EXISTS clause will evaluate to true and the EXISTS condition will be met.

- If the subquery does not return any records, the EXISTS clause will evaluate to false and the EXISTS condition will not be met.

- Oracle SQL statements that use the Oracle EXISTS condition are very inefficient since the sub-query is RE-RUN for EVERY row in the outer query's table. There are more efficient ways to write most queries, that do not use the EXISTS condition.

EXIST AND NON EXIST COMMAND

```
SELECT * FROM customer
WHERE EXISTS
(SELECT * FROM depositor WHERE
   customer.cust_id =depositor.cust_id);
```

EXIST AND NON EXIST COMMAND

```
SELECT * FROM customers
WHERE EXISTS
(SELECT * FROM depositor WHERE
   customer.cust_id =depositor.cust_id);
```

This Oracle EXISTS condition example will return all records from the *customers* table where there is at least one record in the*order_details* table with the matching *customer_id*.

EXIST AND NON EXIST COMMAND

```
SELECT * FROM Customer
WHERE NOT EXISTS
(SELECT * FROM depositor WHERE
   customer.cust_id =depositor.cust_id)
```

This Oracle EXISTS example will return all records from the *customers* table where there are no records in the *order_details*table for the given customer_id.

EXIST AND NON EXIST COMMAND

- **EXAMPLE - WITH INSERT STATEMENT**

CREATE A NEW TABLE NAMED CUSTOMER1 OF THOSE CUSTOMERS WHO HAVE ONLY ACCOUNTS.

EXIST AND NON EXIST COMMAND

```
INSERT INTO CUSTOMER1 (CUSTOMER_NAME,
  CUST_ID)
SELECT CUST_NAME, CUST_ID FROM CUSTOMER
WHERE EXISTS
(SELECT * FROM DEPOSITOR WHERE
  CUSTOMER.CUST_ID = DEPOSITOR.CUST_ID);
```

EXIST AND NON EXIST COMMAND

- **Self Study:**

- Exist Command with UPDATE Statement
- Exist Command with DELETE Statement

References

1. Oracle_Database_11g_The_Complete Reference
2. http://www.w3schools.com/sql/
3. Book: Database System Concepts written by Avi Silberschatz, Henry F. Korth, S. Sudarshan


Thank You