SQL SUB QUERIES

Sub Queries

A Subquery or Inner query or Nested query is a query within another SQL query, and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved

There are a few rules that subqueries must follow:

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause.
- An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY.
- Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.
- The BETWEEN operator cannot be used with a subquery; however, the BETWEEN can be used within the subquery.

Simple Sub Queries

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows:

```
SELECT column_EID [, column_EID ]
FROM  table1 [, table2 ]
WHERE  column_EID OPERATOR
(SELECT column_EID [, column_EID ]
FROM table1 [, table2 ]
[WHERE])
```

Note: Although Subqueries are commonly used with Select statement, these can also be used with Insert, Update or Delete Statements

Correlated Sub Queries

There are ways to incorporate the outer query's values into the subquery's clauses. These types of queries are called correlated subqueries, since the results from the subquery are connected, in some form, to values in the outer query. Correlated queries are sometimes called synchronized queries.

Eg:

Corelatted sub query for average salary of those delhi employees whose salary is >100000

```
select avg(salary) from emp_sal
where eid in (select eid from emp where city='Delhi') and
eid in (select eid from emp_sal where salary > 100000);
```

SQL EXISTS Operator

- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns true if the subquery returns one or more records.

```
SELECT column_EID(s)
FROM table_EID
WHERE EXISTS
(SELECT column_EID FROM table_EID WHERE condition);
```

ASSIGNMENT

ASSIGNMENT – 7

A-1 : EID, NAME, CITY OF GURGAON EMPLOYEES

A-2 : EID, NAME , DOJ ,DEPT, DESI & SALARY OF ALL MANAGERS

A-3: REDUCE THE SALARY OF ALL DELHI EMPLOYEES BY 10%.

A-4 : DISPLAY THE EID, NAME , CITY, DOJ ,DEPT, DESI & SALARY OF THE TEAM MEMBERS OF DAVID & RAMESH GUPTA.

A-5: CREATE A TRAINING TABLE CONTAINING EID, NAME, DEPT. INSERT THE DETAILS OF OPS TEAM MEMBERS IN THE TRAINING TABLE.

A-6: DETAILS OF DIRECTORS SHOULD BE DELETED FROM THE TRAINING TABLE.

A-7: DISPLAY THE SALARY DETAILS OFF ALL EMPLOYES IF ANY OF THE TEAM MEMBER HAS SALARY MORE THAN 200000.

STORED PROCEDURES

Stored Procedures

- A stored procedure is prepared SQL code that we save so we can reuse the code over and over again. So if we think about a query that we write over and over again, instead of having to write that query each time we would save it as a stored procedure and then just call the stored procedure to execute the SQL code that we saved as part of the stored procedure.
- In addition to running the same SQL code over and over again we also have the ability to pass parameters to the stored procedure.

SYNTAX

```
CREATE PROCEDURE <procedure_EID>
AS
BEGIN
<SQL Statement>
END

EXECUTE <procedure_EID>
EXEC  <procedure_EID>
 <procedure_EID>
```

Stored Procedures

Example 1 : Simple Procedure to get the details of Delhi employees

```
CREATE PROCEDURE SHDELEMP
AS
BEGIN
   SELECT * FROM EMP WHERE CITY = 'DELHI';
END;
```

Stored Procedures

Example 2 : **Parameterized** Procedure to get the details of employees of the specified city

```
CREATE PROCEDURE SHOWEMP @X VARCHAR(20)
AS
BEGIN
   SELECT * FROM EMP WHERE CITY = @X;
END;
```

Example 3 : **Parameterized** Procedure to get the contents of the specified table

```
CREATE PROCEDURE SHOW @Y VARCHAR(20)
AS
BEGIN
        EXEC('SELECT * FROM ' + @Y );
END;
```

Stored Procedures

Example 4 : **Parameterized** Procedure to insert the data in the emp_sal table

```
CREATE PROCEDURE IN_EMP_SAL
@ID VARCHAR(5), @A VARCHAR(20), @B VARCHAR(20), @X INT
AS
BEGIN
        SET NOCOUNT ON ;
        INSERT INTO EMP_SAL VALUES
        ( @ID, @A, @B, @X );

        SELECT * FROM EMP_SAL
        WHERE EID=@ID;
END;
```

Stored Procedures

A stored procedure with parameters:

SYNTAX

```
CREATE PROCEDURE <procedure_EID>
@ var1 datatype (size), var2 datatype (size)
AS
BEGIN
[SET NOCOUNT ON/OFF]
<SQL Statement>
END

EXECUTE <procedure_EID> var1 , var2

DROP PROCEDURE <procedure_EID>
```

ASSIGNMENT

ASSIGNMENT – 8

A-1 : CREATE BELOW PROCEDURES IN THE INVENTORY DATABASE AS SPECIFIED :

ADDSUPPLIER – SHOULD ADD THE SUPPLIER IN THE SUPLIER TABLE AND DISPLAY THE DETAILS OF THE NEW SUPPLIER ADDED.

ADDPRO – SHOULD ADD THE PRODUCT IN THE PRODUCT TABLE AND DISPLAY THE DETAILS OF THE NEW PRODUCT ADDED.

ADDCUST – SHOULD ADD THE CUSTOMER IN THE CUSTOMER TABLE AND DISPLAY THE DETAILS OF THE NEW CUSTOMER ADDED.

ADDORDER – SHOULD ADD THE ORDER IN THE ORDERS TABLE AND DISPLAY THE DETAILS OF THE ORDER. ORDER DATE SHOULD BE CURRENT DATE AND SHOULD COME AUTOMATICALLY.