### Assignment No: 3

**Title:** SQL Queries – all types of Join, Sub-Query and View

**Aim:** Write at least10 SQL queries for suitable database application using SQL DML statements. Prerequisites:

* Basic knowledge about databases.

**Objective:** Student will able to learn commands that make changes in relational database and transaction management.

**Outcome:**

* + Student gains the knowledge to implement subqueries using Joins.

**Theory:**

**SQL Subquery**

Subquery or Inner query or Nested query is a query in a query. SQL subquery is usually added in the [WHERE](http://beginner-sql-tutorial.com/sql-where-clause.htm) Clause of the SQL statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.

Subqueries are an alternate way of returning data from multiple tables.

Subqueries can be used with the following SQL statements along with the comparison operators like =, <, >, >=, <= etc.

* [SELECT](http://beginner-sql-tutorial.com/sql-select-statement.htm)
* [INSERT](http://beginner-sql-tutorial.com/sql-insert-statement.htm)
* [UPDATE](http://beginner-sql-tutorial.com/sql-update-statement.htm)
* [DELETE](http://beginner-sql-tutorial.com/sql-delete-statement.htm)

A query is called correlated subquery when both the inner query and the outer query are interdependent. For every row processed by the inner query, the outer query is processed as well. The inner query depends on the outer query before it can be processed.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SELECT | p.product\_name |  | FROM | product | | p |
| WHERE | p.product\_id = (SELECT | | o.product\_id | FROM | order\_items | o |
| WHERE o.product\_id = p.product\_id); | |  | | | | |

**Nested Subquery**

1. You can nest as many queries you want but it is recommended not to nest more than 16 subqueries in oracle

Non-Corelated Subquery

1. If a subquery is not dependent on the outer query it is called a non-correlated subquery Subquery Errors
2. Minimize subquery errors: Use drag and drop, copy and paste to avoid running subqueries with spelling and database typos. Watch your multiple field SELECT comma use, extra or to few getting SQL error message "Incorrect syntax".

SQL Subquery Comments

Adding SQL Subquery comments are good habit (/\* your command comment \*/) which can save you time, clarify your previous work .. results in less SQL headaches

Subquery Reference

SQL Subquery optimization is a good habit .. more on SQL Query Fine Tuning [SQL Query](http://beginner-sql-tutorial.com/sql-query-tuning.htm) [Optimization](http://beginner-sql-tutorial.com/sql-query-tuning.htm).

**View:**

SQL CREATE VIEW Statement

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

**CREATE VIEW Syntax** CREATE VIEW view\_name AS SELECT column1, column2, .. FROM table\_name

WHERE condition;

**SQL CREATE VIEW Examples**

If you have the Northwind database you can see that it has several views installed by default.

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

CREATE VIEW [Current Product List] AS SELECT ProductID, ProductName

FROM Products

WHERE Discontinued = No;

Then, we can query the view as follows: SELECT \* FROM [Current Product List]; **SQL Dropping a View**

You can delete a view with the DROP VIEW command.

SQL DROP VIEW Syntax

DROP VIEW view\_name;

**SQL Joins**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

**Different Types of SQL JOINs**

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
  + RIGHT (OUTER) JOIN: Return all records from the right table, and the matched recordsfrom the left table
  + FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

**SQL INNER JOIN :**

The INNER JOIN keyword selects records that have matching values in both tables.Syntax:

SELECT *column\_name(s)*

FROM *table1*

INNER JOIN *table2* ON *table1.column\_name* = *table2.column\_name*;

**SQL INNER JOIN e.g.:**

SELECT Orders.OrderID,

Customers.CustomerNameFROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

SQL LEFT JOIN :

The LEFT JOIN keyword returns all records from the left table (table1), and thematched records from the right table (table2). The result is NULL from the right side, if there is no match.

**LEFT JOIN Syntax :**

SELECT *column\_name(s)*

FROM *table1*

LEFT JOIN *table2* ON *table1.column\_name* =

*table2.column\_name*; **Example :**

SELECT Customers.CustomerName, Orders.OrderIDFROM Customers

LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerIDORDER BY Customers.CustomerName;

The LEFT JOIN keyword returns all records from the left table (Customers), even if there are nomatches in the right table (Orders).

**SQL RIGHT JOIN :**

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched recordsfrom the left table (table1). The result is NULL from the left side, when there is no match.

RIGHT JOIN Syntax :

SELECT *column\_name(s)*

FROM *table1*

RIGHT JOIN *table2* ON *table1.column\_name* =

*table2.column\_name*; **Example:**

SELECT Orders.OrderID, Employees.LastName, Employees.FirstNameFROM Orders

RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeIDORDER BY Orders.OrderID;

The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are nomatches in the left table (Orders).

**SQL FULL OUTER JOIN Keyword**

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) orright (table2) table records.

Note: FULL OUTER JOIN can potentially return very large result-sets!FULL OUTER JOIN Syntax

SELECT *column\_name(s)*

FROM *table1*

FULL OUTER JOIN *table2* ON *table1.column\_name* = *table2.column\_name*;

**SQL FULL OUTER JOIN :**

The FULL OUTER JOIN keyword returns all the rows from the left table (Customers), and all the rows from the right table (Orders). If there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

SELECT Customers.CustomerName, Orders.OrderIDFROM Customers FULL OUTER JOIN Orders ON

Customers.CustomerID=Orders.CustomerIDORDER BY Customers.CustomerName;

**SQL Self JOIN:**

A self JOIN is a regular join, but the table is joined with itself. Self JOIN Syntax

SELECT

*column\_name(s)* FROM *table1 T1, table1 T2*WHERE *condition*;

**Example:**

The following SQL statement matches customers that are from the same city:

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.CityFROM Customers A, Customers B

WHERE A.CustomerID <> B.CustomerIDAND A.City = B.City

ORDER BY A.City;

Conclusion: Thus we have successfully implemented all joins, subquery and view.

Questions:

* 1. Explain different Joins.
  2. Explain concept of view.
  3. What is Inner Join?
  4. Explain nested query.