**SYEDA FATIMA SHAH**

**29th March 2023**

***Task # 5:***

**Practice following in detail:**

**1 - Sub Queries:**

**• Single Row Sub Queries**

A single-row subquery is a type of SQL subquery that returns only one row of data, usually used in a comparison operator to compare against a value in the outer query.

**Query:**

SELECT \* FROM Orders

SELECT \* FROM [Order Details]

SELECT OrderID, (SELECT MAX(UnitPrice) FROM [Order Details] WHERE OrderID = OrderID) AS MaxUnitPrice FROM Orders;

SELECT ProductName, UnitsInStock,UnitPrice FROM Products

WHERE UnitPrice > (SELECT min(UnitPrice) FROM Products);

**• Multi Row Sub Queries**

A Multi-row subquery is a type of SQL subquery that returns multiple rows of data usually used in a comparison operator to compare against a set of values in the outer query.

**Query:**

SELECT OrderID, CustomerID FROM Orders

WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE Country = 'UK')

SELECT \* FROM Employees

SELECT \* FROM Orders

SELECT e.EmployeeID,e.FirstName,e.Title FROM Employees e Where e.EmployeeID IN(SELECT e.EmployeeID FROM Employees e WHERE e.Extension<500);

**• Nested Sub Queries**

This type of subquery is a subquery that is nested inside another subquery. It can be used to retrieve data from multiple tables and use that data in another query.

**Query:**

SELECT \* FROM Orders WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE Country IN (SELECT Country FROM Suppliers

WHERE SupplierID IN (SELECT SupplierID FROM Products WHERE Discontinued = 0)));

SELECT \* FROM Employees WHERE EmployeeID IN (select EmployeeID FROM Orders where OrderID IN(select OrderID FROM [Order Details]

WHERE UnitPrice>10));

**• Co-Related Sub Queries**

This type of subquery is a subquery that references a column from the outer query. It can be used to retrieve data from multiple tables and use that data in another query.

**Query:**

SELECT o.CustomerID, o.OrderDate, o.ShipCity FROM orders o

WHERE o.Freight > ( SELECT AVG(o.Freight) FROM Orders o)

[**https://youtu.be/m1KcNV-Zhmc**](https://youtu.be/m1KcNV-Zhmc)

**2 - Views:**

**• Purpose Of Views**

The purpose of views is to provide a customized and simplified view of the data for the end-users, without them needing to know the underlying structure of the data. These are also known as virtual tables.

* ***Simplify Queries:*** Views can hide complex queries or calculations, making it easier for users to query the data they need.
* ***Security***: Views can be used to restrict users from accessing certain columns or rows in a table.
* ***Data Abstraction:*** Views can present the data in a different way than the underlying tables, providing a simpler and more logical representation of the data.
* ***Performance***: Views can improve performance by pre-aggregating data or filtering rows, reducing the amount of processing required for certain queries.

**• Creating, Altering and Dropping Views**

--Creating View

CREATE VIEW ProductsView

AS SELECT ProductID, ProductName, UnitPrice FROM Products;

SELECT \* FROM ProductsView

-- Alter View

ALTER VIEW ProductsView As SELECT ProductName FROM Products;

SELECT \* FROM ProductsView

-- Drop View

Drop View ProductsView

**• Simple and Complex Views**

--Simple View

CREATE VIEW EmployeView

As SELECT e.EmployeeID,e.FirstName from Employees e where e.Country='UK'

SELECT \* from EmployeView

-- Complex View

CREATE VIEW OrderViews

As SELECT \* FROM Orders WHERE CustomerID IN

(SELECT CustomerID FROM Customers WHERE Country IN

(SELECT Country FROM Suppliers WHERE SupplierID IN

(SELECT SupplierID FROM Products WHERE Discontinued = 0)))

SELECT \* FROM OrderView

[**https://youtu.be/cLSxasHg9WY**](https://youtu.be/cLSxasHg9WY)

**3 - Indexes**

**• Clustered Index**

**• NonClustered Index**

**• Create, Alter and Drop Indexes**

**• Using Indexes**

CREATE TABLE Carr( CarName varchar(50),

PltNO int Primary key not null, -- i-Identity using Unique attribute in my table

color varchar(50))

Insert into Carr(CarName,PltNO,color) values('merce',7655, 'black'),('Honda', 3453,'White'),('Audi',3456,'Grey')

select \* from Carr

CREATE INDEX IndexCar

ON Carr (pltNO);

-- To create clusteer index we have to delete primary key of our table because clustered are usually made on primary keys by default

execute sp\_helpindex Carr

CREATE clustered INDEX IndexCar ON Carr (pltNO);

alter INDEX IndexCar ON Carr REBUILD

drop clustered index IndexCar

CREATE nonclustered INDEX IndexCar ON Carr (pltNO);

alter INDEX IndexCar ON Carr REBUILD

[**https://youtu.be/NZgfYbAmge8**](https://youtu.be/NZgfYbAmge8)

**At least try two examples of all.**

***Task # 6:***

**Write the following queries using Northwind DB:**

**1 - Write a query to retrieve the names of all products that have been ordered more than 50 time.**

SELECT ProductName from Products p INNER JOIN [Order Details] od ON p.ProductID = od.ProductID

GROUP BY p.ProductName having SUM(Quantity) > 50

**2 - Write a query to retrieve the names of all products that have been ordered at least once.**

select \* from [Order Details]

select \* from Orders

select ProductName FROM Products p INNER JOIN [Order Details] od ON p.ProductID = od.ProductID

**3 - Create a view that shows the total revenue generated by each category.**

CREATE VIEW CategoryRevenue

AS

Select c.CategoryName, SUM(od.UnitPrice \* Od.Quantity \* (1 - Od.Discount)) AS TotalRevenue

From Categories c INNER JOIN Products p ON c.CategoryID = p.CategoryID INNER JOIN [Order Details] od ON p.ProductID = Od.ProductID

GROUP BY c.CategoryName

SELECT \* FROM CategoryRevenue