BYTEWISE FELLOWSHIP PROGRAM

DATA ENGINEERING

Tehreem Fatima

**WEEK 3**

**27– March -2023**

**Task # 1**

Practice following DQL commands in SSMS:

1 - Data Query Language (DQL):

* ***Select***

SELECT CustomerID, CompanyName, ContactName, City

FROM Customers;

* ***Where Clause***

SELECT \*

FROM Orders

WHERE CustomerID = 'ALFKI';

* ***Order by Clause***

SELECT \*

FROM Employees

ORDER BY BirthDate DESC, FirstName ASC;

* ***Distinct Keyword***

SELECT DISTINCT City, Country

FROM Customers;

* ***IsNull Function***

SELECT \*

FROM Customers

WHERE ContactName IS NULL;

* ***Column Aliases***

SELECT ProductName AS 'Product', UnitPrice AS 'Price'

FROM Products;

***Predicates***

* ***Between ... And***

SELECT \*

FROM Orders

WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

* ***In***

SELECT \*

FROM Customers

WHERE Country IN ('USA', 'Canada', 'Mexico');

* ***Like***

SELECT \*

FROM Employees

WHERE LastName LIKE 'D%';

* ***Is Null***

SELECT \*

FROM Customers

WHERE ContactName IS NULL;

* ***Top n Clause***

SELECT TOP 10 \*

FROM Customers;

***Set Operators***

* ***Union***

SELECT City, Country

FROM Customers

WHERE Country = 'USA'

UNION

SELECT City, Country

FROM Customers

WHERE Country = 'Canada';

* ***Intersect***

SELECT City, Country

FROM Customers

WHERE Country = 'USA'

INTERSECT

SELECT City, Country

FROM Orders

WHERE OrderDate BETWEEN '1997-01-01' AND '1997-12-31';

* ***Except***

SELECT City, Country

FROM Customers

WHERE Country = 'USA'

EXCEPT

SELECT City, Country

FROM Orders

WHERE OrderDate BETWEEN '1997-01-01' AND '1997-12-31';

**Task # 2**

Write the following queries using Northwind DB:

***1 - Retrieve the product name, unit price, and units in stock of all products that cost less than $20, ordered by product name.***

SELECT ProductName, UnitPrice, UnitsInStock

FROM Products

WHERE UnitPrice < 20

ORDER BY ProductName;

***2 - Retrieve the order ID, customer ID, and order date of all orders that do not have a customer ID, ordered by the order date.***

SELECT OrderID, CustomerID, OrderDate

FROM Orders

WHERE CustomerID IS NULL

ORDER BY OrderDate;

***3 - Retrieve the category names of all products, as well as the countries to which orders have been shipped, without any duplicates.***

SELECT DISTINCT Categories.CategoryName, ShipCountry

FROM Products

JOIN Categories ON Products.CategoryID = Categories.CategoryID

JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID

JOIN Orders ON OrderDetails.OrderID = Orders.OrderID

GROUP BY Categories.CategoryName, ShipCountry;

**Task # 3**

Practice the following in detail:

***1 - Joins:***

* ***Inner Join***

Inner Join Returns only the matching rows between two tables.

SELECT Orders.OrderID, Customers.CustomerName

FROM Orders

INNER JOIN Customers

ON Orders.CustomerID = Customers.CustomerID;

* ***Outer Join***

Outer Join Returns all the rows from one table and matching rows from another table. If there is no matching row in the other table, it returns null values.

SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

LEFT OUTER JOIN Orders

ON Customers.CustomerID = Orders.CustomerID;

* ***Left Join***

Left Join Returns all the rows from the left table and matching rows from the right table. If there is no matching row in the right table, it returns null values.

SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

LEFT JOIN Orders

ON Customers.CustomerID = Orders.CustomerID;

* ***Right Join***

Right Join Returns all the rows from the right table and matching rows from the left table. If there is no matching row in the left table, it returns null values.

SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

RIGHT JOIN Orders

ON Customers.CustomerID = Orders.CustomerID;

* ***Self Join***

Self Join Joins a table to itself to compare data in different rows.

SELECT e.EmployeeID, e.FirstName, m.FirstName AS ManagerName

FROM Employees e

INNER JOIN Employees m

ON e.ReportsTo = m.EmployeeID;

* ***Cross Join***

Cross Join: Returns the Cartesian product of two tables - all possible combinations of rows.

SELECT \*

FROM Customers

CROSS JOIN Orders;

* ***Left anti-Join***

Left Anti-Join Returns only the rows from the left table that do not have a match in the right table.

SELECT Customers.CustomerName

FROM Customers

LEFT JOIN Orders

ON Customers.CustomerID = Orders.CustomerID

WHERE Orders.OrderID IS NULL;

* ***Right anti-Join***

Right Anti-Join Returns only the rows from the right table that do not have a match in the left table.

SELECT Orders.OrderID

FROM Orders

LEFT JOIN Customers

ON Orders.CustomerID = Customers.CustomerID

WHERE Customers.CustomerID IS NULL;

***Task # 4***

Write the following queries using Northwind DB:

***1 - Write a query to show a list of customers and their corresponding orders.***

SELECT

c.CustomerID, c.CompanyName, o.OrderID, o.OrderDate

FROM

Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID;

***2 - Write a query to show a list of employees and their corresponding managers:***

SELECT

e.EmployeeID, e.FirstName + ' ' + e.LastName AS EmployeeName,

m.EmployeeID AS ManagerID, m.FirstName + ' ' + m.LastName AS ManagerName

FROM

Employees e

LEFT JOIN Employees m ON e.ReportsTo = m.EmployeeID;

***3 - Write a query to show a list of customers who have not placed any orders.***

SELECT

c.CustomerID, c.CompanyName

FROM

Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE

o.OrderID IS NULL;

***4 - Write a query to show a list of employees and their corresponding territories, including those employees who have no territories assigned.***

SELECT

e.EmployeeID, e.FirstName + ' ' + e.LastName AS EmployeeName,

t.TerritoryID, t.TerritoryDescription

FROM

Employees e

LEFT JOIN EmployeeTerritories et ON e.EmployeeID = et.EmployeeID

LEFT JOIN Territories t ON et.TerritoryID = t.TerritoryID;

**Task # 5**

Practice the following in detail:

***1 - Sub Queries:***

* ***Single Row Sub Queries***

SELECT product\_name

FROM products

WHERE product\_id = (

SELECT product\_id

FROM order\_details

WHERE order\_id = 10248

);

* ***Multi-Row Sub Queries***

SELECT customer\_name

FROM customers

WHERE city IN (

SELECT city

FROM suppliers

);

* ***Nested Sub Queries***

SELECT customer\_name

FROM customers

WHERE country IN (

SELECT country

FROM (

SELECT DISTINCT country

FROM suppliers

) AS supplier\_countries

);

* ***Co-Related Sub Queries***

SELECT employee\_name

FROM employees e1

WHERE salary > (

SELECT AVG(salary)

FROM employees e2

WHERE e1.department\_id = e2.department\_id

);

***2 - Views:***

***Purpose of Views***

* ***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*** ***Views***

CREATE VIEW employee\_vw AS

SELECT employee\_id, first\_name, last\_name, hire\_date

FROM employees;

* ***Altering Views***

ALTER VIEW employee\_vw AS

SELECT employee\_id, first\_name, last\_name, hire\_date, department\_id

FROM employees

WHERE department\_id = 1;

* ***Dropping Views***

DROP VIEW employee\_vw;

Simple and Complex Views

* ***Creating a Simple View:***

CREATE VIEW employee\_vw AS

SELECT employee\_id, first\_name, last\_name, hire\_date

FROM employees;

* ***Creating a Complex View:***

CREATE VIEW customer\_orders\_vw AS

SELECT c.customer\_id, c.company\_name, o.order\_id, o.order\_date, od.product\_id, od.quantity

FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

LEFT JOIN order\_details od ON o.order\_id = od.order\_id;

***3 - Indexes***

* ***Clustered Indexes***

CREATE CLUSTERED INDEX idx\_orders\_orderdate

ON orders(orderdate);

* ***NonClustered Indexes***

CREATE NONCLUSTERED INDEX idx\_customers\_contactname

ON customers(contactname);

* ***Creating Indexes***

CREATE CLUSTERED INDEX idx\_orders\_orderdate

ON orders(orderdate);

* ***Altering Indexes***

ALTER INDEX idx\_orders\_orderdate

ON orders REBUILD;

* ***Droping Indexes***

DROP INDEX idx\_customers\_contactname

ON customers;

* ***Using Indexes***

SELECT \*

FROM orders

WHERE orderdate >= '2022-01-01'

AND orderdate < '2023-01-01'

AND customerid = 'ALFKI'

AND employeeid = 4;

**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 times.***

SELECT p.product\_name

FROM products p

JOIN order\_details od ON p.product\_id = od.product\_id

GROUP BY p.product\_name

HAVING COUNT(\*) > 50;

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

SELECT DISTINCT p.product\_name

FROM products p

JOIN order\_details od ON p.product\_id = od.product\_id;

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

CREATE VIEW category\_revenue AS

SELECT c.category\_name, SUM(od.unit\_price \* od.quantity) AS revenue

FROM categories c

JOIN products p ON c.category\_id = p.category\_id

JOIN order\_details od ON p.product\_id = od.product\_id

GROUP BY c.category\_name;

***Task # 7***

Practice following:

***1 - Conditional Control Statements:***

* ***Case***

SELECT ProductName, UnitPrice,

CASE

WHEN UnitPrice < 10 THEN 'Cheap'

WHEN UnitPrice >= 10 AND UnitPrice < 50 THEN 'Moderate'

ELSE 'Expensive'

END AS PriceCategory

FROM Products

ORDER BY UnitPrice;

* ***If***

DECLARE @quantity INT = 10

IF @quantity > 0

SELECT 'There are still products in stock'

ELSE

SELECT 'Out of stock'

Write the following queries using Northwind DB:

***1 - Write a query to retrieve the product name, unit price, and discount for all orders in the "Order Details" table, and apply a discount of 10% if the unit price is greater than or equal to 50.***

SELECT ProductName, UnitPrice,

CASE

WHEN UnitPrice >= 50 THEN UnitPrice \* 0.9

ELSE UnitPrice

END AS DiscountedPrice

FROM Products

***2 - Write a query to retrieve the product name, unit price, and supplier name for all products in the "Products" table, and apply a case statement to display "Not available" if the supplier name is null.***

SELECT ProductName, UnitPrice,

CASE

WHEN SupplierName IS NULL THEN 'Not available’

ELSE SupplierName

END AS SupplierName

FROM Products